White Rose Research Online

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk
Universities of Leeds, Sheffield and York

Deposited via The University of York.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/id/eprint/245709/

Version: Accepted Version

Article:
Dudson, Ben, Kryjak, Mike, Muhammed, Hasan et al. (2024) Hermes-3: Multi-component plasma simulations with BOUT++. Computer Physics Communications. 108991. ISSN: 0010-4655

https://doi.org/10.1016/j.cpc.2023.108991

Reuse
This article is distributed under the terms of the Creative Commons Attribution (CC BY) licence. This licence allows you to distribute, remix, tweak, and build upon the work, even commercially, as long as you credit the authors for the original work. More information and the full terms of the licence here: https://creativecommons.org/licenses/

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

UNIVERSITY OF LEEDS
University of Sheffield
UNIVERSITY of York

Hermes-3: Multi-component plasma simulations with BOUT++

Ben Dudson[a], Mike Kryjak[b], Hasan Muhammed[b,c], Peter Hill[b], John Omotani[c]

[a] *Lawrence Livermore National Laboratory, 7000 East Avenue, Livermore, 94550, CA, USA*
[b] *School of Physics, Engineering and Technology, University of York, Heslington, York, YO10 5DD, UK*
[c] *United Kingdom Atomic Energy Authority, Culham Centre for Fusion Energy, Culham Science Centre, Abingdon, OX14 3DB, UK*

Abstract

A new open source tool for fluid simulation of multi-component plasmas is presented, based on a flexible software design that is applicable to scientific simulations in a wide range of fields. Hermes-3 is built on plasma simulation framework BOUT++, consolidating earlier SD1D and Hermes models into a single code that can be configured at run-time to solve plasma models in 1D, 2D or 3D, either for transport (steady-state) or turbulent (time-evolving) problems, with an arbitrary number of ion and neutral species. We describe the improved numerical algorithms and software design that have been implemented in Hermes-3.

To demonstrate the capabilities of this tool, applications relevant to the boundary of tokamak plasmas are presented: 1D simulations of diveror plasmas evolving equations for all charge states of neon and deuterium; 2D transport simulations of tokamak equilibria in single-null X-point geometry with plasma ion and neutral atom species; and simulations of the time-dependent propagation of plasma filaments (blobs).

Hermes-3 is publicly available on Github under the GPL-3 open source license. The repository includes documentation and a suite of unit, integrated and convergence tests.

Keywords:
plasma, simulation, tokamak, BOUT++

PROGRAM SUMMARY

Program Title: Hermes-3
CPC Library link to program files:
Developer's repository link: https://github.com/bendudson/hermes-3
Code Ocean capsule:
Licensing provisions: GPLv3
Programming language: C++14
Nature of problem: Simulation of dynamics and steady-state solutions of multiple ion and neutral species in magnetised plasmas using drift-reduced fluid plasma models in 1 to 3 spatial dimensions. Focused on but not restricted to the simulation of turbulence in the boundary of tokamaks.
Solution method: Hermes-3 implements a system of flexible software components, that are configured at run-time and coupled by passing a state object between them (the Encapsulate State and Command patterns). Hermes-3 builds on BOUT++ [1], employing the Method of Lines with implicit and explicit time-integration methods in curvilinear coordinates on block-structured meshes, making use of libraries including SUNDIALS, PETSc and Hypre. Hermes-3 implements drift-reduced plasma fluid equations using conservative finite difference methods, and atomic processes that couple plasma and neutral fluids.

1. Bibliography

[1] B.D.Dudson et al. CPC 180, 1467-1480 (2009)

2. Introduction

An important feature of magnetically confined fusion plasmas in tokamaks is that they consist of a mixture of different ion species. Computational studies of fusion plasma phenomena, such as plasma turbulence, have to date frequently treated plasmas as if they consisted of a single ion species, typically deuterium due to its use in experiments. In order to make predictions of phenomena important to the performance and design of the divertor of future fusion reactors, models must capture the interactions between plasma and neutral gas species (deuterium and tritium in a reactor), as well as radiation from impurity species (eg. Be, C, Ne, Ar, W), and pumping of helium "ash" and other species. In many cases none of these species can be treated as "trace" but are all coupled, so that all should be considered self-consistently in the same simulation. For each of these atomic species the density and dynamics of multiple states may need to be considered, for example the

ground state, multiple ionisation stages, and molecular species. In some cases it may also be necessary to track metastable states, such as vibrationally excited states or charged molecules. The result is a potentially large number of species types which must be solved for, together with a complex set of reactions between them.

Historically there has been a division in terms of simulation tools and to an extent also research communities, between multi-fluid transport codes such as SOLPS [1], UEDGE [2], EDGE2D [3] and BOUT++/trans-neut [4], which employ simplified models for the cross-field transport (typically diffusive) but evolve many different species, and the turbulence codes including GBS [5, 6, 7], TOKAM3X [8, 9], (H)ESEL [10], and models built on BOUT++ [11, 12, 13] such as Hermes [14, 15] and STORM [16, 17]. These latter models can solve for the 3D time-varying turbulent transport, but typically only evolved a single ion species. During the last 5 years, and currently ongoing, are several efforts (e.g. [18, 19, 20, 21, 22]) to develop codes which can solve for the turbulent transport self consistently with multiple ion and neutral gas species. Hermes-3 is being developed to address similar research aims as the GBS [5], STORM [16, 23], GRILLIX [24], and SOLEDGE3X [25] codes. Each employs different numerical methods, plasma and neutral gas model formulations. Multi-code comparisons [26, 27, 28] are an important way to understand the advantages and disadvantages of each approach, and to drive improvements to all codes.

Hermes-3 is built on BOUT++, a C++ library that, together with its ecosystem of pre- and post-processing routines [29], provides a platform for developing plasma simulations using finite-difference methods. Early applications were essentially gradient-driven [30], and have evolved [31, 32] into flux-driven models that allow profile variation [33]. In parallel efforts STORM [16, 23] and other flux-driven models [34] have been developed that do not distinguish between background and fluctuating quantities, including zonal flows. All of the above employ finite-difference methods that only conserve total mass or energy in the limit of high resolution (assuming that the analytic model conserves these quantities). Particle balance is a particular challenge when recycling fluxes are large compared to net sources and sinks, as has been noted in other finite difference codes [35] and transport models with monte-carlo neutrals [36]. Hermes [14] and SD1D [37]), from which Hermes-3 developed, are distinct from other BOUT++ models in using conservative finite-difference methods formulated in terms of fluxes through cell faces. As we show in section 5.2, the result is good conservation properties

in the presence of high recycling fluxes.

Hermes-3 is also distinct from previous BOUT++ models in its software implementation: BOUT++ enables spatial operators to be reused, but does not facilitate the reuse of larger model components such as neutral gas models, electric field evolution or sheath boundary conditions. The result is that as models have become more complex, including hot ions and atomic species in Hermes-2 [38] and recent versions of STORM [17], code was added for each new equation. The resulting software was increasingly error prone, difficult to test, and hard to maintain. Hermes-3 addresses these issues by implementing a model component system that splits large models into smaller, testable pieces. By doing this we unify 1D tokamak divertor simulation code SD1D with 2D and 3D transport and turbulence code Hermes, and enable these tools to be extended to multiple ion species simulations that self-consistently include both plasma turbulence and transport (atomic) physics. As a result, SD1D and earlier Hermes codes are no longer developed, with effort focussing on Hermes-3.

In section 3 the numerical methods are described; in section 4 the software architecture is outlined and compared to prior work; and in section 5 a series of applications are presented, which demonstrate some of the capabilities of the new code. We conclude in section 6.

3. Numerical methods

In this section we describe the numerical methods used in Hermes-3 components. The architecture described in section 4.2 does not enforce the choice of numerical method, but this section describes the methods that have been implemented to date and are used in section 5. The system of PDEs is solved using the method of lines, in which the time and spatial dimensions are treated separately: The time integrator simply integrates a set of Ordinary Differential Equations (ODEs), and is discussed in section 3.1; The spatial discretisation is by finite difference methods (section 3.2). Boundary conditions are discussed in section 3.3.

3.1. Time integration

Hermes-3 is built on the BOUT++ framework [13], and so can make use of a range of explicit (e.g RK4), fully implicit (e.g. BDF via SUNDIALS [39]), and implicit-explicit (e.g IMEX-BDF2 and ARKODE via SUNDIALS) methods. These were implemented with the aim of studying time-dependent prob-

lems, such as the study of Edge Localised Mode (ELM) eruptions [30] or tokamak edge turbulence [40], where accurate time evolution is required.

Many of the problems of interest for Hermes-3 are steady state: Axisymmetric tokamak transport solutions, potentially as a starting point for 3D time-dependent turbulence simulations. To find these steady-state solutions efficiently it is desirable to take large timesteps, ideally an infinite timestep, damping transient oscillations in the system. A dissipative time integration scheme that is unconditionally stable is therefore desirable. A first order backward Euler method and preconditioning algorithm similar to that used in UEDGE [2] has been implemented in BOUT++ and its first applications are reported here. This method is stable for any timestep provided that the nonlinear solver, typically a variety of Newton's method, iterations converge. In practice this limits the timestep to a finite, and sometimes quite small, value.

An important ingredient to robustly and efficiently solving the nonlinear problem at each time step is a good preconditioner: It enables the linear inner solve to converge with fewer iterations (and so computational cost) for larger timesteps than would otherwise be possible. Custom preconditioners developed using a methodology such as physics-based preconditioning [41, 42] can be highly effective, but are challenging in multi-fluid contexts: The tokamak edge is a highly nonlinear system, with a potentially large number of species (and so equations), coupled through atomic rates which are typically tabulated rather than analytic, and which vary by orders of magnitude over relatively small temperature ranges. The approach used in UEDGE, and adopted for the Backward Euler solver here, is to use finite differences to calculate the elements of the system Jacobian $\mathbb{J}_{ij} = \frac{\partial}{\partial f_j}\frac{\partial f_i}{\partial t}$ where $\mathbf{f}$ is the vector of evolving quantites. This approximate Jacobian is used to construct a preconditioner that (approximately) solves the linear operator $\mathbb{I} - \Delta t\mathbb{J}$ for timestep Δt. Methods based on incomplete LU factorization (ILU) have been found to be effective: In serial the PETSc ILU(k) solver is used, and in parallel the Euler library [43] within *hypre* [44], both with default settings.

The calculation of a dense Jacobian using finite differences would be prohibitively slow in most cases: A typical simulation might contain $N \simeq 10^5 - 10^8$ evolving quantities, while the Jacobian has N^2 elements. Fortunately the Jacobian is typically sparse, because the finite differences and other interaction terms are local. This is exploited by using the PETSc coloring facilities [45], which are provided with the matrix structure (determined

by the finite difference stencil), and efficiently calculate many Jacobian matrix entries simultaneously. Because each cell only has a fixed number of neighbours, the cost of evaluating the Jacobian is reduced from scaling like N^2 to scaling approximately linearly with the number of grid cells N.

The effectiveness of this time integrator for steady state problems will be applied to 1D transport problems in section 5.1 (fig 5), and for 2D transport in an axisymmetric tokamak geometry in section 5.2.

3.2. Finite differencing spatial operators

The models to be shown here use conservative finite difference operators that were implemented in the Hermes code [14] and have been improved over time and moved into the BOUT++ library. All quantities are cell centred, and advection operators are written in terms of fluxes between cells calculated at cell faces. The cross-field operators presently assume that the grid is orthogonal in the tokamak poloidal plane. This limits the accuracy with which strongly shaped divertor geometries can be simulated with the present code. Non-orthogonal grids which align with wall surfaces can be generated for Hermes using the BOUT++ grid generator [46], but the required off-diagonal metric terms have not yet been implemented. Those terms have long been implemented in UEDGE, and were recently added to SOLPS [47], where they were found to be essential for fluid neutral modelling on distorted grids, but relatively unimportant when kinetic neutrals were used. Implementing these terms is a high priority for future improvements to Hermes-3.

Because all quantities are cell centred, in the absence of dissipation zig-zag modes are likely to develop. In [14] an Added Dissipation [48] artificial dissipation term was used in advection operators along the magnetic field. Here this is replaced with an HLL type flux splitting method [49, 50], that improves on methods used previously for 1D tokamak divertor simulations [37, 51]. Results shown here use either the MinMod or Monotonised Central (MC) slope limiters [52], that have been found to provide a good balance of numerical dissipation and stability. Details of the method are given in Appendix A.

To verify the new implementation of fluid flow along the magnetic field for smooth solutions in Hermes-3, a set of 1D fluid equations along a magnetic field given in equation 1 is tested using the Method of Manufactured solutions (MMS). This testing method has become widely used to verify the correct implementation of complex sets of equations, in tokamak edge plasma codes [53] including BOUT++ [54]. Extended versions of these equations

have been compared to the 2-point model and used for 1D divertor simulations (e.g. [37]).

$$\frac{\partial n}{\partial t} = -\nabla \cdot \left(n \mathbf{b} v_{||} \right) \tag{1a}$$

$$\frac{\partial p}{\partial t} = -\nabla \cdot \left(p \mathbf{b} v_{||} \right) - \frac{2}{3} p \nabla \cdot \left(\mathbf{b} v_{||} \right) \tag{1b}$$

$$\frac{\partial}{\partial t} \left(m n v_{||} \right) = -\nabla \cdot \left(m n v_{||} \mathbf{b} v_{||} \right) - \partial_{||} p \tag{1c}$$

Error norms as a function of mesh cell spacing are presented in figure 1, showing convergence towards the manufactured solution on a 1D periodic domain. Second order convergence is found for both l_2 and l_∞ error norms, consistent with the expected order of accuracy of the numerical methods.

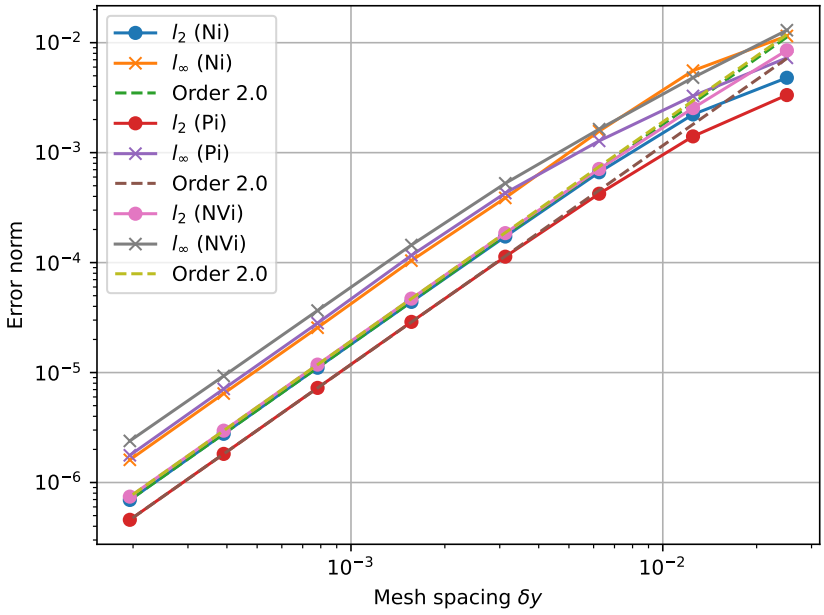


Figure 1: Verification of the convergence of a 1D system of fluid equations on a periodic domain. Showing l_2 (Root-Mean-Square) and l_∞ (Max) errors for the evolving density N_i, pressure P_i and momentum NV_i. Figure produced by `tests/integrated/1D-fluid` in the Hermes-3 repository [55].

The intended application of Hermes-3 is to magnetically confined fusion

plasmas, in which flows are typically subsonic. Nevertheless the code must be robust to transients, and transitions to supersonic flow can occur in tokamak plasmas [56]. Figure 2 shows the results of the standard 1D Sod shock tube test case [57]. In the expanded view shown inset in the left figure 2 it can

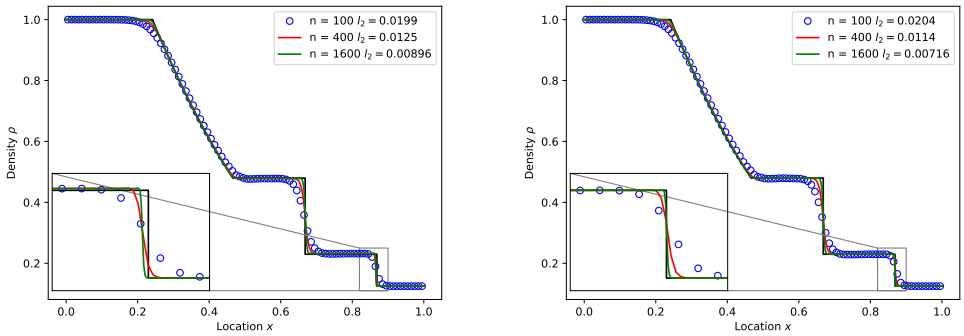


Figure 2: Standard Sod shock tube problem [57] at $t = 0.2$, with exact (analytic) solution in solid black. A solution with reference resolution ($n = 100$ cells) is compared to higher resolutions and their l_2 (Root-Mean-Square) errors. **Left**: Solving the pressure form of the fluid equations. **Right**: Solving the energy form. The inset figures show the shock front in more detail. Figure produced by `tests/integrated/sod-shock` and `tests/integrated/sod-shock-energy`.

be seen that the numerical shock location lags the exact solution, so that the l_2 error norm does not converge to zero. The result is insensitive to time integration method, being observed with both the default CVODE time integrator and the RK3-SSP method implemented in BOUT++. This is due to the choice of pressure as an evolving quantity in equation 1, and so pressure fluxes are calculated at cell edges; Separate limiting of pressure fluxes and momentum fluxes doesn't in general guarantee correct total energy fluxes but can introduce numerical dissipation. Details of how fluxes are calculated are given in appendix Appendix A.

Hermes-3 is structured in a modular way that allows equations to be changed in the input file. If rather than evolving pressure we instead evolve energy density $\mathcal{E} = \frac{3}{2}p + \frac{1}{2}mnv_{||}^2$, so that fluxes of total energy through cell faces are calculated rather than fluxes of pressure (internal energy), then the result is shown on the right of figure 2. Solving the fluid equations in this form recovers the correct shock front location (shown inset in figure 2 right). Further variations on this 1D shock tube problem [58] are included in the Hermes-3 test suite.

We conclude that the methods currently implemented are 2^{nd}-order accurate for smooth solutions (figure 1), and remain robust around shocks, contact discontinuities and expansion waves (figure 2). The modular nature of Hermes-3 allows multiple fluid formulations to be implemented and inter-operate to suit different applications.

3.3. Boundary conditions

The domain typically solved for in 2D and 3D Hermes-3 tokamak simulations is an annulus consisting of a region of closed and open magnetic flux surfaces. An example is discussed in section 5.2 and shown in figure 8. The hot "core" of the plasma is not modelled because the fluid equations solved become invalid in that region. Instead a boundary condition must be imposed at that innermost surface where no boundary physically exists. At the outer edge of the domain the grid is typically close to, but not aligned with, the solid vacuum vessel of the tokamak. Boundary conditions for the thermodynamic variables on both "core" and "wall" boundaries are typically set to either Dirichlet or Neumann.

The boundary condition on the potential ϕ is a variation on the method used in the STORM model [17, 16]: A time-evolving boundary condition that relaxes towards a Neumann boundary. This is implemented in the following way: When inverting the Laplacian-type equation for ϕ from vorticity, the potential is fixed at both core and wall boundaries. If a simple Dirichlet condition is used then narrow boundary layers typically form close to the boundaries in which the imposed boundary potential is matched to the plasma potential. These boundary layers can develop unphysical instabilities. Instead, at every timestep the value of the boundary condition is adjusted towards the value inside the domain with a characteristic timescale that is set by default to $1\mu s$. In this manner the electrostatic potential ϕ evolves smoothly to solutions that can have different potentials on core and wall boundaries.

4. Software architecture

Hermes-3 [55, 59] aims to support a wide range of different models, with an arbitrary number of species and equations. This flexibility presents a challenge for the software design: As previous BOUT++ models were extended it was found that progress became increasingly difficult, motivating the redesign that is presented here.

Existing plasma simulation codes that support multiple species, both transport [1, 2, 3] and turbulence [25, 18, 20] usually exploit the similarity of the equations for each species to enable the number and type of species to be varied at run-time. This is an attractive option but may become complex as special cases are introduced. Here we adopt a more general design that does not assume similar equations for each species, but allows models to be customised to particular applications.

There are many domains besides tokamak plasma physics where performance is important, and where many different software components have to interact in complex ways which need to be extended over time as the software grows and is applied to new problems. In section 4.1 we briefly describe some of the approaches that have been developed within scientific computing and in other fields and that influenced the design of Hermes-3.

4.1. A brief survey of approaches

Within physical sciences there are a number of codes which have adopted designs that enable users and developers to develop components or plugins, and to combine them in novel ways so that the ecosystem becomes increasingly useful as new components are added. An example is LAMMPS [60], which uses a system of "styles" that define interfaces which users can implement to modify the simulation behaviour.

In the software industry Entity Component Systems (ECS) are a design pattern which is commonly used in game development. Those are intended to describe a set of "Entities" that have defined sets of behaviour and can interact with each other. This design pattern offers flexibility through composition rather than inheritance, and considerable run-time configurability. A widely used and high-performance implementation of an ECS is EnTT [61].

Task graphs are another widely applicable and powerful approach to thinking about computations, which focuses on managing the dependencies between components, so that at a high level the whole calculation is a directed acyclic graph (DAG). Examples of task-based systems include StarPU [62] and TaskFlow[63]. The Hermes-3 implementation can be viewed as a task graph with some restrictions on its structure to aid maintainability.

An important aspect of all of these approaches is splitting complex models into simpler components, which interact through standardised interfaces and not global state. This facilitates testing, in particular unit testing, provides a powerful way to mitigate the growth of complexity, and helps to maintain productivity as code becomes larger.

4.2. The design of Hermes-3

The design of Hermes-3 is a combination of the Encapsulate Context [64] and Command patterns [65]. The main elements are a flexible store or database, into which values (e.g. spatially dependent fields like densities, temperatures) can be inserted and later retrieved; and a collection of composable model components that set and use values in the store. The approach has similarities to data oriented design [66], in which loose coupling between components is achieved by focusing on defining the data being operated on. In fusion an example of this is the OMFIT framework [67], which uses a tree data structure to loosely couple data sources, codes and analysis scripts.

The data in a simulation is physical quantities, such as density and temperature fields, and derived quantities representing terms in the equations being solved. Hermes-3 stores these quantities in a nested dictionary structure (a tree), using C++ `variant` to enable different data types to be stored. A schema defines a convention for where values are stored, for example `state["species"]["h+"]["density"]` is the number density of hydrogen ions.

Operations on the simulation state are performed by a collection of composable model components, that set and use values in the state. For example there is a component that evolves an equation for fluid number density, another component that evolves pressure. These components are configured when they are created, so that the same code is used to evolve every species that needs that component. Every component can access the whole state, so some perform calculations for a single species, while others perform calculations involving multiple species (e.g. collisions, sheath boundary conditions). The performance penalty of accessing data in this way has been found to be around 20 microseconds (on an Intel(R) Core(TM) i5-7200U 2.50GHz). This overhead is paid once per field, and so the overhead per cell is sub-microsecond in all cases of interest, small compared to the operations performed and communication overheads.

An important distinction between this design and one with a shared global state is that here the state is an object which is passed to components in a user-defined order. This has two advantages: It controls when state can be modified, making the flow of the program easier to understand, and it facilitates unit testing because the inputs to the components can be precisely specified with no hidden side-channels or large setup/teardown procedures.

Controlling when and how data can be modified is crucial to preventing errors, such as components being run out of order so that quantities are set

or modified after use. The values being calculated are physical quantities at the current simulation time, and so logically do not change. It is therefore tempting to adopt an immutable (persistent) data structure, such as the HAMT used in Clojure [68] and available in C++ libraries such as Immer [69]. There are however situations in which components need to modify fields set by earlier components; an important example is applying boundary conditions. Rather than copy arrays of data to maintain immutability, the approach used here is to mark quantities as immutable after they have been used: A quantity can be modified (e.g. boundary conditions applied) only if that quantity has not already been used in a previous calculation.

The flow of information carried by the state through a sequence of components is shown in figure 3. This design restricts when the state can be

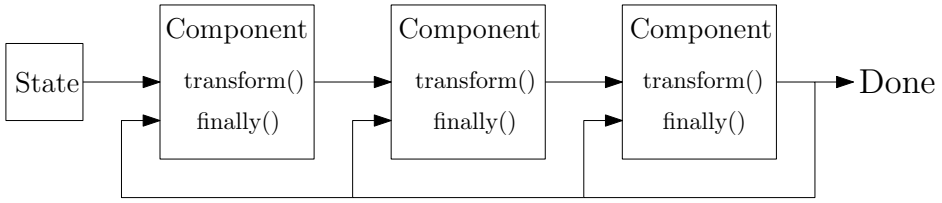


Figure 3: The State of the simulation system is passed through a sequence of Components in two passes. In the first pass components can modify the state passed to their `transform` function; in the second pass the state cannot be modified, but is used to update component internal states in their `finally` functions.

modified, limiting the complexity of interactions between components, and making the logic of the program easier to follow. In Hermes-3 the state is passed to each component twice: The first time the state is mutable, and the component can insert values into it. After all components have been called in this way, the state is "frozen", and passed to each component again but cannot be modified. In this second pass each component can use the final state to update its own internal state, such as time derivatives to be passed to the time integrator. This means that components can depend on each other's outputs, including mutual dependencies, but all modifications must be in the first function (called `transform()`), and in the second function (called `finally()`) all components can assume that the state will not subsequently change. This structure could be exploited to enable all component `finally()` functions to be run in parallel, but this is not currently done in Hermes-3.

The design of Hermes-3 enables run-time configuration of the simulation

equations: All of the examples shown in section 5 use the same executable, despite solving different sets of equations in different numbers of spatial dimensions. The overhead of this flexibility appears to be small, but for high performance scientific simulation codes it is debatable whether run-time configuration is essential: Once the simulation is set up, it is performing the same set of operations repeatedly, calculating time derivatives given different system states. Run-time configuration enables the user (scientist) to modify the equations without recompiling, but means that some errors are only caught at runtime, which might have been caught more quickly at compile time. Compile-time configuration of the equations solved might enable more optimisations, since conditionals can be known and optimised out by the compiler. ReMKiT1D [70] is a plasma simulation code that uses this approach, generating optimised Fortran code from models specified in a higher-level language (python). Just-In-Time (JIT) compilation might offer the best of both worlds; since the operations are the same with different data, run-time analysis of the performance might enable on-the-fly tuning to identify bottlenecks and optimise throughput.

5. Applications

We now describe some applications of Hermes-3, starting from a 1D transport model (section 5.1); then a 2D transport model in 2D (axisymmetric) tokamak geometry (section 5.2). Finally we demonstrate time-dependent capabilities by simulating 2D (drift plane) plasma "blobs" (section 5.3). Application of these capabilities to 3D turbulence simulations is relatively straightforward, but requires significantly more space to describe adequately and so will be explored in separate publications.

5.1. 1D transport

We first apply Hermes-3 to a one-dimensional problem: the flow of heat and particles along a magnetic flux tube which is in contact with a material surface. The model includes electrons, deuterium ions and neutral deuterium atoms. One end of the domain is modelled as being in contact with a material surface, forming a plasma sheath and accelerating ions to the sound speed. The flow of ions to the surface is "recycled" back into the domain as neutral atoms, which then undergo charge exchange and ionisation reactions with the plasma. The other end of the domain has a symmetry (no-flow) boundary condition, where thermodynamic variables (e.g. densities, pressures) have

zero gradient, and flow velocities are zero. This is a widely used model for the divertor region of tokamak plasmas, which has several implementations of varying complexity [71, 72, 73, 74, 75, 76], including the SD1D model [37, 51] which like Hermes-3 is built on BOUT++ [12].

Despite its relative simplicity this model contains many of the nonlinearities and numerically stiff behaviour which make 2D and 3D plasma simulations challenging, including strong nonlinear heat diffusion, and fast reaction rates which are sensitive to electron temperature.

The components to be included in the simulation (section 4.2) are specified in an input text file; the relevant line is shown in listing 1.

```
[hermes]
components = (d+, d, e,
              sheath_boundary, collisions, recycling, reactions,
              electron_force_balance, neutral_parallel_diffusion)
```

Listing 1: Top-level components for 1D hydrogen transport model. Parentheses are used to group multi-line settings. Full input in `examples/1D-recycling` of the Hermes-3 repository [55]

The boundary condition at the material surface, implemented by the `sheath_boundary` component in listing 1, is the multi-ion sheath boundary described in [77]. For the single ion species here this reduces to the standard Bohm-Chodura-Riemann sheath boundary condition [78]. The `collisions` component implements collisions between an arbitrary number of charged and neutral species. Reactions between species are organised into a subsection called `reactions`, and are chosen to have names which are readable and follow a convention for the species labels.

```
[reactions]
type = (
        d + e -> d+ + 2e,     # Deuterium ionisation
        d+ + e -> d,          # Deuterium recombination
        d + d+ -> d+ + d,     # Charge exchange
        )
```

Listing 2: Reactions contained in the 1D hydrogen transport model

Reaction cross-sections for hydrogen and helium have been taken from the Amjuel database [79].

Each particle species has components to evolve the density, pressure and parallel momentum, and a no-flow boundary condition imposed on the upstream boundary. To illustrate how Hermes-3 components combine to form

the equations solved, the `d+` (deuterium ion) species settings are shown in listing 3.

```
[d+]  # Deuterium ions
type = (evolve_density, evolve_pressure, evolve_momentum,
        noflow_boundary, upstream_density_feedback)
charge = 1  # charge
AA = 2      # mass [amu]
density_upstream = 1e19 # Upstream density [m^-3]
recycle_as = d          # Species to recycle as
recycle_multiplier = 1  # Recycling fraction
```

Listing 3: Components to model the deuterium ion species

This implements a set of equations for the density n_{d+}, pressure $p_{d+} = en_{d+}T_{d+}$ and parallel velocity $v_{||d+}$ of the deuterium ions (`d+`) of mass m_{d+} and charge q_{d+}, given in equation 2. These equations are in SI units except

temperatures in eV.

$$\frac{\partial n_{d+}}{\partial t} = -\nabla\cdot\left(n_{d+}\mathbf{b}v_{||d+}\right) + \underbrace{S_{\mathrm{PI}}}_{\texttt{upstream_density_feedback}} + \underbrace{n_e n_d \left\langle\sigma v\right\rangle_{iz}}_{\texttt{d + e -> d+ + 2e}} - \underbrace{n_e n_{d+} \left\langle\sigma v\right\rangle_{rc}}_{\texttt{d+ + e -> d}} \tag{2a}$$

$$\begin{aligned}\frac{\partial p_{d+}}{\partial t} = &-\nabla\cdot\left(p_{d+}\mathbf{b}v_{||d+}\right) - \frac{2}{3}p_{d+}\nabla\cdot\left(\mathbf{b}v_{||d+}\right) + \nabla\cdot\left(\kappa_{||d+}\mathbf{bb}\cdot\nabla T_{d+}\right)\\ &+ \underbrace{n_e n_d \left\langle\sigma v\right\rangle_{iz}\left[eT_d + \frac{1}{2}m_d\left(v_{||,d} - v_{||,d+}\right)^2\right]}_{\texttt{d + e -> d+ + 2e}}\\ &+ \underbrace{n_{d+} n_d \left\langle\sigma v\right\rangle_{cx}\left[e\left(T_d - T_{d+}\right) + \frac{1}{2}m_d\left(v_{||,d} - v_{||,d+}\right)^2\right]}_{\texttt{d + d+ -> d+ + d}}\\ &- \underbrace{n_e n_{d+}\left\langle\sigma v\right\rangle_{rc} eT_{d+}}_{\texttt{d+ + e -> d}} + \underbrace{W_{d+}}_{\texttt{collisions}}\end{aligned} \tag{2b}$$

$$\begin{aligned}\frac{\partial}{\partial t}\left(m_{d+}n_{d+}v_{||d+}\right) = &-\nabla\cdot\left(m_{d+}n_{d+}v_{||d+}\mathbf{b}v_{||d+}\right) - \mathbf{b}\cdot\nabla p_{d+} + q_{d+}n_{d+}E_{||}\\ &+ \underbrace{n_e n_d\left\langle\sigma v\right\rangle_{iz} m_d v_{||d}}_{\texttt{d + e -> d+ + 2e}} - \underbrace{n_e n_{d+}\left\langle\sigma v\right\rangle_{rc} m_{d+}v_{||d+}}_{\texttt{d+ + e -> d}}\\ &+ \underbrace{n_{d+}n_d\left\langle\sigma v\right\rangle_{cx} m_{d+}\left(v_{||d} - v_{||d+}\right)}_{\texttt{d + d+ -> d+ + d}} + \underbrace{F_{d+}}_{\texttt{collisions}}\end{aligned} \tag{2c}$$

Where $\mathbf{b} \equiv \mathbf{B}/B$ is the unit vector in the direction of the magnetic field $\mathbf{B}$, and $\kappa_{||d+}$ is the parallel heat conduction coefficient that depends on the collision frequency calculated by the `collisions` component. For each equation 2 the first line corresponds to the essential transport terms implemented in the `evolve_density`, `evolve_momentum` and `evolve_pressure` components. Additional components, labelled with underbraces in equation 2, add sources and sinks that modify and couple species together. Details of the full system of 7 evolving equations are given in Appendix B.

The equations are integrated in time towards a steady state solution using the backward Euler method described in section 3.1. The result is shown in figure 4. The root-mean-square of the time-derivatives of ion density, pressure and parallel momentum are shown in figure 5 as a function of the

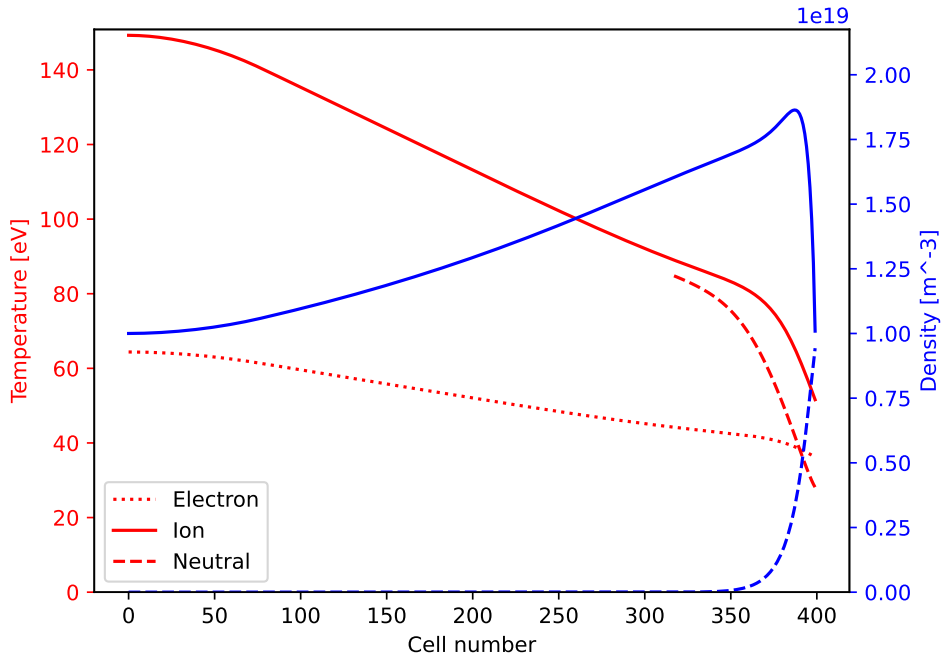


Figure 4: Steady-state solution to system of equations 2 and Appendix B in one dimension. 50MW of power enters a source region on the left, driving plasma-neutral interactions including ionisation, leaving through the sheath boundary on the right. 100% of the plasma ions leaving the right boundary are recycled as neutral atoms. Simulation inputs are in `examples/1D-recycling` in the Hermes-3 repository.

number of right-hand-side (RHS) evaluations, a measure of the computational cost. This evaluation count includes those performed as part of the finite difference Jacobian approximation. This shows a reduction in the time derivatives of the system by almost six orders of magnitude in 10^5 RHS evaluations. For the above system of equations, with 400 grid cells, this calculation takes approximately 30 minutes on a single core. For comparison the convergence towards steady state with the Sundials CVODE [39] library is shown in figure 5. CVODE uses an adaptive order, adaptive timestep Backward Differentiation Formula (BDF) method, and is highly effective for time-dependent problems of interest even without preconditioning (e.g. the plasma blobs application, section 5.3). For this problem only the parallel heat conduction is preconditioned. Figure 5 shows that the Backwards-Euler with Jacobian coloring preconditioner method can provide significantly better performance for steady-state problems, though it would not be a good

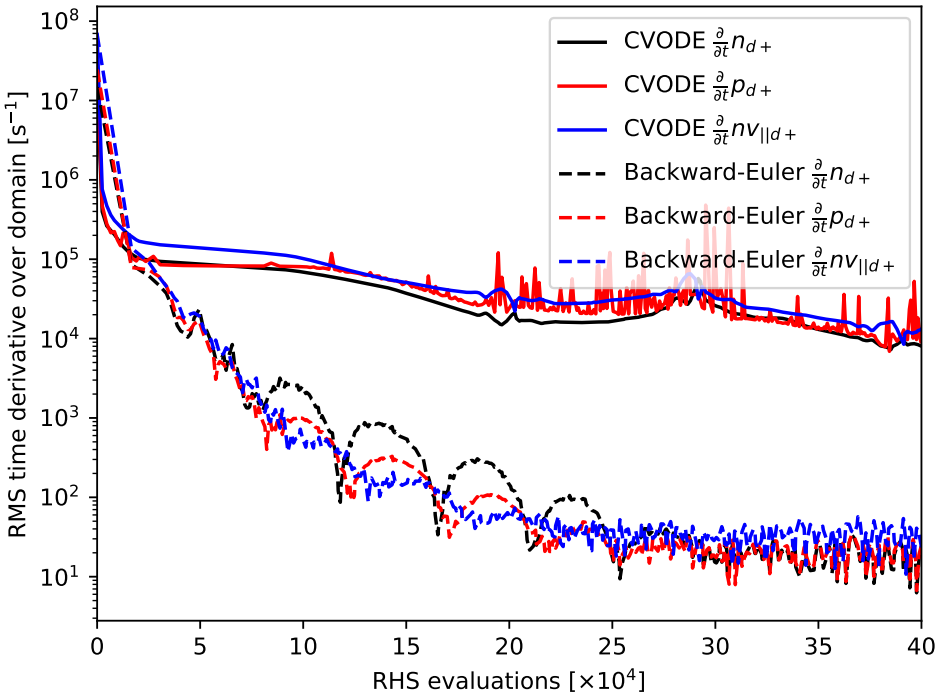


Figure 5: Root-mean-square time derivatives of deuterium density (Nd+), pressure (Pd+) and momentum (NVd+) as a function of iteration (RHS evaluation). These converge towards zero as the system approaches steady state. Results are shown for Backward Euler Newton-Krylov with Jacobian coloring and iLU preconditioning (NK); and the CVODE time integrator. Figure produced by `examples/1D-recycling/plot_convergence.py`.

choice for time-dependent simulations, being only first order accurate in time.

In this simulation the recycling at the "target" end of the domain was set to 100%, while there is a no-flow condition on the upstream boundary. A Proportional-Integral (PI) controller is used to control an upstream particle source (implemented in the `upstream_density_feedback` component); As the target upstream density is approached, this input source should go to zero if mass flow is conserved. Figure 6 shows that this does indeed happen: The source converges towards exponentially towards zero in steady state.

The 1D simulation described here is a useful tool in its own right, for studies of plasma dynamics and detachment in magnetised plasmas. The advantage of the software design used in Hermes-3 (section 4) is that the same code can extend to more complex models and higher dimensions with only changes to the input.

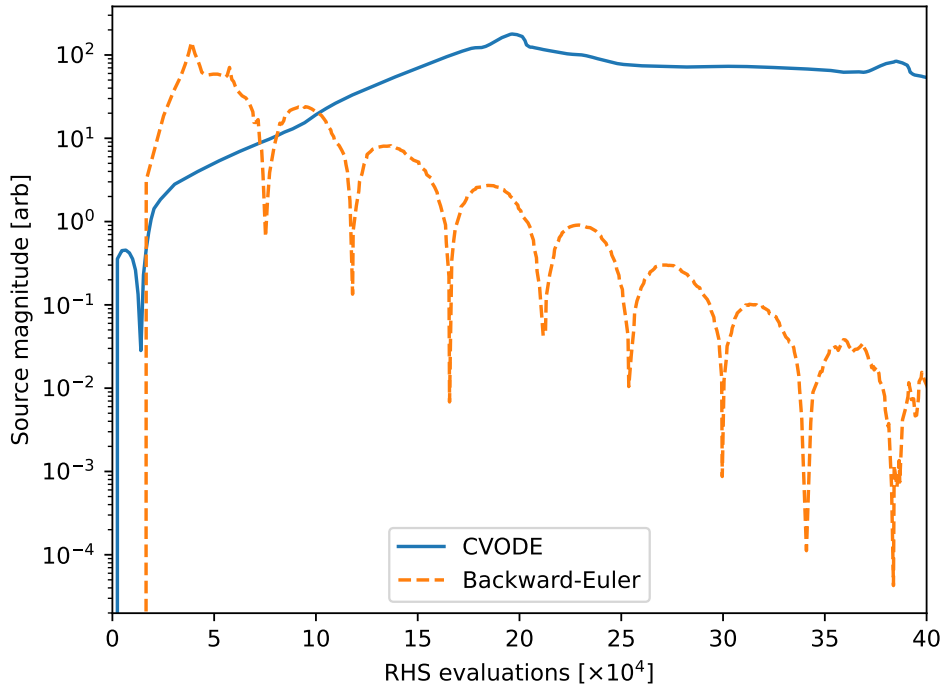


Figure 6: Convergence of the density source with RHS evaluation in a 1D simulation with 100% recycling (figure 4) where the true steady state source is zero. Figure produced by `examples/1D-recycling/plot_convergence.py`.

5.1.1. Impurity seeding

We now extend the 1D simulation described in section 5.1 to multiple ion species, by including all ten charge states of neon as separate species. The simulation now contains 40 evolving fields: The density, pressure and momentum of all deuterium and neon ion and atomic charge states (13 ion species in total), and the electron pressure. These species are coupled through collisions, thermal forces, the parallel electric field, and 32 atomic reactions: ionisation, 3-body recombination and charge exchange recombination (with deuterium ions) of each ionisation level of neon; ionisation and charge exchange of neutral deuterium atoms to deuterium ions. The modular structure of the code (section 4.2) enables this to be accomplished relatively straightforwardly by changing the input file.

```
[hermes]
components = (d+, d, ne, ne+, ne+2, ne+3, ne+4, ne+5, ne+6,
              ne+7, ne+8, ne+9, ne+10, e, sheath_boundary,
```

```
4               thermal_force, collisions, recycling, reactions,
5               electron_force_balance, neutral_parallel_diffusion)
```

Listing 4: Top-level components for 1D transport model with neon. Input `examples/1D-neon`

There are now many more reactions, but the input remains clear:

```
1 [reactions]
2 type = (
3         d + e -> d+ + 2e,     # Deuterium ionisation
4         d + d+ -> d+ + d,     # Charge exchange
5
6         ne + e -> ne+ + 2e,   # Neon ionisation
7         ne+ + e -> ne,        # Neon+ recombination
8         ne+ + d -> ne + d+,   # Neon+ charge exchange recombination
9
10        ...
11        )
```

Listing 5: Reactions contained in the 1D transport model with neon.

The cross-sections and radiated power from the neon reactions are calculated using ADAS [80]: `scd96` and `plt96` for ionisation; `acd96` and `prb96` for recombination; `ccd89` for charge exchange. These files were converted to JSON format using atomic++ [81].

Collisions and the thermal forces between species are calculated as described in section 5.1. Those Braginskii energy and momentum exchange rates are approximations which are only strictly valid when heavy ions are trace impurities. In the simulations shown here the neon concentration is small (a fraction of a percent). More complete models of collisions in a multi-ion plasma have been derived [82] and recently generalised [83]. Implementing these models into Hermes-3 is left as future work, but is not anticipated to present any fundamental difficulty.

Starting from the hydrogen simulation described above (section 5.1), a simulation with 100% recycling and an initial uniform low concentration of neon is run to steady state, and shown in figure 7. In this simulation there is no net flow upstream of the ionisation region, and so thermal forces drive neon impurities upstream. The steady-state solution is shown in figure 7. Future applications of this capabilty include simulating impurity-seeded plasma detachment phenomena.

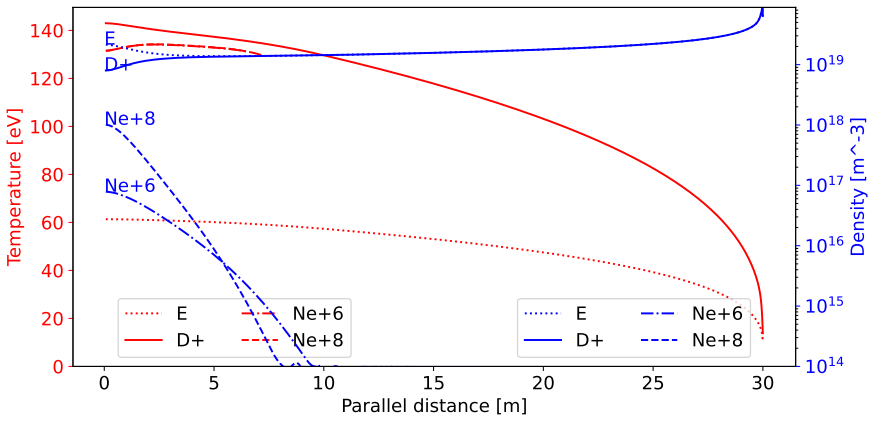
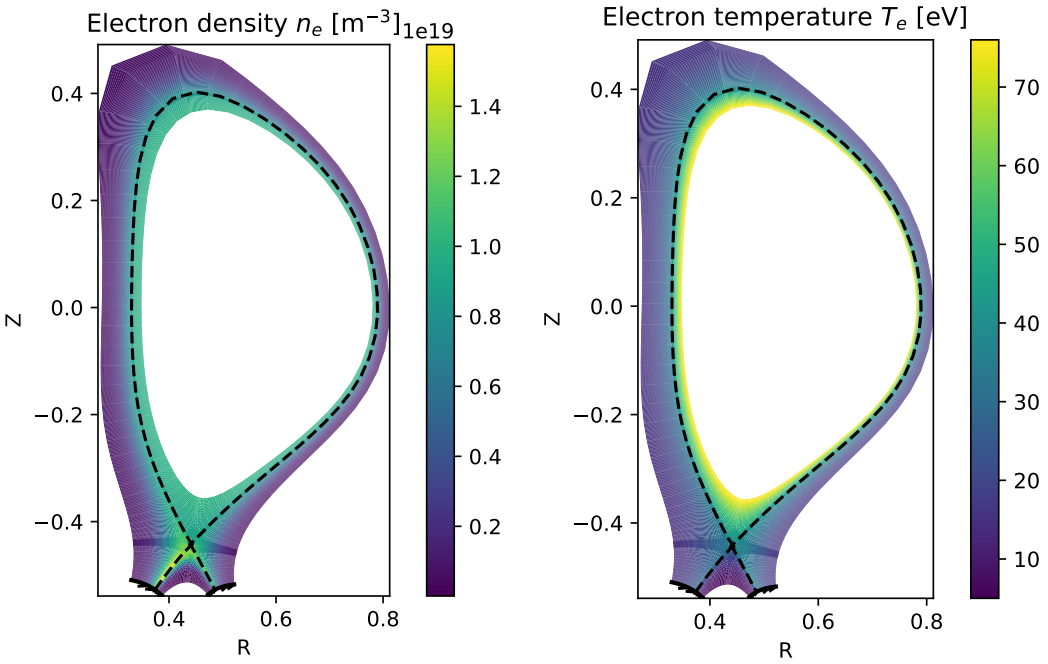


Figure 7: Steady state solution with 100% recycling, evolving all charge states of neon as separate fluids with their own densities, temperatures and flow velocities. A subset of species densities (blue lines) are shown on a logarithmic scale. Simulation inputs in `examples/1D-neon` of the Hermes-3 repository.

5.2. 2D (axisymmetric) transport

The same code that is used in a 1D domain in the previous sections can be applied to 2D tokamak domains with one or two X-points. By introducing cross-field diffusion of both charged and neutral species, an axisymmetric tokamak transport simulation in the spirit of SOLPS [1], EDGE2D [3] or UEDGE [2] can be performed, though not yet at a comparable level of maturity or completeness. To demonstrate the ability of Hermes-3 to solve axisymmetric transport problems, simulations are performed with deuterium ions and neutral atoms. Diffusion coefficients and plasma parameters are taken from [84, 85]: Spatially constant cross-field diffusion coefficients for particle transport $D_n = 0.15\mathrm{m}^2/\mathrm{s}$, electron and ion thermal transport $\chi_e = \chi_i = 4\mathrm{m}^2/\mathrm{s}$. In general these coefficients can be functions of location, and can be different for each species.

The plasma equilibrium is based on a COMPASS-like equilibrium generated using analytic Grad-Shafranov solutions [86, 87]. The domain simulated is shown in figure 8, consisting of a narrow annulus around the separatrix (dashed black lines in figure 8) including closed and open field line regions. The radial boundaries are at normalised poloidal flux (ψ_N) of 0.9 in the core and in the private flux region (PFR), and 1.3 in the Scrape-Off Layer (SOL). The Hypnotoad tool [46] was used to generate a sequence of grids of



(a) Electron density. The core boundary is fixed to a density of $1 \times 10^{19}\mathrm{m}^{-3}$. At divertor targets 99% of ion flux is recycled as neutral atoms.

(b) Electron temperature. At the core boundary both electron and ion temperatures are fixed to 75eV.

Figure 8: Axisymmetric tokamak transport simulation of deuterium ions and atoms. Equations given in Appendix B. Simulation inputs in `examples/tokamak/recycling` of the Hermes-3 repository.

increasing resolution from 16×24 to 64×96 (radial $\times$ poloidal cells).

As in previous examples, the equations solved are specified as a set of components:

```
components = (d+, d, e,
              collisions, sheath_boundary_simple,
              recycling, sound_speed, reactions,
              electron_force_balance)
```

Listing 6: Top-level components for 2D transport model. Full input in `examples/tokamak/recycling` of the Hermes-3 repository.

The deuterium ion species is configured with a set of components representing the equations solved, given in listing 7

```
[d+]
type = (evolve_density, evolve_momentum, evolve_pressure,
```

```
        anomalous_diffusion)
anomalous_D = 0.15   # Density diffusion [m^2/s]
anomalous_chi = 4    # Thermal diffusion [m^2/s]
...
```

Listing 7: Deuterium ion components for 2D transport model

which is similar to the configuration in 1D simulations given in listing 3, but adds anomalous cross-field diffusion terms. Reactions between species are calculated using Amjuel rates [79], comprising ionisation, recombination, and charge exchange processes as described in section 5.1.

At the inner (core) boundary the deuterium density is fixed to $1\times10^{19}\mathrm{m}^{-3}$; electron and ion temperatures are set to 75eV. This core boundary therefore acts as a source of heat and particles. At the target plates a sheath boundary condition is applied in which the plasma flow goes to the sound speed, with a recycling fraction of 0.99 so that there is a flux of neutral atoms into the domain at the target plates. The 1% of ion flux that is not recycled is balanced in steady state by a diffusion of ions from the core boundary. This particle flux balance will be used in section 5.2.2 to verify the conservation of particles in these simulations.

The heat flux along the magnetic field into the target plates is given by $q_{||e,i} = \gamma_{e,i} neTc_s$ with sheath heat transmission factors $\gamma_e = 4.8$ for electrons and $\gamma_i = 3.5$ for ions. The sound speed into the sheath is $c_s = \sqrt{e\left(T_i + T_e\right)/m_i}$. There are no diffusive fluxes to the outer walls because zero-gradient boundary conditions are used there. The power into the target plates should therefore equal the input power through the core boundary, less the power radiated during atomic processes (primarily ionisation). This is used in section 5.2.2 to assess conservation of energy.

The full set of equations solved are given in Appendix B.

5.2.1. Evolution to steady state

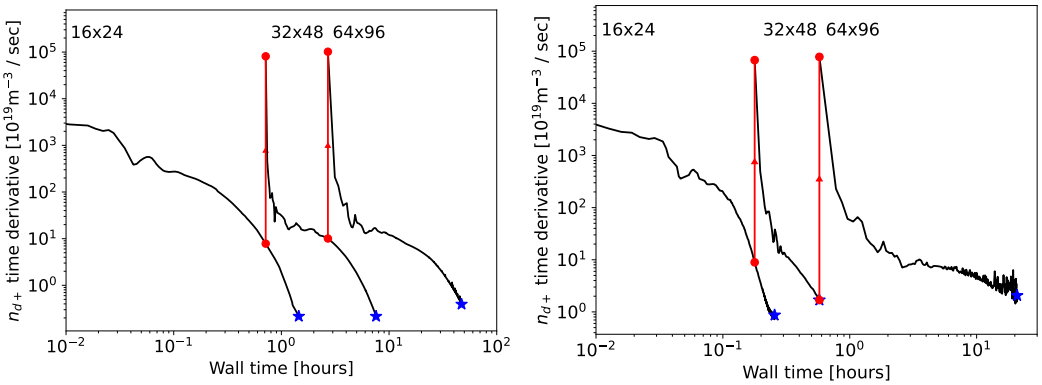
The system of transport equations is relatively small (e.g. 10,752 variables for the 32×48 mesh) but highly nonlinear and with a wide range of timescales, making finding steady state solutions challenging. Simple application of a nonlinear solver does not converge in most cases of interest, and the system must be regularised using a (pseudo-)timestepping approach. As the system approaches steady state the timestep can be made progressively larger, usually in an automated manner based on the number of nonlinear iterations required to converge the previous step. The combination of time

integration method, time step adjustment heuristic, nonlinear solver, inner linear iterative solver, and preconditioner have many parameters that can affect performance. The nested methods interact in ways that are problem-dependent, making general conclusions regarding performance difficult to draw. The performance results presented here (e.g. figure 9) should therefore be taken as somewhat anecdotal, the important aspects being the energy and particle conservation properties (table 1) and scaling with problem size.

In general power balance reaches steady-state on a shorter timescale than the particle balance: The thermal energy content of the system (plasma + neutrals) is $W \simeq 47$J, so the energy confinement time is $\tau_E \equiv W/P_{in} \simeq 0.24$ms. On the other hand the ion particle content is approximately 2×10^{18}, giving a particle throughput timescale of $\tau_p \simeq 45$ms. This longer particle balance timescale becomes increasingly challenging at high recycling fractions relevant to large fusion devices.

An effective strategy, already used routinely and interactively in UEDGE, is progressive mesh refinement: Starting on the coarsest mesh (16×24 here), CVODE is used with an absolute tolerance of 10^{-12} and relative tolerance 10^{-5}, tightening the relative tolerance to 10^{-8} as steady state is approached. These tolerances can be loosened in some cases, but at the risk of numerical instability and convergence failure after a number of steps. Once progress has been made on a coarse mesh, the solution is interpolated onto a higher resolution mesh (using SciPy's RegularGridInterpolator [88] over logically rectangular mesh patches). The simulation is then continued using an increased number of cores. The refinement process may be repeated. Figure 9 shows the Root-Mean-Square (RMS) of the time derivatives of the plasma density averaged over the domain, as a function of wall clock time (running on NERSC's Perlmutter). For each mesh resolution the simulation was continued after interpolation, until the Root-Mean-Square time scale exceeded one second, to minimise the impact of mesh interpolation error on the comparison of solutions. As the mesh was refined the number of cores used was increased following a weak scaling. The increase in run time with grid resolution in figure 9 is primarily driven by the number of iterations required: For CVODE the iteration counts are 3.1×10^6 (16×24 mesh), 1.2×10^7 (32×48 mesh) and 3.7×10^7 (64×96 mesh). The time per iteration (RHS evaluation) is 1.7ms, 2.0ms and 4.3ms respectively.

Figure 9b shows results using the Backward Euler method with Jacobian coloring and Euler ILU preconditioner that was described in section 3.1 and used in 1D simulations (section 5.1). In general a significant improve-



(a) CVODE solver with heat conduction preconditioner

(b) Backward Euler solver with PETSc matrix coloring and Hypre's Euler preconditioner

Figure 9: Evolution of the Root-Mean-Square (RMS) time derivative residuals. Vertical red arrows indicate where the solution is interpolated onto a higher resolution mesh. Blue stars are the solutions that are compared in section 5.2.2. The number of cores used is increased with the grid resolution: 12 (16×24 mesh), 48 (32×48 mesh) and 192 (64×96 mesh).

ment in performance is found, primarily due to a reduction in the number of iterations required. The speedup is most significant for small meshes and processor counts, where direct ILU preconditioners would be expected to have an advantage over matrix-free methods, but that advantage to reduces for large meshes and processor counts.

5.2.2. Convergence and accuracy

The accuracy of the methods are now assessed by examining the conservation properties and convergence of the solution with mesh resolution. Figure 10 shows the profiles of density and temperature along the outer target, for each mesh resolution. Low resolution meshes broaden the profiles of both density and temperature relative to high resolution cases: Numerical dissipation enhances the effective cross-field diffusion. Given the fixed (Dirichlet) core boundary conditions, this enhanced diffusion increases the power into the domain at low resolution. It also be seen in figure 10 that the meshes do not have a consistent boundary location: Due to boundary cell locations, as the grid is refined the outer edges of the domain converge at first order to the specified poloidal flux values. This will limit the global formal convergence to at best first order unless improvements are made to the mesh generator.

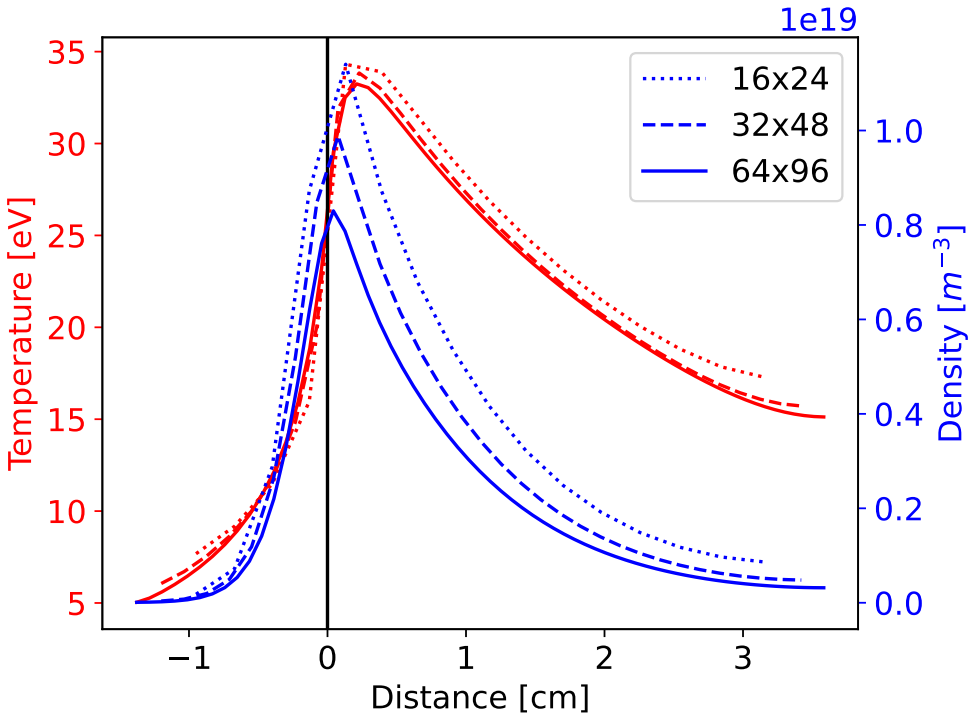


Figure 10: Electron temperature and density at the outer target. Dotted: 16×24 resolution; Dashed: 32×48; Solid: 64×96.

To assess power and particle balance, table 1 lists the flows of power and particles into and out of the domain, for each mesh resolution. Power enters

Table 1: Global power and particle balance in 2D transport simulations

	16×24	32×48	64×96
Input power [kW]	195.6	174.9	162.9
Power to outer target [kW]	104.6	92.5	85.9
Power to inner target [kW]	75.4	64.9	59.5
Power to atomics [kW]	22.8	19.3	17.6
Power balance error [kW]	7.2 (3.7%)	1.7 (0.98%)	0.092 (0.056%)
Input ion flux [10^{19}/s]	5.31	4.56	4.15
Flux to outer target [10^{19}/s]	278	247	230
Flux to inner target [10^{19}/s]	284	229	204
Recycling fraction [0.99]	0.9909	0.9903	0.9904

the domain through the inner (core) boundary, where Dirichlet boundary conditions are set on density and temperatures so that power crossing this boundary depends on the local gradients. The target temperatures in figure 10 are well above the 5eV typical for detachment, so these simulations are in attached conditions and most of the power goes to the outer and inner targets. Some power is lost through atomic processes, both to overcome ionisation potentials and through radiation. Power to atomics includes the deuterium ionisation potential so this potential energy flux is not included in the power to outer and inner targets listed in table 1. As noted in section 3.2 pressure equations are evolved rather than energy, so that energy conservation is in general not exact but converges as the mesh is refined. For comparison, a 1% power balance error has been used as a SOLPS-ITER convergence criterion [89].

Particle fluxes are shown in Table 1 as the flux into the domain through the inner boundary, and the fluxes to inner and outer targets. Due to the imposed recycling fraction of 0.99, we expect 1% of the flux to the targets to be lost (pumped), and replaced by a matching flux of ions into the domain from the core. The core and target fluxes are therefore used to infer the recycling fraction in Table 1. If particle balance is achieved then that fraction should match the 0.99 value set. We find this to be well matched: Particle conservation is significantly easier to achieve in this system of equations than energy conservation, and these results demonstrate that all advection and

diffusion operators, recycling and atomic processes, properly conserve particle fluxes.

5.3. 2D (drift-plane) blobs

We now turn from steady-state transport problems to time-dependent problems involving an evolving vorticity equation and electrostatic potential ϕ. The development of this capability towards full 3D turbulence, particularly in the presence of multiple ion species, will be the subject of a future publication. As an initial step and proof of principle, we present here some examples of 2D drift-plane simulations of plasma "blobs" or filaments.

The significant lines in the input file which configure this model are shown in listing 8.

```
[hermes]
components = e, vorticity, sheath_closure

[e]  # Electrons
type = evolve_density, isothermal
charge = -1
AA = 1/1836      # Mass of species [amu]
temperature = 5  # Temperature in eV

[sheath_closure]
connection_length = 10 # meters
```

Listing 8: Component configuration for isothermal blob simulation. Full input in `examples/blob2d` of the Hermes-3 repository.

These set up components for the electron species density and (isothermal) temperature, a vorticity equation, and a model for the divergence of parallel

current due to the sheath closure. This corresponds to model equations

$$\frac{\partial n_e}{\partial t} = -\nabla \cdot \left(n_e \frac{1}{B} \mathbf{b} \times \nabla \phi \right) + \underbrace{\nabla \cdot \frac{1}{e} \mathbf{j}_{sh}}_{\texttt{sheath_closure}} \tag{3a}$$

$$p_e = \underbrace{e n_e T_e}_{\texttt{isothermal}} \tag{3b}$$

$$\frac{\partial \Omega}{\partial t} = -\nabla \cdot \left(\Omega \frac{1}{B} \mathbf{b} \times \nabla \phi \right) + \nabla \left(p_e \nabla \times \frac{\mathbf{b}}{B} \right) + \underbrace{\nabla \cdot \mathbf{j}_{sh}}_{\texttt{sheath_closure}} \tag{3c}$$

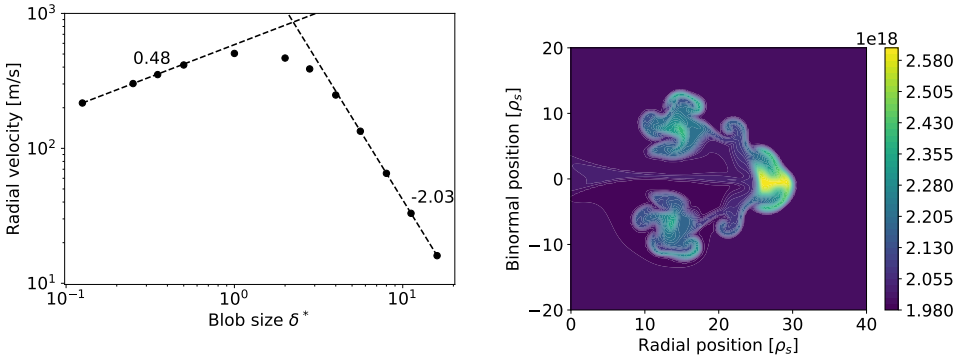
$$\nabla \cdot \left(\frac{\overline{m_i n}}{B^2} \nabla_\perp \phi \right) = \Omega \tag{3d}$$

where n_e is the electron density, p_e the pressure and T_e the (fixed) temperature. The Boussinesq approximation is used here, so the potential ϕ is calculated from vorticity Ω using equation 3d with a constant mass density $\overline{m_i n}$. The divergence of current density to the sheath is $\nabla \cdot \mathbf{j}_{sh} = n_e \phi / L_{||}$ where $L_{||}$ is the connection length (10m here).

The scaling of sheath-connected isothermal plasma blobs with blob size is a well known test case, which can be derived analytically in the limits of large blobs where sheath current balances the divergence of diamagnetic current, and for small blobs where polarisation current balances the divergence of diamagnetic current (see e.g. [90]). The blob size δ for which the divergence of polarisation and sheath currents contribute approximately equally is denoted δ^*.

Simulations are started with a circular Gaussian density perturbation (a plasma "blob"), whose size perpendicular to the magnetic field and the size of the simulation domain is varied. Because we are interested in time-varying solutions to these equations (propagation of plasma blobs), the CVODE time integrator [39] is used, not the backward Euler method used in section 5.1. The result is shown in figure 11a, reproducing well known scaling of plasma blob velocity with blob size [90].

This model extends quite straightforwardly to include hot ion effects and separate ion and electron temperatures, by modifying the input file to introduce a new species `h+` with a separate pressure equation. The vorticity formulation is implemented such that the polarisation current contribution of multiple species is included in calculating the electric field; The self-consistent calculation of the polarisation drift on the ion species density in a multi-ion



(a) Radial velocity v_r of an isothermal plasma blob as a function of blob size δ. Analytical scalings are $v_r \sim \sqrt{\delta}$ for $\delta/\delta^* \ll 1$, and $v_r \sim (\delta/\delta^*)^{-2}$ for $\delta/\delta^* \gg 1$.

(b) Electron density n_e at $t = 1500/\omega_{ci} = 44.7\mu$s for a δ^*-sized seeded blob.

Figure 11: Solution to equations 3 in a 2D domain perpendicular to the magnetic field, starting with a circular cross section density perturbation and driven by magnetic field curvature. Input and analysis scripts in Hermes-3 repository `examples/blob2d`.

species calculation has recently been implemented and is being tested. Further examples, tests and applications may be found in the Hermes-3 manual [59] and source code repository [55].

6. Conclusions

Advancing understanding of the physics of the edge of tokamak plasmas drives the need for increasingly complex models. To address this need a new open-source plasma simulation tool has been developed that enables researchers to perform complex multi-species plasma simulations by combining reusable software components. This is achieved by building on the BOUT++ framework of partial differential equation solvers; adding conservative finite difference methods that are implemented in terms of fluxes between cells to enable accurate particle and power flux balance; and defining a flexible yet robust method of coupling components together within a parallelised high-performance code.

An important benefit of the Hermes-3 code structure is the ease with which components can be unit tested, due to the lack of global state and control of where and in what order evolving quantities can be modified. The public Git repository includes a suite of unit, integrated and Method of Manufactured Solutions (MMS) tests that are used routinely to check

the correctness of code changes. As discussed in section 4.2, a potential disadvantage of this modular structure is the increased difficulty of optimising performance for a specific application. BOUT++ and Hermes-3 are intended to be capable of production simulations of complex 3D plasma phenomena on leadership-class computing facilities, but the focus of their development is on versatility, to facilitate development of better models.

Applications of Hermes-3 to simulations of tokamak plasmas have been presented: Time dependent simulations of plasma filament/blob propagation and steady-state transport including atomic reactions. Convergence tests and comparisons to analytic solutions have been carried out, demonstrating good conservation properties and convergence of the methods. A new backward-Euler solver that has been implemented in BOUT++, employs a Jacobian coloring method in PETSc, and was inspired by the approach used in UEDGE has been applied to efficiently find steady-state solutions (section 5.2.1). In these applications where accurate time integration is not required, only the steady state solution, low order methods with robust iLU preconditioning can be effective. High-order time integration using SUNDIALS is used in time-dependent simulations where accurate time evolution is required.

Future work includes application of Hermes-3 to 3D tokamak edge turbulence: Simulations of the TCV-X21 diverted L-mode reference case [28] were recently presented [91] and will be submitted for publication shortly. 3D simulations motivate future work to characterise and improve the computational efficiency and scalability of Hermes-3 including use of GPU hardware, building on new capabilities in the BOUT++ framework and *hypre* library [92].

Hermes-3 is publicly available [55] on Github under a GPL-3 license. To maximise its utility to the plasma community a set of examples are included, and a manual [59] provides an introduction for new users.

Acknowledgements

This work was in part performed under the auspices of the U.S. DoE by LLNL under Contract DE-AC52-07NA27344, and received funding from LLNL LDRD 23-ERD-015. B.Dudson would like to thank Dr. Wayne Arter (CCFE) for useful discussions and suggestions, and reviewers for their constructive comments that improved the manuscript. The Hermes-3 source code, simulation inputs, and processing scripts needed to reproduce the results shown in this paper are available at `https://github.com/bendudson/hermes-3`, Git commit `5f56919` (Version 1.1.0). LLNL-CODE-845139.

7. Bibliography

[1] R. Schneider, X. Bonnin, K. Borrass, D. P. Coster, H. Kastelewicz, D. Rieter, V. A. Rozhansky, B. J. Braams, Contrib. Plasma Phys. 46 (1-2) (2006) 3–191. doi:10.1002/ctpp.200610001.

[2] T. D. Rognlien, X. Q. Xu, A. C. Hindmarsh, J. Comput. Phys. 175 (2002) 249–268.

[3] R. Simonini, G. Corrigan, G. Radford, J. Spence, A. Taroni, Models and Numerics in the Multi-Fluid 2-D Edge Plasma Code EDGE2D/U, Contrib. Plasma Phys. 34 (1994) 368–373. doi:10.1002/ctpp.2150340242.
URL https://doi.org/10.1002/ctpp.2150340242

[4] Z. H. Wang, X. Q. Xu, T. Y. Xia, T. D. Rognlien, 2D simulations of transport dynamics during tokamak fuelling by supersonic molecular beam injection, Nucl. Fusion 54 (4) (2014) 043019. doi:10.1088/0029-5515/54/4/043019.
URL https://doi.org/10.1088/0029-5515/54/4/043019

[5] M. Giacomin, P. Ricci, A. Coroado, G. Fourestey, D. Galassi, E. Lanti, D. Mancini, N. Richart, L. Stenger, N. Varini, The gbs code for the self-consistent simulation of plasma turbulence and kinetic neutral dynamics in the tokamak boundary, J. Comput. Phys. 463 (2022) 111294. doi:https://doi.org/10.1016/j.jcp.2022.111294.
URL https://www.sciencedirect.com/science/article/pii/S0021999122003564

[6] F. D. Halpern, P. Ricci, S. Jolliet, J. Loizu, J. Morales, A. Mosetto, F. Musil, F. Riva, T. M. Tran, C. Wersal, The GBS code for tokamak scrape-off layer simulations, Journal of Computational Physics 315 (2016) 388–408. doi:10.1016/j.jcp.2016.03.040.

[7] P. Ricci, F. D. Halpern, S. Jolliet, J. Loizu, A. Mosetto, A. Fasoli, I. Furno, C. Theiler, Plasma Phys. Control. Fusion 54 (2012) 124047.

[8] P. Tamain, P. Ghendrih, E. Tsitrone, V. Grandgirard, X. Garbet, Y. Sarazin, E. Serre, G. Ciraolo, G. Chiavassa, J. Comput. Phys. 229 (2) (2010) 361–378. doi:10.1016/j.jcp.2009.09.031.

[9] P. Tamain, H. Bufferand, G. Ciraolo, C. Colin, D. Galassi, P. Ghendrih, F. Schwander, E. Serre, J. Comput. Phys. 321 (2016) 606–623. doi:10.1016/j.jcp.2016.05.038.

[10] J. Madsen, V. Naulin, A. H. Nielsen, J. J. Rasmussen, Collisional transport across the magnetic field in drift-fluid models, Physics of Plasmas 23 (3) (2016) 032306. doi:10.1063/1.4943199.

[11] B. D. Dudson, et al., Comp. Phys. Comm. 180 (2009) 1467–1480. doi:DOI:10.1016/j.cpc.2009.03.008, [link].
URL http://www.sciencedirect.com/science/article/B6TJ5-4VTCM95-3/2/ed200cd23916d02f86fda4ce6887d798

[12] B. D. Dudson, et al., J. Plasma Phys. 81 (01) (2015) 365810104. doi:10.1017/S0022377814000816, [link].
URL http://doi.org/10.1017/S0022377814000816

[13] BOUT++ contributors, BOUT++ manual, https://bout-dev.readthedocs.io/.

[14] B. D. Dudson, J. Leddy, Plasma Phys. Control. Fusion 59 (2017) 054010. doi:10.1088/1361-6587/aa63d2, [link].
URL https://doi.org/10.1088/1361-6587/aa63d2

[15] J. Leddy, B. Dudson, H. Willett, Simulation of the interaction between plasma turbulence and neutrals in linear devices, Nucl. Mat. Energy 12 (2017) 994–998, Proceedings of the 22nd International Conference on Plasma Surface Interactions 2016, 22nd PSI. doi:10.1016/j.nme.2016.09.020.
URL https://www.sciencedirect.com/science/article/pii/S2352179116300497

[16] L. Easy, F. Militello, J. Omotani, B. Dudson, E. Havlíčková, P. Tamain, V. Naulin, A. H. Nielsen, Three dimensional simulations of plasma filaments in the scrape off layer: A comparison with models of reduced dimensionality, Physics of Plasmas 21 (12) (2014) 122515. arXiv:https://doi.org/10.1063/1.4904207, doi:10.1063/1.4904207.
URL https://doi.org/10.1063/1.4904207

[17] L. Easy, F. Militello, T. Nicholas, J. Omotani, F. Riva, N. Walkden, UKAEA, Storm drift-reduced plasma fluid model, https://github.com/boutproject/STORM.

[18] A. Coroado, P. Ricci, A self-consistent multi-component model of plasma turbulence and kinetic neutral dynamics for the simulation of the tokamak boundary, arXiv (2021) 2110.13335.
URL https://arxiv.org/abs/2110.13335

[19] S. Raj, N. Bisai, V. Shankar, A. Sen, J. Ghosh, R. L. Tanna, M. B. Chowdhuri, K. A. Jadeja, K. Asudani, T. Macwan, S. Aich, K. Singh, Studies on impurity seeding and transport in edge and SOL of tokamak plasma, Nuclear Fusion (2021). doi:10.1088/1741-4326/ac44b0.
URL http://iopscience.iop.org/article/10.1088/1741-4326/ac44b0

[20] A. Poulson, J. Juul Rasmussen, M. Wiesenberger, V. Naulin, Collisional multispecies drift fluid model, Physics of Plasmas 27 (2020) 032305. doi:10.1063/1.5140522.
URL https://doi.org/10.1063/1.5140522

[21] W. Zholobenko, A. Stegmeir, M. Griener, G. Conway, T. Body, D. Coster, F. Jenko, the ASDEX Upgrade Team, The role of neutral gas in validated global edge turbulence simulations, Nucl. Fusion 61 (11) (2021) 116015. doi:10.1088/1741-4326/ac1e61.
URL https://dx.doi.org/10.1088/1741-4326/ac1e61

[22] H. Bufferand, J. Balbin, S. Baschetti, J. Bucalossi, G. Ciraolo, P. Ghendrih, R. Mao, N. Rivals, P. Tamain, H. Yang, G. Giorgiani, F. Schwander, M. S. d'Abusco, E. Serre, J. Denis, Y. Marandet, M. Raghunathan, P. Innocente, D. Galassi, J. Contributors, Implementation of multi-component zhdanov closure in soledge3x, Plasma Physics and Controlled Fusion 64 (5) (2022) 055001. doi:10.1088/1361-6587/ac4fac.
URL https://dx.doi.org/10.1088/1361-6587/ac4fac

[23] F. Riva, F. Militello, S. Elmore, J. T. Omotani, B. Dudson, N. R. Walkden, Three-dimensional plasma edge turbulence simulations of the Mega Ampere Spherical Tokamak and comparison with experimental measurements, Plasma Phys. Control. Fusion 61 (2019) 095013.

doi:10.1088/1361-6587/ab3561.
URL https://doi.org/10.1088/1361-6587/ab3561

[24] A. Stegmeir, D. Coster, A. Ross, O. Maj, K. Lackner, E. Poli, Grillix: a 3d turbulence code based on the flux-coordinate independent approach, Plasma Physics and Controlled Fusion 60 (3) (2018) 035005. doi:10.1088/1361-6587/aaa373.
URL https://dx.doi.org/10.1088/1361-6587/aaa373

[25] H. Bufferand, P. Tamain, S. Baschetti, J. Bucalossi, G. Ciraolo, N. Fedorczak, P. Ghendrih, F. Nespoli, F. Schwander, E. Serre, Y. Marandet, Three-dimensional modelling of edge multi-component plasma taking into account realistic wall geometry, Nuclear Materials and Energy 18 (2019) 82–86. doi:https://doi.org/10.1016/j.nme.2018.11.025.
URL https://www.sciencedirect.com/science/article/pii/S2352179118302035

[26] B. D. Dudson, W. A. Gracias, R. Jorge, A. H. Nielsen, J. M. B. Olsen, P. Ricci, C. Silva, P. Tamain, G. Ciraolo, N. Fedorczak, D. Galassi, J. Madsen, F. Militello, N. Nace, J. J. Rasmussen, F. Riva, E. Serre, Edge turbulence in isttok: a multi-code fluid validation, Plasma Phys. Control. Fusion 63 (5) (2021) 055013. doi:10.1088/1361-6587/abe21d.
URL https://dx.doi.org/10.1088/1361-6587/abe21d

[27] D. Galassi, C. Theiler, T. Body, F. Manke, P. Micheletti, J. Omotani, M. Wiesenberger, M. Baquero-Ruiz, I. Furno, M. Giacomin, E. Laribi, F. Militello, P. Ricci, A. Stegmeir, P. Tamain, H. Bufferand, G. Ciraolo, H. De Oliveira, A. Fasoli, V. Naulin, S. L. Newton, N. Offeddu, D. S. Oliveira, E. Serre, N. Vianello, Validation of edge turbulence codes in a magnetic X-point scenario in TORPEX, Physics of Plasmas 29 (1) (2022) 012501. arXiv:https://pubs.aip.org/aip/pop/article-pdf/doi/10.1063/5.0064522/16624860/012501_1_online.pdf, doi:10.1063/5.0064522.
URL https://doi.org/10.1063/5.0064522

[28] D. Oliveira, T. Body, D. Galassi, C. Theiler, E. Laribi, P. Tamain, A. Stegmeir, M. Giacomin, W. Zholobenko, P. Ricci, H. Bufferand,

J. Boedo, G. Ciraolo, C. Colandrea, D. Coster, H. de Oliveira, G. Fourestey, S. Gorno, F. Imbeaux, F. Jenko, V. Naulin, N. Offeddu, H. Reimerdes, E. Serre, C. Tsui, N. Varini, N. Vianello, M. Wiesenberger, C. Wüthrich, the TCV Team, Validation of edge turbulence codes against the tcv-x21 diverted l-mode reference case, Nucl. Fusion 62 (9) (2022) 096001. doi:10.1088/1741-4326/ac4cde.
URL https://dx.doi.org/10.1088/1741-4326/ac4cde

[29] BOUT++ contributors, BOUT++ project repositories, https://github.com/boutproject/.

[30] X. Q. Xu, B. Dudson, P. B. Snyder, M. V. Umansky, H. Wilson, Nonlinear Simulations of Peeling-Ballooning Modes with Anomalous Electron Viscosity and their Role in Edge Localized Mode Crashes, Phys. Rev. Lett. 105 (2010) 175005.

[31] T. Y. Xia, X. Q. Xu, P. W. Xi, Six-field two-fluid simulations of peeling–ballooning modes using BOUT++, Nucl. Fusion 53 (7) (2013) 073009. doi:10.1088/0029-5515/53/7/073009.
URL https://doi.org/10.1088/0029-5515/53/7/073009

[32] X. Q. Xu, P. W. Xi, A. Dimits, I. Joseph, M. V. Umansky, T. Y. Xia, B. Gui, S. S. Kim, G. Y. Park, T. Rhee, H. Jhang, P. H. Diamond, B. Dudson, P. B. Snyder, Gyro-fluid and two-fluid theory and simulations of edge-localized-modes, Physics of Plasmas 20 (2013) 056113. doi:10.1063/1.4801746.
URL https://doi.org/10.1063/1.4801746

[33] B. Zhu, H. Seto, X. Q. Xu, M. Yagi, Drift reduced landau fluid model for magnetized plasma turbulence simulations in BOUT++ framework, Comp. Phys. Comm. 267 (2021) 108079. doi:10.1016/j.cpc.2021.108079.
URL https://doi.org/10.1016/j.cpc.2021.108079

[34] M. L. Magnussen, Global numerical modeling of magnetized plasma in a linear device, Ph.D. thesis, Department of Physics, Technical University of Denmark (2017).
URL https://findit.dtu.dk/en/catalog/5953d28a5010df5cc9601961

[35] A. Coroado, P. Ricci, Moving toward mass-conserving simulations of plasma turbulence and kinetic neutrals in the tokamak boundary with the GBS code, Physics of Plasmas 28 (2021) 022310. doi:10.1063/5.0027977.
URL https://doi.org/10.1063/5.0027977

[36] B. Mortier, G. Samaey, T. Baelmans, Enforcing conservation at monte carlo level in a coupled finite volume - monte carlo simulation, Contrib. Plasma Phys. 58 (6-8) (2018) 681–687. doi:10.1002/ctpp.201700180.
URL https://doi.org/10.1002/ctpp.201700180

[37] B. D. Dudson, J. Allen, T. Body, B. Chapman, C. Lau, L. Townley, D. Moulton, J. Harrison, B. Lipschultz, The role of particle, energy and momentum losses in 1d simulations of divertor detachment 61 (6) (2019) 065008. doi:10.1088/1361-6587/ab1321.
URL https://doi.org/10.1088/1361-6587/ab1321

[38] Ben Dudson, Jarrod Leddy, Hasan Muhammed, Hermes-2 hot ion drift-reduced model., https://github.com/bendudson/hermes-2.

[39] A. C. Hindmarsh, et al., SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers, ACM Transactions on Mathematical Software 31 (3) (2005) 363–396.

[40] H. Seto, X. Q. Xu, B. D. Dudson, M. Yagi, Interplay between fluctuation driven toroidal axisymmetric flows and resistive ballooning mode turbulence, Physics of Plasmas 26 (2019) 052507. doi:10.1063/1.5086998.
URL https://doi.org/10.1063/1.5086998

[41] L. Chacón, D. A. Knoll, J. M. Finn, An Implicit, Nonlinear Reduced Resistive MHD Solver, J. Comput. Phys. (2002) 15–36doi:10.1006/jcph.2002.7015.
URL https://doi.org/10.1006/jcph.2002.7015

[42] L. Chacón, An optimal, parallel, fully implicit newton–krylov solver for three-dimensional viscoresistive magnetohydrodynamics, Physics of Plasmas 15 (2008) 056103. doi:10.1063/1.2838244.
URL https://doi.org/10.1063/1.2838244

[43] D. Hysom, A. Pothen, A scalable parallel algorithm for incomplete factor preconditioning, SIAM J. Sci. Comput. 22 (6) (2001) 2194–2215.

[44] R. D. Falgout, J. E. Jones, U. M. Yang, The design and implementation of hypre, a library of parallel high performance preconditioners, in: A. M. Bruaset, A. Tveito (Eds.), Numerical Solution of Partial Differential Equations on Parallel Computers, Springer–Verlag, 2006, p. 267–294, also available as LLNL technical report UCRL-JRNL-205459.

[45] S. Balay, S. Abhyankar, M. F. Adams, S. Benson, J. Brown, P. Brune, K. Buschelman, E. Constantinescu, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, V. Hapla, T. Isaac, P. Jolivet, D. Karpeev, D. Kaushik, M. G. Knepley, F. Kong, S. Kruger, D. A. May, L. C. McInnes, R. T. Mills, L. Mitchell, T. Munson, J. E. Roman, K. Rupp, P. Sanan, J. Sarich, B. F. Smith, S. Zampini, H. Zhang, H. Zhang, J. Zhang, PETSc/TAO users manual, Tech. Rep. ANL-21/39 - Revision 3.16, Argonne National Laboratory (2021).

[46] J. Omotani, et al., Hypnotoad grid generator, https://github.com/boutproject/hypnotoad (2019-2021).

[47] W. Dekeyser, X. Bonnin, S. W. Lisgo, R. A. Pitts, B. LaBombard, Solps-iter and implications for alcator c-mod divertor plasma simulations, J. Nucl. Materials 18 (2019) 125–130. doi:10.1016/j.nme.2018.12.016.
URL https://doi.org/10.1016/j.nme.2018.12.016

[48] J. Y. Murthy, Numerical methods in heat, mass and momentum transfer, lecture notes ME608, 2002, Purdue University.

[49] A. Harten, P. D. Lax, B. van Leer, On upstream differencing and Godunov-type schemes for hyperbolic conservation laws, SIAM Rev. 25 (1983) 35. doi:10.1137/1025002.
URL https://doi.org/10.1137/1025002

[50] R. Donat, A. Marquina, Capturing Shock Reflections: An Improved Flux Formula, J. Comput. Phys. 125 (1996) 42–58. doi:10.1006/jcph.1996.0078.
URL https://doi.org/10.1006/jcph.1996.0078

[51] B. Dudson, et al., SD1D SOL and Divertor model in 1D, https://github.com/boutproject/SD1D.

[52] B. Van Leer, Towards the ultimate conservative difference scheme III. Upstream-centered finite-difference schemes for ideal compressible flow,

J. Comput. Phys. 23 (3) (1977) 263–275. doi:10.1016/0021-9991(77)90094-8.
URL https://doi.org/10.1016/0021-9991(77)90094-8

[53] F. Riva, P. Ricci, F. D. Halpern, S. Jolliet, J. Loizu, A. Mosetto, Physics of Plasmas 21 (2014) 062301.

[54] B. D. Dudson, J. Madsen, J. Omotani, P. Hill, L. Easy, M. Loiten, Physics of Plasmas 23 (6) (2015) 062303. doi:10.1063/1.4953429, [link].
URL http://doi.org/10.1063/1.4953429

[55] Ben Dudson and others, Hermes-3, https://github.com/bendudson/hermes-3.

[56] P. Ghendrih, K. Bodi, H. Bufferand, G. Chiavassa, G. Ciraolo, N. Fedorczak, L. Isoardi, A. Paredes, Y. Sarazin, E. Serre, Plasma Phys. Control. Fusion 53 (5) (2011) 054019. doi:10.1088/0741-3335/53/5/054019.

[57] G. A. Sod, A Survey of Several Finite Difference Methods for Systems of Nonlinear Hyperbolic Conservation Laws, J. Comput. Phys. 27 (1) (1978) 1–31. doi:10.1016/0021-9991(78)90023-2.
URL https://doi.org/10.1016/0021-9991(78)90023-2

[58] Eleuterio F. Toro, Riemann Solvers and Numerical Methods for Fluid Dynamics, Springer Berlin, Heidelberg, 2009. doi:10.1007/b79761.
URL https://doi.org/10.1007/b79761

[59] Ben Dudson, Hermes-3 manual, https://hermes3.readthedocs.io.

[60] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in 't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, S. J. Plimpton, LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales, Comp. Phys. Comm. 271 (2022) 10817. doi:10.1016/j.cpc.2021.108171.
URL https://doi.org/10.1016/j.cpc.2021.108171

[61] M. Caini, EnTT: Gaming meets modern C++, https://github.com/skypjack/entt (2017-2021).

[62] INRIA, StarPU - A Unified Runtime System for Heterogeneous Multicore Architectures, https://starpu.gitlabpages.inria.fr/.

[63] Taskflow - A General-purpose Parallel and Heterogeneous Task Programming System, https://github.com/taskflow/taskflow.

[64] A. Kelly, Encapsulate Context, in: Proceedings of the 8th European Conference on Pattern Languages, UVK, 2003.
URL https://accu.org/journals/overload/12/63/kelly_246/

[65] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1994.

[66] R. Joshi, Data-oriented architecture: A loosely-coupled real-time soa (2007).
URL http://community.rti.com/sites/default/files/archive/Data-Oriented_Architecture.pdf

[67] O. Meneghini and S.P. Smith and L.L. Lao and O. Izacard and Q. Ren and J.M. Park and J. Candy and Z. Wang and C.J. Luna and V.A. Izzo and B.A. Grierson and P.B. Snyder and C. Holland and J. Penna and G. Lu and P. Raum and A. McCubbin and D.M. Orlov and E.A. Belli and N.M. Ferraro and R. Prater and T.H. Osborne and A.D. Turnbull and G.M. Staebler, Integrated modeling applications for tokamak experiments with OMFIT, Nucl. Fusion 55 (8) (2015) 083008. doi:10.1088/0029-5515/55/8/083008.
URL https://doi.org/10.1088/0029-5515/55/8/083008

[68] R. Hickey, A history of clojure, Proc. ACM Program. Lang. 4 (2020) 71. doi:10.1145/3386321.
URL https://doi.org/10.1145/3386321

[69] J. P. B. Puente, Persistence for the Masses: RRB-Vectors in a Systems Language, Proc. ACM Program. Lang. 1 (ICFP) (2017) 16. doi:10.1145/3110260.
URL https://doi.org/10.1145/3110260

[70] S. Mijin, D. Power, R. Holden, W. Hornsby, D. Moulton, F. Militello, Remkit1d – a framework for building reactive multi-fluid models of the tokamak scrape-off layer with coupled electron kinetics in 1d (2023). arXiv:2307.15458.

[71] S. Nakazawa, et al., Plasma Phys. Control. Fusion 42 (2000) 401. doi:10.1088/0741-3335/42/4/303, [link].
URL https://doi.org/10.1088/0741-3335/42/4/303

[72] R. Goswami, et al., Physics of Plasmas 8 (2001) 857. doi:10.1063/1.1342028, [link].
URL https://doi.org/10.1063/1.1342028

[73] M. Nakamura, et al., J. Plasma Fusion Res. 6 (2011) 2403098.

[74] S. Togo, et al., J. Plasma Fusion Res. 8 (2013) 2403096.

[75] E. Havlíčková, et al., Plasma Phys. Control. Fusion 55 (2013) 065004.

[76] G. L. Derks, J. P. K. W. Frankemölle, J. T. W. Koenders, M. van Berkel, H. Reimerdes, M. Wensing, E. Westerhof, Benchmark of a self-consistent dynamic 1D divertor model DIV1D using the 2D SOLPS-ITER code, Plasma Phys. Control. Fusion 64 (12) (2022) 125013. doi:10.1088/1361-6587/ac9dbd.
URL https://dx.doi.org/10.1088/1361-6587/ac9dbd

[77] D. Tskhakaya, S. Kuhn, Boundary conditions for the multi-ion magnetized plasma-wall transition, J. Nucl. Materials 337-339 (2005) 405–409. doi:10.1016/j.jnucmat.2004.10.073.
URL https://doi.org/10.1016/j.jnucmat.2004.10.073

[78] P. C. Stangeby, The Bohm–Chodura plasma sheath criterion, Physics of Plasmas 2 (1995) 702. doi:10.1063/1.871483.
URL https://doi.org/10.1063/1.871483

[79] D. Reiter, et al., The data file AMJUEL: Additional Atomic and Molecular Data for EIRENE, Tech. rep., Forschungszentrum Julich GmbH (2011).

[80] H. P. Summers, The ADAS user manual, version 2.6 (2004).
URL http://www.adas.ac.uk

[81] T. Body, et al., atomic++, https://github.com/TBody/atomicpp (2021).

[82] V. M. Zhdanov, Transport processes in multicomponent plasma, Taylor and Francis, London, 2002.

[83] M. Raghunathan, Y. Marandet, H. Bufferand, G. Ciraolo, P. Ghendrih, P. Tamain, E. Serre, Multi-temperature generalized Zhdanov closure for scrape-off layer/edge applications, Plasma Phys. Control. Fusion (2021). doi:10.1088/1361-6587/ac414d.
URL http://iopscience.iop.org/article/10.1088/1361-6587/ac414d

[84] K. Hromasová, D. Coster, M. Komm, J. Seidl, D. Tskhakaya, SOLPS-ITER simulations of the COMPASS tokamak, in: 47th EPS Conference on Plasma Physics, 2021, p. P5.1028.

[85] K. Hromasová, SOLPS-ITER simulations of the COMPASS tokamak, Ph.D. thesis, Czech Technical University in Prague (2021).

[86] A. J. Cerfon, J. P. Freidberg, "One size fits all" analytic solutions to the Grad–Shafranov equation, Phys. Plasmas 17 (2010) 032502. doi:10.1063/1.3328818.
URL https://doi.org/10.1063/1.3328818

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

[88] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, SciPy 1.0 Contributors, SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python, Nature Methods 17 (2020) 261–272. doi:10.1038/s41592-019-0686-2.

[89] S. Wiesen, D. Reiter, V. Kotov, M. Baelmans, W. Dekeyser, A. Kukushkin, S. Lisgo, R. Pitts, V. Rozhansky, G. Saibene, I. Veselova, S. Voskoboynikov, The new solps-iter code package, Journal of Nuclear Materials 463 (2015) 480–484, pLASMA-SURFACE INTERACTIONS 21. doi:https://doi.org/10.1016/j.jnucmat.2014.10.012.
URL https://www.sciencedirect.com/science/article/pii/S0022311514006965

[90] J. T. Omotani, F. Militello, L. Easy, N. R. Walkden, The effects of shape and amplitude on the velocity of scrape-off layer filaments 58 (1) (2015) 014030. doi:10.1088/0741-3335/58/1/014030.
URL https://doi.org/10.1088/0741-3335/58/1/014030

[91] B. Dudson, M. Kryjak, H. Muhammed, J. Omotani, Turbulence and transport in the edge of magnetically confined plasmas, 19^{th} international workshop on Plasma Edge Theory in Fusion Devices (2023).

[92] R. Li, B. Sjögreen, U. M. Yang, A new class of amg interpolation methods based on matrix-matrix multiplications, SIAM Journal on Scientific Computing 43 (5) (2021) S540–S564. arXiv:https://doi.org/10.1137/20M134931X, doi:10.1137/20M134931X.
URL https://doi.org/10.1137/20M134931X

[93] F. L. Hinton, Collisional transport in plasma, in: Basic Plasma Physics: Selected Chapters, Handbook of Plasma Physics, Volume 1, 1984, p. 147.

[94] Wikipedia: Kinetic diameter, https://en.wikipedia.org/wiki/Kinetic_diameter (2021).

[95] J. D. Huba, NRL PLASMA FORMULARY Supported by The Office of Naval Research, Naval Research Laboratory, Washington, DC, 2019.
URL http://wwwppd.nrl.navy.mil/nrlformulary/

Appendix A. Numerical method for parallel dynamics

Dynamics parallel to the magnetic field are solved using a 2^{nd}-order slope-limiter method, briefly described in section 3.2. For any number of fluids we solve the number density n, momentum along the magnetic field, $mnv_{||}$, and either pressure p or energy $\mathcal{E}$. Here m is the particle mass, so that mn is the mass density. $v_{||}$ is the component of the flow velocity in the direction of the magnetic field, and is aligned with one of the mesh coordinate directions. All quantities are cell centered.

Cell edge values are by default reconstructed using a MinMod method (other limiters are available, including 1st-order upwind, Monotonized Central, and Superbee). If f_i is the value of field f at the center of cell i, then using MinMod slope limiter the gradient g inside the cell is:

$$g_i = \begin{cases} 0 & \text{if } (f_{i+1} - f_i)(f_i - f_{i-1}) < 0 \\ f_{i+1} - f_i & \text{if } |f_{i+1} - f_i| < |f_i - f_{i-1}| \\ f_i - f_{i-1} & \text{Otherwise} \end{cases} \tag{A.1}$$

The values at the left and right of cell i are:

$$\begin{aligned} f_{i,R} &= f_i + g_i/2 \\ f_{i,L} &= f_i - g_i/2 \end{aligned} \tag{A.2}$$

This same reconstruction is performed for n, $v_{||}$ and p (or $\mathcal{E}$). The flux $\Gamma_{i+1/2}$ between cell i and $i+1$ is:

$$\Gamma_{f,i+1/2} = \frac{1}{2}\left(f_{i,R} v_{||i,R} + f_{i+1,L} v_{||i+1,L}\right) + \frac{a_{max,i+1/2}}{2}\left(f_{i,R} - f_{i+1,L}\right) \tag{A.3}$$

This includes a Lax flux term that penalises jumps across cell edges, and depends on the maximum local wave speed, a_{max}. Momentum is not reconstructed at cell edges; Instead the momentum flux is calculated from the cell edge densities and velocities:

$$\Gamma_{nv,i+1/2} = \frac{1}{2}\left(n_{i,R} v_{||i,R}^2 + n_{i+1,L} v_{||i+1,L}^2\right) + \frac{a_{max,i+1/2}}{2}\left(n_{i,R} v_{||i,R} - n_{i+1,L} v_{||i+1,L}\right) \tag{A.4}$$

The wave speeds, and so a_{max}, depend on the model being solved, so can be customised to e.g include or exclude Alfvén waves or electron thermal

speed. For simple neutral fluid simulations it is:

$$a_{max,i+1/2} = \max\left(\left|v_{||i}\right|, \left|v_{||i+1}\right|, \sqrt{\frac{\gamma p_i}{m n_i}}, \sqrt{\frac{\gamma p_{i+1}}{m n_{i+1}}}\right) \tag{A.5}$$

The divergence of the flux, and so the rate of change of f in cell i, depends on the cell area perpendicular to the flow, A_i, and cell volume V_i:

$$\nabla \cdot \left(\mathbf{b} f v_{||}\right)_i = \frac{1}{V_i}\left[\frac{A_i + A_{i+1}}{2}\Gamma_{f,i+1/2} - \frac{A_{i-1} + A_i}{2}\Gamma_{f,i-1/2}\right] \tag{A.6}$$

Appendix A.1. Boundaries

At boundaries along the magnetic field the flow of particles and energy are set by e.g. Bohm sheath boundary conditions or no-flow conditions. To ensure that the flux of particles is consistent with the boundary condition imposed at cell boundaries, fluxes of density n and also p or $\mathcal{E}$ are set to the simple mid-point flux:

$$\Gamma^{boundary}_{f,i+1/2} = f_{i+1/2} v_{||i+1/2} \tag{A.7}$$

where $f_{i+1/2} = \frac{1}{2}\left(f_i + f_{i+1}\right)$ and $v_{||i+1/2} = \frac{1}{2}\left(v_{||i} + v_{||i+1}\right)$ are the mid-point averages where boundary conditions are imposed. It has been found necessary to include dissipation in the momentum flux at the boundary, to suppress numerical overshoots due to the narrow boundary layers that can form:

$$\Gamma^{boundary}_{nv,i+1/2} = n_{i,R} v_{||i,R} v_{||i+1/2} + a_{max,i+1/2}\left[n_{i,R} v_{||i,R} - n_{i+1/2} v_{||i+1/2}\right] \tag{A.8}$$

where $n_{i+1/2} = \frac{1}{2}\left(n_i + n_{i+1}\right)$.

Appendix B. Plasma and neutral atom transport equations

The equations solved in 1D in section 5.1 and in 2D in section 5.2 are detailed here for completeness. A total of seven spatially varying quantities are evolved in time: The deuterium ion and atom densities (n_{d+} and n_d); the flow of ions and atoms parallel to the magnetic field ($v_{||,d+}$ and $v_{||,d}$); and the pressure of the ions, atoms, and electrons (p_{d+}, p_d and p_e). SI units are used except temperatures, which are in eV. In a 1D domain (section 5.1) the anomalous diffusion terms are omitted and all derivatives perpendicular to the magnetic field are assumed to be zero.

The equations for the deuterium ion species density n_{d+}, parallel velocity $v_{||,d+} \equiv \mathbf{b} \cdot \mathbf{v}_{d+}$ and pressure $p_{d+} = e n_{d+} T_{d+}$ are:

$$\frac{\partial}{\partial t} n_{d+} = -\nabla \cdot \left[\left(\mathbf{b} v_{||,d+} + \mathbf{v}_{\perp,d+} \right) n_{d+} \right] + R_{iz} - R_{rc} \tag{B.1a}$$

$$\begin{aligned} \frac{\partial}{\partial t} \left(m_{d+} n_{d+} v_{||,d+} \right) = & -\nabla \cdot \left[\left(\mathbf{b} v_{||,d+} + \mathbf{v}_{\perp,d+} \right) m_{d+} n_{d+} v_{||,d+} \right] - \mathbf{b} \cdot \nabla p_{d+} \\ & + e n_{d+} E_{||} + R_{cx} m_d \left(v_{||,d} - v_{||,d+} \right) \\ & - R_{rc} m_d v_{||,d+} + R_{iz} m_d v_{||,d} \end{aligned} \tag{B.1b}$$

$$\begin{aligned} \frac{\partial}{\partial t} \left(\frac{3}{2} p_{d+} \right) = & -\nabla \cdot \left[\left(\mathbf{b} v_{||,d+} + \mathbf{v}_{\perp,d+} \right) \frac{5}{2} p_{d+} \right] + v_{||,d+} \mathbf{b} \cdot \nabla p_{d+} \\ & + \nabla \left(\mathbf{b} \kappa_{||,d+} \mathbf{b} \cdot \nabla T_{d+} \right) + \nabla \cdot \left(\chi_{d+} n_{d+} \nabla_\perp T_{d+} \right) \\ & + \frac{1}{2} m_d \left(R_{cx} + R_{iz} \right) \left(v_{||,d} - v_{||,d+} \right)^2 \\ & + R_{iz} \frac{3}{2} e T_d - R_{rc} \frac{3}{2} e T_{d+} + W_{d+,e} \end{aligned} \tag{B.1c}$$

where $\mathbf{b} = \mathbf{B}/B$ is the unit vector in the direction of the magnetic field, and the gradient in the plane perpendicular to the magnetic field is $\nabla_\perp \equiv \nabla - \mathbf{b}\mathbf{b} \cdot \nabla$. Particle diffusion across the magnetic field is implemented as a cross-field ion drift velocity $\mathbf{v}_{\perp,d+}$ with diffusion coefficient D:

$$\mathbf{v}_{\perp,d+} = -D \frac{1}{n_{d+}} \nabla_\perp n_{d+} \tag{B.2}$$

The charge exchange (CX), ionization (IZ) and recombination (RC) reactions between species have rates (events per m^3 per second):

$$R_{cx} = n_{d+} n_d \left\langle \sigma v \right\rangle_{cx} \tag{B.3a}$$

$$R_{iz} = n_e n_d \left\langle \sigma v \right\rangle_{iz} \tag{B.3b}$$

$$R_{rc} = n_e n_{d+} \left\langle \sigma v \right\rangle_{rc} \tag{B.3c}$$

where the Maxwellian-averaged cross sections $\langle \sigma v \rangle$ are taken from Amjuel [79], specifically Amjuel reaction H.4 2.1.5 (ionisation), H.4 2.1.8 (recombination) and H.3 3.1.8 (charge exchange). Hydrogenic charge-exchange reactions are adjusted for isotope mass, ion and neutral temperatures by calculating an effective temperature $T_{eff} = T_{atom}/A_{atom} + T_{ion}/A_{ion}$ as described in the Amjuel manual.

There is a transfer of thermal energy to ions from electrons due to collisions, $W_{d+,e}$:

$$W_{d+,e} = 3\nu_{d+,e} n_{d+} \frac{m_{d+}}{m_{d+} + m_e} e \left(T_e - T_{d+}\right) \tag{B.4}$$

with ion-electron collision frequency $\nu_{d+,e} = \nu_{e,d+} m_e / m_{d+}$.

The electron density $n_e = n_{d+}$ is set by quasineutrality; the electron parallel velocity $v_{||,e} = v_{||,d+}$ from assuming that the parallel current is zero (Note that this is a choice in this particular model, not a general feature of Hermes-3). The electron pressure equation is:

$$\begin{aligned}\frac{\partial}{\partial t}\left(\frac{3}{2}p_e\right) = &- \nabla\cdot\left[\left(\mathbf{b}v_{||,e} + \mathbf{v}_{\perp,d+}\right)\frac{5}{2}p_e\right] + v_{||,e}\mathbf{b}\cdot\nabla p_e \\ &+ \nabla\left(\mathbf{b}\kappa_{||,e}\mathbf{b}\cdot\nabla T_e\right) + \nabla\cdot\left(\chi_e e n_e \nabla_\perp T_e\right) \\ &- E_{iz} + E_{rc} - W_{d+,e}\end{aligned} \tag{B.5a}$$

where E_{iz} and E_{rc} are the energy cost and gain due to ionization and recombination atomic processes respectively. These are calculated using Amjuel [79], reactions 2.1.5 and 2.1.8. Ionization always removes energy from the electrons; Recombination may be either a source or sink of electron energy, depending on the temperature and density. Electron force balance is used to calculate the parallel electric field $E_{||} \equiv \mathbf{b}\cdot\mathbf{E}$ and so transfer electron pressure p_e forces to the ions:

$$e n_e E_{||} = -\mathbf{b}\cdot\nabla p_e - \nabla\cdot\left[\mathbf{v}_{\perp,d+} m_e n_e v_{||,e}\right] \tag{B.6}$$

The equations for the neutral deuterium atom density n_d, parallel velocity

$v_{||,d}$ and pressure $p_d = en_dT_d$ are:

$$\frac{\partial}{\partial t} n_d = -\nabla \cdot \left[\left(\mathbf{b} v_{||,d} + \mathbf{v}_{\perp,d} \right) n_d \right] - R_{iz} + R_{rc} \tag{B.7a}$$

$$\begin{aligned} \frac{\partial}{\partial t} \left(m_d n_d v_{||,d} \right) = & -\nabla \cdot \left[\left(\mathbf{b} v_{||,d} + \mathbf{v}_{\perp,d} \right) m_d n_d v_{||,d} \right] - \mathbf{b} \cdot \nabla p_d \\ & - R_{cx} m_d \left(v_{||,d} - v_{||,d+} \right) + R_{rc} m_d v_{||,d+} \\ & - R_{iz} m_d v_{||,d} \end{aligned} \tag{B.7b}$$

$$\begin{aligned} \frac{\partial}{\partial t} \left(\frac{3}{2} p_d \right) = & -\nabla \cdot \left[\left(\mathbf{b} v_{||,d} + \mathbf{v}_{\perp,d} \right) \frac{5}{2} p_d \right] + v_{||,d} \mathbf{b} \cdot \nabla p_d \\ & + \nabla \left(\kappa_d \nabla T_d \right) + \frac{1}{2} m_d \left(R_{cx} + R_{rc} \right) \left(v_{||,d} - v_{||,d+} \right)^2 \\ & - R_{iz} \frac{3}{2} e T_d + R_{rc} \frac{3}{2} e T_{d+} \end{aligned} \tag{B.7c}$$

The flow of neutral atoms across the magnetic field, $\mathbf{v}_{\perp,d}$, is derived by balancing friction forces against pressure gradient [2]:

$$\mathbf{v}_{\perp,d} = -\frac{T_d}{m_d \nu_d p_d} \nabla_\perp p_d \tag{B.8}$$

The thermal conduction coefficients for each species are:

$$\kappa_{||,d+} = 3.9 \frac{p_{d+}}{m_{d+} \nu_{d+}} \qquad \kappa_{||,e} = 3.16 \frac{p_e}{m_e \nu_e} \qquad \kappa_d = \frac{p_d}{m_d \nu_d} \tag{B.9}$$

The collision frequencies for each species, ν_{d+}, ν_e and ν_d, are:

$$\nu_{d+} = \nu_{d+,d+} + \frac{m_e}{m_{d+}} \nu_{e,d+} + n_d \left\langle \sigma v \right\rangle_{cx} \tag{B.10a}$$

$$\nu_e = \nu_{e,d+} + \nu_{e,e} \tag{B.10b}$$

$$\nu_d = n_{d+} \left\langle \sigma v \right\rangle_{cx} + n_d a_0 \sqrt{2eT_d / m_d} \tag{B.10c}$$

The collision frequency of charged species a on charged species b is given by [93]:

$$\nu_{a,b} = \frac{q_a^2 q_b^2 n_b \ln \Lambda \left(1 + m_a / m_b \right)}{3\pi^{3/2} \epsilon_0^2 m_a^2 \left(v_a^2 + v_b^2 \right)^{3/2}} \tag{B.11}$$

with $v_a^2 = 2T_a / m_a$. Neutral-neutral collisions assume a kinetic diameter of

2.8×10^{-10}m (cross-section $a_0 = 2.5 \times 10^{-19}\text{m}^2$), chosen based on typical values for species considered (2.89×10^{-10}m for H_2, 2.60×10^{-10}m for He, 2.75×10^{-10}m for Ne [94]).

The Coulomb logarithm is different for electron-electron, ion-ion and electron-ion species interactions, and is calculated using the NRL formulary [95] (page 34). Converted to SI units with T in eV the Coulomb logarithms are:

$$\log \Lambda_{e,e} = 30.4 - 0.5 \log n_e + \frac{5}{4} \log T_e - \sqrt{\epsilon + (\log T_e - 2)^2 / 16} \tag{B.12a}$$

$$\log \Lambda_{e,i} = \begin{cases} 31 - 0.5 \log n_e + \log T_e & \text{if } T_i \frac{m_e}{m_i} < 10 Z_i^2 < T_e \\ 30 - 0.5 \log n_e - \log Z_i + 1.5 \log T_e & \text{if } T_i \frac{m_e}{m_i} < T_e < 10 Z_i^2 \\ 23 - 0.5 \log n_i + 1.5 \log T_i - \log (Z_i^2 A_i) & \text{if } T_e < T_i m_e / m_i \end{cases} \tag{B.12b}$$

$$\log \Lambda_{i,i} = 29.91 - \log \left[\frac{Z_1 Z_2 (A_1 + A_2)}{A_1 T_2 + A_2 T_1} \sqrt{n_1 Z_1^2 / T_1 + n_2 Z_2^2 / T_2} \right] \tag{B.12c}$$

Note that earlier versions of the NRL formulary had a factor of A_i^{-1} rather than A_i in equation B.12b.