SANDIA REPORT

2011-8523
Unlimited Release
Printed November 2011

Supersedes SAND2010-5549
Dated August 2010

Peridynamics with LAMMPS: A User Guide v0.3 Beta

Michael L. Parks, Pablo Seleson, Steven J. Plimpton, Stewart A. Silling, and Richard B. Lehoucq

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

2011-8523
Unlimited Release
Printed November 2011

Supersedes SAND2010-5549
dated August 2010

Peridynamics with LAMMPS: A User Guide v0.3 Beta

Michael L. Parks [*] Pablo Seleson [§]
Steven J. Plimpton [†] Stewart A. Silling [*]
Richard B. Lehoucq [*]

[*]Multiphysics Simulation Technologies
[†]Scalable Algorithms
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1320

[§]Institute for Computational Engineering and Sciences
The University of Texas at Austin 1 University Station C0200 Austin, TX 78712

Abstract

Peridynamics is a nonlocal extension of classical continuum mechanics. The discrete peridynamic model has the same computational structure as a molecular dynamics model. This document provides a brief overview of the peridynamic model of a continuum, then discusses how the peridynamic model is discretized within LAMMPS. An example problem is also included.

Acknowledgments

The lead author acknowledges helpful feedback and bug fixes from Axel Kohlmeyer at Temple University and Peter Gordon and Rohan Panchadhara at the ExxonMobil Research and Engineering Company.

Contents

1 Introduction **7**
1.1 Quick Start Guide 7
1.2 Typographical Conventions 7

2 Getting Started **8**
2.1 Building the Peridynamic Module within LAMMPS 8
2.2 Input Script Basics 8

3 Peridynamic Model of a Continuum **11**
3.1 Basic Notation 11
3.2 Linear Peridynamic Solid (LPS) Model 12
3.3 Prototype Microelastic Brittle (PMB) Model 13
3.4 Microplastic (MP) Model 13
3.5 Damage 14

4 Discrete Peridynamic Model and LAMMPS Implementation **16**
4.1 Spatial Discretization 16
4.2 Short-Range Forces 16
4.3 Modification to the Particle Volume 17
4.4 Temporal Discretization 17
4.5 Breaking Bonds 18
4.6 Damage Compute Style 18
4.7 Bond Family Compute Style 19
4.8 Visualizing Simulation Results 20
4.9 Pitfalls 21
4.10 Bugs 22
4.11 Modifying and Extending the Peridynamic Module 22

5 A Numerical Example **23**
5.1 Problem Description and Setup 23
5.2 The Projectile 23
5.3 Writing the LAMMPS Input File 24
5.4 Numerical Results and Discussion 25

References **27**

Appendix

Appendix A: LPS Pseudocode **28**

Appendix B: PMB Pseudocode **30**

Appendix C: MP Pseudocode **31**

Figures

1 Bond force vs. bond stretch for microplastic material. 14
2 Diagram showing horizon of a particular particle. The volume associated with particles near the boundary of the horizon is not completely contained within the horizon. 18
3 Target during (a) and after (b,c) impact. 26

Tables

1 Notational conventions. 7

1 Introduction

This document details the implementation of a discrete peridynamic model within the LAMMPS molecular dynamic code, as described in the original article [5].

In §2 we discuss how to build the peridynamic module within LAMMPS, and discuss basic requirements for input scripts to use the peridynamic module. In §3 we overview the relevant portions of the peridynamic model of a continuum. In §4 we discuss the discretization of the peridynamic model and its LAMMPS implementation. Finally, in §5, we discuss a LAMMPS simulation of a specific numerical experiment described in [11].

1.1 Quick Start Guide

For those who hate reading users' guides[1], please try the following:

1. Download LAMMPS from `http://lammps.sandia.gov` and untar the source.
2. In the LAMMPS `src/` directory do `make yes-peri` followed by `make <your platform>` (for example, `make g++`).
3. In the LAMMPS `examples/peri` directory, run the example input script (for example, `lmp_g++ < in.peri`).
4. Follow instructions in §4.8 to visualize results.[2]

1.2 Typographical Conventions

Our typographical conventions are found in Table 1.

Table 1. Notational conventions.

Notation	Example	Description
`Verbatim text`	`make g++`	Text to be typed at your command prompt
<text in angle brackets>	<your platform>	User specified statement
Bold lowercase letter	$\mathbf{x}$, $\boldsymbol{\xi}$	A vector in $\mathbb{R}^3$
Non-bold letter	K, α	A scalar in $\mathbb{R}$
Underlined lowercase letter	$\underline{t}$, $\underline{\omega}$	Scalar state (see §3.1)
Underlined bold uppercase letter	$\underline{\mathbf{T}}$, $\underline{\mathbf{M}}$	Vector state (see §3.1)

Finally, note all norms $\|\cdot\|$ are taken to be the 2-norm, $\|\cdot\|_2$.

[1] Congratulations on getting this far!

[2] For a more meaningful example, try running the input script in Algorithm 3 on page 25.

2 Getting Started

We assume that you already have a working LAMMPS installlation. For more on downloading and building LAMMPS, see `http://lammps.sandia.gov`. This document only provides information related to the peridynamic module within LAMMPS. For questions regarding the usage of LAMMPS, please see the LAMMPS documentation.

2.1 Building the Peridynamic Module within LAMMPS

In the LAMMPS distribution, the peridynamic model is distributed as an add-on module, which means that it is not by default compiled with the rest of LAMMPS. To instruct LAMMPS to build the peridynamic module, go to the LAMMPS source subdirectory (`/src`) and type

```
make yes-peri
```

followed by

```
make <your platform>
```

to compile LAMMPS on your particular platform.

2.2 Input Script Basics

Here we provide a listing of commands that must be included in a LAMMPS input script to utilize the peridynamic module. These commands assume knowledge of peridynamics (§3) and its discretization (§4). This is not an inclusive list of LAMMPS commands. For a complete example script, see §5.

LAMMPS has been modified to support SI units. To use SI units, your LAMMPS input script should contain the command

```
units si
```

All quantities specified in the input script and data file, as well as quantities output to the screen, log file, and dump files will be in SI units.

Only a simple cubic lattice is currently supported. Your LAMMPS input script should contain the command

```
lattice sc <lattice constant>
```

A peridynamic simulation requires the "peri" atom style be used. Your input script should contain the command

```
atom_style peri
```

An associated required command tells LAMMPS to create a data structure used to index particles. Your input script should contain the command

```
atom_modify map array
```

The "skin" distance used when computing neighborlists should be defined appropriately for your choice of simulation parameters. Your input script should contain the command

```
neighbor <skin> bin
```

where the "skin" should be set to a value such that the peridynamic horizon plus the skin distance is larger than the maximum possible distance between two bonded particles (before their bond breaks). A peridynamic simulation also requires a peridynamic pair style be used. Your input script should contain either the commands

```
pair_style peri/lps
pair_coeff <type 1> <type 2> <bulk modulus> <shear modulus> <delta> <s00> <α>
```

to invoke the "peri/lps" pair style, or the commands

```
pair_style peri/pmb
pair_coeff <type 1> <type 2> <c> <delta> <s00> <α>
```

to invoke the "peri/pmb" pair style, or the commands

```
pair_style peri/mp
pair_coeff <type 1> <type 2> <c> <delta> <Y> <s00> <α>
```

to invoke the "peri/mp" pair style. See §3.2 for more on the linear peridynamic solid (LPS) model, §3.3 for more on the prototype microelastic brittle (PMB) model, and §3.4 for more on the microplastic model.

The mass density and volume fraction for each particle must be defined. Your input script should contain the commands

```
set group all density <ρ>
set group all volume <Vi>
```

In the second line, you are setting the volume of each peridynamic particle. For a simple cubic lattice, the volume should be equal to the cube of the lattice constant, i.e., $V_i = \Delta x^3$.

If you wish to start a simulation with the velocity of the peridynamic particles set to zero, your input script should contain the command

```
velocity all set 0.0 0.0 0.0 sum no units box
```

We use a velocity-Verlet time integrator (algebraically equivalent to a centered difference in time, but numerically more stable.) To use a velocity-Verlet time integrator, your input script should contain the command

```
fix <fix id> all nve
```

You can compute the damage (see §3.5) at each particle with the compute style `damage/atom`:

```
compute <compute id> all damage/atom
```

To periodically dump snapshots of your simulation to disk, use the LAMMPS `dump` command:

```
dump <dump id> all custom <N> <dump filename> id type x y z c_<compute id>
```

where `N` is the number of steps between snapshots and `<compute id>` is the id of the `damage/atom` compute style above. You can visualize these snapshots (see §4.8).

To periodically dump the bond family to disk, use the LAMMPS `compute` and `dump` commands:

```
compute <compute id> all peri/local dump <dump id> all local <N> <output filename>
c_<compute id>
```

where `N` is the number of steps between snapshots. This output can be useful for post-processing data about fragment sizes and distributions. See §4.7 for more about the output format of the bond families.

3 Peridynamic Model of a Continuum

The following is not a complete overview of peridynamics, but a discussion of only those details specific to the model we have implemented within LAMMPS. For more on the peridynamic theory, the reader is referred to [12, 9]. To begin, we define the notation we will use.

3.1 Basic Notation

Within the peridynamic literature, the following notational conventions are generally used. The position of a given point in the reference configuration is $\mathbf{x}$. Let $\mathbf{u}(\mathbf{x}, t)$ and $\mathbf{y}(\mathbf{x}, t)$ denote the displacement and position, respectively, of the point $\mathbf{x}$ at time t. Define the relative position and displacement vectors of two bonded points $\mathbf{x}$ and $\mathbf{x}'$ as $\boldsymbol{\xi} = \mathbf{x}' - \mathbf{x}$ and $\boldsymbol{\eta} = \mathbf{u}(\mathbf{x}', t) - \mathbf{u}(\mathbf{x}, t)$, respectively. We note here that $\boldsymbol{\eta}$ is time-dependent, and that $\boldsymbol{\xi}$ is not. It follows that the relative position of the two bonded points in the current configuration can be written as $\boldsymbol{\xi} + \boldsymbol{\eta} = \mathbf{y}(\mathbf{x}', t) - \mathbf{y}(\mathbf{x}, t)$.

Peridynamic models are frequently written using *states*, which we briefly describe here. For the purposes of our discussion, all states are operators that act on vectors in $\mathbb{R}^3$. For a more complete discussion of states, see [12]. A *vector state* is an operator whose image is a vector, and may be viewed as a generalization of a second-rank tensor. Similarly, a *scalar state* is an operator whose image is a scalar. Of particular interest is the vector force state $\underline{\mathbf{T}}\left[\mathbf{x}, t\right]\left\langle \mathbf{x}' - \mathbf{x} \right\rangle$, which is a mapping, having units of force per volume squared, of the vector $\mathbf{x}' - \mathbf{x}$ to the force vector state field. The vector state operator $\underline{\mathbf{T}}$ may itself be a function of $\mathbf{x}$ and t. The constitutive model is completely contained within $\underline{\mathbf{T}}$.

In the peridynamic theory, the deformation at a point depends collectively on all points interacting with that point. Using the notation of [12], we write the peridynamic equation of motion as

$$\rho(\mathbf{x})\ddot{\mathbf{u}}(\mathbf{x}, t) = \int_{\mathcal{H}_{\mathbf{x}}} \left\{ \underline{\mathbf{T}}\left[\mathbf{x}, t\right]\left\langle \mathbf{x}' - \mathbf{x} \right\rangle - \underline{\mathbf{T}}\left[\mathbf{x}', t\right]\left\langle \mathbf{x} - \mathbf{x}' \right\rangle \right\} dV_{\mathbf{x}'} + \mathbf{b}(\mathbf{x}, t), \tag{3.1}$$

where ρ represents the mass density, $\underline{\mathbf{T}}$ the force vector state, and $\mathbf{b}$ an external body force density. A point $\mathbf{x}$ interacts with all the points $\mathbf{x}'$ within the neighborhood $\mathcal{H}_{\mathbf{x}}$, assumed to be a spherical region of radius $\delta > 0$ centered at $\mathbf{x}$. δ is called the *horizon*, and is analogous to the cutoff radius used in molecular dynamics. Conditions on $\underline{\mathbf{T}}$ for which (3.1) satisfies the balance of linear and angular momentum are given in [12].

We consider only force vector states that can be written as

$$\underline{\mathbf{T}} = \underline{t}\,\underline{\mathbf{M}},$$

with $\underline{t}$ a *scalar force state* and $\underline{\mathbf{M}}$ the *deformed direction vector state*, defined by

$$\underline{\mathbf{M}}\left\langle \boldsymbol{\xi} \right\rangle = \begin{cases} \frac{\boldsymbol{\xi}+\boldsymbol{\eta}}{\|\boldsymbol{\xi}+\boldsymbol{\eta}\|} & \|\boldsymbol{\xi}+\boldsymbol{\eta}\| \neq 0 \\ \mathbf{0} & \text{otherwise} \end{cases}. \tag{3.2}$$

Such force states correspond to so-called *ordinary* materials ([12]). These are the materials for which the force between any two interacting points $\mathbf{x}$ and $\mathbf{x}'$ acts along the line between the points.

3.2 Linear Peridynamic Solid (LPS) Model

We summarize the linear peridynamic solid (LPS) material model. For more on this model, the reader is referred to [12]. This model is a nonlocal analogue to a classical linear elastic isotropic material. The elastic properties of a a classical linear elastic isotropic material are determined by (for example) the bulk and shear moduli. For the LPS model, the elastic properties are analogously determined by the bulk and shear moduli, along with the horizon δ.

The LPS model has a force scalar state

$$\underline{t} = \frac{3K\theta}{m}\underline{\omega}\,\underline{x} + \alpha\underline{\omega}\,\underline{e}^{\mathrm{d}}, \tag{3.3}$$

with K the bulk modulus and α related to the shear modulus G as

$$\alpha = \frac{15G}{m}.$$

The remaining components of the model are described as follows. Define the reference position scalar state $\underline{x}$ so that $\underline{x}\langle\boldsymbol{\xi}\rangle = \|\boldsymbol{\xi}\|$. Then, the weighted volume m is defined as

$$m[\mathbf{x}] = \int_{\mathcal{H}_{\mathbf{x}}} \underline{\omega}\langle\boldsymbol{\xi}\rangle\,\underline{x}\langle\boldsymbol{\xi}\rangle\,\underline{x}\langle\boldsymbol{\xi}\rangle\,dV_{\boldsymbol{\xi}}. \tag{3.4}$$

Let

$$\underline{e}[\mathbf{x},t]\langle\boldsymbol{\xi}\rangle = \|\boldsymbol{\xi}+\boldsymbol{\eta}\| - \|\boldsymbol{\xi}\|$$

be the extension scalar state, and

$$\theta[\mathbf{x},t] = \frac{3}{m[\mathbf{x}]}\int_{\mathcal{H}_{\mathbf{x}}} \underline{\omega}\langle\boldsymbol{\xi}\rangle\,\underline{x}\langle\boldsymbol{\xi}\rangle\,\underline{e}[\mathbf{x},t]\langle\boldsymbol{\xi}\rangle\,dV_{\boldsymbol{\xi}}$$

be the dilatation. The isotropic and deviatoric parts of the extension scalar state are defined, respectively, as

$$\underline{e}^{\mathrm{i}} = \frac{\theta\underline{x}}{3}, \qquad \underline{e}^{\mathrm{d}} = \underline{e} - \underline{e}^{\mathrm{i}},$$

where the arguments of the state functions and the vectors on which they operate are omitted for simplicity. We note that the LPS model is linear in the dilatation θ, and in the deviatoric part of the extension $\underline{e}^{\mathrm{d}}$.

Remark 3.1. The weighted volume m is time-independent, and does not change as bonds break. It is computed with respect to the bond family defined at the reference (initial) configuration.

The nonnegative scalar state $\underline{\omega}$ is an *influence function* [12, Defn. 3.2]. For more on influence functions, see [8]. If an influence function $\underline{\omega}$ depends only upon the scalar $\|\boldsymbol{\xi}\|$, (i.e., $\underline{\omega}\langle\boldsymbol{\xi}\rangle = \underline{\omega}\langle\|\boldsymbol{\xi}\|\rangle$), then $\underline{\omega}$ is a spherical influence function. For a spherical influence function, the LPS model is isotropic [12, Prop. 14.1].

Remark 3.2. In the PDLAMMPS implementation of the LPS model, the influence function $\underline{\omega}\langle\|\boldsymbol{\xi}\|\rangle = 1/\|\boldsymbol{\xi}\|$ is used. However, the user can define their own influence function by altering the method `influence_function` in the file `pair_peri_lps.cpp`. The PDLAMMPS code permits both spherical and non-spherical influence functions (e.g., isotropic and non-isotropic materials).

Pseudocode for the LPS model appears in Appendix A.

3.3 Prototype Microelastic Brittle (PMB) Model

We summarize the prototype microelastic brittle (PMB) material model. For more on this model, the reader is referred to [9, 11]. This model is a special case of the LPS model; see [8] for the derivation. The elastic properties of the PMB model are determined by the bulk modulus K and the horizon δ.

The PMB model is expressed using the scalar force state field

$$\underline{t}\left[\mathbf{x}, t\right]\langle\boldsymbol{\xi}\rangle = \frac{1}{2} f\left(\boldsymbol{\eta}, \boldsymbol{\xi}\right), \tag{3.5}$$

with f a scalar-valued function. We assume that f takes the form

$$f = cs,$$

where

$$c = \frac{18K}{\pi\delta^4}, \tag{3.6}$$

with K the bulk modulus and δ the horizon, and s the bond stretch, defined as

$$s(t, \boldsymbol{\eta}, \boldsymbol{\xi}) = \frac{\|\boldsymbol{\eta} + \boldsymbol{\xi}\| - \|\boldsymbol{\xi}\|}{\|\boldsymbol{\xi}\|}. \tag{3.7}$$

Bond stretch is a unitless quantity, and identical to a one-dimensional definition of strain. As such, we see that a bond at its equilibrium length has stretch $s = 0$, and a bond at twice its equilibrium length has stretch $s = 1$. The constant c given above is appropriate for 3D models only. For more on the origins of the constant c, see [11]. For the derivation of c for 1D and 2D models, see [2].

Given (3.5), (3.1) reduces to

$$\rho(\mathbf{x})\ddot{\mathbf{u}}(\mathbf{x}, t) = \int_{\mathcal{H}_{\mathbf{x}}} \mathbf{f}\left(\boldsymbol{\eta}, \boldsymbol{\xi}\right) dV_{\boldsymbol{\xi}} + \mathbf{b}(\mathbf{x}, t), \tag{3.8}$$

with

$$\mathbf{f}\left(\boldsymbol{\eta}, \boldsymbol{\xi}\right) = f\left(\boldsymbol{\eta}, \boldsymbol{\xi}\right) \frac{\boldsymbol{\xi} + \boldsymbol{\eta}}{\|\boldsymbol{\xi} + \boldsymbol{\eta}\|}.$$

Unlike the LPS model, the PMB model has a Poisson ratio of $\nu = 1/4$ in 3D, and $\nu = 1/3$ in 2D. This is reflected in the input for the PMB model, which requires only the bulk modulus of the material, whereas the LPS model requires both the bulk and shear moduli.

Pseudocode for the PMB model appears in Appendix B.

3.4 Microplastic (MP) Model

We summarize the microplastic (MP) material model. For more on this model, the reader is referred to [4]. The material properties of the MP model are determined by the bulk modulus K, the yield stretch Y, and the horizon δ.

The MP model is expressed using the scalar force state field

$$\underline{t}\left[\mathbf{x}, t\right]\langle\boldsymbol{\xi}\rangle = \frac{1}{2} f\left(\boldsymbol{\eta}, \boldsymbol{\xi}\right), \tag{3.9}$$

with f a scalar-valued function. We assume that f takes the form

$$f = c\left(s - s_p(t)\right),$$

where c is defined in (3.6), s is defined in (3.7), and $s_p = s_p(t)$ is the plastic stretch history, defined as

$$\frac{d}{dt} s_p = \begin{cases} 0 & \text{if } |s - s_p| < Y \\ \frac{d}{dt} s & \text{otherwise} \end{cases}, \qquad s_p(0) = 0.$$

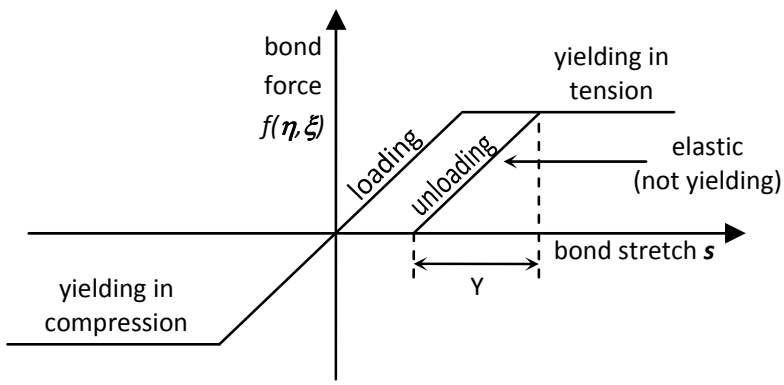


Figure 1. Bond force vs. bond stretch for microplastic material.

The MP model, like the PMB model, is a bond-based model and has a Poisson ratio of $\nu = 1/4$ in 3D, and $\nu = 1/3$ in 2D. This is reflected in the input for the MP model, which requires only the bulk modulus of the material. In the LAMMPS `thermo` output, only the elastic portion of the potential energy is output. The work done through plastic deformation is not recorded.

Pseudocode for the MP model appears in Appendix C.

3.5 Damage

Bonds are made to break when they are stretched beyond a given limit. Once a bond fails, it is failed forever [11]. Further, new bonds are never created during the course of a simulation. We discuss only one criterion for bond breaking, called the *critical stretch* criterion.

Define μ to be the history-dependent scalar boolean function

$$\mu(t, \boldsymbol{\eta}, \boldsymbol{\xi}) = \left\{ \begin{array}{ll} 1 & \text{if } s(t', \boldsymbol{\eta}, \boldsymbol{\xi}) < \min\left(s_0(t', \boldsymbol{\eta}, \boldsymbol{\xi}), s_0(t', \boldsymbol{\eta}', \boldsymbol{\xi}')\right) \text{ for all } 0 \le t' \le t \\ 0 & \text{otherwise} \end{array} \right\}. \tag{3.10}$$

where $\boldsymbol{\eta}' = \mathbf{u}(\mathbf{x}'', t) - \mathbf{u}(\mathbf{x}', t)$ and $\boldsymbol{\xi}' = \mathbf{x}'' - \mathbf{x}'$. Here, $s_0(t, \boldsymbol{\eta}, \boldsymbol{\xi})$ is a critical stretch defined as

$$s_0(t, \boldsymbol{\eta}, \boldsymbol{\xi}) = s_{00} - \alpha s_{\min}(t, \boldsymbol{\eta}, \boldsymbol{\xi}), \qquad s_{\min}(t) = \min_{\boldsymbol{\xi}} s(t, \boldsymbol{\eta}, \boldsymbol{\xi}), \tag{3.11}$$

where s_{00} and α are material-dependant constants. The history function μ breaks bonds when the stretch s exceeds the critical stretch s_0.

Although $s_0(t, \boldsymbol{\eta}, \boldsymbol{\xi})$ is expressed as a property of a particle, bond breaking must be a symmetric operation for all particle pairs sharing a bond. That is, particles $\mathbf{x}$ and $\mathbf{x}'$ must utilize the same test when deciding to break their common bond. This can be done by any method that treats the particles symmetrically. In the definition of μ above, we have chosen to take the minimum of the two s_0 values for particles $\mathbf{x}$ and $\mathbf{x}'$ when determining if the $\mathbf{x}$–$\mathbf{x}'$ bond should be broken.

Following [11], we can define the damage at a point $\mathbf{x}$ as

$$\varphi(\mathbf{x}, t) = 1 - \frac{\int_{\mathcal{H}_{\mathbf{x}}} \mu(t, \boldsymbol{\eta}, \boldsymbol{\xi}) dV_{\mathbf{x}'}}{\int_{\mathcal{H}_{\mathbf{x}}} dV_{\mathbf{x}'}}. \tag{3.12}$$

4 Discrete Peridynamic Model and LAMMPS Implementation

In LAMMPS, instead of (3.1), we model this equation of motion:

$$\rho(\mathbf{x})\ddot{\mathbf{y}}(\mathbf{x},t) = \int_{\mathcal{H}_{\mathbf{x}}} \left\{ \underline{\mathbf{T}}\left[\mathbf{x},t\right]\left\langle \mathbf{x}'-\mathbf{x}\right\rangle - \underline{\mathbf{T}}\left[\mathbf{x}',t\right]\left\langle \mathbf{x}-\mathbf{x}'\right\rangle \right\} dV_{\mathbf{x}'} + \mathbf{b}(\mathbf{x},t),$$

where we explicitly track and store at each timestep the positions and not the displacements of the particles. We observe that $\ddot{\mathbf{y}}(\mathbf{x},t) = \ddot{\mathbf{x}} + \ddot{\mathbf{u}}(\mathbf{x},t) = \ddot{\mathbf{u}}(\mathbf{x},t)$, so that this is equivalent to (3.1).

4.1 Spatial Discretization

The region defining a peridynamic material is discretized into particles forming a simple cubic lattice with lattice constant Δx, where each particle i is associated with some volume fraction V_i. For any particle i, let $\mathcal{F}_i$ denote the family of particles for which particle i shares a bond in the reference configuration. That is,

$$\mathcal{F}_i = \{p \mid \|\mathbf{x}_p - \mathbf{x}_i\| \leq \delta\}. \tag{4.1}$$

The discretized equation of motion replaces (3.1) with

$$\rho\ddot{\mathbf{y}}_i^n = \sum_{p\in\mathcal{F}_i} \left\{ \underline{\mathbf{T}}\left[\mathbf{x}_i,t\right]\left\langle \mathbf{x}_p'-\mathbf{x}_i\right\rangle - \underline{\mathbf{T}}\left[\mathbf{x}_p,t\right]\left\langle \mathbf{x}_i-\mathbf{x}_p\right\rangle \right\} V_p + \mathbf{b}_i^n, \tag{4.2}$$

where n is the timestep number and subscripts denote the particle number.

4.2 Short-Range Forces

In the model discussed so far, particles interact only through their bond forces. A particle with no bonds becomes a free non-interacting particle. To account for contact forces, short-range forces are introduced [10]. We add to the force in (4.2) the following force

$$\mathbf{f}_S(\mathbf{y}_p,\mathbf{y}_i) = \min\left\{0, \frac{c_S}{\delta}(\left\|\mathbf{y}_p-\mathbf{y}_i\right\| - d_{pi})\right\} \frac{\mathbf{y}_p-\mathbf{y}_i}{\left\|\mathbf{y}_p-\mathbf{y}_i\right\|}, \tag{4.3}$$

where d_{pi} is the short-range interaction distance between particles p and i, and c_S is a multiple of the constant c from (3.6). Note that the short-range force is always repulsive, never attractive. In practice, we choose

$$c_S = 15\frac{18K}{\pi\delta^4}. \tag{4.4}$$

For the short-range interaction distance, we choose [10]

$$d_{pi} = \min\left\{0.9\left\|\mathbf{x}_p-\mathbf{x}_i\right\|, 1.35(r_p+r_i)\right\}, \tag{4.5}$$

where r_i is called the *node radius* of particle i. Given a discrete lattice, we choose r_i to be half the lattice constant.[3] Given this definition of d_{pi}, contact forces appear only when particles are under compression.

When accounting for short-range forces, it is convenient to define the short-range family of particles

$$\mathcal{F}_i^S = \{p \mid \left\|\mathbf{y}_p - \mathbf{y}_i\right\| \leq d_{pi}\}.$$

4.3 Modification to the Particle Volume

The right-hand side of (4.2) may be thought of as a midpoint quadrature of (3.1). For situations when a particle is not completely contained within the neighborhood of another, the quadrature over this particle should take account only the volume fraction lying within the within the neighborhood. The quadrature in such cases is not done exactly due to cost, but instead a approximate modification to the particle volume used in (4.2). In a situation where two particles share a bond with $\|\mathbf{x}_p - \mathbf{x}_i\| = \delta$, for example, we suppose that only approximately half the volume of each particle is "seen" by the other [10]. When computing the force of each particle on the other we use $V_p/2$ rather than V_p in (4.2). As such, we introduce a nodal volume scaling function for all bonded particles where $\delta - r_i \leq \|\mathbf{x}_p - \mathbf{x}_i\| \leq \delta$ (c.f. Figure 2).

We choose to use a linear unitless nodal volume scaling function

$$\nu(\mathbf{x}_p - \mathbf{x}_i) = \left\{ \begin{array}{cl} -\frac{1}{2r_i}\|\mathbf{x}_p - \mathbf{x}_i\| + \left(\frac{\delta}{2r_i} + \frac{1}{2}\right) & \text{if } \delta - r_i \leq \|\mathbf{x}_p - \mathbf{x}_i\| \leq \delta \\ 1 & \text{if } \|\mathbf{x}_p - \mathbf{x}_i\| \leq \delta - r_i \\ 0 & \text{otherwise} \end{array} \right\}$$

If $\|\mathbf{x}_p - \mathbf{x}_i\| = \delta$, $\nu = 0.5$, and if $\|\mathbf{x}_p - \mathbf{x}_i\| = \delta - r_i$, $\nu = 1.0$, for example.

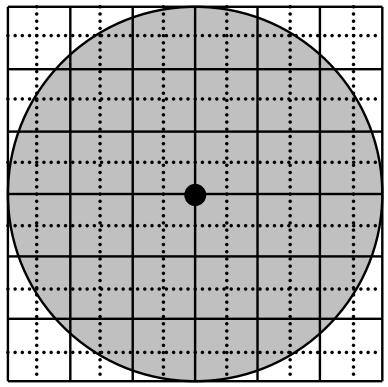
4.4 Temporal Discretization

When discretizing time in LAMMPS, we use a velocity-Verlet scheme, where both the position and velocity of the particle are stored explicitly. The velocity-Verlet scheme is generally expressed in three steps. In Algorithm 1, ρ_i denotes the mass density of a particle and $\widetilde{\mathbf{f}}_i^n$ denotes the the net force density on particle i at timestep n. The LAMMPS command `fix nve` performs a velocity-Verlet integration.

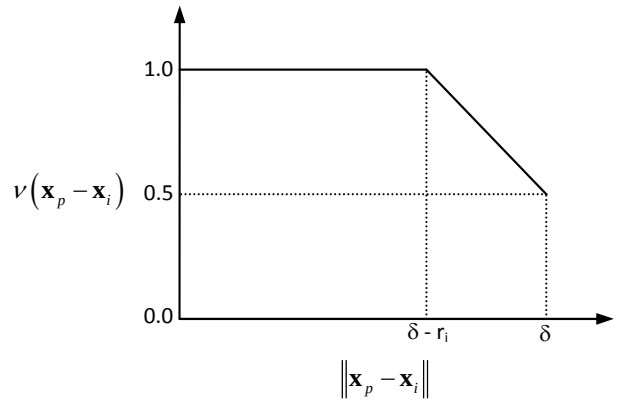
Algorithm 1 Velocity Verlet

1: $\mathbf{v}_i^{n+1/2} = \mathbf{v}_i^n + \frac{\Delta t}{2\rho_i}\widetilde{\mathbf{f}}_i^n$
2: $\mathbf{y}_i^{n+1} = \mathbf{y}_i^n + \Delta t \mathbf{v}_i^{n+1/2}$
3: $\mathbf{v}_i^{n+1} = \mathbf{v}_i^{n+1/2} + \frac{\Delta t}{2\rho_i}\widetilde{\mathbf{f}}_i^{n+1}$

[3] For a simple cubic lattice, $\Delta x = \Delta y = \Delta z$.



(a) Two-dimensional diagram showing particle on mesh (solid lines) with neighborhood $\mathcal{H}_{\mathbf{x}}$ as grey circular region. Dual mesh (dotted lines) shows boundaries of each particle.



(b) Plot of $\nu(\mathbf{x}_p - \mathbf{x}_i)$ vs. $\|\mathbf{x}_p - \mathbf{x}_i\|$.

Figure 2. Diagram showing horizon of a particular particle. The volume associated with particles near the boundary of the horizon is not completely contained within the horizon.

4.5 Breaking Bonds

During the course of simulation, it may be necessary to break bonds, as described in §3.5. Bonds are recorded as broken in a simulation by removing them from the bond family $\mathcal{F}_i$ (see (4.1)).

A naïve implementation would have us first loop over all bonds and compute s_{min} in (3.11), then loop over all bonds again and break bonds with a stretch $s > s0$ as in (3.10), and finally loop over all particles and compute forces for the next step of Algorithm 1. For reasons of computational efficiency, we will utilize the values of s_0 from the *previous* timestep when deciding to break a bond.

Remark 4.1. For the first timestep, s_0 is initialized to ∞ for all nodes. This means that no bonds may be broken until the second timestep. As such, it is recommended that the first few timesteps of the peridynamic simulation not involve any actions that might result in the breaking of bonds. As a practical example, the projectile in §5 is placed such that it does not impact the target brittle plate until several timesteps into the simulation.

4.6 Damage Compute Style

The damage associated with every particle (see (3.12)) can optionally be computed and output with a LAMMPS data dump. To do this, your input script must contain the command

```
compute <ComputeID> all damage/atom
```

This enables a LAMMPS per-atom compute to calculate the damage associated with each particle every time a LAMMPS data dump is called. To output the results of this compute in your dump file, you must use the LAMMPS `dump` command, as

```
dump <DumpID> all custom <N> <output filename> id type x y z c_<ComputeID>
```

where `N` is the number of timesteps between dumps.

4.7 Bond Family Compute Style

Some post-processing steps require knowledge of the bond family $\mathcal{F}_i$ from (4.1). For example, the bond families for each particle can be used to determine the number of fragments n a simulation, and the volume of each fragment. This section describes how to output all the *unbroken* bonds in each particle's bond family, which can be used as input to post-processing tools.

Output of the bond families is accomplished via a local compute style.

```
compute <ComputeID> all peri/local
```

Group IDs may be used if one does not wish to dump out the bond families for all particles. A separate dump file is recommended with this compute style, due to the size of the data output. To output the results of this compute in your dump file, you must use the LAMMPS `dump` command, as

```
dump <DumpID> all local <N> <output filename> c_<ComputeID>
```

where `N` is the number of timesteps between dumps.

The format for the dump file containing the bond families is consists of: (1) the ID of a particle, followed by (2) the number of unbroken bonds in the bond family for that particle, followed by (3) a listing of the IDs of each particle in the bond family, repeated for each particle.

As an example, suppose that we have a system of three particles with IDs 1, 2, and 3, and that each particle is bonded to the other two. Output from one timestep in the dump file will look like the figure in the box below.

```
ITEM: TIMESTEP
0
ITEM: NUMBER OF ENTRIES
12
ITEM: ENTRIES c_C2
1
2
2
3
2
2
1
3
3
2
1
2
```

We see that 12 bonds have been output. Each i-j bond is output twice, once for particle i and once for particle j, so there are 6 bonds in total. Reading this data directly, we see that particle 1 has two bonds, which are to particles 2 and 3. Particle 2 has 2 bonds, which are to particles 1 and 3, and particle 3 has two bonds, which are to particles 1 and 2.

Note that the ordering of the particles within the dump file may change from one timestep to the next.

4.8 Visualizing Simulation Results

LAMMPS does not visualize your simulation results. You'll need to post-process your LAMMPS data dump for use by a third-party visualization tool. Use of the pizza.py toolkit [6] is recommended for conversion of LAMMPS data dump to another format suitable for use of your visualization package of choice.

As an example, we outline here one possible means of visualizing the output of a peridynamic simulation using only freely available open-source software. We assume that you have a dump file named `dump.peri` constructed using line 35 in Algorithm 3, and that you have Algorithm 2 saved as a python script named `convert.py`.

1. Install the pizza.py toolkit from `http://www.sandia.gov/~sjplimp/pizza.html`.
2. Install the `Numeric` Python package (v.24.2) from `http://sourceforge.net/projects/numpy/files/Old%20Numeric/`, or the numpy Python package (current version) from `http://numeric.scipy.org/`. The package is installed correctly if you can type `>>> import Numeric` or `>>> import numpy` from an interactive python prompt.
3. Install ParaView from `www.paraview.org`.

Algorithm 2 Example Python Script to Convert LAMMPS Dump to Ensight .case File Format

```
import sys
from dump import dump
from ensight import ensight
d = dump("dump.peri");
d.scale();
d.map(1,"id",2,"type",3,"x",4,"y",5,"z",6,"damage");
e = ensight(d);
e.one("disk","damage","Damage")
```

4. Run the conversion script: `python convert.py`.

This will produce an Ensight-format `.case` file, as well as a `.xyz` file and a `.damage` file. Launch ParaView, open the `.case` file, and click the green `Apply` button on the left. For more on Paraview, see `www.paraview.org`.

4.9 Pitfalls

Parallel Scalability. LAMMPS operates in parallel in a spatial-decomposition mode [7], where each processor owns a spatial subdomain of the overall simulation domain and communicates with its neighboring processors via distributed-memory message passing (MPI) [13] to acquire ghost atom information to allow forces on the atoms it owns to be computed. LAMMPS also uses Verlet neighbor lists which are recomputed every few timesteps as particles move. On these timesteps, particles also migrate to new processors as needed. LAMMPS decomposes the overall simulation domain so that spatial subdomains of nearly equal volume are assigned to each processor. When each subdomain contains nearly the same number of particles, this results in a reasonable load balance among all processors. As is more typical with some peridynamic simulations, some subdomains may contain many particles while other subdomains contain few particles, resulting in a load imbalance that impacts parallel scalability.

Setting the "skin" distance. The `neighbor` command with LAMMPS is used to set the so-called "skin" distance used when building neighbor lists. All atom pairs within a cutoff distance equal to the horizon δ plus the skin distance are stored in the list. Unexpected crashes in LAMMPS may be due to too small a skin distance. The skin should be set to a value such that δ plus the skin distance is larger than the maximum possible distance between two bonded particles. For example, if s_{00} is increased, the skin distance may also need to be increased.

"Lost" particles. All particles are contained within the "simulation box" of LAMMPS. The boundaries of this box may change with time, or not, depending on how the LAMMPS `boundary` command has been set. If a particle drifts outside the simulation box during the course of a simulation, it is called *lost*.

As an option of the `themo_modify` command of LAMMPS, the lost keyword determines whether LAMMPS checks for lost atoms each time it computes thermodynamics and what it does if atoms

are lost. If the value is `ignore`, LAMMPS does not check for lost atoms. If the value is `error` or `warn`, LAMMPS checks and either issues an error or warning. The code will exit with an error and continue with a warning. This can be a useful debugging option. The default behavior of LAMMPS is to exit with an error if a particle is lost.

The peridynamic module within LAMMPS does not check for lost atoms. If a particle with unbroken bonds is lost, those bonds are marked as broken by the remaining particles.

Defining the peridynamic horizon δ. In the `pair_coeff` command, the user must specify the horizon δ. This argument determines which particles are bonded when the simulation is initialized. It is recommended that δ be set to a small fraction of a lattice constant larger than desired.

For example, if the lattice constant is 0.0005 and you wish to set the horizon to three times the lattice constant, then set δ to be 0.0015001, a value slightly larger than three times the lattice constant. This guarantees that particles three lattice constants away from each other are still bonded. If δ is set to 0.0015, for example, floating point error may result in some pairs of particles three lattice constants apart not being bonded.

Breaking bonds too early. For technical reasons, the bonds in the simulation are not created until the end of the first timestep of the simulation. Therefore, one should not attempt to break bonds until at least the second step of the simulation.

4.10 Bugs

The user is cautioned that this code is a beta release. If you are confident that you have found a bug in the peridynamic module, please send an email to the developers. First, check the "New features and bug fixes" section of the LAMMPS website site to see if the bug has already been reported or fixed. If not, the most useful thing you can do for us is to isolate the problem. Run it on the smallest number of atoms and fewest number of processors and with the simplest input script that reproduces the bug. In your email, describe the problem and any ideas you have as to what is causing it or where in the code the problem might be. We'll request your input script and data files if necessary.

4.11 Modifying and Extending the Peridynamic Module

To add new features or peridynamic potentials to the peridynamic module, the user is referred to section 8 of the LAMMPS user manual, *Modifying & extending LAMMPS*. To develop a new bond-based material, start with the PMB pair style as a template. To develop a new state-based material, start with the LPS pair style as a template.

5 A Numerical Example

To introduce the peridynamic implementation within LAMMPS, we replicate a numerical experiment taken from section 6 of [11].

5.1 Problem Description and Setup

We consider the impact of a rigid sphere on a homogeneous disk of brittle material. The sphere has diameter 0.01 m and velocity 100 m/s directed normal to the surface of the target. The target material has density $\rho = 2200$ kg/m^3. A PMB material model is used with $K = 14.9$ GPa and critical bond stretch parameters given by $s_{00} = 0.0005$ and $\alpha = 0.25$. A three-dimensional simple cubic lattice is constructed with lattice constant 0.0005 m and horizon 0.0015 m. (The horizon is three times the lattice constant.) The target is a cylinder of diameter 0.074 m and thickness 0.0025 m, and the associated lattice contains 103,110 particles. Each particle i has volume fraction $V_i = 1.25 \times 10^{-10}$ m^3.

The spring constant in the PMB material model is (see (3.6))

$$c = \frac{18k}{\pi\delta^4} = \frac{18(14.9 \times 10^9)}{\pi(1.5 \times 10^{-3})^4} \approx 1.6863 \times 10^{22}.$$

The CFL analysis from [11] shows that a timestep of 1.0×10^{-7} is safe.

We observe here that in IEEE double-precision floating point arithmetic when computing the bond stretch $s(t, \boldsymbol{\eta}, \boldsymbol{\xi})$ at each iteration where $\|\boldsymbol{\eta} + \boldsymbol{\xi}\|$ is computed during the iteration and $\|\boldsymbol{\xi}\|$ was computed and stored for the initial lattice, it may be that $fl(s) = \varepsilon$ with $|\varepsilon| \leq \varepsilon_{machine}$ for an unstretched bond. Taking $\varepsilon = 2.220446049250313 \times 10^{-16}$, we see that the value $csV_i \approx 4.68 \times 10^{-4}$, computed when determining f, is perhaps larger than we would like, especially when the true force should be zero. One simple way to avoid this issue is to insert the following instructions in Algorithm 7 after instruction 21 (and similarly for Algorithm 4):

1: **if** $|dr| < \varepsilon_{machine}$ **then**
2: $\quad dr = 0.$
3: **end if**

Qualitatively, this says that displacements from equilibrium on the order of 10^{-6}Å are taken to be exactly zero, a seemingly reasonable assumption.

5.2 The Projectile

The projectile used in the following experiments is not the one used in [11]. The projectile used here exerts a force

$$F(r) = -k_s(r - R)^2$$

on each atom where k_s is a specified force constant, r is the distance from the atom to the center of the indenter, and R is the radius of the projectile. The force is repulsive and $F(r) = 0$ for $r > R$.

For our problem, the projectile radius $R = 0.05$ m, and we have chosen $k_s = 1.0 \times 10^{17}$ (compare with (3.6) above).

5.3 Writing the LAMMPS Input File

We discuss the example input script from Algorithm 3. In line 2 we specify that SI units are to be used. We specify the dimension (3) and boundary conditions ("shrink-wrapped") for the computational domain in lines 3 and 4. In line 5 we specify that peridynamic particles are to be used for this simulation. In line 7, we set the "skin" distance used in building the LAMMPS neighborlist. In line 8 we set the lattice constant (in meters) and in line 10 we define the spatial region where the target will be placed. In line 12 we specify a rectangular box enclosing the target region that defines the simulation domain. Line 14 fills the target region with atoms. Lines 15 and 17 define the peridynamic material model, and lines 19 and 21 set the particle density and particle volume, respectively. The particle volume should be set to the cube of the lattice constant for a simple cubic lattice. Line 23 sets the initial velocity of all particles to zero. Line 25 instructs LAMMPS to integrate time with velocity-Verlet, and lines 27-30 create the spherical projectile, sending it with a velocity of 100 m/s towards the target. Line 32 declares a compute style for the damage (percentage of broken bonds) associated with each particle. Line 33 sets the timestep, line 34 instructs LAMMPS to provide a screen dump of thermodynamic quantities every 200 timesteps, and line 35 instructs LAMMPS to create a data file (`dump.peri`) with a complete snapshot of the system every 100 timesteps. This file can be used to create still images or movies. Finally, line 36 instructs LAMMPS to run for 2,000 timesteps.

Algorithm 3 Example LAMMPS Input Script

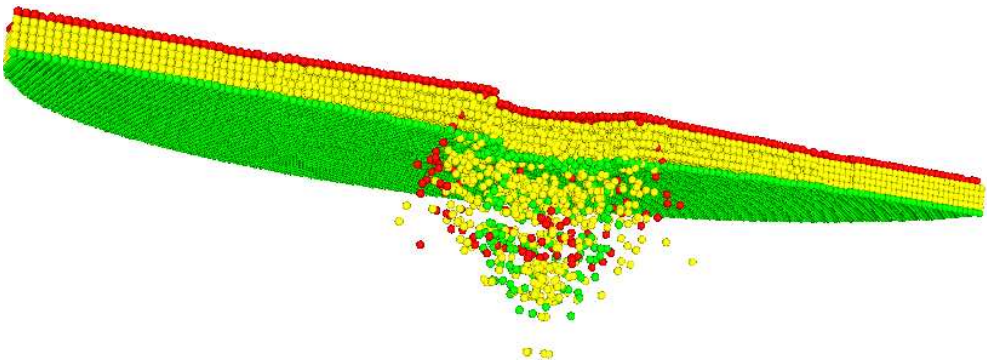
```
1: # 3D Peridynamic simulation with projectile
2: units           si
3: dimension       3
4: boundary        s s s
5: atom_style      peri
6: atom_modify     map array
7: neighbor        0.0010 bin
8: lattice         sc 0.0005
9: # Create desired target
10: region         target cylinder y 0.0 0.0 0.037 -0.0025 0.0 units box
11: # Make 1 atom type
12: create_box     1 target
13: # Create the atoms in the simulation region
14: create_atoms   1 region target
15: pair_style     peri/pmb [4]
16: #              <type1> <type2>    <c>    <horizon>  <s00> <alpha>
17: pair_coeff        *       *    1.6863e22 0.0015001 0.0005  0.25
18: # Set mass density
19: set            group all density 2200
20: # volume = lattice constant^3
21: set            group all volume 1.25e-10
22: # Zero out velocities of particles
23: velocity       all set 0.0 0.0 0.0 sum no units box
24: # Use velocity-Verlet time integrator
25: fix            F1 all nve
26: # Construct spherical indenter to shatter target
27: variable       y0 equal 0.00510
28: variable       vy equal -100
29: variable       y equal "v_y0 + step*dt*v_vy"
30: fix            F2 all indent 1e17 sphere 0.0000 v_y 0.0000 0.0050 units box
31: # Compute damage for each particle
32: compute        C1 all damage/atom
33: timestep       1.0e-7
34: thermo         200
35: dump           D1 all custom 100 dump.peri id type x y z c_C1
36: run            2000
```

5.4 Numerical Results and Discussion

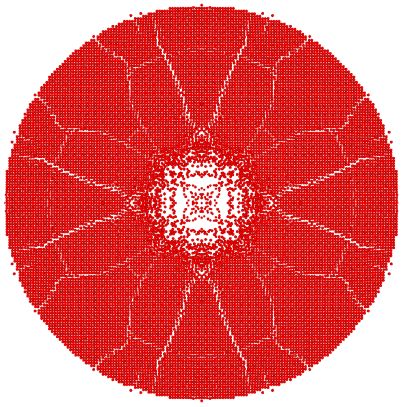
We ran the input script from Algorithm 3. Images of the disk (projectile not shown) appear in Figure 3. The LAMMPS dump file was converted to an EnSight data format with the pizza.py toolkit [6]. Visualization was done with the EnSight visualization package [1]. Use of the Paraview visualization package is also recommended, as it reads EnSight data files, and is open-source and freely available [3]. See §4.8 for more on visualizing peridynamic simulations results. The plot of damage on the top monolayer was created by coloring each particle according to its damage (see (3.12)).

The symmetry in the computed solution arises because a "perfect" lattice was used, and a

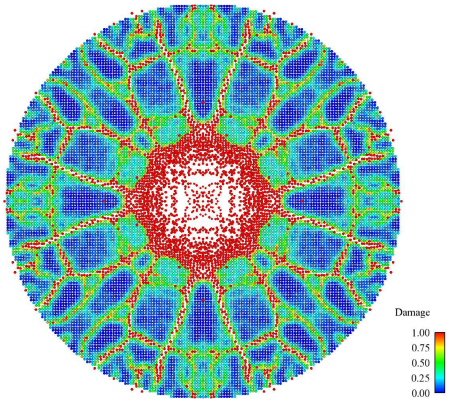
[4] To use the LPS or MP models, replace line 15 with `pair_style peri/lps` or `pair_style peri/mp` and modify line 17 accordingly; see page 9.



(a) Cut view of target during impact.



(b) Top monolayer showing fragmentation.



(c) Top monolayer showing damage. (blue = 0% broken bonds; red = 100% broken bonds)

Figure 3. Target during (a) and after (b,c) impact.

because a perfectly spherical projectile impacted the lattice at its geometric center. To break the symmetry in the solution, the nodes in the peridynamic body may be perturbed slightly from the lattice sites. To do this, a perturbed lattice of points can be prepared in a data file and read into LAMMPS using the `read_data` command.

References

[1] CEI, *EnSight web page*. `http://www.ensight.com/`.

[2] E. Emmrich and O. Weckner, *On the well-posedness of the linear peridynamic model and its convergence towards the Navier equation of linear elasticity*, Commun. Math. Sci., 5 (2007), pp. 851–864.

[3] Kitware Inc., *ParaView web page*. `http://www.paraview.org/`.

[4] R. W. Macek and S. A. Silling, *Peridynamics via finite element analysis*, Finite Elements in Analysis and Design, 43 (2007), pp. 1169–1178.

[5] M. L. Parks, R. B. Lehoucq, S. J. Plimpton, and S. A. Silling, *Implementing peridynamics within a molecular dynamics code*, Computer Physics Communications, 179 (2008), pp. 777–783.

[6] S. J. Plimpton, *Pizza.py webpage.* `http://www.cs.sandia.gov/~sjplimp/pizza.html`.

[7] ——, *Fast parallel algorithms for short-range molecular dynamics*, J. Comp. Phys., 117 (1995), pp. 1–19. Available at `http://lammps.sandia.gov`.

[8] P. Seleson and M. Parks, *On the role of the influence function in the peridynamic theory*, Int. J. Mult. Comp. Eng., (2010). Submitted.

[9] S. A. Silling, *Reformulation of elasticity theory for discontinuities and long-range forces*, Journal of the Mechanics and Physics of Solids, 48 (2000), pp. 175–209.

[10] S. A. Silling. Personal communication, 2007.

[11] S. A. Silling and E. Askari, *A meshfree method based on the peridynamic model of solid mechanics*, Computer and Structures, 83 (2005), pp. 1526–1535.

[12] S. A. Silling, M. Epton, O. Weckner, J. Xu, and E. Askari, *Peridynamic states and constitutive modeling*, J. Elasticity, 88 (2007), pp. 151–184.

[13] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra, *MPI: The Complete Reference*, vol. 1, The MIT Press, Cambridge, MA., 2 ed., 1998.

Appendix A: LPS Pseudocode

A sketch of the LPS model implementation in PDLAMMPS appears in Algorithm 4. This algorithm makes use of the routines in Algorithms 5 and 6.

Algorithm 4 LPS Peridynamic Model Pseudocode

Fix s_{00}, α, horizon δ, bulk modulus K, shear modulus G, timestep Δt, and generate initial lattice of particles with lattice constant Δx. Let there be N particles. Define constant c_S for repulsive short-range forces.
Initialize bonds between all particles $i \neq j$ where $\|\mathbf{x}_j - \mathbf{x}_i\| \leq \delta$
Initialize weighted volume m for all particles using Algorithm 5
Initialize $s_0 = \infty$ {*Initialize each entry to MAX_DOUBLE*}
while not done **do**
  Perform step 1 of Algorithm 1, updating velocities of all particles
  Perform step 2 of Algorithm 1, updating positions of all particles
  $\tilde{s}_0 = \infty$ {*Initialize each entry to MAX_DOUBLE*}
  for $i = 1$ to N **do**
    {Compute short-range forces}
    for all particles $j \in \mathcal{F}_i^S$ (the short-range family of nodes for particle i) **do**
      $r = \left\|\mathbf{y}_j - \mathbf{y}_i\right\|$
      $dr = \min\{0, r - d\}$. {*Short-range forces are only repulsive, never attractive*}
      $k = \frac{c_S}{\delta} V_k dr$ {c_S *defined in* (4.4)}
      $\mathbf{f} = \mathbf{f} + k \frac{\mathbf{y}_j - \mathbf{y}_i}{\|\mathbf{y}_j - \mathbf{y}_i\|}$
    end for
  end for
  Compute the dilatation for each particle using Algorithm 6
  for $i = 1$ to N **do**
    {Compute bond forces}
    for all particles j sharing an unbroken bond with particle i **do**
      $e = \left\|\mathbf{y}_j - \mathbf{y}_i\right\| - \|\mathbf{x}_j - \mathbf{x}_i\|$
      $\omega_+ = \underline{\omega}\left\langle \mathbf{x}_j - \mathbf{x}_i \right\rangle$ {*Influence function evaluation*}
      $\omega_- = \underline{\omega}\left\langle \mathbf{x}_i - \mathbf{x}_j \right\rangle$ {*Influence function evaluation*}
      $\hat{f} = \left[(3K - 5G)\left(\frac{\theta(i)}{m(i)}\omega_+ + \frac{\theta(j)}{m(j)}\omega_-\right)\|\mathbf{x}_j - \mathbf{x}_i\| + 15G\left(\frac{\omega_+}{m(i)} + \frac{\omega_-}{m(j)}\right)e\right]\nu(\mathbf{x}_j - \mathbf{x}_i)V_j$
      $\mathbf{f} = \mathbf{f} + \hat{f}\frac{\mathbf{y}_j - \mathbf{y}_i}{\|\mathbf{y}_j - \mathbf{y}_i\|}$
      if $(dr/\|\mathbf{x}_j - \mathbf{x}_i\|) > \min(s_0(i), s_0(j))$ **then**
        Break i's bond with j {*j's bond with i will be broken when this loop iterates on j*}
      end if
      $\tilde{s}_0(i) = \min(\tilde{s}_0(i), s_{00} - \alpha(dr/\|\mathbf{x}_j - \mathbf{x}_i\|))$
    end for
  end for
  $s_0 = \tilde{s}_0$ {*Store for use in next timestep*}
  Perform step 3 of Algorithm 1, updating velocities of all particles
end while

Algorithm 5 Computation of Weighted Volume m

```
for i = 1 to N do
    m(i) = 0.0
    for all particles j sharing a bond with particle i do
        m(i) = m(i) + ω̲ ⟨x_j − x_i⟩ ‖x_j − x_i‖² ν(x_j − x_i) V_j
    end for
end for
```

Algorithm 6 Computation of Dilatation θ

```
for i = 1 to N do
    θ(i) = 0.0
    for all particles j sharing an unbroken bond with particle i do
        e = ‖y_i − y_j‖ − ‖x_i − x_j‖
        θ(i) = θ(i) + ω̲ ⟨x_j − x_i⟩ ‖x_j − x_i‖ e ν(x_j − x_i) V_j
    end for
    θ(i) = (3 / m(i)) θ(i)
end for
```

Appendix B: PMB Pseudocode

A sketch of the PMB model implementation in PDLAMMPS appears in Algorithm 7.

Algorithm 7 PMB Peridynamic Model Pseudocode

Fix s_{00}, α, horizon δ, spring constant c, timestep Δt, and generate initial lattice of particles with lattice constant Δx. Let there be N particles.
Initialize bonds between all particles $i \neq j$ where $\|\mathbf{x}_j - \mathbf{x}_i\| \leq \delta$
Initialize $s_0 = \infty$ {*Initialize each entry to MAX_DOUBLE*}
while not done **do**
  Perform step 1 of Algorithm 1, updating velocities of all particles
  Perform step 2 of Algorithm 1, updating positions of all particles
  $\tilde{s}_0 = \infty$ {*Initialize each entry to MAX_DOUBLE*}
  for $i = 1$ to N **do**
    {Compute short-range forces}
    for all particles $j \in \mathcal{F}_i^S$ (the short-range family of nodes for particle i) **do**
      $r = \left\|\mathbf{y}_j - \mathbf{y}_i\right\|$
      $dr = \min\{0, r - d\}$ {*Short-range forces are only repulsive, never attractive*}
      $k = \frac{c_S}{\delta} V_k dr$ {c_S *defined in* (4.4)}
      $\mathbf{f} = \mathbf{f} + k \frac{\mathbf{y}_j - \mathbf{y}_i}{\|\mathbf{y}_j - \mathbf{y}_i\|}$
    end for
  end for
  for $i = 1$ to N **do**
    {Compute bond forces}
    for all particles j sharing an unbroken bond with particle i **do**
      $r = \left\|\mathbf{y}_j - \mathbf{y}_i\right\|$
      $dr = r - \|\mathbf{x}_j - \mathbf{x}_i\|$
      $k = \frac{c}{\|\mathbf{x}_i - \mathbf{x}_j\|} \nu(\mathbf{x}_i - \mathbf{x}_j) V_j dr$ {c *defined in* (3.6)}
      $\mathbf{f} = \mathbf{f} + k \frac{\mathbf{y}_j - \mathbf{y}_i}{\|\mathbf{y}_j - \mathbf{y}_i\|}$
      if $(dr / \|\mathbf{x}_j - \mathbf{x}_i\|) > \min(s_0(i), s_0(j))$ **then**
        Break i's bond with j {*j's bond with i will be broken when this loop iterates on j*}
      end if
      $\tilde{s}_0(i) = \min(\tilde{s}_0(i), s_{00} - \alpha(dr / \|\mathbf{x}_j - \mathbf{x}_i\|))$
    end for
  end for
  $s_0 = \tilde{s}_0$ {*Store for use in next timestep*}
  Perform step 3 of Algorithm 1, updating velocities of all particles
end while

Appendix C: MP Pseudocode

A sketch of the MP model implementation in PDLAMMPS appears in Algorithm 8.

Algorithm 8 MP Peridynamic Model Pseudocode

```
Fix s_00, α, horizon δ, spring constant c, timestep Δt, and generate initial lattice of particles with lattice constant
  Δx. Let there be N particles.
Initialize bonds between all particles i ≠ j where ‖x_j − x_i‖ ≤ δ
Initialize s_p(i, j) = 0 for all bonded particles i ≠ j. {s_p defined in §3.4}
Initialize s_0 = ∞ {Initialize each entry to MAX_DOUBLE}
while not done do
    Perform step 1 of Algorithm 1, updating velocities of all particles
    Perform step 2 of Algorithm 1, updating positions of all particles
    s̃_0 = ∞ {Initialize each entry to MAX_DOUBLE}
    for i = 1 to N do
        {Compute short-range forces}
        for all particles j ∈ F_i^S (the short-range family of nodes for particle i) do
            r = ‖y_j − y_i‖
            dr = min{0, r − d} {Short-range forces are only repulsive, never attractive}
            k = (c_S/δ) V_k dr {c_S defined in (4.4)}
            f = f + k (y_j − y_i)/‖y_j − y_i‖
        end for
    end for
    for i = 1 to N do
        {Compute bond forces}
        for all particles j sharing an unbroken bond with particle i do
            r = ‖y_j − y_i‖
            dr = r − ‖x_j − x_i‖
            stretch = dr/‖x_j − x_i‖
            if (s_p(i, j) − Y ≤ stretch) then
                s_p(i, j) = stretch − Y {Y defined in §3.4}
            else
                if (stretch ≤ s_p(i, j) − Y) then
                    s_p(i, j) = stretch + Y
                end if
            end if
            k = c ν(x_i − x_j) V_j {c defined in (3.6)}
            f = f + k(stretch − s_p(i, j)) (y_j − y_i)/‖y_j − y_i‖
            if stretch > min(s_0(i), s_0(j)) then
                Break i's bond with j {j's bond with i will be broken when this loop iterates on j}
            end if
            s̃_0(i) = min(s̃_0(i), s_00 − α · stretch)
        end for
    end for
    s_0 = s̃_0 {Store for use in next timestep}
    Perform step 3 of Algorithm 1, updating velocities of all particles
end while
```

DISTRIBUTION:

1	MS 1322	John Aidun, 1435
1	MS 1318	Rob Hoekstra, 1414
1	MS 1320	Scott Collis, 1416
1	MS 1320	Richard Lehoucq, 1414
1	MS 1320	Michael Parks, 1414
1	MS 1316	Steven Plimpton, 1416
1	MS 1322	Stewart Silling, 1435
1	MS 0899	RIM-Reports Management, 9532 (electronic copy)
1	MS 0359	D. Chavez, LDRD Office, 1911

Sandia National Laboratories