Report 2047356-TN-17 (D5.3): Progress on implementation of system 2-6 equations

David Moxey, King's College London
Ben Dudson, Peter Hill, Ed Higgins, David Dickinson & Steven Wright, University of York

15th December 2022

Contents

1 Executive summary

This report briefly outlines the initial efforts on implementing the system 2-6 equation set within the *Nektar++* framework. In particular, we discuss changes made to the equation set since the previous deliverable, and outline a prototype solver for a simplified set of equations, as well as the future development pathway for system 2-6.

2 Introduction

In the initial phase of the NEPTUNE project, the goal of the team at KCL and ICL has been to enable the production of a prototype code, based on the *Nektar++* framework, for a set of spatially 2D plasma model incorporating velocity space effects. As outlined in previous deliverables, and as taken from the initial equations document supplied at the beginning of NEPTUNE [1], these were formulated in the system 2-6 equations.

As discussed in the previous report [2], this system was highly complex involving not only a large number of parameters, variables and equations, but a large amount of domain-specific knowledge in terms of the underlying physics. In addition the following issues were identified:

- the lack of an existing code for these equations, which does not provide an easy route to validation of the code;
- the equations in their current form may be challenging to discretise under the typical high-order approaches, and some aspects are unclear (e.g. numerical flux terms);

- there seemed to be limited opportunity to produce simplified versions of this equation set to naturally 'build-up' the complexity of the solver and test numerical approaches in a more constrained fashion.

As part of discussions with UKAEA and the NEPTUNE partners, we have therefore aimed to define a new equation set which can address at least a subset of these issues. Noting the development of Hermes [3] at the University of York, these discussions led to an adjustment in the latest equations document [4], considering a system modelled as a modified version of those found in the latest Hermes3 code [5]:

$$\frac{\partial n_e}{\partial t} = -\nabla \cdot (n_e \mathbf{v}_{E\times B}) + \nabla \cdot \frac{1}{|q_e|}\mathbf{j}_{\text{sh}} + \frac{n_e c_s}{L_\parallel} + S_e^n \tag{1}$$

$$\frac{\partial p_e}{\partial t} = -\nabla \cdot (p_e \mathbf{v}_{E\times B}) - \frac{\gamma_e p_e c_s}{L_\parallel} + S_e^p + D_{fpe}\nabla \cdot (\kappa_{e\perp} n_e \nabla_\perp kT_e) \tag{2}$$

$$\frac{\partial p_i}{\partial t} = -\nabla \cdot (p_i \mathbf{v}_{E\times B}) - \frac{\gamma_e p_i c_s}{L_\parallel} + S_i^p + D_{fpi}\nabla \cdot (\kappa_{i\perp} n_i \nabla_\perp kT_i) \tag{3}$$

$$\frac{\partial \omega}{\partial t} = -\nabla \cdot (\omega \mathbf{v}_{E\times B}) + \nabla \cdot \left[(p_e + p_i)\nabla \times \frac{\mathbf{b}}{B}\right] + \nabla \cdot \mathbf{j}_{\text{sh}} + D_{fvs}\nabla \cdot \nu \nabla_\perp \omega \tag{4}$$

where

$$\begin{aligned}
p &= \sum_\alpha n_\alpha kT_\alpha \\
\rho_m &= \sum_\alpha A_\alpha m_u n_\alpha \\
c_s &= \sqrt{\frac{p}{\rho_m}} \\
n_e &= \sum_{\alpha \neq e} Z_\alpha n_\alpha = Z_i n_i \\
\omega &= \nabla \cdot \left[\frac{m_i}{Z_i |q_e| B^2}\nabla_\perp \left(N_{\text{ref}}|q_e|\Phi + \frac{1}{Z_i}p_i\right)\right] \\
\nabla \cdot \mathbf{j}_{\text{sh}} &= -\frac{|q_e| n_e c_s}{L_\parallel}\frac{|q_e|\Phi}{kT_{\text{ref}}}
\end{aligned}$$

The terms above are left undefined for brevity, but can be found in the latest equations document [4].

This set of equations provides a more flexible approach to developing a solver:

- there is a known reference code with documented examples;
- the terms above are more amenable to discretisation and numerical fluxes are more readily defined;
- moreover there is a tranche of equations of increasing complexity that can be used as a basis for development.

As a pathway to development, we therefore plan to first implement simpler models with the numerical hallmarks of the system above: namely, the Blob2D examples, starting from a single species isothermal model, and then moving onto versions incorporating multiple species and full turbulence case. These are found on the Hermes3 website [6]. The rest of this document therefore sets out initial progress in the implementation of the first (and simplest) of these examples.

3 Initial solver for Blob2D example

As a first step, we have implemented a sample solver for the isothermal transport of a seeded plasma filament. This is governed by the equation set

$$\frac{\partial n_e}{\partial t} = -\nabla \cdot (n_e \mathbf{v}_{E\times B}) + \frac{1}{e}\nabla \cdot \mathbf{j}_{\text{sh}}, \tag{5}$$

$$\frac{\partial \omega}{\partial t} = -\nabla \cdot (\omega \mathbf{v}_{E\times B}) + \nabla \cdot \left(p_e \nabla \times \frac{\mathbf{b}}{B}\right) + \nabla \cdot \mathbf{j}_{\text{sh}}, \tag{6}$$

$$\frac{1}{B^2}\nabla^2 \phi = \omega, \tag{7}$$

where

- $p_e = en_eT_e$ is the pressure;
- $\nabla \cdot \mathbf{j}_{\text{sh}} = n_e\phi/L_{\|}$ is the sheath closure

In addition, we look to implement the diamagnetic drift term as

$$\nabla \cdot \left(p_e \nabla \times \frac{\mathbf{b}}{B}\right) = \frac{eT_e}{R^2}\frac{\partial n_e}{\partial y},$$

where $\nabla \times (\mathbf{b}/B) = (0, \frac{1}{R^2})$ and R is a constant. We use the constants:

- $e = -1$ is the electron charge;
- $B = 0.35\,\text{T}$;
- T_e is a fixed electron temperature ($5\,\text{eV}$);
- $L_{\|} = 10\,\text{m}$ is the connection length;
- $R = 1.5\,\text{m}$

3.1 Discretisation strategy

Functionally these equations are similar in form to the Hasegawa-Wakatani equations, for which a solver based on *Nektar++* already exists [7]. We therefore adopt the following procedure, using an explicit timestepping scheme:

- Compute the potential ϕ by solving equation (7) in a continuous discretisation.
- Use this to compute the drift velocity $\mathbf{v}_{E\times B} = B^{-1}(\partial_y\phi, -\partial_x\phi)$.
- Evaluate the terms $-\nabla \cdot (n_e\mathbf{v}_{E\times B})$ and $-\nabla \cdot (\omega\mathbf{v}_{E\times B})$ using a DG discretisation.
- Finally evaluate all other source terms.

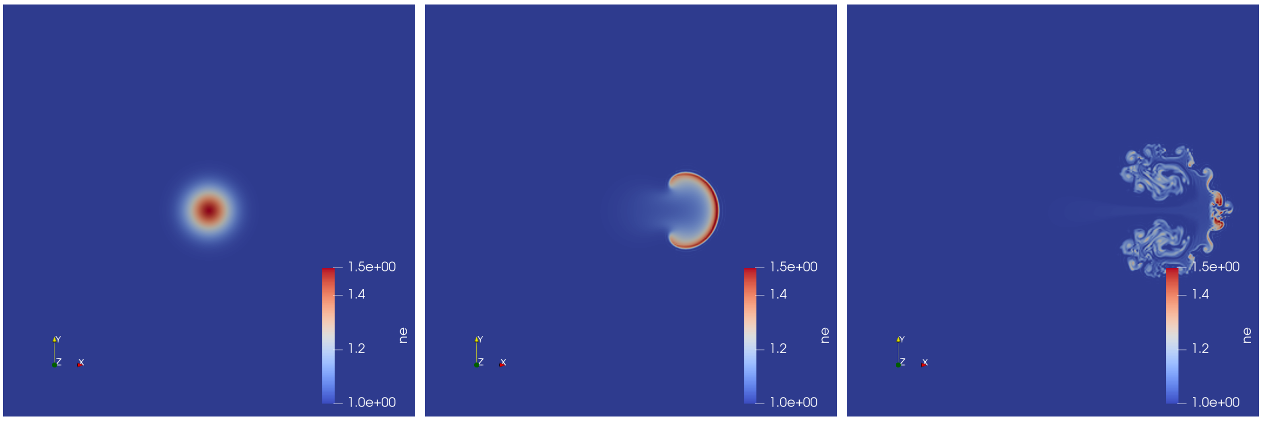
This solver is encapsulated in the `nektar-driftplane` solver [8]

3.2 Test case

Following the test case[1], we set up a basic simulation of a filament, using the following simulation parameters:

- The domain is taken as $\Omega = [-0.5, 0.5]^2$, with periodic boundary conditions used on all sides.

[1] Test case found here: `https://github.com/bendudson/hermes-3/tree/master/examples/blob2d`



(a) $t = 0$ (b) $t = 2$ (c) $t = 4$

Figure 1: Example of Blob2D simulation with snapshots of number density n_e at three time intervals.

- A mesh of quadrilateral elements at order 5 is used to discretise Ω.
- Fourth order Runge-Kutta timestepping is used with $\Delta t = 2 \times 10^{-4}$.
- The various parameter values above are passed into the simulation in the session file.
- As initial condition we use $\omega = 0$ and set

$$n_e(x, y, 0) = 1 + h \exp\left(-\frac{x^2 + y^2}{w^2}\right)$$

  where $h = 0.5$ and $w = 0.05$ as in the Hermes3 example.

The simulation is encapsulated in the `examples/blob2d` directory of the drift plane solver, and can be run either in serial or in parallel, using e.g. `mpirun -n 28 DriftPlaneSolver driftwave.xml square.xml`. A sample of the output as the solution evolves in time can be seen in figure 1. Visually, there is a clear similarity between the Hermes-3 output: however there are a few differences that require additional investigation:

- Hermes-3 uses a mixture of periodic and Neumann boundary conditions;
- The Hermes-3 test case uses a Gaussian centred initially at $x = -0.25$, rather than $x = 0$ which is investigated here;
- the coefficient in front of the dn/dy term of eq. (6) is difficult to determine precisely from Hermes-3 and may explain some of the visual differences.

4 Conclusions and outlook

This deliverable has briefly outlined the likely development route for the revised system 2-6 equations. There is a relatively clear development pathway, in combination with partners at UKAEA and University of York.

- Consider the addition of artificial diffusion terms to the Blob2D solver, akin to the diffusion noted in eqs. (2), (3) and (4). Use of an explicit-in-time diffusion term is likely to add increasingly severe timestep restrictions. Therefore it may be beneficial to consider a semi-implicit approach for the addition of diffusion.

- Implementation of the Hermes-3 Blob2D-T_e-T_i equations, and the turbulent version of these.

- Finally, extension to the full system 2-6 equations.

At each step, numerical performance can be examined in detail and compared against the equivalent Hermes-3 solvers, up to the implementation of the full system 2-6.

References

[1] Wayne Arter. Equations for NEPTUNE Proxyapps (Version 1.00). Technical report, UK Atomic Energy Authority, 2020.

[2] D. Moxey, B. Dudson, P. Hill, E. Higgins, D. Dickinson, and S. Wright. Report 2047356-tn-13 (d5.1 + d5.2): Spatially 2-d plasma model incorporating velocity space effects. Technical report, King's College London and University of York, March 2022.

[3] Benjamin Daniel Dudson and Jarrod Leddy. Hermes: global plasma edge fluid turbulence simulations. *Plasma Physics and Controlled Fusion*, 59(5):054010, 2017.

[4] Wayne Arter. Equations for NEPTUNE Proxyapps (Version 1.23). Technical report, UK Atomic Energy Authority, 2022.

[5] Hermes plasma edge simulation model: Hermes-3, a hot ion multifluid drift-reduced model. `https://github.com/bendudson/hermes-3`. Accessed 15th December 2022.

[6] Hermes-3 examples. `https://hermes3.readthedocs.io/en/latest/examples.html`. Accessed 15th December 2022.

[7] Nektar++ drift-wave solver. `https://github.com/ExCALIBUR-NEPTUNE/nektar-driftwave`. Accessed 15th December 2022.

[8] Nektar++ drift-plane solver. `https://github.com/ExCALIBUR-NEPTUNE/nektar-driftplane`. Accessed 15th December 2022.