The Journal of Open Source Software

ASTE: An artificial solver testing environment for partitioned coupling with preCICE

David Schneider [1¶], **Muhammed Kürşat Yurt** [2], **Frédéric Simonis** [1], and **Benjamin Uekermann** [1]

1 Institute for Parallel and Distributed Systems, University of Stuttgart, Germany 2 TUM School of Engineering and Design, Technical University of Munich, Germany ¶ Corresponding author

DOI: 10.21105/joss.07127

Software

- Review
- Repository
- Archive

Editor: Patrick Diehl

Reviewers:

- @asalmgren
- @hverhelst

Submitted: 01 August 2024
Published: 14 November 2024

Summary

Simulating multi-physics phenomena for real-world applications states various challenges in scientific computing. Each individual physical domain has behavior that is often described through a distinct set of partial-differential equations that needs to be solved in that domain. Their interaction is then achieved through bidirectional exchange of suitable coupling data between all involved domains. Partitioned coupling tackles multi-physics simulations by glueing together separate models, typically implemented in separate software environments. To facilitate such partitioned multi-physics simulations effectively, so-called coupling libraries offer commonly required functionality. We focus in particular on coupling through the open-source library preCICE (Chourdakis et al., 2022), which offers functionality for data communication, data mapping, coupling schemes, and more. In the most basic setup, at least two executables call preCICE to perform a coupled simulation. As additional software components, so-called adapters bridge the gap between the preCICE API and the software environments used by the coupled models. Creating and using this overall setup for early development purposes is not only cumbersome, but also very inefficient. The artificial solver testing environment (ASTE) allows for replacing models coupled via preCICE with artificial ones, potentially in parallel distributed across multiple ranks on distributed memory. This helps in the development of preCICE, adapters, or simulation setups by reducing the necessary software components, simplifying execution workflows, and reducing runtime of the case. In addition, ASTE provides performance and accuracy metrics of the configured simulation setup.

Statement of need

Figure 1 illustrates the software stack required for a coupled simulation setup using FEniCS and OpenFOAM as examples, and compares it to a simulation setup using ASTE. Besides preCICE itself, core ingredients for practical applications are preCICE API language bindings, preCICE adapters, the simulation frameworks, and their dependencies. ASTE, on the other hand, replaces coupled models and only requires a reduced set of dependencies. It abstracts the computational complexity of the models away by extracting the relevant information from VTK files instead and passing extracted data to preCICE, potentially in parallel on distributed memory. While the VTK files may stem from actual simulations, ASTE can also generate artificial VTK files with prescribed coupling data. On top of that, the entire tool chain of ASTE enables easily altering the simulation setup through different mesh partitionings, and specifically for the configuration of data mappings in preCICE, ASTE can evaluate additional accuracy metrics of used mappings.

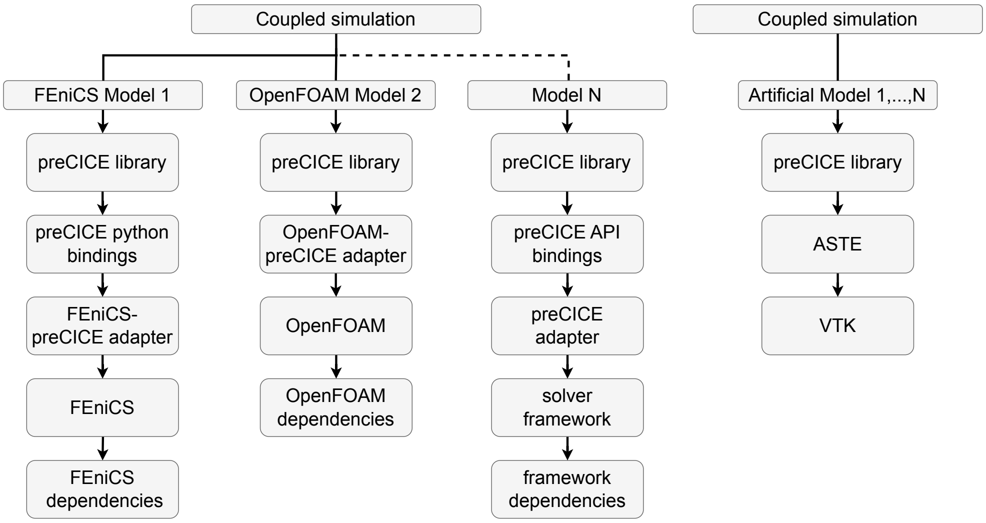

Figure 1: Dependency graph between models, applications, and libraries for a coupled simulation with FEniCS and OpenFOAM compared to a dependency graph using ASTE.

From an application standpoint, ASTE provides a reproducible environment which enables sharing and rerunning of scenarios, regardless of the availability of involved software components. This capability is particularly useful for debugging issues reported by users of preCICE, who can share their scenarios (e.g., through the preCICE forum) for developers to analyze, even when the involved software is unavailable due to licensing terms or being closed-source.

A further crucial argument for emulating models with ASTE is computational efficiency. For coupled simulations, the main computational load is typically carried by the models instead of the coupling library. Hence, running the original models repeatedly for development purposes of preCICE or adapter components is both time-consuming and inefficient. This inefficiency not only complicates software development, but also applies to parameter tuning for real-world applications, where the execution of involved models might become prohibitively expensive already due to the problem size.

In this regard, ASTE provides a lightweight and valuable tool. It enables the efficient development of preCICE by testing new features on real-world applications in an artificial solver-like setup, e.g., for developing new communication algorithms (Lindner, 2019; Totounferoush et al., 2021) or to develop new mapping methods, e.g., (Ariguib, 2022; Chourdakis et al., 2022; Martin, 2022; Schneider et al., 2023). In fact, testing and developing preCICE was the use case behind the first prototype of ASTE, which was developed as part of Lindner (2019). Beyond the development in preCICE, ASTE also fosters the development of new adapter codes to be coupled via preCICE, as it aids in debugging and enhances the transparency of data flow. Moreover, combining preCICE's performance instrumentation with the ASTE's flexibility and insight, it enables finding appropriate settings for specific scenarios, as effectively demonstrated in the large-scale example in Lindner et al. (2020).

Although coupling libraries like MUI offer their own testing and benchmarking infrastructure, e.g., MUI's testing framework, many tools do not provide such testing environments at all. Instead, evaluating, testing and benchmarking of these libraries relies on hard-coded solutions tailored to individual test setups, e.g., the benchmarking performed by Valcke et al. (2022). ASTE covers a comprehensive, flexible and resuable toolchain for development, testing, and parameter tuning.

Functionality & Use

The central interface of ASTE is given through a VTK mesh file, which contains information about the geometric shape of the model we emulate. The VTK files can be generated from mesh generation tools (e.g., GMSH (Geuzaine & Remacle, 2009)), included Python scripts, other simulation software, or directly reused from a completed preCICE simulation. Given a VTK file, ASTE offers different algorithms to repartition them (e.g., through METIS (Karypis & Kumar, 2009)) for parallel runs. Moreover, ASTE can generate artificial data using pre- or user-defined functions on the mesh and store them in the VTK file format. The core module of ASTE then reads the VTK file and passes the data to preCICE, potentially in every time step of the coupled simulation. Once the simulation is finished, the generated data is stored in another VTK file and can be compared against the original artificial data. Performance metrics are accessible through the preCICE performance framework.

While the core module of ASTE is written in C++, the pre- and postprocessing scripts are implemented in Python. The core module relies on VTK (Schroeder et al., 2006), Boost, and MPI for parallel execution. It provides a command line interface for simple simulations and can be configured in JSON (Lohmann, 2023) for more complex scenarios.

ASTE is hosted on GitHub and releases are published using GitHub releases. The documentation is part of the ASTE repository and rendered on the preCICE website. In addition, a tutorial and ready-to-use examples are available. Building is handled via CMake and, as part of the preCICE distribution (Chen et al., 2024), ASTE can be used through a Vagrant box.

Acknowledgements

The authors are funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under project numbers 515015468 and 528693298, and under Germany's Excellence Strategy - EXC 2075 – 390740016. We acknowledge the support of the Stuttgart Center for Simulation Science (SimTech).

References

Ariguib, B. (2022). *Second-order projection-based mapping methods for coupled multi-physics simulations* [Bachelor's thesis]. University of Stuttgart. https://doi.org/10.18419/opus-12128

Chen, J., Chourdakis, G., Desai, I., Homs-Pons, C., Rodenberg, B., Schneider, D., Simonis, F., Uekermann, B., Davis, K., Jaust, A., Kelm, M., Kotarsky, N., Kschidock, H., Mishra, D., Mühlhäußer, M., Schrader, T. P., Schulte, M., Seitz, V., Signorelli, J., … Zonta, E. (2024). *preCICE distribution version v2404.0* (Version V1) [Data set]. DaRUS. https://doi.org/10.18419/darus-4167

Chourdakis, G., Davis, K., Rodenberg, B., Schulte, M., Simonis, F., Uekermann, B., Abrams, G., Bungartz, H., Cheung Yau, L., Desai, I., Eder, K., Hertrich, R., Lindner, F., Rusch, A., Sashko, D., Schneider, D., Totounferoush, A., Volland, D., Vollmer, P., & Koseomur, O. (2022). PreCICE v2: A sustainable and user-friendly coupling library [version 2; peer review: 2 approved]. *Open Research Europe*, *2*(51). https://doi.org/10.12688/openreseurope.14445.2

Geuzaine, C., & Remacle, J.-F. (2009). Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities. *International Journal for Numerical Methods in Engineering*, *79*(11), 1309–1331. https://doi.org/10.1002/nme.2579

Karypis, G., & Kumar, V. (2009). *MeTis: Unstructured graph partitioning and sparse matrix ordering system, version 4.0*. University of Minnesota, Minneapolis, MN; http:

//www.cs.umn.edu/~metis.

Lindner, F. (2019). *Data transfer in partitioned multi-physics simulations: Interpolation & communication* [Dissertation, University of Stuttgart]. https://doi.org/10.18419/opus-10581

Lindner, F., Totounferoush, A., Mehl, M., Uekermann, B., Pour, N. E., Krupp, V., Roller, S., Reimann, T., C. Sternel, D., Egawa, R., Takizawa, H., & Simonis, F. (2020). ExaFSA: Parallel fluid-structure-acoustic simulation. In H.-J. Bungartz, S. Reiz, B. Uekermann, P. Neumann, & W. E. Nagel (Eds.), *Software for exascale computing - SPPEXA 2016-2019* (pp. 271–300). Springer International Publishing. https://doi.org/10.1007/978-3-030-47956-5_10

Lohmann, N. (2023). JSON for modern C++. In *GitHub repository* (Version 3.11.3). https://github.com/nlohmann; GitHub. https://json.nlohmann.me

Martin, B. G. (2022). *Robust and efficient barycentric cell-interpolation for volumetric coupling with preCICE* [Master's thesis, Technical University of Munich]. https://mediatum.ub.tum.de/doc/1685618/1685618.pdf

Schneider, D., Schrader, T., & Uekermann, B. (2023). Data-parallel radial-basis-function interpolation in preCICE. *X International Conference on Computational Methods for Coupled Problems in Science and Engineering*. https://doi.org/10.23967/c.coupled.2023.016

Schroeder, W., Martin, K., & Lorensen, B. (2006). *The visualization toolkit (4th ed.)*. Kitware. ISBN: 978-1-930934-19-1

Totounferoush, A., Simonis, F., Uekermann, B., & Schulte, M. (2021). Efficient and scalable initialization of partitioned coupled simulations with preCICE. *Algorithms*, *14*(6). https://doi.org/10.3390/a14060166

Valcke, S., Piacentini, A., & Jonville, G. (2022). Benchmarking regridding libraries used in earth system modelling. *Mathematical and Computational Applications*, *27*(2). https://doi.org/10.3390/mca27020031