UKRI Science and Technology Facilities Council

Techniques and software relevant to the coupling of continuum (fluid) and particle models of plasma for NEPTUNE

Technical Report 2060049-TN-02
Deliverables 3.1 and 3.3

Hussam Al Daas[*] Niall Bootland[*] Tyrone Rees[*] Andrew Sunderland[†]
Josh Williams[†] Philippa Rubin[†] Sue Thorne[‡] Josh Williams[§]

June 2022 (Revised October 2022 and February 2023)

1 Introduction

The aim of the "Fusion Modelling" use case within ExCALIBUR is the development of new algorithms, software and related infrastructure that will allow for the efficient use of current Petascale and future Exascale supercomputing hardware to draw insights for ITER, and guide and optimise the design of STEP, the UK demonstration nuclear fusion power plant, and related fusion technology. With existing legacy codes not scaling well and not containing all of the required latest physics, significant investment is needed. However, it is important that the different pieces of the modelling and simulation strategy must not only scale well in isolation, these pieces need to couple together and limit any losses in scalability.

1.1 The limitations of existing legacy codes

Figure 1 highlights the areas of the tokamak plasma simulations that are being targeted by the NEPTUNE project. Within the UK, researchers currently make extensive use of fluid-based codes when modelling the edge-plasma region of fusion devices. However, the electrically charged nature of the plasma adds significant complications when using a fluid approach and adds significant uncertainty to the models, particularly, when close to the wall. In these regions, a particle-based (kinetic) method such as particle-in-cell (PIC) is preferable. It is therefore necessary to couple together the two different approaches for different regions of the plasma and this is the focus of this report.

In Section 2, we review some of the methods available within the literature for performing this coupled approach. We then review some of the available code coupling libraries in Section 3 to try to identify "off the shelf" solutions for these problems. We draw our conclusions in Section 4.

2 Methodologies for coupling the fluid and particle simulations

Borodin *et al.* [4] have considered a number of methods for hybridising the fluid and particle (concentrating on the use of EIRENE [37]) approaches. The fluid approach is, in general, used to simulate plasma ions and electrons; kinetic (particle) or hybrid approaches are used for the neutrals including molecular species. The

[*]The authors are with the Scientific Computing Department, STFC Rutherford Appleton Laboratory, Harwell Campus, Didcot, OX11 0QX, UK.
[†]The authors are with the Hartree Centre, STFC Daresbury Laboratory, Sci-Tech Daresbury, Keckwick, Daresbury, Warrington, WA4 4AD, UK.
[‡]Sue Thorne is with the Hartree Centre, STFC Rutherford Appleton Laboratory, Harwell Campus, Didcot, OX11 0QX, UK. Email contact: `sue.thorne@stfc.ac.uk`
[§]Josh Williams with the STFC Hartree Centre, Edinburgh Royal Observatory, Blackford Hill, Edinburgh, EH9 3HJ, UK.

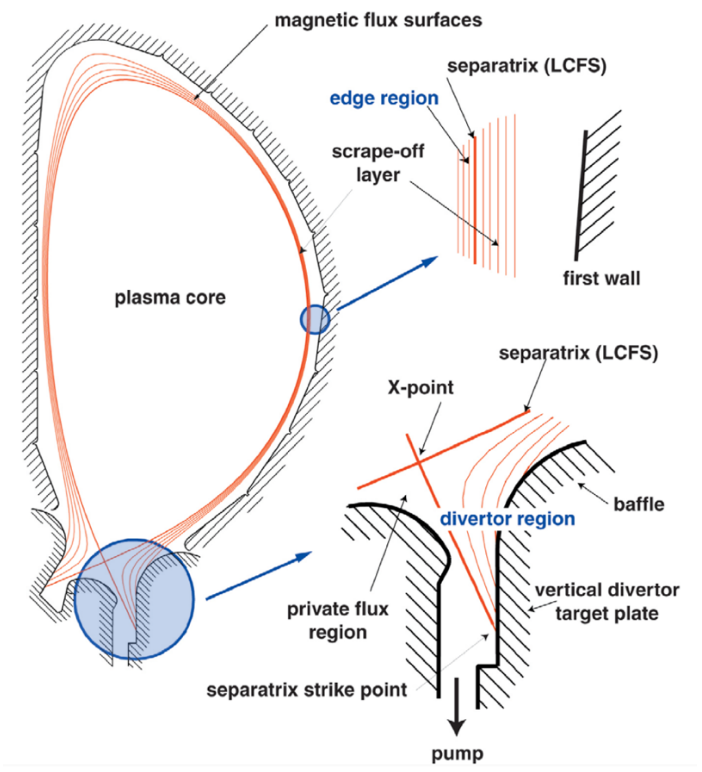


Figure 1: Schematic diagram of a generic tokamak "poloidal cross section". The shaded circles highlight the areas of plasma and first wall that are being targeted by the NEPTUNE project. Attribution: G. Federici et al. [CC BY 3.0 (`https://creativecommons.org/licenses/by/3.0`), minor modifications to figure].

authors do review the possibility of using a purely fluid treatment of the neutral atoms (taking moments of the kinetic equation), which is known as the **advanced fluid neutral (AFN) model**: they conclude that whilst there are benefits in terms of computational costs, the modelling error can be quite significant, particularly in lower recycling regimes.

In the following, it will help to define K_n, the Knudsen number. In fusion-relevant plasmas, let λ be the mean free path for neutrals (expressed in length units) and L be the characteristic linear dimension (expressed in length units), then the Knudsen number is defined as $K_n = \lambda / L$ and is a dimensionless value. For large values of K_n, there is no accurate fluid closure available for the neutrals, which is why a particle approach is used. In high-collision regions, K_n becomes low and there is very strong coupling of the neutrals with the background plasma. We note that different authors may use different values for L and they often relate to properties of the domain under consideration and, in the cases of gases, the compressibility also needs factoring in. Thus, the exact definition of L will need careful consideration for the simulations of interest to the NEPTUNE Programme.

2.1 Spatial Hybridization (SpH) Approach

The regions where K_n is sufficiently small to allow the use of the fluid model typically occur in only a small part of a dense divertor. A number of *spatially hybrid* approaches have been proposed in literature and usually distinguish between the particle and fluid regions, coupling the regions together by using appropriate boundary conditions at the interfaces: how this is done varies between the different methods.

The choice of interface between the two regions poses interesting questions. It is reported that it must be such that the boundary conditions can be defined based on fluid-like distributions, which is where the Knudsen number is required [4], but if a naive method is used for coupling the regions, we could lose too much computational speed. When considering Exascale simulations, it will be important to spread the work for the coupling activity, and particle and fluid approaches across an HPC system to optimally load balance the whole simulation. We discuss this further in Section 3.

SpH methods are not just applicable to fusion problems. For example, the space craft re-entry problem is a hot topic at the moment. The modelling of the hypersonic flows around the re-entry vehicle is one of the most challenging problems within aerospace engineering. In [4], the plasma domain is separated into two regions with the boundary conditions being used to couple within the interface. In the space craft re-entry problem, there are three different regions [16]: (i) for $K_n < 0.001$, the gas can be regarded as a continuum; (ii) for $K_n > 10$, there is free-molecule flow, and (iii) for intermediary values, there is a slip-transitional flow region, where both molecular transports and collisions are just as important. Thus, in this slip-transitional flow region, there are two possibilities: (i) perform both continuum and free-molecule flow simulations in this region and then set this region as the coupling interface with the coupler combining the two simulations, or (ii) explicitly do a Micro-Macro Hybrid method (Section 2.2) in this region and then use the coupler to transfer information between the different regions. The former is considered in [39], which is encompassed in the Modular Particle Continuum (MPC) hybrid approach and demonstrated for a number of test examples including the simulation of hypersonic nitrogen flow over a cylinder. The MPC approach is also used for examples without a slip-transitional flow regions.

2.2 Micro-Macro Hybrid (mMH) Approach

In [12], Horsten *et al.* propose splitting the (complete) neutral distribution function f into a fluid part f^f and a kinetic correction f^δ such that $f = f^f + f^\delta$ holds across the entire simulation domain. For the fluid-part, the approach leads to the AFN equations but the kinetic correction leads to additional corrections on source and transport terms. A modified kinetic simulation is used to obtain the kinetic correction: positive and negative correction particles are used with net zero density and momentum, which provide exact closure for the AFN model. This approach has benefits, namely, the method has been shown to be produce solutions that are equivalent to solving the fully kinetic equations and is independent of the recycling regime used. However, the method requires a significant amount of developmental effort and the fluid method will need to act across the whole of the domain instead of just a subdomain. The use of such an approach would require significant changes in the approach currently being used by some of the NEPTUNE projects but, given the interesting nature of this methodology, it is being explored as part of one of the other NEPTUNE Projects (Oxford), [3, 28–30].

2.3 Kinetic-Diffusion Monte Carlo Method (KDMC)

The Kinetic-Diffusion Monte Carlo (KDMC) scheme is an asymptotic-preserving scheme that is fully Monte Carlo. Each Monte Carlo particle alternates between acting according to a fluid limit (moving via a random walk that incorporates a Monte Carlo discretization of the limiting fluid equation) and acting in a kinetic manner. Whilst this method has merit in that it does not require intricate couplings between fluid and kinetic neutral sub-domains, it has so far been only demonstrated on simplified problems involving single-species scattering or absorption, for example, [2]. The decision to use such a method would result in a significant change of steer for the NEPTUNE project.

3 Software libraries for code coupling

In his 2015 paper on coupling, Tang [42] neatly summarised a dilemma for code developers wishing to create coupled multi-physics models: "Despite existing theoretical developments, implementing concurrently coupled simulations remains difficult. On one hand, hard-coding remains commonplace in projects that employ ad hoc coupling approaches. Such practice can quickly become an obstacle when further development is needed. On the other hand, despite the richness of available coupling schemes and tools, adapting existing code to meet the programming interface specification of a coupling framework frequently leads to code refactoring that consumes a substantial amount of man-hours". Since 2015, a wide range of new software coupling libraries have been developed to largely mitigate these difficulties. Many coupling libraries are now minimally invasive to the underlying computational models. Many also include high-level Application Programming Interfaces (APIs) with support for a range of programming languages and are based on MPI (the ubiquitous message-passing standard) for data exchange management.

Given the potential disruption of applying the coupling approaches described in Section 2.2 and 2.3 to the existing code base, the SpH (Section 2.1) approach may be favourable. If SpH is to be used, there are a range of code coupling libraries already available, therefore rendering it unnecessary for the NEPTUNE project to develop its own. A preferable solution is a generalised approach using code coupling middleware or libraries.

In this section, we review some of the available libraries and note that the Software Outlook Project [41] performed an extensive review of a number of code coupling libraries [38] and refer the reader to performance comparisons within their report for further details.

3.1 The advantages of using Coupling libraries with a partitioned approach

The main advantage of using third-party coupling libraries is that they provide a framework for a *separation of concerns* for complex workflows, where individual application codes can remain separate entities. With the provision that data provided to the coupling middleware remains consistent, each application model can be developed in isolation of the other. This also results in increased potential for HPC optimisations. For example, load-balancing algorithms can be applied separately to each application domain, individual computational grids can be redefined, coarsened or refined, and the different applications can even be run on separate platforms if appropriate. It should be noted that where heterogeneous CPU/GPU coupling schemes are implemented using the current class of code coupling libraries, the coupling framework software itself resides on the CPU platform. Most coupling libraries provide high-level APIs which ensure ease of use and facilitate high degrees of flexibility. The data exchange between the coupled applications usually takes place within standardized tools for message passing e.g. MPI, thereby minimizing the invasiveness of the integration. Moreover, the library approach usually ensures that the data exchange is undertaken in an efficient manner. These properties lead to high-performance and portability. We note that within the NEPTUNE Programme, the multiple domains are planned to use the same underlying mesh structure at the coupling interfaces.

Other features of code-coupling libraries include:

- Dynamic load balancing of the coupling interface across processors. This may become especially important for high-performance computing at scale.
- Variable overlapping of domains.
- Support for dynamic meshes (vs static meshes).
- Multi-language APIs.
- Support for multiple communication methods (MPI, TCP/IP).
- Multi-platform, heterogeneous HPC environment coupling (e.g. CPU/GPU).
- Support for coupling multiple applications.
- Open source or free-to-use software.

Presentations on many of the coupling libraries discussed in this report are available on recordings from the Excalibur Online Workshop *The State of the Art in Scientific Coupling Technology for HPC* [44].

3.2 MUI

The Multiscale Universal Interface (MUI) is an open source code-coupling library, which was originally developed by Brown University [42]. Originally designed for multiscale coupling, the software is also applicable to multi-physics coupling of problems at the same scale.

At present MUI's primary developers and maintainers are UKRI-STFC [11]. The MUI project aims to create "...a light weight plugin library that can glue together essentially all numerical methods including, but not limited to, Finite Difference, Finite Volume, Finite Element, Spectral Method, Spectral Element Method, Lattice Boltzmann Method, Molecular Dynamics, Dissipative Particle Dynamics and Smoothed Particle Hydrodynamics" [42]. MUI developers claim that the coupling process is so lightweight and minimally invasive that MUI can be integrated into a solver using only ten lines of code.

MUI provides a small set of programming interfaces to conduct send and receive messages between domains. As such, it does not put restrictions on the multi-physics solvers themselves; "MUI also follows PLE's philosophy of not prescribing how the physics of a problem should be coupled, only that coupling should be achieved by passing data through an interface. Assuming the data in question is associated with a point structure then the only requirement is that the data can be associated with a single point in space" [17]. Following

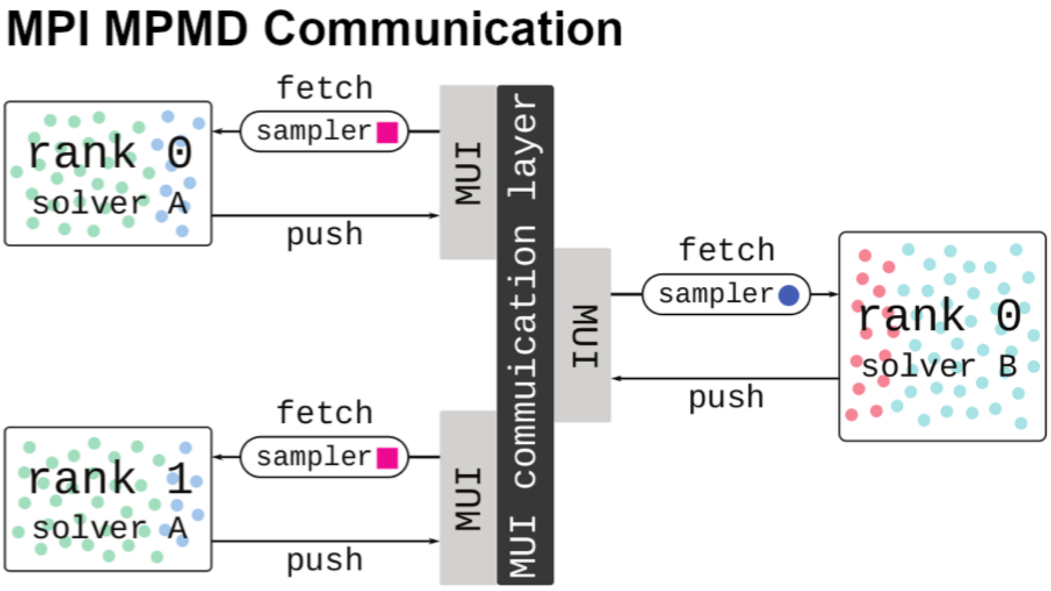


Figure 2: The exchange of information across solvers in MUI via *push* and *fetch* data points [42].

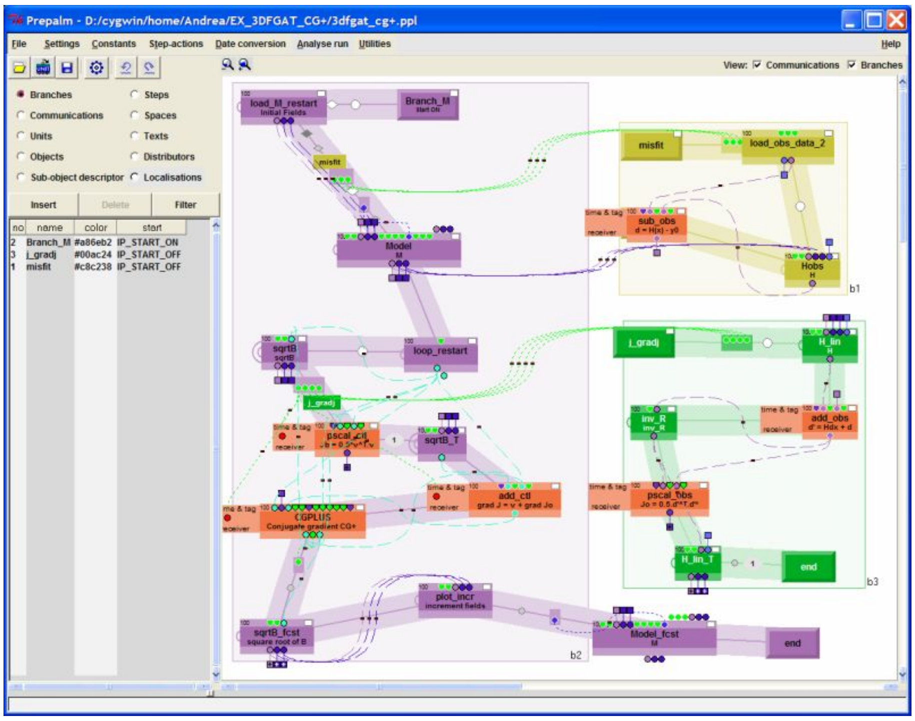
its original design targeting multi-scale problems, MUI does not rely on a computational mesh approach, instead using a particle paradigm for its data transport layer. In [42], the author demonstrates the use of MUI applied to three test problems where the analytical solution is known: (i) Couette flow: Smoothed Particle Hydrodynamics–Smoothed Particle Hydrodynamics coupling; (ii) Soft matter: Smoothed Particle Hydrodynamics–Dissipative Particle Dynamics coupling, and (iii) Conjugate heat transfer: energy-conserving Dissipative Particle Dynamics-Finite Element coupling. Whilst no values are given that compare the error of the simulations with the analytical solutions, visually, there is a good match.

The MUI software is written in C++11 with wrappers for C, Fortran and Python. A header-based approach is used for the entire library and the only external library is the Message Passing Interface (MPI). As such, it can be used in the same way any other C++ standard library would be used, without the need for pre-compilation. However, it has the bonus advantage of not interfering with pre-existing MPI communications within the pre-existing solver. MUI is open-source software, dual licensed as either GPLv3 or Apache 2.0. The library employs techniques such as dynamic typing, MPI MPMD execution, asynchronous I/O, generic programming and template metaprogramming to improve both performance and flexibility. As shown in Fig. 2, MUI assumes a push–fetch workflow and serves as the data exchange and interpretation layer between solvers. A data sampling construct is provided for the coupling interface. For example, this may be used where there exists no one-to-one correspondence between data points at the interface of the coupled applications and therefore some sort of weighted interpolation technique is required. Using the flexible data interpretation engine provided, users can plug in the most appropriate interpolation algorithm with minimal effort.

MUI has a well maintained website with documentation, example applications, demonstrations and tutorials [21]. MUI also provides a testing framework which allows a wide-range of complex coupling scenarios to be simulated via artificial benchmarking [20]. MUI is currently under development as part of the ExCALIBUR projects. The software optimisation is focusing on memory overhead reductions, improved MPI usage and better GPU integration. At the time of writing it is also planned that MUI will be made available as part of the Nektar++ Spectral/HP framework [24].

3.3 OpenPALM

OpenPALM (Projet d'Assimilation par Logiciel Multimethodes) is another coupling library that enables a user to execute components of code concurrently with communication between them. There are two parts to OpenPALM: PrePALM and PALM. PrePALM is the graphical user interface, and PALM is the driver of coupling framework itself. PALM uses MPI communication to handle the exchange of data for the coupling algorithm.



no	name	color	start
2	Branch_M	#a86eb2	IP_START_ON
3	j_gradj	#00ac24	IP_START_OFF
1	misfit	#c8c238	IP_START_OFF

Figure 3: The PrePALM Interface. This example was sourced from OpenPALM's website [27].

On the OpenPALM website, it states they have had simulations of 130,000 cores on Titan, 130,000 cores on Turing and 12,000 cores on Curie [27]. Keyes et al. [13] include OpenPALM as one of the successes in multiphysics software. OpenPALM does not appear to be restricted to particular scientific fields and it has a lengthy user guide [19]. Therefore, OpenPALM maybe of interest to the NEPTUNE Programme.

The OpenPALM team is joint between CERFACS and ONERA. CERFACS have also created another coupler called OASIS [25] but it is dedicated to geophysical applications. In contrast to OASIS, OpenPALM provides a more generic interpolation framework based on an unstructured mesh formalism [6], provided by the CWIPI library [43]. CWIPI can be used independently of the OpenPALM interface, see Section 3.6 for further information.

OpenPALM applications are implemented via a user interface called PrePALM. The user designs their coupling algorithm into sequential and parallel sections, loops, conditional executions, and communication between components. The designed algorithm is presented to the user clearly in a window as seen in Fig. 3. OpenPALM stress you can create parallel code "...without anything to know about MPI, only by drawing! It is one of the PALM features: one can make parallel computing without any further specific knowledge" [19].

3.4 preCICE

The Precise Code Interaction Coupling Environment (preCICE) [34] is a coupling library developed by the University of Stuttgart and the Technical University of Munich. It is another available open source coupling library that aims to couple existing solvers together, creating what is known as 'partitioned' simulations. In this way, they have similar motivations to MUI (Section 3.2) and aim for the highest flexibility possible in reusing existing components. Their team have particular interests in fluid-structure interaction and conjugate heat transfer simulations, but they stress that they are not limited to such fields. The preCICE API has wrappers for C++, C, Fortran, Python, Matlab and Julia and the project used a policy of continuous integration with automatic testing undertaken at all levels.

The software supports coupling of CFD, FEM, Particle (with background meshes) and in-house solvers using an *adapter* to interface with the preCICE coupling library, as shown in Fig. 4. This adapter is a stand-alone software package that may either be provided by preCICE (for commonly used solvers) or may be user-defined (e.g. for in-house solver methods). The preCICE library provides software for communication, data mapping, coupling schemes and time interpolation. Coupling meshes are defined only once during initialization, so

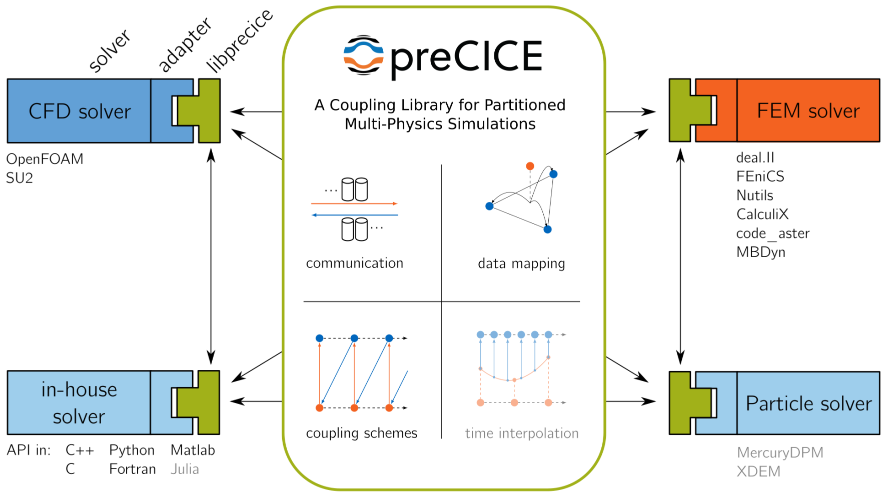


Figure 4: The preCICE Coupling Methodology [33].

preCISE is not suitable for problems where the coupling mesh may be modified during the simulation.

To couple your code with preCICE, an input file must be developed. preCICE is configured with an XML file: a reference guide is available [35]. One particular difference between preCICE and the other coupling libraries that we discuss is that in this XML file, you can establish an `m2n` communication channel (i.e. from `m` processes of `Solver1` to `n` processes of the `Solver2`) based on TCP/IP sockets. It is possible to configure preCICE to use MPI ports if you wish, but the preCICE team recommend using `sockets`. Once this XML configuration is complete, the user can then insert calls to the preCICE API in original solvers and, at this stage, preCICE is very similar to MUI. In [38], Rubin documents the steps that she had to go through to install preCISE on ARCHER2 [1] and reports that it was not straightforward.

preCICE has extensive user documentation available with an active community exchange chat page and forum. The website also has articles on many coupling projects and includes detailed performance analyses from HPC systems. It is used by over one hundred research groups in academia and industry. Ongoing and future planned developments include support for dynamic-adaptive meshes (i.e. re-initialization), higher-order data mapping and additional support for mesh-particle coupling.

3.5 PLE

The Point Location Exchange library, PLE, is part of Code_Saturne [8]: an open-source CFD software released by EDF, available under an LGPL licence. PLE was specifically designed to simplify parallel couplings using a minimalist API and and minimal dependencies. Both volume and surface-based multiple couplings (with overlap) are supported. Data is transferred between the coupled solvers at specific spatial points [17]. The library has much in common with other coupling libraries, such as CWIPI, MUI and preCICE [10]. It is used to couple Code_Saturne to other EDF codebases, such as EDF's thermal software SYRTHES [9]. Code_Saturne is available on GitHub [7], with PLE included in this repository [32].

The PLE developers have developed a coupling framework through which data can be exchanged between different solvers. To do this, PLE creates an interface between the solvers and an MPI intra-communicator is created for each interface, thus integrating with pre-existing MPI functionality within the original solvers. Although MPI is currently required, the library could easily be extended to other communication libraries with only minor changes. Each coupled code provides their own set of local points in a mesh location algorithm. The mapping of data from one mesh to another is done independently for each coupling, within the associated MPI intra-communicator. In this way, PLE works in a very similar way to MUI (Section 3.2) and CWIPI (Section 3.6). A PLE *locator* object tracks this mapping. In cases of partial overlap, different variables may be exchanged either both ways or one way only on some part of the computational domain. A lightweight *coupling set* object is defined which is used to synchronize time stepping status across multiple codes. Time stepping can involve both *locators* object types and *coupling sets* object types for field data exchange and global synchronization respectively. Meshes can have any type of elements, which do not have to match each side of the interface.

Data points at the interface could be mesh vertices, face-centred or cell-centred elements, or a representation of independent particles. Each code defines its own interpolation algorithms.

Aside from the Doxygen documentation, PLE does not have any extensive documentation or user guide. However, there is a brief description of the software at [31]. With other coupling libraries having much in common in terms of technical specification, PLE's limited documentation means that it is not one of our recommended libraries for use within the NEPTUNE Programme.

3.6 CWIPI

The CWIPI (Coupling With Interpolation Parallel Interface) library can be used for either chaining and coupling application codes in a massively parallel distributed computing environment (chaining is defined as being when the characteristic times of the physics studied in the individual solvers are very different and the physics must therefore be processed consecutively). It ensures the exchange of interpolated fields through a geometric coupling interface and allows the control of the coupling algorithm using control parameters. The development of CWIPI started in 2009, based on the FVM library at EDF, now developed into PLE [32], also described here. It therefore shares many common approaches and features with PLE. Call interfaces are provided to C/C++, Fortran, Python/Numpy and Python/CGNS and the software is freely available under an LGPL3 license. The main CWIPI website [43] currently has only limited documentation available in French only. CWIPI is currently the only coupling software available in the Spectral/HP element framework Nektar++ [40].

There are two alternative operating modes for CWIPI - via direct calls or via the OpenPALM coupler. When using the direct call mode, the coupling algorithm is defined by the different codes, with no dedicated coupling processes. This solution is less flexible than the use of a coupling manager but it has the advantage of directly processing the MPI-based data exchanges between the codes without going through another intermediate code, which limits the memory copies and therefore ensures better performance, especially massively parallel computing environment. When using the OpenPALM route, the coupling algorithm is defined using the OpenPALM GUI and driven using the OpenPALM coupler. Both MPI-based communications and TCP/IP communications are available for use in this mode.

When using CWIPI, both coupled codes are run under the same MPI environment: the library creates MPI intra-communicators for each coupled code. Multiple parallel codes are executed within a single MPI run using the same MPI implementation (this is a constraint that the CWIPI developers are currently attempting to eliminate). No predefined coupling schemes are provided. Instead users define their own coupling scheme using CWIPI control parameters. These are global variables that are shared and synchronised between the codes.

CWIPI constructs a point-to-point communication graph between the multiple codes. This graph is then used to exchange interpolated fields using either blocking or non-blocking MPI communications. Elements of any geometric order are supported (line, surface, volume). In advanced use, the geometric order and the order of the numerical method can be disassociated. As with other coupling libraries, meshes can have any type of elements which do not have to match each side of the interface. Data points at the interface could be mesh vertices, face-centred or cell-centred elements, or a representation of independent particles. Recently, the geometric localization algorithm has been redesigned to use a parallel distributed octree structure for dynamic load-balancing of the coupling interface across processors. this has resulted in much improved parallel performance, particularly on large-scale processing platforms.

In [15], Langenais *et al.* consider an example that couples the Navier-Stokes equations with the Euler equations. They solve the Navier-Stokes equations by using a cell-centered finite volume method on generalized unstructured grids, solve the nonlinear Euler equations by using a nodal discontinuous Galerkin method (DG) on unstructured grids, and use CWIPI to couple the regions. Each region uses the same spatial discretisations within the interface region. Good accuracy of solution is demonstrated for a number of test problems including pulsating sphere, wave guide, standing wave in cavity, and shock tube examples. Lackhove [14] provides a detailed explanation around the use of Nektar++ and CWIPI for aeroacoustic problems where the two regions have different time steps and the additional algorithmic steps used to reduce the induced error from the coupling.

Both of CWIPI's operating modes are currently being used in coupling projects by engineers, researchers and developers primarily in the aerospace domain. CWIPI is notably used at ONERA [26] in unifying "lightning" and "plasma" research projects, in the " elsA – CEDRE synergy" research project [5], for post-processing of the CEDRE code, and for aeroacoustic couplings (CEDRE / CEDRE coupling). Current users also include Safran, Siemens, ArianeGroup and and Airbus CERFACS, CNES Meteo France and the University of Surrey.

A major new version of CWIPI is expected to be released in late-2022 or early-2023, which will contain new features including (i) non-blocking synchronization of control parameters using MPI RMA (Remote Memory Access), (ii) access to multiple types of spatial interpolation methods, (iii) arbitrary distributions of the coupled codes to processes, i.e. processes will be able to run multiple applications.

In Appendix A, we provide observations and results from preliminary experiments that couple Nektar++ and a simple particle-style code with an interface that involves CWIPI.

3.7 MUSCLE3

MUSCLE3 is the third version of the Multiscale Coupling Library and Environment, developed by the University of Amsterdam [22]. Most of the libraries discussed here target tightly-coupled scale-overlapping multiphysics simulations and emphasise efficient execution on high-performance computers. According to its principal reference [45] "The MUSCLE3 framework takes a somewhat different approach, focusing on *scale-separated* coupled simulation. These types of coupled simulations have specific communication patterns which occupy a space in-between tightly-coupled, high communication intensity multiphysics simulations, and highly parallel computations in which there is no communication between components at all". MUSCLE3 connects simultaneously running instances of different sub-models together based on an MMSL (Multiscale Modelling and Simulation Language) description of how to do so, which can be expressed in the form of a diagram. The software is particularly suitable for undertaking efficient VVUQ (Verification, Validation, Uncertainty and Quantification) analyses of ensemble multiscale simulations.

MUSCLE 3 consist of three components: **libmuscle**, **ymmsl-python**, and **muscle-manager**. **libmuscle** is a library that is used to connect compute element implementations (e.g. new or existing models) to a coupled simulation. It offers functions for sending and receiving messages to and from the outside world, determining whether to restart the model at the end of its run, and various utilities like centralised settings (for convenience and UQ), centralised logging (to make debugging easier) and basic profiling. **ymmsl-python** is a Python library that contains class definitions to represent an MMSL model description. **muscle-manager** is the central runtime component of MUSCLE 3. It is started together with the compute element instances, and provides a central coordination point that the instances use to find each other.

MUSCLE3 can be used with sockets-based communication or MPI communication. It can be built as a C++ library, with wrappers to Python, or a Fortran library. The documentation provided [22] is quite extensive, with several tutorials and examples provided for different configurations. MUSCLE3 allows peer instances to find each other, and then transmit messages between themselves in a peer-to-peer fashion. Communication between the models therefore does not suffer from typical bottlenecks created by a central component. MUSCLE3 also supports time-domain adjacency.

4 Conclusions

To conclude, of the three algorithmic approaches considered within Section 2, the spatial hybridization (SpH) and micro-macro hybridization (mMH) approaches are of the most interest. The latter approach would involve significantly larger software development time within the NEPTUNE Programme, with the current set-up of NEPTUNE more closely aligning to the SpH approach. Additionally, the open source libraries considered in Section 3, align with the SpH approach, which should also reduce the software development efforts required so long as the fluid and kinetic/particle projects are scoped in such a manner that enables the use of one of these libraries. Many relevant general features of the available coupling libraries have been discussed in this report, however problem-specific coupling questions, such as the required depth of overlapping regions required for stable, accurate simulations when coupling fluid and kinetic fusion-based models will require further analysis and experimentation [36]. The general attributes of most appeal for this project are that the coupling software used is (i) lightweight and minimally invasive; (ii) provides APIs for several programming languages; (iii) supports coupling fluid and kinetic/particle models and (iv) performs well at scale. Of all of the libraries discussed here, we find the attributes of MUI and CWIPI to be the most favourable for NEPTUNE. With the major new version of CWIPI on the horizon and it already being the favoured coupler within Nektar++, we will be building a proxyapp that uses CWIPI to couple together fluid and kinetic approaches within the Nektar++ framework. We will then use this proxyapp to investigate how large the overlap of the domains needs to be. In Appendix A, we provide our initial experiences and results from successfully coupling a simple particl-style code to Nektar++, where the coupling interface uses CWIPI.

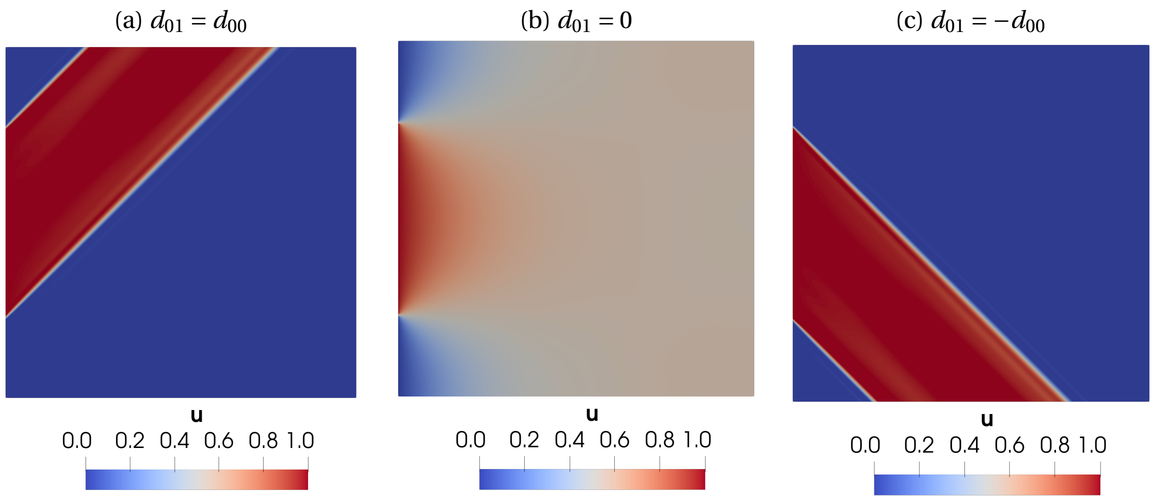


Figure 5: Visualisation showing the effect of varying the diffusion coefficient sent from simple-particles to Nektar++ on the fluid velocity in Nektar++. The diffusion coefficients are spatially uniform in all three cases.

A Coupling implementation with CWIPI

We have created a coupling interface to send information from an external solver to Nektar++ using CWIPI. To later be replaced by NESO-Particles (in development by UKAEA), we have created `simple-particles` which creates a 2D mesh and writes nodal values that are then passed to Nektar++ via CWIPI. This aims to represent the expected data obtained from a particle-in-cell (PIC) method that can be coupled to the continuum solver at the interface between the plasma and the cold neutral gas in the edge region (Figure 1). We have used this approach to develop the coupling interface and show its application to two cases: (i) writing global and spatially-varying diffusion coefficients, (ii) computing a source term for fluid velocity based on received temperature field.

A.1 Implementation

Coupling an external solver such as `simple-particles` to Nektar++ requires the addition of five functions to the solver. The first function is an initialisation of the coupler that defines the mesh and relevant simulation control parameters (`cwipi_coupling`), a function to send nodal values to Nektar++ (`cwipi_send`), as well as a function to wait for sending to finish (`cwipi_wait`). Similarly, we defined a function for receiving data (`cwipi_receive`) as well as a function to wait for the receiving of data to have completed (`cwipi_receive_wait`). Finally, there is a function to terminate the coupling once the both simulations have finished (`cwipi_exit`). This is similar to the OpenFOAM-Nektar++ one-way coupling by Moratilla-Vega et al. [18]. Our source code for this is available on Github [23].

A.2 Coupling results

In our first example, we illustrate that Nektar++ can receive data from our coupling interface by varying the anisotropic diffusion coefficient d_{01}. Specifically, the two isotropic components are equal $d_{00} = d_{11} = 10^{-3}$ and the ratio d_{01}/d_{00} was varied from 1 to -1. The value was applied uniformly across the domain. In Figure 5, we observe that the diffusion coefficients were successfully sent from `simple-particles` to Nektar++. When $d_{01} = d_{00}$, the velocity field follows a 45° angle (Figure 5a). When $d_{01} = -d_{00}$, the flow has a -45° angle (Figure 5c). When $d_{01} = 0$, the flow is symmetrical due to no anisotropic component (Figure 5b).

In our second diffusion example, we define a spatially varying d_{01} value. As a simple representation of the interfacial region discussed in Section 2, we define an interfacial region between x_{i0} and x_{i1}, over which d_{01} varies linearly from $d_{01} = d_{00}$ to $d_{01} = 0$. Therefore, near the end of the interface and downstream the flow should appear much more blurred (similar to Figure 5b). This is observed in Figure 6, as we observe that the flow becomes significantly more blurred as the size of the interface grows.

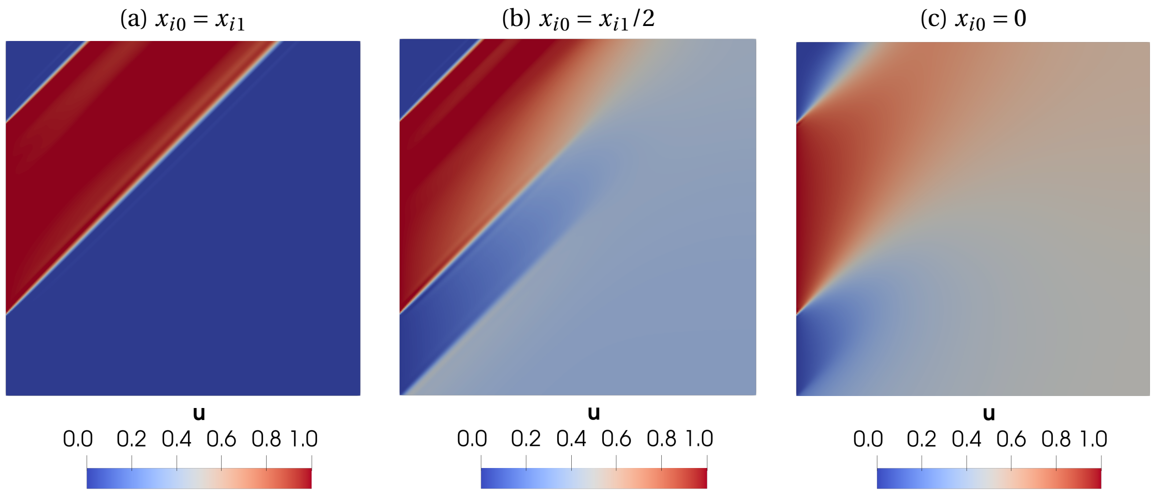


Figure 6: Visualisation showing the effect of spatially varying the diffusion coefficient, where the diffusion coefficient is change based on the nodal x coordinate. In all cases, the interface end (x_{i1}) is at the simulation outlet (right boundary). Panels show varying interface start point, x_{i0}.

In our third case, we illustrate that the coupling interface can send and receive information throughout the simulation. This is shown for the Navier-Stokes equation solver. We provide a source term to the solver from CWIPI. This requires defining a dummy variable which is used to store the information sent to Nektar++. The source term is then found by evaluating the symbolic relationship for the source term defined in the Nektar++ simulation configuration file. In our case, this simply evaluates $F_u = \alpha_{cwipi}$ where α_{cwipi} is a dummy variable. This must be prefixed with "dummy" in the Nektar++ configuration file to avoid being treated as a normal variable and therefore integrated by the advection solver. This check for the "dummy" string is implemented in `VelocityCorrectionScheme.cpp`.

Our simulation aims to replicate a case provided by UKAEA, where a ball of hot gas is initialised and rises upwards based on the Grashof number, $Gr = 5 \times 10^9$. The source term for velocity is $F_u = GrT$, where T is the temperature. In `simple-particles`, we receive T, multiply it by Gr and send it back as a source term F_u. In Figure 7, we quantify the influence of the `simple-particles` solver mesh resolution on the temperature field computed by Nektar++. The Nektar++ mesh had $N = 102400$, where N is the total number of cells. This gave $N_x = 320$, where N_x is the number of cells per axis. The `simple-particles` resolution was then varied relative to this, from $N_x = 10$ to 256. At the coarsest resolution, $N_x = 10$, the maximum error in the Nektar++ temperature field was 27% and the root-mean-squared (RMS) error was 1.7%. At $N_x = 50$ (6.4 times coarser than the Nektar++ mesh), the maximum error reduced to 5%. This shows that the mesh resolution for the PIC solvers in future coupling studies could be significantly less than the continuum solver's mesh resolution without incurring significant error.

To visualise the influence of the `simple-particles` mesh resolution on the temperature field, we show the temperature and source term for a fine (Figure 8a,c) and coarse mesh (Figure 8b,d). The hot gas rises following a classic plume structure in the fine mesh simulation (Figure 8a). On the coarse mesh, the plume structure appears very different (Figure 8b) due to inaccuracies in the source term provided to Nektar++ (Figure 8d). The maximum relative temperature error on the coarse mesh is 26% as shown in Figure 7.

References

1. *ARCHER2 Website* `https://www.archer2.ac.uk/`.
2. Arter, W. & Eastwood, J. Particle-mesh schemes for advection dominated flows. *Journal of Computational Physics* **117,** 194–204 (1995).
3. Barnes, M., Parra, F. I., *et al. Numerical study of 1D drift kinetic models with periodic boundary conditions* tech. rep. 2047357-TN-02-02 (UKAEA Project Neptune, 2021).

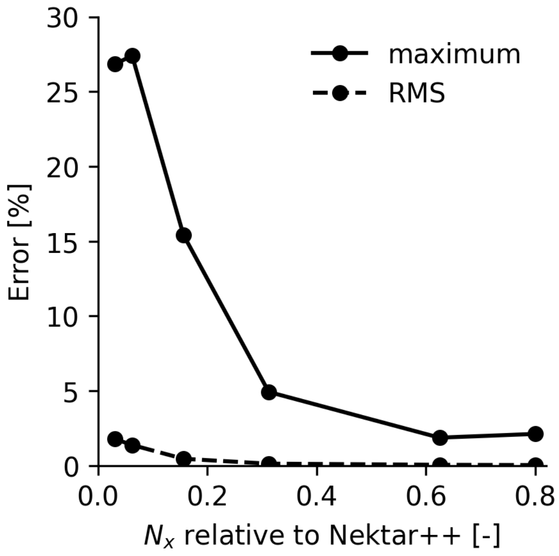


Figure 7: Maximum and mean-square error for temperature, relative to baseline simulation provided by UKAEA (where source term is explicitly declared in Nektar++ XML file). If the mesh resolution of the external solver is too coarse the temperature field will have high maximum error. Root-mean-squared (RMS) error is always negligible. Error is evaluated at $t = 2\mu s$ (1000 iterations).

4. Borodin, D. V., Schluck, F., *et al.* Fluid, kinetic and hybrid approaches for neutral and trace ion edge transport modelling in fusion devices. *Nuclear Fusion* (2021).
5. *CEDRE France, Research Institute* `http://wwz.cedre.fr/CEDRE`.
6. *CERFACS Research Areas: Code Coupling and User Interface* `https://cerfacs.fr/en/code-coupling-and-user-interface/`.
7. *Code Saturne GitHub Repository* `https://github.com/code-saturne/code_saturne`.
8. *Code_Saturne Website* `https://www.code-saturne.org/cms/`.
9. *EDF R&D Simulation Software Webpage* EDF France. `https://www.edf.fr/en/the-edf-group/world-s-largest-power-company/activities/research-and-development/scientific-communities/simulation-softwares`.
10. Fournier, Y. Massively Parallel Location and Exchange Tools for Unstructured Meshes. *International Journal of Computational Fluid Dynamics* **34,** 549–568 (Sept. 13, 2020).
11. *Hartree Centre - Multiphysics and Multiphase Code Coupler | Scalable and flexible code coupling* `https://www.hartree.stfc.ac.uk/Pages/scalable-and-flexible-code-coupling-.aspx`.
12. Horsten, N., Samaey, G., *et al.* 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).
13. Keyes, D. E., McInnes, L. C., *et al.* Multiphysics simulations: Challenges and opportunities. *The International Journal of High Performance Computing Applications* **27,** 4–83 (Feb. 1, 2013).
14. Lackhove, K. *Hybrid Noise Simulation for Enclosed Configurations* PhD thesis (2018).
15. Langenais, A., Vuillot, F., *et al.* Assessment of a Two-Way Coupling Methodology Between a Flow and a High-Order Nonlinear Acoustic Unstructured Solvers. *Flow, Turbulence and Combustion* **101,** 681–703. `https://doi.org/10.1007/s10494-018-9928-0` (Oct. 2018).
16. Li, J., Jiang, D., *et al.* Kinetic comparative study on aerodynamic characteristics of hypersonic reentry vehicle from near-continuous flow to free molecular flow. *Advances in Aerodynamics* **3,** 10. `https://doi.org/10.1186/s42774-021-00063-0` (May 2021).

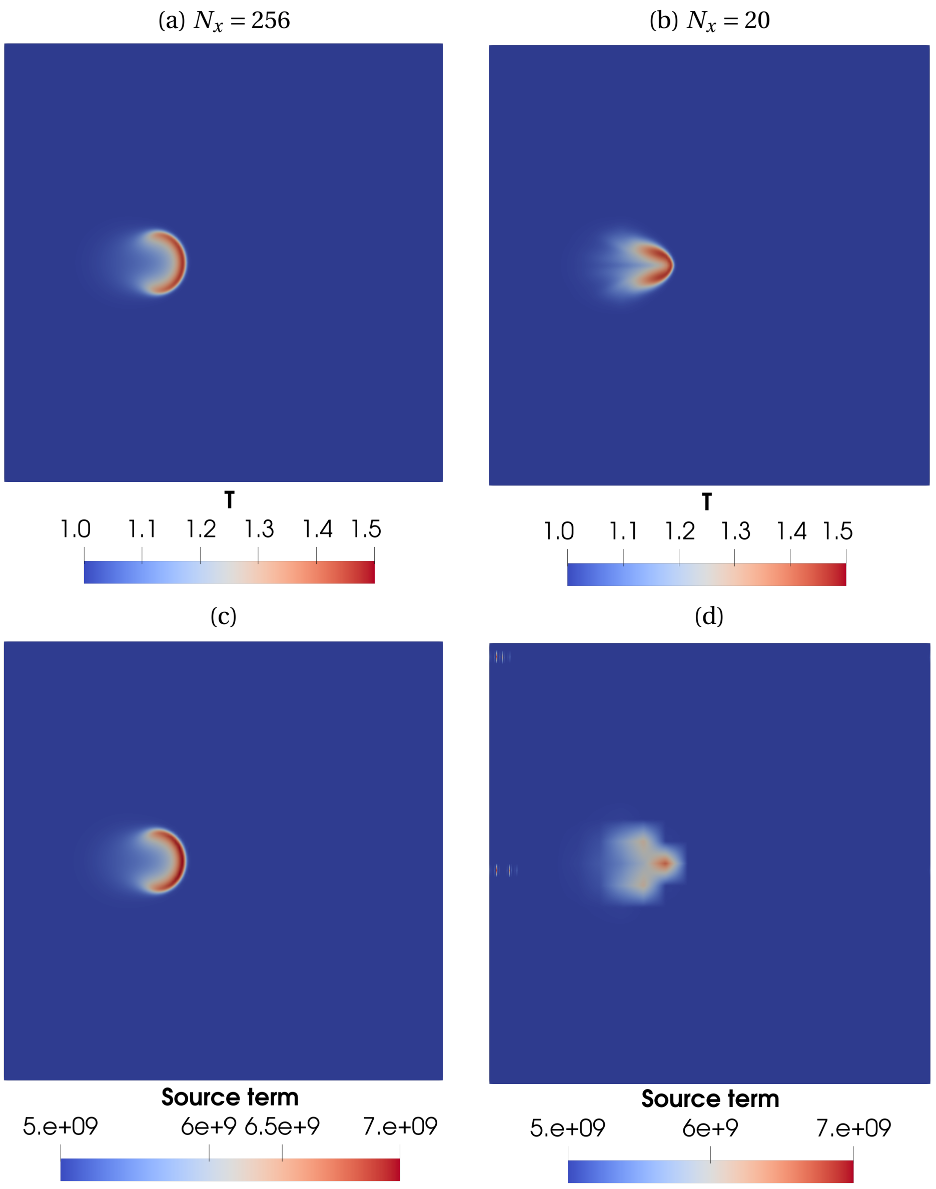


Figure 8: Temperature and source term fields with varying mesh resolution. Rows show (a,b) temperature, and (c,d) velocity source term. Columns show (a,c) $N_x = 256$, (b,d) $N_x = 20$. Figures are shown at $t = 2\mu s$ (1000 iterations).

17. Longshaw, S., Skillen, A., *et al.* in, 28 (May 30, 2017). ISBN: 978-1-874672-72-2.
18. Moratilla-Vega, M. A., Angelino, M., *et al.* An open-source coupled method for aeroacoustics modelling. *Computer Physics Communications* **278,** 108420 (2022).
19. Morel, T., Duchaine, F., *et al. OpenPALM coupler version 4.3.0 User guide and training manual* `https://www.cerfacs.fr/globc/PALM_WEB/pdfs/user_guide.pdf` (Apr. 2019).
20. *MUI Demos* `https://github.com/MxUI/MUI-demo`.
21. *MUI Website* `https://mxui.github.io`.
22. *MUSCLE3 Homepage* `https://muscle3.readthedocs.io`.
23. *Nektar-cwipi STFC coupling* `https://github.com/ExCALIBUR-NEPTUNE/nektar-cwipi/tree/stfc-coupling-simplediffusion` (2023).
24. *Nektar++ A Spectal/HP Element Framework* `https://www.nektar/info`.
25. *OASIS Website* `https://portal.enes.org/oasis` (2021).
26. *ONERA, The French Aerospace Laboratory* `http://onera.fr`.
27. *OpenPALM Website* `https://www.cerfacs.fr/globc/PALM_WEB/index.html`.
28. Parra, F. I., Barnes, M., *et al. 1D drift kinetic models with wall boundary conditions* tech. rep. 2047357-TN-05-01 (UKAEA Project Neptune, 2021).
29. Parra, F., Barnes, M., *et al. 1D drift kinetic models with periodic boundary conditions* tech. rep. 2047357-TN-01-02 (UKAEA Project Neptune, 2021).
30. Parra, F., Barnes, M., *et al. Physics in the edge of fusion devices* tech. rep. 2047357-TN-03-01 (UKAEA Project Neptune, 2021).
31. *PLE (Parallel Location and Exchange) documentation* `https://www.code-saturne.org/documentation/ple-2.0/html/index.html`.
32. *PLE GitHub Repository* `https://github.com/code-saturne/code_saturne/tree/master/libple`.
33. *preCICE GitHub Repository* `https://github.com/precice/precice`.
34. *preCICE Website* `https://precice.org/index.html`.
35. *preCICE XML reference* `https://precice.org/configuration-xml-reference.html`.
36. Prudhomme, S., Ben Dhia, H., *et al.* Computational analysis of modeling error for the coupling of particle and continuum models by the Arlequin method. *Computer Methods in Applied Mechanics and Engineering* **197,** 3399–3409. `https://www.sciencedirect.com/science/article/pii/S0045782508001242` (2008).
37. Reiter, D., Baelmans, M., *et al.* The EIRENE and B2-EIRENE Codes. *Fusion Science and Technology* **47,** 172–186 (2005).
38. Rubin, P. *Comparison of Code Coupling Libraries for High Performance Multi-Physics Simulation* tech. rep. ADD (Science and Technology Facilities Council, 2022).
39. Schwartzentruber, T. E. & Boyd, I. D. Progress and future prospects for particle-based simulation of hypersonic flow. *Progress in Aerospace Sciences* **72,** 66–79. `https://www.sciencedirect.com/science/article/pii/S0376042114000827` (2015).
40. Sherwin, S., Kirby, M., *et al. NEKTAR++* online. Jan. 21, 2021. `https://www.nektar.info`.
41. *Software Outlook Website* `https://www.softwareoutlook.ac.uk/`.
42. Tang, Y.-H., Kudo, S., *et al.* Multiscale Universal Interface: A concurrent framework for coupling heterogeneous solvers. *Journal of Computational Physics* **297,** 13–31 (Sept. 15, 2015).
43. *The CWIPI coupling library* `https://w3.onera.fr/cwipi/bibliotheque-couplage-cwipi`.
44. *The State of the Art in Scientific Coupling Technology for HPC* Excalibur Online Workshop, 2021. `https://excalibur-coupling.github.io/workshop`.
45. Veen, L. & Hoekstra, A. Easing Multiscale Model Design and Coupling with MUSCLE 3. *Computational Science – ICCS 2020,* 425–438. `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7304760/` (2020).