UK Atomic Energy Authority

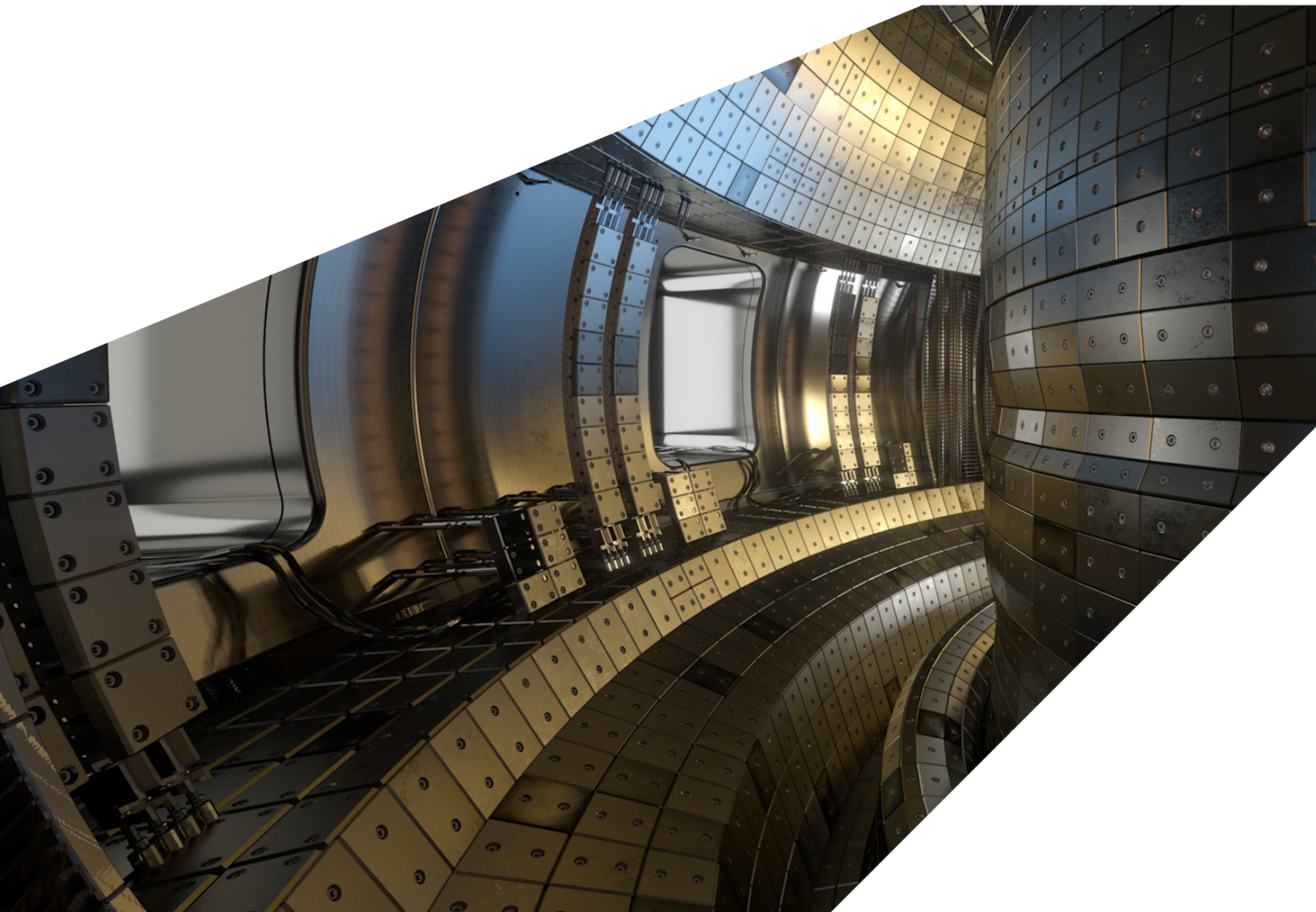
ExCALIBUR

2-D integrated particle and continuum model

M4c.2 Version 1.00

Abstract

The report describes work for ExCALIBUR project NEPTUNE at Milestone M4c.2. An integrated particle and continuum model has been developed in $2d3v$, where $2d3v$ implies that there is variation in two space dimensions (2-D) and 3 velocity-phase space coordinates. NEKTAR++ provides the 2-D finite elements to represent the plasma as a fluid, whereas 3-D velocity-phase space is populated with particles to represent the neutral particles and the complete calculation is orchestrated by the NESO-PARTICLES software developed under project NEPTUNE. Much of the new code uses SYCL to pave the way to Exascale in a performance-portable manner. Initial tests using a small range of boundary conditions and sources demonstrate successful strong coupling between the fluid and the new particle code. Additionally, summaries are provided of work by a grantee to develop a high-dimensional finite element library intended to provide a complementary approach to the use of particles.



UKAEA REFERENCE AND APPROVAL SHEET

	Client Reference:	
	UKAEA Reference:	CD/EXCALIBUR-FMS/0072
	Issue:	1.00
	Date:	March 21, 2023
Project Name: ExCALIBUR Fusion Modelling System		

	Name and Department	Signature	Date
Prepared By:	James Cook Will Saunders Owen Parry BD	N/A N/A N/A	March 21, 2023 March 21, 2023 March 21, 2023
Reviewed By:	Wayne Arter Project Technical Lead		March 21, 2023

1 Introduction

The reach of this milestone is to create a proxyapp that combines at least one complex element from several aspects of exhaust physics. This $2d3v$ proxyapp is based on an advection-diffusion-reaction (ADR) equation system where the source terms arise from the ionisation of neutral particles which are treated as Lagrangian markers.

The system of three coupled fluid equations for number density, velocity and energy, encapsulated in a state vector $\vec{\varsigma}$ and expressed in a form familiar to the area of CFD and specifically Finite Volume formulations as

$$\frac{\partial \vec{\varsigma}}{\partial t} + \nabla \cdot (\mathbf{F}(\vec{\varsigma})) = \nabla \cdot \mathbf{G}(\vec{\varsigma}, \nabla\vec{\varsigma}) + S(\vec{\varsigma}), \tag{1}$$

where $\mathbf{F}$ represents the spatial component of a general hyperbolic conservation law, $\mathbf{G}$ denotes the diffusion terms and $S(\vec{\varsigma})$ is the source term for the components of the state vector. We provide the particular system of equations of interest in Section 2.1.1.

In this report we present details of the implementation and present initial results of an exhaust relevant ADR solver constructed from NEKTAR++ and NESO-PARTICLES(NP) and realised as part of the NESO framework. The targeted real-world scenario is the scrape-off-layer (SOL) which connects the tokamak divertor (heat sink) to the bulk plasma at the outer mid-plane edge (heat source). Furthermore the plasma is coupled to a system of neutral particles. The fluid species exists as a finite element representation provided by the NEKTAR++ library. The neutral particle species is implemented using the NP framework. Within NESO there exists a tight coupling between finite element functions within NEKTAR++ and particles in NP.

The computationally challenging dynamics are largely aligned to the magnetic field, which predicates the use of a long, thin computational domain also aligned with the magnetic field. In this domain, the plasma source is located in the middle and the upper and lower divertors are located at the ends. This proxyapp implements a system with an element of complexity from many areas of the parameter space: advection-diffusion in NEKTAR++, performance portable particles via NP, domain decomposition via MPI, mixed boundary conditions, tight coupling of fluids and particles, IO, visualisation, and crucially 2 spatial dimensions and 3 velocity dimensions. The final element of complexity is critical for providing functionality to the tokamak edge modelling community who wish to calculate the heat fluxes required for so-called mean field equations in 2-D with neutral physics via particles with all three velocity components.

The structure of the rest of the report is as follows. In Section 2 we describe the technical work done for this report and describe the equations that are solved for the fluid dynamics and particles, how they are coupled and the how the ionisation rates are obtained. We also present an exposition of the ionisation algorithm. We provide a description of the basis functions in NEKTAR++ and their efficient evaluation in NESO via SYCL. In Section 3, we summarise and give context to two associated grantee reports on this topic. The final section provides a summary of this work.

2 Task Work

2.1 Plasma, particle and ionisation equations

The plasma exhaust is modelled in 2-D in a long and thin domain, where the centre represents the outer mid-plane edge of the plasma, ie. broadly where the hot plasma enters the scrape off layer domain, and the ends represent the divertor, ie. the plasma from the centre travels along the domain in both directions towards the exhaust region. Critical to fusion power plant design is the concept of a detached plasma whereby a region of ionisation and recombination exists between the plasma and the divertor. This detached state is beneficial as the divertor is in contact with merely a hot gas rather than a plasma carrying orders of magnitudes larger heat fluxes.

The region of ionisation and heat flux near the ends of this long and thin 2-D domain is represented in this mini-app as locations of neutral particle sources (whose presence in a real tokamak would arise via gas puffs, recycling from the wall and the processes of ionisation and recombination between the neutral gas and the detached plasma). In this instance, we focus on ionisation only and leave recombination as an item of further work. The following sections describe the fluid and particle dynamical equations and the origin of the ionisation rates.

2.1.1 Dynamics equations

The plasma exhaust relevant fluid equations, in the form of an advection-diffusion-reaction (ADR) equation, are expressed in 2-D by Eulerian methods, where the source terms in the fluid equations arise from the ionisation of neutral particles. We treat these neutral particles as Lagrangian markers. We describe the time evolution of number density n, velocity $\vec{u}$ and *double* energy $E = (g-2)nT + nu^2$ as

$$U_r \frac{\partial}{\partial t} n = -\nabla \cdot (n\vec{u}) + S^n, \tag{2}$$

$$U_r \frac{\partial}{\partial t} n\vec{u} = -\nabla \cdot (n\vec{u} \otimes \vec{u} + nT\vec{I}) + S^u, \tag{3}$$

$$U_r \frac{\partial}{\partial t} (n((g-2)T + u^2)) = -\nabla \cdot (n\vec{u}(gT + u^2)) - \kappa_d \nabla^2 T + S^E, \tag{4}$$

where T and g are the plasma temperature and equation of state constant respectively and S^i is the source term for variable i arising from the ionisation of neutral particles. The neutral computational macroparticles follow straight trajectories from birth until either loss due to complete ionisation or because they have travelled past the ends of the domain in the long direction ie. hit the divertor target.

We sample initial neutral particle properties from a fixed density and temperature profile. We advect each neutral particle collisionlessly, ie. without external forces or inter-particle interaction, and at each time step perform an ionisation procedure. This ionisation procedure requires evaluation of the plasma density and temperature at the location of each particle to calculate an ionisation rate. The ionisation rate determines the proportion of the weight of each neutral particle which will be deposited onto the grid as plasma.

The deposition of newly ionised plasma takes the form of source terms in the plasma density, momentum density and energy density equation. For conservation of mass, momentum and energy, the source terms in the plasma equations must be commensurate with the loss of neutral density, momentum and kinetic energy from particles. We weakly equate particle and finite element representations of a quantity via a L2 Galerkin projection. We describe the ionisation algorithm below and describe the ionisation rate calculation the next section.

for $particle \in species$ **do**
  $x \leftarrow position(particle)$
  $w \leftarrow weight(particle)$
  $u \leftarrow velocity(particle)$
  $n \leftarrow density(x)$
  $T \leftarrow temperature(x)$
  $r \leftarrow rate(T)$
  $\Delta w \leftarrow -rnw\Delta t$
  $remove \leftarrow 0$
  if $w + \Delta w < 0$ **then**
    $\Delta w \leftarrow -w$
    $remove \leftarrow 1$
  end if
  $w \leftarrow w + \Delta w$
  $n^S \leftarrow n^S - \Delta wu$
  $u^E \leftarrow u^E - \Delta wmu^2/2$
end for

2.1.2 Ionisation rates

We use the following formula, taken from Ref. [1], as the ionisation cross section

$$\sigma \;=\; \sum_{i=1}^{N} a_i q_i \frac{ln\left(\frac{E}{E_i}\right)}{EE_i}[1 - b_i e^{-c_i\left(\frac{E}{E_i}-1\right)}] \tag{5}$$

where E is the energy of the impacting electron, E_i is the binding energy of electrons in subshell i and q_i is the number of electrons in the i-th subshell. The constants a_i, b_i and c_i are to be determined from theory or experiment. Note that for hydrogen and Helium like ions $N = 1$. In the following table we list the reference values presented in [1] for hydrogen and $i = 1$.

Parameter	Value
a_1	$4\text{x}10^{-14} cm^2(eV)^2$
b_1	0.6
c_1	0.56
E_1	13.6 eV
q_1	1

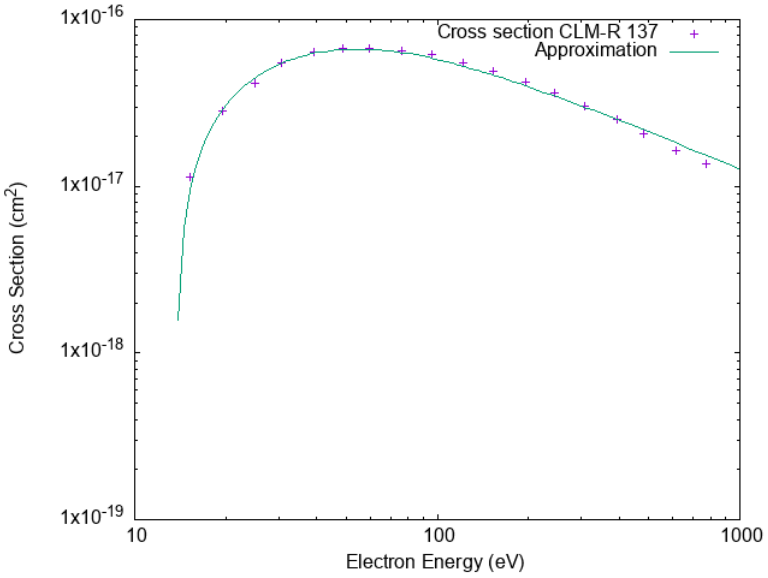


Figure 1: Cross section plotted against electron energy.

Figure 1 plots this approximation for the cross section against the data from Freeman and Jones [2]. We use the cross section applied to a Maxwellian electron distribution at temperature T to generate the ionisation rate in units of cm^3/s (see Ref. [1]),

$$\begin{aligned} S &= 6.7\text{x}10^7 \sum_{i=1}^{N} \frac{a_i q_i}{T^{\frac{3}{2}}} \left\{ \frac{T}{E_i} \int_{\frac{E_i}{T}}^{\infty} \frac{e^{-x}}{x} dx - \frac{b_i e^{c_i}}{\frac{E_i}{T} + c_i} \int_{\frac{E_i}{T}+c_i}^{\infty} \frac{e^{-y}}{y} dy \right\} \\ &= -6.7\text{x}10^7 \sum_{i=1}^{N} \frac{a_i q_i}{T^{\frac{3}{2}}} \left\{ \frac{T}{E_i} Ei\Big(- \frac{E_i}{T} \Big) - \frac{b_i e^{c_i}}{\frac{E_i}{T} + c_i} Ei\Big(- \frac{E_i}{T} - c_i \Big) \right\} \end{aligned} \tag{6}$$

where T is in eV. Figure 2 compares the approximation against ADAS provided data. We assess that the approximation is adequate for this report. Furthermore we use the closed form approximation to the exponential integral by Barry et al. [3].

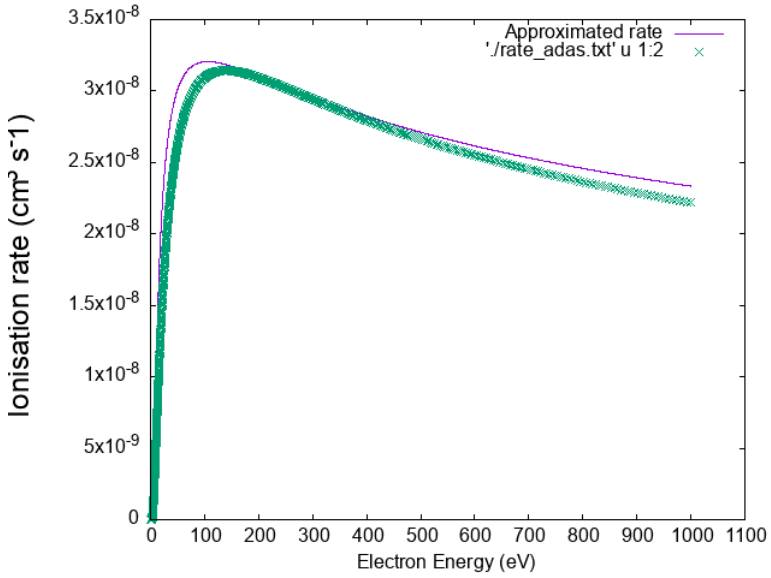


Figure 2: Comparison between ADAS data and approximation plotted against electron energy.

2.2 NEKTAR++

Although NEKTAR++ includes implementations of the prerequisite algorithms required by NESO to project particle data onto finite element functions and evaluate these functions at particle locations these methods are not efficiently implemented for the particle use case. The original implementations in NEKTAR++ are adequately implemented for use in the setup phase of a simulation where small inefficiencies are amortised by the cost of the time stepping loop or main solve.

In the particle use case, these once setup only functions are called on a per-particle basis at every time step of the simulation. This per particle and per time step regime places these functions on the critical path of the simulation where small inefficiencies in each function call contribute significantly to the overall runtime. Furthermore the implementations of these methods are provided only on the CPU host hence particle data, which is stored on a compute device such as a GPU, must be copied to and from the host to call these functions. To improve the performance and portability of NESO we provide SYCL implementations of these critical algorithms.

2.2.1 Function Evaluation

NEKTAR++ considers two representations of a given function. The first representation in the function values at the quadrature points in the domain. The second representation is the classic finite element degrees of freedom (DOFs) α_j such that

$$u(\vec{x}) = \sum_j \alpha_j \psi_j(\vec{\xi}(\vec{x})) \tag{7}$$

where u is the function represented by the α_j, ψ_j is the j^{th} basis function and $\vec{\xi}(\vec{x})$ is the position in the reference cell of the point $\vec{x}$. Hence evaluating Equation (7) at a point $\vec{x}$ requires evaluating all the individual basis functions at $\vec{x}$. We note that in the case where a function u is evaluated at all particle locations with a mesh cell there are two properties that are computationally beneficial on modern architectures; firstly the DOFs α_j are constant and identical for all evaluations in the cell secondly the functional form of the basis functions is identical for all the evaluations within the cell. The first property indicates that evaluating a function at all points within a cell as a grouped operation should improve cache reuse and hence improve the arithmetic intensity of the implementation. The second property indicates that the basis function evaluations for different points in the cell can occur in adjacent lanes in vector computing architectures such as GPUs and CPU SIMD units.

Until now we have indexed basis functions and DOFs with a single index j, in practice there exists at least n basis function indices for an n dimensional space (in NEKTAR++). In this report we consider 2-D space where the computational mesh is constructed as a collection of quadrilateral and triangular elements. Functions defined over quadrilaterals and triangles are constructed as the tensor product of a 1-D basis function in the first and second dimensions of the reference element. For these two element types there are two 1-D basis functions defined in NEKTAR++; the "modified A" and "modified B" basis which we label as $\psi^a_p(z)$ and $\psi^b_{p,q}(z)$ respectively.

For n "modes" (polynomial degree plus one) the modified A basis ψ^a_p is defined as

$$\psi^a_p(z) = \left\{ \begin{array}{ll} \frac{1}{2}(1-z) & \text{if } p = 0 \\ \frac{1}{2}(1+z) & \text{if } p = 1 \\ \frac{1}{2}(1-z)\frac{1}{2}(1+z)P^{(1,1)}_{p-2}(z) & \text{otherwise} \end{array} \right\} \tag{8}$$

for all p such that $0 \leq p < n$. In this report $P^{(\alpha,\beta)}_p(z)$ denotes the standard (α, β) Jacobi polynomials of order p. The $\psi^b_{p,q}(z)$ basis is defined over a triangle number sized iteration set $\{(p, q) : 0 \leq p < n, 0 \leq q < (n - p)\}$ and is defined as

$$\psi^b_{p,q}(z) = \left\{ \begin{array}{ll} \psi^a_q(z) & \text{if } p = 0 \\ \left(\frac{1}{2}(1+z)\right)^p & \text{if } q = 0 \\ \frac{1}{2}(1-z)\left(\frac{1}{2}(1+z)\right)^p P^{(2p-1,1)}_{q-1}(z) & \text{otherwise.} \end{array} \right\} \tag{9}$$

A scalar valued function u of n modes defined over a quadrilateral by coefficients α is evaluated at a point $\vec{x}$ as the double sum

$$u(\vec{x}) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} \alpha_{jn+i}\psi^a_i(\vec{\xi}(\vec{x})_0)\psi^a_j(\vec{\xi}(\vec{x})_1). \tag{10}$$

We evaluate Equation (10) with $\mathcal{O}(n^2)$ complexity by using the recursion relation of the Jacobi polynomials to compute and store the values $\{\psi^a_i(\vec{\xi}(\vec{x})_0)\ \forall\ i : 0 \leq i < n\}$ and $\{\psi^a_j(\vec{\xi}(\vec{x})_1)\ \forall j : 0 \leq j < n\}$ each with $\mathcal{O}(n)$ complexity. These basis function evaluations are then reduced along with the appropriate coefficient α to construct $u(\vec{x})$. This two stage approach places the code that contains multiple conditionals into the $\mathcal{O}(n)$ basis evaluation stage and allows the $\mathcal{O}(n^2)$ operation to consist purely of a double loop and a reduction. We exploit the local memory feature of SYCL to allocate temporary memory for the basis function evaluations in each dimension.

The evaluation of a function u over a triangle follows the same structure with three modifications; the modified B basis is applied in the second dimension, a correction is applied for the second mode and the reference coordinate is converted into a collapsed coordinate prior to evaluation. For triangular elements modes, and hence DOFs, are not indexed in a lexicographic manner but are arranged such that

$$\mathsf{mode}(p,q) = q + \sum_{0<i\leq p}\left(\sum_{0\leq j<(n-i)} 1\right). \tag{11}$$

Although the mode indexing appears convoluted, in actual code this results in an indexing that increments by one in each iteration of the nested double loop. The function is readily evaluated as

$$u(\vec{x}) = \sum_{i=0}^{(n-1)}\sum_{j=0}^{(n-i-1)} \alpha_{\mathsf{mode}(i,j)}\left[\delta_{1,\mathsf{mode}(i,j)} + (1-\delta_{1,\mathsf{mode}(i,j)})\psi_i^a(\vec{\eta}(\vec{x})_0)\right]\psi_{i,j}^b(\vec{\eta}(\vec{x})_1). \tag{12}$$

where $\delta_{a,b}$ is the Kronecker delta. The middle square terms involving the Kronecker delta enclosed by square brackets represents the correction applied internally by NEKTAR++ for the second mode.

The coordinate η refers to the reference coordinate in the collapsed coordinate space which is applied to triangles in NEKTAR++. This collapsed mapping is defined from the reference space $\vec{\xi}(\vec{x})$ to the collapsed space $\vec{\eta}(\vec{\xi}(\vec{x}))$. Note that in Equation (12) we wrote $\vec{\eta}(\vec{x})$ instead of $\vec{\eta}(\vec{\xi}(\vec{x}))$ for brevity of notation. The mapping to collapsed coordinates is given by

$$\vec{\eta}(\vec{\xi}) = \begin{bmatrix} 2(1+\vec{\xi}_0)/(1-\vec{\xi}_1) - 1 \\ \vec{\xi}_1 \end{bmatrix} \tag{13}$$

As in the quadrilateral case we create local arrays for the evaluations of $\psi_i^a(\vec{\eta}_0)$ and $\psi_{i,j}^b(\vec{\eta}_1)$. These basis function evaluations are then multiplied together and reduced in a double loop along with the DOFs.

2.2.2 Barycentric Interpolation

The alternative function representation in NEKTAR++ is the function values at the quadrature points across the entire mesh. These are referred to as the "physical values" and can be converted to or computed from the DOFs. NEKTAR++ provides methods to compute derivatives from these physical values and represents these derivatives again as quadrature point values (as opposed to DOFs). As converting physical values to DOFs has a non-negligible cost it is beneficial for NESO to also provide a SYCL implementation that evaluates a function defined by quadrature point values at each particle location. Hence we provide a SYCL implementation to evaluate NEKTAR++ functions via the Barycentric interpolation [4] approach. Note that this algorithm also exhibits $\mathcal{O}(n^2)$ computational complexity for n modes in 2-D but unlike the basis function evaluation approach the critical path consists of $\mathcal{O}(n^2)$ divisions.

2.2.3 Projection onto Finite Element Functions

In Section 1.3 of report M4c [5] we describe the method we apply to convert a particle representation of a field to a weakly equivalent finite element representation via an L2 Galerkin projection. This projection approach solves a linear system where the RHS is constructed by evaluating each basis function at the location of each particle.

In prior versions of NESO these basis function evaluations were performed on the CPU host and not in a SYCL parallel loop. With the SYCL implementation of the basis functions described in Section 2.2.1, NESO now contains a performance portable implementation to compute the RHS of Equation (11) in M4c [5]. The mass-matrix solve required to solve this linear system is however still performed on the CPU host.

2.2.4 The Simple-SOL solver

The SIMPLE-SOL solver was developed initially as a NEPTUNE proxyapp in its own right [6] to solve *System 2–3* from the equations document [7].

While the simple SOL problem is only formulated in 1-D, it can be treated as a pseudo-2-D problem by giving the 'field line' a finite width. This approach allows the fluid-particle coupling framework to be tested on a 2-D domain whilst retaining access to existing, relatively simple 1-D analytical solutions for validation. In the following sections, we use the terms 'SOL axis' and 'cross-SOL axis' as shorthand for directions parallel to the long and short sides of the domain respectively.

$$U_r \frac{\partial}{\partial t} n = -\nabla \cdot (n\vec{u}) + S^n, \tag{14}$$

$$U_r \frac{\partial}{\partial t} n\vec{u} = -\nabla \cdot (n\vec{u} \otimes \vec{u} + nT\vec{I}) + S^u, \tag{15}$$

$$U_r \frac{\partial}{\partial t} (n((g-2)T + u^2)) = -\nabla \cdot (n\vec{u}(gT + u^2)) - \kappa_d \nabla^2 T + S^E, \tag{16}$$

Starting from Equations (14)– (16), we set $U_r = 1$ and choose to neglect heat conduction ($\kappa_d = 0$). We assume that the equation of state is that of an ideal gas with isentropic expansion factor $\gamma = 5/3$, which equates to setting $g = 2$ in Equations(15) and (16).

In the Nektar solver framework, the addition of a second dimension entails adding an extra momentum density field, nv and modifying the construction of the flux vector used in the Riemann solver when computing the advection terms.

We apply a Discontinuous Galerkin (DG) formulation to model the fields and fluxes between elements are calculated using a Riemann solver (implementation due to Toro [8]). We use basis functions constructed from a modified form of (5^{th} order) Legendre polynomial, described by Karniadakis and Sherwin [9]. The fluid fields are evolved using an explicit, 4^{th} order, Runge Kutta time stepping scheme.

In order to test the solver with non-zero momentum terms in both dimensions, we rotate the domain about the origin by an angle θ, such that the SOL axis is no longer aligned with the x-axis. The

rotation angle is set as a Nektar parameter in the configuration file and is used to rotate the fluid source terms such that they vary along the SOL axis, but are constant in the cross-SOL direction. The (Dirichlet) boundary conditions at the ends of the flux tube also need to account for the rotation. They become

$$n = n_{\text{inf}}, \tag{17}$$

$$nu = \alpha\, n_{\text{inf}}\, u_{\text{inf}}\, \cos(\pi/4), \tag{18}$$

$$nv = \alpha\, n_{\text{inf}}\, u_{\text{inf}}\, \sin(\pi/4), \tag{19}$$

$$E = \frac{p_{\text{inf}}}{\gamma - 1} + \frac{n_{\text{inf}}\, u_{\text{inf}}^2}{2}, \tag{20}$$

where u and v are speeds in the x and y directions respectively. We apply outflow conditions by setting $\alpha = -1$ at one end of the domain, and 1 at the other. We set $n_{\text{inf}} = p_{\text{inf}} = u_{\text{inf}} = 1$, noting that the density and velocity have already been non-dimensionalised with factors of B_0/N_0 and the sound speed, c_s, respectively. On the (long) sides of the flux tube, boundary conditions are set to be periodic for all fluid fields.

2.2.5 Fluid-only validation

In the following, we first consider validation of the SIMPLE-SOL solver in isolation, that is, without coupling to the particle framework. We choose a rectangular mesh which is 110 units long, 1 unit high and rotated about the origin by an angle $\theta = \pi/4$. Fluid elements are quadrilateral, with 56 along the SOL axis and 3 in the cross-SOL direction.

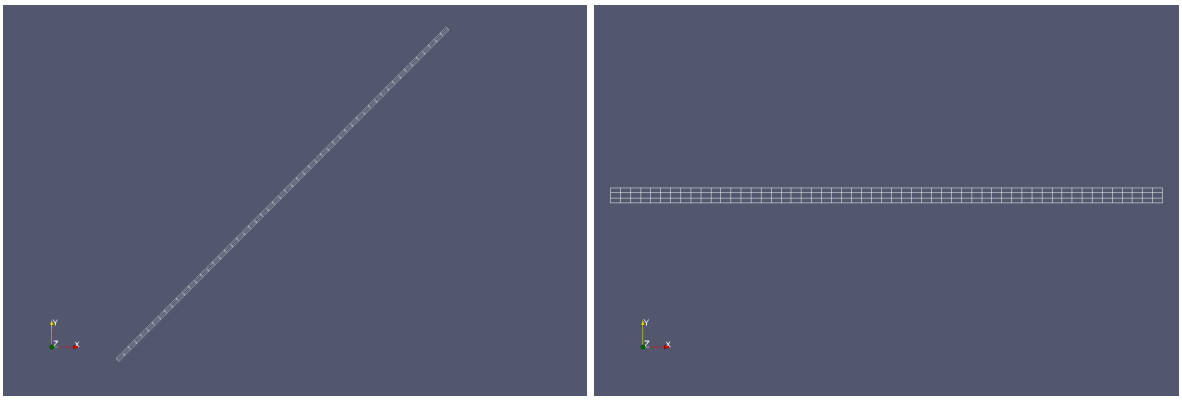


Figure 3: Left: The SOL 2-D domain, with a rotation angle of $\theta = \pi/2$. Right: The SOL 2-D domain without any rotation; the domain width is exaggerated by a factor of three, in this panel.

The left panel of Figure 3 shows the rotated mesh used for validation. The right panel shows a version of the same mesh prior to rotation, with the aspect ratio reduced to make individual quadrilateral elements easier to see.

We set initial conditions (in non-dimensionalised units) according to Table 1 and choose initial conditions to differ significantly from the expected solution. Our fluid source terms are set to

Table 1: Initial conditions used for validation.

Field label	Parameterised value	Numeric value
n	n_{inf}	1
nu	0	0
nv	0	0
E	$p_{\mathrm{inf}}/(\gamma-1)+\frac{n_{\mathrm{inf}}\ u_{\mathrm{inf}}^2}{2}$	1.5

match one of the configurations used to validate the SOLDRAKE solver [10]. When plotted parallel to the SOL axis, the sources closely resemble those shown in Figure 8 of that reference, but with Gaussian widths five times narrower. We set the magnitude of each source to be constant in the cross-SOL direction.

When the simulation starts, the fields initially vary rapidly, before settling down into a steady state. This is seen clearly in the number density profiles shown in Figure 4, which are computed at five output times between $t = 0$ and $t = 200$. The label 's', here, refers to the distance along the SOL axis from one end of the field line to the other. Although we evaluate the number density along the lower edge of the domain the symmetry of the source terms means that the same profile can be obtained along any line parallel to the SOL axis.

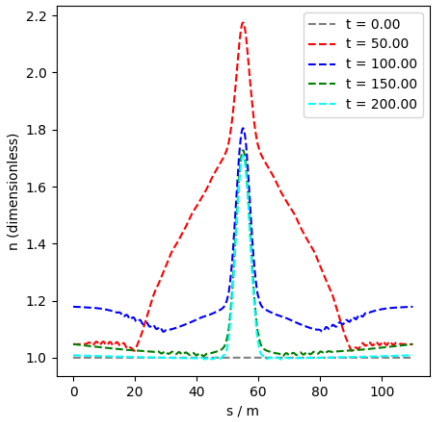


Figure 4: Evolution of the number density profile along the SOL axis, from initial conditions to steady state.

In Figure 5 we plot a comparison between the steady state reached by the SIMPLE-SOL solver and an analytic solution presented by Arter [11]. The top row of panels are 1-D profiles in number density, velocity and energy along the SOL axis, while the bottom row shows the difference between the SIMPLE-SOL and analytic solutions.

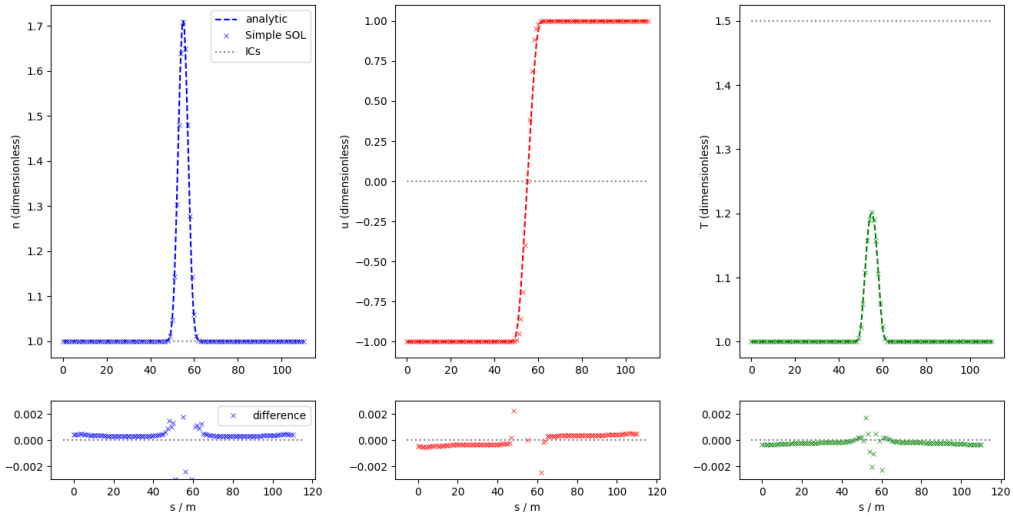


Figure 5: Upper panels: Comparison of the density (left), velocity (centre) and temperature (right) profiles along the SIMPLE-SOL field line. Lower panels: Difference between the simulated and analytic profile.

2.2.6 Coupling with NESO-PARTICLES

We make modifications to the SIMPLE-SOL solver to facilitate coupling with the NESO-PARTICLESframework as follows:

1. The temperature is computed based on the equation of state and stored in a separate Nektar field at the beginning of each time step, then evaluated at the particle positions before calculating the ionisation rates.

2. Four new fields were added, one for each particle source term. These fields are explicitly excluded from the advection operation.

3. The source terms class was updated to add the particle source fields to the RHS of their respective fluid equations.

2.2.7 Particle Creation Along Lines

The model system includes two locations where a neutral species is introduced into the domain. We inject particles at both ends of the domain which represents the situation where neutral species are deliberately injected into the plasma or where charged species are neutralised on contact with a wall and are repelled back into the plasma.

For parallel plasma flow in the x direction, we initialise neutral particles along straight lines in the y direction. At each end of the domain, at $L - a$ and a for a constant offset a and domain length L, we define a collection of lines centred around these two predetermined points. Each collection of lines spans the entire domain in the y direction and Gaussian spacing in the x direction. Each line is discretised into a large number of uniformly spaced points where particles may be created.

Discretising each line into a large number of points allows performance optimisations to be made that reduce the cost of adding particles to the domain. We pre-compute the MPI ranks that own points on the line to reduce the communication overhead of adding particles. Secondly we pre-compute the owning cell and NEKTAR++ reference positions of the points to avoid a costly cell binning process which would only occur on a subset of MPI ranks.

At each time step, when we inject particles, we initialise each particle with a computational weight that represents the physical number of particles that computational particle represents. This computational weight is subsequently reduced by the ionisation process we describe in this report. Secondly the velocity of each particle is sampled from a Gaussian distribution such that the overall ensemble of neutral particles exhibits a desired temperature.

2.2.8 Particle Boundary Conditions

The 2-D system we describe in this report is constructed with two types of behaviour for particles at the domain edges. In the y direction, which is the perpendicular direction to the field lines, we periodically map particles over the boundary. In the x direction, which is the direction parallel to the field lines, we remove particles from the simulation domain when they reach the domain edge.

Previous iterations of NESO-PARTICLES(NP) did not provide users with a high level interface to remove particles. For the simulation we describe here, and to extend the functionality of NP, we implemented a high level user facing interface that allows specification of which particles should be removed from the simulation. To use this interface a user tags particles that should be removed from the system with a user defined value in a user defined particle property. The user then calls the remove method and passes their chosen tag value and chosen particle property. On completion of the call to the remove method all particles matching the tag value in the property are removed.

2.3 Coupled proxyapp results

In Figure 6 plot the distribution of fluid plasma (upper image) and neutral particles (lower image) towards one end of the domain, corresponding to the region closest to the divertor. The position of the particle source in the lower panel is easily identified by the concentration of points with high computational weights (redder colours). As the simulation evolves, computational particles move away from the source location, becoming progressively more ionised (bluer colours) and depositing material onto the plasma density field (redder colours in the upper panel). This is then advected towards the domain boundaries by the fluid solver. Particles crossing the left-boundary

are automatically removed from the simulation at each time step, using the mechanism described in Section 2.2.8.

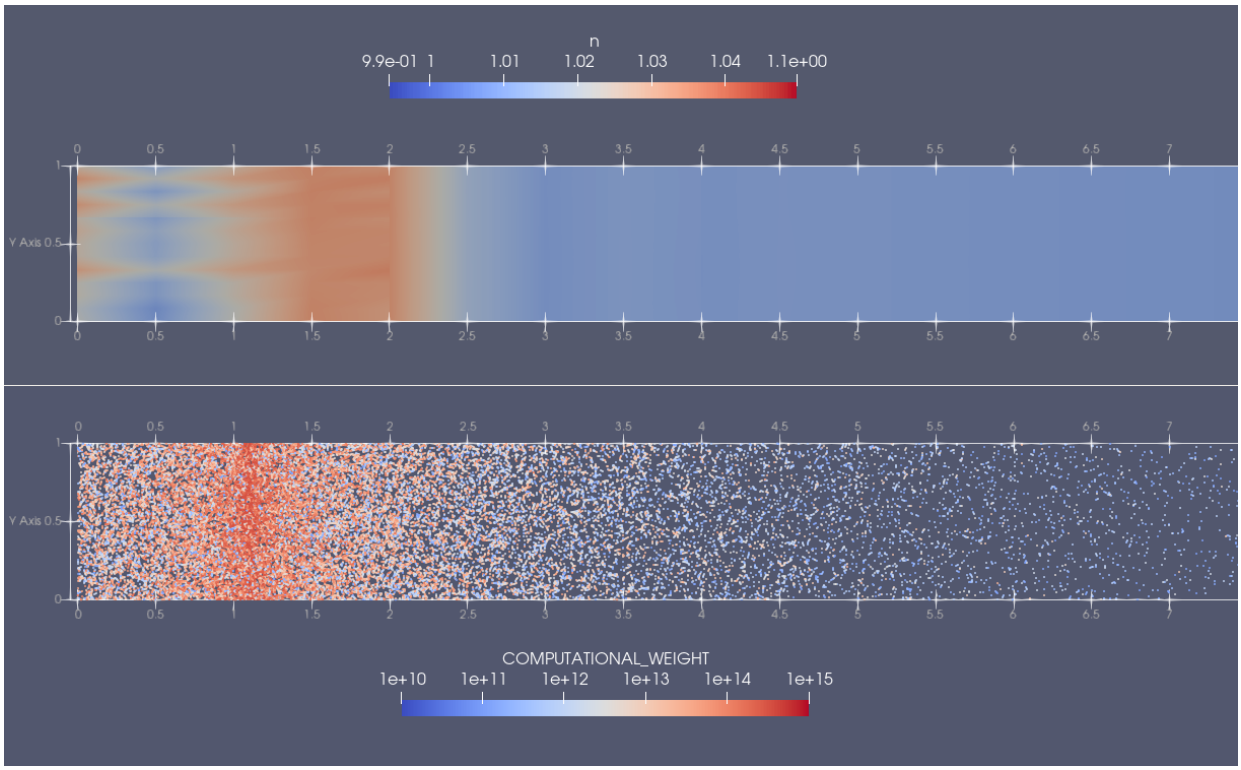


Figure 6: Zoom-in on one end of the 2-D domain showing the plasma number density (top) and distribution of neutral particles (bottom) coloured by weight.

To initially test the mass conservation properties of the simulation we created a closed system by replacing the outflow boundary conditions in the x direction with periodic boundary conditions. This new system, with fully periodic boundary conditions, contains zero sinks for neutral particles or plasma fluid and hence will rapidly "fill" with plasma. As the system is closed, accurately measuring the total mass in the system is now straightforward.

We constructed a system where the plasma fields n and $\vec{u}$ are initialised to 1 and $\vec{0}$ respectively. Furthermore we disabled all source terms except the mass coupling between the plasma fluid and neutral particles. This remaining term is responsible for ionising neutral particle species as a plasma fluid density source. Mass is created at each time step at the injection points we describe earlier in this report. Hence the total mass in the system linearly increases with time at a chosen rate.

In Figure 7 we plot the mass stored as both neutral particles and plasma fluid along with the relative error in total system mass. As the system is closed except for our neutral particle injection we computed the correct total system mass as the mass of the injected particles plus the initial plasma field density. By comparing the expected system mass against the particle mass and fluid mass we quantified any loss in mass due to conversion between particle and fluid representations or loss of mass due to implementation errors. We observe good conservation of mass from this

test case.

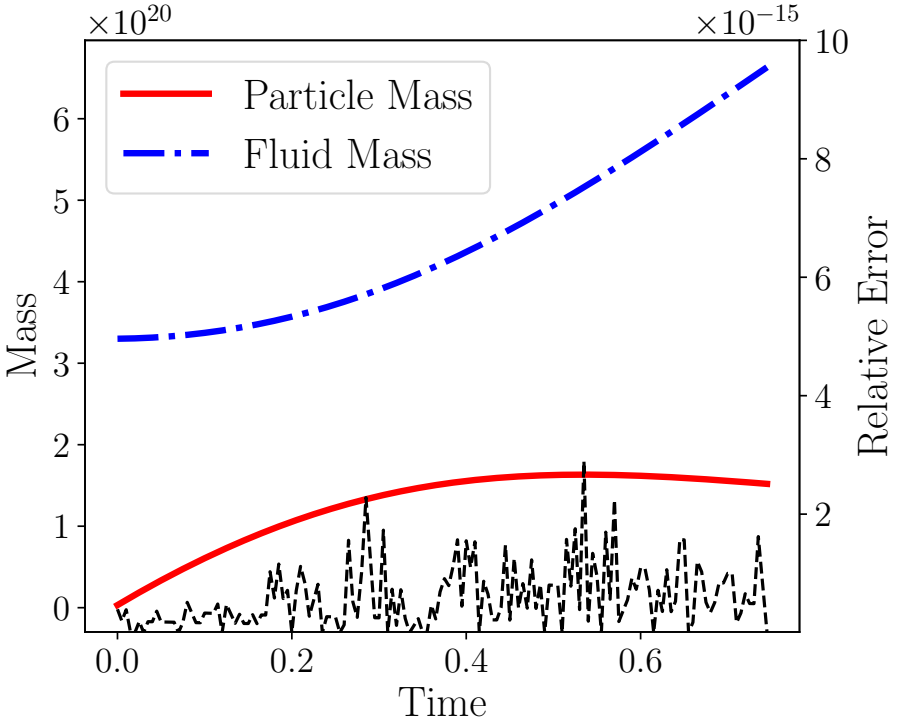


Figure 7: Mass conservation test for closed system under periodic boundary conditions. Mass expressed in terms of particle number. Black: relative error in total system mass. Blue: mass contained in fluid density fields. Red: mass stored on neutral particles.

3 Executive Summaries

This section both introduces the context and summarises two reports in the higher dimensional work package. The first relates to implementing higher dimensional models using a continuum and the second is an investigation into the performance of code related to the first.

Accepted reports are held on the Documents repository of the ExCALIBUR-NEPTUNE organisation on GitHub, to which belong other repos containing software developed under project NEPTUNE. Note that some of these repos may be access-controlled, please email `neptune@ukaea.uk` if difficulties are encountered in seeing the material.

3.1 Implementation of TensorRegions library in Nektar++

Authors: Chris Cantwell, Spencer Sherwin, and David Moxey.

The context of this brief report[12] is the step change in exhaust power of a power-plant scale tokamak compared to that of current experimental devices. The resulting increase in local plasma temperature will reduce the collisionality of the plasma in the scrape-off-layer thereby rendering the existing set of fluid models and fluid numerical methods much less accurate in this parameter regime. Although NEPTUNE includes a particle modelling workstream, which also enables kinetic effects, particle simulations have limitations due to sampling noise which deterministic finite element calculations do not suffer from. Higher dimensional finite element calculations have their own issues however, notably regarding cost, and to some extent the two approaches are complementary.

The extension of NEKTAR++ to accommodate one- or two-dimensional velocity space continuum grids (typically referred to by 1V or 2V) superimposed onto 1-D, 2-D and 3-D spatial domains is detailed in this report. The introduction describes how existing components have been re-used in the development of this feature. The second section on implementation explains the how to configure and create the additional continuum dimensions to resolve velocity space: the `TensorRegion`(ie. the velocity space representation embedded in a spatial region); the `TensorStorage` or storage arrays for calculations; and a read/write `View` of this underlying data. Next code snippets are given in the Example Usage section, where it is explained how this work fits into the wider context of NEKTAR++ developments. The report closes with automatically generated API documentation.

3.1.1 Report on DG Performance in NEKTAR++

Authors: Edward Laughton, David Moxey, Chris Cantwell, and Spencer Sherwin.

This report[13] concerns high order discontinuous Galerkin (DG) methods, which have many attractive properties for Exascale fluid dynamics. Arguably first among these is the accuracy achieved per unit amount of MPI communication. Since Nektar++ makes use of spatial DG methods for higher dimensional methods, which comes at a higher computational cost due to the curse of dimensionality, it is imperative that the implementation is as performant as possible. This report details a real world simulation case, a Taylor-Green vortex, to attempt to identify bottlenecks

for this high dimensional relevant method. Ultimately, no out-of-the-ordinary bottlenecks exist for the current implementation on x86-64 architectures. However, having performed some tests and analysed the profiling date, the authors highlight a few areas where algorithm improvements may be made.

The first section summarises the DG formulation for the Navier-Stokes (NS) system of equations, giving emphasis to components of it that will feature later in the report: the number of communicated degrees of freedom; the requirement for an inverse matrix multiplication; the auxillary variable required to accommodate the diffusion operator separating NS from compressible Euler. The experimental setup of 6th order 3-D advection-diffusion system (Navier-Stokes) on an $8 \times 8 \times 8$ grid is introduced.

The second section describes the profiling procedure as well as some caveats around different approaches to MPI communication that may influence results. As such it is noted that care was taken when profiling MPI communication.

The results section comes third. It notes the requirement of an `MPI_AllReduce` in each substep of the Runge-Kutta 4 integrator, that the flux values must be communicated across sub-domain boundaries, and that there is some jitter in the MPI but would not adversely affect wall-clock time for the whole simulation. It is noted that the fluid solver spends much of its time in low level matrix-multiply BLAS calls, as expected.

The report concludes with several observations that might help improve performance:

- that the cost of the inverse matrix multiply may be alleviated by a real Schur decomposition of the matrix or by swapping to the use of orthogonal polynomials.
- an MPI call could be streamlined so as not to send extraneous data for this use-case.
- an area of the code has been found where it is possible to interleave MPI with compute.

4 Summary

In this report we describe a spatially 2-D model for fluid plasma equations relevant to the plasma exhaust. This model consists of a fluid system to describe a plasma species and a particle system to describe a neutral species. These two species are coupled through an interaction between plasma and neutrals where neutral species are ionised at a particular rate. The ionised particles create a source term for plasma species in the associated fluid equations. Hence there exists a coupling from neutral species to plasma species. Secondly the ionisation rate, evaluated at particle locations, is a function of the plasma density and temperature. Hence there exists a coupling from plasma species to neutral species.

The implementation of the coupling operations relies on the capability to evaluate finite element functions at particle locations and the ability to convert particle representations to finite element representations. These evaluation and projection operations are frequently performed and expensive operations and hence are performance critical. As part of the task work for this report we implemented algorithms to perform these operations within SYCL. SYCL implementations of

these core operations improve the overall performance portability of the NESO framework and allow these operations to be executed on a wider range of hardware. We complete the description of our own work in this report by presenting initial validation results from the fluid and particle systems. These validation results are important to demonstrate that the coupling between fluid and particles is correctly implemented and conserves key system quantities.

Lastly, summaries and links to external grant work associated with D4c are provided.

Acknowledgement

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

References

[1] W Lotz. Electron-impact ionization cross sections and ionization rate coefficients for atoms and ions from hydrogen to calcium. *Z. Phys., 216: 241-7(1968).*, 1 1968.

[2] E.L. Freeman and E.M. Jones. Atomic Collision Processes in Plasma Physics Experiments. Technical Report CLM-R137, Culham Laboratory, 1974. `https://scientific-publications.ukaea.uk/wp-content/uploads/CLM-R137.pdf`.

[3] D.A. Barry, J.-Y. Parlange, and L. Li. Approximation for the exponential integral (Theis well function). *Journal of Hydrology*, 227(1-4):287–291, 2000.

[4] Jean-Paul Berrut and Lloyd N. Trefethen. Barycentric lagrange interpolation. *SIAM Review*, 46(3):501–517, 2004.

[5] W. Arter, J. Cook, and W. Saunders. 1-D and 2-D particle models. Technical Report CD/EXCALIBUR-FMS/0070-M4c.1, UKAEA, 2022. `https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/ukaea_reports/CD-EXCALIBUR-FMS0070-M4c.1.pdf`.

[6] D. Moxey et al. 1D SOL Solver. `https://github.com/ExCALIBUR-NEPTUNE/nektar-1d-sol`, 2022.

[7] W. Arter et al. Equations for EXCALIBUR/NEPTUNE Proxyapps. Technical Report CD/EXCALIBUR-FMS/0021-1.24-M1.2.1, UKAEA, 1 2023. `https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/ukaea_reports/CD-EXCALIBUR-FMS0021-1.24-M1.2.1.pdf`.

[8] Eleuterio F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics*. Springer Berlin Heidelberg, 2009.

[9] G. Karniadakis and S. Sherwin. *Spectral/hp element methods for computational fluid dynamics 2nd Ed.* Oxford University Press, 2005. `https://doi.org/10.1093/acprof:oso/9780198528692.001.0001`.

[10] W. Saunders and E. Threlfall. Finite Element Models: Performance. Technical Report CD/EXCALIBUR-FMS/0047-M2.2.2, UKAEA, 9 2021. https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/ukaea_reports/CD-EXCALIBUR-FMS0047-M2.2.2.pdf.

[11] Wayne Arter. Study of source terms in the SOLF1D edge code. Technical report, Eurofusion/CCFE, 2015. CCFE-DETACHMENT-RP2-Draft.

[12] C. Cantwell, S.J. Sherwin, and D. Moxey. D1.1: Implementation of TensorRegions library in Nektar++. Technical Report 2060042-TN-01-2, UKAEA Project Neptune, 2022. https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/2060042/TN-01-2.pdf.

[13] D. Moxey, C. Cantwell, and S.J. Sherwin. D3.1: Report on DG performance in Nektar++. Technical Report 2060042-TN-02-4, UKAEA Project Neptune, 2022. https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/2060042/TN-02-4.pdf.