UK Atomic Energy Authority

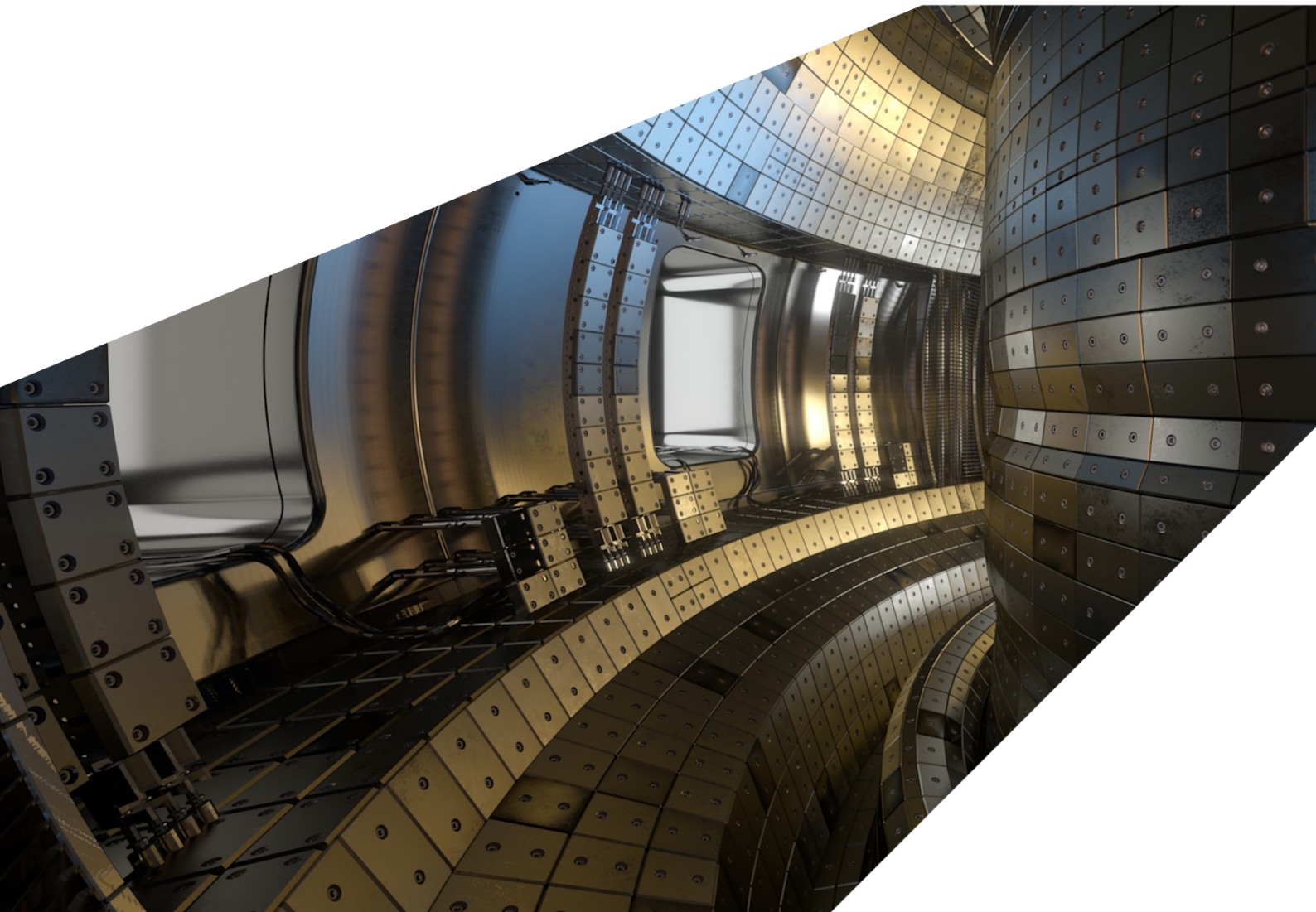
ExCALIBUR

Complementary actions. Code-coupling, physics databases and benchmarking techniques 3.

M7c.3 Version 1.00

Abstract

The report describes work for ExCALIBUR project NEPTUNE at Milestone M7c.3. Complementary actions are described which assist code-coupling and the use of benchmarking techniques and physics databases, generally providing support to more than one Y5 deliverable. Internal work throughout Y5 has been incremental with respect to the activities described in the previous report, namely provision of introductory technical material, text processing tools and improvements to the developers' website. Further extensive work by the grantees at York, on the development of benchmark problems, DSLs and multi-species physics, and at STFC on relevant preconditioning and timestepping numerical techniques, is summarised, and ongoing internal work on atomic physics databases is again annexed as a separate report.



UKAEA REFERENCE AND APPROVAL SHEET

	Client Reference:	
	UKAEA Reference:	CD/EXCALIBUR-FMS/0085
	Issue:	1.00
	Date:	15 March 2024
Project Name: ExCALIBUR Fusion Modelling System		

	Name and Department	Signature	Date
Prepared By:	Wayne Arter Matthew Barton Seimon Powell CD	N/A N/A N/A	15 March 2024 15 March 2024 15 March 2024
Reviewed By:	Will Saunders CD		15 March 2024

1 Introduction

This work for Deliverable 7c is part of a programme of work items designed to provide an uninterrupted programme of coordination work providing support to more than one Y5 deliverable. Internal work throughout Y5 has been incremental with respect to the activities described in the previous report M7c.2 Report [1]. Work on the use of atomic physics data again forms a separate Deliverable 8c, the M8c.2 note on which appears as an annex to the present report.

Work conducted internally by UKAEA is described in Section 2. Aside from routine maintenance tasks such as posting and indexing reports received from grantees after acceptance, continuing work is efficiently described under four subsection headings, namely (1) educational material in Section 2.1, (2) technical material in Section 2.2, (3) text processing tools in Section 2.3, and (4) improvements to the developers' website Section 2.4. Experience during Y5 indicates that the incremental nature of the work does not seem to preclude its becoming time-consuming. Benchmarking work aimed at helping optimise NESO execution is described in Section 2.5. Work described in 16 reports accepted from the external grantees during the year Y5, is summarised in Section 3.

2 Task Work by UKAEA

2.1 Educational Material

Further introductory material has been added to demonstrate important features of finite element algorithms relevant to Project NEPTUNE. In due course it is intended that this “SIAM Review" style material will be made public along with reports already available internally to UKAEA (such as the guide to finite elements aimed at people who have formerly only used finite difference schemes).

The added material uses examples of low order finite elements to illustrate basic finite element concepts such as ‘mass lumping’, then proceeds to more advanced concepts such as Petrov-Galerkin (mixed basis) schemes, finishing with the discrete ‘de Rham’ complex.

2.2 Technical Material

There are a number of issues raised by the mixing of particle and continuum representations of fields. A minimum requirement would seem to be conservation of mass, momentum and energy. The achievement of this goal in the case of charge conservation is described below.

2.2.1 Particle and Mesh Interaction

It may help allay confusion if it is realised that the current implementation of particle-mesh interaction as described in the annexed M8c.2 report, makes changes to both neutral super-particle mass and velocity in order to avoid the need to introduce a new particle at each step. However, since this simpliflication necessarily fails to conserve energy, it is envisaged that a more complex model of particle production and merging, incorporating the formulae below will be needed.

Consider the case where a (super-)particle undergoes a charge-exchange interaction with an ion of the background plasma modelled as a continuum, where initially the continuum is considered as modelled by a zeroth order finite element, with the extension to order p elements deferred to the end of the section. Dealing with the end-points of the neutral particle interaction, ie. the initial and final weights and velocities, is straightforward. Suppose the neutral super-particle begins with weight-velocity product $W\mathbf{v}_0$ and interacts with a plasma ion of weight w_i and velocity $\mathbf{v}_i$, then the neutral continues with product $W^+\mathbf{v}_0$, where $W^+ = W - w_i$, giving birth to a neutral with weight-velocity product $w_0\mathbf{v}_i$ where in the usual event that the particle and the background ion species are the same, $w_0 = w_i$.

Difficulties arise in the treatment of the continuum which has nominally lost a particle $w_0\mathbf{v}_i$ and gained one with weight-velocity product $w_0\mathbf{v}_0$. It is important that mass, momentum and energy should be properly accounted for, eg. should the charge-exchange interaction be inelastic, these three quantities should be conserved for the combined neutral-plasma system. Evidently it is only necessary to consider the case where the plasma gains a particle with weight-velocity product $w_0\mathbf{v}_0$, since the case of loss may be taken care of by letting $w_0 \to -w_0$.

The moments of the continuum of species α, following Cercignani [2, § II.8] may be defined at a point $\mathbf{x}$ as

$$\begin{aligned} n_\alpha = & \quad \int f_\alpha d\mathbf{v}_\alpha & (1)\\ n_\alpha \mathbf{u}_\alpha = & \quad \int f_\alpha \mathbf{v}_\alpha d\mathbf{v}_\alpha & (2)\\ n_\alpha K_I T = & \quad \tfrac{1}{2}\int f_\alpha (\mathbf{v}_\alpha - \mathbf{u}_\alpha)^2 d\mathbf{v}_\alpha & (3) \end{aligned}$$

where $K_I T$ replaces Cercignani's internal energy 'e'$= \frac{3}{2}kT/m_i$, so that $K_I = \frac{3}{2}k/m_i$. Following the introduction of temperature T in this way, the energy density at $\mathbf{x}$ is

$$\mathcal{E}_\alpha/m_i = \frac{1}{2}\int f_\alpha \mathbf{v}_\alpha^2 d\mathbf{v}_\alpha = n_\alpha K_I T + \frac{1}{2} n_\alpha {\mathbf{u}_\alpha}^2 \tag{4}$$

In the discrete case, suppose the moments are calculated by summation over the distribution of particles $w_p \mathbf{v}_{\alpha p}$ labelled $p > 0$ within a spatial volume V^e local to the point $\mathbf{x}$, then

$$\begin{aligned} n_\alpha V^e = & \quad \Sigma_p w_p & (5)\\ n_\alpha V^e \mathbf{u}_\alpha = & \quad \Sigma_p w_p \mathbf{v}_{\alpha p} & (6)\\ n_\alpha V^e K_I T = & \quad \tfrac{1}{2}\Sigma_p w_p (\mathbf{v}_{\alpha p} - \mathbf{u}_\alpha)^2 & (7) \end{aligned}$$

and the energy

$$\mathcal{E}_\alpha . V^e/m_i = \frac{1}{2}\Sigma_p w_p \mathbf{v}_{\alpha p}^2 = \frac{1}{2}\Sigma_p w_p \left[(\mathbf{v}_{\alpha p} - \mathbf{u}_\alpha)^2 + \mathbf{u}_\alpha^2\right] \tag{8}$$

the last equality following by use of Equation (6) as explained by Cercignani.

Dropping the species label α and setting $N_P = nV^e$, gives the moments in a form convenient for manipulation as

$$\begin{aligned} N_P = & \quad \Sigma_p w_p & (9)\\ N_P \mathbf{u} = & \quad \Sigma_p w_p \mathbf{v}_p & (10)\\ N_P K_I T = & \quad \tfrac{1}{2}\Sigma_p w_p (\mathbf{v}_p - \mathbf{u})^2 & (11) \end{aligned}$$

and the energy as

$$\mathcal{E} . V^e = m_i (N_P K_I T + \frac{1}{2} N_P \mathbf{u}^2) \tag{12}$$

Following the gain of a particle with weight-velocity product $w_0 \mathbf{v}_0$, the discrete expressions for the moments become

$$\begin{aligned} N_P^+ = & \quad w_0 + \Sigma_p w_p & (13)\\ N_P^+ \mathbf{u}^+ = & \quad w_0 \mathbf{v}_0 + \Sigma_p w_p \mathbf{v}_p & (14)\\ N_P^+ K_I T^+ = & \quad \tfrac{1}{2} w_0 (\mathbf{v}_0 - \mathbf{u}^+)^2 + \tfrac{1}{2}\Sigma_p w_p (\mathbf{v}_p - \mathbf{u}^+)^2 & (15) \end{aligned}$$

It will be seen at once that the number of particles satisfies $N_P^+ = N_P + w_0$, so the number density updates as

$$n^+ = n + w_0/V^e \tag{16}$$

and similarly for the continuum velocity

$$\mathbf{u}^+ = \frac{n\mathbf{u} + (w_0/V^e)\cdot\mathbf{v}_0}{n + w_0/V^e} \tag{17}$$

whence it follows at once that the continuum's mass and momentum satisfy

$$(m_i n^+ - m_i n)V^e = w_0 m_i \tag{18}$$

$$(m_i n^+ \mathbf{u}^+ - m_i n\mathbf{u})V^e = w_0 m_i \mathbf{v}_0 \tag{19}$$

Regarding energy and temperature updating for the continuum, first remark that

$$\mathcal{E}^+.V^e = m_i(N_P^+ K_I T^+ + \frac{1}{2}N_P^+ \mathbf{u}^{+2}) = \frac{m_i}{2}w_0\mathbf{v}_0^2 + \frac{m_i}{2}\Sigma_p w_p \mathbf{v}_p^2 \tag{20}$$

which follows immediately from Equation (8), although Cercignani's explanations are helpful, thus the energy updates as

$$\mathcal{E}^+ = \mathcal{E} + \frac{m_i}{2}(w_0/V^e)\mathbf{v}_0^2 \tag{21}$$

and energy conservation is immediate. However, in practice, the temperature field has often to be updated. An explicit expression may be obtained by substituting for $\mathcal{E}$ using Equation (12) and for $\mathcal{E}^+$ using the first relation in Equation (20), then using Equation (16) and Equation (17) to simplify so that

$$K_I T^+ = \frac{n}{n^+}\left[K_I T + \frac{w_0}{2n^+}(\mathbf{u} - \mathbf{v}_0)^2\right] \tag{22}$$

where n^+ is taken from Equation (16). This result has been checked by substituting in expressions (cf. Equation (7)) for temperature alone, and note that, as expected, the new temperature reduces if $\mathbf{u} = \mathbf{v}_0$.

As stated above, the continuum has so far been treated as though it were modelled with $p = 0$ elements. However, provided the finite elements satisfy the property known as 'partition of unity' (true for interpolatory bases such as usually employed in Nektar++) then it is consistent in the above expressions Equations(16) and (17) and Equation (22) to replace $(N, \mathbf{u}, T)$ with nodal values in the element. Thus, for example, suppose that the continuum is represented within the element by a basis ψ_i such that $n(\mathbf{x}) = \Sigma_i n_i \psi_i(\mathbf{x})$, and that it is to be updated by addition of a particle at $\mathbf{x}_q$, using the formula Equation (16), then equating in the weak sense over each ψ_j,

$$\Sigma_i n_i \langle\psi_i, \psi_j\rangle = n^+\psi_j(\mathbf{x}_q) \tag{23}$$

where $\langle , \rangle$ denotes the usual inner product on the finite element basis space. Assuming orthogonality, the formula reduces to $n_j w_j = n^+\psi_j(\mathbf{x}_q)$ (no summation), where $w_j = \langle\psi_j, \psi_j\rangle$ is a continuum quantity.

2.3 Text processing tools

The tools are only available on the private CCFE gitlab site. Their wider publication awaits user testing and feedback. For UKAEA staff with access to CCFE gitlab, the tools are to be found either in directory `tex` or in its subdirectory `importools`. They are largely based on Linux `sed` or `vim` editors. There have been only minor improvements to the toolkit, changes to .ed files aimed at suppressing unwanted messages from these editors. (.ed files are briefly described in the README, beware that the input is invariably overwritten so should be copied before editing in case of mistakes.)

2.4 Improvements to the Developers' Website

The main improvements were additions and corrections to the tables that listed the mathematical symbols and acronyms used in NEPTUNE reports. It also became necessary to add material on algorithmic approaches to NEPTUNE such as AMR and Lattice Boltzmann, to make clear that these were deprecated in the very early stages of defining the project. Thus any drive to re-introduce such techniques to NEPTUNE, should have to pass a high barrier, so that the agreed NEPTUNE procedure for introducing novelties to be found in the Management File section of the developers' website [3], would be the absolute minimum needed to force re-consideration.

The automatic deployment of updated web-pages ceased in late 2024, due apparently to a failure of the GitHub developers to preserve upwards compatibility of workflow scripts. This was not initially realised and only fixed by UKAEA staff in mid-February 2024.

2.5 NESO Optimization

Profiling of NESO's *SimpleSOL* solver indicates that the most significant hotspot is the function that projects the particle fields on to the finite element solution - Algorithm 2.1 illustrates the basic algorithm. Two features in particular contribute to its runtime. Firstly, a computationally expensive calculation of the modified basis (line 2). Secondly, updating a cell's degrees of freedom (DOFs) in parallel, line 5, requires some, potentially slow, mechanism to prevent a data race e.g. atomic operations.

```
1 for(auto &particle: all_particles) {
2     modified_basis = calculate_basis(particle.position);
3     //loop over the degrees of freedom in the cell
4     for (auto &dof: particle.cell) {
5         dof += calculate_dof(modified_basis,particle.value);
6     }
7 }
```

Algorithm 2.1: Illustrative example for serial implementation projection algorithm

The next section (2.5.1) will benchmark various ways in which Algorithm 2.1 can be parallelised. Then in Section 2.5.3 we briefly discuss how template meta-programming has been used to optimize the evaluation of the modified basis functions.

2.5.1 Alternative algorithms for projection function

In this section we first describe 4 variations of the algorithm in Figure 2.1 followed by benchmarking results. Some details and SYCL boilerplate are omitted in the pseudo-code listings below for clarity. In particular SYCL kernels execute for each element of an index-space, each index represents a work-item. The index space can be a simple (N-dimensional) **RANGE** object or an **ND_RANGE** object which enables work-items to be collected into work-groups that execute on the

same processing element [4]. With one exception (Algorithm 2.3) every algorithm described below executes over an **ND_RANGE**. SYCL work-items are do not necessarily map onto hardware threads but we will use the terms interchangeably. Indeed the pseudo-code is not specifically tied to SYCL they could just as easily be implemented in OpenMP or CUDA for example.

Thread per particle In Algorithm 2.2 every index in the index space represents a single particle. Each thread calculates a particle's contribution and atomically updates its cell's degrees of freedom. This is the current version in NESO.

```
1 parallel_for(auto &particle: all_particles) {
2     modified_basis = calculate_basis(particle.position);
3     //loop over the degrees of freedom in the cell
4     for (auto &dof: particle.cell) {
5         val = calculate_dof(modified_basis,particle.value);
6         dof = atomic_fetch_add(dof,val);
7     }
8 }
```

Algorithm 2.2: Thread per particle

Thread per cell In contrast to Algorithm 2.2 in Algorithm 2.3 each index represents a single cell. As each thread has its own cell there is no need for an atomic update.

```
1 parallel_for(auto &cell: all_cells) {
2     for (auto &par: cell) {
3         modified_basis = calculate_basis(particle.position);
4         //loop over the degrees of freedom in the cell
5         for (auto &dof: particle.cell) {
6             dof += calculate_dof(modified_basis,particle.value);
7         }
8     }
9 }
```

Algorithm 2.3: Thread per cell

Kernel per cell Algorithm 2.4 launches one or more [1] kernels for every cell. Within each kernel the work is parallel over particles before each particle's contribution is reduced and atomically added to the cell's DOFs.

[1] Availability of device memory limits the number of particles that can be processed in a single kernel

```
1  for(auto &cell: all_cells) {
2      //Kernel launches here
3      parallel_for (auto &par: cell) {
4          //calculate basis for every particle in local group
5          modified_basis = calculate_basis(particle.position);
6          group_barrier();
7          //loop over the degrees of freedom in the cell
8          for (auto &dof: cell) {
9              var = group_reduction(modified_basis)
10             if (thread_id == 0) {
11                 atomic_fetch_add(dof,var);
12             }
13         }
14     }
15 }
```

Algorithm 2.4: Kernel per cell

Thread per DOF In Algorithm 2.5 each thread is responsible for calculating the modified basis for a particle as in Algorithm 2.2. But unlike Algorithm 2.2 there is synchronisation barrier after which each thread is responsible for single degree of freedom in a cell. The thread then calculates its contribution in the loop beginning on line 7 before the atomic update. Compared to Algorithm 2.2 this reduces the number of atomics by a factor equal to the number of particles in the local group.

```
1  parallel_for(auto &thd: nthread ) {
2      //Assign one thread per particle
3      modified_basis[thd] = calculate_basis(particles[thd].position);
4      //synchronize
5      group_barrier();
6      //Now one thread per degree of freedom
7      for (auto &par: particles) {
8          temp += calculate_dof(modified_basis[par]);
9      }
10     atomic_fetch_add(dof[thd],temp);
11 }
```

Algorithm 2.5: Thread per DOF

2.5.2 Benchmark results

Note that for all results presented in this section, with AdaptiveCPP for the CPU we use the accelerated OpenMP/LLVM backend, on the GPU the clang CUDA backend i.e. we add the following to

the `cmake` command line `-DACPP_TARGETS="omp.accelerated"`, or `-DACPP_TARGETS="cuda:sm_80"` for CPU or GPU respectively. For IntelSYCL we are using the `opencl:cpu` backend.

Scaling with increasing problem size Keeping resources fixed[2] we increase the problem size to compare each algorithm's performance but also to highlight issues with under under-utilisation for low numbers of cells. Figure Figure 1 shows that overall the Thread per DOF kernel (Algorithm 2.5) is usually the fastest. A manually vectorised version of Algorithm 2.3 is faster given a sufficient number of cells and then only on a CPU when using AdaptiveCPP (formally known as HipSYCL/OpenSYCL). The performance of the benchmarks are broadly similar between IntelSYCL and AdaptiveCPP except that AdaptiveCPP is, for some as yet unknown reason, significantly quicker for Algorithm 2.3.

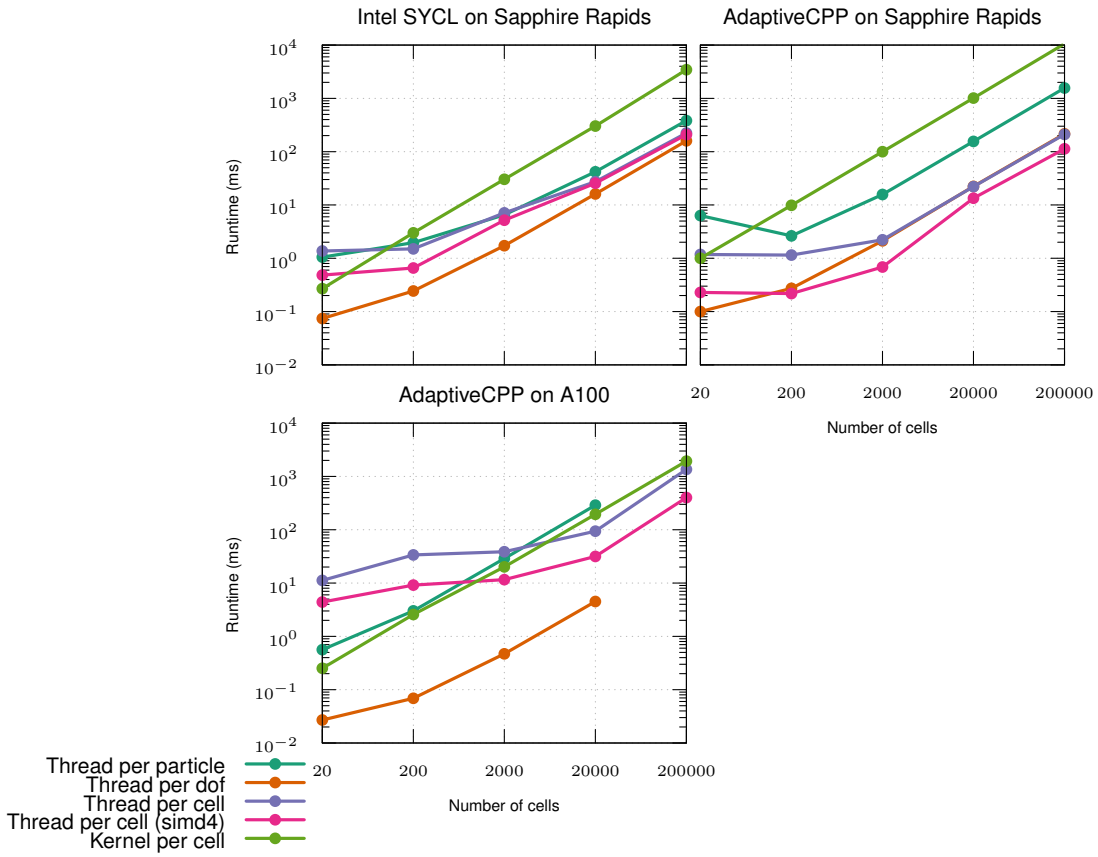


Figure 1: Comparison of algorithms - 2000 quadrilateral cells, 5000 particles per cell, polynomial order 4 (5 modes). Sapphire rapids nodes are dual socket 56 core per socket

Strong scaling Figure 2 and Figure 3 show again that the Algorithm 2.5 and Algorithm 2.3 are quicker. They also demonstrate the strong dependence on the problem structure. Both bench-

[2]Either a 112 core sapphire rapid CPU (56 core per socket) or a single A100 GPU

marks process the same total number of particles (10,000,000) but in Figure 3 we see worse performance probably due to under-utilisation of hardware resources.

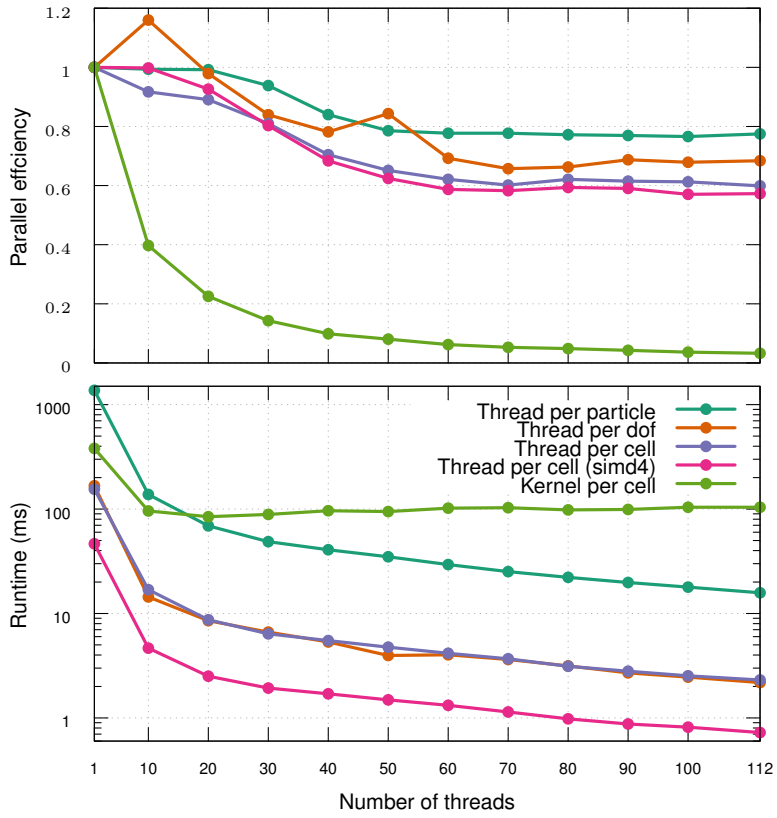


Figure 2: AdaptiveCPP: 2x56 core sapphire rapid 2000 quadrilateral cells, 5000 particles per cell, polynomial order 4 (5 modes)

2.5.3 Template function to evaluate modified basis functions

In the previous section each algorithm calls a `calculate_basis` function. Using C++ templates we can replace this function with one that pre-calculates as much as possible at compile time. For example this calculation heavily rely on computing Jacobi polynomials, these can be defined recursively,

$$a(\alpha,\beta,n)P_n^{\alpha,\beta}(z) = b(\alpha,\beta,n)P_{n-2}^{\alpha,\beta}(z) + (c(\alpha,\beta,n)z + d(\alpha,\beta,n))\,P_{n-1}^{\alpha,\beta}(z), \tag{24}$$

where the coefficients a, b, c and, d only depend on α, β and n. As α, β and n are known at compile-time, a C++template function can pre-calculate the coefficients and automatically unwind the recursion. This should *in principle* be more efficient - at the cost of longer compile times. A template function corresponding to Equation 24 can be defined, e.g

```
template <typename T, int64_t n, int64_t alpha, int64_t beta>
auto jacobi([[maybe_unused]] T const z) {/*details omitted*/}
```

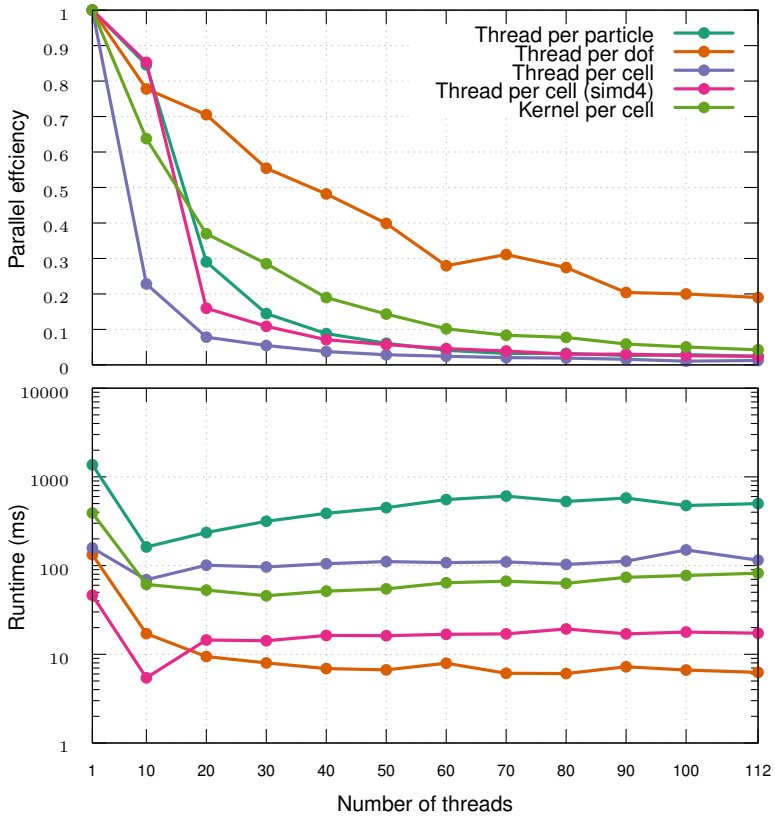


Figure 3: AdaptiveCPP: 2x56 core sapphire rapid 20 quadrilateral cells, 500,000 particles per cell, polynomial order 4 (5 modes)

, and used as a building block to construct function templates that calculate the modified basis functions e.g. Algorithm 2.6. Using this function to calculate the modified basis for quadrilateral elements there is an approximate 30-40% speed-up in the projection function.

2.5.4 Solver results

Algorithm 2.3 and Algorithm 2.5, have been implemented in NESO, using template functions to evaluate the basis functions. Table 1 lists shows the speed-up achieved in NESO's SimpleSOL and Electrostatic2D3V solvers using the alternative algorithms. [3]. Clearly there is a significant speed-up - to the degree that the function itself is no longer a significant fraction of the total runtime.

[3]Note these numbers are from a basic laptop so may not extend perfectly to a HPC system - build problems are currently preventing running on HPC system

```
template <typename T, int64_t N, int64_t alpha, int64_t beta>
T eModA(T z) {
  if constexpr (N == 0)
    return 0.5 * (1.0 - z);
  else if constexpr (N == 1)
    return 0.5 * (1.0 + z);
  else
    return 0.25 * (1.0 - z) * (1.0 + z) *
           jacobi<T, N - 2, alpha, beta>(z);
}
```

Algorithm 2.6

		Speed-up TpCell		Speed-up TpDOF	
Solver name	Example	Runtime	Projection	Runtime	Projection
SimpleSOL	2DWithParticles	2.99	349	2.80	29.2
Electrostatic2D3V	two_streams	2.31	90.0	2.18	28.0

Table 1: Speed-up on desktop (4 threads), TpCell - Thread per cell (Algorithm 2.3), TpDOF - Thread per DOF (Algorithm 2.5). Input files are as in the `example` folder in the NESO repository. Both problems use quadrilateral cells, `2DWithParticles` uses 4 modes (polynomial order 3) and `two_streams` uses 2 modes (polynomial order 1)

2.5.5 Conclusions and future work

Having identified a significant hotspot in NESO we have developed and benchmarked alternative algorithms to achieve a significant speed-up. The manually vectorised (labelled "simd4" in the Figures) version of Algorithm 2.3 is consistently faster (see all Figures in this section) - at least using AdaptiveCPP. This suggests that there may be scope to further optimize Algorithm 2.5 further via the same technique. Note that SYCL vector types [4] combined with C++templating mean explicit vectorisation is straightforward - at least compared to compiler intrinsics or ASM. Currently only quadrilateral elements are supported and work is ongoing to add support for all element types. Once build issues are resolved on HPC systems further benchmarking is required to confirm whether the thread per DOF kernel is superior on GPU and for large thread counts as suggested by the benchmarks in Section 2.5.2.

3 Task Work by Grantees

Each section summarises reports received from a grantee, one report per subsection, with a link in the title to the location of the accepted report. 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 Reports received under Grant T/AW087/22, PO 2068625

The STFC grantee produced the following reports

3.1.1 Report 2068625-TN-01 [5]

Report 2068625–TN–01 is a minor revision of Report 2047353–TN–03, previously summarised and bound within [6].

3.1.2 Report 2068625-TN-02 [7]

Report 2068625–TN–02 is a minor revision of Report 2047353–TN–04, previously summarised and bound within [6].

3.1.3 Report 2068625-TN-03 [8]

Report 2068625–TN–03 is a review of time-stepping techniques and preconditioning for hyperbolic and anisotropic elliptic problems, treating with some 90 previous works on the topics. The work on elliptics is pursued in ref [9] and described below under the TN-04 heading. The TN-03 summary of results for time-stepping is the basis for the following synopsis:

- Time-stepping using Diagonally-Implicit Runge–Kutta (DIRK), ie. methods with a lower triangular coefficient matrix, where applicable, is well developed and the review by Kennedy and Carpenter (ref.[45] in report) provides a nice summary.
- Fully implicit Runge–Kutta methods were largely dismissed as impractical for hyperbolic problems, but recent advances in preconditioning techniques may mean this is no-longer the case; Southworth et al. (refs.[83, 82] in report) describe (and advance) the state-of-the-art here.
- The theory of ImplicitExplicit (IMEX) methods is also well developed and should be used where appropriate.

- Recent work on parallel-in-time methods may help break the inherently sequential nature of traditional time integration algorithms, which rely exclusively on the solution of the spatial discretization for parallelism.
- All approaches for solving implicit time-stepping problems require good methods for solving at least one variant of the stationary system as reviewed in Section 3 of the document and pursued in the subsequent report 2068625-TN-04 below.

3.1.4 Report 2068625-TN-04 [9]

Report 2068625–TN–04 numerically examines the time-stepping techniques and preconditioning methods which were highlighted above in TN-03 [8] as most relevant to the hyperbolic and anisotropic elliptic problems arising in Project NEPTUNE. The summary section from TN-04 is the basis for the following

- The most promising method for solving matrices from high order elliptic PDEs is to precondition with a low order finite element operator: low-order solvers have become robust enough to solve the resulting (challenging) linear systems, see Pazner, Kolev and Dohrmann, (2023, ref.[̃30] in report).
- Multigrid and Domain Decomposition-based preconditioners are the leading contenders for performant parallel iterative linear solvers on anisotropic elliptic problems of low order.
- There are several excellent algebraic multigrid packages available that are highly parallel. However, to get good performance, it is vital to tune the parameters for a given problem. A well-implemented geometric multigrid method would give superior performance to an algebraic multigrid.
- The software package HPDDM contains spectral multilevel overlapping domain decomposition methods based on GenEO which seem promising with a good trade-off between performance and ease of setups. (The GenEO method of Spillane et al. introduces a spectral coarse space into domain decomposition approaches as described in TN-03 [8].) The recent development of fully algebraic variations by H. Al Daas et al. (2022,2023, refs.[̃3, 2] in report) make this even more the case. However, implementation improvements for these methods are needed for them to compete against highly-tuned multigrid implementations.
- Higher order fully implicit time-stepping schemes show spectacular performance in terms of run time and accuracy when compared to other higher-order time-stepping schemes.

3.1.5 Report 2068625-TN-05 [10]

Report 2068625–TN–05 discusses techniques and software relevant to the coupling of continuum (fluid) and particle models of plasma for NEPTUNE. It is a revised version of 2060049–TN–02, which was summarised in ref [1]. The most significant addition consists of Appendix A, together with two linking paragraphs at the end of Section 3.6, to describe implementation of code coupling between toy codes and NEKTAR++ using CWIPI. The test cases seem to have been carefully

chosen to demonstrate economically correct implementation of the different mechanisms, and the figures nicely present the results The results in Figure 7 in respect of particle noise are pleasantly optimistic, but this has to be caveated by the presence of a relatively uniform distribution of particle locations at nodes of a mesh.

3.1.6 Report 2068625-TN-06 [11]

Report 2068625–TN–06 discusses implementation and scalability analysis of coupling continuum (fluid) and particle models of plasma for NEPTUNE. It continues the work started as described in Appendix A of the note TN-05 [10], to describe implementation of code coupling between NESO-PARTICLES and NEKTAR++ using CWIPI. The test case has been carefully chosen as an example of the type of coupling expected in a NEPTUNE application. Studies are performed on Hartree Centres Scafell Pike supercomputer using Skylake nodes, of the scaling of the speedup and efficiency. CWIPI is found to add negligible cost to execution, the main bottleneck which reduces speed by a factor of approximately two, is found by separate NEKTAR++ calculation to be the pressure and viscous solves. It is concluded that improved preconditioners should be used in these solvers as most likely to reduce overall code execution times. In addition, the introduction includes a small number of references to coupling in the literature on magnetic fusion and spacecraft re-entry.

3.1.7 Report 2068625-TN-07 [12]

Report 2068625–TN–07 gives an in-depth review of techniques for numerical bifurcation analysis. In the NEPTUNE project, nonlinearity is expected frequently to lead to short scalelengths that will make the computation of isolated solution very challenging and possibly of suspect validity. In any event, design problems require an understanding of how solutions vary with parameters, and in particular those points at which solutions fail to be stable or even exist. Section 1 explains the importance of bifurcation analysis in understanding the *system* behaviour of nonlinear problems, which may possess many branches of both steady and oscillatory solutions, and is a wide-ranging introduction to the subject. Subsequent sections go into further details on aspects expected to be relevant to NEPTUNE.

Section 2 discusses in seven sub-sections the now well- established methods for continuation of an initial solution into a steady solution branch. The initial solution may have been discovered numerically, for example as the result of an instability due to say increasing thermal effects or by use of an approximate analytic solution, at parameters where it was possible to establish authenticity by h/p refinement. Deflation techniques described in Section 3 complement those of the preceding section, by helping to find and continue potentially disconnected branches of solutions. Section 4 describes how branches of time-dependent solutions may be discovered, and introduces the dynamical systems point-of-view. The latter classifies the system behaviours to be expected generically when for example there is an elementary bifurcation of a steady state or to an oscillatory solution *aka* periodic orbit. Symmetry plays an important role in constraining system behaviour and Section 5 deals with how this factor may be best accounted for. Thereafter, available 'bifurcation' software is considered and key aspects tabulated in Section 6, focusing on those

codes capable of tackling the larger-scale problems arising from PDEs. Lastly Section 7 summarises important implications for NEPTUNEif it is to discover system behaviour by the methods described in the report.

3.1.8 Report 2068625-TN-08 [13]

Report 2068625–TN–08 begins with a description in Section 2, of the basic theory illuminating the utility of the adjoint, and thereafter concentrates on relatively recent (most within the last decade) literature concerning algorithmic developments in Sections 2.1 and 2.2, with 2.2.1 covering parallel-in-time and 2.2.2 the use of checkpoints in unsteady problems. The latter is notable as data compression is needed to make the problem tractable, and both down-sampling and SVD-like approaches have been explored with the iSVD method found to be unexpectedly accurate.

The applications cited in Section 3 have been chosen to be at least tangentially and frequently directly relevant to plasma physics modelling, and are dealt with under the headings of 3.1 Sensitivity and stability analysis, 3.2 Parameter optimisation and sensitivity, 3.3 Uncertainty quantification, 3.4 Adaptive mesh refinement, and 3.5 Shape optimisation. The last includes work by W.Dekeyer on the shaping of divertor surfaces for improved power-handling. Reported results tend to discourage use of adjoints in Monte-Carlo/particle problems. The work constitutes a very valuable review of the state-of-the-art, with some $70+$ citations carefully entered.

3.1.9 Report 2068625-TN-09 [14]

Report 2068625–TN–09 demonstrates application of adjoints to an illustrative problem, namely slot convection, using a specially developed proxyapp built on the NEKTAR++ library. The material relies heavily on the preceding review TN-08, but the report is largely self-contained. The adjoint approach is used to calculate the sensitivity of the linear stability of the convecting flow, ie. the eigenvalue of the most unstable mode becomes the objective for optimisation, subject to one of two thermal horizontal boundary conditions (conducting profile or insulating). The numerics are carefully validated and it is instructive to see the abstract theory of TN-08 in application.

Time-dependent adjoint systems of equations are explicitly set out in Appendix A for solution by NEKTAR++, with Appendix B containing the proxyapp interface. Direct-product multiplication of the direct solution and its adjoint shows where the unstable mode is likely most sensitive to forcing, contrasting the effect of the different thermal boundary conditions. Thereafter, the adjoint approach is applied to understanding the effect on the eigenvalue of changes to the temperature field imposed on the hot wall. Allowed changes are separately of two types, either a steady imposed perturbation (in fact considered after the other option) or a general linear perturbation, both perturbations being functions of position. In either case the most critical position for perturbation is identified.

3.1.10 Report 2068625-TN-10 [15]

Two sections 6 and 7, and 5 more citations have been added to the report 2060049–TN–03 which which was previously summarised and bound within [6]. These new sections consider classical fluid flow problems as follows:

1. Steady 2-D and 3-D lid-driven cavity
2. Steady 2-D 'slot convection'
3. Sinusoidally time-dependent 2-D flow past driven past a cylinder in a channel.

using Firedrake software which represents application of geometric multigrid ideas in the context of possibly unstructured meshes. The first case is particularly valuable for investigating a common scenario expected in application, whereby a sequence of solutions is calculated at smaller and smaller viscosity values, using the final state of the previous one to initialise the next. In both 2-D and 3-D variants, the cost as measured by number of additional iterations required at the new value of ν, remains remarkably small, under ten in nearly all cases. There are caveats that here uniform gridding is used and the solutions at different ν resemble each other qualitatively.

The results presented for the slot convection case relate only to a fixed pair of parameter values and again employ a rectangular grid, but usefully demonstrate that the proposed algorithm scales well for up to 128 cores in both weak and strong senses, implying that communication costs remain minor. The cylinder case demonstrates for a unstructured mesh on a single compute node, that the number of additional iterations at each timestep is close to one, although it is disappointing that each timestep only represents $1/6400^{th}$ period when a practical calculation might need to advance by steps which are hundreds of times larger.

3.2 Reports received under Grant T/AW088/22, PO 2067270

The York grantee produced the following reports

3.2.1 Report 2067270-TN-01 [16]

The report 2067270–TN–01 has been constructed as a series of incremental updates to an initial submission. The changes at each update largely consist of short multiline additions to the previous text, but in any case are accurately described in the Changelog, reflecting the history of a rapidly changing field.

As we enter the Exascale era with the noticeable exception of Fugaku all machines will be made of heterogeneous nodes containing both a multi-core cpu and a gpu. NEPTUNE will be required to be performance portable, targeting these multiple all these cpu and gpu architectures, while maintaining high levels of performance across them. Several programming models exist to parallelise code, but as of yet there is no definitive frontrunner which shows itself as the best candidate for use at the exascale. Therefore, the objective of this report is to investigate the effectiveness of

differing programming model used for parallelisation to achieve performance portable code. This was achieved by evaluating several fluid and particle mini-codes using a variety of performance benchmarks, across different hardware architectures and compilers.

The key findings of this investigation found that:

- the majority of the NEPTUNE codebase should be written in modern C++, for reasons such as that most mathematical and scientific libraries are written in C/C++ providing NEPTUNE with a native API, the ability to do templating and meta-templating, and new parallel programming models usually target C/C++ first
- While vendor specific parallelisation techniques can achieve higher performance than open standards, they suffer from the disadvantage of producing code which can only be run on particular hardware, making them unportable. The disadvantage of open standards are that they general take longer to target newer hardware and hardware features.
- NEPTUNE will likely require both low level and high level DSLs. The higher level DSLs will need to be chosen or developed to allow scientists to be able to accurately represent their science without being overly restrictive, so new operators and features can be added in future
- Data storage should be abstracted away from algorithms as much as possible
- Code such be modularised as much as possible to allow key kernels which are performance critical to be re-evaluated in isolation as new technologies emerge
- Where possible code should be reused including external libraries. Since vendor-optimised versions of these libraries exist, performance improvements can be found for free.

3.2.2 Report 2067270-TN-02 [17]

The report 2067270-TN-02 has been constructed as a series of incremental updates to an initial submission. The changes at each update largely consist of short multiline additions to the previous text, but in any case are accurately described in the Changelog, reflecting the history of a rapidly changing field. However at the seventh revision, the original Sections 2 and 3 have been substantially altered to form a MS for a now accepted submission to a journal (CPC) as carefully and accurately described in the Changelog.

There have been many changes to the hardware makeup of supercomputers as they have reached ever higher levels of computing performance. Be it the move towards to first mainstream multi-core cpus in 2004, to the move from cpu-only compute nodes to the majority of machines now having both a cpu and gpu. The UKRI intends to build the UK's first Exascale supercomputer by 2025. This report details the hardware that NEPTUNE is likely to encounter both pre- and post-Exascale. This report is split into three parts. The first part outlines the hardware currently deployed or scheduled to be delivered in next five years according to each vendor's respective roadmap. A noticeable change to these these roadmaps since the last revision of this report is the cancellation of Intel's Rialto Bridge GPU, and subsequent replacement by Falcon Shores GPUs. The outline provided includes details such as the core count range for a particular cpu range,

supported memory technology used, peak double precision performance, amongst others. The second part outlines what machines are currently being built or proposed within Europe and the USA, and elsewhere. The final part details both homogeneous and heterogenous machines currently available with which the performance portability of codes developed as part of NEPTUNE could be evaluated.

3.2.3 Report 2067270-TN-03 [18]

The report 2067270–TN–03 considers the development of a DSL for electrostatic Particle In Cell (PIC) in the context of the authors' previous construction of the OP2 DSL for problems with unstructured meshes. Based on OP2, a DSL called OP-PIC is developed.

A section contains description of the transfer of particle ownership between MPI ranks. Two approaches are described, a neighbour based "multi-hop" approach and a "direct-hop" approach. The multi-hop approach repeatedly sends particles to neighbouring ranks until and owner is found. The direct-hop approach combines a global move approach with the neighbour based multi-hop approach. The global move approach appears to be identical to that already described in NESO, in algorithm and implementation. Preliminary results from OP-PIC based code include comparisons with alternative implementations and strong/weak scaling, see the summary of TN-05 below for more details.

3.2.4 Report 2067270-TN-04 [19]

The Report 2067270-TN-04 contains material which has subsequently been extended to form the TN-06 report which is summarised below.

3.2.5 Report 2067270-TN-05 [20]

The report 2067270–TN–05 presents performance and scaling comparisons for a Finite Element Method (FEM) Particle In Cell (PIC) proxyapp used as a (micro-)benchmark for the OP-PIC DSL developed in TN-03. The MINI-FEM-PIC proxyapp implements 3-D electrostatic PIC on an unstructured mesh of tetrahedral cells forming a rectangular duct. The report investigates both the overall runtime and the charge deposition process on different architectures for a case with $48\,000$ cells and up to 70 million particles. The charge deposition process is particularly interesting as this operation typically involves memory-write contention which must be approached in an architecture dependent manner for application to NEPTUNE.

Specifically, the work compares a single CPU node with a single GPU, which is reasonable as these two categories of hardware have roughly comparable power draw. The results presented demonstrate an improveed or roughly equivalent performance per Watt on GPU platforms in comparison to the CPU platforms. The weak scaling results obtained appear perfectly reasonable.

3.2.6 Report 2067270-TN-06 [21]

The report 2067270-TN-06 is a much extended version of TN-04, describing the solution of the plasma fluid models of System 2-7 described in the Equations document [22, § 9.1]. It treats two systems: the `Blob2D` problem, in which a two-dimensional electron density blob evolves under what is basically a centrifugal force, and the Hasegawa-Wakatani (HW) system of equations describing drift-wave turbulence in three dimensions. The equation systems are followed by a discussion of the implementations of `Blob2D` in HERMES-3 and the Hasegawa-Wakatani equations in BOUT++, in particular the 3-D version of HW.

The 2-D blob problems are evolved with and without additional imposed dissipation, showing the importance of the treatment of small spatial scales in any code comparison. Three choices of the adiabaticity parameter α, which measures the coupling between plasma number density and electric potential fluctuations, are simulated in the existing BOUT++ implementation of the HW3D equations and the outputs are illustrated graphically and in terms of the total energy and generalized enstrophy time-series, as well as the power spectral density of the fluctuations in the density field. The main utility of this work for NEPTUNE is that the outputs can be compared with those obtained from an implementation of the same model in the UKAEA NESO software in order to provide validation as well as an initial performance comparison. Supplementary GitHub material in repo `https://github.com/mikekryjak/neptune-docs` includes animations showing the Hasegawa-Wakatani outputs for the three representative choices of parameters. Useful references to the literature are included.

Acknowledgement

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

References

[1] W. Arter, E. Threlfall, J. Cook, M. Barton, O. Parry, and W. Saunders. Complementary actions. Code-coupling, physics databases and benchmarking techniques 2. Technical Report CD/EXCALIBUR-FMS/0075-M7c.2, UKAEA, 3 2023. https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/ukaea_reports/CD-EXCALIBUR-FMS0075-M7c.2_and_M8c.1.pdf.

[2] C. Cercignani. *Theory and application of the Boltzmann Equation*. Scottish Academic, 1975.

[3] ExCALIBUR Project NEPTUNE website. https://excalibur-neptune.github.io/Developers-Website/main/main.html, 2023. Accessed: January 2023.

[4] Sycl 2020 specification (revision 8). https://registry.khronos.org/SYCL/specs/sycl-2020/pdf/sycl-2020.pdf.

[5] M. Abalenkovs, V. Alexandrov, E. Sahin, N. Bootland, and S. Thorne. Preconditioners for the NEPTUNE Programme: Numerical Results. Technical Report 2068625-TN-01, UKAEA Project Neptune, 2024. https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/2068625/TN-01.pdf.

[6] W. Saunders and W. Arter. Identification of suitable preconditioner techniques. Technical Report CD/EXCALIBUR-FMS/0045-M2.7.2, UKAEA, 8 2021. https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/ukaea_reports/CD-EXCALIBUR-FMS0045-M2.7.2.pdf.

[7] M. Abalenkovs, V. Alexandrov, E. Sahin, N. Bootland, and S. Thorne. Implicit-factorization preconditioners for non-symmetric problems. Technical Report 2068625-TN-02, UKAEA Project Neptune, 2024. https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/2068625/TN-02.pdf.

[8] H. Al Daas, N. Bootland, T. Rees, and S. Thorne. A Review of Time Stepping Techniques and Preconditioning for Hyperbolic and Anisotropic Elliptic Problems. Technical Report 2068625-TN-03, UKAEA Project Neptune, 2024. https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/2068625/TN-03.pdf.

[9] H. Al Daas, N. Bootland, T. Rees, and S. Thorne. Time Stepping Techniques and Preconditioning for Hyperbolic and Anisotropic Elliptic Problems: Numerical Results. Technical Report 2068625-TN-04, UKAEA Project Neptune, 2024. https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/2068625/TN-04.pdf.

[10] H. Al Daas, N. Bootland, T. Rees, A. Sunderland, J. Williams, P. Rubin, and S. Thorne. Techniques and software relevant to the coupling of continuum (fluid) and particle models of plasma for NEPTUNE. Technical Report 2068625-TN-05, UKAEA Project Neptune,

2024. https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/2068625/TN-05.pdf.

[11] J. Williams and S. Thorne. Implementation and scalability analysis of coupling continuum (fluid) and particle models of plasma for NEPTUNE. Technical Report 2068625-TN-06, UKAEA Project Neptune, 2024. https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/2068625/TN-06.pdf.

[12] N. Bootland, T. Rees, and S. Thorne. A Review of Solution Continuation Techniques: Methods and Software. Technical Report 2068625-TN-07, UKAEA Project Neptune, 2023. https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/2068625/TN-07.pdf.

[13] J. Williams, U.A. Quadri, and S. Thorne. Review of adjoint method in plasma physics for NEPTUNE. Technical Report 2068625-TN-08, UKAEA Project Neptune, 2023. https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/2068625/TN-08.pdf.

[14] J. Williams, U.A. Quadri, and S. Thorne. Adjoint-based sensitivity analysis of a differentially-heated cavity. Technical Report 2068625-TN-09, UKAEA Project Neptune, 2024. to be posted at https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/2068625/TN-09.pdf.

[15] H. Al Daas, N. Bootland, T. Rees, and S. Thorne. Time Stepping Techniques and Preconditioning for Hyperbolic and Anisotropic Elliptic Problems: Numerical Results. Technical Report 2068625-TN-10, UKAEA Project Neptune, 2024. to be posted at https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/2068625/TN-10.pdf.

[16] S. Wright, C. Ridgers, E. Higgins, B. Dudson, P. Hill, D. Dickinson, G. Mudalige, B. McMillan, and T. Goffrey. Hardware at the Exascale Revision 7.0. Technical Report 2067270-TN-01, UKAEA Project Neptune, 2024. to be posted at https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/2067270/TN-01.pdf.

[17] S. Wright, C. Ridgers, E. Higgins, B. Dudson, P. Hill, D. Dickinson, G. Mudalige, Z. Lantra, B. McMillan, and T. Goffrey. Developing an Exascale-Ready Fusion Simulation Revision 7.0. Technical Report 2067270-TN-02, UKAEA Project Neptune, 2024. to be posted at https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/2067270/TN-02.pdf.

[18] S. Wright, E. Higgins, C. Ridgers, G. Mudalige, and Z. Lantra. Progress towards DSL adoption Revision 2.0. Technical Report 2067270-TN-03, UKAEA Project Neptune, 2024. to be posted at https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/2067270/TN-03.pdf.

[19] C. Ridgers and M. Kryjak. Benchmarks for basic turbulence cases (Report on Test Cases and Proxy-App). Technical Report 2067270-TN-04-2, UKAEA Project Neptune, 2024. to be posted at https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/2067270/TN-04-2.pdf.

[20] Z. Lantra, G. Mudalige, S. Wright, E. Higgins, and C. Ridgers. DSL Mini-application Performance Report Revision 1.0. Technical Report 2067270-TN-05, UKAEA Project Neptune, 2024. to be posted at https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/2067270/TN-05.pdf.

[21] C. Ridgers and M. Kryjak. Benchmarks for basic turbulence cases (Report on Test Cases and Proxy-App) - updated . Technical Report 2067270-TN-06, UKAEA Project Neptune, 2024. to be posted at https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/2067270/TN-06.pdf.

[22] W. Arter et al. Equations for EXCALIBUR/NEPTUNE Proxyapps. Technical Report CD/EXCALIBUR-FMS/0021-1.31-M1.2.1, UKAEA, 10 2023. https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/ukaea_reports/CD-EXCALIBUR-FMS0021-1.30-M1.2.1.pdf.

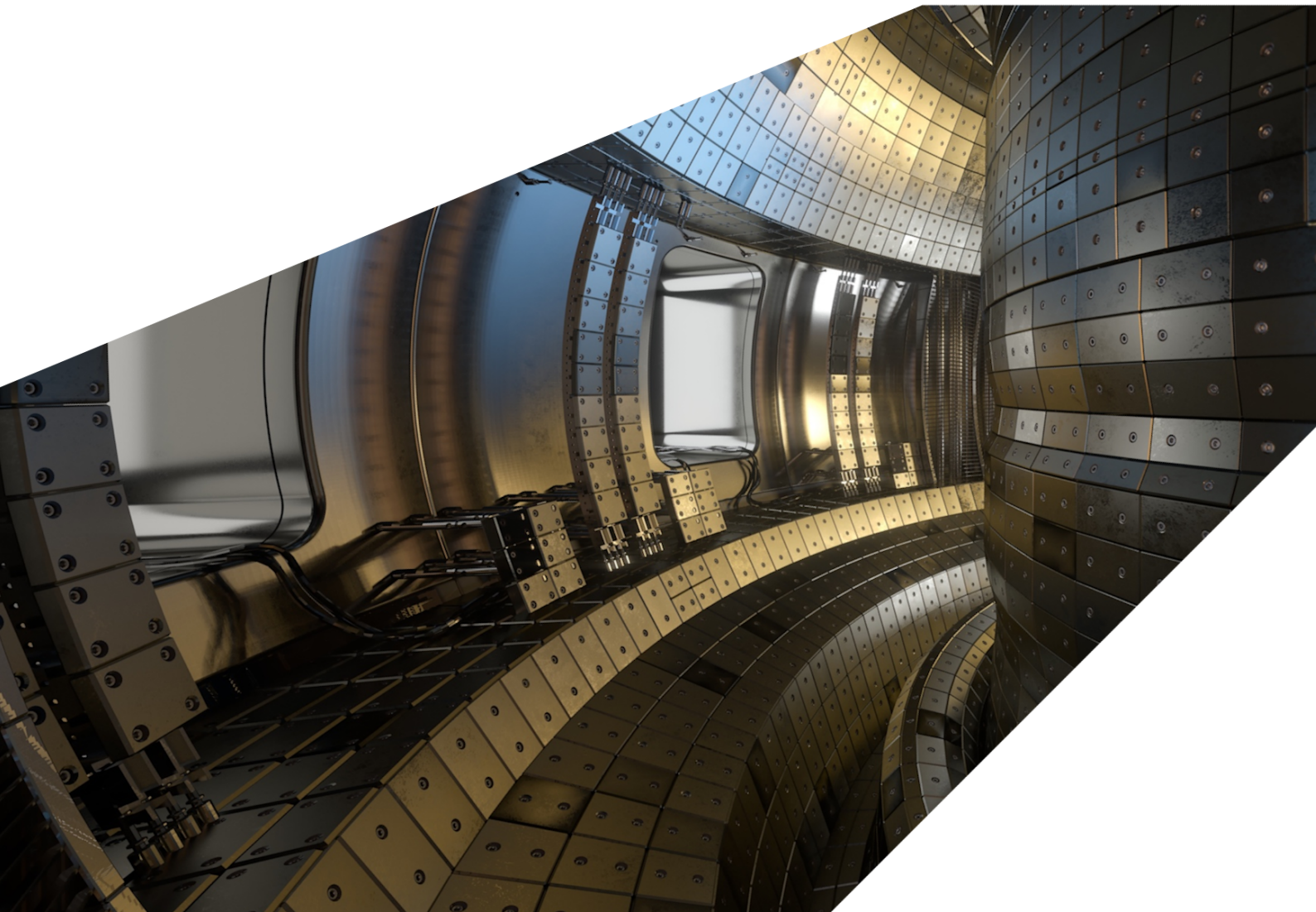
UK Atomic Energy Authority

ExCALIBUR

Complementary actions. Atomic data usage 2.

M8c.2 Version 1.00

Abstract
The report describes work for ExCALIBUR project NEPTUNE at Milestone M8c.2. This report describes how charge exchange has been implementated into NEPTUNE. It will go through the theory of charge exchange, approximations made within this work, and their associated errors.



UKAEA REFERENCE AND APPROVAL SHEET

	Client Reference:	
	UKAEA Reference:	CD/EXCALIBUR-FMS/0086
	Issue:	1.00
	Date:	March 15, 2024
Project Name: ExCALIBUR Fusion Modelling System		

	Name and Department	Signature	Date
Prepared By:	Matthew Barton CD	N/A	March 15, 2024
Reviewed By:	James Cook CD		March 15, 2024

1 Introduction

Charge exchange is the process in which an electron is transferred between between an atom and an ion during a collision. The charge exchange process that has been implemented in NESO for this report supports Hydrogen, and can be written as

$$H + H^+ \rightarrow H^+ + H. \tag{1}$$

If the ion is only singly charged, such as the above interaction, then the density of the ion population is unchanged by the process since there outcoming charge states must be the same as those incoming. However as the momenta of the individual hydrogen neutrals and hydrogen ions may differ, then the momentum of the ions in the plasma may change as a result of this process. This simple example therefore shows that charge exchange provides a mechanism for the fluid and particles to exchange momentum without affecting their respective densities. This report will outline a method the implementating charge exchange into NESO by having particles undergo charge exchange, then merging them to keep the problem computationally tractable. The next chapter is split into 3 sections: the first is the theory behind the implementation; the second discusses the error associated with this method; the final part discusses the results of testing this implementation.

2 Task Work

The purpose of this section is to give a more detailed breakdown of the work undertaken for Milestone M8c.2.

2.1 Theory

There are many ways to model charge exchange within a particle-in-cell (PIC) code. The way it was implemented in this work relies on the merging of neutral macroparticles after a charge exchange event has taken place. The rate of change of weight of the j^{th} macroparticle undergoing charge exchange is

$$\dot{w}_{CE,j} = R_{CE,j}\, w_j\, n_{ions} \tag{2}$$

where $\dot{w}_{CE,j}$ is has units of per unit time, $R_{CE,j}$ is the charge exchange rate, w_j is the weight of the j^{th} particle with dimensionless units, and n_{ions} is the plasma ion density at the particle location with units of per cubic length.

Every time a charge exchange process takes place (i.e. once per particle at every timestep Δt), one creates a new macroparticle with a weight determined by the above equation. Since it is an exchange process, the newly created neutral particle takes velocity of the ion it interacted with, and takes on a weight determined by Eq. 2 multiplied by Δt. This weight is subtracted from that of incoming neutral macroparticle and consequently the total weight of neutrals is conserved as expected by the exchange process. The critical issue with this approach is that it generates twice the number of macroparticles at each call, which left unchecked would rapidly cause the program to run out of memory. Another point to note is that many macroparticles have small weights and thereby have little effect on the physics, whilst still taking up just as much memory. The solution is to arrive at a strategy to merge particles. We implement an alternative method, which combines the macroparticle that undergoes a charge exchange event with the newly created macroparticle. This is done in such a way as to conserve the total momentum of the pair of merged macroparticles. The transformation of the j^{th} particle's velocity v_{pj} is

$$\frac{dv_{pj}}{dt} = -\dot{w}_{CE,j}\frac{v_{pj}}{w_j} + \dot{w}_{CE,j}\frac{v_f}{w_j}, \tag{3}$$

where v_f is the fluid velocity.

The evolution of the velocity of the elemental volume δV of fluid through which the particles are passing is therefore governed by

$$\frac{dv_f}{dt} = \sum_j \left(\dot{w}_{CE,j}\frac{v_{pj}}{\delta V n_{ions}} - \dot{w}_{CE,j}\frac{v_f}{\delta V n_{ions}} \right). \tag{4}$$

If we now sum equation 3 over j we get

$$\frac{d < v_p >}{dt} \;=\; -R_{CE,n} n_{ions} < v_p > + R_{CE,n} n_{ions} v_f, \tag{5}$$

where $< v_p >$ is the average velocity of all the macroparticles. Writing out w explicitly it can also be seen that 4 can be written as follows

$$\frac{dv_f}{dt} = R_{CE,n}(n_{neutrals} < v_p > -n_{neutrals} v_f). \tag{6}$$

It is easy to see by comparing Eqs. 5 and 6 that the systems conserves momentum for like masses.

2.2 Error Analysis

Although the approximation above makes the problem numerically tractable it does come with the drawback that it causes the energy of the fluid to be calculated incorrectly. The objective of this subsection is to derive an analytical formula for this error and to present numerically derived values for the error at equilibrium.

The kinetic energy of an average ion drawn from a thermal distribution is

$$\frac{m}{2\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} v^2 e^{-\frac{1}{2}\left(\frac{v-\mu}{\sigma}\right)^2} dv, \tag{7}$$

where μ is the mean velocity, and σ is the particle thermal velocity defined by

$$\sigma \ = \ \sqrt{\frac{kT}{m}}. \tag{8}$$

Substituting $u = \dfrac{x-\mu}{\sqrt{2}\sigma}$,

$$K = \frac{m}{2\sqrt{\pi}} \int_{-\infty}^{\infty} (2\sigma^2 u^2 + 2\sqrt{2}\mu\sigma u + \mu^2) e^{-u^2} du, \tag{9}$$

recalling that the integral over ue^{-u^2} can be seen to be 0 by symmetry, the other components of the integral are as follows

$$\int_{-\infty}^{\infty} x^2 e^{-x^2} \ = \ \frac{\sqrt{\pi}}{2}, \tag{10}$$

and

$$\int_{-\infty}^{\infty} e^{-x^2} \ = \ \sqrt{\pi}, \tag{11}$$

leading to an expression for the average kinetic energy of an ion

$$K = \frac{m}{2}(\mu^2 + \sigma^2). \tag{12}$$

The total energy in the like-massed neutral macroparticles can be calculated by simply performing the sum

$$\frac{1}{2} m_{neutral} \sum_{j=0}^{N_{macroparticles}} w_j v_j^2. \tag{13}$$

After performing charge exchange we end up with double the number of macroparticles we began with, and if we add the kinetic energy of these particles separately, i.e. before amalgamation, we arrive at

$$E_{seperate} = \frac{m_{neutral}}{2} \sum_j \left((w_j - \delta_j) v_j^2 + \delta_j v_i^2 \right). \tag{14}$$

where we have introduced δ_j to represent the amount of weight changing hands between the incoming macroparticle and the ions. Now let's see what the kinetic energy of the combined particle is: remember we are merging particles in a manner that conserves momentum. The total kinetic energy after merging is

$$E_{combined} = \frac{m_{neutral}}{2} \sum_j w_j \left(\frac{(w_j - \delta_j) v_j + \delta_j v_i}{w_j} \right)^2. \tag{15}$$

Expanding this out one gets

$$E_{combined} = \frac{m_{neutral}}{2} \sum_j \left(\frac{(w_j - \delta_j)^2 v_j + 2\delta_j (w_j - \delta_j) v_j v_i + \delta_j^2 v_i^2}{w_j} \right). \tag{16}$$

It can clearly be seen that combining these macroparticles to conserve momentum has caused the kinetic energy of all the macroparticles to change, thereby adversely affecting energy conservation properties.

This change in kinetic energy is

$$E_{combined} - E_{seperate} = \frac{m_{neutral}}{2} \sum_j \left(\frac{(\delta_j - w_j)(v_i^2 + v_j^2 - 2 v_i v_j)}{w_j} \right), \tag{17}$$

and can be rewritten as

$$= \frac{m_{neutral}}{2} \sum_j \left((-1 + dt R_{CE,j} n_{ions}) \delta_j (v_j - v_i)^2 \right), \tag{18}$$

from which can easily be seen to be less than 0 due to enforcing $\delta_j < w_j$, which means that the collective kinetic energy of all the macroparticles decreases when upon combination of macroparticles to conserve momentum. If energy conservation is enforced, one may choose to increase the temperature of the fluid to make up the deficit.

2.3 Results

Before implementing this method of charge exchange into NESO[1] a prototype code was created to investigate the errors associated with the algorithm. This prototype can be found in Ref. [2]. This section will outline the results of what was found with this prototype code and what was needed to implement this method into NESO. As the objective of this method is to combine macroparticles while conserving momentum, a momentum conservation test was first devised to check how much

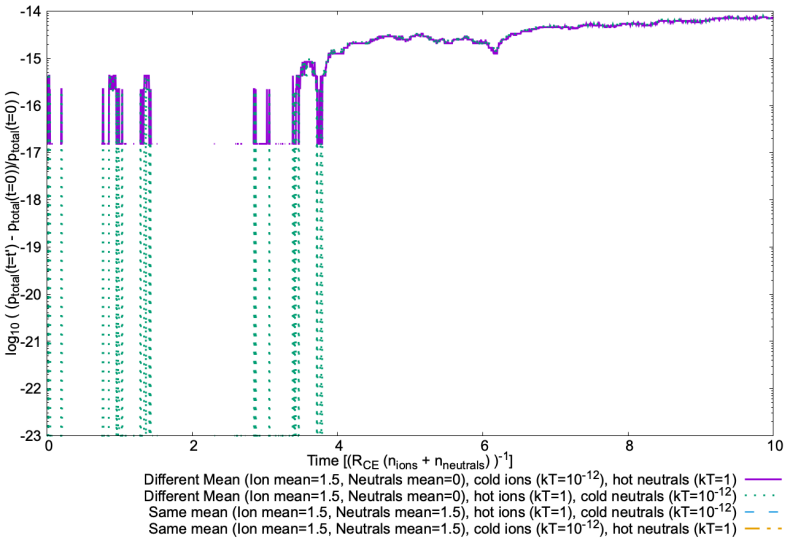


Figure 1: Normalised change in total momentum with respect to time (prototype). In the case where the fluid and macroparticles started with the same mean, the error was calculated to be 0. When the fluid and macroparticles start with different means the error is at the level of machine precision.

the momentum deviated from the initial momentum. The log of this fractional error is shown in Fig. 1. It can easily be seen that the method works as expected and the total momentum is conserved within machine precision.

Now that momentum was found to be conserved the next step was to verify that the calculated error in the energy of the plasma for a single timestep, was as expected by Eq. 18, for various initial conditions of the fluid and particles. This is shown in Fig. 2. The exact result can only be compared exactly at t=0 when both fluid and particles have Maxwellian distribution, and it was found to give general agreement. Although only a direct comparison can be compared at t=0, the calculated error associated with this technique reduces as the two populations reach equilibrium, but never reaches zero, which agrees with Eq. 18 that the error will always be non zero.

This result however doesn't give one an idea of the accumulated error up to that point, so next an attempt was made to estimate the total error up to time t as a consequence of merging macroparticles. This total error estimate was found by simply summing up the approximated error up to time t. This is shown in Fig 3. To determine whether this approximation is valid, the results were compared to a code, also in [2], where charge exchange was performed for 18 timesteps allowing the macroparticles to remain umerged. It was found that within this timeframe there is general agreement. Also, it was found that the total expected accumulated error when the system had reached equilibrium could vary significantly depending on the initial conditions.

Given the error on the fluid temperature can be large in certain circumstances, an attempt was made to see whether this error could be reduced. One test which was investigated was changing the ratio of the ion and neutral densities, while keeping the same initial conditions. The results of this search are shown in Fig. 4. The ratio of these densities was found to have an observable effect on the accumulated error, but not in a way which could be taken advantage of when implemented

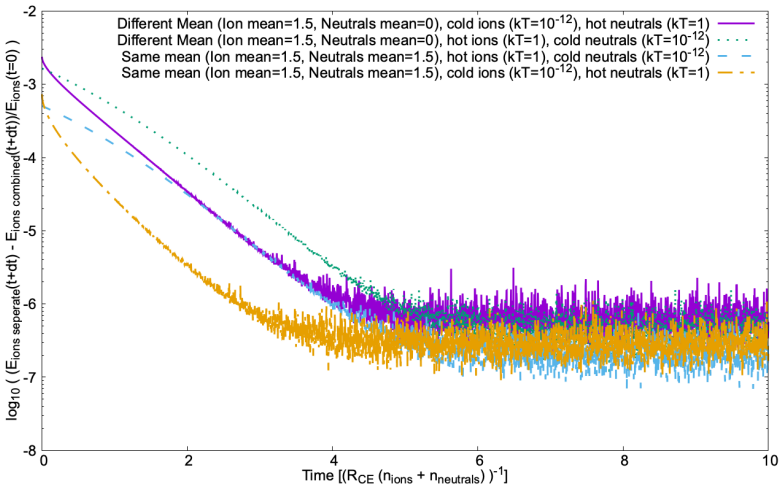


Figure 2: Instantaneous kinetic energy error as a function of time of the fluid when merging particles normalised by the error had the macroparticles been allowed to remain separate at the end of each time step. It shows that regardless of the initial conditions that the error never reduces to 0.

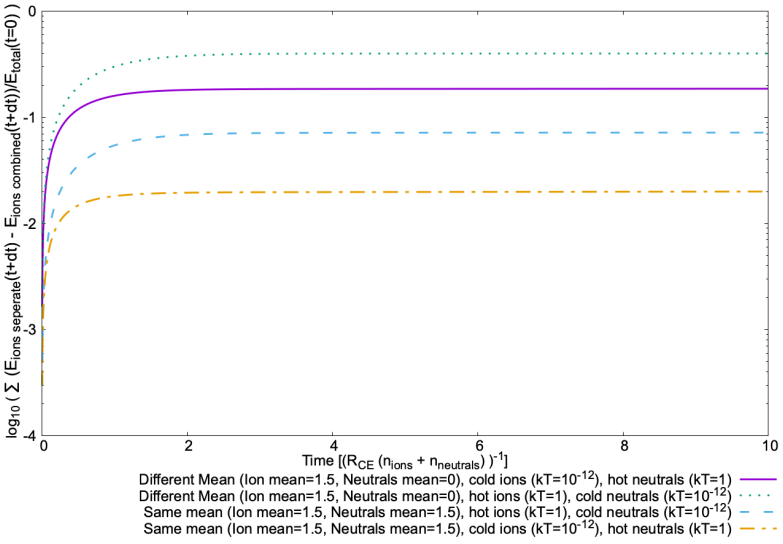


Figure 3: An estimate of the total accumulated error up to time t due to macroparticle merging.

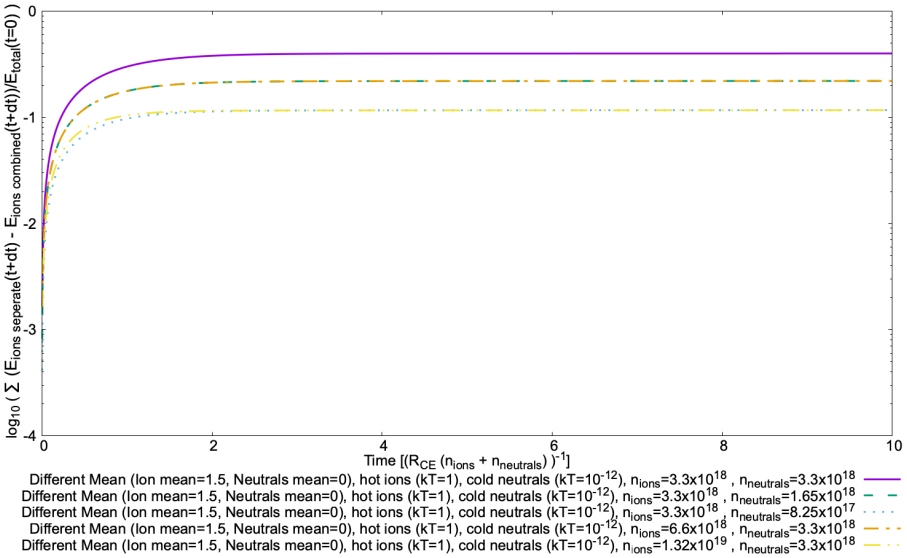


Figure 4: An estimate of the total accumulated energy error due to particle merging, relative to remaining separate at the end of each time step for different densities of ions and neutrals.

in NESO.

All results shown so far were performed using 2000 macroparticles, so the dependence of the error on the number of macroparticles was investigated. This results of this investigation are shown in Fig 5 and Fig. 6, covering both the circumstances where the fluid and particles have the different and same initial conditions respectively. Despite increasing the number of macroparticles by 2 orders it was found that the error would reduce only by negligible amount, and would not be worth the increased computational cost.

Once the particle merging method had been found to be viable through the prototype code, it was implemented into NESO. It was decided that the previously implemented SimpleSol solver would be extended to include charge exchange physics. To do this, first an estimate for the charge exchange rate $R_{CE,j}$ needed to be found. This was taken from Fig. 37 of Ref. [3]. The data from this figure was converted into a csv file using [4], which was subsequently read into NESO. As this data is discrete and the calculated kinetic energy of the neutrals in NESO is continuous, a linear interpolator was written. Similar to the prototype type, a momentum conservation test was implemented to check that charge change had been implemented correctly. The results of which are shown in Fig. 7. This figure shows two things: firstly that the total momentum is a constant of the simulation; and secondly that the long term trend of momentum is that the species end up with an equal share of that initialised in system, as expected.

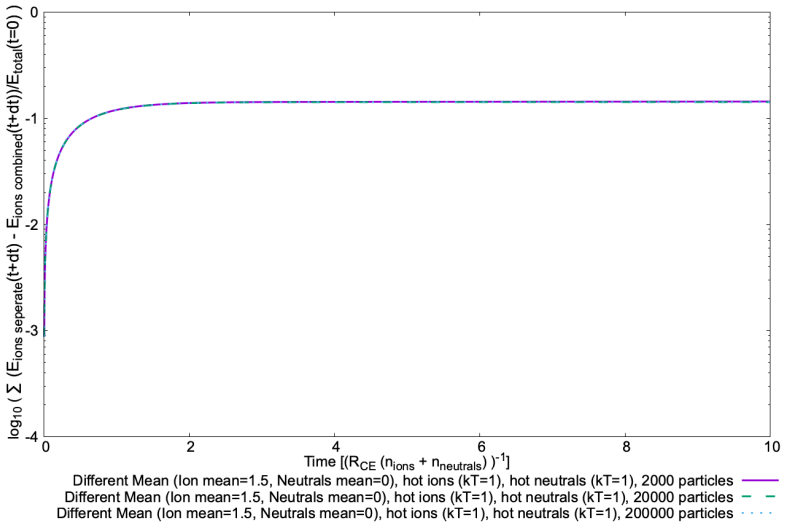


Figure 5: An estimate of the total accumulated energy error due particle merging when merging particles relative to particles remaining separate at the end of each timestep for a scan in macroparticle number when neutral and fluid temperatures are initially identical but mean flow speeds are distinct. Note that the result is the same for all number of particles.

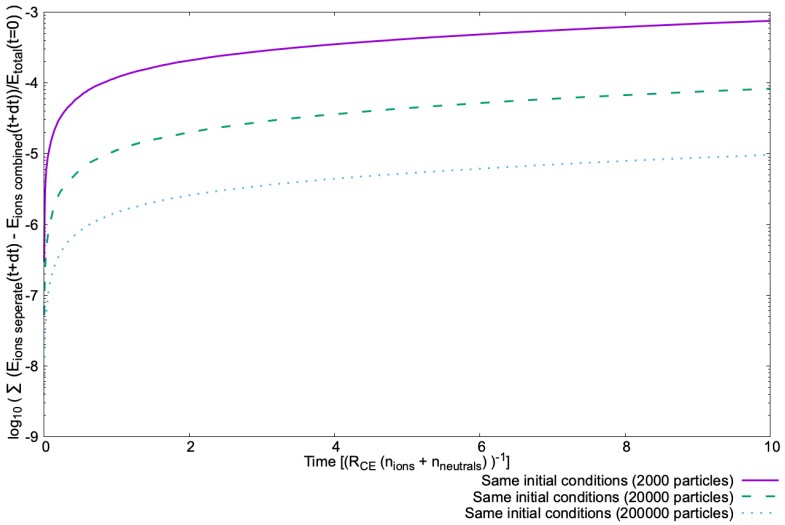


Figure 6: As in Fig. 5 except that the initial mean flow speeds of both neutrals and ions are identical. Note that with identical initial conditions, increasing the number of macroparticles reduces the error in energy.

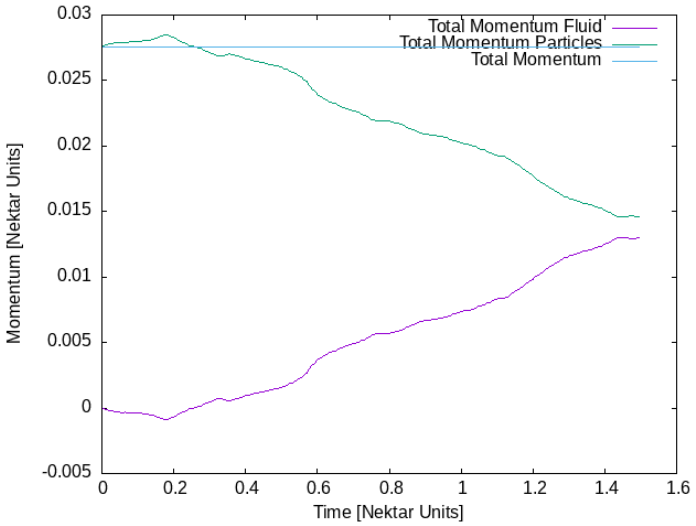


Figure 7: The total momentum of the particles and fluid when charge exchange is implemented into NESO. Note that momentum is conserved and that the trend is towards equipartition of momentum.

3 Summary

This report details the implementation of charge exchange into NESO, which allows for a mechanism of momentum exchange between the neutral and ion populations. It was implemented in such a way as to keep the total number of macroparticles constant, in order make the problem computationally tractable. This current implementation has its limitations as it can can cause a large discrepancy in calculated fluid temperature and the real temperature incertain circumstances. These limitations could be lifted in future work where many macroparticles are merged into two macroparticles, allowing an extra degree of freedom with which to fix the energy conservation.

Acknowledgement

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

References

[1] J. W. Cook, J. T. Parker, W. R. Saunders, and Parry O. Neptune Exploratory SOftware (NESO) repository. `https://github.com/ExCALIBUR-NEPTUNE/NESO`, 2023. Accessed: January 2023.

[2] Atomic-Physics-In-Plasmas-Notes. `https://github.com/mbukaea/Atomic-Physics-In-Plasmas-Notes`.

[3] Atomic Collision Processes in Plasma Physics Experiments CLM-R137. `https://scientific-publications.ukaea.uk/wp-content/uploads/CLM-R137.pdf`.

[4] Automeris WebPlotDigitizer. `https://apps.automeris.io/wpd/`.