The Journal of Open Source Software

Parallel-CDM: Parallel Implementation of Continuum Damage Mechanics Simulations using FEM and MATLAB

Habiba Eldababy [1,2], **Roshan Philip Saji**[1,2], **Panos Pantidis**[3], **and Mostafa Mobasher**[1,3]

1 Mechanical Engineering Department, Tandon School of Engineering, New York University, USA **2** Mechanical Engineering Department, New York University Abu Dhabi, UAE **3** Civil and Urban Engineering Department, New York University Abu Dhabi, UAE

DOI: 10.21105/joss.07610

Software

- Review
- Repository
- Archive

Editor: Patrick Diehl

Reviewers:

- @vijaysm
- @tiburoch

Submitted: 25 October 2024
Published: 13 August 2025

Statement of Need

Modeling fracture in materials and structures holds immense importance in our efforts to understand how materials fail and hence design more fracture-resistant structures. Among the several theories developed over the last decades, continuum damage mechanics (CDM) studies the behavior of cracks in materials and structures from the viewpoint of continuous stiffness degradation as the crack propagates inside the domain (Lemaitre, 2012). CDM simulations are commonly implemented using the finite element method, however the associated computational cost is notoriously elevated. With parallel computing becoming increasingly widespread, it presents an efficient strategy for reducing the high computational cost associated with CDM simulations. This open source code utilizes parallelization techniques for MATLAB in order to significantly accelerate CDM simulations. Building upon previous work by the authors (Saji et al., 2024), we develop a parallel MATLAB code and demonstrate the additional efficiency over its serial counterpart. The code is geared for quasi-brittle materials, and it is implemented with two relevant damage models (Mazars' model (Mazars, 1986) and Geers' model (Geers et al., 1998)). Both the unified arc-length (UAL) and Newton–Raphson solvers can be used.

There are several FEM libraries with parallel capabilities publicly available, such as FEniCS (The FEniCS Project, 2024), OOFEM (Öhman et al., 2020), Akantu (Richart et al., 2024), OpenSees (Pacific Earthquake Engineering Research Center, 2025), and deal.II (Arndt et al., 2021), but they use parallelization strategies such as domain decomposition and/or Message Passing Interface (MPI) and require coding expertise from the user. Our code is unique in its user-accessibility, given that it is written in MATLAB which many users are familiar with, while also implementing parallelization techniques for complex continuum damage mechanics simulations. Also, it features the implementation of a newly developed and robust Unified arc-length solver (UAL) developed in Saji et al. (2024), which has demonstrated its superior performance against the force-controlled arc-length (FAL) and Newton–Raphson (NR) solvers both in terms of accuracy and time efficiency. For example, in a 1D bar problem modeled using the non-local gradient damage and a weakened region in the middle (modulus of elasticity is reduced by a factor of two), the UAL solver models the entire non-linear equilibrium path in 0.98 seconds with 97 increments, while the FAL and NR solvers require one to two orders of magnitude more time and increments to trace the same path with a more relaxed convergence tolerance (Saji et al., 2024). The UAL solver can trace equilibrium paths with snap-backs, making it a suitable solver for a wide range of scenarios, which the NR solver cannot capture. Furthermore, the UAL solver is not constrained by the need for small step sizes and relaxed convergence tolerances that limit the usage of the FAL solver.

Methodology and Software Implementation

A flowchart of the code's general functionality is shown in Figure 1. The function on which we focus is the `func_globalstiffness()` function, which calculates and assembles the global stiffness matrix, force vectors, and residual vector. The function also calculates the projection matrices required for contour plotting once the numerical analysis of each load increment is complete.

In MATLAB, parallelization relies on the Parallel Computing Toolbox, in which the user can generate a parallel pool of workers. This parallelization is illustrated in Figure 2. In the "solver" mode of the code, the function will assemble global matrices, so we implement the single program multiple data (SPMD) construct in MATLAB since ordered execution and data sharing between workers is needed. In the "plotting properties" mode, we use "parfor" as the loop body is independent, and iterations can be executed in any order.

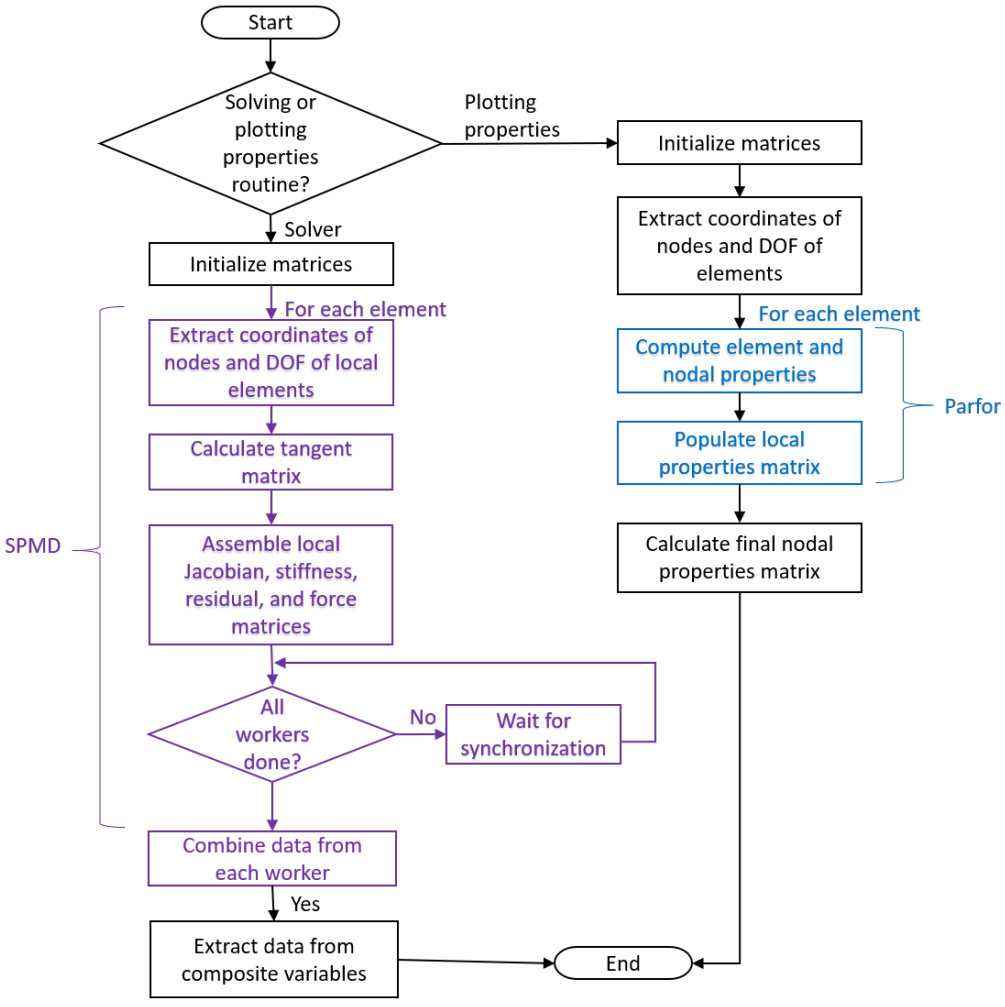

Figure 1: Flowchart of the general code structure

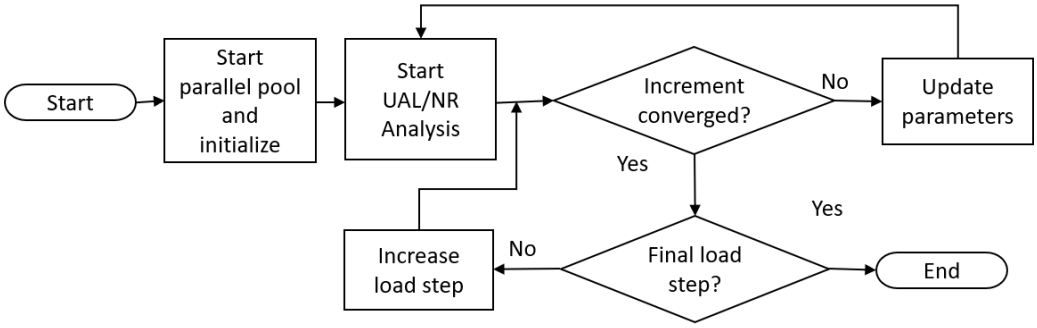

Figure 2: Flowchart of the parallel `func_globalstiffness()` function which includes solving and plotting properties routines

Testing and Results

A symmetric single notch tension (SSNT) problem is used to test the code's effectiveness following Saji et al. (2024) and Pantidis & Mobasher (2023). In Figure 3 we present the total runtime of a fine (10201 elements) mesh on the New York University Abu Dhabi (NYUAD) High Performance Computing (HPC) cluster, for serial and parallel implementation. From this figure, it is clear that the parallelization exhibits significant cost improvement, with a factor of three reduction in the total runtime of the code for a serial computation to parallel with 8, 16, or 32 threads. Above 32 threads, Figure 3 reveals that additional parallel resources do not provide further improvement for the size of this problem. When pursuing faster runtimes with parallel computing, the overhead costs should be weighed against potential improvement in speed.

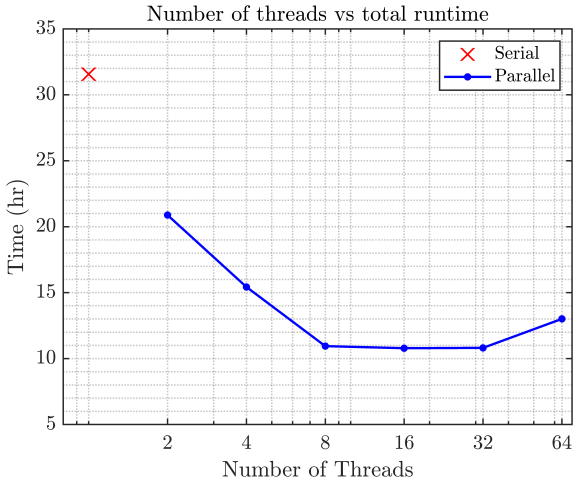

Figure 3: Total runtime using fine mesh on HPC with the UAL solver. The number of threads is represented on a logarithmic scale for clarity. Additional parallel resources reduce the runtime of a fine mesh on the HPC cluster.

Acknowledgements

This work was partially supported by the Sand Hazards and Opportunities for Resilience, Energy, and Sustainability (SHORES) Center, funded by Tamkeen under the NYUAD Research Institute. The authors would also like to acknowledge the support of the NYUAD Center for Research Computing for providing resources, services, and staff expertise.

References

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

EPFL LSMS. (2024). *Akantu: A finite element library for complex multiphysics simulations*. https://akantu.ch/

Geers, M. G. D., Borst, R. de, Brekelmans, W. A. M., & Peerlings, R. H. J. (1998). Strain-based transient-gradient damage model for failure analyses. *Computer Methods in Applied Mechanics and Engineering*. https://doi.org/10.1016/S0045-7825(98)80011-X

Lemaitre, J. (2012). *A course on damage mechanics*. Springer. https://doi.org/10.1007/978-3-642-18255-6

Mazars, J. (1986). A description of micro-and macroscale damage of concrete structures. *Engineering Fracture Mechanics*, *25*(5-6), 729–737. https://doi.org/10.1016/0013-7944(86)90036-6

Öhman, M., Patzak, B., Brouzoulis, J., Smilauer, V., milanjirasek, Grassl, P., graspel, feymark, CarlSandstrom, nitramkaroh, MartinFagerstrom, pedroskop, Främby, J., eudoxos, Sciegaj, A., editaDvorakova, Sulc, S., Stránský, J., karelmikes, … vmonkey. (2020). *Oofem/oofem: OOFEM, version 2.5* (Version v2.5). Zenodo. https://doi.org/10.5281/zenodo.4339630

Pacific Earthquake Engineering Research Center. (2025). *OpenSees: Open system for earthquake engineering simulation*. https://opensees.berkeley.edu/

Pantidis, P., & Mobasher, M. E. (2023). Integrated finite element neural network (i-FENN) for non-local continuum damage mechanics. *Computational Methods in Applied Mechanics and Engineering*, *404*, 115766. https://doi.org/10.1016/j.cma.2022.115766

Richart, N., Anciaux, G., Gallyamov, E., Frérot, L., Kammer, D., Pundir, M., Vocialta, M., Ramos, A. C., Corrado, M., Müller, P., Barras, F., Zhang, S., Ferry, R., Durussel, S., & Molinari, J.-F. (2024). Akantu: An HPC finite-element library for contact and dynamic fracture simulations. *Journal of Open Source Software*, *9*(94), 5253. https://doi.org/10.21105/joss.05253

Saji, R. P., Pantidis, P., & Mobasher, M. E. (2024). A new unified arc-length method for damage mechanics problems. *Computational Mechanics*. https://doi.org/10.1007/s00466-024-02473-5

The FEniCS Project. (2024). *FEniCS project*. https://fenicsproject.org/