The Journal of Open Source Software

The Alamo multiphysics solver for phase field simulations with strong-form mechanics and block structured adaptive mesh refinement

Brandon Runnels [1], **Vinamra Agrawal** [2], **and Maycon Meier** [3]

1 Department of Aerospace Engineering, Iowa State University, Ames, IA, USA **2** Materials and Physical Data Group, Los Alamos National Laboratory, Los Alamos, NM, USA **3** Arizona State University, Phoenix, AZ, USA

DOI: 10.21105/joss.08581

Software

- Review
- Repository
- Archive

Editor: Fei Tao

Reviewers:

- @aroccon
- @duhd1993

Submitted: 08 March 2025
Published: 20 November 2025

Summary

Alamo is a high-performance scientific code that uses block-structured adaptive mesh refinement to solve such problems as: the ignition and burn of solid rocket propellant, plasticity, damage and fracture in materials undergoing loading, and the interaction of compressible flow with eroding solid materials. Alamo is powered by AMReX, and provides a set of unique methods, models, and algorithms that enable it to solve solid-mechanics problems (coupled to other physical behavior such as fluid flow or thermal diffusion) using the power of block-structured adaptive mesh refinement.

Statement of need

The phase field (PF) method is a powerful theoretical framework that enables the systematic description of complex physical systems (Burger & Stainko, 2006; Qin & Bhadeshia, 2010; Steinbach, 2009). PF methods have been successfully used to describe phenomena such as solidification, microstructure evolution, fracture, damage, dislocations, and many more. Beyond materials science, PF methods have also enjoyed great success in other applications ranging from deflagration of solid rocket propellant to topology optimization.

The success of the PF method is derived from its implicit, diffuse representation of boundaries and surfaces, which avoids the need for cumbersome interface tracking. However, the PF method also can incur great computational expense, due to the need for high grid resolution across the diffuse boundary. In order for the PF method to be feasible, strategic algorithms are necessary in order to provide sufficient boundary resolution without wasting grid points on uninteresting regions. Such algorithms fall typically into two main categories. (1) Spectral methods solve the phase field equations in the frequency domain, e.g. (Kochmann et al., 2015), and (2) Real-space methods employing adaptive mesh refinement (AMR). Spectral methods offer a number of performance advantages, especially when coupling to global mechanical solvers. However, they can be limited in their ability to resolve fine-scale features, and can be very cumbersome to use when implementing novel types of models. On the other hand, real-space methods with AMR are often able to attain very good performance, can be easily suited to the domain of interest, and provide an attractive platform for prototyping new physical models.

A number of open-source real-space PF codes exist and have enjoyed significant popularity. Some of the most widely known codes with PF implementations are Moose, (Giudicelli et al., 2024), Fenics (Baratta et al., 2023), and Prisms-PF (DeWitt et al., 2020), which employ octree style AMR. Though effective, octree-AMR can result on complex and expensive mesh

management. It is can also be challenging to achieve optimal load balancing, due to the high degree of unpredictable connectivity within the octree mesh.

Block-structured AMR (BSAMR) is an alternative AMR strategy. BSAMR divides the domains into distinct levels, with each level usually consisting of a collection of Cartesian grid regions (patches), that effectively evolve independently. Communication between patches and levels is then handled through ghost cells, interpolation, and restriction. This data structure is extremely efficient and scalable, while also being highly amenable to efficient code prototyping. Importantly, BSAMR also acts as a seamless extension to geometric multigrid, making naturally efficient at performing global mechanical equilibrium solves. The AMReX framework (Zhang et al., 2019) provides a powerful platform for development of BSAMR codes. However, the use of AMReX has been limited in PF (with only a few exceptions, (Kumar et al., 2023, 2024)) and solid mechanics, due to the inherent challenges of solving the mechanical equilibrium equations on a patch-based mesh.

The Alamo multiphysics solver leverages the power of BSAMR for phase-field problems. Alamo provides a unique, strong-form finite-deformation, matrix-free mechanics solver, enabling the efficient solution of the solid mechanics calculations. It also provides a set of numerical integration routines, myriad material models, and numerous examples covering a broad cross-section of PF modeling interests.

Methods

Mechanical solver

The Alamo mechanical solver extends the multi-level multi-grid (MLMG) solver to address the problem of quasi-static mechanical equilibrium, that is,

$$\mathrm{Div}\left(\mathrm{DW}(\mathbf{F})\right) + \mathbf{B} = \mathbf{0}, \tag{1}$$

where $\mathbf{F}$ is the deformation gradient, $\mathbf{B}$ is a body force, and W is an arbitrary Helmholtz free energy with derivatives $\mathbf{P} = \mathrm{DW}(\mathbf{F}) = d\mathrm{W}/d\mathbf{F}$ the Piola-Kirchhoff stress tensor, and $\mathbb{C} = \mathrm{DDW}(\mathbf{F}) = d^2\mathrm{W}/d\mathbf{F}/d\mathbf{F}$ the tangent modulus. Usually, the solution of Equation 1 is achieved using the finite element method (FEM). However, the main advantages of FEM (conformal meshing, ability to handle material discontinuities) are not relevant in the BSAMR framework. Moreover, FEM often presents difficulties in achieving consistent shape functions between levels, causing problems in achieving consistent derivatives at the coarse/fine boundary without a global matrix.

The Alamo solver, on the other hand, is developed to be native to the BSAMR framework, and takes full advantage of the integration with geometric multigrid. It is matrix-free, which is necessary in order to avoid additional communication overhead. It is strong-form, using finite differences instead of shape functions to calculate derivatives, ensuring consistency between levels and compatibility with restriction/prolongation operations. It handles coarse-fine boundaries using a novel "reflux-free" method, which avoids the special treatment of boundaries by including an extra layer of smoothed ghost nodes. Details on these aspects of the solver are available in (Runnels et al., 2021).

The Alamo mechanical solver is versatile, allowing any type of mechanical model to be used though the abstract solid model interface (per the norm for most FEM codes). Alamo uses templated AMReX BaseFab structures to encapsulate model parameters, which enables users to implement sophisticated models without any knowledge of the external Alamo/AMReX infrastructure. An optional interface is provided that allows users to communicate model information to the Alamo I/O routines, in order to include internal solid mechanics variables (such as accumulated plastic slip) with the field output. Because mechanical models are instantiated as field variables, they require arithmetic operations in order to be interpolated/averaged as the mesh is adapted. Alamo implements a "model vector space" framework that provides an

intuitive and efficient framework for automatic creation of arithmetic operations. This allows model developers to implement all necessary arithmetic operators with only three additional lines of code, ensuring systematic compliance of the model and increased readability of the code. The flexibility of this framework is exemplified through Alamo's library of solid models, which include implementations of solids ranging from linear elastic isotropic materials to finite deformation strain gradient crystal plasticity.

Because the mechanical solver is coupled to problems defined with diffuse boundaries, often involving "void" regions in which there is no mechanical strength, additional steps are necessary to avoid convergence issues. Alamo contains methods for accounting for diffuse boundary conditions, and uses a joint cell/node based interpolation scheme to ensure good convergence even when the operator is near-singular. Details on the near-singular solver capability, the model vector space implementation, and the methods for applying boundary conditions in the diffuse boundary framework, have been documented in (Agrawal & Runnels, 2023).

Multiple inheritance polymorphic integrators

In Alamo, each type of physical behavior is encapsulated by an "Integrator" class. Each integrator is responsible for solving a certain set of equations: the HeatConduction integrator solves the thermal diffusion equation; the AllenCahn integrator solves the Allen-Cahn equations; the Mechanics integrator solves the equations of mechanical equilibrium; etc. All integrators inherit from the base integrator, which interfaces with AMReX and manages the creation/deletion/evolution of field variables. This partitions the code so that each integrator is responsible only for its own physics, without excessive bookkeeping infrastructure.

Most PF problems of interest feature the non-trivial interaction of multiple disparate physical behaviors. For example, microstructure evolution in metals is strongly coupled to mechanical loading. Or, some phase field models may require a fully resolved flow-field. Such couplings can rapidly increase the complexity of the code base.

The Alamo solution is **multiple-inheritance polymorphic integrators** (MIPI). The MIPI schema encapsulates each physical system into its own integrator. Single-application integrators inherit directly from the base integrator, whereas multi-physics application link other integrators together using multiple inheritance.

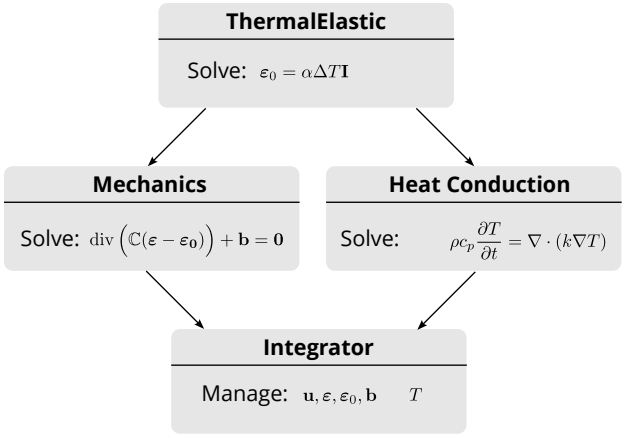

Figure 1: The MIPI schema for solving thermoelasticity.

The MIPI method is exemplified through the case of coupled thermal evolution and mechanical equilibrium (Figure 1). In this example, the Mechanics integrator is only responsible for solving mechanical equilibrium and can be run independently. Similarly, the Heat Conduction integrator is concerned only with thermal evolution; all physical aspects related to heat conduction are encapsulated here. Both Mechanics and Heat Conduction inherit from the integrator base

class, which tracks and updates all field variables on a shared grid. To link these together, the ThermalElastic integrator inherits jointly from Mechanics and HeatConduction. Its only function is to link the Mechanics strain field to the Heat Conduction temperature field; everything else is managed by the respective integrator. This allows for integrators to be combined in arbitrary ways, with minimal (or possible no) instrumentation required within the linked integrator itself.

Numerous alamo integrators have been developed and used for scientific applications. A brief summary is included here:

- Microstructure evolution is simulated using the multi-phase field method (Eren et al., 2022) combined with the strong-form mechanics solver to simulate grain boundary anisotropy (Ribot et al., 2019), phase field disconnections (Gokuli & Runnels, 2021; Runnels & Agrawal, 2020), and twin growth in magnesium (Hu et al., 2024).
- The phase field fracture mechanics model couples crack evolution and mechanics to capture crack propagation (Agrawal & Runnels, 2021, 2023).
- Deflagration of solid rocket propellant is captured using a phase field method (Kanagarajan et al., 2022), coupled to heat transfer (Meier et al., 2024) and the hyperelastic solver (Meier & Runnels, 2024) to accurately predict ignition, deflagration, and mechanical response.
- Recent work demonstrates the ability of Alamo to simulate a hydrodynamic compressible flow through a domain governed by phase field equations including the Allen Cahn equation (for porous media) and the dendrite growth equation (Boyd et al., 2025).

Infrastructure

Alamo is intended to bridge the gap between *prototype/research codes* (often written with *ad-hoc* structure in a non-scalable language with minimal documentation) and *production codes* (which are well-organized and documented, but have a steep learning curve and restrictive contribution requirements). In other words, it is designed to be easy for an inexperienced researcher to learn, contribute to, and run without excessive time spent learning the infrastructure. Automation is therefore the key to ensure that Alamo retains production-level quality without requiring excessive effort on the part of the developers.

Parameter parsing and automatic documentation

Alamo employs a recursive object-oriented parameter parsing system. Each class contains a standard `Parse` function to read its member variables, and is located as close as possible to the location in the code as where the variables are used. Parser commands self-document the nature of each variable: for instance, `query_default` provides a default value, `query_required` triggers an error if the variable is not provided, `query_validate` provides a list of acceptable values, and so on. This is effective at eliminating uninitialized variables, and localizes the variable's usage with its parsing statement. All parameters are read in hierarchically, systematically using prefixes to avoid naming conflicts (especially in the case of MIPI integrators).

In addition, Alamo contains a set of python-based code scrapers to scan the source code for all input parameters. The inputs are cataloged, along with their source code location and comment string, and formatted into the automatically-generated documentation. This allows users to browse the inputs and link directly to their usage in the code (a common difficulty when encountering a new code), or to search an input index to determine how the inputs are used. It is also used to automatically generate input file builders that are guaranteed to be accurate and consistent with the current source code.

Continuous integration is used to require that all inputs be adequately documented. Then, the documentation is automatically generated and posted online with every addition to the main development branches. This ensures that all documentation is kept current with the source

code, without requiring anything of the developer beyond a single comment string for each input.

Automatic regression and benchmark testing system

Regression tests are essential to ensure reliability of a continuously developed code. Alamo has a self-contained regression test system designed to run with a single line of code added to an input file stored in the repository. It also contains a suite of python helper functions that use the YT library to automatically extract data for comparison and to determine whether runs have completed accurately (Turk et al., 2010). The regression test system is automated using the Github Actions CI system, and different variants of tests are executed automatically in different stages of code development. For example, full-scale benchmark tests are run on a self-hosted runner in each of the main development branches; lightweight suite of tests are run upon every feature branch commit. Selections of the tests are used to check code coverage using gcov, and memory safety using ASan and MSan.

Regression tests are beneficial also for providing starting points for new users and developers of the code. Alamo's automatic documentation system integrates with the automatic test system to generate comprehensive documentation for each test, including figures and run instructions. Since it is implemented with CI, the online test documentation is guaranteed to be up-to-date.

Guaranteed reproducibility

A hallmark feature of Alamo is its guaranteed reproducibility system. Alamo integrates with the AMReX ParmParse system to track all input parameters. On every execution, Alamo creates a metadata file in the output directory that stores all input parameters, as well as platform information, git commit ID, amrex git commit ID, etc.

Because Alamo is a development code, sometimes last-minute modifications are made to the source that are not committed prior to the code's execution. To resolve this, as part of the build process, Alamo always runs a git diff on its own source code, and stores the results. When the code is subsequently executed, the git diff is stored with the output, ensuring that future users can always revert the code back to the exact same state at a later time. This guarantees that every Alamo result, regardless of the state of the code, is reproducible.

Acknowledgments

The authors acknowledge the many funding sources that have supported the development of Alamo. This includes: Support from Lawrence Berkeley National Laboratory, subcontracts #7473053, #7645776; National Science Foundation, grants #OAC-2017971, #MOMS-2142164, #MOMS-2341922; and the Office of Naval Research, grants #N00014-21-1-2113, #N00014-25-1-2029.

References

Agrawal, V., & Runnels, B. (2021). Block structured adaptive mesh refinement and strong form elasticity approach to phase field fracture with applications to delamination, crack branching and crack deflection. *Computer Methods in Applied Mechanics and Engineering*, *385*, 114011. https://doi.org/10.1016/j.cma.2021.114011

Agrawal, V., & Runnels, B. (2023). Robust, strong form mechanics on an adaptive structured grid: Efficiently solving variable-geometry near-singular problems with diffuse interfaces. *Computational Mechanics*. https://doi.org/10.1007/s00466-023-02325-8

Baratta, I. A., Dean, J. P., Dokken, J. S., Habera, M., HALE, J., Richardson, C. N., Rognes, M. E., Scroggs, M. W., Sime, N., & Wells, G. N. (2023). *DOLFINx: The next generation FEniCS problem solving environment*. https://doi.org/10.5281/zenodo.10447666

Boyd, E. M., Sandall, E., Meier, M., Quinlan, J. M., & Runnels, B. (2025). A diffuse boundary method for phase boundaries in viscous compressible flow. *arXiv Preprint arXiv:2502.16053*. https://doi.org/10.48550/arXiv.2502.16053

Burger, M., & Stainko, R. (2006). Phase-field relaxation of topology optimization with local stress constraints. *SIAM Journal on Control and Optimization*, *45*(4), 1447–1466. https://doi.org/10.1137/05062723X

DeWitt, S., Rudraraju, S., Montiel, D., Andrews, W. B., & Thornton, K. (2020). PRISMS-PF: A general framework for phase-field modeling with a matrix-free finite element method. *Npj Computational Materials*, *6*(1), 29. https://doi.org/10.1038/s41524-020-0298-5

Eren, E., Runnels, B., & Mason, J. (2022). Comparison of evolving interfaces, triple points, and quadruple points for discrete and diffuse interface methods. *Computational Materials Science*, *213*, 111632. https://doi.org/10.1016/j.commatsci.2022.111632

Giudicelli, G., Lindsay, A., Harbour, L., Icenhour, C., Li, M., Hansel, J. E., German, P., Behne, P., Marin, O., Stogner, R. H., Miller, J. M., Schwen, D., Wang, Y., Munday, L., Schunert, S., Spencer, B. W., Yushu, D., Recuero, A., Prince, Z. M., … Permann, C. (2024). 3.0 - MOOSE: Enabling massively parallel multiphysics simulations. *SoftwareX*, *26*, 101690. https://doi.org/10.1016/j.softx.2024.101690

Gokuli, M., & Runnels, B. (2021). Multiphase field modeling of grain boundary migration mediated by emergent disconnections. *Acta Materialia*, *217*, 117149. https://doi.org/10.1016/j.actamat.2021.117149

Hu, Y., Kochmann, D. M., & Runnels, B. (2024). Atomistic-informed phase field modeling of magnesium twin growth by disconnections. *Acta Materialia*. https://doi.org/10.1016/j.actamat.2024.120564

Kanagarajan, B., Quinlan, J. M., & Runnels, B. (2022). A diffuse interface method for solid-phase modeling of regression behavior in solid composite propellants. *Combustion and Flame*. https://doi.org/10.1016/j.combustflame.2022.112219

Kochmann, J., Wulfinghoff, S., Svendsen, B., & Reese, S. (2015). Phase-field modeling of martensitic phase transformations in polycrystals coupled with crystal plasticity–a spectral-based approach. *PAMM*, *15*(1), 317–318.

Kumar, P., Hoffmann, M., Nonaka, A., Salahuddin, S., & Yao, Z. (2024). 3D ferroelectric phase field simulations of polycrystalline multi-phase hafnia and zirconia based ultra-thin films. *Advanced Electronic Materials*, *10*(10), 2400085. https://doi.org/10.1002/aelm.202400085

Kumar, P., Nonaka, A., Jambunathan, R., Pahwa, G., Salahuddin, S., & Yao, Z. (2023). FerroX: A GPU-accelerated, 3D phase-field simulation framework for modeling ferroelectric devices. *Computer Physics Communications*, *290*, 108757. https://doi.org/10.1016/j.cpc.2023.108757

Meier, M., & Runnels, B. (2024). Finite kinematics diffuse interface mechanics coupled to solid composite propellant deflagration. *Computer Methods in Applied Mechanics and Engineering*, *427*, 117040. https://doi.org/10.1016/j.cma.2024.117040

Meier, M., Schmidt, E., Martinez, P., Quinlan, J. M., & Runnels, B. (2024). Diffuse interface method for solid composite propellant ignition and regression. *Combustion and Flame*, *259*, 113120. https://doi.org/10.1016/j.combustflame.2023.113120

Qin, R., & Bhadeshia, H. (2010). Phase field method. *Materials Science and Technology*, *26*(7), 803–811. https://doi.org/10.1179/174328409X453190

Ribot, J. G., Agrawal, V., & Runnels, B. (2019). A new approach for phase field modeling of grain boundaries with strongly nonconvex energy. *Modelling and Simulation in Materials Science and Engineering*, *27*(8), 084007. https://doi.org/10.1088/1361-651X/ab47a0

Runnels, B., & Agrawal, V. (2020). Phase field disconnections: A continuum method for disconnection-mediated grain boundary motion. *Scripta Materialia*, *186*, 6–10. https://doi.org/10.1016/j.scriptamat.2020.04.042

Runnels, B., Agrawal, V., Zhang, W., & Almgren, A. (2021). Massively parallel finite difference elasticity using block-structured adaptive mesh refinement with a geometric multigrid solver. *Journal of Computational Physics*, *427*, 110065. https://doi.org/10.1016/j.jcp.2020.110065

Steinbach, I. (2009). Phase-field models in materials science. *Modelling and Simulation in Materials Science and Engineering*, *17*(7), 073001. https://doi.org/10.1088/0965-0393/17/7/073001

Turk, M. J., Smith, B. D., Oishi, J. S., Skory, S., Skillman, S. W., Abel, T., & Norman, M. L. (2010). Yt: A multi-code analysis toolkit for astrophysical simulation data. *The Astrophysical Journal Supplement Series*, *192*(1), 9. https://doi.org/10.1088/0067-0049/192/1/9

Zhang, W., Almgren, A., Beckner, V., Bell, J., Blaschke, J., Chan, C., Day, M., Friesen, B., Gott, K., Graves, D., & others. (2019). AMReX: A framework for block-structured adaptive mesh refinement. *The Journal of Open Source Software*, *4*(37), 1370. https://doi.org/10.21105/joss.01370