SOFTWARE FOR FUSION REACTOR DESIGN: EXCALIBUR PROJECT NEPTUNE
Towards exascale plasma edge simulations

E.J. THRELFALL
UK Atomic Energy Authority
Culham Science Centre, Abingdon OX14 3DB, UK
Email: ed.threlfall@ukaea.uk

[1]R.J. AKERS, [1]W. ARTER, [2]M. BARNES, [1]M. BARTON, [3]C. CANTWELL, [4]P. CHALLENOR, [1]J.W.S. COOK, [5]P.V. COVENEY, [4]T. DODWELL, [6]B. DUDSON, [7]P.E. FARRELL, [8]T. GOFFREY, [9]M. GREEN, [10]S. GUILLAS, [2]M. HARDMAN, [11]P. HILL, [4]L. KIMPTON, [1]C. MACMACKIN, [8]B. MCMILLAN, [9]D. MOXEY, [12]G. MUDALIGE, [1]J. OMOTANI, [1]J.T. PARKER, [2]F. PARRA DIAZ, [1]O. PARRY, [13]T. REES, [6]C. RIDGERS, [1]W. SAUNDERS, [3]S.J. SHERWIN, [13]S. THORNE, [13]J. WILLIAMS, [14]S. WRIGHT, and [10]Y. YANG.

[1]UKAEA, D3 Culham Science Centre, Abingdon, Oxon., OX14 3DB, UK
[2]Rudolf Peierls Centre for Theoretical Physics, University of Oxford, Oxford, OX1 3PU, UK
[3]Department of Aeronautics, Imperial College London, Exhibition Road, London, SW7 2AZ, UK
[4]College of Engineering, Mathematics, and Physical Sciences, University of Exeter, Exeter, EX44QF, UK
[5]Department of Chemistry, University College London, London, WC1E 6BT, UK
[6]York Plasma Institute, Department of Physics, University of York, Heslington, York, YO10 5DD, UK
[7]Mathematical Institute, University of Oxford, Oxford, OX2 6GG, UK
[8]Centre for Fusion, Space and Astrophysics, Department of Physics, University of Warwick, Coventry, CV4 7AL, UK
[9]Department of Engineering, King's College London, WC2R 2LS, UK
[10]Department of Statistical Science, University College London, London, WC1E 6BT, UK
[11]School of Physics, Engineering and Technology, University of York, Heslington, York, YO10 5DD, UK
[12]Department of Computer Science, University of Warwick, Coventry, CV4 7EZ, UK
[13]Hartree Centre STFC, Warrington, WA4 4AD, UK
[14]Department of Computer Science, University of York, Deramore Lane, Heslington, York, YO10 5GH, UK

Abstract

This article provides a broad overview of Project NEPTUNE, which aims to create code for simulating edge plasma physics at scales approaching the exascale, and is run by the UKAEA as part of the UK's ExCALIBUR framework. It is a representative survey of scope and associated outputs, rather than a detailed exposition; the structure is such as to describe the core components of NEPTUNE (efforts toward simulation of plasma fluid and kinetic effects, and their synthesis into exascale-ready combined codes) and then give an indication of the wide range of other work performed under the project.

1. INTRODUCTION

Project NEPTUNE (NEutrals and Plasma TUrbulence Numerics for the Exascale) aims to provide simulation capability for the physics of the plasma edge region via code designed to operate effectively on forthcoming exascale-class HPC hardware. The project, which represents the fusion use case of the UK Government's ExCALIBUR (Exascale Computing Algorithms and Infrastructures Benefitting UK Research) programme ([1]), has a five-year duration, commencing on 1 April 2019, plus a limited amount of funding for a sixth year. NEPTUNE is being managed by a core team within UKAEA, with approximately two-thirds of the work defrayed to institutions within the UKRI community on the basis of competitive tender.

A basic surmise of the project is that it aims to tackle a complicated set of partial differential equations (PDEs) describing the physics of the plasma edge region of a tokamak such as to UK's forthcoming STEP project ([2]); the scope is restricted to the plasma edge region and thus idealised formulations may not be valid, e.g. linearised fluctuations or interactions minorities. The relevant physics has a natural division into plasma fluid phenomena, described by time-dependent PDEs in up to three dimensions of space, and kinetic phenomena which involve up to six dimensions of phase space; the basic arbiter of which regime applies is the degree of collisionality of the plasma (high collisionality validates a fluid description, which does not otherwise apply). The fluid simulation aspect encompasses plasma turbulence and is targeted at reproducing, at least to a

degree, the capabilities of existing edge plasma simulation frameworks, e.g. Hermes-3 ([3]), but differing in the use of the spectral / hp method as opposed to the finite-difference numerical method (as will be explained, the former is expected to scale better when approaching the exascale, as well as offering a natural way to incorporate realistic CAD geometries). Due to the higher dimensionality of the phase space, continuum simulations of kinetic phenomena are a major challenge and so NEPTUNE adopts a particle-based description of kinetics, with this capability being developed in-house at UKAEA.

In addition to the above physical considerations, it is necessary, if exascale credibility is sought, to tailor code development to the landscape of existing and anticipated future high-performance computing. At the time of writing, cutting-edge HPC largely comprises systems with a large GPU component, with the GPUs expected to be provided by one of Nvidia, AMD, or Intel. This diversity necessitates the use of performance-portable programming paradigms and SYCL ([4]) has been chosen as the preference for NEPTUNE.

The overall scope of the project has included a significant body of other work, which is briefly described.

Note that the codes described in the sequels can be found in the GitHub organization [5] and a large body of supporting material is in the documents repository [6].

2. FLUID SOLVERS

There are several well-established and capable frameworks for simulating plasma fluid equations. Software developed under NEPTUNE is intended to surpass these existing codes by (a) offering an integrated framework for kinetic particles, (b) avoiding the use of finite-difference methods, and (c) being implemented in such a way as to offer true performance portability. The latter two points are intended to enable effective operation at the exascale. The reason to avoid finite-difference methods is that the convergence order of these methods is fixed (unless the algorithm is rewritten to incorporate a larger finite-difference stencil); this leads to codes with a relatively small number of arithmetic operations per unit of stored data, meaning that the codes are memory-bound on modern hardware, particularly so in view of the need to transfer large amounts of data onto and off of a GPU. A better fit to current HPC hardware is the use of high-order methods, which generally employ smaller numbers of degrees of freedom but have a higher operation count per unit of stored data (the so-called 'arithmetic intensity'). One such higher-order strategy is the spectral / hp method, which is a particularly efficient implemention of high-order finite elements.

One major strand of the project is therefore the implementation of plasma fluid equations using a spectral / hp method. The 'price of admission' to higher-order methods is the need to write the necessary machinery including the finite-element basis functions and the geometric transformations needed to handle various shapes of element in 2D and 3D; this work can be avoided by adopting code from an existing implementation, and the one used in NEPTUNE is Nektar++. This code, which is open-source under the MIT licence, has existed since 2006, and which has a well-established user community, is described at [7] as

> *'Nektar++ is a tensor product based finite element package designed to allow one to construct efficient classical low polynomial order h-type solvers (where h is the size of the finite element) as well as higher p-order piecewise polynomial order solvers.'*

Nektar++ caters for a range of PDEs, which can be hyperbolic, parabolic, or elliptic; the code includes pre-written solvers for acoustic, advection-diffusion-reaction, cardiac electrophysiology, compressible flow, incompressible Navier-Stokes, linear elasticity, pulse wave, and shallow water problems. The underlying method can be continuous or nodal discontinuous Galerkin. There are the options of implicit, explicit, or IMEX time-stepping (though availability depends on solver choice). The solvers can be coupled to others from the Nektar++ library or to entirely separate applications to provide multiphysics capability; an MPI coupling implementation consists of a smooth interpolating field layer with data transfer handled using the CWIPI library ([8]).

The framework is multi-layered in that there is a structured hierarchy of C++ components and the library structure is conducive to the addition of new physics solvers, which make use of lower-level objects and kernels via C++ features e.g. inheritance and overloading. One aim of the project has been the implementation of plasma physics solvers in the Nektar++ framework. These solvers begin at the level of a simple one-dimensional model of the scrape-off-layer ([9]), which was implemented by deriving a new solver class from the compressible flow solver of Nektar++. There are currently 2D solvers for driftwave turbulence ([10] and Fig.1), edge plasma filaments, and turbulence ([11]), with 3D solvers currently in development; the current phase of the project can be described as an attempt to implement the physics capability of codes such as Hermes-3 in a spectral / hp framework, using where possible, and extending as necessary, the existing Nektar++ API.

Nektar++ is at present a CPU-dominant code with parallelism via MPI and the current iteration has been updated to the C++17 standard under a NEPTUNE grant. It is known to scale effectively to at least hundreds of thousands of cores. There is a current plan to port parts of Nektar++ incrementally from being a CPU code to a code incorporating true performance portability on GPUs: a current NEPTUNE grant supports conversion to a device-aware framework with interchangeable operator kernels. Further, there is work aiming to convert Nektar++ to use SYCL (this programme enjoys a degree of synergy with the aforementioned work in terms of the refactored data structures).

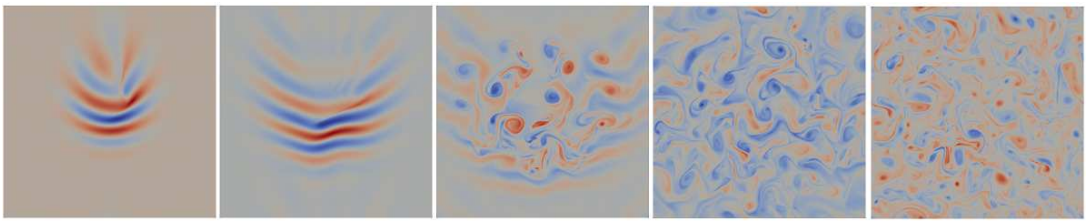

FIG. 1. A series of frames from the time-evolution of a turbulent drift-wave from a Gaussian initial profile, in the Nektar++ solver Nektar-Driftwave (vorticity shown).

3. KINETIC SOLVERS

The above fluid description of a plasma is only valid if there is sufficient collisionality for the assumption of local thermodynamic equilibrium to hold. In regions where this is not the case, such as in the regions of the plasma edge where collision rates are lower, a kinetic theory description of matter must be used; for the full three-dimensional case the resulting equations must be solved on a five- or six-dimensional phase-space. This high-dimensionality largely precludes reliance only on continuum approaches and so the decision was made to use particle methods to handle kinetic aspects of neutral species in NEPTUNE (the trade-off this brings is the problem of sampling noise due to the finite number of particles). The further decision to implement an entirely new particles framework has the benefit that the development can take place in the preferred choice of language and parallel abstraction framework.

Core components focus on efficient particle data communication with the ability to handle both highly-directional flows and fast global omnidirectional flows, both on an unstructured high-order mesh; these two main scenarios are representative respectively of charged particle motion in the direction of magnetic field lines, and the motion of neutral particles. The particle data communication uses MPI and tracks particles between MPI ranks by means of a coarse octree space partitioning and halo cells augmenting the mesh at the spatial boundaries demarcating MPI rank ownership. In the charged case the particles act as sources for continuum electric and magnetic fields according to the particle-in-cell paradigm. Note that pairwise interaction of particles is considered, at least for the present work, to be of lesser importance than collective effects and interactions with fluid species.

The implemented code has the name NESO-Particles (the main in-house NEPTUNE developments have been named NESO - NEPTUNE Exploratory SOftware - which is also a satellite of the planet Neptune) and can be found at the repository [12]. It has thus far been used to provide implementations of test problems involving charged particles and also to model kinetic neutral populations which are coupled to the plasma fluid framework described in the prequel. Wider use (beyond the scope of NEPTUNE) is anticipated. Initial testing and scaling results can be found in the report [13]. Early works were informed by interaction with the developers of the EPOCH particles framework ([14]).

4. PERFORMANCE PORTABILITY

A significant challenge for any software project aspiring to the cutting edge of HPC is the need to achieve high performance on current and forthcoming supercomputers, which, for the forseeable future, are expected to comprise heterogeneous architectures with CPU hosts and specialized accelerators, and which demand massive parallelization. One possible approach is to maintain multiple versions of a software, or of components thereof (e.g. compute kernels), for execution on specific hardware types; this approach, however, has the associated disadvantages of the need to write, debug, and maintain multiple incarnations of the same functional code and the need for a deep level of understanding of multiple specific hardware APIs. More promising is a separation-of-concerns approach in which the specification of a particular algorithm is separated from the hardware-specific implementation by means of a cross-platform abstraction layer such as SYCL or Kokkos.

Modern C++ was selected early on as the language of choice for NEPTUNE, due both to the range of available features and the suitablility for HPC, due to the fact that Nektar++ is C++, and also the skillsets present in the NEPTUNE team. SYCL has been selected as a natural choice of performance portability framework for Project NEPTUNE due to its close integration with modern C++ (the above-mentioned fluid and kinetic codes are both implemented in C++).

The considerations of this section were informed in large part by work done by one of the grantees, and are the subject of a forthcoming co-authored paper currently in submission to Computer Physics Communications ([15]).

5. SYNTHESIS

The project's deliverables schedule has specified the development of coupled solvers of increasing complexity, which embody the 'proxyapps' development philosophy of NEPTUNE: there is an ecosystem of codes, each treating one or more aspects of

the problem (and giving a path to a credible, integration-tested codebase). One initial solver ([16]) tackles the classic two-stream instability from plasma physics using NESO-Particles to model electrons in a uniform neutralizing background with the electrostatic field solved using the Helmholtz solver on Nektar++. A further two-dimensional model, consisting of a plasma fluid coupled to a population of neutral particles, is described in [17], and see Fig.2; this represents a crude model of the SOL with neutral particles ionizing to create electron density. Current developments aim to extend this work to 3D, with neutral particles interacting with 3D plasma fluid models of increasing complexity (starting with drift-wave models and continuing to more realistic scenarios e.g. a simulation of the Large Plasma Device ([18]), mirroring an implementation of the same in Hermes-3). It is anticipated that these 3D codes will shortly begin testing at scale in order to demonstrate the benefits of the NEPTUNE approach.

These coupled solvers currently reside in the NESO framework, which has the convenience of being able to be built with Spack ([19]). The implementations currently use a tight-coupling approach but the capabilities of Nektar++ will in future enable more flexible coupling paradigms.

Reactions (the scope thus far being the interactions between kinetic neutrals and the plasma fluid) are implemented using physics data taken from the Atomic Data and Analysis Structure (ADAS) database, which is interrogated by means of an existing Python interface. This approach is supplemented with analytic formulae. This enables the incorporation of a hierarchy of physical effects, with perhaps the main one being the ability for neutrals to act as a source of plasma by ionization (the latter, and charge exchange, are currently implemented). More advanced extensions (recombination, excitation, radiation, impurity modelling) are expected to be added. The reactions code can be found in the NESO-Reactions repository ([20]).

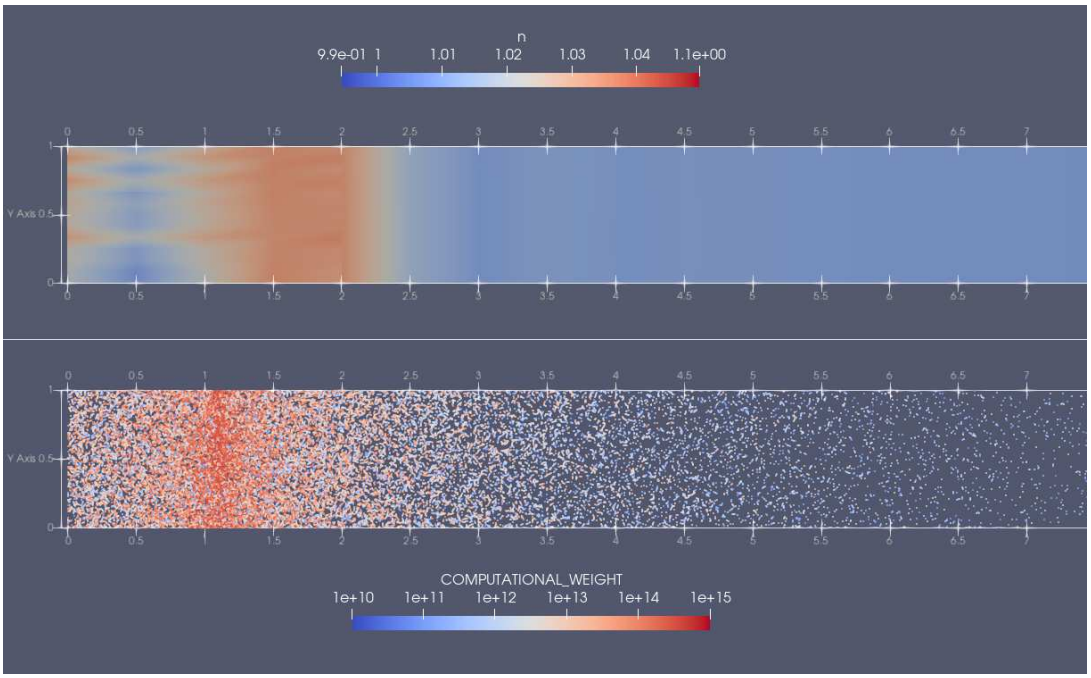

FIG. 2. Fluid / particles proxyapp test case: electron density n (top) relative to an initial reference value, and neutral particle positions (bottom), coupled by ionization. The plots show the left-most part of an elongate domain (the horizontal axis is displacement parallel to a magnetic field line and the vertical axis is transverse displacement), with the left boundary representing the divertor. As neutral particles ionize into the plasma they create electron density which is advected in the direction of the divertor. Mass conservation means that the computational weight of the ionizing particle, represented in the plot by the colour coding, is reduced (conceptually, each computational particle represents some number of physical particles and that number is proportional to the computational weight).

6. OTHER GRANTEE WORK

The work described hitherto is supported and supplemented by a portfolio of work delivered largely by external tender. Much of this latter work operates in line with the proxyapp development philosophy, meaning that the code outputs treat part of the problem, with the results to be incorporated into the NESO code base in future. Additionally, the grantee involvement has led to the establishment of a diverse group of NEPTUNE contributors within the UKRI community (and which includes several senior developers of Nektar++).

6.1. Meshing

The need to provide accurate simulations for engineering necessitates the ability to generate computational meshes representing accurately detailed features of e.g. the tokamak exhaust and heating systems. Consistent with the philosophy of avoiding re-implementing code, the existing meshing capabilities of Nektar++ have been co-opted and augmented (see e.g. [21]) for NEPTUNE. Nektar++ has associated to it a mesh processing suite, called NekMesh, into which several methods of optimizing a 2D mesh for plasma computations have been added under grant work (Fig.3), with extensions to 3D problems a current focus.

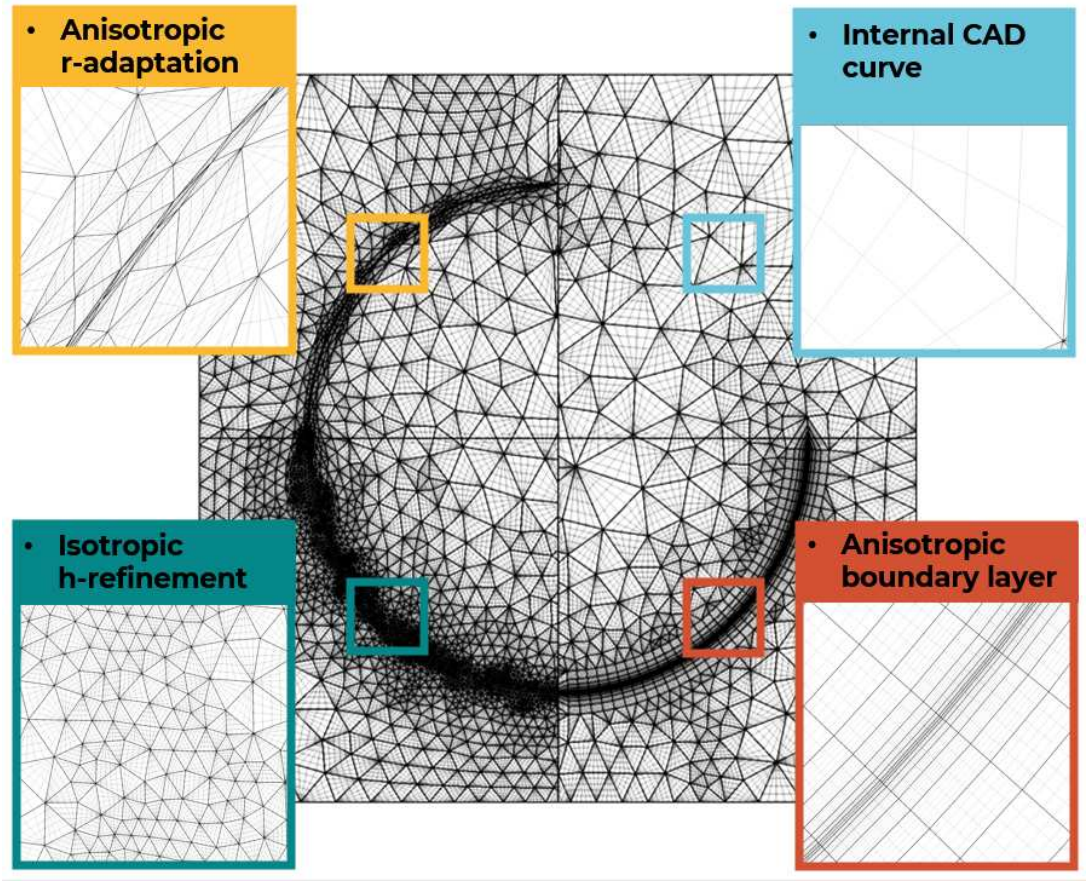

FIG. 3. Demonstration of some of the 2D mesh improvement capabilities added to NekMesh as part of Project NEPTUNE.

6.2. Preconditioning

Another of the grantees has explored the use of preconditioners for specific plasma problems, extracted from either Nektar++ or the BOUT++ simulation framework. This has led to significant improvements (see e.g. [22]). The same grantee has provided extensive support on a range of numerical analysis issues.

6.3. Advanced fluid referent model

One of the grantees has worked towards a new approach to plasma simulation, called moment kinetics. This has resulted in the establishment of a concomitant code base written in Julia demonstrating the new techniques (the use of a type of spectral element here dovetails with the Nektar++ approach). This code, which is already being used to explore new physics, can be found in the repository [23].

6.4. Uncertainty Quantification (UQ)

UQ is necessary for an 'actionable' code that might in future provide outputs to be used by engineers - simply put, it enables error bars to be added. It was decided early on that UQ would be included in a 'non-intrusive' manner - meaning that UQ is performed by running ensembles of simulations and analyzing the statistics of the outputs, as opposed to making internal modification to the code.

The project has benefitted from an extensive collaboration with the SEAVEA (Software Environment for Actionable VVUQ-equipped Exascale Applications) project ([24]). That project maintains a suite of applications to manage code operation on HPC (e.g. campaigns of large numbers of runs intended to extract statistics) and to analyze the derived uncertainties. During the course of NEPTUNE these tools have been applied to a variety of relevant scenarios, including runs from the BOUT++ plasma simulation framework, Nektar++ simulations of heat transfer by fluids, and the classic two-stream plasma instability simulated using NESO-Particles coupled to Nektar++ ([25], and Fig.4). These same techniques will be applied to the more-sophisticated codes that are currently in development (in particular, the coupled 3D plasma turbulence - kinetic neutral particle model).

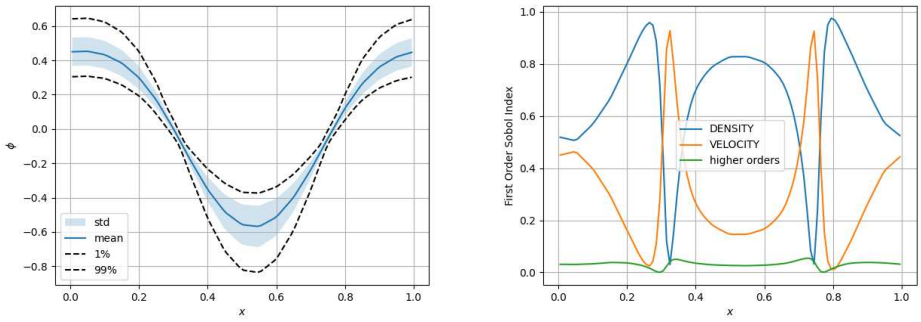

FIG. 4. (Left) mean, variance, and confidence intervals for the position-dependent electrostatic potential in the two-stream instability calculated using a polynomial chaos expansion sampler as a response to specified uncertainty in the initial particle density and velocity. (Right) first-order Sobol indices, which show how much of the output uncertainty is due to each respective input uncertainty, for the particle density and velocity in the same model.

7. OUTLOOK

Project NEPTUNE has involved a large amount of investigation into ways to mitigate known numerical difficulties associated to the numerical simulation of plasmas and also issues that are expected to arise when performing such simulations at or near the exascale (this work is documented in the repository [6]). The project is now entering a phase of rapid code production and it is expected that the outputs will be capable of addressing current research questions in the near future. With further development, the code is expected to contribute to a 'digital twin' capability for future tokamaks on the road to commercially-viable fusion energy production.

The main novel features of NEPTUNE are the use of higher-order numerical methods for fluid simulation and the complementary particle simulation capability, and also the aspiration toward exascale performance portability via SYCL. There is a clear path to extreme-scale simulations. At present, Nektar++ simulations are largely limited to CPU operation, though operation on an Nvidia-based system is scheduled for March 2024 and there is a credible plan in place for a SYCL-based version of Nektar++. In the interim, GPU simulations of certain scenarios are feasible (in cases where the number of particle degrees of freedom dominates that associated to the spectral / hp part of the code).

In addition to the above-described work, it is envisaged that NEPTUNE will include a domain-specific language interface (anticipated to be Python-based) to facilitate the use of the code by a wider audience than software developers (the authors have in mind here the research plasma physicist and tokamak engineering communities).

ACKNOWLEDGEMENTS

The support of the UK Meteorological Office and the Strategic Priorities Fund is acknowledged. The main author would like to note that this article does not provide a comprehensive overview of the broad scope of impactful work carried out by grantees under NEPTUNE.

REFERENCES

REFERENCES

[1] UK Goverment. *Exascale Computing Algorithms and Infrastructures Benefiting UK Research (ExCALIBUR)*. 2023. URL: https://excalibur.ac.uk/.

[2] UK Atomic Energy Authority. *STEP - Spherical Tokamak for Energy Production*. 2023. URL: https://step.ukaea.uk/.

[3] DUDSON, B. *Hermes-3: multifluid drift-reduced model*. 2023. URL: https://github.com/bendudson/hermes-3.

[4] Khronos Group. *SYCL*. 2023. URL: https://www.khronos.org/sycl/.

[5] NEPTUNE community. *ExCALIBUR-NEPTUNE GitHub Organization*. 2023. URL: https://github.com/ExCALIBUR-NEPTUNE/.

[6] NEPTUNE community. *Documents*. 2023. URL: https://github.com/ExCALIBUR-NEPTUNE/Documents.

[7] Nektar++ community. *Nektar++ spectral / hp element framework*. 2023. URL: https://www.nektar.info/.

[8] ONERA. *Coupling With Interpolation Parallel Interface (CWIPI)*. 2023. URL: https://w3.onera.fr/cwipi/bibliotheque-couplage-cwipi.

[9] NEPTUNE community. *nektar-1d-sol*. 2022. URL: https://github.com/ExCALIBUR-NEPTUNE/nektar-1d-sol.

[10] NEPTUNE community. *nektar-driftwave*. 2023. URL: https://github.com/ExCALIBUR-NEPTUNE/nektar-driftwave.

[11] NEPTUNE community. *nektar-driftplane*. 2023. URL: https://github.com/ExCALIBUR-NEPTUNE/nektar-driftplane.

[12] UK Atomic Energy Authority. *NESO-Particles*. 2023. URL: https://github.com/ExCALIBUR-NEPTUNE/NESO-Particles.

[13] SAUNDERS, W. et al. *High-dimensional models: complementary actions 2*. internal report. Culham Science Centre: UKAEA, (2022).

[14] EPOCH developers. *EPOCH*. 2023. URL: https://github.com/Warwick-Plasma/epoch.

[15] WRIGHT, S. et al. "Developing Performance Portable Simulations for the Design of a Nuclear Fusion Reactor". In: *Computer Physics Communications; submitted* (2023).

[16] UK Atomic Energy Authority. *Electrostatic 2D3V Two-Stream Instability Example*. 2022. URL: https://github.com/ExCALIBUR-NEPTUNE/NESO/tree/main/examples/Electrostatic2D3V/two_stream.

[17] UK Atomic Energy Authority. *Simple SOL*. 2023. URL: https://github.com/ExCALIBUR-NEPTUNE/NESO/tree/main/solvers/SimpleSOL.

[18] UCLA. *Large Plasma Device*. 2023. URL: https://plasma.physics.ucla.edu/large-plasma-device.html.

[19] UK Atomic Energy Authority. *NESO-Spack*. 2023. URL: https://github.com/ExCALIBUR-NEPTUNE/NESO-Spack.

[20] UK Atomic Energy Authority. *NESO-Reactions*. 2023. URL: https://github.com/ExCALIBUR-NEPTUNE/NESO-Reactions.

[21] MACMACKIN, C. *NESO-FAME*. 2023. URL: https://github.com/ExCALIBUR-NEPTUNE/neso-fame.

[22] THORNE S. et al. *Timestepping Techniques and Preconditioning for Hyperbolic and Anisotropic Elliptic Problems: Numerical Results*. internal report. STFC, (2022).

[23] Michael Barnes. *moment_kinetics*. 2023. URL: https://github.com/mabarnes/moment_kinetics.

[24] SEAVEA Community. *Software Environment for Actionable and VVUQ-Equipped Exascale Applications*. 2023. URL: https://www.seavea-project.org/.

[25] UK Atomic Energy Authority. *NESO-UQ*. 2023. URL: https://github.com/ExCALIBUR-NEPTUNE/NESO-UQ.