UK Atomic Energy Authority

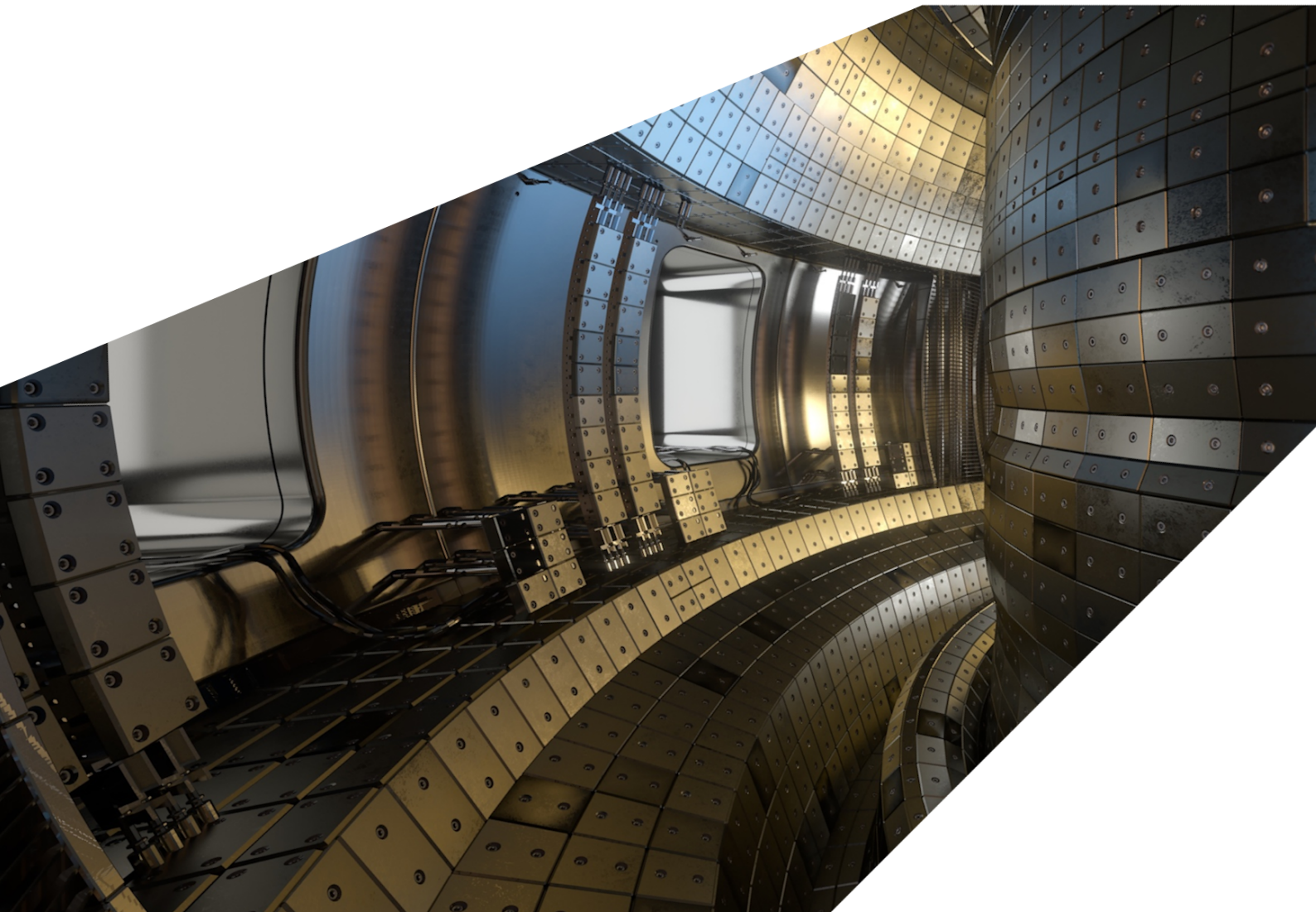
ExCALIBUR

Management of external research. Supports Numerical Analysis Procurement and Software Support Procurement.

M7c.1 Version 1.00

Abstract

The report describes work for ExCALIBUR project NEPTUNE at Milestone M7c.1. A report collating technical material to prepare Calls Contract T/AW087/22 and Contract T/AW088/22.



UKAEA REFERENCE AND APPROVAL SHEET

	Client Reference:	
	UKAEA Reference:	CD/EXCALIBUR-FMS/0068
	Issue:	1.00
	Date:	30 June 2022
Project Name: ExCALIBUR Fusion Modelling System		

	Name and Department	Signature	Date
Prepared By:	Joseph Parker Wayne Arter BD	N/A N/A	30 June 2022 30 June 2022
Reviewed By:	Wayne Arter Project Technical Lead	W. Arter	30 June 2022

1 Introduction

This report provides a literature review designed to inform the production of Call T/AW087/22 “Numerical Analysis procurement” and Call T/AW088/22 “Software Support procurement”. It contains two main sections concerning relevant material corresponding to each call respectively.

In Section 2, two reports produced by STFC under the previous call for Numerical Analysis support are discussed. The first report deals with the state-of-the-art of time advance methods (Section 2.1) and preconditioning methods for the corresponding elliptic problems (Section 2.2). The second report discusses code coupling (Section 2.3) and code coupling libraries (Section 2.4).

In Section 3, a report produced by York and Warwick under the previous Software Support call is discussed. It recommends two approaches to implementing DSLs for multiple species in NEPTUNE.

In Section 4, the work is summarized. The recommendations from these reports laid out in this section were used to define the two procurement calls.

2 Numerical Procurement

In this section we discuss the work which informed the Numerical Analysis support procurement, call T/AW087/22. Two reports were produced by STFC as part of the previous call, namely "A Review of Time Stepping Techniques and Preconditioning for Hyperbolic and Anisotropic Elliptic Problems" [1] and "Techniques and software relevant to the coupling of continuum (fluid) and particle models of plasma for NEPTUNE" [2].

2.1 Time advance techniques

The first report discusses time advance techniques for the solution of hyperbolic partial differential equations (PDEs) and their corresponding elliptic equations in the presence of strong spatial anisotropy. The focus is on methods that are efficient and accurate on Exascale platforms. As such, the methods must be scalable and robust. Moreover, with the need to use high arithmetic intensity methods at the Exascale, the report focusses on methods that can be used in conjunction with Spectral/hp (or finite element) methods.

The use of purely explicit methods is not feasible, due to the disparity of timescales in the physical system. The timestep in explicit methods is limited by the smallest time scale in the problem, so that the solution in the whole domain is limited by finest mesh size or fastest dynamics.

The report therefore focusses on implicit or semi-implicit methods. Such schemes allow larger timesteps, but even when unconditionally unstable, the timestep may be limited by accuracy. Moreover implicit schemes require the solution of a large system of equations, and therefore careful, scalable implementation.

The five schemes discussed are: (1) Diagonally-implicit Runge–Kutta (DIRK) methods, (2) High order implicit Runge–Kutta (IRK) methods, (3) Linear Multistep methods, (4) Implicit-Explicit (IMEX) methods, and (5) Parallel-in-time methods.

Of these approaches, the use of Runge–Kutta methods (1,2) and IMEX methods (4) are recommended by the report. These are discussed in more detail in the following sections.

2.1.1 Implicit Runge–Kutta methods

Implicit Runge–Kutta methods require the solution of a $ns \times ns$ linear block matrix system where n is the problem size and s is the number of Runge–Kutta sub-steps. As well as presenting possible memory problems at large scales, it is found that typically for spatial discretizations of PDEs, the blocks are large and the system is ill-conditioned [3], so that even direct methods may fail.

DIRK methods In DIRK, the Runge–Kutta method is chosen such that the linear block matrix is tridiagonal, meaning that a series of $n \times n$ systems replaces the single $ns \times ns$ system. There are many subclasses of this approach, surveyed by Yan et al. [4].

While the block tridiagonal structure makes the system much easier to solve, there are two main drawbacks with DIRK. Firstly, the errors from spatial and temporal discretization can interact in non-trivial ways, meaning it is easy for a user to do unnecessary work though refining the timestep without seeing any reduction in the global error. This phenomenon is discussed by Pan et al. [5] for a discontinuous Galerkin spatial discretization and Explicit First Stage DIRK (ESDIRK) temporal discretization. This problem may be mitigated using local error estimates [5] but it remains an open question how to control the global error.

Secondly, in practice for stiff nonlinear problems, the order of accuracy in DIRK is observed to be limited by the smaller of the formal integration order of accuracy and the stage order of accuracy plus one. Consequently, symplectic DIRK methods can be at most 4th order, with the added complication that the known DIRK methods of third and fourth order have negative values on the diagonal of the Runge–Kutta matrix, making for linear systems that are harder to solve.

Despite this, low-order DIRK schemes are often sufficient. Many of these schemes are implemented in Nektar++.

IRK methods Fully implicit Runge–Kutta (IRK) methods are not limited in their order of accuracy, be their use requires the solution to a linear problem that is, according to the STFC report, "formidable, and is intractable for realistic sized problems without careful handling of the linear algebra." Efficient preconditioning is vital for solving this problem, and finding relevant preconditioners is an area of active research.

The Runge–Kutta equation may be rewritten in the form of the Sylvester matrix equation, a system commonly solved in model order reduction and control applications. This link has led to various recent efforts to solve the Runge–Kutta equations using techniques from these fields; see [1] for a full list of references.

One technique of particular note is Southworth et al.'s algorithmic preconditioning framework for solving the Runge–Kutta equations in the linear [6] and nonlinear [7] setting. Under quite general assumptions, they show that their preconditioned operator has a condition number bounded by an order-one constant, independent of spatial mesh size and timestep, and with only a weak dependence on p. They give the example of the preconditioned operator for 10th-order Gauss IRK was has condition number less than two, independent of the spatial and temporal discretization.

In [6], Southworth et al. apply their method to various high-order finite-difference and finite element discretizations of linear parabolic and hyperbolic problems, demonstrating fast, scalable solutions of up to 10th-order accuracy. They also demonstrate in several cases, that fourth order accuracy can be achieved with only half the number of preconditioner applications and half the wallclock time of standard DIRK methods.

2.1.2 Implicit–Explicit (IMEX) schemes

In IMEX schemes, the terms in the physics model are integrated differently depending on the timescale they represent. Slow timescales are integrated explicitly (as these permit a long timestep), while fast timescales are integrated implicitly (which is often easier to do in the absence of slow

terms). These schemes have been shown to be effective for multiscale physics problems [8], including plasma physics problems [9]. However, the split between different timescales is not always obvious, with an incorrect split heavily impacting performance. IMEX schemes merit further attention, but only when the exact set of equations to be solved have been determined.

2.2 Preconditioning elliptic problems

On Exascale machines, the cost of moving data far exceeds the cost of operating on it, and therefore numerical methods with high arithmetic intensity methods are required for good performance. Project NEPTUNE is particularly interested in Spectral/hp methods with high-order p. It is also necessary to make use of matrix-free methods, both for memory and computational considerations.

The condition number of relevant discretization matrices increases quadratically with polynomial degree p. Due to bandwidth requirements, direct methods are not viable, and so iterative methods – Krylov solvers with appropriate preconditioners – must be used.

Preconditioning has been well-studied at low to medium order in p, with good implementations are readily available. Standard methods however have been observed to fail when applied to high order in p. One approach that has been successful is to observe the spectral equivalence of low-order and high-order operators, known as FEM-SEM equivalence. This allows low-order discretizations to be used as fast, approximate preconditioners for the high-order system. Low order solvers are now sufficiently robust to solve the resulting low order systems [10].

Given a low order anisotropic elliptic system, currently the two most promising approaches for a performant parallel preconditioner are multigrid methods and domain decomposition based approaches.

There are two classes of multigrid method, algebraic multigrid (AMG), which only uses properties of the assembled problem matrix, and geometric multigrid (GMG), which uses information about the underlying mesh connectivity in the problem geometry. GMG methods can be over an order-of-magnitude faster than AMG, since they can be matrix-free, and different operators can be used on each multigrid level. However, they are harder to set up and need much specialization to each problem. In contrast, AMG solvers have many excellent implementation (for example in PETSc and Trilinos), and are often robust and performant when used as black box solvers. To get best performance however, it is still necessary to tune the (sometimes obscure) parameters for a specific problem.

In domain decomposition methods, the problem matrix is partitioned into sub-domains which are largely independent, and on which the more tractable sub-problems can be solved and iteratively composed to find the full solution. See Dolean et al. [11] for an overview. The key to an efficient method is the careful selection of the coarse space (*i.e.* the domain subdivision). One of the most promising approaches is the GenEO method [12], which requires minimal user input, but nonetheless performs well regardless of problem size, mesh, discretization type and order and parameters for elliptic PDEs [13, 14]. Moreover, the preconditioner can be constructed so that its condition number does not exceed a user-specified number!

The STFC authors are actively researching GenEO [15, 16], and have shown the method to be

robust and scalable, even for challenging non-symmetric problems. Implementations of their Algebraic GenEO methods are available in PETSc (as the preconditioner PCHPDDM), and only require the Runge–Kutta coefficient matrix as input.

2.3 Code coupling

The second report "Techniques and software relevant to the coupling of continuum (fluid) and particle models of plasma for NEPTUNE" [2] discusses three different methodologies to code coupling that could be pursued by NEPTUNE: Spatial Hybridization (SpH), Micro-Macro Hybrid (mMH) and Kinetic-Diffusion Monte Carlo Method (KDMC). Of these, the report recommends Spatial Hybridization, and then goes on to analyse existing software for coupling codes with this approach.

The discussion is specific to tokamak physics, and largely focussed on how best to incorporate kinetic effects from the burning plasma core region into the fluid models of the divertor region that are crucial for tokamak design. The parameter of interest for the relevance of kinetic neutral physics is the Knudsen number, $\mathrm{Kn} = \lambda/L$, where λ is the neutral particles' mean free-path and L is the gradient length. Where $\mathrm{Kn} \ll 1$, neutrals behaviour is strongly coupled to the background plasma and fluid models are appropriate, while for $\mathrm{Kn} \gg 1$, fluid closures for neutrals are inaccurate and particle approaches must be used.

Spatial Hybridization (SpH) In SpH, the spatial domain is decomposed into distinct regions, with separate models in each, and boundary conditions along the interface. In NEPTUNE, this would entail using a fluid model in the small, low Knudsen number regime around the divertor and a particle model in the rest of the domain.

Using this approach leads to questions that must be resolved: How does one choose the interface between models – should it be a boundary or an overlap region? How can one define boundary conditions based on evolving fluid quantities without losing computational performance? How can one ensure load balancing between the particle and fluid calculations?

Micro-Macro Hybrid (mMH) In MmH [17], the neutral distribution function is decomposed into a fluid and an kinetic part. One then evolves a fluid equation for the neutral fluid part, with the neutral kinetic part providing kinetic corrections to the source and transport terms in the fluid system.

This has been shown to produce results that are equivalent to solving the full kinetic system. However, it is now necessary to evolve the fluid system throughout the whole domain, while also everywhere computing some model for the kinetic terms. Without some additional insight, this approach can be at least as expensive as solving for the full kinetic system.

The MmH approach is similar to the moment based approach investigated under call T/NA085/20 (which decomposes the distribution function for all species, not just neutrals). As with the approach in NEPTUNE, significant theoretical development is required before MmH could be used.

Kinetic-Diffusion Monte Carlo (KDMC) KDMC [18] is a recently-developed scheme that is asymptotic-preserving and designed to allow both high- and low-collisional regions within a model without incurring the costs of a fully kinetic simulation across the whole domain. In KDMC, particles are treated in the usual kinetic manner for low-collisionality, but when local collisionality is high, particles behaviour in accordance with the fluid limit (following a diffusive random walk).

This approach removes the need for coupling between a fluid and kinetic models. However KDMC is novel and has thus far only been used for simplified Boltzmann-BGK problems for a single species.

2.4 Code coupling libraries

The report discusses four libraries: Multiscale Universal Interface (MUI), OpenPALM, preCICE, and Point Location Exchange library (PLE), and points to a more extensive review undertaken by STFC in a separate project [19].

All the libraries discussed function in a broadly similar manner, providing a programming interface to couple codes via MPI. The couplers are agnostic to the physical systems solved (though some have variants targetted at specific physics use cases). The only generic requirement is that all the data required for code coupling can be associated with a single point in space.

As such, the recommendation of the report follows from other considerations, like the scalability, quality of documentation, and the staff of the development team. The report recommends using MUI, noting that is developed at STFC and supported by a (non-NEPTUNE) ExCALIBUR project. It also has the advantage of being a header-only library, and designed to use MPI independently of existing communicators in the coupled code. The report also notes that at the time of writing (March 2022) all frameworks were in a stage of developing user tutorials, and indeed only MUI and preCICE could be set up for use on Archer2. This highlights the importance of maintaining a working relationship with the developers of coupling codes.

3 Domain Specific Language Procurement

In this section we discuss the work which informed the Software Support procurement, call T/AW088/22, in particular the work which led to defining the call section on Domain Specific Languages (DSLs). The previous Software Support project produced a report on Domain Specific Languages “DSL and code design pattern for NEPTUNE” [20], which describes the use of DSLs for multi-species simulations in the BOUT++ [21] and Nektar++ [22].

The report discussed the software developments done in BOUT++ to enable multi-species simulations within the existing DSL. Two approaches are described, both of which have been implemented in different BOUT++ physics models, SD1D [23] and Hermes-3 [24]. The report goes on to propose that a hierarchy of DSLs be provided for NEPTUNE developments.

3.1 Approaches to multi-species DSLs

When there are a small number of species (say only electrons and ions), it is simple to write code for each species individually. However this approach fails when realistic atomic physics is introduced, and perhaps $\mathcal{O}(50)$ species must be modelled. Without imposing some structure on how the code represents a general species, the source becomes increasingly difficult to read, test and maintain.

Two approaches are proposed in the report: (1) "subclassing", where each species inherits from a base class and is modified to describe its specific behaviour; and (2) "state-command pattern", where the species states are stored in a dictionary, and each is evolved according to a collection of model components.

Subclassing The subclassing approach was implemented in SD1D [23], where the model is constructed from Species classes which represent each species, and Reactions classes that represent interactions between species. Another object representing the whole system contains a list of Species and Reaction objects, and controls the interaction between them.

This approach lends itself to native implementation in C++, and is indeed the approach taken in the NEPTUNE Exploratory SOftware (NESO) prototype [25] where a Plasma object contains a list of all Species objects. Reactions are not yet implemented in NESO, but would be straightforward to implement within this structure.

State-command pattern The state-command pattern approach was implemented in Hermes-3 [24]. In this approach, each species is defined as a member of a "state" dictionary, following a simple schema. To evolve the state, a set of composable model components are defined, *e.g.* a particle pusher that would act on a single species, or collisions that would act on multiple species. In each timestep, the state of the system is passed through the model components in two passes. In the first pass, model components modify the state, while in the second pass, the state cannot be modified, but is used to update the model components. An example of a first-pass operation is particle pushing, which modifies the properties of the species being pushed. An example of a second-pass operation would be using the updated state to compute a new species thermal velocity. This operation does not alter the state, but changes the behaviour of the model component (here the particle pusher) on the first pass of the next timestep.

Both of the above approaches can be integrated with a task graph approach to managing dependencies between model components. This is of particular interest, as the SYCL programming model allows for dependencies between work kernels to be specified, allowing for more efficient parallelization and load balancing.

The report recommends implementing a hierarchy of DSLs, to make development at each level of code abstraction easier. At a low level it recommends using DSLs like OP2 or OPS to allow developers to define iterations over (for example) mesh edges or nodes, rather than needing to work directly with array indices. At a higher level, it recommends a natural language DSL, as is currently provided in BOUT++, Nektar++ and UFL. Given the similarity of these DSLs, it further suggests that common features could be extracted to create a single DSL that could be used in

NEPTUNE for both BOUT++ and Nektar++. However, it notes that such work would be a significant undertaking, and moreover, adding multi-species support to Nektar++ will require further code developments.

4 Summary

In this report, we have summarized the findings of the three reports “A Review of Time Stepping Techniques and Preconditioning for Hyperbolic and Anisotropic Elliptic Problems” [1], “Techniques and software relevant to the coupling of continuum (fluid) and particle models of plasma for NEPTUNE” [2], and “DSL and code design pattern for NEPTUNE” [20].

Time advance techniques The first report discusses the state-of-the-art for time advance techniques and preconditioning of the linear systems that arise. It recommends the use of Diagonally Implicit Runge–Kutta (DIRK) or Implicit Runge–Kutta (IRK). DIRK methods are constrained to have a low order of accuracy (at most 4th order). Such a low order methods may be sufficient for NEPTUNE.

Alternatively, IRK methods allow arbitrarily high orders of accuracy, but the linear system that is required to solve is extremely challenging – to the extent that it is only recent advances in preconditioning that makes the approach feasible.

Preconditioning Preconditioning is crucial for efficient solution of Runge–Kutta matrix. The most promising approach to preconditioning high order elliptic PDEs is to exploit the FEM-SEM equivalence, and precondition with the corresponding low-order operators. While this also requires the solution of a challenging linear system, library solvers are now sufficiently robust that this approach can be reliably used.

The most performant parallel preconditioners are multigrid and domain decomposition-based approaches. Both algebraic and geometric multigrid have various suitable library implementations, but both require parameter tuning to obtain the best performance. Geometric multigrid can be orders-of-magnitude faster than algebraic multigrid, but it is also commensurately harder to performance tune.

The recommended domain decomposition-based preconditioner is GenEO. It offers a favourable compromise between performance and ease of calibration, and its development is an area of active research by STFC authors.

Code coupling The second report recommends code coupling via a Spatial Hybridization approach. That is, the spatial domain should solve different models in separate regions of space. Questions remain over the details of this coupling – for example, should the models interface at a boundary, or in an overlap region?

Various suitable libraries exist for coupling with Spatial Hybridization. Typically these provide an MPI interface for passing data between codes, provided that each coupling data point can be associated with a single point in physical space.

While sufficient, the available coupling libraries are immature, and the report stresses the importance of collaborating with library developers. To that end, it recommends using the Multiscale Universal Interface (MUI) library that is developed by STFC and within ExCALIBUR.

Domain Specific Languages The third report recommends that a hierarchy of DSLs should be used, to facilitate development at each level of code abstraction. The lowest level should be a performance portable paradigm like SYCL, Kokkos or Raja, to simplify implementation of the performance-critical loops. A medium level DSL should be implemented in OP2 or OPS to allow easier interaction with higher-level objects, like meshes. Finally a high-level, natural language DSL should be offered after the style of BOUT++, Nektar++ and UFL to increase the accessibility of the code to non-HPC specialists.

We consider that NEPTUNE is now placed to implement domain specific languages within NESO. Thus call T/AW088/22 specifies that "two DSLs for NEPTUNE, one for the end-user and one to assist developers should be produced in conjunction with UKAEA".

Acknowledgement

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

References

[1] H. Al Dass, T. Rees, and S. Thorne. A Review of Time Stepping Techniques and Preconditioning for Hyperbolic and Anisotropic Elliptic Problems. Technical Report 2060049-TN-01, UKAEA Project Neptune, 2022. `https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/2060049/TN-01.pdf`.

[2] H. Al Dass, T. Rees, A. Sunderland, E. Sahin, and S. Thorne. Techniques and software relevant to the coupling of continuum (fluid) and particle models of plasma for NEPTUNE. Technical Report 2060049-TN-02, UKAEA Project Neptune, 2022. `https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/2060049/TN-02.pdf`.

[3] U. M. Ascher and L. R. Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations*, volume 61. Siam, 1998.

[4] Z.-G. Yan, Y. Pan, G. Castiglioni, K. Hillewaert, J. Peiró, D. Moxey, and S.J. Sherwin. Nektar++: Design and implementation of an implicit, spectral/hp element, compressible flow solver using a Jacobian-free Newton Krylov approach. *Computers & Mathematics with Applications*, 81:351–372, 2021.

[5] Y. Pan, Z.-G. Yan, J. Peiró, and S. J. Sherwin. Development of a balanced adaptive time-stepping strategy based on an implicit JFNK-DG compressible flow solver. *Communications on Applied Mathematics and Computation*, 4(2):728–757, 2022.

[6] B. S. Southworth, O. Krzysik, W. Pazner, and H. De Sterck. Fast Solution of Fully Implicit Runge–Kutta and Discontinuous Galerkin in Time for Numerical PDEs, Part I: the Linear Setting. *SIAM Journal on Scientific Computing*, 44(1):A416–A443, 2022.

[7] B. S. Southworth, O. Krzysik, and W. Pazner. Fast Solution of Fully Implicit Runge–Kutta and Discontinuous Galerkin in Time for Numerical PDEs, Part II: Nonlinearities and DAEs. *SIAM Journal on Scientific Computing*, 44(2):A636–A663, 2022.

[8] David E Keyes, Lois C McInnes, Carol Woodward, William Gropp, Eric Myra, Michael Pernice, John Bell, Jed Brown, Alain Clo, Jeffrey Connors, et al. Multiphysics simulations: Challenges and opportunities. *The International Journal of High Performance Computing Applications*, 27(1):4–83, 2013.

[9] Sean T Miller, Eric C Cyr, John N Shadid, Richard Michael Jack Kramer, Edward Geoffrey Phillips, Sidafa Conde, and Roger P Pawlowski. IMEX and exact sequence discretization of the multi-fluid plasma model. *Journal of Computational Physics*, 397:108806, 2019.

[10] Will Pazner, Tzanio Kolev, and Clark Dohrmann. Low-order preconditioning for the high-order finite element de rham complex. *arXiv preprint arXiv:2203.02465*, 2022.

[11] V. Dolean, P. Jolivet, and F. Nataf. *An Introduction to Domain Decomposition Methods: Algorithms, Theory, and Parallel Implementation*. SIAM, 2015.

[12] N. Spillane, V. Dolean, P. Hauret, F. Nataf, C. Pechstein, and R. Scheichl. Abstract robust coarse spaces for systems of PDEs via generalized eigenproblems in the overlaps. *Numerische Mathematik*, 126(4):741–770, 2014.

[13] H. Al Daas, L. Grigori, P. Jolivet, and P.-H. Tournier. A multilevel schwarz preconditioner based on a hierarchy of robust coarse spaces. *SIAM Journal on Scientific Computing*, 43(3):A1907–A1928, 2021.

[14] P. Bastian, R. Scheichl, L. Seelinger, and A. Strehlow. Multilevel spectral domain decomposition. *SIAM Journal on Scientific Computing*, (0):S1–S26, 2022.

[15] H. Al Daas, P. Jolivet, and T. Rees. Efficient Algebraic Two-Level Schwarz Preconditioner For Sparse Matrices. *arXiv preprint arXiv:2201.02250*, 2022.

[16] H. Al Daas, P. Jolivet, and Scott. J. A. A robust algebraic domain decomposition preconditioner for sparse normal equations. *In press. preprint:arxiv.org/abs/2107.09006*, 2022.

[17] Niels Horsten, Giovanni Samaey, and Martine Baelmans. A hybrid fluid-kinetic model for hydrogenic atoms in the plasma edge of tokamaks based on a micro-macro decomposition of the kinetic equation. *Journal of Computational Physics*, 409:109308, 2020.

[18] B. Mortier, M. Baelmans, and G. Samaey. A kinetic-diffusion asymptotic-preserving monte carlo algorithm for the Boltzmann-BGK model in the diffusive scaling. *SIAM Journal on Scientific Computing*, 44(2):A720–A744, 2022.

[19] P. Rubin. Comparison of code coupling libraries for high performance multi-physics simulation. Technical Report DL-TR-2022-001, STFC, 2022. `https://epubs.stfc.ac.uk/work/52144823`.

[20] B. Dudson, P. Hill, E. Higgins, D. Dickinson, S. Wright, and D. Moxey. DSL and code design pattern for NEPTUNE. Technical Report 2047356-TN-14, UKAEA Project Neptune, 2022. `https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/2047356/TN-14.pdf`.

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

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

[23] One-dimensional plasma-neutral simulation for SOL and divertor studies. https://github.com/boutproject/SD1D, 2021. Accessed: August 2021.

[24] Hermes plasma edge simulation model: Hermes-3, a hot ion multifluid drift-reduced model. https://github.com/bendudson/hermes-3, 2021. Accessed: June 2021.

[25] J. W. Cook, J. T. Parker, and W. R. Saunders. Neptune Exploratory SOftware (NESO) repository. https://github.com/ExCALIBUR-NEPTUNE/NESO, 2022. Accessed: June 2022.

A Extracts from External Grants for NEPTUNE Y4-6

A.1 Numerical Analysis Procurement

The objective is to provide mathematical, particularly numerical analytic, support to project NEPTUNE. This is expected to involve development of mathematical libraries suitable for use at Exascale and specially adapted to project NEPTUNE needs, namely for use in conjunction with high order, curvilinear finite elements and particles. The focus is expected to be on methods for treating system stiffness in respect of both spatial and temporal variation. These libraries might build on work conducted under Grant T/AW086/21 regarding schemes for special treatment of stiff time-dependence, efficient use of particles in conjunction with finite elements, and developments of suitable matrix preconditioners, now extending to coupled elliptic and parabolic systems.

The outputs should be the libraries, conforming to emerging NEPTUNE high software standards where possible and proxyapps designed to demonstrate their suitability for use at the Exascale. Advice should be provided as to best use of the libraries to ensure timely project completion.

A.2 Software Support Procurement

The objectives are (1) to continue seamlessly the existing external contract T/AW087/21 so that the co-design of the NEPTUNE code base for Exascale hardware and software follows through to the end of the project, and (2) to provide consultancy on edge physics computation supported by verification and benchmark calculations with existing plasma modelling software such as BOUT++. Previous contracts relating to (1) had four key deliverables:

1. report summarising available and upcoming technologies likely to be Exascale-relevant, with particular focus on DSLs and performance-portable programming models;

2. repository of benchmarks and data sets for assessing the various technologies identified above;

3. report on the performance of the benchmarks, and recommendations on best approaches for Exascale software development; and

4. final report on best approaches to scientific software development for portable performance on post-Exascale HPC systems.

The information contained in these deliverables must be kept up-to-date and must be expanded upon where appropriate. This may be achieved either by maintaining the benchmark repository and treating the reports as "living documents", or by providing an equivalent repository and set of reports. Concerning (2) the objective is to help produce proxyapps as described under numbers 6-8 in FM-WP2, providing verification and benchmarking only for continuum models (ie. no particles).

Two DSLs for NEPTUNE, one for the end-user and one to assist developers should be produced in conjunction with UKAEA. In addition, assistance should be provided to other partners in NEPTUNE, as they seek to produce software that conforms to the emerging NEPTUNE standards, so that eg. optimal use is made of the DSLs.