PCMS: Parallel Coupler for Multimodel Simulations

Jacob Merson, Cameron Smith, Mark S. Shephard

April 2023

1 Coupler Design

1.1 Background

PCMS is a library that is designed to make in-memory tight coupling a range of large scale parallel simulation codes as easy as possible without modification of their internal data-structures or algorithms and without making compromises with respect to efficiency or accuracy. It is focused on multimodel simulations, which encompasses both partitioned multiphysics analysis as well as concurrent and hierarchical multiscale analysis.

To support the broadest set of codes, we have developed the following set of design requirements

- Does not modify existing datastructures or computational algorithms in the simulation codes.
- Makes efficient use of the largest supercomputers available (today, exascale).
- Efficiently handles data and coordinate transformations.
- Performs efficient operations with the complete range of structured and unstructured meshes.
- Handles parallel coordination and communication of distributed field data.

The combination of three components sets PCMS apart from the other coupling efforts: 1) Integration of coordinate and data transformations. 2) Parallel control and transfer algorithms for fields on structured and unstructured meshes. 3) Focus on ease of use and limiting the need for intrusive code changes. 4) Use of an intermediate data representation when direct coupling is not possible to avoid quadratic growth of coupling implementations.

One of the motivating use cases for PCMS is to support Tokamak coupling. Many of the codes associated with Tokamaks use specialized physics based coordinate systems that follow the magnetic field lines. These coordinate systems are not unique, so codes take advantage of those that make their computational methods more efficient. From a design perspective, this implies that we need to be able to handle an open set of coordinate systems.

1.2 Review of Existing Coupling Codes

The well versed reader will recognize that individually some of these components are available in existing software. However, it is the combination of the four components and the strict adherence to our design requirements that sets PCMS apart. The following section will provide a brief description of other existing coupling tools.

- Moose: intrusive/monolithic solver [12]
- preCICE: easy to user interface, uses RBF for field transfer, no coordinate system support, no mesh based field support. Currently unsure about scalability...need to search through papers. [3, 4, 5].
- DTK: implemented rendevouz algorithm, no support for coordinate systems, focusses solely on field transfer part of coupling [24], no scalability results for current supercomputers (with GPUs) found, although they do seem to have concept of memory space in current API. developed as part of CASL. Requires Trilinos
- MPCCI: commercial code requires license, no indication that it can work with supercomputers as it is focused on coupling commercial codes [15, 17]
- Integrated Plasma Simulator (IPS)/ One Modeling Framework for Integrated Tasks (OMFIT) File based, freeform treelike structure, coupling through application native ”datastructures” (file-io), contend that only a few pairs of

well defined codes will ever need to be coupled, so no need to worry about the n^2 problem of letting n independent formats couple [11, 18]

- Integrated Modeling and Analysis System (IMAS), standard fileformat/layout developed by ITER for experimental and simulation data. Not clear that this can/should be used for tight coupling [22].
- European Transport Simulator (ETS) [16, 6]

1.3 Design

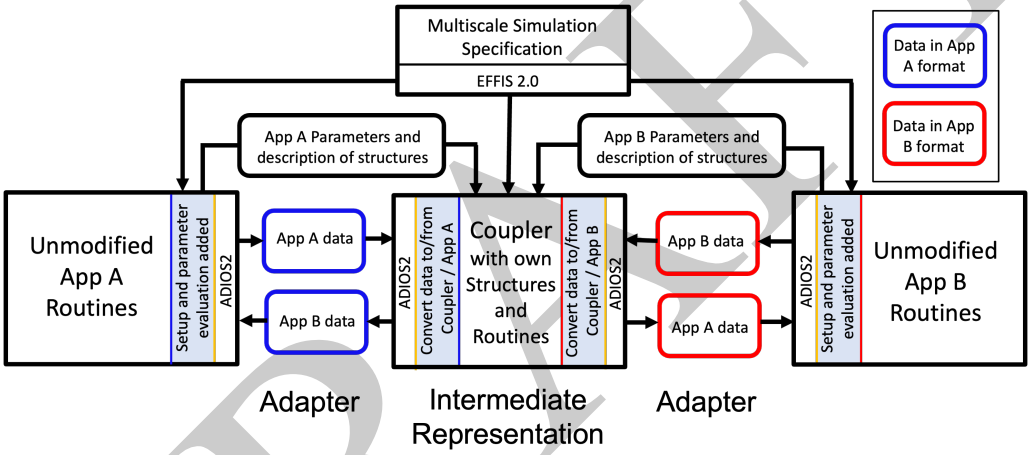

Figure 1: Overall structure of PCMS

Most multiscale and coupling libraries seek to make developing a new coupling as flexible as possible by promoting coupling setup through input files [5, 27]. We contend that developing a new coupling is done infrequently and by domain experts and is better reified in the form of source code that be partially verified through compile time checking. That said, we also intend to provide runtime configuration capabilities that can be used to quickly iterate during the development phase of new coupling pairs. To this end, we are collaborating with Manish Parashar and Philip Davis on the development of an interface to Benesh, a domain specific language and configuration layer for dynamic multimodel coupling [7].

Currently, PCMS only natively supports compile time coupling definitions that can provide optimizations and safety benefits over the runtime generic interfaces. Although these are compile time interfaces, it is important to understand that the

Language	Client	Server
C++	✓	✓
C	✓	
Fortran	✓	

Table 1: PCMS support for various languages in server and client mode.

coupler API design allows for an open set of field types and coordinate systems. This is enabled through a mix of compile time polymorphism and type erasure. However, the set of underlying data types (e.g., int, double, bool) are limited to datatypes that are supported in MPI and Adios2 which are used for the underlying data transport. The user can always communicate an arbitrary type by casting the data to `char`, however in this case, the user is responsible for maintaining type safety in their serialization and deserialization routines.

In the future, we expect to support two modes of runtime coupling definitions. The first makes use of a preprogrammed set of field layouts that may be able to take advantage of certain optimizations. The second, uses a schema to define all of the necessary data for coupling at runtime. Although the use of a schema provides the greatest flexibility, it gives up some level of optimization and compile time safety.

Figure 1 shows the structure of a two-way coupling of App A and App B through a independent coupling server making use of PCMS. The coupler has two modes of operation: client and server mode. The client mode has a simplified interface that makes it amenable to inclusion in APIs of a variety of other programming languages. The Server interface includes operations that make use of compile time polymorphism that is hard to capture in other languages without template support. I have plans to develop a `C`/`Fortran` server interface, however it will have limitations on the types of fields/data and will either have to make use of a coupling schema (for generic field inputs), or selection of a predeveloped list of field types and will limit certain optimizations to be selected at compile time. Table 1 shows the current status of language support in PCMS.

The use of an independent coupling server in figure 1 is useful in cases where both applications are unable to be linked to the libraries that support the field intermediate representation, namely "Omega" for unstructured meshes. Although we have developed test cases that eliminate the use of an independent coupling server, we have only performed full scale testing in a real application (XGC total-f/delta-f axisymmetric coupling) that makes use of the independent coupling server.

PCMS supports fields and field transformations that are stored in host or device memory. However, currently Redev/Adios2 only supports host memory, so serialized fields are copied back to host memory before they are sent.

Each PCMS server and client can create a number of applications which each represent a communication pair. Each application contains a number of fields that represent the data that must coupled. In the case where an independent coupling server is used, each client will only have a single application that it communicates with, specifically the coupling server.

PCMS sends and receives fields in phases that are independent for each application. The fields that are sent and received in each phase must be identical in the server and client. Each application may only have either a send or receive phase open at a time, however this can be overlapped with phases from other applications. Depending on the `RDV` backend that is selected, no data is guaranteed to have been send/received until the phase has been completed.

The only operations that are currently available on the coupling client are to send and receive the associated fields. The coupling server additionally has gather and scatter operations. The gather operation receives fields from participating applications, performs a field transfer to the coupler's internal field representation, and combines the data from each of the participating fields into a single combined field with a user provided callback. The scatter operation splits the combined field onto the appropriate portion of the overlap domain, performs a field transfer from the internal field representation to the native field datastructures, and sends the field to the appropriate applications.

In the case of simple two-way coupling the use of gather and scatter on the server mirror the send and receive operations on the client. For more complex coupling paradigms with asymmetric send and receive phases, I have found that independently calling the field transfer routines was more convenient. This indicates that the gather and scatter API may need some degree of improvement.

1.3.1 TODO

Here is a nonexhaustive list of enhancements that will be needed to support a wide variety of "generalized" coupling schemes.

- PCMS has a limited metadata exchange round. This needs to be to extended account for various components that are currently hardcoded such as analysis status (running/done), time stepping information, overlap domain definition, and for "schema analysis" many additional components such as shape function definition, coordinate system, etc.

- Control of time-stepping for implicit and explicit coupling approaches.
- Verification of coordinate system transformation implementation/transformation routines.
- Support for generalized schema field type.
- Refactor a ”Field Layout” representation from fields to reduce the amount of field metadata exchange in initialization phase (applications tend to have many fields that share the same
- ”input file” coupling
- Point to point communication protocol rather than coupling through centralized coupling server (if multiple applications are capable of performing coupling, should determine this through metadata exchange and have a protocol).
- GPU support for Redev

2 Parallel Control

The rendezvous algorithm coordinates the information passing between pairs of coupled parallel simulations “when processors neither know which other processors to send data to, nor which other processors will be sending data to them” [21]. It is used in LAMMPS [21], the PUMI unstructured mesh library for loading multi-billion element meshes from file [23] and in the Data Transfer Toolkit [24].

Figure 2 depicts two unstructured meshes of the same domain where information from one mesh is needed by the other but they have a different partitioning across the same set of processes. In Figure 4 the colored ovals depict the domain owned by a given process in the two meshes. Since the domains are not intersecting the Rendezvous method is used to efficiently exchange data between the processes that do have the intersecting portion of the domain. A common domain partition, depicted by the red structured grid in the bottom two figures, allows processes that own data within a given cell in grid to ’rendezvous’ with the other processes that own data in that cell and perform an exchange.

The key to the Rendezvous algorithm is a partitioning of the common portion of the domain that (1) has a relatively low memory usage and (2) supports computationally efficient queries for membership within the partition given a entity

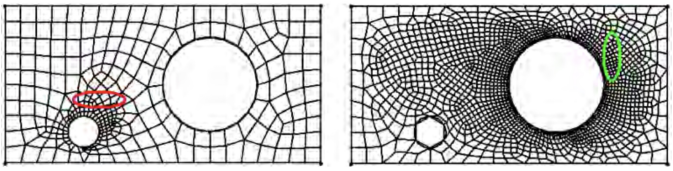

Fig. 4. Thermal (left) and stress (right) grids. The colored ovals are clumps of grid cells the same processor owns in the two grids.

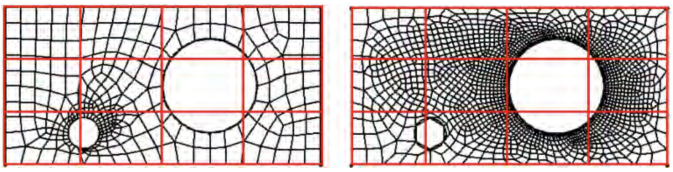

Fig. 5. Rendezvous decomposition overlaid on thermal and stress grids. [1]

Figure 2: [21]

within the domain (e.g., a mesh element or vertex). The structured grid from the above figure is one possible partition that satisfies these requirements.

The `RDV` library implements the rendezvous algorithm using `ADIOS2` data transport engines and thus enables coupling simulations that run in-memory or via files. Use of an alternative `DataSpaces` [8] backend has been demonstrated as part of the Benesh integration by Philip Davis. However, we anticipate that any demonstrated performance improvements will be contributed to `ADIOS2` and it will remain the primary communication layer as it provides the most flexibility.

3 Field Transfer

The field transfer operations in PCMS are unique in that they are designed as algorithms that can operate generically on fields with any underlying data structures and coordinate systems. Another key aspect is that they make use of a field's native methods for query and interpolation.

To use these generic field transfer algorithms, a minimal set of customization points (free functions) must be implemented for each of the input field types. These customization points don't require any modifications to the underlying field structures, but allow for a uniform query methodology that can be used with generic algorithms.

Various field transfer methodologies exist and the choice is largely driven by cost, accuracy, and physics constraints such as the need for conservation of the

underlying fields. Broadly speaking these methodologies either use a direct evaluation of the source field at the target fields's node points (interpolation/extrapolation), or solve a secondary problem to minimize some residual measure of the error in the target field (projection). Projection based methods tend to be more expensive, but make it easier to bring in physics constraints. Some authors have managed to develop conservative interpolation methods [1].

The field evaluation methods can further be categorized into mesh based methods, or pointwise methods. Mesh based methods use the underlying structure of the mesh whereas the pointwise methods consider the nodal points of the source data as a point cloud and impose shape functions that are independent of the discretization that is used in the solution procedures. The most common pointwise evaluation method is the use of radial basis functions (RBF) [26]. This approach is convenient in the context of coupling as it doesn't require the coupler to handle the complexities of the mesh structure. This approach has found wide success in other coupling libraries and is used to get conservative particle to cell transfers in PIC codes [4, 20]. Slattery, provides a comparison of radial basis functions and mesh intersection methods and concludes that the accuracy and conservative properties of RBF and mesh based methods are dependent on the geometry and field structure [26].

Most mesh based conservative field transfer methods are based on the approach presented in [14]. In this method the fields are conservatively transferred the polyhedra that represent the intersection of the source and target meshes [2, 13, 19, 9, 10].

Currently PCMS has an implementation for the direct evaluation (interpolation) methods that makes use of the source field's native evaluation procedures including (mesh based) nearest neighbor and Lagrange interpolation. We anticipate supporting conservative mesh based projection methods (mesh intersection) as well as point wise interpolation and conservative projection methods that make use of radial basis functions.

Note: the various field transfer methodologies work better/worse depending on the fields data and underlying geometric representation. In some cases the RBF does better w.r.t. field replication and conservation than the mesh based methods. Despite the significant increase in complexity that is added by including mesh based field transfer methods, they must be included if one is unwilling to compromise on accuracy and conservation properties in general (i.e., it is critical that the users have access to both methods and are able to test what works best for their problem of interest).

4 Other Items to Address

- A overview of PUMIPic including design, how to use it, consideration of alternative mesh/particle relationships and ability to alter the particle data structure, etc. (Cameron).
  - An introductory paragraph is here: `https://github.com/SCOREC/pumipic-docs/blob/d253fc0859c2e8d9d460c5b4c53b335846efb03e/pumipic_2020/introduction.tex#L81-L93`
  - unstructured mesh data structure summary: `https://github.com/SCOREC/pumipic-docs/blob/d253fc0859c2e8d9d460c5b4c53b335846efb03e/pumipic_2020/data-structures.tex#L21`
  - partitioning an unstructured mesh for PIC: `https://github.com/SCOREC/pumipic-docs/blob/d253fc0859c2e8d9d460c5b4c53b335846efb03e/pumipic_2020/data-structures.tex#L39`
  - particle data structure summary: `https://github.com/SCOREC/pumipic-docs/blob/d253fc0859c2e8d9d460c5b4c53b335846efb03e/pumipic_2020/data-structures.tex#L210`
  - alternative mesh/particle relationships: The PUMIPic API provides a consistent set of mesh-based particle operations across multiple implementations of the particle data structure. Of the implementations, Sell-C-σ [2020 PUMIPIC PAPER] and CabanaM (a implementation of the pumipic particle structure based on the ECP COPA Cabana AoSoA [25]) have been demonstrated to provide high performance for PIC simulations with hundreds, or more, of particles per element [XGCM PAPER]. Methods that utilize many fewer particles per element, such as the material point method or PIC for impurity tracking, may want to avoid the need to explicitly store particles on a per element basis while still supporting the ability to operate on particles at the element level. For this use case a new structure, DPS, is under development. Relative to Sell-C-σ and CabanaM, it is designed to have a very fast update of the particle-to-element association at the expense of non-coalesced memory access when looping over elements and the particles within them. The expectation is that this trade-off will net in increased overall performance for these applications. Critically, these performance studies can be performed without disrupting application

code as all of the aforementioned particle data structures use the same PUMIPic APIs.

- Supporting unstructured meshes and fields on CPU/GPUs - functionalities, tools, and adapativity (Cameron)
  - PUMI (`https://www.scorec.rpi.edu/reports/view_report.php?id=666`) major features:
    - Scaled to millions of processes on meshes with billions of elements
    - Support for parallel adaptation of straight sided, curved and semi-structured boundary layer like meshes.
    - Parallel field interface for defining physical fields, their distribution across the mesh, and basis/shape functions for supporting field transfer during mesh adaptation and estimating errors to drive mesh adaptation.
    - CPU only
  - Omega_h (`https://www.scorec.rpi.edu/REPORTS/2016-25.pdf`) major features:
    - On a single Summit V100 Omega_h GPU adaptation to an isotropic size field generated by the PUMI-MFEM RF analysis from 260k to 1.4M tets completes in 4.06 seconds. On 16 processes (of the 24 available) on an AMD EPYC 7451 (1st generation) CPU PUMI completes adaptation in 20.6 seconds. Figure 3 depicts the adapted mesh.
    - Parallel mesh adaptation on GPU and CPU - ran on AMD, Intel and Nvidia GPUs
    - GPU aware parallel field interface for defining physical fields and their distribution is being developed. The interface follows the concepts defined by PUMI but using array-based APIs for efficient execution on GPUs. The interface is implemented with Kokkos or ECP COPA-Cabana. Users who need runtime sizing of the data associated with each mesh entity (e.g., the number of dofs per vertex for a given field) would select the Kokkos implementation (The Cabana AoSoA only supports compile-time sizing) at the possible expense of some performance.
    - Supports simplicies (triangles and tets) only.

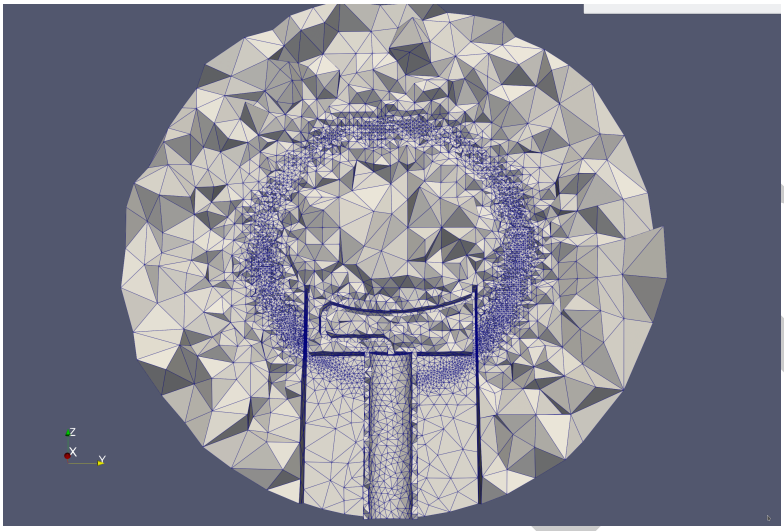

Figure 3: Omega_h adapted mesh on the isotropic size field from a PUMI-MFEM RF analysis.

References

[1] Frédéric Alauzet. "A Parallel Matrix-Free Conservative Solution Interpolation on Unstructured Tetrahedral Meshes". In: *Computer Methods in Applied Mechanics and Engineering* 299 (Feb. 2016), pp. 116–142. DOI: 10.1016/j.cma.2015.10.012.

[2] Ghislain Blanchard and Raphaël Loubère. "High Order Accurate Conservative Remapping Scheme on Polygonal Meshes Using a Posteriori MOOD Limiting". In: *Computers & Fluids* 136 (Sept. 2016), pp. 83–103. DOI: 10.1016/j.compfluid.2016.06.002.

[3] Hans-Joachim Bungartz et al. "A Plug-and-Play Coupling Approach for Parallel Multi-Field Simulations". In: *Computational Mechanics* 55 (2015), pp. 1119–1129. DOI: 10.1007/s00466-014-1113-2.

[4] Hans-Joachim Bungartz et al. "preCICE – A Fully Parallel Library for Multi-Physics Surface Coupling". In: *Computers & Fluids* 141 (Dec. 2016), pp. 250–258. DOI: 10.1016/j.compfluid.2016.04.003.

[5] Gerasimos Chourdakis et al. "preCICE v2: A Sustainable and User-Friendly Coupling Library". In: *Open Research Europe* 2 (Apr. 29, 2022), p. 51. DOI: 10.12688/openreseurope.14445.1.

[6] D.P. Coster et al. “Building a Turbulence-Transport Workflow Incorporating Uncertainty Quantification for Predicting Core Profiles in a Tokamak Plasma”. In: *Nuclear Fusion* 61.12 (Dec. 1, 2021), p. 126068. DOI: 10.1088/1741-4326/ac359f.

[7] Philip E. Davis et al. “Benesh: A Programming Model for Coupled Scientific Workflows”. In: *2020 IEEE/ACM Fifth International Workshop on Extreme Scale Programming Models and Middleware (ESPM2)*. 2020 IEEE/ACM Fifth International Workshop on Extreme Scale Programming Models and Middleware (ESPM2). GA, USA: IEEE, Nov. 2020, pp. 1–9. DOI: 10.1109/ESPM251964.2020.00008.

[8] Ciprian Docan, Manish Parashar, and Scott Klasky. “DataSpaces: An Interaction and Coordination Framework for Coupled Simulation Workflows”. In: *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*. HPDC ’10: The 19th International Symposium on High Performance Distributed Computing. Chicago Illinois: ACM, June 21, 2010, pp. 25–36. DOI: 10.1145/1851476.1851481.

[9] P.E. Farrell and J.R. Maddison. “Conservative Interpolation between Volume Meshes by Local Galerkin Projection”. In: *Computer Methods in Applied Mechanics and Engineering* 200.1-4 (Jan. 2011), pp. 89–100. DOI: 10.1016/j.cma.2010.07.015.

[10] P.E. Farrell et al. “Conservative Interpolation between Unstructured Meshes via Supermesh Construction”. In: *Computer Methods in Applied Mechanics and Engineering* 198.33-36 (July 2009), pp. 2632–2642. DOI: 10.1016/j.cma.2009.03.004.

[11] Samantha S Foley, Wael R Elwasif, and David E Bernholdt. *The Integrated Plasma Simulator: A Flexible Python Framework for Coupled Multiphysics Simulation*. ORNL/TM-2012/57, 1034707. Nov. 1, 2011, ORNL/TM–2012/57, 1034707. DOI: 10.2172/1034707.

[12] Derek Gaston et al. “MOOSE: A Parallel Computational Framework for Coupled Systems of Nonlinear Equations”. In: *Nuclear Engineering and Design* 239.10 (2009), pp. 1768–1778.

[13] R.K. Jaiman et al. “Conservative Load Transfer along Curved Fluid–Solid Interface with Non-Matching Meshes”. In: *Journal of Computational Physics* 218.1 (Oct. 2006), pp. 372–397. DOI: 10.1016/j.jcp.2006.02.016.

[14] Xiangmin Jiao and Michael T. Heath. “Common-Refinement-Based Data Transfer between Non-Matching Meshes in Multiphysics Simulations”. In: *International Journal for Numerical Methods in Engineering* 61.14 (Dec. 14, 2004), pp. 2402–2427. DOI: 10.1002/nme.1147.

[15] W. Joppich and M. Kürschner. “MpCCI—a Tool for the Simulation of Coupled Applications”. In: *Concurrency and Computation: Practice and Experience* 18.2 (2006), pp. 183–192. DOI: 10.1002/cpe.913.

[16] D. Kalupin et al. “Numerical Analysis of JET Discharges with the European Transport Simulator”. In: *Nuclear Fusion* 53.12 (Dec. 1, 2013), p. 123007. DOI: 10.1088/0029-5515/53/12/123007.

[17] Miriam Mehl et al. “Parallel Coupling Numerics for Partitioned Fluid–Structure Interaction Simulations”. In: *Computers & Mathematics with Applications* 71.4 (Feb. 2016), pp. 869–891. DOI: 10.1016/j.camwa.2015.12.025.

[18] O. Meneghini et al. “Integrated Modeling Applications for Tokamak Experiments with OMFIT”. In: *Nuclear Fusion* 55.8 (Aug. 1, 2015), p. 083008. DOI: 10.1088/0029-5515/55/8/083008.

[19] Sandeep Menon and David P. Schmidt. “Conservative Interpolation on Unstructured Polyhedral Meshes: An Extension of the Supermesh Approach to Cell-Centered Finite-Volume Variables”. In: *Computer Methods in Applied Mechanics and Engineering* 200.41-44 (Oct. 2011), pp. 2797–2804. DOI: 10.1016/j.cma.2011.04.025.

[20] Albert Mollén et al. “Implementation of Higher-Order Velocity Mapping between Marker Particles and Grid in the Particle-in-Cell Code XGC”. In: *Journal of Plasma Physics* 87.2 (Apr. 2021), p. 905870229. DOI: 10.1017/S0022377821000441.

[21] Steven J. Plimpton and Christopher Knight. “Rendezvous Algorithms for Large-Scale Modeling and Simulation”. In: *Journal of Parallel and Distributed Computing* 147 (2021), pp. 184–195. DOI: 10.1016/j.jpdc.2020.09.001.

[22] M Romanelli et al. “Code Integration, Data Verification, and Models Validation Using the ITER Integrated Modeling and Analysis System (IMAS) in EUROfusion”. In: *Fusion Science and Technology* 76.8 (2020), pp. 894–900.

[23] Christian Pelties Sebastian Rettenberger C.W. Smith. “Optimizing CAD and Mesh Generation Workflow for SeisSol”. In: SC ’14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. New Orleans, LA: American Nuclear Society, Nov. 16, 2014.

[24] S R Slattery, P P H Wilson, and R P Pawlowski. “The Data Transfer Kit: A Geometric Rendezvous-Based Tool for Multiphysics Data Transfer”. In: International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering (M&C 2013). Sun Valley, ID: American Nuclear Society, May 5–9, 2013, p. 11.

[25] Stuart Slattery et al. “Cabana: A Performance Portable Library for Particle-Based Simulations”. In: *Journal of Open Source Software* 7.72 (2022), p. 4115. DOI: 10.21105/joss.04115.

[26] Stuart R. Slattery. “Mesh-Free Data Transfer Algorithms for Partitioned Multiphysics Problems: Conservation, Accuracy, and Parallelism”. In: *Journal of Computational Physics* 307 (Feb. 2016), pp. 164–188. DOI: 10.1016/j.jcp.2015.11.055.

[27] William Reading Tobin. “The Adaptive Multiscale Simulation Infrastructure”. Dept. Comput. Sci. PhD thesis. Troy, NY USA: Rensselaer Polytechn. Inst., 2018.