Implementation of Peridynamics utilizing HPX – the C++ standard library for parallelism and concurrency

Patrick Diehl[1,4,*], Prashant K. Jha[2], Hartmut Kaiser[3,4], Robert Lipton[2], Martin Levesque[1]

1. Department for Multi scale Mechanics, Polytechnique Montreal
2. Department of Mathematics, Louisiana State University
3. Department of Computer Science, Louisiana State University
4. Ste||ar group
* patrick.diehl@polymtl.ca (https://orcid.org/0000-0003-3922-8419)

June 20, 2018

Abstract

Peridynamics is a non-local generalization of continuum mechanics tailored to address discontinuous displacement fields arising in fracture mechanics. As many non-local approaches, peridynamics requires considerable computing resources to solve practical problems. Several implementations of peridynamics utilizing CUDA, OpenCL, and MPI were developed to address this important issue. On modern supercomputers, asynchronous many task systems are emerging to address the new architecture of computational nodes. This paper presents a peridynamics EMU nodal discretization implementation with the C++ Standard Library for Concurrency and Parallelism (HPX), an open source asynchronous many task run time system. The code is designed for modular expandability, so as to simplify it to extend with new material models or discretizations. The code is convergent for implicit time integration and recovers theoretical solutions. Explicit time integration, convergence results are presented to showcase the agreement of results with theoretical claims in previous works. Two benchmark tests on code scalability are applied demonstrating agreement between this code's scalability and theoretical estimations.

Nomenclature

$\ddot{u}$ Acceleration $[\mathrm{m\,s^{-2}}]$

δ Horizon [m]

κ Perturbation

η Relative displacement

$\mathbf{f}$ Internal force density

$\mathbf{u}$ Displacement vector

$\mathbf{v}$ Velocity $[\mathrm{m\,s^{-1}}]$

$\mathbf{X}$ Material point

$\mathbf{x}$ Discrete node

μ Shear modulus [Pa]

Ω_0	Reference configuration
σ	Stress [Pa]
τ	Tolerance of the solver for implicit time integration
θ	Dilitation
$\underline{e}$	Extension state
$\underline{e}^d$	Deviatoric extension state
$\underline{t}$	Scalar force state
ε	Strain
ϱ	Density
B_δ	Neigborhood with the radius δ
K	Bulk modulus [Pa]
m	Weighted volume
U	Strain energy [J]
V	Surrounding volume [m^3]

1 Introduction

Peridynamics is a non-local generalization of continuum mechanic, tailored to address discontinuous displacement fields that arise in fracture mechanics [7, 9, 21, 8, 34]. Several peridynamics implementations utilizing the EMU nodal discretization [28] are available. Peridigm [24] and PDLammps [25] rely on the widely used Message Passing Interface (MPI) for the inter-node parallelization. Other approaches reply on acceleration cards, like OpenCL [23] and CUDA [3], for parallelization. These approaches, however, cannot deal with large node clouds due to hardware memory limitations.

When compared to local approaches, like finite element, peridynamics computations require substantial computational resources, which hinders its widespread use. Authors have devised local-non-local bridging schemes to target domains where peridynamics calculations are required and domains where local approaches yield acceptable results, to reduce the computational costs [33, 16, 6]. However, verification and validation of peridynamics codes still require important computational resources [19, 20, 4, 31, 1] and the community would greatly benefit from efficient implementations.

Modern supercomputers' many core architectures, like field-programmable gate arrays (FPGAs) and Intel Knights Landing, provide more threads per computational node as before. Fine-grain parallelism algorithms that use more threads on a single node are utilized rather than a few threads on several computational nodes must be developed.

Asynchronous Many Task (AMT) run time systems represent an emerging paradigm for addressing fine-grain parallelism since they handle the increasing amount of threads per node and concurrency. The C++ programming Language standard 11 laid the foundations for concurrency by introducing the concept of futurization to enable a task-based parallelism. In addition, the support for parallel execution with the so-called parallel algorithms were introduced in the C++ 17 standard.
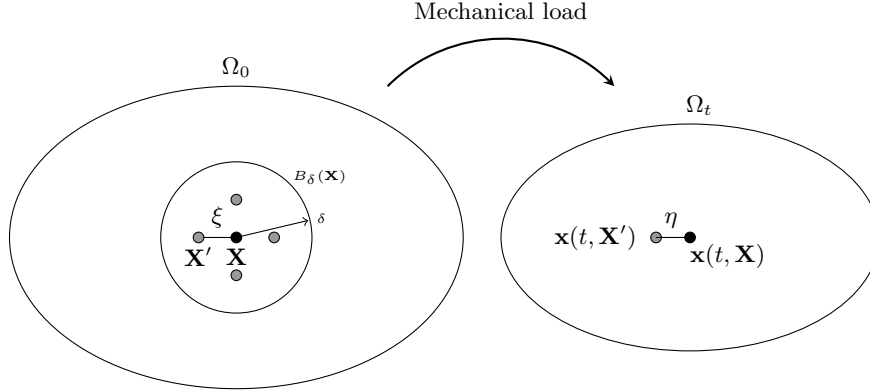
Figure 1: Left: Schematic representation of PD where the material point $\mathbf{X} \in \Omega$ interacts non-locally with all other material points inside $B_\delta(\mathbf{X})$. Right: Positions $\mathbf{x}(t, \mathbf{X})$ and $\mathbf{x}(t, \mathbf{X}')$ of the material points in the configuration Ω_t and the *bond-deformation vector* η.

HPX is an open source C++ standard library a asynchronous many task run time system that focuses on high performance computing [10]. HPX provides wait-free asynchronous execution and futurization for synchronization. It also features parallel execution policies utilizing a task scheduler, which enables a fine-grained and parallelization and synchronization due to work stealing. HPX is in strict adherence to the C++ 11 and C++ 17 standard definitions.

This paper presents two peridynamics models discretized with the EMU nodal discretization making use of the features of AMT within HPX. We show in this paper how to take advantage of the fine-grain parallelism arising on modern supercomputers. In addition, the scalability of the algorithm is compared to its theoretical complexity and the promised work stealing of HPX is addressed. The implementation is validated against results from classical continuum mechanics and numerical analysis.

The paper is structured as follows. Section 2 reviews peridynamics models, the EMU-ND discretization, the time integration, convergence rates for bond-based material models, and the concepts of HPX. Section 3 describes the proposed modular design and the implementation within the concepts of HPX. The parallel implementation is validated against classical continuum mechanics and numerical analysis in Section 4. The actual computational time for the HPX peridynamics implementation is benchmarked against the theoretical computation time in Section 5. Section 6 concludes this work.

2 Background

2.1 Peridynamics theory

Let a material domain be $\Omega_0 \subset \mathbb{R}^d$, for $d = 1, 2$, and 3. Peridynamics (PD) [30, 26] assumes that every material point $\mathbf{X} \in \Omega_0$ interacts non-locally with all other material points inside a horizon of length $\delta > 0$, as illustrated in Figure 1. Let $B_\delta(\mathbf{X})$ be the ball of radius δ centered at $\mathbf{X}$. When Ω_0 is submitted to mechanical loads, the material point $\mathbf{X}$ assumes position $\mathbf{x}(t, \mathbf{X}) = \mathbf{X} + \mathbf{u}(t, \mathbf{X})$, where $\mathbf{u}(t, \mathbf{X})$ is the displacement of material point $\mathbf{X}$ at time t. The vector $\eta := \mathbf{u}(t, \mathbf{X}') - \mathbf{u}(t, \mathbf{X})$ is called the *bond-deformation vector* and $\xi := \mathbf{X}' - \mathbf{X}$ denotes the initial bond vector.

Let $\mathbf{f}(t, \mathbf{u}(t, \mathbf{X}') - \mathbf{u}(t, \mathbf{X}), \mathbf{X}' - \mathbf{X})$ denote the peridynamic force as a function of time t, bond-deformation vector $\mathbf{u}(t, \mathbf{X}) - \mathbf{u}(t, \mathbf{X})$, and reference bond vector $\mathbf{X} - \mathbf{X}'$. The peridynamics equation
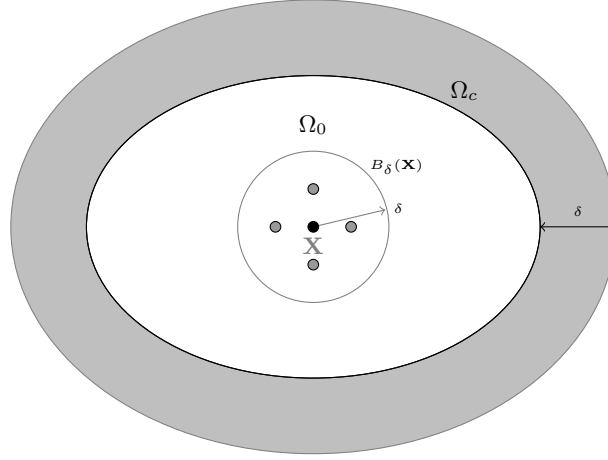
Figure 2: A collar domain Ω_c of thickness δ surrounding Ω_0 is used for both the Dirichlet formulation and the Neumann formulation.

of motion is given by

$$\varrho(\mathbf{X})\ddot{\mathbf{u}}(t, \mathbf{X}) = \int_{B_\delta(\mathbf{X})} \mathbf{f}(t, \mathbf{u}(t, \mathbf{X}') - \mathbf{u}(t, \mathbf{X}), \mathbf{X}' - \mathbf{X})d\mathbf{X}' + \mathbf{b}(t, \mathbf{X}), \tag{2.1}$$

where $\mathbf{b}$ is the external force density, and $\varrho(\mathbf{X})$ is the material's mass density. Equation (2.1) is complemented by initial conditions $\mathbf{u}(0, \mathbf{X}) = \mathbf{u}_0(\mathbf{X})$ and $\dot{\mathbf{u}}(0, \mathbf{X}) = v_0(\mathbf{X})$. In contrast to local problems, boundary conditions are non-local and are specified on a boundary layer or collar Ω_c that surrounds the domain Ω_0. The non-local boundary conditions are detailed in the next section.

The formulation given above depends on two-point interactions and is referred to as a bond-based peridynamics model. Bond based models allow for only two point non-local interactions and because of this the material's Poisson ratio is constrained to 0.25 [17, 18]. On the other hand, state-based peridynamics models [30] allow for multi-point non-local interactions and overcomes the restriction on the Poisson's ratio. It is conveniently formulated in terms of displacement dependent tensor valued functions. Let $\underline{T}[t, \mathbf{X}]$ be the peridynamic state at time t and material point $\mathbf{X}$. The peridynamics equation of motion for a state-based model is given by

$$\varrho(\mathbf{X})\ddot{\mathbf{u}}(t, \mathbf{X}) =$$
$$\int_{B_\delta(\mathbf{X})} \left(\underline{T}[\mathbf{X}, t]\langle \mathbf{X}' - \mathbf{X}\rangle - \underline{T}[\mathbf{X}', t]\langle \mathbf{X} - \mathbf{X}'\rangle \right) d\mathbf{X}' + \mathbf{b}(t, \mathbf{X}). \tag{2.2}$$

Equation (2.2) is complemented by initial conditions $\mathbf{u}(0, \mathbf{X}) = \mathbf{u}_0(\mathbf{X})$ and $\dot{\mathbf{u}}(0, \mathbf{X}) = v_0(\mathbf{X})$. The description of initial boundary value problem is completed by the non-local boundary conditions described in the next section.

2.2 Non-local boundary conditions

2.2.1 Dirichlet formulation

We introduce a boundary layer Ω_c of thickness δ, referred to as a collar, surrounding Ω_0, (see Figure 2). Let $\Omega_0 \cup \Omega_c$ define the complete material domain and we prescribe a displacement field on Ω_c. These are the non-local Dirichlet conditions. With the displacement field for $\mathbf{X} \in \Omega_c$ specified, we then solve for the displacement in $\mathbf{X} \in \Omega_0$ using the peridynamic equation of motion.

4

The peridynamic equation of motion for $\mathbf{X} \in \Omega_0$ reads

$$\varrho(\mathbf{X})\ddot{\mathbf{u}}(t,\mathbf{X}) = \int_{B_\delta(\mathbf{X})} \mathbf{f}(t,\mathbf{u}(t,\mathbf{X}') - \mathbf{u}(t,\mathbf{X}),\mathbf{X}' - \mathbf{X})d\mathbf{X}' + \mathbf{b}(t,\mathbf{X}), \qquad (2.3)$$

for a bond-based model. Similarly, the equation of motion for the state-based model is formulated for $\mathbf{X} \in \Omega_0$ and given by

$$\varrho(\mathbf{X})\ddot{\mathbf{u}}(t,\mathbf{X}) = \int_{B_\delta(\mathbf{X})} \left(\underline{T}[\mathbf{X},t]\langle\mathbf{X}' - \mathbf{X}\rangle - \underline{T}[\mathbf{X}',t]\langle\mathbf{X} - \mathbf{X}'\rangle\right)d\mathbf{X}' + \mathbf{b}(t,\mathbf{X}). \qquad (2.4)$$

2.2.2 Neumann formulation

The non-local Neumann condition is given in terms of a body force field localized to the collar Ω_c and we form $\mathbf{X} \in \tilde{\Omega} = \Omega_c \cup \overline{\Omega}_0$. We seek a solution of the balance of momentum equation for $\mathbf{X}$ in $\tilde{\Omega}$, see [26]. The peridynamic equation of motion for $\mathbf{X} \in \tilde{\Omega}$ reads

$$\varrho(\mathbf{X})\ddot{\mathbf{u}}(t,\mathbf{X}) = \int_{B_\delta(\mathbf{X})\cap\tilde{\Omega}} \mathbf{f}(t,\mathbf{u}(t,\mathbf{X}') - \mathbf{u}(t,\mathbf{X}),\mathbf{X}' - \mathbf{X})d\mathbf{X}' + \mathbf{b}(t,\mathbf{X}), \qquad (2.5)$$

for a bond-based model. Similarly, the equation of motion for the state-based model is given by

$$\varrho(\mathbf{X})\ddot{\mathbf{u}}(t,\mathbf{X}) = \int_{B_\delta(\mathbf{X})\cap\tilde{\Omega}} \left(\underline{T}[\mathbf{X},t]\langle\mathbf{X}' - \mathbf{X}\rangle - \underline{T}[\mathbf{X}',t]\langle\mathbf{X} - \mathbf{X}'\rangle\right)d\mathbf{X}' + \mathbf{b}(t,\mathbf{X}). \qquad (2.6)$$

2.3 Material models

2.3.1 Bond-based nonlinear peridynamics model

Bond based material forces are often formulated as initially elastic and then failing abruptly; this is known as the prototypical microelastic bond model (PMB) [27, 29, 25]. We use a smooth version of this type of model based on a nonlinear potential (NP) where the bond force depends on the tensile strain between two points [19, 20]. The bond force is initially elastic, after reaching a critical strain, softens to zero. The peridynamic force f for the NP material model is given by

$$\mathbf{f}(t,\mathbf{u}(t,\mathbf{X}') - \mathbf{u}(t,\mathbf{X}),\mathbf{X}' - \mathbf{X})) = \qquad (2.7)$$

$$\frac{2}{|B_\delta(\mathbf{X})|} \int_{B_\delta(\mathbf{X})} \frac{\partial}{\partial S} W^\delta(S(\mathbf{X}',\mathbf{X},\mathbf{u}(t))) \frac{\mathbf{X}' - \mathbf{X}}{|\mathbf{X}' - \mathbf{X}|} d\mathbf{X}'. \qquad (2.8)$$

Here, $|B_\delta(\mathbf{X})|$ denotes the volume of a sphere of radius δ in dimension $d = 1, 2, 3$. The bond strain S between two material points $\mathbf{X}'$ and $\mathbf{X}$ is given by
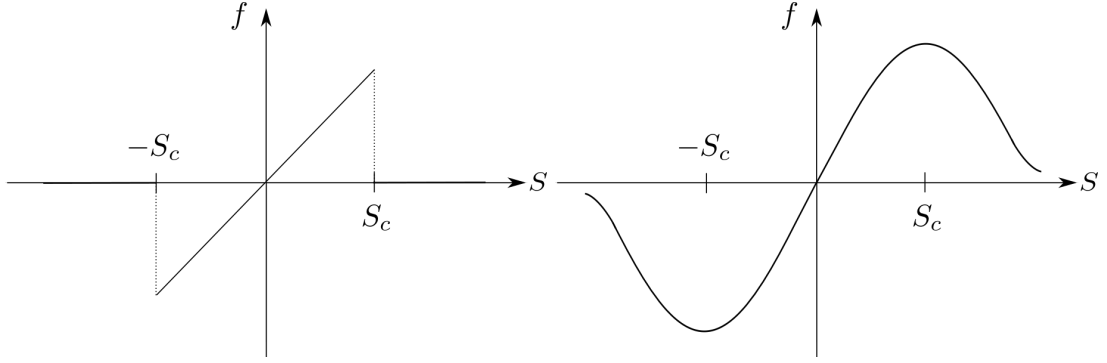
$$S(\mathbf{X}',\mathbf{X},\mathbf{u}(t)) = \frac{\mathbf{u}(t,\mathbf{X}') - \mathbf{u}(t,\mathbf{X})}{|\mathbf{X}' - \mathbf{X}|} \cdot \frac{\mathbf{X}' - \mathbf{X}}{|\mathbf{X}' - \mathbf{X}|}, \qquad (2.9)$$

where $\cdot$ denotes the dot product of the two vectors. The potential energy density W^δ is described by

$$W^\delta(S,\mathbf{X}' - \mathbf{X}) = \frac{J^\delta(|\mathbf{X}' - \mathbf{X}|)}{\delta|\mathbf{X}' - \mathbf{X}|}\psi(|\mathbf{X}' - \mathbf{X}|S^2), \qquad (2.10)$$

where $J^\delta(|\mathbf{X}' - \mathbf{X}|)$ is the influence function that is zero for $|\mathbf{X}' - \mathbf{X}| \geq \delta$. The nonlinear function $\psi(r)$, where $r = |\mathbf{X}' - \mathbf{X}|S^2$, is the potential function which is assumed to be smooth, positive and concave, with

$$\lim_{r\to 0^+} \frac{\psi(r)}{r} = \psi'(0), \qquad \lim_{r\to\infty} \psi(r) = \psi_\infty < \infty. \qquad (2.11)$$

(a) Strain vs force response of a PMB material.

(b) Strain vs Force response associated with a NP material.

Figure 3: Strain S as a function of force f for a linear and nonlinear potential ψ

Figure 3a, plots the bond-force as a function of a bond-strain for the prototypical micro-elastic bond-based model [25, 27, 29]. For a PMB material, the bond-force drops to zero once the strain S exceeds the critical value S_c. Figure 3b shows the force as a function of bond strain for the NP material characterized by the potential $\psi(r) = 1 - \exp(-r)$. The figure shows that the NP material softens after a critical strain and the force smoothly decays to zero with the increasing strain.

The energy density at the material point $\mathbf{X}$ is given by

$$U(\mathbf{X}) = \frac{1}{|B_\delta(\mathbf{X})|} \int_{B_\delta(\mathbf{X})} |\mathbf{X}' - \mathbf{X}| W^\delta(S(\mathbf{u}), \mathbf{X}' - \mathbf{X}) d\mathbf{X}' \tag{2.12}$$

and is a quantity to obtain stable time steps.

2.3.2 State-based linear peridynamics model

The state $\underline{T}$ for a specific linear material can be given by [30]

$$\underline{T}[\mathbf{X}, t] < \mathbf{X}' - \mathbf{X} > = \underline{M} \left(\frac{3}{m_\mathbf{X}} K \theta_\mathbf{X} |\xi| + \frac{15\mu}{m_\mathbf{X}} \underline{e}^d \right) \tag{2.13}$$

$$\underline{M} = \frac{\eta + \xi}{|\eta + \xi|} \tag{2.14}$$

$$\underline{e}^d = \underline{e}_\mathbf{X} - \frac{\theta_\mathbf{X} |\xi|}{3}, \tag{2.15}$$

where K and μ are respectively the material's bulk and shear modulus. The weighted volume $m_\mathbf{X}$ yields

$$m_\mathbf{X} = \int_{B_\delta(\mathbf{X})} |\xi|^2 d\mathbf{X}' \tag{2.16}$$

and the dilatation is given by

$$\theta_\mathbf{X} = \int_{B_\delta(\mathbf{X})} \frac{3}{m_\mathbf{X}} |\xi| \underline{e} d\mathbf{X}' \text{ with } \underline{e} = |\xi + \eta| - |\xi|. \tag{2.17}$$

2.4 Discretization of peridynamics equations

Continuous and discontinuous Galerkin finite element methods [2], Gauss quadrature [32] and spatial discretization [5, 25] are different discretization approaches for PD. Owing its to efficient

Figure 4: The initial positions of the discrete nodes $\mathbf{x} := \{\mathbf{x}_i \in \mathbb{R}^3 : i = 1, \dots, 9\}$ in the reference configuration Ω_0. For the discrete node $\mathbf{x}_5$ its surrounding volume V_5 is schematized.

load distribution scheme, the EMU nodal discretization (EMU ND) [25] is chosen for this implementation.

In the EMU ND scheme, every material point $\mathbf{X}$ is placed at the nodes $\mathbf{x} := \{x_i \in \mathbb{R}^3 : i = 1, \dots, n\}$, (e.g. in a regular grid in the reference configuration Ω_0, as schematized in Figure 4). The discrete nodal spacing h between x_j and x_i is given by $h = |x_j - x_i|$. The discrete neighborhood $B_\delta(x_i)$ of the node x_i yields $B_\delta(x_i) := \{x_j | |x_j - x_i| \le \delta\}$. In addition, all nodes $\mathbf{x}$ are associated with a surrounding volume $\mathbf{V} := \{V_i \in \mathbb{R} : i = 1, \dots, n\}$. These volumes $\mathbf{V}$ recover the reference configuration volume as per $\sum_{i=1}^{n} V_i = V_{\Omega_0}$.

The discrete bond-based equation of motion yields, for all $x_i \in \Omega_0$,

$$\varrho(x_i)\ddot{\mathbf{u}}(t, x_i) = \sum_{B_\delta(x_i)} \mathbf{f}(t, \mathbf{u}(t, x_j) - u(t, x_i), x_j - x_i)V_j + \mathbf{b}(t, x_i) \tag{2.18}$$

and the discrete state-based equation of motion yields, for all $x_i \in \Omega_0$,

$$\varrho(x_i)\ddot{\mathbf{u}}(t, x_i) = \sum_{B_\delta(x_i)} \left(\underline{T}[x_i, t]\langle x_j - x_i \rangle - \underline{T}[x_j, t]\langle x_i - x_j \rangle\right) V_j + \mathbf{b}(t, x_i). \tag{2.19}$$

These equations are discretized in space and need additional time integration schemes.

2.5 Time integration

2.5.1 Implicit scheme for quasi-static problem

Assume that, acceleration $\ddot{\mathbf{u}}(t, x_i)$ is equal to zero. The state-based equation of motion (2.19) becomes

$$\sum_{B_\delta(x_i)} \left(\underline{T}[x_i, t]\langle x_j - x_i \rangle - \underline{T}[x_j, t]\langle x_i - x_j \rangle\right) V_j - \mathbf{b}(t, x_i) = 0 \qquad \forall x_i \in \Omega_0. \tag{2.20}$$

When a external force $\mathbf{b}(t, x_i)$ is applied as a load, the system of equations

$$
\begin{aligned}
\mathbf{f}(x_1) \quad &= \sum_{B_\delta(x_1)} \left(\underline{T}[x_1, t]\langle x_j - x_1 \rangle - \quad \underline{T}[x_j, t]\langle x_1 - x_j \rangle\right) V_j = -\mathbf{b} \qquad (t, x_1) \\
\vdots \quad & \qquad\qquad \vdots \qquad\qquad\qquad\qquad \vdots \\
\mathbf{f}(x_n) \quad &= \sum_{B_\delta(x_n)} \left(\underline{T}[x_n, t]\langle x_j - x_n \rangle - \quad \underline{T}[x_j, t]\langle x_n - x_j \rangle\right) V_j = -\mathbf{b} \qquad (t, x_n) \tag{2.21}
\end{aligned}
$$

is not in equilibrium and must be solved for the displacement vector $\underline{\mathbf{u}}$. This nonlinear system of equations can be cast under

$$\mathbf{K}(\underline{\mathbf{u}}) = -\mathbf{r} \tag{2.22}$$

where $\mathbf{K}$ is the tangent stiffness matrix and $\mathbf{r}$ the residual vector. This system can be solved using Newton's method. One approach is to compute the entries of the tangent stiffness matrix with respect to the displacement is to use a central difference scheme [22] as

$$\mathbf{K}_{ij} \approx \frac{\mathbf{f}(x_i, u_i + \epsilon^j) - \mathbf{f}(x_i, u_i - \epsilon^j)}{2\epsilon}. \tag{2.23}$$

The residual vector is given by

$$\mathbf{r} = \sum_{\Omega_0} \mathbf{f}(t, x_i) + \mathbf{b}(t, x_i) \tag{2.24}$$

and $|r|$ is the l_2 norm of the residual vector $\mathbf{r}$.

2.5.2 Explicit time integration scheme for dynamic problems

The the central difference scheme can be used to evolve discrete dynamic problems and the evolution is given by

$$\mathbf{u}(t^{k+1}, x_i) = 2\mathbf{u}(t^k, x_i) - \mathbf{u}(t^{k-1}, x_i) + \frac{\Delta t^2}{\varrho(x_i)} \left[\mathbf{b}(t^k, x_i) + \sum_{B_\delta(x_i)} \mathbf{f}(t^k, \eta, \xi) \right]. \tag{2.25}$$

The velocity-verlet scheme is also used and the evolution is given by

$$v(t^{k+1/2}, x_i) = v(t^k, x_i) + \frac{(\Delta t/2)}{\varrho(x_i)} \left[\mathbf{b}(t^k, x_i) + \sum_{B_\delta(x_i)} \mathbf{f}(t^k, \eta, \xi) \right],$$

$$\mathbf{u}(t^{k+1}, x_i) = \mathbf{u}(t^k, x_i) + \Delta t v(t^{k+1/2}, x_i),$$

$$v(t^{k+1}, x_i) = v(t^{k+1/2}, x_i) + \frac{(\Delta t/2)}{\varrho(x_i)} \left[\mathbf{b}(t^{k+1}, x_i) + \sum_{B_\delta(x_i)} \mathbf{f}(t^{k+1}, \eta, \xi) \right]. \tag{2.26}$$

2.6 Convergence rate for the nonlinear bond-based model

Suppose that there exists an exact solution to the peridynamic equation (2.1). Define the error as the difference between the exact and the approximate solutions. For a function $f : \Omega \to \mathbb{R}^d$, the $L^2(\Omega, R^d)$ norm is given by

$$||f||_{L^2} = \left[\int_\Omega |\mathbf{f}(\mathbf{X})|^2 d\mathbf{X} \right]^{1/2}. \tag{2.27}$$

We are interested in the L^2 norm of the error and how the L^2 norm of the error behaves as we change the discrete nodal spacing h. The error estimate for the nonlinear bond-based model considered in this work was developed in [13, 14]. For exact and differentiable solutions, the L^2 norm of the error has been shown to converge to zero at a linear rate in both the spatial discretization h and time step Δt for a fixed horizon.

Consider a fixed horizon δ and three mesh sizes h_1, h_2, h_3 with $p = h_1/h_2 = h_2/h_3 > 1$. Let $u_{h_1}, u_{h_2}, u_{h_3}$ be three solutions corresponding to the three mesh sizes. Assume that the L^2 norm of the error behaves as $||u_h - u_{exact}||_{L^2} \le Ch^\alpha$. Theoretically, we expect $\alpha = 1$, however, α has to be computed from numerical results to validate the theoretical findings. We use $||u_h - u_{exact}||_{L^2} \le Ch^\alpha$ to get

$$||u_{h_1} - u_{h_2}||_{L^2} \le ||u_{h_1} - u_{exact}||_{L^2} + ||u_{h_2} - u_{exact}||_{L^2}$$
$$\le C(h_1^\alpha + h_2^\alpha) \le C(1 + (h_1/h_2)^\alpha)h_2^\alpha = C(1 + p^\alpha)h_2^\alpha,$$
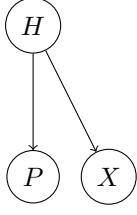
Figure 5: Example dependency graph.

```
1 //Vector with all dependencies of h
2 std::vector<hpx::lcos::future<void>> dependencies;
3 dependcy.push_back(compute(p));
4 dependcy.push_back(compute(x));
5 hpx::wait_all(dependencies);
6 compute(h,p,x);
```

Listing 1: Modeling the dependency graph in Figure 5 with composition in HPX.

where we used triangle inequality for L^2 norm in first step. Taking the logarithm on both sides yield

$$\log\left(||u_{h_1} - u_{h_2}||_{L^2}\right) \leq \log(C) + \log\left(1 + p^\alpha\right) + \alpha\log(h_2) = \bar{C} + \alpha\log(h_2),$$

where $\bar{C} = \log(C) + \log\left(1 + p^\alpha\right)$. Similarly,

$$\log\left(||u_{h_2} - u_{h_3}||_{L^2}\right) \leq \bar{C} + \alpha\log(h_3).$$

The rate of convergence α is bounded below ($\bar{\alpha} \leq \alpha$) by

$$\bar{\alpha} = \frac{\log\left(||u_1 - u_2||_{L^2}\right) - \log\left(||u_2 - u_3||_{L^2}\right)}{\log(p)}. \tag{2.28}$$

$\bar{\alpha}$ can be computed from numerical solutions and its value compared with the theoretically expected value of 1.

2.7 HPX – an open source C++ standard library for parallelism and concurrency

The HPX library [10] is a C++ standard compliant Asynchronous Many Task (AMT) run time system tailored for HPC applications. It provides well-known concepts such as data flow, futurization, and Continuation Passing Style (CPS), as well as new and unique overarching concepts. The combination of these concepts results in a unique model. The concept of futurization and parallel for loops, which are used to synchronize and parallelize, are recalled here.

2.7.1 Futurization

An important concept for synchronization provided by HPX is futurization. The API provides an asynchronous return type `hpx::lcos::future<T>`. This return type, a so-called future, is based on templates and hides the function call return type. The difference here is that the called function immediately returns, even if the return value is not computed. The return value of a future is accessible through the `.get()` operator that is an synchronous call and waits until the return type is computed. The standard-conform API functions `hpx::wait_all`, `hpx::wait_any`, and `hpx::lcos::future<T>::then` are available for combining futures and generate parallel executions graphs [15].

A typical example dependency graph is shown in Figure 5. On the Figure, H depends asynchronously on P and X. Listing 1 provides these dependencies resolutions within HPX. First, a `std::vector` is utilized to collect the dependencies. Second, the computations futures are added to this vector. Note, that the `compute` function returns a future in both cases, immediately, and the computations are executed asynchronously. Third, a barrier with `hpx::wait_all` for the dependencies has to be defined before H can be computed. HPX internally ensures that the function in line 6 is only called when the the previous two futures computations are finished.

9

```
1  //Sequential loop
2  for (size_t i=0,i<n;i++)
3  {
4      h[i] = p[i]+x[i];
5  }
```

Listing 2: C++ code for storing the sum of two vectors sequentially in third vector.

```
1  //Synchronizing parallel for loop
2  hpx::parallel::for_loop(
3  hpx::parallel::execution::par,
4  0,n,[h,p,x](boost::uint64_t i)
5  {
6      h[i] = p[i]+x[i];
7  });
```

Listing 3: HPX equivalent code for storing the sum of two vectors parallel in a third vector.

2.7.2   Parallelization

Consider the addition of n elements for the two vectors `p` and `x`, where the sum is stored piece-wise in vector `h`. Listing 2 shows the sequential approach for this computation while Listing 3 shows the same computational task but the sum is executed in parallel. The `for` loop is replaced with `hpx::parallel::for_loop`. The first argument defines the execution policy while the second and third define the loop range. The last argument is the lambda function, executed in parallel for all i ranging from 0 to n. Note that only two lines of codes are required to execute the code in parallel. The parallel for loop can be combined with futurization for synchronization. Therefore, the parallel execution policy is changed to `hpx::parallel::execution::par(hpx::parallel::execution::task)`.

The future can now be synchronized with other futures. Listing 4 shows an example for synchronization. Vectors p and x should be independently manipulated before the pairwise sum is computed. Therefore, the execution policy is changed and the futures of the manipulations are synchronized with the third future in line 27. Here, the `hpx::wait_all` ensures that the manipulations are finished and `then` describes the sum's dependencies.

3   Implementation of PeridynamicHPX

PeridynamicHPX is an modern C++ code utilizing HPX for parallelism. The classes design is modular and template-based to easily extend the code with new material models. The code's design is presented herein and parallelism and concurrency utilization within HPX is detailed.

3.1   Design with respect to modular expandability

Figure 6 shows the modular design class. PeridynamicHPX contains three modules that are affected by the discretization extensions and material models. First, the *Deck* module handles the loading of the discretization and the configuration in the YAML Ain't Markup Language file format. Each new Deck inherits the common functions from the *AbstractDeck* and is extended with the new problem/material specific attributes.

Second, the abstractions for bond-based and state-based materials are provided in the *Material* module. The nonlinear bond-based elastic material in Section 3.2.1 and the linear state-based elastic material of Section 3.2.2 were implemented. The abstract material must be inherited and the abstract methods, e.g. force and strain, are implemented if a new material model is to be

```
1  std::vector<hpx::lcos::future<void>> dep;
2
3  dep.push_back(hpx::parallel::for_loop(
4  hpx::parallel::execution::par(
5  hpx::parallel::execution::task),
6  0,n,[p](boost::uint64_t i)
7  {
8      p[i] = p[i]+1;
9  });
10 dep.push_back(hpx::parallel::for_loop(
11 hpx::parallel::execution::par(
12 hpx::parallel::execution::task),
13 0,n,[x](boost::uint64_t i)
14 {
15     x[i] = x[i]-1;
16 });)
17
18 hpx::lcos::future f = hpx::parallel::for_loop(
19 hpx::parallel::execution::par(
20 hpx::parallel::execution::task),
21 0,n,[h,p,x](boost::uint64_t i)
22 {
23     h[i] = p[i]+x[i];
24 });
25
26 hpx::wait_all(dep).then(f);
```

Listing 4: Example for the synchronization of three parallel for loopswithin HPX by using the concept of futurization.

implemented.

The different time integration schemes and the discretizations are considered third. All new *Problem* classes inherit their common and abstract functions from *AbstractProblem*. The schemes of Section 2.5.1 and Section 2.5.2 were implemented. Note that a problem implementation takes the abstract classes as arguments. The specific implementation can therefore be used for state-based and bond-based materials.

The design aims to hide most of the HPX specific features and making the code easy for adding new material models by implementing the abstract classes. For new problem classes, the user must to deal with the parallel for loops instead of using the C++ standard for loop. Thus, the code is accessible to users that do not have advanced programming experience in parallel computing, but still yields acceptable scalability without optimization.

3.2 Parallelization with HPX

The implementation of bond-based material and the explicit time integration is adapted from earlier one-dimensional code developed by the second author [14]. The implementation of the state-based material and the implicit time integration is adapted from [22]. These sequential algorithms are analyzed to make use of HPX parallelism. The use of HPX tools to achieve parallelism is discussed in the sequel.

3.2.1 Nonlinear elastic bond-based material

The internal force density and strain energy are computed for a peridynamic NP material, as described in Section 2.3.1. Algorithm 1 shows the use of HPX for computing force and energy. Material parameters are read from the input file and the list of neighbors for each node in the
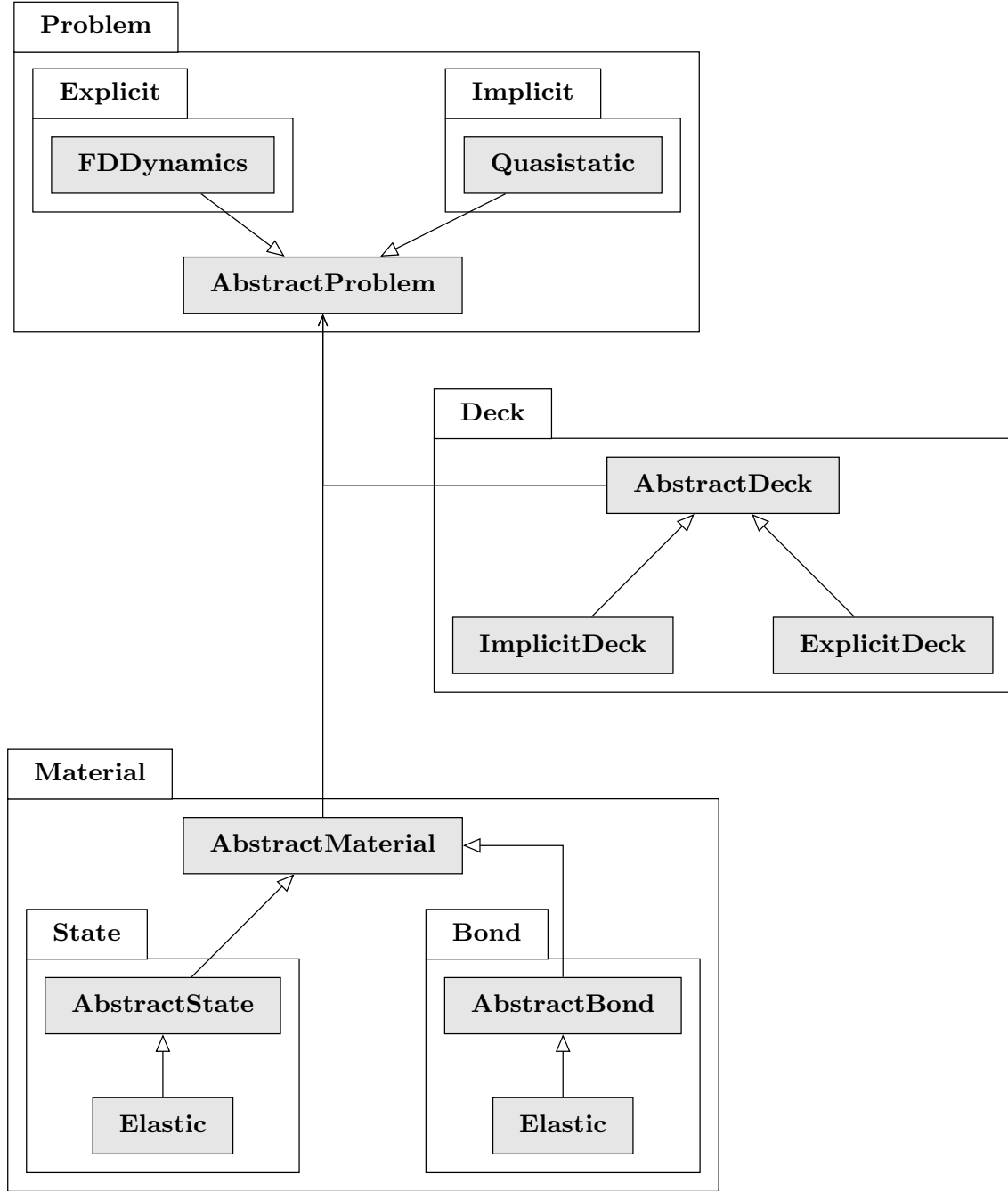
Figure 6: Class diagram of PeridynamicHPX which is designed to easily extend the code with new materials or dicretizations. The functionality of the code is defined in three packages: *Problem* containing the different classes for discretizations, *Deck* which handles the input and output, and *Material* providing the different kind of material models. All packages provide a abstract class which needs to be inherited by all sub classes for extending the code.

Algorithm 1 Computation of internal force for bond-based material given by (2.8).

1: future f1 =
2: **parallel_for** $i < n$ **do** ▷Compute force at mesh nodes
3:     **for** $j \in B_\delta(\mathbf{x}_i)$ **do**
4:         $\xi = \mathbf{x}_j - \mathbf{x}_i$
5:         $S = \frac{\mathbf{u}(x_j) - \mathbf{u}(x_i)}{|\xi|} \cdot \frac{\xi}{|\xi|}$
6:         ▷Compute force at x_i using (2.8)
7:         $\mathbf{f}(\mathbf{x}_i) += \frac{4}{\delta|B_\delta(0)|} J^\delta(|\xi|) \psi'(|\xi|S^2) S \frac{\xi}{|\xi|} V_j$
8:         ▷Compute strain energy at x_i using (2.12)
9:         $U(\mathbf{x}_i) += \frac{1}{|B_\delta(0)|} J^\delta(|\xi|) \psi(|\xi|S^2) V_j/\delta$
10:     **end for**
11: **end**
12: hpx::wait_all(f1)

---

reference configuration is computed at the beginning. The force and energy at each mesh node are computed by adding the forces and energies from the neighboring nodes.

3.2.2 Linearly elastic state-based material

The internal force density and strain energy are computed for a state based elastic peridynamic solid material [30], as described in Section 2.3.2. Algorithm 2 shows the adapted algorithm [22] parallized and synchronized with HPX. First, the weighted volume m is computed for each node in parallel. Second, the dilation θ is computed for each node in parallel. The internal force density and the strain energy can be computed independently from each other. Therefore, the execution policy `hpx::parallel::execution::task` is utilized to obtain futures back of these two loops. Third, the internal force is computed asynchronously. Fourth, the strain energy is computed asynchronously, when needed, *e.g.* for output. A synchronization for these two futures is needed before the force and strain energy are computed in future steps.

3.2.3 Implicit time integration

Figure 7 shows the implicit integration implementation flow chart. The external force $\mathbf{b}$ is updated for each load step s. Next, the residual (2.24) and its norm are computed and compared against the tolerance τ. If the residual is too large, the tangent stiffness matrix (2.23) is assembled, (see Algorithm 3). The displacement of the previous load step was used to assembly the first matrix $\mathbf{K}(u)$. Line 6 perturbs the displacement by $\pm \upsilon$, where υ is infinitesimally small. Line 9 computes the internal forces $f^{\pm \upsilon}$ and Line 13 evaluates the central difference to construct the stiffness matrix $\mathbf{K}(u)$. Note that the nodes neighborhood B_δ is represented and has several zero entries where nodes do not have neighbors. Next, the guessed displacement is updated with the solution from solving $\mathbf{K}\underline{u} = -\mathbf{r}$. The residual is evaluated once and the Newton method is iterated until $|r| \leq \tau$.

The high-performance open-source C++ library Blaze [11, 12] was used for matrix and vector operations. Blaze supports HPX for parallel execution and can be easily integrated. The library BlazeIterative[1] was used for solving $\mathbf{K}\underline{u} = -\mathbf{r}$. The Biconjugate gradient stabilized method (BiCGSTAB) or the conjugate gradient (CG) solver were used for solving.

3.2.4 Explicit time integration

Figure 8 shows the flow chart for the explicit time integration. Algorithm 4 outlines the steps to implement the velocity-verlet scheme (2.26) to obtain the displacement u^{k+1} for the time step $k+1$. Line 3 calls a function of either bond-based Material or state-based Material class to compute the

---

[1]https://github.com/tjolsen/BlazeIterative

Algorithm 2 Computation of internal force and strain energy. Adapted from [22]

1: **parallel_for** $i < n$ **do** ▷Compute weighted volumes using (2.16)
2:     $m_i = 0$
3:     **for** $j \in B_\delta(\mathbf{x}_i)$ **do**
4:         $\xi = \mathbf{x}_j - \mathbf{x}_i$
5:         $m_i + = |\xi|^2 V_j$
6:     **end for**
7: **end**
8: **parallel_for** $i < n$ **do** ▷Compute dilatation using (2.17)
9:     $\theta_i = 0$
10:     **for** $j \in B_\delta(\mathbf{x}_i)$ **do**
11:         $\xi = \mathbf{x}_j - \mathbf{x}_i$
12:         $\eta = \mathbf{u}(\mathbf{x}_j) - \mathbf{u}(x_i)$
13:         $\underline{e} = |\xi + \eta| - |\xi|$
14:         $\theta_i + = {}^3/m_i |\xi| \underline{e} V_j$
15:     **end for**
16: **end**
17: future f1 $=$
18: **parallel_for** $i < n$ **do** ▷Compute internal forces using (2.15)
19:     **for** $j \in B_\delta(\mathbf{x}_i)$ **do**
20:         $\xi = \mathbf{x}_j - \mathbf{x}_i$
21:         $\eta = \mathbf{u}(\mathbf{x}_j) - \mathbf{u}(\mathbf{x}_i)$
22:         $\underline{e}^d = e - {}^{(\theta_i|\xi|)}/3$
23:         $\underline{t} = {}^3/m_i K \theta_i |\xi| + {}^{(15\mu)}/m_i \underline{e}^d$
24:         $\underline{M} = {}^{\eta + \xi}/|\xi + \eta|$
25:         $\mathbf{f}_i + = \underline{t}\underline{M}V_j$
26:         $\mathbf{f}_j - = \underline{t}\underline{M}V_i$
27:     **end for**
28: **end**
29: future f2 $=$
30: **parallel_for** $i < n$ **do** ▷Compute strain energy, if needed
31:     **for** $j \in B_\delta(x_i)$ **do**
32:         <do stuff>
33:     **end for**
34: **end**
35: hpx::wait_all(f1,f2)
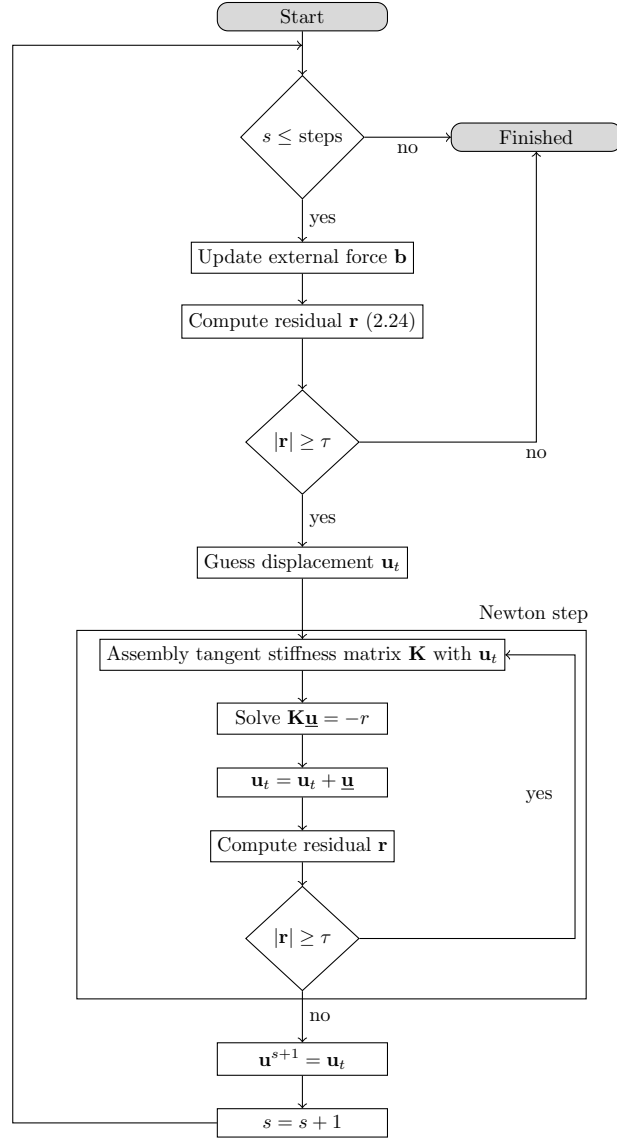
Figure 7: Flow chart of the implicit time integration scheme adapted from [22]. For each time step s the external force $\mathbf{b}$ is updated and the residual $\mathbf{r}$ is evaluated. When the norm of the residual is larger than the tolerance τ the displacement $\mathbf{u}^{s+1}$ is obtained via a Newton step.

15
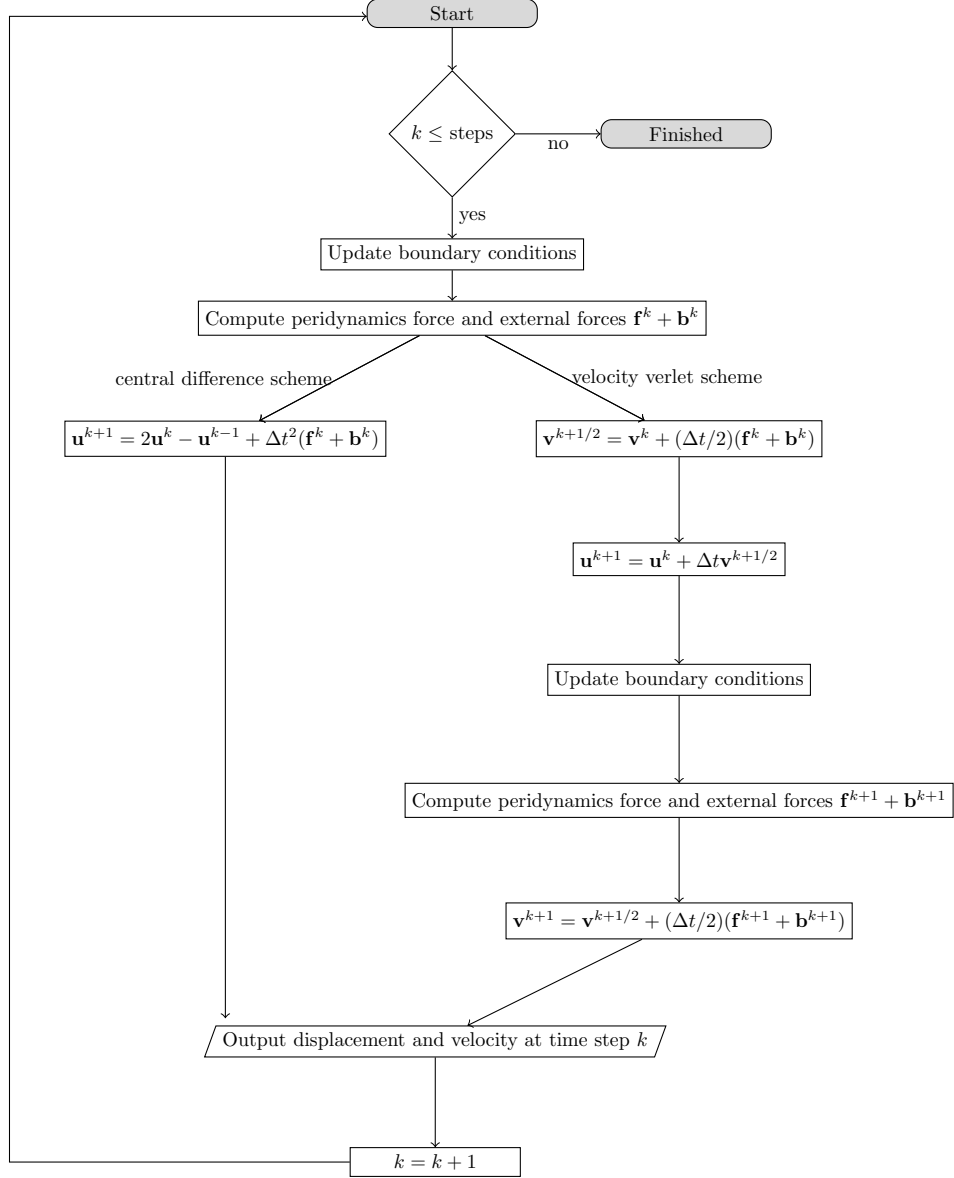
Figure 8: Flow chart of the explicit time integration scheme. For each time step k the boundary conditions are updated and the internal and external forces are computed. Depending on a central difference scheme or a velocity scheme the displacement $\mathbf{u}^{k+1}$ and velocity $\mathbf{v}^{k+1}$ is obtained.

Algorithm 3 Assembly of the tangent stiffness matrix by central finite difference. Adapted from [22]

1: $\mathbf{K}^{d\cdot n \times d\cdot n} = 0$ ▷Set matrix to zero
2: **parallel_for** $i < n$ **do**
3:     **for** $i \in \{B_\delta(\mathbf{x}_i), i\}$ **do**
4:         ▷Evaluate force state under perturbations of displacement
5:         **for each** displacement degree of freedom r at node j **do**
6:             $\underline{T}[\mathbf{x}_i](\mathbf{u} + v^r)$
7:             $\underline{T}[\mathbf{x}_i](\mathbf{u} - v^r)$
8:             **for** $k \in B_\delta(\mathbf{x}_i)$ **do**
9:                 $\mathbf{f}^{v+} = \underline{T}^{v+}\langle\mathbf{x}_k - \mathbf{x}_i\rangle V_i V_j$
10:                $\mathbf{f}^{v-} = \underline{T}^{v-}\langle\mathbf{x}_k - \mathbf{x}_i\rangle V_i V_j$
11:                $\mathbf{f}_{\text{diff}} = \mathbf{f}^{v+} - \mathbf{f}^{v-}$
12:                **for each** degree of freedom s at node k **do**
13:                    $\mathbf{K}_{sr}+ = f_{\text{diff}_s}/2v$
14:                **end for**
15:             **end for**
16:         **end for**
17:     **end for**
18: **end**
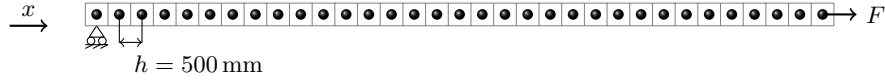
Figure 9: Sketch of the one dimensional bar benchmark test. The node on the left-hand side is clamped and its displacement is set to zero. A force F is applied on the node at the right-hand side.

forces and energies corresponding to displacement u^k. The velocity-verlet algorithm is used to compute the velocity at $k + 1/2$ and displacement u^{k+1}. Line 12 invokes the Material class again to compute the forces at new displacements u^{k+1}. The velocity at $k + 1$ is computed with the updated forces.

4 Validation of PeridynamicHPX

In this section we demonstrate the convergence of HPX implementations for both implicit and explicit schemes.

4.1 Implicit

4.1.1 One dimensional tensile test

Consider the simple geometry of Figure 9 for comparing the one dimensional implicit time integration against a classical continuum mechanics (CCM) solution. The node on the left-hand side is clamped and displacement is set to zero. A force F is applied to the node at the right-hand side.

The strain, stress, and strain energy for this configuration are compared with the values obtained from classical continuum mechanics (CCM) where $\sigma = E\cdot\varepsilon$, where σ is the stress, E is the Young's modulus and ε is the strain. The stress $\sigma = F/S$, is the defined by the force F per cross section S. Thus, the strain is obtained by $\varepsilon = \sigma/E = F/(SE)$. For a force F of 40 N, a cross section of $1\,\text{mm}^2$ and a Young's modulus of $4\,\text{GPa}$, the resulting strain reads $\varepsilon_{\text{CCM}} = 0.01$ an the stress is of

17

Algorithm 4 Time integration using velocity-verlet scheme

1: ▷Loop over time steps
2: **for** $0 \leq k < N$ **do**
3:     Compute peridynamic force f^k, body force $\mathbf{b}^k$, external force $\mathbf{f}^k_{ext}$, and total energy U^k_{total}
4:
5:     **parallel_for** $i < n$ **do**
6:         ▷Compute velocity
7:         $\mathbf{v}^{k+1/2}(\mathbf{x}_i) = \mathbf{v}^k(x_i) + (\Delta t/2)(\mathbf{f}^k(\mathbf{x}_i) + \mathbf{b}^k(\mathbf{x}_i) + \mathbf{f}^k_{ext}(\mathbf{x}_i))$
8:         ▷Compute displacement
9:         $\mathbf{u}^{k+1}(\mathbf{x}_i) = \mathbf{u}^k(\mathbf{x}_i) + \Delta t \mathbf{v}^{k+1/2}\mathbf{f}(\mathbf{x}_i)$
10:     **end**
11:     Update boundary condition for time $t = (k+1)\Delta t$
12:     Compute $\mathbf{f}^{k+1}$, $\mathbf{b}^{k+1}$, $\mathbf{f}^{k+1}_{ext}$, and U^{k+1}_{total}
13:     **parallel_for** $i < n$ **do** ▷Loop over nodes
14:         ▷Compute velocity
15:         $\mathbf{v}^{k+1}(x_i) = \mathbf{v}^{k+1/2}(\mathbf{x}_i) + (\Delta t/2)(\mathbf{f}^{k+1}(\mathbf{x}_i) + \mathbf{b}^{k+1}(x_i) + \mathbf{f}^{k+1}_{ext}(\mathbf{x}_i))$
16:     **end**
17:
18:     ▷Check if Δt corresponds to stable time step
19:     **if** $U^{k+1}_{total} > U^k_{total}$ **then**
20:         ▷Output message
21:         Stop and decrease time step Δt
22:     **end if**
23: **end for**

$\sigma_{\mathrm{CCM}} = 40\,\mathrm{MPa}$. The strain energy density is given by $U_{\mathrm{CCM}} = \sigma^2/(2E) = 200\,\mathrm{kPa}$.

The bar was discretized with 33 nodes with a nodal spacing h of $500\,\mathrm{mm}$. The horizon was $\delta = 1000\,\mathrm{mm}$. The tolerance τ for the Biconjugate gradient stabilized method (BiCGSTAB) was 1×10^{-9}. Figure 10 shows that stresses, strains and strain energy match perfectly the theoretical values inside the bar but all these quantities diverge at the boundaries. These effects are the well-known surface effects within the EMU nodal discretization.

4.1.2 Two dimensional tensile test

Figure 11 shows the two-dimensional tensile benchmark. The line of nodes of the right-hand side of the plate are clamped in x-direction and y-direction. On each of the nodes of the line at the left-hand side a force of force $F = -50\,\mathrm{kN}$ in x-direction was applied. The displacement of a node $\mathbf{x}_i$ for a tensile behavior can be determined with CCM as follows

$$u_x(x_i) = \frac{F}{E \cdot W \cdot H} x_{i_x} - \frac{F}{E \cdot W \cdot H}, \tag{4.1}$$
$$u_y(x_i) = \frac{-F \cdot \nu}{E \cdot W \cdot H} x_{i_y} - \frac{F \cdot \nu}{E \cdot W \cdot H}.$$

where F is applied and W and H are respectively the plate's width and height.

H and W were set to $375\,\mathrm{mm}$ and $h = 25\,\mathrm{mm}$. The tolerance for the BiCGSTAB solver was $\tau = 1 \times 10^{-3}$. The m-value was, 4, which means that $2m + 1$ nodes are within $[x_i - \delta, x_i + \delta]$. Table 1 lists the actual position at the node in the center of the plate x_{mid} and its comparison with the one from CCM (4.1). The relative error for the actual position in x-direction is sufficiently small. With the applied boundary conditions the displacement at the point in the center of the plate in y-direction is zero.
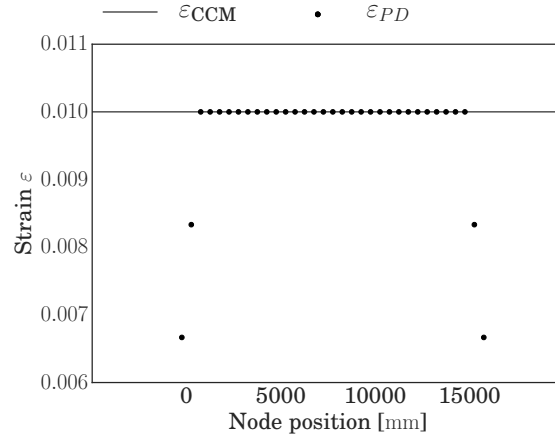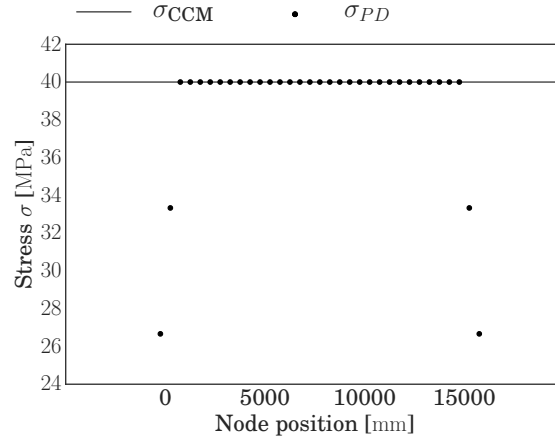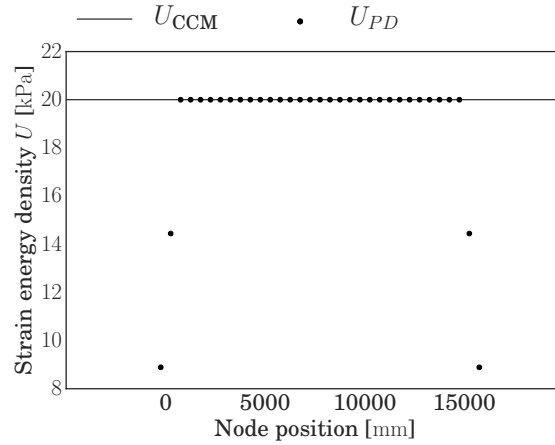
Figure 10: Comparison of strain ϵ (a), stress σ (b), and strain energy U (c) with classical continuum mechanics. Close to the boundary the surface effect influences the accuracy.
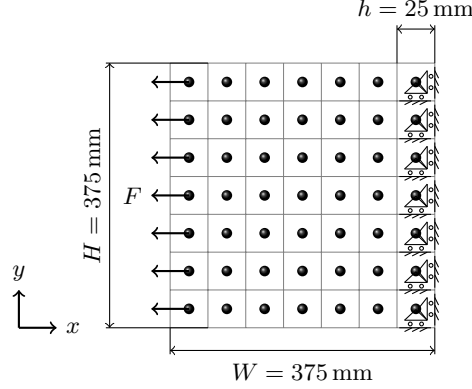
Figure 11: Sketch of the two dimensional tensile test. All nodes on the line of the right-hand side of the plate are clamped in x-direction and y-direction. A force of $-50\,\text{kN}$ is applied in $x - direction$ to each node of the line on the left-hand side.

Actual position	CCM	PD	Relative error
x-direction	$-1.574\,96\,\text{m}$	$-1.5744\,\text{m}$	-3×10^{-4}
y-direction	$-1.575\,\text{m}$	$-1.575\,\text{m}$	0

Table 1: Comparison of the actual position in meters of the node in the center of the plate obtained in the simulation with those from classical continuum mechanics (4.1).

4.2 Explicit

The theoretical convergence presented in Section 2.6 is now compared to actual HPX implementations.

4.2.1 One dimensional

The linearized bond-based peridynamic force is given by

$$\mathbf{f}(t, u(t, X') - u(t, X), X' - X)) =$$
$$\frac{4}{\delta|B_\delta(X)|} \int_{B_\delta(X) \cap \Omega_0} J^\epsilon(|X' - X|)\psi'(0)S(X', X, u(t)) \frac{X' - X}{|X' - X|} dX'. \tag{4.2}$$

Here the non-linear force model defined by the potential function ψ' is linearized by retaining the first term in its taylor series about zero. For convenience, we refer to, the peridynamics solution with, a linearized force as LP and that with a nonlinear force as NP.

Here we take $\psi(r) = (1 - \exp[-r])$. The influence function is of the form: $J^\delta(r) = J(r/\delta)$, where $J(r) = c_1 r \exp(-r^2/c_2)$, $c_1 = 1$, and $c_2 = 0.4$.

Let $\Omega = [0, 1]$ be the material domain with an horizon $\delta = 0.01$. The time domain is $[0, 2]$ with a time step $\Delta t = 10^{-5}$. Consider four mesh sizes $h_1 = \delta/2$, $h_2 = \delta/4$, $h_3 = \delta/8$, and $h_4 = \delta/16$, and compute equation (2.28) for two sets $\{h_1, h_2, h_3\}$ and $\{h_2, h_3, h_4\}$ of mesh sizes. The boundary conditions are those described in Figure 12. Apply either one of the initial conditions:

Initial condition 1(IC 1): Let the initial condition on the displacement u_0 and the velocity v_0 be given by

$$u_0(X) = \exp[-|X - x_c|^2/\beta]a, \quad v_0(X) = 0,$$

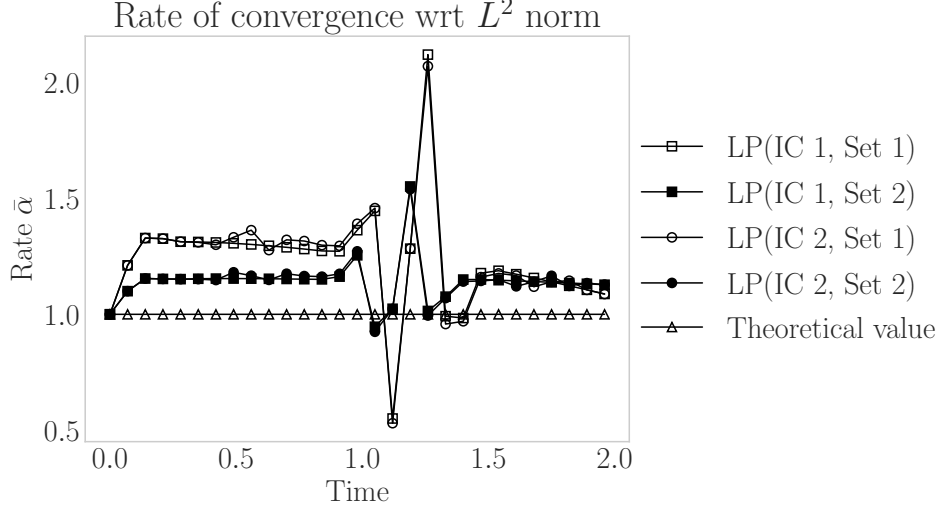with $x_c = 0.5, a = 0.001, \beta = 0.003$. u_0 is the Gaussian function centered at x_c.

20

Figure 12: Time vs rate of convergence with respect to mesh size. These results are for linear peridynamic force (LP). Set 1 corresponds to convergence rate obtained from solutions of mesh sizes $\{h_1, h_2, h_3\}$ and set 2 corresponds to convergence rate obtained from solutions of mesh sizes $\{h_2, h_3, h_4\}$. The boundary condition is $u = 0$ on non-local boundary $\Omega_c = [-\delta, 0] \cup [1, 1+\delta]$. These results validate the implementation of the explicit scheme for peridynamics in one dimension.

Initial condition 2(IC 2): The initial condition u_0 and v_0 are described as

$$u_0(X) = \exp[-|X - x_{c1}|^2/\beta]a + \exp[-|X - x_{c2}|^2/\beta]a, \quad v_0(X) = 0,$$

with $x_{c1} = 0.25, x_{c2} = 0.75, a = 0.001, \beta = 0.003$. u_0 is the sum of two Gaussian functions centered at x_{c1} and x_{c2}.

Figure 12 shows the rate of convergence $\bar{\alpha}$ as a function of time for solutions having the initial conditions 1 and 2. The convergence rate for $\{h_1, h_2, h_3\}$ and $\{h_2, h_3, h_4\}$ follows the same trend. The bump in the plot is due to wave reflecting at the boundary. For rapidly varying (spatially) displacement field, the length scale at which displacement varies is small, and to capture it more accurately the mesh size should be smaller than the length scale associated to the displacement field. We, thus, see from the plot of rate that near wave reflection time the rate for finer mesh is closer to the theoretical rate of 1. Similar results, not shown here, were obtained for the nonlinear model of equation (2.1). The convergence results presented in Figure 12 agree with the theoretical convergence rate, which suggests that the implementation is robust.

4.2.2 Two dimensional

Let $\Omega = [0,1]^2$ be the material domain with a horizon $\delta = 0.1$. The 2-d vector $\boldsymbol{X}$ is written $\boldsymbol{X} = (X_1, X_2)$ where X_1 and X_2 are the components along the x and y axes. The time domain is taken to be $[0,2]$ and $\Delta t = 10^{-5}$ is the time step. The influence function is $J^\delta(r) = 1$ for $r < \delta$ and 0 otherwise. The rate $\bar{\alpha}$ is computed for three mesh sizes $h = \delta/2, \delta/4, \delta/8$. Both Dirichlet boundary conditions as well as mixed of Dirichlet and Neumann boundary conditions are used in the convergence analysis.

1. Dirichlet formulation: Let the initial condition on displacement vector $\boldsymbol{u}_0 = (u_{0,1}, u_{0,2})$ and velocity vector $\boldsymbol{v}_0 = (v_{0,1}, v_{0,2})$ be

$$u_{0,1}(X_1, X_2) = \exp[-(|X_1 - x_{c,1}|^2 + |X_2 - x_{c,2}|^2)/\beta]a_1,$$
$$u_{0,2}(X_1, X_2) = \exp[-(|X_1 - x_{c,1}|^2 + |X_2 - x_{c,2}|^2)/\beta]a_2,$$
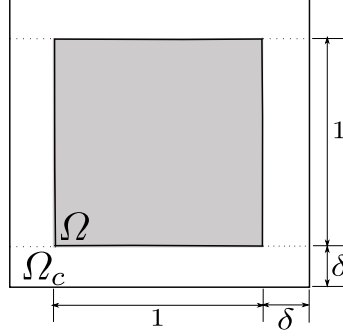$$v_{0,1}(X_1, X_2) = 0, v_{0,2}(X_1, X_2) = 0,$$

Figure 13: Square domain (grey area) $\Omega = [0,1]^2$ and a nonlocal boundary $\Omega_c = [-\delta, 1+\delta]^2 - [0,1]^2$. The area outside Ω and within the outer boundary is Ω_c. Dashed lines show the division of Ω_c into left, right, bottom, and top parts.

where $\boldsymbol{a} = (a_1, a_2)$ and $\boldsymbol{x}_c = (x_{c,1}, x_{c,2})$ are 2-d vectors and β is a scalar parameter. Two different types of initial conditions are considered:

    Initial condition 1(IC 1): $\boldsymbol{a} = (0.001, 0)$, $\boldsymbol{x}_c = (0.5, 0.5)$, and $\beta = 0.003$.
    Initial condition 2(IC 2): $\boldsymbol{a} = (0.0002, 0.0007)$, $\boldsymbol{x}_c = (0.25, 0.25)$, and $\beta = 0.01$.

    In Figure 14, the nonlocal Dirichlet boundary condition $\boldsymbol{u} = (0,0)$ is applied over the set Ω_c. Here Ω_c is the collar set $[-\delta, 1+\delta]^2 - [0,1]^2$ of $[0,1]^2$, see Figure 13. Figure 14 shows $\bar{\alpha}$ with respect to time for the nonlinear (NP) and linear (LP) peridynamics solutions.

    2. Mixed Dirichlet and Neumann formulation: We consider zero initial conditions for both the initial displacement $\boldsymbol{u}_0 = (u_{0,1}, u_{0,2})$ and velocity $\boldsymbol{v}_0 = (v_{0,1}, v_{0,2})$. On the bottom part of $\Omega_c = [-\delta, 1+\delta]^2 - [0,1]^2$, the Dirichlet boundary condition is given by $\boldsymbol{u} = (0,0)$. On the left and right components of Ω_c, u_1 is set to zero and a zero force is applied along Y-axis. On the top component of Ω_c we apply a force $\boldsymbol{b}(t) = (b_1(t), b_2(t))$ of the form:

$$b_1(t) = 0, \qquad b_2(t) = \sin(10 * t). \tag{4.3}$$

The boundary conditions given in this example are a mix of Dirichlet and Neumann boundary conditions. Figure 15 shows $\bar{\alpha}$ with respect to time for the nonlinear (NP) peridynamics solution.

    The rate of 1 was proved theoretically for the Dirichlet type formulation, see [13, 14], and the results in Figure 14 and Figure 15 validate the theoretical study.

5 Benchmark for PeridynamicHPX

5.1 Implicit

The benchmark is realized by comparing the the computational time to the theoretical complexity. The theoretical complexity is the number of operations an algorithm requires to perform its task. The computational time is the measured physical time the program required to complete its tasks.

The algorithm in Figure 7 features several loops for which the maximal amount of iterations can be estimated with $\mathcal{O}(n^2)$, with n being the number of discrete nodes. The computational complexity of the conjugate gradient (CG) solver for k iterations can be estimated with $\mathcal{O}(kn^2)$. The complexity for the computational time can be approximated with $\mathcal{O}(\frac{n^2+km}{p})$, where p is the amount of CPUs.

    The test case of Section 4.1.2 served as benchmark for the two dimensional implicit time integration. Figure 17 shows the 20436 nonzero entries of the tangent stiffness matrix $K^{360 \times 360}$ with the condition number $\kappa(K) = 90.688$. The solver required 28 iterations. The benchmark was run on Fedora 25 with kernel 4.8.10 on Intel(R) Xeon(R) CPU E5-1650 v4 @ 3.60GHz with up to
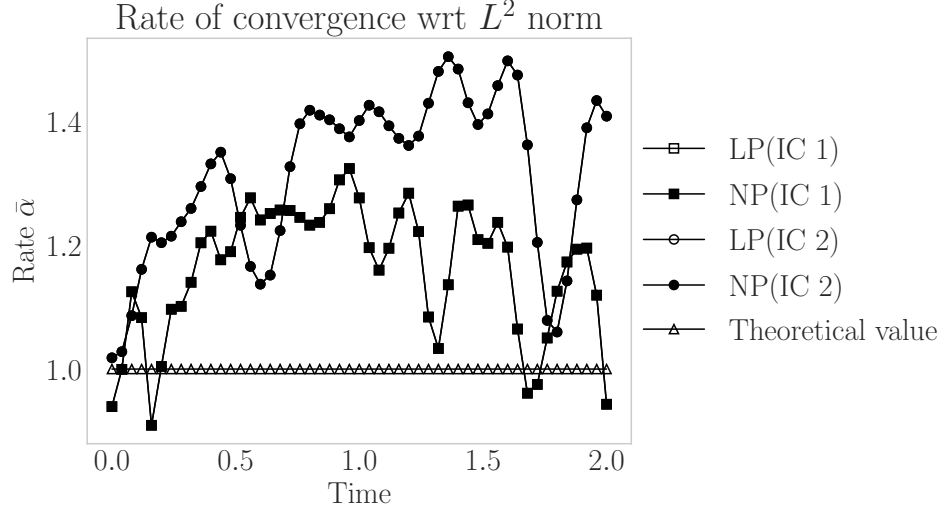
Figure 14: Time vs rate of convergence with respect to mesh size. Results are for the Dirichlet type formulation with two different initial conditions (IC 1 and IC 2) described in section 4.2.2. The boundary condition is $\boldsymbol{u} = (0,0)$ on the non-local boundary $\Omega_c = [-\delta, 1 + \delta]^2 - [0,1]^2$. The rate of convergence is similar for linear (LP) and nonlinear (NP) peridynamics. These results validate the implementation of the explicit scheme in two dimension.
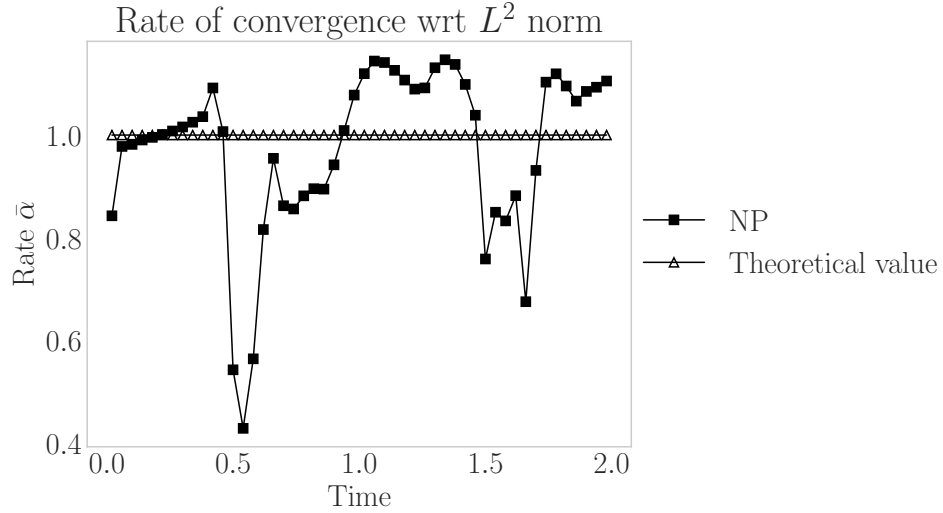

Figure 15: Time vs rate of convergence with respect to the mesh size for Dirichlet and Neumann mixed boundary condition, see description in section 4.2.2. The square domain is pulled by a sinusoidal external force on top, see (4.3), along the y-axis. The result correspond to the nonlinear (NP) peridynamics.

6 physical cores. HPX (bd2f240 b1565b4) and PeridynamicHPX were compiled with gcc 6.2.1, boost 1.61 and blaze 3.2 libraries were used.

Figure 16a shows the obtained computational time for up to 6 CPUs. The figure shows that the computational time and theoretical complexity followed similar trends. Note that only parallel for loops, synchronization, and futurization were utilized for parallelization.

Figure 16b shows the CPUs idle-rates. The idle-rate is obtained with the performance counters within HPX and measured, in percentage, how long the CPU was idling with respect to the overall computation time. Here, the idle-rate is 0.01%, for the default execution policy. These observations suggest that the implicit time integration seems to scale with the same trend as the theoretical complexity without any code optimization. More sophisticated execution policies (*e.g.* dynamic or adaptive chunk size) could be applied, to decrease the computational time.

5.2 Explicit

The setup presented in Section 4.2 was discretized with 196249 nodes and an horizon of $0.05\,\mathrm{m}$ and $m = 20$ as chosen. The benchmark was run on CentOS 7 with kernel 3.10.0 on Intel(R) Xeon(R) CPU E5-2690 @ 2.90GHz. HPX ($82f7b281$) and PeridynamicHPX were compiled with gcc 7.2, boost 1.61 and blaze 3.2 libraries.

Figure 18a shows the measured computational time for up to 8 CPUs. The algorithm in Figure 8 combines several loops over twice the amount of nodes n, which can be estimated by $\mathcal{O}(n^2)$ and for parallel for loops by $\mathcal{O}(\frac{n^2}{p})$. The computational time shows the same behavior as the theoretical complexity.

Figure 18b shows the idle rate. The idle rate is independent of the amount of CPUs and work is well distributed with the default execution policy.

6 Conclusion

Bond-based and state-based elastic peridynamic material models and implicit and explicit time integration schemes were implemented within a asynchronous many task run time system. These run time systems, like HPX, are essential for utilizing the full performance of the cluster with many cores on a single node.

One important part of the design was the modular expandability for the extensions. New material models can be integrated into the code by inheriting the functions of an abstract class. Consequently, only the material specific functionality, like forces or strain, is provided by the user and implementation details for the parallelism and concurrency are hidden as much as passable by the user. Additional HPX-related functionality needs to be considered for the extension to other integration schemes.

Materials models and the different time integration schemes were validated against theoretical solutions and classical continuum mechanics. All are in good agreement with reference solutions. The convergence rate was shown to be closer to theoretical value, which suggests that the code behaves as expected. The solutions converge to the exact solution at a rate.

The code scaling with respect to computational resource is important and our benchmark results show that the scaling achieved is very close to the theoretical estimates. Both integration schemes were compared against theoretical estimations. The trend of the theoretical estimates fits with measured computational time and both integration schemes scale with increasing amounts
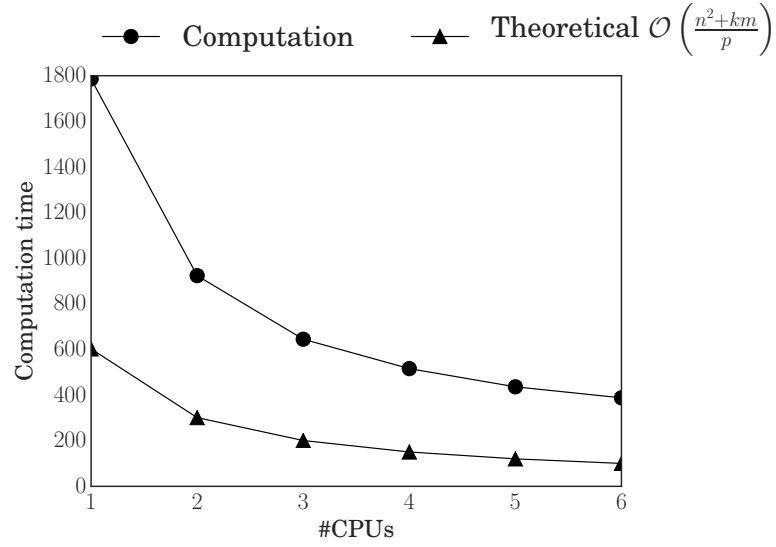
(a) Computational time with increasing amount of CPUs for the test case in Section 4.1.2.
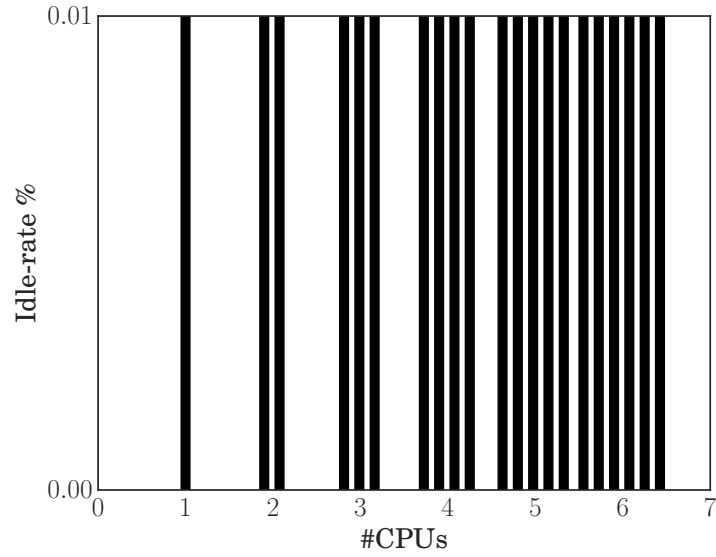
(b) Idle rate with increasing CPUs for the test case in Section 4.1.2.

Figure 16: The computational time with increasing increasing amount of CPUs (a) and the idle-rate of each CPU which means the percentage of overall computational time a CPU did not do any work (b).
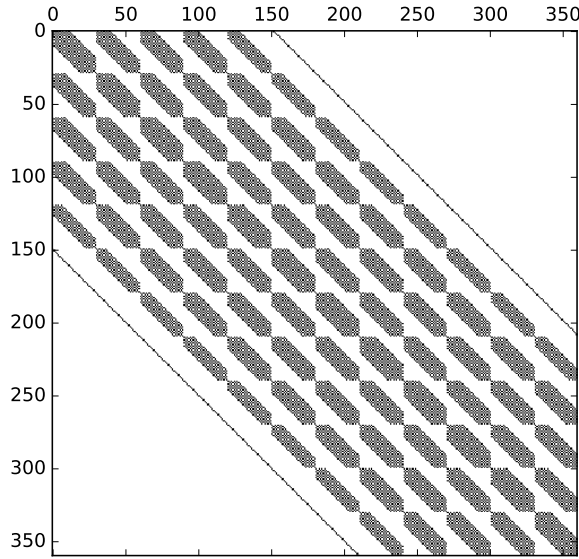
Figure 17: Nonzero matrix elements (20436) of the tangent stiffness matrix K with the condition number $\kappa(K) = 90.688$.

of CPUs. These results were obtained by the default execution policies without any optimization.

References

[1] Aksoylu, B., and Unlu, Z. Conditioning analysis of nonlocal integral operators in fractional sobolev spaces. *SIAM Journal on Numerical Analysis 52* (2014), 653–677.

[2] Chen, X., and Gunzburger, M. Continuous and discontinuous finite element methods for a peridynamics model of mechanics. *Computer Methods in Applied Mechanics and Engineering 200*, 9 (2011), 1237–1250.

[3] Diehl, P. Implementierung eines Peridynamik-Verfahrens auf GPU. Diplomarbeit, Institute of Parallel and Distributed Systems, University of Stuttgart, 2012.

[4] Emmrich, E., Lehoucq, R. B., and Puhst, D. Peridynamics: a nonlocal continuum theory. In *Meshfree Methods for Partial Differential Equations VI*. Springer, 2013, pp. 45–65.

[5] Emmrich, E., and Weckner, O. The peridynamic equation and its spatial discretisation. *Mathematical Modelling and Analysis 12*, 1 (2007), 17–27.

[6] Galvanetto, U., Mudric, T., Shojaei, A., and Zaccariotto, M. An effective way to couple fem meshes and peridynamics grids for the solution of static equilibrium problems. *Mechanics Research Communications 76* (2016), 41–47.

[7] Gerstle, W., Sau, N., and Silling, S. Peridynamic modeling of concrete structures. *Nuclear engineering and design 237*, 12 (2007), 1250–1258.

[8] Ghajari, M., Iannucci, L., and Curtis, P. A peridynamic material model for the analysis of dynamic crack propagation in orthotropic media. *Computer Methods in Applied Mechanics and Engineering 276* (2014), 431–452.
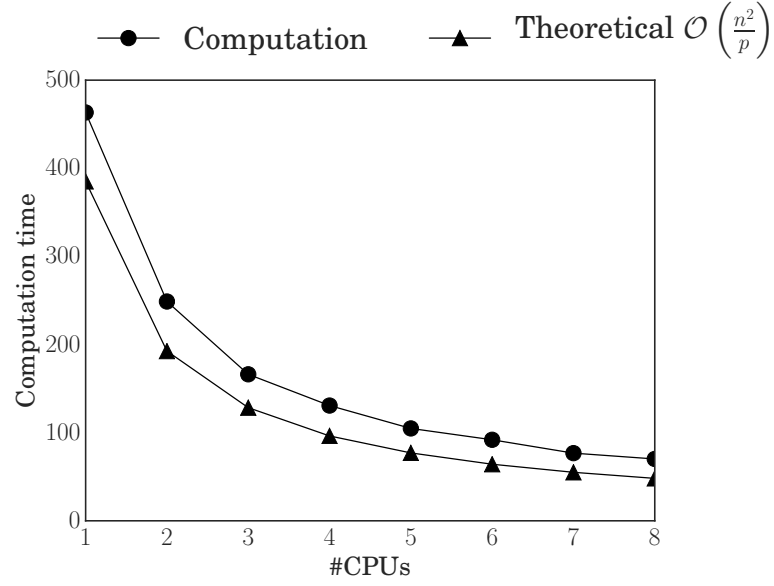
(a) Computational time with increasing CPUs for the test case in Section 4.2.
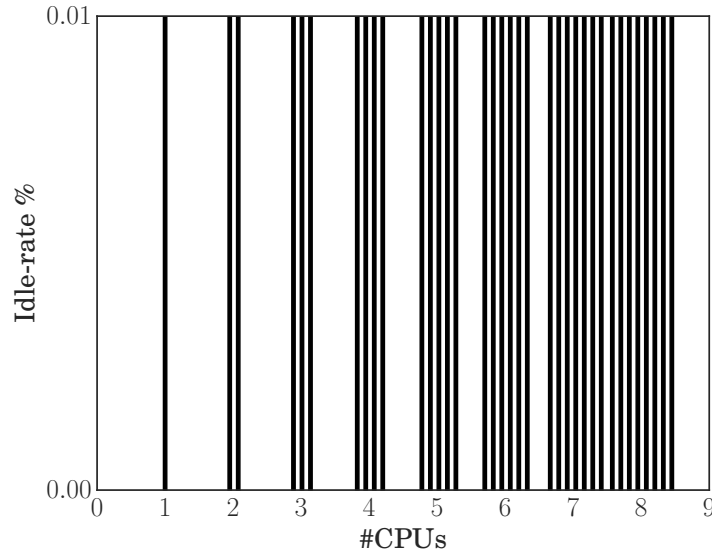
(b) Idle rate with increasing amount of CPUs for the test case in Section 4.2.

Figure 18: The computational time with increasing increasing amount of CPUs (a) and the idle-rate of each CPU which means the percentage of overall computational time a CPU did not do any work (b).

[9] HA, Y. D., AND BOBARU, F. Studies of dynamic crack propagation and crack branching with peridynamics. *International Journal of Fracture 162*, 1-2 (2010), 229–244.

[10] HELLER, T., DIEHL, P., BYERLY, Z., BIDDISCOMBE, J., AND KAISER, H. HPX – An open source C++ Standard Library for Parallelism and Concurrency. In *Proceedings of OpenSuCo 2017, Denver , Colorado USA, November 2017 (OpenSuCo17)* (2017), p. 5.

[11] IGLBERGER, K., HAGER, G., TREIBIG, J., AND RÜDE, U. Expression templates revisited: a performance analysis of current methodologies. *SIAM Journal on Scientific Computing 34*, 2 (2012), C42–C69.

[12] IGLBERGER, K., HAGER, G., TREIBIG, J., AND RÜDE, U. High performance smart expression template math libraries. In *2012 International Conference on High Performance Computing Simulation (HPCS)* (July 2012), pp. 367–373.

[13] JHA, P. K., AND LIPTON, R. Numerical analysis of peridynamic models in holder space. *SIAM J. Numer. Anal. 56*, 2 (2018), 906–941.

[14] JHA, P. K., AND LIPTON, R. Numerical convergence of nonlinear nonlocal continuum models to local elastodynamics. *International Journal for Numerical Methods in Engineering* (March 2018), DOI: 10.1002/nme.5791.

[15] KAISER, H., HELLER, T., BOURGEOIS, D., AND FEY, D. Higher-level parallelization for local and distributed asynchronous task-based programming. In *Proceedings of the First International Workshop on Extreme Scale Programming Models and Middleware* (New York, NY, USA, 2015), ESPM '15, ACM, pp. 29–37.

[16] KILIC, B., AND MADENCI, E. Coupling of peridynamic theory and finite element method. In *50th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference 17th AIAA/ASME/AHS Adaptive Structures Conference 11th AIAA No* (2009), p. 2395.

[17] KUNIN, I. A. *Elastic Media with Microstructure I: One-Dimensional Models (Springer Series in Solid-State Sciences)*, softcover reprint of the original 1st ed. 1982 ed. Springer, 12 2011.

[18] KUNIN, I. A. *Elastic Media with Microstructure II: Three-Dimensional Models (Springer Series in Solid-State Sciences)*, softcover reprint of the original 1st ed. 1983 ed. Springer, 1 2012.

[19] LIPTON, R. Dynamic brittle fracture as a small horizon limit of peridynamics. *Journal of Elasticity 117*, 1 (2014), 21–50.

[20] LIPTON, R. Cohesive dynamics and brittle fracture. *Journal of Elasticity 124*, 2 (2016), 143–191.

[21] LITTLEWOOD, D. J. Simulation of dynamic fracture using peridynamics, finite element modeling, and contact. In *Proceedings of the ASME 2010 International Mechanical Engineering Congress and Exposition (IMECE)* (2010).

[22] LITTLEWOOD, D. J. Roadmap for Peridynamic Software Implementation. Tech. Rep. 2015-9013, Sandia National Laboratories, 2015.

[23] MOSSAIBY, F., SHOJAEI, A., ZACCARIOTTO, M., AND GALVANETTO, U. Opencl implementation of a high performance 3d peridynamic model on graphics accelerators. *Computers & Mathematics with Applications 74*, 8 (2017), 1856 – 1870.

[24] PARKS, M., LITTLEWOOD, D., MITCHELL, J., AND SILLING, S. Peridigm Users Guide. Tech. Rep. SAND2012-7800, Sandia National Laboratories, 2012.

[25] Parks, M. L., Lehoucq, R. B., Plimpton, S. J., and Silling, S. A. Implementing peridynamics within a molecular dynamics code. *Computer Physics Communications 179*, 11 (2008), 777–783.

[26] Silling, S. A. Reformulation of elasticity theory for discontinuities and long-range forces. *Journal of the Mechanics and Physics of Solids 48*, 1 (2000), 175–209.

[27] Silling, S. A. Reformulation of elasticity theory for discontinuities and long-range forces. *Journal of the Mechanics and Physics of Solids 48*, 1 (2000), 175–209.

[28] Silling, S. A. Peridynamic modeling of the kalthoff-winkler experiment. Tech. rep., 2002.

[29] Silling, S. A., and Askari, E. A meshfree method based on the peridynamic model of solid mechanics. *Computers & structures 83*, 17 (2005), 1526–1535.

[30] Silling, S. A., Epton, M., Weckner, O., Xu, J., and Askari, E. Peridynamic states and constitutive modeling. *Journal of Elasticity 88*, 2 (2007), 151–184.

[31] Silling, S. A., and Lehoucq, R. B. Convergence of peridynamics to classical elasticity theory. *Journal of Elasticity 93*, 1 (2008), 13–37.

[32] Weckner, O., and Emmrich, E. Numerical simulation of the dynamics of a nonlocal, inhomogeneous, infinite bar. *Journal of Computational and Applied Mechanics 6*, 2 (2005), 311–319.

[33] Zaccariotto, M., Mudric, T., Tomasi, D., Shojaei, A., and Galvanetto, U. Coupling of fem meshes with peridynamic grids. *Computer Methods in Applied Mechanics and Engineering 330* (2018), 471 – 497.

[34] Zhang, G., Le, Q., Loghin, A., Subramaniyan, A., and Bobaru, F. Validation of a peridynamic model for fatigue cracking. *Engineering Fracture Mechanics 162* (2016), 76–94.