The Journal of Open Source Software

OpenCMP: An Open-Source Computational Multiphysics Package

Elizabeth Julia Monte[1], **Alexandru Andrei Vasile**[1], **James Lowman**[1], and **Nasser Mohieddin Abukhdeir**[1,2,¶]

1 Department of Chemical Engineering, University of Waterloo, Ontario, Canada **2** Department of Physics and Astronomy, University of Waterloo, Ontario, Canada ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Juanjo Bazán

Submitted: 30 November -001
Published: unpublished

Summary

OpenCMP is a computational multiphysics software package based on the finite element method (Ferziger & Perić, 2002). It is primarily intended for physicochemical processes in which fluid convection plays a significant role. OpenCMP uses the NGSolve finite element library (Schöberl, 2020) for spatial discretization and provides a configuration file-based interface for pre-implemented models and time discretization schemes. It also integrates with Netgen (Schöberl, 2020) and Gmsh (Geuzaine & Remacle, 2009) for geometry construction and meshing. Additionally, it provides users with built-in functionality for post-processing, error analysis, and data export for visualisation using Netgen (Schöberl, 2020) or ParaView (Ahrens et al., 2005).

OpenCMP development follows the principles of ease of use, performance, and extensibility. The configuration file-based user interface is intended to be concise, readable, and intuitive. Furthermore, the code base is structured and documented (Monte, Elizabeth J, 2021) such that experienced users with appropriate background can add their own models with minimal modifications to existing code. The finite element method enables the use of high-order polynomial interpolants for increased simulation accuracy, however, continuous finite element methods suffer from stability and accuracy (conservation) for fluid convection-dominated problems. OpenCMP addresses this by providing discontinuous Galerkin method (Cockburn et al., 2000) solvers, which are locally conservative and improve simulation stability for convection-dominated problems. Finally, OpenCMP implements the diffuse interface or diffuse domain method [Nguyen et al. (2018)], which a type of continuous immersed boundary method (Mittal & Iaccarino, 2005), which allows complex simulation domains to be meshed by non-conforming structured meshes for improved simulation stability and reduced computational complexity (under certain conditions (Monte et al., 2021)).

Statement of Need

Simulation-based analysis continues to revolutionize the engineering design process. Simulations offer a fast, inexpensive, and safe alternative to physical prototyping for preliminary design screening and optimization. Simulations can also offer more detailed insight into the physical phenomena of interest than is feasible from experimentation. Computational multiphysics is an area of particular importance since coupled physical and chemical processes are ubiquitous in science and engineering.

However, there are several barriers to more wide spread use of simulations. One such barrier is the lack of user-friendly finite element-based open-source simulation software. Many of the most widely-used computational multiphysics software packages, such as COMSOL Multiphysics (*COMSOL Multiphysics ®*, n.d.) or ANSYS Fluent (*Ansys ® Fluent*, n.d.), are closed-source and cost-prohibitive. Furthermore, the predominance of the finite volume method for fluid dynamics simulations inherently limits simulation accuracy for a given mesh compared to the finite element method (Ferziger & Perić, 2002). Many high-quality open-source packages exist which use the finite element method, such as the computational multiphysics software MOOSE (Permann et al., 2020) or the finite element libraries NGSolve (Schöberl, 2020) and FEniCS (Alnaes et al., 2015). However, they require that the user has a relatively detailed understanding of the finite element method, such as derivation of the weak formulation of the problem, along with programming background. Alternatively, open-source finite volume-based packages such OpenFOAM and SU2 are more accessible to the broader scientific and engineering community in that they require a less detailed understanding of numerical methods and programming, instead requiring an understanding of the command line interface (CLI) and configuration through a set of configuration files.

A second barrier is the complex geometries inherent to many real-world problems. Conformal meshing of these complex geometries is time and labour intensive and frequently requires user-interaction, making conformal mesh-based simulations infeasible for high-throughput design screening of many industrially-relevant processes (Yu et al., 2015). A possible solution is the use of immersed boundary methods (Mittal & Iaccarino, 2005) to allow the use of non-conforming structured meshes - simple and fast to generate - for any geometry. Use of structured meshes can also potentially improve simulation stability compared to unstructured meshes (Yu et al., 2015). The diffuse interface has been shown by Monte *et al* (Monte et al., 2021) to significantly speed-up inherently low accuracy simulations - such as those used for preliminary design screening and optimization - compared to conformal mesh-based simulations. Providing an easy-to-use publicly available implementation of this method would enable broader use by the research community and further development.

The goal of OpenCMP is to fill this evident need for an open-source computational multiphysics package which is user-friendly, based on the finite element method, and which implements the diffuse interface method. OpenCMP is built on top of the NGSolve finite element library (Schöberl, 2020) to take advantage of its extensive finite element spaces, high performance solvers, and preconditioners. OpenCMP provides pre-implemented models and a configuration file-based user interface in order to be accessible to the general simulation community, not just finite element experts. The user interface is designed to be intuitive, readable, and requires no programming experience - solely knowledge of the CLI. Users must choose the model that suites their application, but need no experience with the actual numerical implementation of said model. Finally, OpenCMP provides a publicly available implementation of the diffuse interface method with support for stabilized Dirichlet boundary conditions (Nguyen et al., 2018).

Features

The table below summarizes the current capabilities of OpenCMP. The numerical solvers for each of these models have been verified using common benchmarks (Monte, Elizabeth J, 2021). Future work on OpenCMP will focus on adding models - for multi-phase flow, turbulence, and heat transfer - and enabling running simulations in parallel over multiple nodes with MPI (Forum, 2015).

Feature	Description
Meshing	Accepts Netgen (Schöberl, 2020) or Gmsh (Geuzaine & Remacle, 2009) meshes
Numerical Methods	Standard continuous Galerkin finite element method Discontinuous Galerkin finite element method Diffuse interface method
Models	Poisson (Heat) equation

Feature	Description
	Stokes equations Incompressible Navier-Stokes (INS) equations Multicomponent Mixture INS equations
Time Schemes	First-, second-, and third- order discretizations Adaptive time-stepping
Solvers	Direct or iterative solvers Direct, Jacobi, or multigrid preconditioners Oseen (Cockburn et al., 2003) or IMEX (Ascher et al., 1995) linearization of nonlinear models
Post-Processing	Error norms calculated based on reference solutions Mesh and polynomial refinement convergence tests General simulation parameters (surface traction, divergence of velocity…) Exports results to Netgen (Schöberl, 2020) or ParaView (Ahrens et al., 2005) format
Performance	Multi-threading

Further information, including installation instructions and tutorials can be found on the OpenCMP website. The tutorials are intended to guide new users through the various features offered in OpenCMP. Notes on the mathematical foundations of the various models and code documentation are also provided.

Software testing is provided through integration tests, which confirm the accuracy of the implemented models and time discretization schemes, and unit tests which currently offer 72% line coverage. Further information regarding performance verification of OpenCMP may be found in ref. (Monte, Elizabeth J, 2021).

User Interface

Drawing inspiration from packages like OpenFOAM and SU2, the OpenCMP user interface is organized around configuration files and the CLI. Each simulation requires its own directory to hold its configuration files and outputs. This is known as the run directory or `run_dir/`. The standard layout of this directory is shown below.

```
simulation
 ├── config
 ├── bc_dir
 │     └── bc_config
 ├── ic_dir
 │     └── ic_config
 ├── model_dir
 │     └── model_config
 ├── ref_sol_dir
 │     └── ref_sol_config
 └── output
```

The Journal of Open Source Software

The main directory and each subdirectory contain a configuration file `< >_config`. These are plaintext files that specify the simulation parameters and run conditions.

The configuration file in the main directory contains general information about the simulation including which model, mesh, finite element spaces, and solver should be used. It also contains information about how the simulation should be executed such as the level of detail in the output messages and the number of threads to use.

The `bc_dir/` subdirectory contains information about the boundary conditions. Its configuration file specifies the type and value of each boundary condition. This subdirectory also contains files describing boundary condition data if a boundary condition value is to be loaded from file instead of given in closed form.

The `ic_dir` subdirectory holds information about the initial conditions. Its configuration file specifies the value of the initial condition for each model variable. Like `bc_dir/`, `ic_dir/` may contain additional files from which the initial condition data is loaded during the simulation.

The `model_dir/` subdirectory contains information about model parameters and model functions. Its configuration file specifies the values of any model parameters or functions for each model variable and the subdirectory may hold additional data files to be loaded during the simulation.

The `ref_sol_dir/` subdirectory contains information about the error analysis to be conducted on the final simulation result. Its configuration file specifies what error metrics should be computed during post-processing. This configuration file also contains the reference solutions the results should be compared against, either in closed form or as references to other files in the subdirectory that are loaded during post-processing.

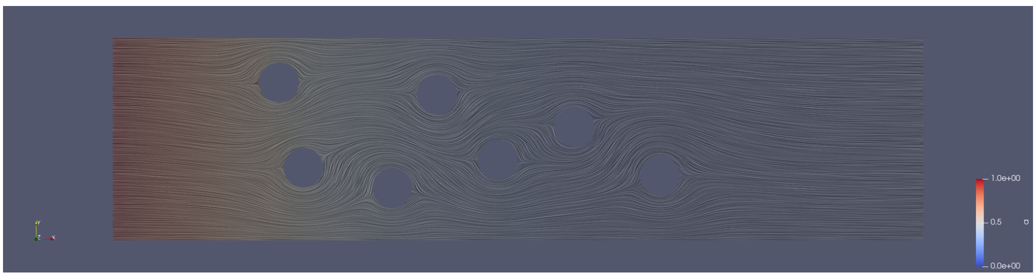
The `output/` subdirectory contains the saved simulation data. It does not need to be created before running the simulation, it will be generated automatically if results should be saved to file.

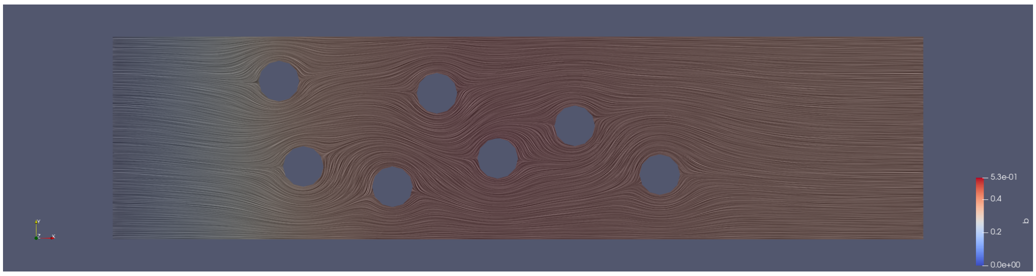
Examples of Usage

Several examples of usage of OpenCMP are present in ref. (Monte, Elizabeth J, 2021) and also available via the its website. Three relevant examples are summarized here: multi-component incompressible flow (transient 2D, Tutorial 8), using the diffuse interface method to approximate complex geometries (steady-state 3D, Tutorial 9), and incompressible flow around an immersed cylinder (steady-state 3D, Tutorial 10).

Tutorial 8 demonstrates the usage of OpenCMP to solve a multiphysics problem, two-dimensional transient multi-component incompressible flow around a set of immersed circular objects within a rectangular channel, shown below.

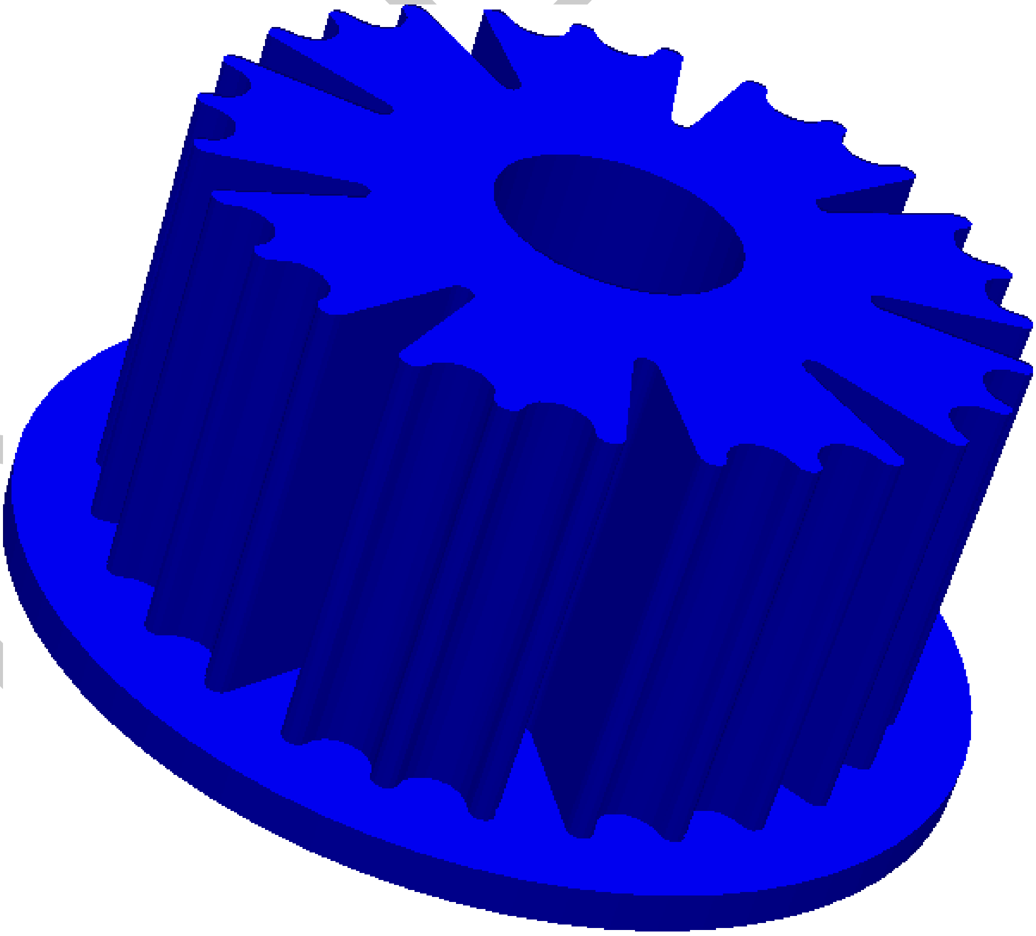
The mixture is composed of two components (A, B) where A undergoes an irreversible reaction A->B. A parabolic inlet flow of pure A is imposed, such that the mixture undergoes convection, reaction, and molecular diffusion throughout the channel. Sample simulation results of the steady-state are shown resulting from a cosine ramp of the inlet velocity an initial condition with no flow and pure A.



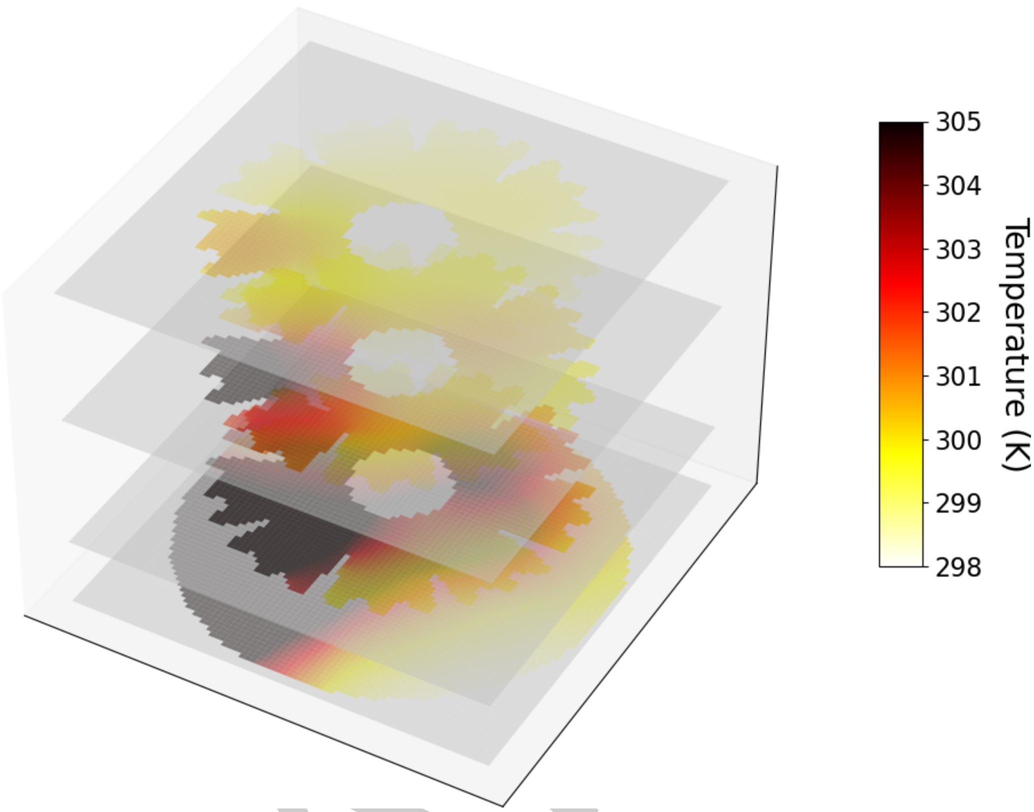


Tutorial 9 demonstrates how to use the diffuse interface method to approximate complex geometries with nonconformal structured quadrilateral/hexahedral meshes, versus the use of a traditional unstructured mesh which conforms to the complex geometry boundary. The sample problem is based on simulation of heat transfer within an LED heat sink from ref. (Monte et al., 2021), with a geometry shown below.

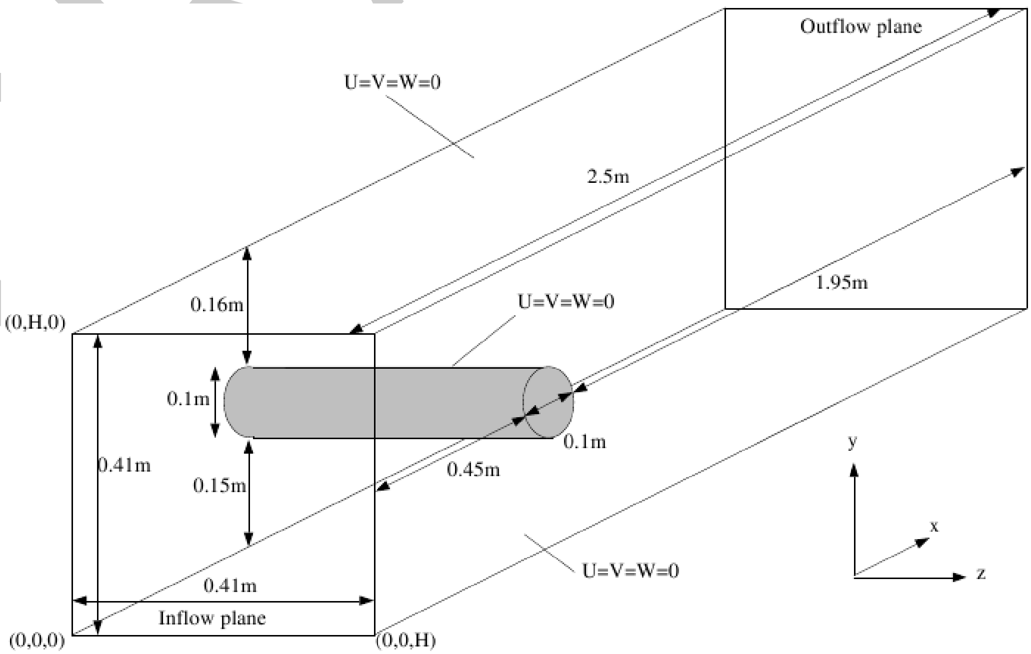
The base of the heat sink is exposed to a spatially varying heat flux profile, corresponding to heat generated by an LED assembly, with convective heat transfer conditions assumed on the exterior fins. A script is provided for ease of post-processing visualization, showing the diffuse-interface boundaries (comparable to the geometry above) and steady-state temperature profile.



The Journal of Open Source Software



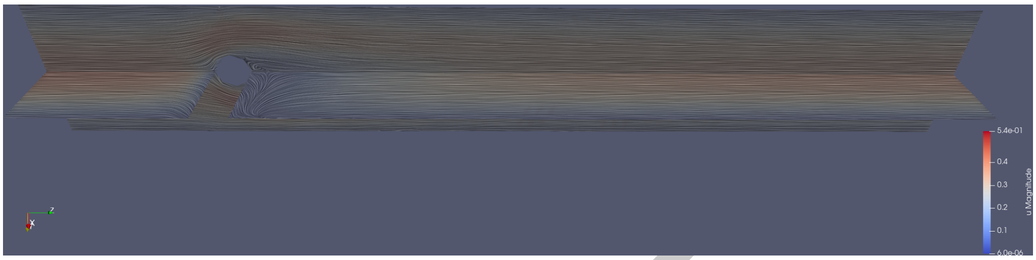
[Tutorial 10](#) demonstrate using OpenCMP to perform a standard benchmark for 3D flow around an immersed cylinder ([Bayraktar et al., 2012](#)) under laminar flow conditions. Both continuous and discontinuous Galerkin finite element solver configurations are provided, defaulting to the continuous Galerkin variation due to reduced memory constraints. The geometry is shown below with parabolic inlet velocity resulting in Re=20.



This example also uses the built-in error analysis functionality of OpenCMP to automatically compute the force vector on the immersed cylinder through integration of the surface traction

The Journal of Open Source Software

over its boundary. This facilitates the calculation of the resulting drag and lift forces on the immersed cylinder. A visualization of the three-dimensional steady-state velocity field is shown below.



Several additional examples of usage of OpenCMP in tutorial form are available via the website.

Acknowledgements

The authors would like to thank Prof. Sander Rhebergen for useful discussions regarding the discontinuous Galerkin method and Prof. Joachim Schöberl for useful discussions regarding IMEX time integration, preconditioning, and usage of the NGSolve finite element library. This research was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada and Compute Canada.

References

Ahrens, J., Geveci, B., & Law, C. (2005). *ParaView: An end-user tool for large data visualization* (Technical Report LA-UR-03-1560). Los Alamos National Laboratory. https://doi.org/10.1016/b978-012387582-2/50038-1

Alnaes, M., Blechta, J., Hake, J., Johansson, A., Kehlet, B., Logg, A., Richardson, C., Ring, J., Rognes, M. E., & Wells, G. N. (2015). The FEniCS Project Version 1.5. *Archive of Numerical Software*, *3*. https://doi.org/10.11588/ans.2015.100.20553

Ansys ® Fluent. (n.d.). ANSYS Inc. Online. https://www.ansys.com/products/fluids/ansys-fluent

Ascher, U. M., Ruuth, S. J., & Wetton, B. T. R. (1995). Implicit-explicit methods for time-dependent partial differential equations. *SIAM Journal on Numerical Analysis*, *32*(3), 797–823. https://doi.org/10.1137/0732037

Bayraktar, E., Mierka, O., & Turek, S. (2012). Benchmark computations of 3D laminar flow around a cylinder with CFX, OpenFOAM and FeatFlow. *International Journal of Computational Science and Engineering*, *7*(3), 253–266. https://doi.org/10.1504/ijcse.2012.048245

Cockburn, B., Kanschat, G., & Schötzau, D. (2003). The local discontinuous Galerkin method for the Oseen equations. *Mathematics of Computation*, *73*(246), 569–593. https://doi.org/10.1090/s0025-5718-03-01552-7

Cockburn, B., Karniadakis, G. E., & Shu, C.-W. (2000). *Discontinuous Galerkin methods: Theory, computation and applications* (1st ed., pp. 3–50). Springer-Verlag. ISBN: 3-540-66787-3

COMSOL Multiphysics ®. (n.d.). COMSOL AB; Online. https://www.comsol.com/

Ferziger, J. H., & Perić, M. (2002). *Computational methods for fluid dynamics* (3rd ed.). Springer-Verlag. ISBN: 3-540-42074-6

Forum, M. P. I. (2015). *MPI: A message-passing interface standard version 3.1*. High-Performance Computing Center.

Geuzaine, C., & Remacle, J.-F. (2009). Gmsh: A three-dimensional 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

Mittal, R., & Iaccarino, G. (2005). Immersed boundary methods. *Annual Review of Fluid Mechanics*, *37*, 239–261. https://doi.org/10.1146/annurev.fluid.37.061903.175743

Monte, E. J., Lowman, J., & Abukhdeir, N. M. (2021). A diffuse interface method for simulation-based screening of heat transfer processes with complex geometries. *The Canadian Journal of Chemical Engineering*. https://doi.org/10.1002/cjce.24320

Monte, Elizabeth J. (2021). *OpenCMP: An open-source computational multiphysics package* [Master's thesis, UWSpace]. http://hdl.handle.net/10012/17239

Nguyen, L. H., Stoter, S. K. F., Ruess, M., Sanchez Uribe, M. A., & Schillinger, D. (2018). The diffuse Nitsche method: Dirichlet constraints on phase-field boundaries. *Int. J. Numer. Methods Eng.*, *113*(4), 601–633. https://doi.org/10.1002/nme.5628

Permann, C. J., Gaston, D. R., Andrš, D., Carlsen, R. W., Kong, F., Lindsay, A. D., Miller, J. M., Peterson, J. W., Slaughter, A. E., Stogner, R. H., & Martineau, R. C. (2020). MOOSE: Enabling massively parallel multiphysics simulation. *SoftwareX*, *11*, 100430. https://doi.org/10.1016/j.softx.2020.100430

Schöberl, J. (2020). *Netgen/NGSolve*. Online. https://ngsolve.org/

Yu, W., Zhang, K., & Li, X. (2015, July). Recent algorithms on automatic hexahedral mesh generation. *The 10th International Conference on Computer Science & Education*. https://doi.org/10.1109/iccse.2015.7250335