The Journal of Open Source Software

Ratel: Performance portable solid mechanics with libCEED and PETSc

Zachary R. Atkins [1], **Jed Brown** [1], **Fabio Di Gioacchino** [1], **Layla Ghaffari** [1], **Zachariah T. Irwin** [1], **Rezgar Shakeri** [1], **Karen Stengel** [1], **and Jeremy L Thompson** [1]

1 University of Colorado at Boulder

DOI: 10.21105/joss.08388

Software

- Review
- Repository
- Archive

Editor: Mojtaba Barzegari

Reviewers:

- @gtheler
- @thelfer
- @hvonwah

Submitted: 08 April 2025
Published: 11 February 2026

Summary

Ratel is a solid mechanics library and applications based on libCEED (Abdelfattah, Barra, Beams, Brown, et al., 2021; Brown et al., 2021) and PETSc (Balay et al., 2025; Zhang et al., 2021) with support for efficient high-order elements and CUDA and ROCm GPUs.

Solid mechanics simulations provide vital information for many engineering applications, using a large amount of computational resources from workstation to supercomputing scales. The industry standard for implicit analysis uses assembled sparse matrices with low-order elements, typically Q_1 hexahedral and P_2 tetrahedral elements, with the linear systems solved using sparse direct solvers, algebraic multigrid, or multilevel domain decomposition. This approach has two fundamental inefficiencies: poor approximation accuracy per Degree of Freedom (DoF) and high computational and memory cost per DoF due to choice of data structures and algorithms. High-order finite elements implemented in a matrix-free fashion with appropriate preconditioning strategies can overcome these inefficiencies. Integrating more efficient data structures and algorithms into solid mechanics software libraries can greatly improve engineering workflows and allow users to better utilize their computational resources while new technologies like Automatic Differentiation can be used to shorten development time.

State of the field

Most finite element method (FEM) software packages for implicit analysis have limited performance due to the centrality of assembled sparse matrices, which limit performance to less than 2% of peak floating-point operations per second (FLOPs) due to low arithmetic intensity (FLOP per byte of memory transferred from memory) (Williams et al., 2009), rendering performance limited by memory bandwidth. Modern hardware architectures, including GPUs, favor algorithms with an intensity of 10 FLOP/byte or more (Rupp, 2020), while sparse matrices have intensities of less than 0.25 FLOP/byte. Matrix-free methods deliver higher performance than assembled matrices at both high- and low-order (Abdelfattah, Barra, Beams, Bleile, et al., 2021; Brown et al., 2022; Kolev et al., 2021; May et al., 2014). Additionally, users who want to run their applications on High Performance Computers (HPC) are limited by the underlying requirement to write backend (hardware) specific code since different chips have different libraries needed to interface with the hardware; for example, NVIDIA devices require the use of *CUDA* (2025) while AMD GPUs require *HIP* (2025). Users who wish to run their code on different hardware must take the time to ensure that their code is compatible with all of the targeted hardware, significantly increasing code development time.

MFEM (Anderson et al., 2021) and deal.ii (Arndt et al., 2021) are two libraries with some comparable features. In contrast to MFEM, Ratel does not currently offer the ability for users

to customize the formulation of the material models past the parameters; however, Ratel offers a growing material point method capability and focuses on delivering high performance with libCEED's code generation GPU backends. In contrast to deal.ii, Ratel uses C99 and code generation for performance over C++ templates, in an effort to provide error messages that are easier to debug. Also, Ratel has more robust and mature GPU support than deal.ii.

Abaqus (*Abaqus*, 2025) is another widely used commercial package with comparable features. Developing constitutive models requires tensor derivatives of nonlinear scalar functions. In Abaqus, extensions through UMAT or UHYPER interfaces are typically implemented via manual derivation and coding of these derivatives, following solver-specific conventions that can be error-prone and, in some cases, lead to instabilities (Shakeri et al., 2024). While it is possible to integrate external automatic differentiation (AD) tools such as Enzyme or ADOL-C into UMAT implementations, doing so generally requires additional setup and expertise. However, Ratel addresses this by leveraging AD to compute exact derivatives directly from a nonlinear strain energy function, providing first derivatives for constitutive modeling and second derivatives for the consistent tangent.

Statement of need

Ratel is a solid mechanics library that provides material models and boundary conditions that utilize the high-order matrix-free capabilities of libCEED (Abdelfattah, Barra, Beams, Brown, et al., 2021; Brown et al., 2021) and the linear and non-linear solvers from PETSc (Balay et al., 2025; Zhang et al., 2021). Ratel supports both FEM and implicit material point method (iMPM) implementations (Coombs et al., 2020; Moresi et al., 2003), allowing users to select their method of choice at run-time. Ratel has an extensible materials library that includes finite-strain hyperelastic, elastoplastic, viscoelastic, poroelastic, and fracture models, including stable mixed formulations for near-incompressible regimes. These models are written in C99 and only contain code syntax also supported by CUDA and HIP. Because libCEED lets users write hardware-agnostic code, Ratel users can run solid mechanics simulations with any libCEED-supported backend, selecting between CPUs or GPUs at runtime.

Ratel users can take advantage of all the packages and algorithms supported by PETSc, including Hypre (Falgout et al., 2021) and Kokkos (Trott et al., 2022). Ratel accepts runtime arguments as command-line flags and/or YAML format, providing users the flexibility to use different libCEED backends, PETSc solvers and preconditioners, or Ratel material models easily. The performance benefits of Ratel's approach to solid mechanics are explored in Brown et al. (2022).

Concepts and interface

The role of Ratel is to properly set up a PETSc domain management (`DM`) object representing the mesh for the user's prescribed solid mechanics simulation and create the libCEED operators to compute the non-linear residual evaluations for PETSc time-steppers (`TS`) and non-linear solvers (`SNES`) as well as the Jacobian evaluations for PETSc linear solvers (`KSP`), as well as any ancillary operators for solution postprocessing. libCEED provides performant implementations of these operators on the targeted hardware (Figure 1). These libCEED operators are attached to the `DM`'s non-linear solver context (`DMSNES`) or time-stepper context (`DMTS`) which configures the user's selected solver with the appropriate function callbacks to run the user's simulation.

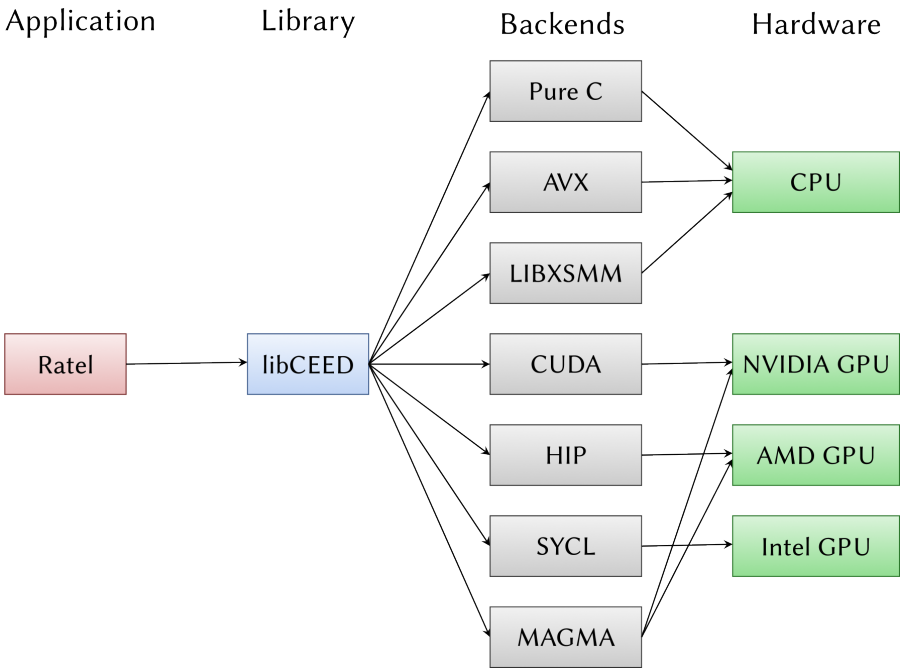

Figure 1: Runtime backend options for libCEED

Ratel currently only uses Continuous Galerkin finite elements; however other types of finite elements supported by PETSc and libCEED could be added in the future.

Ratel is organized around `RatelMaterial` objects, separate material regions in the domain (Figure 2). Each `RatelMaterial` is responsible for adding volumetric terms to the residual and Jacobian operators as well as any surface terms that require volumetric or material model values. Additionally, each `RatelMaterial` is responsible for building and configuring corresponding preconditioner components, as needed. Users can configure `RatelMaterials` at runtime, identifying the material parameters and mesh regions on which to apply use each material.

Boundary and forcing terms that do not require volumetric or material model parameters, such as Dirichlet boundary conditions, are handled separately from `RatelMaterial`. `RatelBoundary` objects can also be specified at runtime.

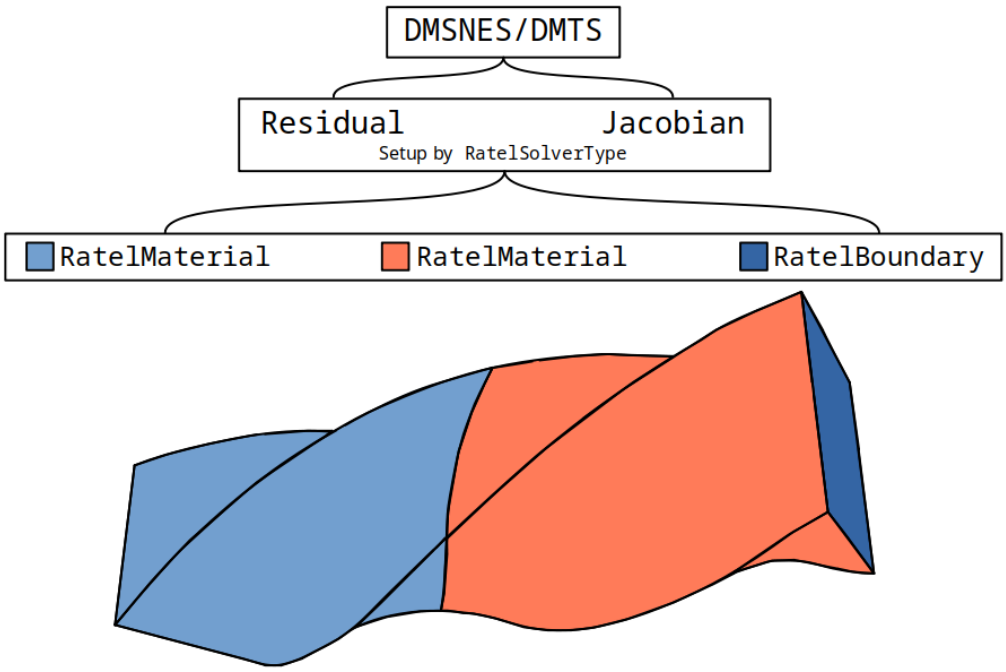

Figure 2: Ratel interface design

Material models and boundary conditions

Similar to other FEM packages, Ratel provides users with several material models to use in simulations such as linear elasticity, Hencky, neo-Hookean, Mooney-Rivlin, and Ogden hyperelasticity models (Holzapfel, 2000; Hughes, 2012), available in both compressible and incompressible limits. These models are written in a numerically stable way (Shakeri et al., 2024) to improve accuracy by reducing floating-point errors. In addition to the elasticity models, Ratel material development can be broken up into three main categories: matrix-free implementations of complex material models, the use of AD to simplify material model implementation, and MPM versions of material models.

Advanced material models in Ratel include a phase-field model for brittle fracture with AT1 and AT2 damage (Amor et al., 2009; Miehe et al., 2010; Tanné et al., 2018), linear and finite strain von Mises plasticity (Eterovic & Bathe, 1990; Weber & Anand, 1990), and both linear (Biot's) (Cheng, 2016; Ding et al., 2013) and finite strain (Irwin et al., 2024) poroelasticity models. Ratel supports running simulations with multiple material models on different regions of a mesh.

There is also ongoing research around incorporating AD tools such as Enzyme (W. S. Moses et al., 2021; W. Moses & Churavy, 2020) and ADOL-C (Walther & Griewank, 2012) into the material model development pipeline. AD allows the user to avoid the time-consuming and error-prone process of calculating complex derivatives by hand for the Jacobians. There are several existing examples in Ratel demonstrating the use of both Enzyme and ADOL-C for hyperelastic material models, with ongoing work towards an AD plasticity model.

Finally, Ratel has an iMPM implementation, based on Coombs et al. (2020) and Moresi et al. (2003), with iMPM versions of the neo-Hookean and damage material models. Like the FEM material models, the iMPM material models utilize the libCEED interface, allowing iMPM simulations to be run on any libCEED-supported architecture, including AMD and NVIDIA GPUs.

Ratel also provides several boundary conditions. Ratel offers two main forms of Dirichlet or

essential boundaries, clamp and slip. Clamp boundary conditions are displacement-specific and constrain the entire displacement field on the face; the constrained values specify linear and rotational displacement on the face at a given point in time. Slip boundary conditions are a more general option, allowing for individual components of a field to have values prescribed. Slip boundaries allow for the specification of symmetry or linearly interpolated values.

Traction (Neumann) boundary conditions, which apply external forces on a surface are supported. Pressure boundary conditions from liquids or gases acting on the surface of the solid structure are also supported, with the load depending on the current deformation state. Finally, Ratel supports contact boundary conditions with rigid obstacles using Nitsche's method (Mlika, 2018) either frictionless or with Coulomb or Threlfall friction models (Marques et al., 2016). These boundary conditions are configured via command line options and may be time varying.

Examples

Ratel is a library that can be integrated into existing applications and also provides driver applications for `static`, `quasistatic`, and `dynamic` simulation and analysis. Here, we demonstrate some example simulations using these drivers.

The first example is a coarse quasistatic simulation of a compressive shear test for a generic brittle material. It employs the AT2 phase-field model in a monolithic scheme, incorporating damage viscous regularization and adaptive time-stepping. This example highlights Ratel's capability to handle advanced material models in non-trivial loading conditions.

```
bin/ratel-quasistatic -options_file examples/ymls/ex02/linear-damage-compressiveshear-AT2-face-forces.yml
```

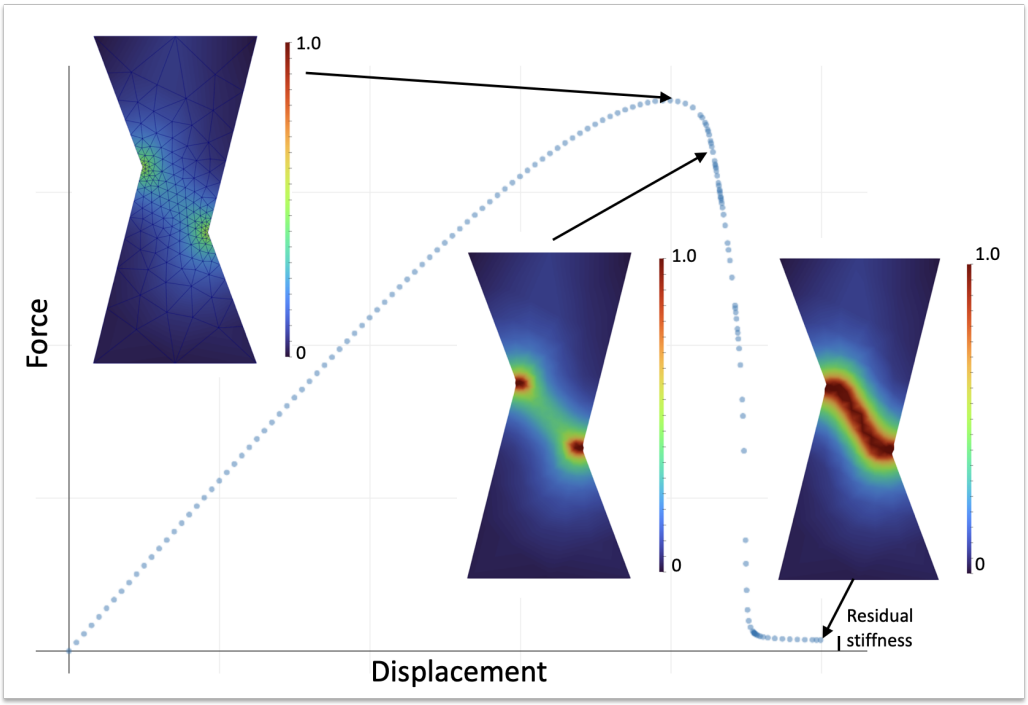

Figure 3: Output from the compressive shear test example showing the force-displacement curve with phase field results for damage at selected amounts of applied strain.

The second example is a quasistatic iMPM simulation of a cylinder with a dense high-modulus inclusion surrounded by a low-density low-modulus near-incompressible "foam" (Figure 4).

This simulation demonstrates Ratel's ability to handle mixed materials and curved geometry as well as the iMPM simulation capability. Note that for the iMPM simulation there is no mesh deformation like there is in FEM simulations since only the particles move in iMPM.

```
bin/ratel-quasistatic -options_file examples/ymls/ex02/mpm-neo-hookean-damage-current-sinker-cylinder.yml
```

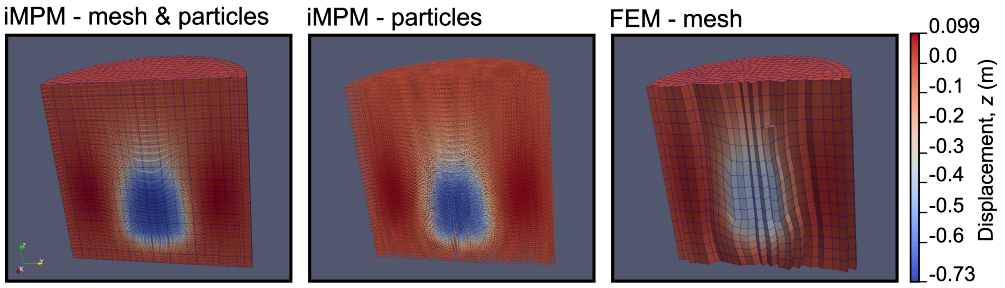

Figure 4: Output from the sinker example showing iMPM mesh with particles overlaid, iMPM particles, and the equivalent mesh with FEM.

The next example is a dynamic simulation of 8 Schwarz-P cells with an applied traction force creating pendulum-like motion (Figure 5). Schwarz-P meshes are especially helpful in studying solver and preconditioner robustness and scaling.

```
bin/ratel-dynamic -options_file examples/ymls/ex03/schwarz-pendulum-enzyme.yml
```

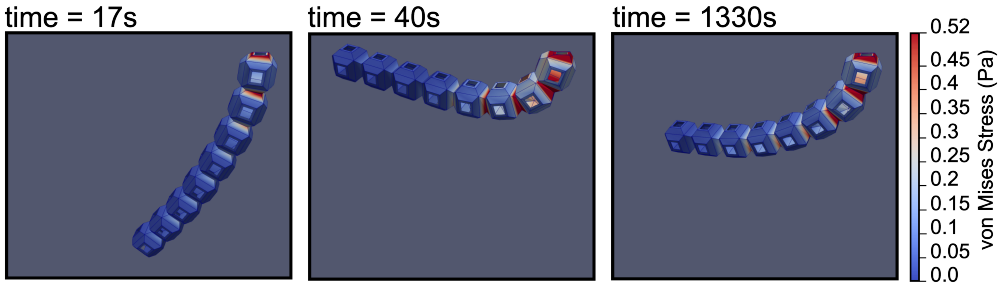

Figure 5: Example time steps from the dynamic pendulum simulation using the Enzyme implementation of the neo-Hookean model.

Acknowledgements

This research is supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research under contract DE-AC02-06CH11357 and Award Number DE-SC0016140. The authors acknowledge support by the Department of Energy, National Nuclear Security Administration, Predictive Science Academic Alliance Program (PSAAP) under Award Number DE-NA0003962.

References

Abaqus. (2025). https://www.3ds.com/products/simulia/abaqus

Abdelfattah, A., Barra, V., Beams, N., Bleile, R., Brown, J., Camier, J.-S., Carson, R., Chalmers, N., Dobrev, V., Dudouit, Y., Fischer, P., Karakus, A., Kerkemeier, S., Kolev,

T., Lan, Y.-H., Merzari, E., Min, M., Phillips, M., Rathnayake, T., … Weiss, K. (2021). GPU algorithms for efficient exascale discretizations. *Parallel Computing*, *108*, 102841. https://doi.org/10.1016/j.parco.2021.102841

Abdelfattah, A., Barra, V., Beams, N., Brown, J., Camier, J.-S., Dobrev, V., Dudouit, Y., Ghaffari, L., Kolev, T., Medina, D., Pazner, W., Rathnayake, T., Thompson, J. L., & Tomov, S. (2021). *libCEED user manual* (Version 0.8). Zenodo. https://doi.org/10.5281/zenodo.4895340

Amor, H., Marigo, J.-J., & Maurini, C. (2009). Regularized formulation of the variational brittle fracture with unilateral contact: Numerical experiments. *Journal of the Mechanics and Physics of Solids*, *57*(8), 1209–1229. https://doi.org/10.1016/j.jmps.2009.04.011

Anderson, R., Andrej, J., Barker, A., Bramwell, J., Camier, J.-S., Cerveny, J., Dobrev, V., Dudouit, Y., Fisher, A., Kolev, T., Pazner, W., Stowell, M., Tomov, V., Akkerman, I., Dahm, J., Medina, D., & Zampini, S. (2021). MFEM: A Modular Finite Element Methods Library. *Computers & Mathematics with Applications*, *81*, 42–74. https://doi.org/10.1016/j.camwa.2020.06.009

Arndt, D., Bangerth, W., Davydov, D., Heister, T., Heltai, L., Kronbichler, M., Maier, M., Pelteret, J.-P., Turcksin, B., & Wells, D. (2021). The deal.II finite element library: Design, features, and insights. *Computers & Mathematics with Applications*, *81*, 407–422. https://doi.org/10.1016/j.camwa.2020.02.022

Balay, S., Abhyankar, S., Adams, M. F., Bensor, S., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Dener, A., Eijkhout, V., Faibussowitsch, J., Gropp, W. D., Hapla, V., Isaac, T., Jolivet, P., Dmitry Karpeyev, and, Kaushik, D., Knepley, M. G., Kong, F., … Zang, J. (2025). *PETSc users manual* (ANL-95/11 - Revision 3.23). Argonne National Laboratory. https://doi.org/10.2172/2476320

Brown, J., Abdelfattah, A., Barra, V., Beams, N., Camier, J.-S., Dobrev, V., Dudouit, Y., Ghaffari, L., Kolev, T., Medina, D., Pazner, W., Ratnayaka, T., Thompson, J., & Tomov, S. (2021). libCEED: Fast algebra for high-order element-based discretizations. *Journal of Open Source Software*, *6*(63), 2945. https://doi.org/10.21105/joss.02945

Brown, J., Barra, V., Beams, N., Ghaffari, L., Knepley, M., Moses, W., Shakeri, R., Stengel, K., Thompson, J. L., & Zhang, J. (2022). *Performance portable solid mechanics via matrix-free p-multigrid*. arXiv. https://doi.org/10.48550/ARXIV.2204.01722

Cheng, A. (2016). *Poroelasticity*. Springer Cham. https://doi.org/10.1007/978-3-319-25202-5

Coombs, W. M., Augarde, C. E., Brennan, A. J., Brown, M. J., Charlton, T. J., Knappett, J. A., Ghaffari Motlagh, Y., & Wang, L. (2020). On lagrangian mechanics and the implicit material point method for large deformation elasto-plasticity. *Computer Methods in Applied Mechanics and Engineering*, *358*, 112622. https://doi.org/10.1016/j.cma.2019.112622

CUDA. (2025). https://developer.nvidia.com/about-cuda

Ding, B., Cheng, A. H.-D., & Chen, Z. (2013). Fundamental solutions of poroelastodynamics in frequency domain based on wave decomposition. *Journal of Applied Mechanics*, *80*(6), 061021. https://doi.org/10.1115/1.4023692

Eterovic, A. L., & Bathe, K.-J. (1990). A hyperelastic-based large strain elasto-plastic constitutive formulation with combined isotropic-kinematic hardening using the logarithmic stress and strain measures. *International Journal for Numerical Methods in Engineering*, *30*(6), 1099–1114. https://doi.org/10.1002/nme.1620300602

Falgout, R. D., Li, R., Sjögreen, B., Wang, L., & Yang, U. M. (2021). Porting hypre to heterogeneous computer architectures: Strategies and experiences. *Parallel Computing*, *108*, 102840. https://doi.org/10.1016/j.parco.2021.102840

HIP. (2025). https://rocmdocs.amd.com/en/latest/what-is-rocm.html

Holzapfel, G. (2000). *Nonlinear solid mechanics: A continuum approach for engineering*. Wiley. ISBN: 978-0-471-82319-3

Hughes, T. J. (2012). *The finite element method: Linear static and dynamic finite element analysis*. Courier Corporation. https://doi.org/10.1016/0045-7825(87)90013-2

Irwin, Z. T., Clayton, J. D., & Regueiro, R. A. (2024). A large deformation multiphase continuum mechanics model for shock loading of soft porous materials. *International Journal for Numerical Methods in Engineering*, *125*(6), e7411. https://doi.org/10.1002/nme.7411

Kolev, T., Fischer, P., Min, M., Dongarra, J., Brown, J., Dobrev, V., Warburton, T., Tomov, S., Shephard, M. S., Abdelfattah, A., Barra, V., Beams, N., Camier, J.-S., Chalmers, N., Dudouit, Y., Karakus, A., Karlin, I., Kerkemeier, S., Lan, Y.-H., … Tomov, V. (2021). Efficient exascale discretizations: High-order finite element methods. *International Journal of High Performance Computing Applications*. https://doi.org/10.1177/10943420211020803

Marques, F., Flores, P., Pimenta Claro, J. C., & Lankarani, H. M. (2016). A survey and comparison of several friction force models for dynamic analysis of multibody mechanical systems. *Nonlinear Dynamics*, *86*(3), 1407–1443. https://doi.org/10.1007/s11071-016-2999-3

May, D. A., Brown, J., & Pourhiet, L. L. (2014). pTatin3D: High-performance methods for long-term lithospheric dynamics. *Proceedings of SC14: International Conference for High Performance Computing, Networking, Storage and Analysis*. https://doi.org/10.1109/SC.2014.28

Miehe, C., Hofacker, M., & Welschinger, F. (2010). A phase field model for rate-independent crack propagation: Robust algorithmic implementation based on operator splits. *Computer Methods in Applied Mechanics and Engineering*, *199*(45-48), 2765–2778. https://doi.org/10.1016/j.cma.2010.04.011

Mlika, R. (2018). *Nitsche method for frictional contact and self-contact: Mathematical and numerical study* [PhD thesis, Université de Lyon]. https://theses.hal.science/tel-02067118

Moresi, L., Dufour, F., & Mühlhaus, H.-B. (2003). A lagrangian integration point finite element method for large deformation modeling of viscoelastic geomaterials. *Journal of Computational Physics*, *184*(2), 476–497. https://doi.org/10.1016/S0021-9991(02)00031-1

Moses, W. S., Churavy, V., Paehler, L., Hückelheim, J., Narayanan, S. H. K., Schanen, M., & Doerfert, J. (2021). Reverse-mode automatic differentiation and optimization of GPU kernels via enzyme. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. https://doi.org/10.1145/3458817.3476165

Moses, W., & Churavy, V. (2020). Instead of rewriting foreign code for machine learning, automatically synthesize fast gradients. *Advances in Neural Information Processing Systems (NeurIPS)*, *33*, 12472–12485.

Rupp, K. (2020). *CPU-GPU-MIC comparision charts*. https://github.com/karlrupp/cpu-gpu-mic-comparison

Shakeri, R., Ghaffari, L., Thompson, J. L., & Brown, J. (2024). Stable numerics for finite-strain elasticity. *International Journal for Numerical Methods in Engineering*, *125*(21), e7563. https://doi.org/10.1002/nme.7563

Tanné, E., Li, T., Bourdin, B., Marigo, J.-J., & Maurini, C. (2018). Crack nucleation in variational phase-field models of brittle fracture. *Journal of the Mechanics and Physics of Solids*, *110*, 80–99. https://doi.org/10.1016/j.jmps.2017.09.006

Trott, C. R., Lebrun-Grandié, D., Arndt, D., Ciesko, J., Dang, V., Ellingwood, N., Gayatri, R., Harvey, E., Hollman, D. S., Ibanez, D., Liber, N., Madsen, J., Miles, J., Poliakoff, D., Powell, A., Rajamanickam, S., Simberg, M., Sunderland, D., Turcksin, B., & Wilke, J. (2022). Kokkos 3: Programming model extensions for the exascale era. *IEEE Transactions on Parallel and Distributed Systems*, *33*(4), 805–817. https://doi.org/10.1109/TPDS.2021.3097283

Walther, A., & Griewank, A. (2012). Getting started with ADOL-c. *Computing in Science & Engineering*, *14*(6), 20–29. https://doi.org/10.1201/b11644-8

Weber, G., & Anand, L. (1990). Finite deformation constitutive equations and a time integration procedure for isotropic, hyperelastic-viscoplastic solids. *Computer Methods in Applied Mechanics and Engineering*, *79*(2), 173–202. https://doi.org/10.1016/0045-7825(90)90131-5

Williams, S., Waterman, A., & Patterson, D. (2009). Roofline: An insightful visual performance model for multicore architectures. *Communications of the ACM*, *52*(4), 65–76. https://doi.org/10.1145/1498765.1498785

Zhang, J., Brown, J., Balay, S., Faibussowitsch, J., Knepley, M., Marin, O., Mills, R. T., Munson, T., Smith, B. F., & Zampini, S. (2021). The PetscSF scalable communication layer. *IEEE Transactions on Parallel and Distributed Systems*. https://doi.org/10.1109/TPDS.2021.3084070