SoftwareX 26 (2024) 101700

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

PeriLab — Peridynamic Laboratory

Christian Willberg [a,b,*], Jan-Timo Hesse [a], Anna Pernatii [c]

[a] Magdeburg-Stendal University of Applied Science, Breitscheidstr. 2, 39114 Magdeburg, Germany
[b] German Aerospace Center, Lilienthalplatz 7, 38126 Braunschweig, Germany
[c] Otto von Guericke University Magdeburg, Universitätsplatz 2, 39106 Magdeburg, Germany

ARTICLE INFO

Dataset link: https://zenodo.org/records/10229386

Keywords:
Peridynamics
Fracture
HPC
Computational science
MPI
Julia

ABSTRACT

This paper introduces `PeriLab`, a modern Peridynamics solver developed in the Julia programming language. Emphasizing easy installation, usability, and implementation of new features, the code's structure is detailed, accompanied by illustrative examples illustrating some of the code's core functionality. The fully Message Passing Interface (MPI) parallelized code undergoes a separate benchmark problem with two million degrees of freedom, revealing large scale capabilities and analyzing the communication cost occurring in such analysis. The paper highlights key considerations for the adoption of Peridynamics, including the need for a straightforward installation process, user-friendly interfaces, efficient research code development, and well-documented as well as tested functionalities. Overcoming these challenges is crucial for Peridynamics' widespread acceptance in engineering applications, and `PeriLab` serves as a valuable contribution to addressing these issues.

Code metadata

Current code version	v1.0.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-24-00031
Permanent link to Reproducible Capsule	https://doi.org/10.5281/zenodo.10229386
Legal Code License	BSD 3-Clause License
Code versioning system used	git
Software code languages, tools, and services used	Julia
Compilation requirements, operating environments & dependencies	ArgParse 1.1.4, CSV 0.10.11, DataFrames 1.6.1, Exodus 0.11.0, JSON3 1.13.2, LoggingExtras 1.0.3, MPI 0.20.19, NearestNeighbors 0.4.13, ProgressBars 1.5.1, Rotations 1.6.1, TensorOperations 4.0.7, Tensors 1.16.1, TimerOutputs 0.5.23, YAML 0.4.9, ZipArchives 1.1.1, julia 1
If available Link to developer documentation/manual	https://dlr-perihub.gitlab.io/PeriLab.jl/
Support email for questions	christian.willberg@h2.de; jan-timo.hesse@dlr.de

Software metadata

Current software version	v1.0.0
Permanent link to executables of this version	https://juliahub.com/ui/Packages/General/PeriLab
Permanent link to Reproducible Capsule	
Legal Software License	BSD 3-Clause License
Computing platforms/Operating Systems	Linux
Installation requirements & dependencies	ArgParse 1.1.4, CSV 0.10.11, DataFrames 1.6.1, Exodus 0.11.0, JSON3 1.13.2, LoggingExtras 1.0.3, MPI 0.20.19, NearestNeighbors 0.4.13, ProgressBars 1.5.1, Rotations 1.6.1, TensorOperations 4.0.7, Tensors 1.16.1, TimerOutputs 0.5.23, YAML 0.4.9, ZipArchives 1.1.1, julia 1
User Guide	https://perihub.github.io/PeriLab.jl/stable/man/getting_started/
Support email for questions	christian.willberg@h2.de; jan-timo.hesse@dlr.de

* Corresponding author.
E-mail addresses: christian.willberg@h2.de (Christian Willberg), jan-timo.hesse@dlr.de (Jan-Timo Hesse), anna.pernatii@ovgu.de (Anna Pernatii).

https://doi.org/10.1016/j.softx.2024.101700
Received 16 January 2024; Received in revised form 11 March 2024; Accepted 19 March 2024
Available online 4 April 2024

Glossary

BB	Bond-based
BSD	Berkeley Software Distribution
CSV	Comma-separated values
DCB	Double cantilever beam
DOF	Degrees of freedom
FEM	Finite Element Method
HPC	High-performance computing
MPI	Message Passing Interface
NOSB	Non-ordinary state-based
OSB	Ordinary state-based
PD	Peridynamics

1. Motivation and significance

In engineering applications, the mechanical behavior of materials and structures is typically modeled using classical continuum mechanics. Because of its limitations in describing fracture problems an alternative modeling approach was proposed in the early 2000's in the form of PD [1,2], thought of as a continuum version of molecular dynamics. In recent years a lot of research in PD took place summarized in various review papers and books [3–11].

Mesh-free particle-based methods are most commonly used for the numerical approximation of PD equations, Table 1 provides an overview of current PD software developments. In addition to this list, many local research groups are also developing their own software. Most of the listed tools are problem-specific or developed by a single person. `Peridigm` and `EMU` are the most sophisticated software tools. `EMU` is not open source, but researchers can request the code. However, because the code development is done at the Sandia National labs, the code is officially not usable in many countries because the terms of use lead to a violation of the "Treaty on the Non-Proliferation of Nuclear Weapons".

`Peridigm` on the other hand is an open-source tool. It includes multiple material laws, simple damage models, and all three PD modeling approaches (Bond-based (BB), Ordinary state-based (OSB) and Non-ordinary state-based (NOSB)). `Peridigm` is usable under a Berkeley Software Distribution (BSD) license. Currently, there is some irregular development ongoing. It provides multiple ways of model input and provides Paraview[1] readable output. The whole code allows the analysis of large-scale problems using MPI parallelization. A lot of research has been done utilizing the software [12–16].

One significant challenge still remains: How can PD gain widespread adoption? In the engineering field, classical continuum mechanics is extensively utilized through the Finite Element Method (FEM). A plethora of both commercial and non-commercial software tools are readily available for this purpose. However, when it comes to PD, the situation is quite different. While `Peridigm` stands as one of the most advanced options, it offers an open-source code with a rich set of functionalities. Nevertheless, the implementation process is quite demanding, and the installation can be both arduous and time-consuming.

To implement even the simplest material law in `Peridigm`, you are required to modify a minimum of five files. Additionally, by introducing new files and directories to the project, you must contend with CMake scripts and the associated complexities.

Furthermore, it is worth noting that `Peridigm` is only partially maintained. This results in valuable research discoveries being underutilized. Consequently, PD struggles to find applications beyond specific niche problems. There is an evident and pressing need for software that seamlessly combines functionality, user-friendly installation, and easy integration.

This paper introduces a solution called `PeriLab`, which is built using Julia and incorporates MPI to manage a large number of degrees of freedom [17]. The objectives for `PeriLab` encompass:

- Installation: The installation process must be straightforward and not pose a barrier to using the tool.
- Usability: The tool should offer simplicity and flexibility to cater to various user needs.
- Implementation & Extension: It should be designed in a way that facilitates efficient research code development and extension.
- Functionality: All existing functionalities must be well-documented and thoroughly tested.

2. Software description

The main focus of `PeriLab` is the easy installation, implementation and extension of PD methods in a parallelizable code. For this purpose the Julia language was chosen. It is a modern language specifically designed to create high performance code for large scale problems and has some key advantages compared to C, Fortran or C++. Julia is comparable fast and has a modern packaging system. This allows an easy installation and compiling of code. Also, it allows the easy use of packages and external functionalities. This modern language minimizes the length of the code, simplifies the software testing and allow the developers far more comfort and less errors.

The installation part of `PeriLab` is realized via this modern packaging system. The software project dependencies will be automatically downloaded. This reduces the installation time to minutes.

2.1. Software architecture

For implementing, e.g. a new material model in `PeriLab`, only one file is created with a provided template and placed in the material directory or sub directory. The integration is done by using the meta programming feature provided by Julia. Fig. 1 shows the workflow. The software searches for available modules within the project folder. Within the yaml definition the model name is specified. This has to be defined within the module in a certain function. If the name of any module and the definition are equal, the module is included into the code base and the software will be compiled again. The whole procedure is done automatically and works for the basic models (additive, damage, material and thermal). Extensions are possible using the same strategy.

After starting the main.jl with a batch command the yaml file is expected, cf. Fig. 2. This file is parsed in the IO module. This module is used to read all files needed for model creation and to distribute all the data in the datamanager and onto different processors if MPI is specified. This was a design decision, because the communication with the MPI.jl package is comparable to its C++ counterpart [32] and more efficient for large scale problems than working with shared memory. This is supported by the findings of Diehl et al. [33]. They modeled a simple 1D problem and used Julia threads which resulted in a deficiency compared to C++ MPI solutions. For the currently implemented solver the step width is determined and all fields for the different solver options are initialized. Also the physics models are initialized if they are needed. An Exodus output file and / or the Comma-separated values (CSV) file export will then be created. Afterwards the solving process starts. The solver runs and within the physical models are evaluated. The results will be written in parallel and can be accessed while the simulation is running. At the end of this process the result files are merged if needed, and the program stops. The results are than accessible via Paraview or another exodus reader.

[1] https://www.paraview.org/

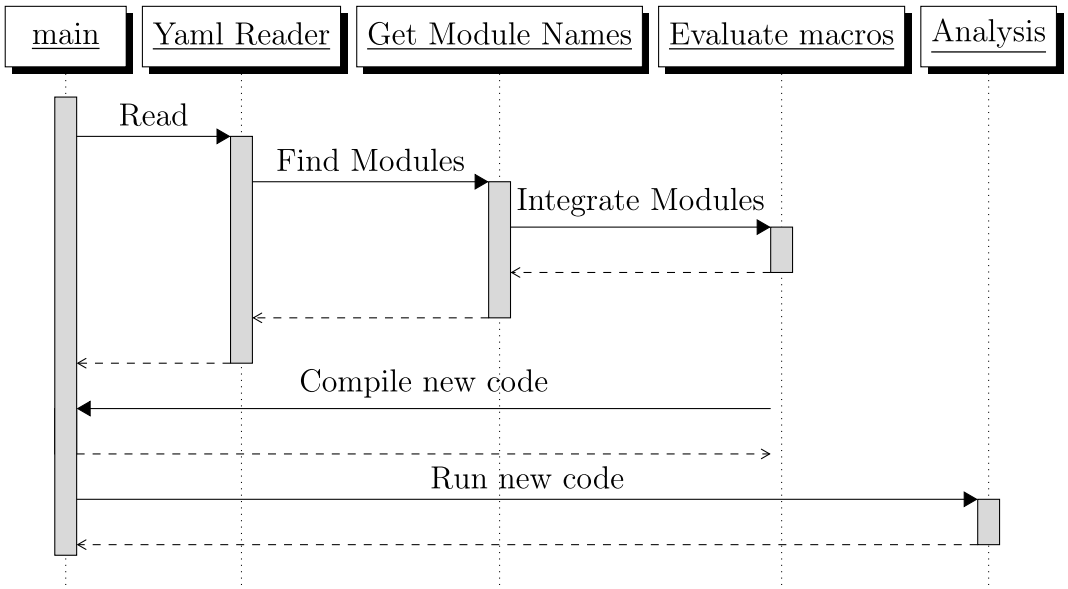


Fig. 1. Concept of macro usage to include modules in PeriLab.

Table 1
Overview about PD software tools.

Name	Focus	Language	Maturity level	License	References
PeriPy	bond-based	Python	Medium	MIT	[18]
PeriDem	Digital image correlation	C++, Python	Low		[19]
PeriPyDIC	2D	Python	Low	BSL	[20]
PeriPyVFM	Virtual field	Python	Low	GPLv3.0	[21]
BB_PD	2D, bond-based	Matlab, C	Low-medium	No defined	[22]
LAMMPS	Molecular dynamics, bond-based	Python	Low	GPLv2	[23]
PeriFlakes	2D, coupling FEM	Python, C	Low	GPLv3.0	[24]
NLMech	2D	C++	Medium		[19]
Relation-Based Software	Ordinary state-based, bond-based	C++	Medium	MIT	[25]
EMU	3D, multiphysics, large scale problems	Fortran	High	Closed source	[26]
`Peridigm`	3D, multiphysics, large scale problems	C, C++	High	BSD	[27–29]
PeriHub	2D & 3D, Extension of `Peridigm`	Python, JS	High	BSD	[30,31]

2.2. Software functionalities

The system's design prioritizes ease of development and scalability. Consequently, structs have been avoided in favor of a data manager. It plays a pivotal role in memory allocation and ensuring the assignment of unique names to variables or fields.

Listing 1: Template for material modules

```
module Material_template
export compute_forces
export material_name

function material_name()
  return "Material Template"
end

function compute_forces(datamanager, nodes,
material_parameter, time, dt)

  return datamanager
end
end
```

As discussed before, integrating custom models in `Peridigm` can be challenging. You need to have a deep understanding of how the compiler works and where to call your functions. In the context of `PeriLab`, creating a new material model involves copying your module file into the 'material' folder, effectively creating a new material model using the template shown in Listing 1. You have the flexibility to choose your module name. The material name serves as an identifier within your input deck. When you use this name as a material model option in your input deck, this module is activated. It is also possible to combine material models via +, e.g. to create an elastic plastic model.

3. Illustrative examples

`PeriLab` provides basic features. Within this paper three examples of those features are shown. All of them were compared with FEM or `Peridigm` and were in good agreement. The dogbone model highlights the deformation of a one dimensional tensile test applying OSB PD, cf. Fig. 3. The undeformed state is shown with the black dots. A DCB model shows a correspondence elastic model coupled with a energy based damage criterion [30], cf. Fig. 4. And the temperature distribution of a rectangular plate is visible in Fig. 5. Because parallelization is an important part a simple benchmark of the current not optimized version 1.0.0 was done.

A squared grid was defined with 10, 100, 1000 numbers of points np in x- and y-directions with a structured mesh with a point distance of dx in x- and y-direction. The size of the neighborhood was varied between 3 and $5dx$. To avoid potential solver issues, no loads were applied to the benchmark problem. The number of time integration steps was equal for all discretizations. To measure the time and memory consumption the `TimeOutputs.jl` package was used.

In Fig. 6 the results show that for lower discretizations, e.g. 200 Degrees of freedom (DOF), the single core usage is already quite efficient and due to the MPI communication overhead adding more processes lead to slower simulations. By utilizing 100 processes the

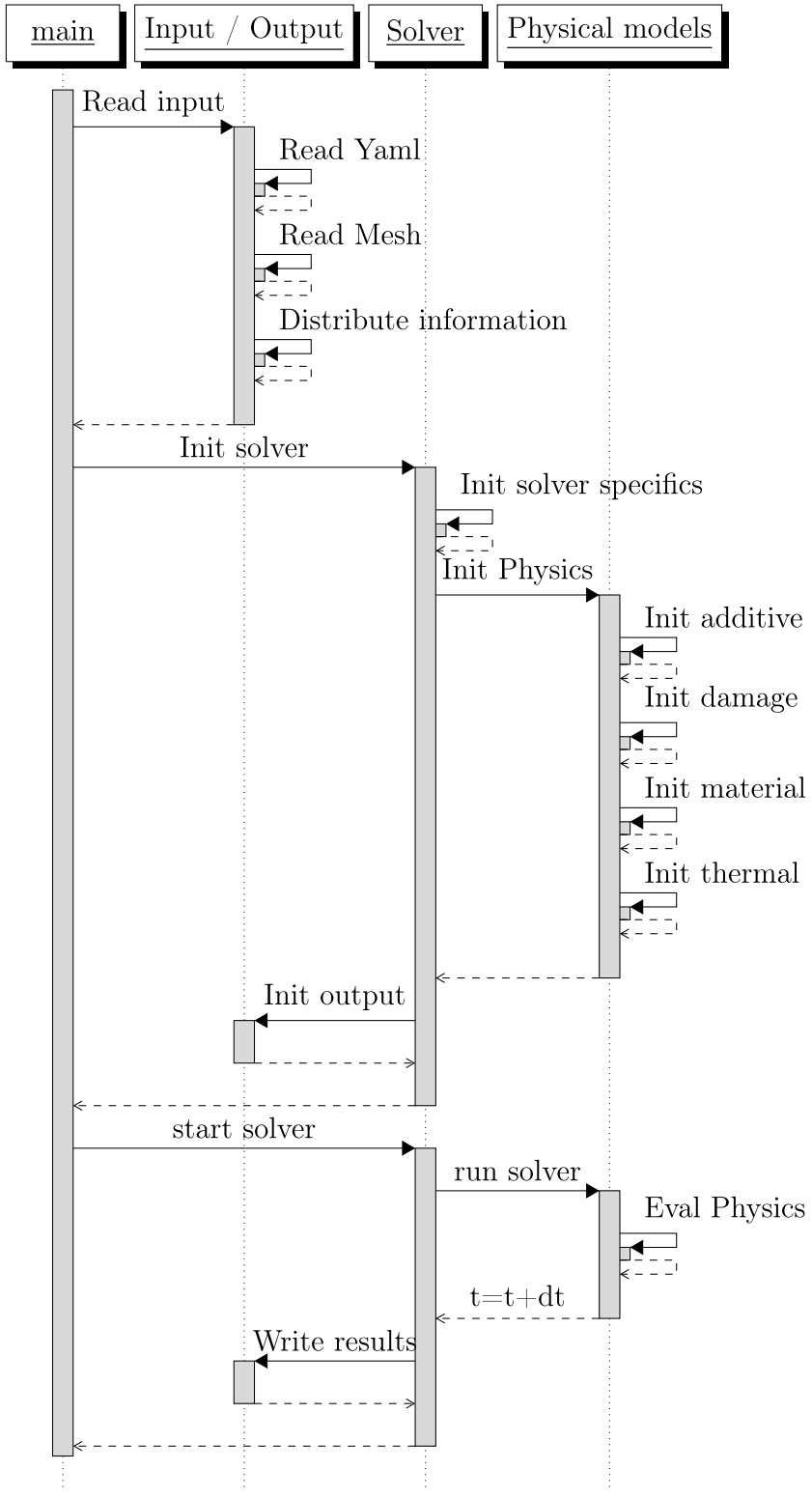


Fig. 2. Program flow of PeriLab.

Listing 2: Automated integration of modules

```
function create_module_specifics(name::String, module_list::Vector{Any},
specifics::Dict{String,String}, values::Tuple)
for m in module_list
    parse_statement =
        "module_name=" *m["Module Name"] * "." * specifics["Name"]* "()"
    if eval(Meta.parse(parse_statement)) == name
        parse_statement = m["Module Name"] *"." * specifics["Call Function"]
        function_call =eval(Meta.parse(parse_statement))
        return function_call(values...)
    end
end
end
```

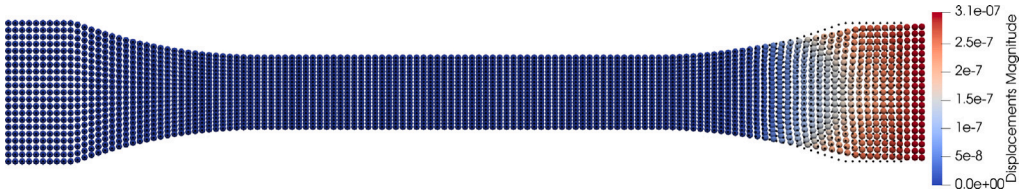
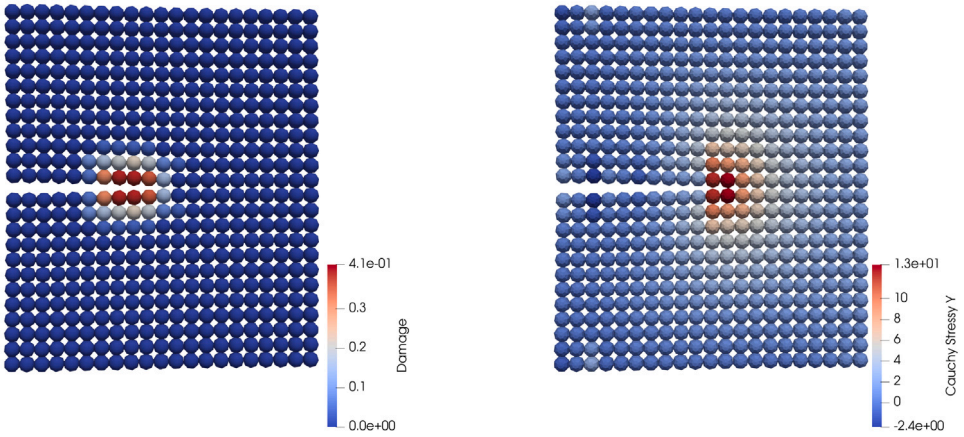


Fig. 3. Displacement plot of a dogbone model, where the black points are the undeformed state and the colored points the scaled deformation state.



(a) Color plot of damage index (b) Color plot of Cauchy stresses

Fig. 4. Model of a DCB virtual experiment scaled displacements of factor 200.

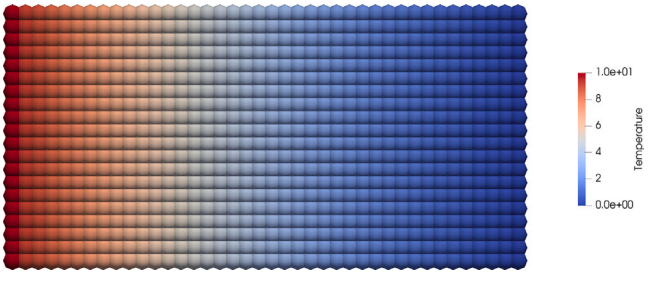


Fig. 5. Linear temperature distribution for $\tau = 10\,\mathrm{K}$ at the left side and $\tau = 0\,\mathrm{K}$ at the right side.

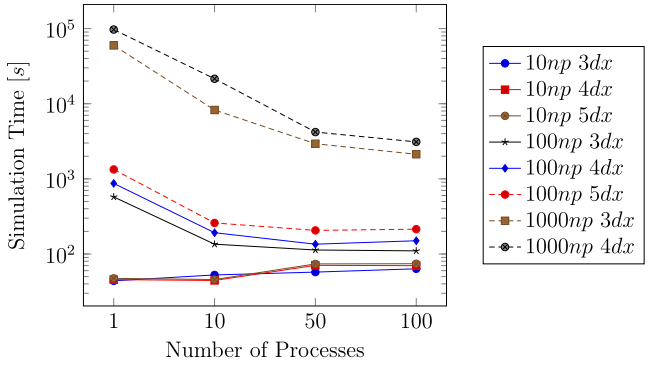


Fig. 6. Simulation time of different meshes depending on the number of processes.

running time of the two million degrees of freedom example will be reduced by a factor of 30. Overall the benchmark results acknowledge the correct MPI implementations and the future software potential.

4. Conclusions

The paper introduces `PeriLab`, showcasing its fundamental functionalities successfully. This encompasses damage models, material, thermal, and additive models. The integration of additional models was also presented, demonstrating a uniform procedure across all material types. To illustrate parallelization capabilities, a benchmark was conducted, revealing a significant acceleration in computational time. The installation of the software on an High-performance computing (HPC)

cluster proved to be straightforward, requiring only a minimal time investment of minutes.

CRediT authorship contribution statement

Christian Willberg: Writing – review & editing, Writing – original draft, Software, Funding acquisition. **Jan-Timo Hesse:** Writing – review & editing, Writing – original draft, Software. **Anna Pernatii:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

https://zenodo.org/records/10229386.

Acknowledgments

The authors like to acknowledge the helpful support from the `Exodus.jl` package owner https://github.com/cmhamel.

The work was funded by the German Research Foundation funded project: "Gekoppelte Peridynamik-Finite-Elemente-Simulationen zur Schädigungsanalyse von Faserverbundstrukturen" Grant number: WI 4835/5-1 and the M-ERA.NET funded project Exploring Multi-Method Analysis of composite structures and joints under consideration of uncertainties engineering and processing (EMMA)

This measure is co-financed with tax funds on the basis of the budget passed by the Saxon state parliament. Grant number: 3028223.

References

[1] Silling SA. Reformulation of elasticity theory for discontinuities and long-range forces. J Mech Phys Solids 2000;48(1):175–209. http://dx.doi.org/10.1016/S0022-5096(99)00029-0.

[2] Silling SA, Askari E. A meshfree method based on the peridynamic model of solid mechanics. Comput Struct 2005;83(17–18):1526–35. http://dx.doi.org/10.1016/j.compstruc.2004.11.026.

[3] Bobaru F, Foster JT, Geubelle PH, Silling SA. Handbook of peridynamic modeling. Advances in applied mathematics, CRC Press; 2016.

[4] Dias JP, Bazani MA, Paschoalini AT, Barbanti L. A review of crack propagation modeling using peridynamics. In: Ekwaro-Osire S, Gonçalves AC, Alemayehu FM, editors. Probabilistic prognostics and health management of energy systems. Cham: Springer International Publishing; 2017, p. 111–26. http://dx.doi.org/10.1007/978-3-319-55852-3_7.

[5] Javili A, Morasata R, Oterkus E, Oterkus S. Peridynamics review. Math Mech Solids 2018. http://dx.doi.org/10.1177/1081286518803411.

[6] Diehl P, Prudhomme S, Lévesque M. A review of benchmark experiments for the validation of peridynamics models. J Peridyn Nonlocal Model 2019. http://dx.doi.org/10.1007/s42102-018-0004-x.

[7] Han D, Zhang Y, Wang Q, Lu W, Jia B. The review of the bond-based peridynamics modeling. J Micromech Mol Phys 2019;04(01):1830001. http://dx.doi.org/10.1142/S2424913018300013.

[8] Brighenti R, Zeleke MA, Ageze MB. A review of peridynamics (PD) theory of diffusion based problems. J Eng 2021;20. http://dx.doi.org/10.1155/2021/7782326.

[9] D'Elia M, Li X, Seleson P, Tian X, Yu Y. A review of local-to-nonlocal coupling methods in nonlocal diffusion and nonlocal mechanics. J Peridyn Nonlocal Model 2021. http://dx.doi.org/10.1007/s42102-020-00038-7.

[10] Isiet M, Mišković I, Mišković S. Review of peridynamic modelling of material failure and damage due to impact. Int J Impact Eng 2021;147:103740. http://dx.doi.org/10.1016/j.ijimpeng.2020.103740, URL https://www.sciencedirect.com/science/article/pii/S0734743X20308101.

[11] Diehl P, Lipton R, Wick T, Tyagi M. A comparative review of peridynamics and phase-field models for engineering fracture mechanics. Comput Mech 2022;69:1259–93. http://dx.doi.org/10.1007/s00466-022-02147-0.

[12] Bobaru F, Duangpanya M. The peridynamic formulation for transient heat conduction. Int J Heat Mass Transfer 2010;53(19–20):4047–59. http://dx.doi.org/10.1016/j.ijheatmasstransfer.2010.05.024.

[13] Littlewood DJ, Parks ML, Mitchell JA, Silling SA. The peridigm framework for peridynamic simulations. In: 12th U.S. national congress on computational mechanics, 22-25 July 2013, Raleigh, North Carolina, USA. (SAND2013-5927C). Albuquerque, New Mexico 87185 and Livermore, California 94550, USA: Sandia National Laboratories; 2013, URL https://cfwebprod.sandia.gov/cfdocs/CompResearch/docs/Littlewood_USNCCM2013.pdf.

[14] Mitchell JA, Silling SA, Littlewood DJ. A position-aware linear solid constitutive model for peridynamics. Mech Mater Struct 2015;10(5):539–57. http://dx.doi.org/10.2140/jomms.2015.10.539.

[15] Hoppe L. Numerical simulation of fiber-matrix debonding in single fiber pull-out tests. GAMM Arch Stud 2020;2(1):21–35. http://dx.doi.org/10.14464/gammas.v2i1.437.

[16] Li X, Ye H, Zhang J. Redesigning peridigm on SIMT accelerators for high-performance peridynamics simulations. In: 2021 IEEE international parallel and distributed processing symposium. 2021, p. 433–43. http://dx.doi.org/10.1109/IPDPS49936.2021.00052.

[17] Willberg C, Hesse J-T. PeriLab - peridynamic laboratory v1.0. 2023, http://dx.doi.org/10.5281/zenodo.10229386, GitLab repository. URL https://gitlab.com/dlr-perihub/perilab.

[18] Boys B, Dodwell T, Hobbs M, Girolami M. PeriPy - A high performance OpenCL peridynamics package. Comput Methods Appl Mech Engrg 2021;386:114085. http://dx.doi.org/10.1016/j.cma.2021.114085.

[19] Jha PK, Desai PS, Bhattacharya D, Lipton R. Peridynamics-based discrete element method (PeriDEM) model of granular systems involving breakage of arbitrarily shaped particles. J Mech Phys Solids 2021;151:104376. http://dx.doi.org/10.1016/j.jmps.2021.104376.

[20] Delorme R, Tabiai I, Laberge Lebel L, Lévesque M. Generalization of the ordinary state-based peridynamic model for isotropic linear viscoelasticity. Mech Time-Depend Mater 2017;21(4):549–75. http://dx.doi.org/10.1007/s11043-017-9342-3.

[21] Delorme R, Diehl P, Tabiai I, Lebel LL, Lévesque M. Extracting constitutive mechanical parameters in linear elasticity using the virtual fields method within the ordinary state-based peridynamic framework. J Peridyn Nonlocal Model 2020;2(2):111–35. http://dx.doi.org/10.1007/s42102-019-00025-7.

[22] Hobbs M, Hattori G, Orr J. Predicting shear failure in reinforced concrete members using a three-dimensional peridynamic framework. Comput Struct 2022;258:106682. http://dx.doi.org/10.1016/j.compstruc.2021.106682.

[23] Lehoucq RB, Silling SA, Plimpton SJ, Parks ML. Peridynamics with LAMMPS: a user guide. Tech. rep., 2008, http://dx.doi.org/10.2172/959309.

[24] Queiruga A-F, Moridis G. Numerical experiments on the convergence properties of state-based peridynamic laws and influence functions in two-dimensional problems. Comput Methods Appl Mech Engrg 2017;322:97–122. http://dx.doi.org/10.1016/j.cma.2017.04.016.

[25] Jenabidehkordi A, Fu X, Rabczuk T. An open source peridynamics code for dynamic fracture in homogeneous and heterogeneous materials. Adv Eng Softw 2022;168:103124. http://dx.doi.org/10.1016/j.advengsoft.2022.103124.

[26] Silling SA. Stability of peridynamic correspondence material models and their particle discretizations. Comput Methods Appl Mech Engrg 2017;332:42–57. http://dx.doi.org/10.1016/j.cma.2017.03.043.

[27] Parks M, Littlewood D, Mitchell J, Silling S. Peridigm users' guide. Tech. rep., Report SAND2012-7800, Sandia National Laboratories; 2012.

[28] Rädel M, Willberg C. PeriDoX. 2018, http://dx.doi.org/10.5281/zenodo.1403015, GitHub repository. URL https://github.com/PeriDoX/PeriDoX.

[29] Littlewood DJ, Parks ML, Foster JT, Mitchell JA, Diehl P. The peridigm meshfree peridynamics code. J Peridyn Nonlocal Model 2023. http://dx.doi.org/10.1007/s42102-023-00100-0.

[30] Willberg C, Hesse J-T, Heinecke F. Peridynamic simulation of a mixed-mode fracture experiment in PMMA utilizing an adaptive-time stepping for an explicit solver. J Peridyn Nonlocal Model 2022. http://dx.doi.org/10.1007/s42102-021-00079-6.

[31] Willberg C, Hesse J-T, Garbade M, Rädel M, Heinecke F, Schuster A, et al. A user material interface for the Peridynamic Peridigm framework. SoftwareX 2023. http://dx.doi.org/10.1016/j.softx.2023.101322.

[32] Byrne S, Wilcox LC, Churavy V. MPI.jl: Julia bindings for the Message Passing Interface. In: Proceedings of the JuliaCon conferences. Vol. 1, (1):The Open Journal; 2021, p. 68. http://dx.doi.org/10.21105/jcon.00068.

[33] Diehl P, Brandt SR, Morris M, Gupta N, Kaiser H. Benchmarking the parallel 1D heat equation solver in chapel, Charm++, C++, HPX, Go, Julia, Python, Rust, Swift, and Java. 2023, arXiv:2307.01117.