

# MESH PARTITIONING FOR HPX PARALLEL COMPUTING FOR NONLOCAL COMPUTATIONAL MODELS

PRASHANT K. JHA<sup>†</sup>

*Oden Institute for Computational Engineering and  
Sciences,  
The University of Texas at Austin,  
Austin, TX 78712, USA*

PATRICK DIEHL<sup>‡</sup>

*Center for Computation and Technology,  
Louisiana State University,  
Baton Rouge, LA 70803 USA*

In this document, we consider a general class of non-local computational model seen in various fields, such as Peridynamics [21, 4, 14, 1, 12, 8, 19, 16, 17], discrete element method [6, 7, 13, 20, 9], Peridynamics plus discrete element method for granular media [26, 3], nonlocal cell-cell adhesion in computational biology [2, 11, 22], non-local heat equation [5, 10]. References above are only minuscule fraction of what is available. The application of nonlocal modeling method for understanding of complex spatially multiscale phenomenon is seen in various new fields such as fluid mechanics, particulate media, directed self-assembly of Block-Copolymers, tumor modeling, etc. What is more interesting and also convenient is that underlying algorithm which is used to numerically solve these nonlocal models is more or less same and therefore the efficient computational method available for one type of nonlocal model can be easily applied to the nonlocal model in other fields.

It is our understanding that efficient computational method to solve the nonlocal model is very important. It is widely known that the nonlocal models are much more computationally demanding compared to their counterpart, namely local models (such as heat equation, wave equation, etc). This is because the length of interaction in nonlocal model is typically 3 to 10 times larger than the size of discretization and therefore, the usual local assembly of matrix and vector considered in solution of pdes (partial differential equation) is not feasible. The efficient parallel implementation because of long-range interaction is difficult due to dependence over much larger length scale.

---

*E-mail addresses:* (†) [pjha.sci@gmail.com](mailto:pjha.sci@gmail.com), (‡) [pdiehl@cct.lsu.edu](mailto:pdiehl@cct.lsu.edu).

In this document we present this problem using simple example of nonlocal diffusion equation, see [5] and references therein for more information. Our main goal is to highlight key difficulties in designing massively parallel scheme for nonlocal models while keeping the model related complexity to minimum. For this purpose non-local heat equation is suitable. Further, to fix the ideas we only consider two dimensional setting.

In this work we use parallel API called HPX. HPX is an open source asynchronous many task run time system that focuses on high performance computing [15, 23, 18]. 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 load balancing parallelization and synchronization due to work stealing. HPX is in strict adherence to the C++ 11 [24] and C++ 17 standard definitions [25].

## 1. NONLOCAL DIFFUSION EQUATION

In this section we give brief overview of nonlocal diffusion equation for temperature field  $u$  over an square domain  $D = [0, 1]^2$  subjected to zero temperature condition on its boundary and subjected to given heat source distribution within the domain. It is a first order transient equation in time. For simplification, we only consider explicit time integration scheme, namely forward Euler scheme. For spatial discretization, we consider uniform mesh over domain  $D$  and consider finite difference approximation (also commonly referred to as particle discretization). The resulting final equations are very simple and serial implementation requires only few lines of code, see 1. With HPX, thread-level parallel implementation and shared-memory parallel implementation is almost as simple as serial implementation, see 2.

Our goal is to fully parallelize the solver, i.e. starting from mesh partition to computation of fields at mesh nodes. We proceed step wise starting from serial implementation, to shared-memory parallel implementation, and to fully parallel implementation and discuss the challenges going from one step to next.

Let material domain is  $D = [0, 1]^2$  and time domain  $I = [0, T]$ . We fix size of horizon (length over one point interacts with another, also called nonlocal length scale)  $\epsilon > 0$ . Let  $D_\epsilon = (-\epsilon, 1 + \epsilon)^2 - D$  be the nonlocal boundary of thickness  $\epsilon$  surrounding  $D$ . See Figure 1. We define  $H_r(x)$  as two-dimensional open ball of radius  $r$  centered at  $x \in \mathbf{R}^2$ .

Let  $u : I \times D \cup D_\epsilon \rightarrow \mathbf{R}$  is a temperature field. Let  $J : \mathbf{R} \rightarrow \mathbf{R}$  be positive function such that  $0 \leq J(r) \leq M$

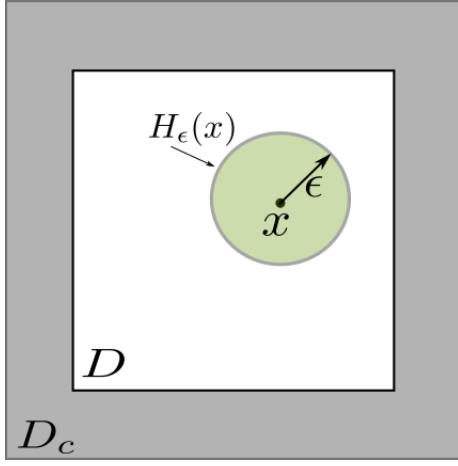


FIGURE 1. Material domain  $D$  and non-local boundary  $D_c$ . Figure shows typical material point  $x \in D$  and ball  $H_\epsilon(x)$ .

for  $r \in [0, 1]$  and  $J(r) = 0$  for  $r \notin [0, 1]$ . We refer to  $J$  as influence function.

$u(t, x)$  satisfies following nonlocal diffusion equation,  $\forall x \in D$  and  $\forall t \in I$ ,

$$(1.1) \quad \frac{\partial}{\partial t} u(t, x) = \frac{1}{\epsilon^4} \int_{H_\epsilon(x)} J(|y - x|/\epsilon) (u(t, y) - u(t, x)) dy.$$

Initial condition is given by

$$(1.2) \quad u(0, x) = u_0(x) \quad \forall x \in D$$

and boundary condition is given by

$$(1.3) \quad u(t, x) = 0 \quad \forall x \in D_c \text{ and } \forall t \in I.$$

## 2. FINITE DIFFERENCE APPROXIMATION

Consider uniform mesh  $D_h = (D \cup D_c) \cap (h\mathbf{Z})^2$  of mesh size  $h > 0$ , see Figure 2.  $\mathbf{Z}$  is the set of positive and negative integers. Let  $\Delta t$  is size of time step and  $[0, T] \cap (\Delta t\mathbf{Z})$  be the discretization of time domain. We assume  $\epsilon = mh$  where  $m \geq 1$  is some integer.

We consider index set  $K \subset \mathbf{Z}^2$  such that for  $i \in K$ ,  $x_i = hi \in D$ . Similarly, consider index set  $K_c \subset \mathbf{Z}^2$  such that for  $i \in K_c$ ,  $x_i = hi \in D_c$ .

Let  $\hat{u}_i^k$  be the solution of forward Euler discretization in time. It satisfies,  $\forall 1 \leq k \leq T/\Delta t, \forall i \in K$ ,

$$(2.1) \quad \frac{\hat{u}_i^{k+1} - \hat{u}_i^k}{\Delta t} = \frac{1}{\epsilon^4} \sum_{\substack{j \in K \cup K_c, \\ |x_j - x_i| \leq \epsilon}} J(|x_j - x_i|/\epsilon) (\hat{u}_j^k - \hat{u}_i^k) V_j$$

and

$$(2.2) \quad \hat{u}_i^k = 0 \quad \forall 0 \leq k \leq T/\Delta t, \forall i \in K_c.$$

At  $k = 0$ , we have  $\hat{u}_i^0 = u_0(x_i)$  for all  $i \in K$ .  $V_j$  is the volume occupied by node  $j$  and in our case we have  $V_j = h^2$ .

---

### Algorithm 1 Serial implementation

---

```

1: % Create neighbor list
2: for each integer  $i \in K$  do
3:   if  $|x_i - x_j| \leq \epsilon$  then
4:     Add  $j$  to neighborList[ $i$ ]
5:   end if
6: end for
7:
8: %  $U$  is the vector of temperatures of
9: % all mesh nodes.
10:
11: % integrate in time
12: for each integer  $0 \leq k \leq T/\Delta t$  do
13:   % time step  $k$ 
14:   for each integer  $i \in K$  do
15:     % Loop over neighbors of  $i$ 
16:     val_i = 0
17:     for each integer  $j \in \text{neighborList}[i]$  do
18:       val_i = val_i +
19:          $\frac{1}{\epsilon^4} J(|x_j - x_i|/\epsilon) (U[j] - U[i]) V_j$ 
20:     end for
21:     % Update temperature
22:      $U[i] = U[i] + \Delta t \times \text{val}_i$ 
23:   end for
24:   % Output temperature at time  $t^k = k\Delta t$ 
25:   Output( $U$ )
26: end for

```

---

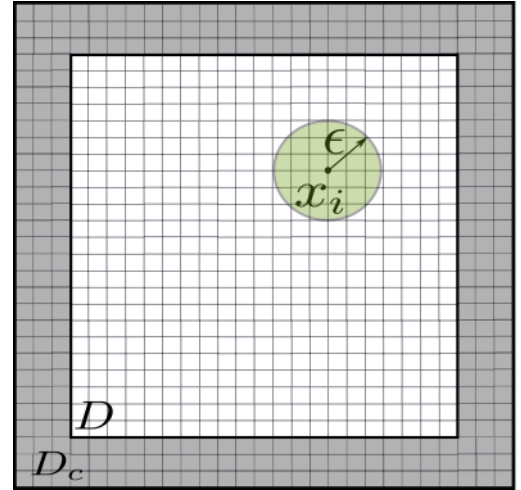


FIGURE 2. Uniform mesh of size  $h$ . Shaded area corresponds to nonlocal boundary  $D_c$ . We specify zero temperature on all the mesh nodes in the shaded area. For mesh node  $x_i$  in  $D$ , we consider interaction of  $x_i$  with all the mesh nodes inside the green shaded ball.

**2.1. Serial and semi-parallel implementation of forward euler scheme.** Algorithm for serial implementation is given in Algorithm 1. For parallel implementation, we will use parallel for loop (HPX utility).

See Algorithm 2. In Algorithm 2, we observe that temperature data of all mesh nodes is stored in single data variable, “ $U$ ”, and similarly, neighbor list of all the mesh nodes are stored in single data variable, “neighborList”. We refer to this as semi-parallel as we have only parallelized for-loop and not the data. In fully parallel implementation, data will also be divided in number of computational nodes and each computational node will own data corresponding to it. This requires mesh partition. We discuss this in next section.

---

**Algorithm 2** Semi-parallel implementation
 

---

```

1: % Create neighbor list using
2: % hpx parallel for loop
3: hpx::parallel::for_loop each integer  $i \in K$  do
4:   if  $|x_i - x_j| \leq \epsilon$  then
5:     Add  $j$  to neighborList[ $i$ ]
6:   end if
7: end parallel for
8:
9: % is the vector of temperatures of
10: % all mesh nodes.
11:
12: % integrate in time
13: for each integer  $1 \leq k \leq T/\Delta t$  do
14:   % time step  $k$ 
15:   % process mesh nodes in parallel
16:   hpx::parallel::for_loop each integer  $i \in K$ 
17:     % Loop over neighbors of  $i$ 
18:     val_i = 0
19:     for each integer  $j \in \text{neighborList}[i]$  do
20:       val_i = val_i +
21:          $\frac{1}{\epsilon^4} J(|x_j - x_i|/\epsilon)(U[j] - U[i])V_j$ 
22:     end for
23:     % Update temperature
24:      $U[i] = U[i] + \Delta t \times \text{val}_i$ 
25:   end parallel for
26:   % Output temperature at time  $t^k = k\Delta t$ 
27:   Output( $U$ )
28: end for

```

---

### 3. PARALLELIZATION AND MESH PARTITION

Given  $N$  number of computational nodes, mesh will be partitioned in  $N$  parts such that each computational node will store the information corresponding to the mesh partition it owns. For example, consider 4 computers connected by network. We would like to run the problem on all 4 computers in parallel. We will partition the mesh in four parts, see Figure 3.

In Figure 3, the mesh is colored to show the partition. Consider part of mesh colored as green. Let us suppose the id of computer who owns the green, blue, yellow, and red are 1, 2, 3, and 4 respectively. We now focus

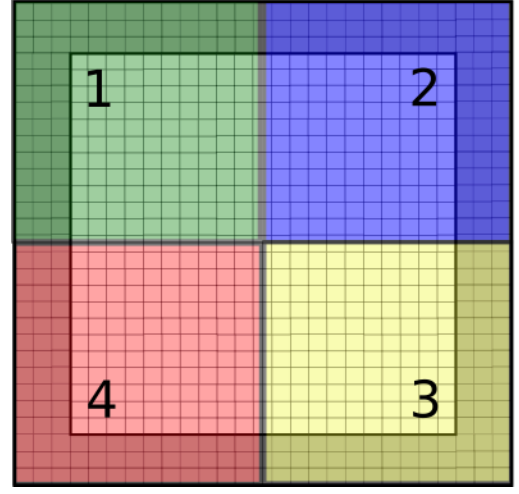


FIGURE 3. Typical mesh partition. Mesh nodes under each color are owned by the respective computer.

on one computational node, say green, and present two possible situations.

**Case 1:** Consider mesh node  $x_i$  which belongs to computer 1’s partition. Let us suppose that  $x_i$  is within the dashed line, see Figure 4. For such  $x_i$ , all mesh nodes  $x_j \in H_\epsilon(x_i)$  will be within the green partition. Since computer 1 owns the mesh data of green partition, calculation of right hand side of Equation 2.1, i.e.  $\frac{1}{\epsilon^4} J(|x_j - x_i|/\epsilon)(\hat{u}_j^k - \hat{u}_i^k)V_j$ , can be carried out without needing to interact with other computers.

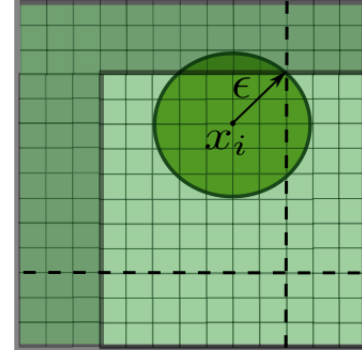


FIGURE 4. Mesh nodes of green partition which are within dashed line do not interact with partition owned by other computers.

**Case 2:** Now consider another mesh node  $x_i$  in green partition. This mesh node is close to the boundary of green partition, and we see that there are mesh nodes  $x_j$  which are in ball  $H_\epsilon(x_i)$ , but which are not part of green partition. For example, we consider  $x_j \in H_\epsilon(x_i)$  in Figure 5. Mesh node  $x_j$  is in Blue partition. Blue partition is owned by computer 2. Therefore, to compute the right hand side term in Equation 2.1, computer 1 has to request the information associated to mesh node  $x_j$

from computer 2. We see that for all the mesh nodes in Green partition which are outside dash line, computer 1 will have to communicate with other computers for the information.

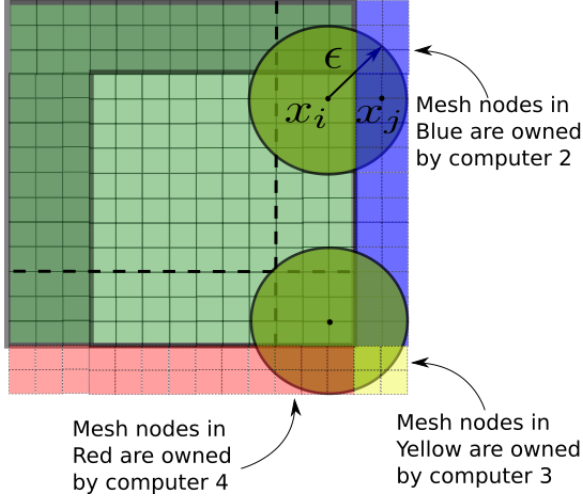


FIGURE 5. In first example, mesh node  $x_i$  of Green partition interacts with few mesh nodes of Blue partition owned by computer 2. In second example, we see that  $x_i$  of Green mesh interacts with mesh nodes owned by computer 2, 3, and 4.

**3.1. Algorithm for fully parallel code.** Consider [Algorithm 3](#) which outlines the steps needed to run the problem in fully parallel framework. Following are the list of steps which will require effort in implementing [Algorithm 3](#).

**1. Partitioning of mesh:** Libraries are available which can partition the mesh into given number of computational nodes.

**2. List of interacting mesh nodes owned by other computational node:** If global mesh data is available to each computational node then we can create a list of interacting neighboring mesh nodes which are owned by other computational node. This list will have global id, which is unique, of mesh nodes.

**3. Sharing of information:** For given computational node, we need to implement the method which shares information to other computational node and which receives the information from other computational node. For this we need a list of information to be shared and communication method available in HPX framework.

---

### Algorithm 3 Fully parallel implementation

---

```

1: % mesh_file is the file containing mesh data
2: % N is the number of computational nodes
3:
4: % get the id of this computational node
5: my_id = get_id()
6:
7: % read mesh data and create mesh partition
8: myMeshNodes = create_mesh_partition(mesh_file)
9:
10: % create neighbor list.
11: neighborList = create_neighbor_list(mesh_file)
12:
13: % next task is to create a list of mesh
14: % nodes, associated to other computational
15: % nodes, which interact with mesh nodes
16: % owned by this computational node
17: for each integer  $0 \leq i \leq N$  do
18:   if  $i == \text{my\_id}$  then
19:     % skip this i
20:   else
21:     % create a list of interacting nodes
22:     % owned by comp. node i
23:     neighborsOutside[i]
24:     = create_list_interacting_nodes(my_id, i)
25:   end if
26: end for
27:
28: % U is the vector of temperatures of
29: % mesh nodes owned by this comp. node
30:
31: % integrate in time
32: for each integer  $0 \leq k \leq T/\Delta t$  do
33:   % time step k
34:
35:   % get data from other computer
36:   % before integrating in time
37:   for each integer  $0 \leq i \leq N$  do
38:     if  $i == \text{my\_id}$  then
39:       % skip this i
40:     else
41:       % request data from comp. node i
42:       getData[i] = request_data(my_id, i)
43:
44:       % share data to comp. node i
45:       % as comp. node i may also have
46:       % mesh nodes which have neighbors
47:       % in this comp. node
48:       share_data(my_id, i)
49:     end if
50:   end for

```

---



---

```

51: % we now have all the relevant data so
52: % we can solve for temperature
53: % at next time step
54: hpx::parallel::for loop  $i \in \text{myMeshNodes}$  do
55:   % Loop over neighbors of  $i$ 
56:    $\text{val}_i = 0$ 
57:   for each integer  $j \in \text{neighborList}[i]$  do
58:     % check if  $j$  is in  $\text{myMeshNode}$ 
59:     if  $j \in \text{myMeshNode}$  then
60:       % this mesh node is within dashed line
61:
62:       % compute contribution
63:        $\text{val}_i = \text{val}_i$ 
64:          $+ \frac{1}{\epsilon^4} J(|x_j - x_i|/\epsilon)$ 
65:          $(U[j] - U[i])V_j$ 
66:     else
67:       % this mesh node is outside dashed line
68:       % search and find which comp. node
69:       % owns mesh node  $j$ 
70:        $\text{get\_comp\_id} =$ 
71:          $\text{search\_for\_mesh\_node}(\text{my\_id}, j)$ 
72:
73:       % Look for  $j$  in  $\text{getData}[\text{get\_comp\_id}]$ 
74:       % and get the temperature of mesh node  $j$ 
75:        $t_j = \text{find\_in}(j, \text{getData}[\text{get\_comp\_id}])$ 
76:
77:       % compute contribution
78:        $\text{val}_i = \text{val}_i$ 
79:          $+ \frac{1}{\epsilon^4} J(|x_j - x_i|/\epsilon)$ 
80:          $(t_j - U[i])V_j$ 
81:     end if
82:   end for
83:   % Update temperature
84:    $U[i] = U[i] + \Delta t \times \text{val}_i$ 
85: end parallel for
86: % Output temperature at time  $t^k = k\Delta t$ 
87:  $\text{Output}(U)$ 
88: end for

```

---

## REFERENCES

- [1] Abigail Agwai, Ibrahim Guven, and Erdogan Madenci. Predicting crack propagation with peridynamics: a comparative study. *International journal of fracture*, 171(1):65–78, 2011. ([document](#))
- [2] Nicola J Armstrong, Kevin J Painter, and Jonathan A Sherratt. A continuum approach to modelling cell–cell adhesion. *Journal of theoretical biology*, 243(1):98–113, 2006. ([document](#))
- [3] Masoud Behzadinasab, Tracy J Vogler, Amanda M Peterson, Rezwanur Rahman, and John T Foster. Peridynamics modeling of a shock wave perturbation decay experiment in granular materials with intra-granular fracture. *Journal of Dynamic Behavior of Materials*, 4(4):529–542, 2018. ([document](#))
- [4] Florin Bobaru and Wenke Hu. The meaning, selection, and use of the peridynamic horizon and its relation to crack branching in brittle materials. *International journal of fracture*, 176(2):215–222, 2012. ([document](#))
- [5] Nathaniel Burch and Richard Lehoucq. Classical, nonlocal, and fractional diffusion equations on bounded domains. *International Journal for Multiscale Computational Engineering*, 9(6), 2011. ([document](#))
- [6] Prathamesh Desai. Tribosurface interactions involving particulate media with dem-calibrated properties: Experiments and modeling. 2017. ([document](#))
- [7] Prathamesh S Desai, Akash Mehta, Patrick SM Dougherty, and Fred C Higgs. A rheometry based calibration of a first-order dem model to generate virtual avatars of metal additive manufacturing (am) powders. *Powder technology*, 342:441–456, 2019. ([document](#))
- [8] P Diehl, R Lipton, and MA Schweitzer. Numerical verification of a bond-based softening peridynamic model for small displacements: Deducing material parameters from classical linear theory. 2016. ([document](#))
- [9] Maksym Dosta, Steven Dale, Sergiy Antonyuk, Carl Wassgren, Stefan Heinrich, and James D Litster. Numerical and experimental analysis of influence of granule microstructure on its compression breakage. *Powder technology*, 299:87–97, 2016. ([document](#))
- [10] Qiang Du, Max Gunzburger, Richard B Lehoucq, and Kun Zhou. Analysis and approximation of nonlocal diffusion problems with volume constraints. *SIAM review*, 54(4):667–696, 2012. ([document](#))
- [11] Christian Engwer, Christian Stinner, and Christina Surulescu. On a structured multiscale model for acid-mediated tumor invasion: the effects of adhesion and proliferation. *Mathematical Models and Methods in Applied Sciences*, 27(07):1355–1390, 2017. ([document](#))
- [12] M Ghajari, L Iannucci, and P Curtis. A peridynamic material model for the analysis of dynamic crack propagation in orthotropic media. *Computer Methods in Applied Mechanics and Engineering*, 276:431–452, 2014. ([document](#))
- [13] Anton Gladky and Meinhard Kuna. Dem simulation of polyhedral particle cracking using a combined mohr–coulomb–weibull failure criterion. *Granular Matter*, 19(3):41, 2017. ([document](#))
- [14] Youn Doh Ha and Florin Bobaru. Studies of dynamic crack propagation and crack branching with peridynamics. *International Journal of Fracture*, 162(1-2):229–244, 2010. ([document](#))
- [15] Thomas Heller, Patrick Diehl, Zachary Byerly, John Biddiscombe, and Hartmut Kaiser. HPX – An open source C++ Standard Library for Parallelism and Concurrency. In *Proceedings of OpenSuCo 2017, Denver, Colorado USA, November 2017 (OpenSuCo17)*, page 5, 2017. ([document](#))
- [16] Prashant K. Jha and Robert Lipton. Numerical convergence of nonlinear nonlocal continuum models to local elastodynamics. *International Journal for Numerical Methods in Engineering*, 114(13):1389–1410, 2018. ([document](#))
- [17] Prashant K Jha and Robert Lipton. Numerical convergence of finite difference approximations for state based peridynamic fracture models. *Computer Methods in Applied Mechanics and Engineering*, 351:184–225, July 2019. ([document](#))
- [18] Hartmut Kaiser, Thomas Heller, Bryce Adelstein-Lelbach, Adrian Serio, and Dietmar Fey. Hpx: A task based programming model in a global address space. In *Proceedings of the 8th International Conference on Partitioned Global Address Space Programming Models*, page 6. ACM, 2014. ([document](#))
- [19] Robert Lipton, Stewart Silling, and Richard Lehoucq. Complex fracture nucleation and evolution with nonlocal elastodynamics. *arXiv preprint arXiv:1602.00247*, 2016. ([document](#))

- [20] Sebastian Lobo-Guerrero and Luis E Vallejo. Discrete element method evaluation of granular crushing under direct shear test conditions. *Journal of Geotechnical and Geoenvironmental Engineering*, 131(10):1295–1300, 2005. ([document](#))
- [21] SA Silling, O Weckner, E Askari, and Florin Bobaru. Crack nucleation in a peridynamic solid. *International Journal of Fracture*, 162(1-2):219–227, 2010. ([document](#))
- [22] Christian Stinner, Christina Surulescu, and Michael Winkler. Global weak solutions in a pde-ode system modeling multiscale cancer cell invasion. *SIAM Journal on Mathematical Analysis*, 46(3):1969–2007, 2014. ([document](#))
- [23] Alexandre Tabbal, Matthew Anderson, Maciej Brodowicz, Hartmut Kaiser, and Thomas Sterling. Preliminary design examination of the parallex system from a software and hardware perspective. *ACM SIGMETRICS Performance Evaluation Review*, 38(4):81–87, 2011. ([document](#))
- [24] The C++ Standards Committee. ISO International Standard ISO/IEC 14882:2011, Programming Language C++. Technical report, Geneva, Switzerland: International Organization for Standardization (ISO)., 2011. <http://www.open-std.org/jtc1/sc22/wg21>. ([document](#))
- [25] The C++ Standards Committee. ISO International Standard ISO/IEC 14882:2017, Programming Language C++. Technical report, Geneva, Switzerland: International Organization for Standardization (ISO)., 2017. <http://www.open-std.org/jtc1/sc22/wg21>. ([document](#))
- [26] F Zhu and J Zhao. A peridynamic investigation on crushing of sand particles. *Géotechnique*, 69(6):526–540, 2018. ([document](#))