NONLOCAL DIFFUSION PROBLEM: MESH PARTITIONING FOR HPX PARALLEL COMPUTING

In this document we discuss the problem where we want to solve nonlocal diffusion equation in 2-d using parallel API called HPX. Our goal is to fully parallelize the code, i.e. starting from mesh partition to computation of fields at mesh nodes. As our main goal is to implement the parallel algorithm, we minimize the complexity by considering simple nonlocal diffusion model, Dirichlet boundary condition, and finite difference approximation over uniform mesh. The extension to more complicated nonlocal diffusion equation and also to peridynamics (nonlocal mechanics) is not difficult. Also, once the parallel code is implemented for finite difference approximation, one can look for extension of this to finite element approximation.

1. Nonlocal diffusion equation

We consider material domain $D = (0,1)^2$ and time domain $I = [0,T]$. We fix size of horizon $\epsilon > 0$. Let $D_c = (-\epsilon, 1+\epsilon)^2 - D$ be the nonlocal boundary of thickness ϵ 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 $u : I \times D \cup D_c \to \mathbf{R}$ is a temperature field. Let $J : \mathbf{R} \to \mathbf{R}$ be positive function such that $0 \leq J(r) \leq M$ for $r \in [0,1]$ and $J(r) = 0$ for $r \notin [0,1]$. We refer to J as influence function.

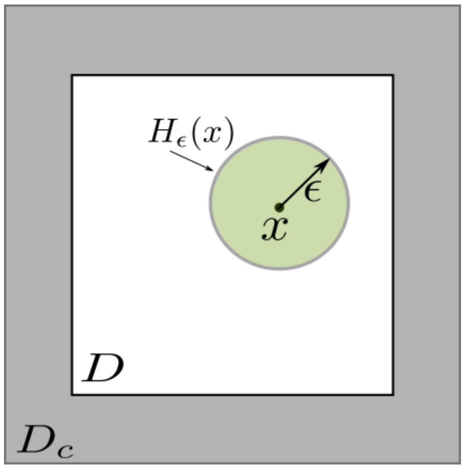

FIGURE 1. Material domain D and nonlocal boundary D_c. Figure shows typical material point $x \in D$ and ball $H_\epsilon(x)$.

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

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

Initial condition is given by

$$u(0,x) = u_0(x) \qquad \forall x \in D \tag{1.2}$$

and boundary condition is given by

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

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 $j \in$ 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:         $U[i] = U[i] + \Delta t \times val\_i$
23:     end for
24:     % Output temperature at time $t^k = k\Delta t$
25:     Output($U$)
26: end for
```

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. Let Δt 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,$\forall 1 \leq k \leq T/\Delta t, \forall i \in K$,

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

and

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

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 $V_j = h^2$.

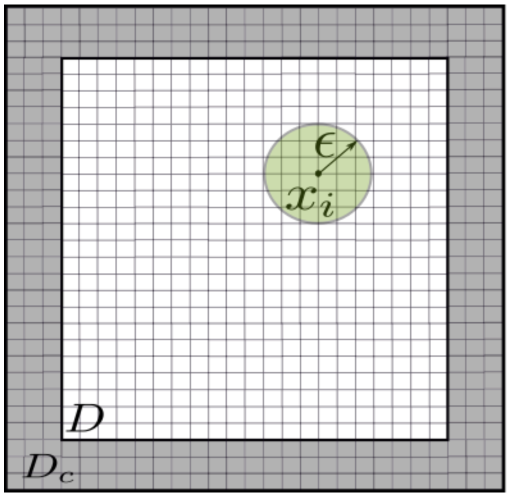

FIGURE 2. Uniform mesh of size h. Shaded area corresponds to nonlocal boundary D_c. We specify zero temperature on all the mesh nodes in the shaded area. For mesh node x_i in D, we consider interaction of x_i with all the mesh nodes inside the green shaded ball.

Algorithm 2 Semi-parallel implementation

```
% Create neighbor list using
% hpx parallel for loop
hpx::parallel::for_loop each integer $i \in K$ do
    if $|x_i - x_j| \leq \epsilon$ then
        Add $j$ to neighborList[$i$]
    end if
end parallel for

% is the vector of temperatures of
% all mesh nodes.

% integrate in time
for each integer $1 \leq k \leq T/\Delta t$ do
    % time step $k$
    % process mesh nodes in parallel
    hpx::parallel::for_loop each integer $i \in K$
        % Loop over neighbors of $i$
        $val\_i = 0$
        for each integer $j \in$ neighborList[$i$] do
            $val\_i = val\_i +$
                $+ \frac{1}{\epsilon^4} J(|x_j - x_i|/\epsilon)(U[j] - U[i]) V_j$
        end for
        % Update temperature
        $U[i] = U[i] + \Delta t \times val\_i$
    end parallel for
    % Output temperature at time $t^k = k\Delta t$
    Output($U$)
end for
```

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.

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.

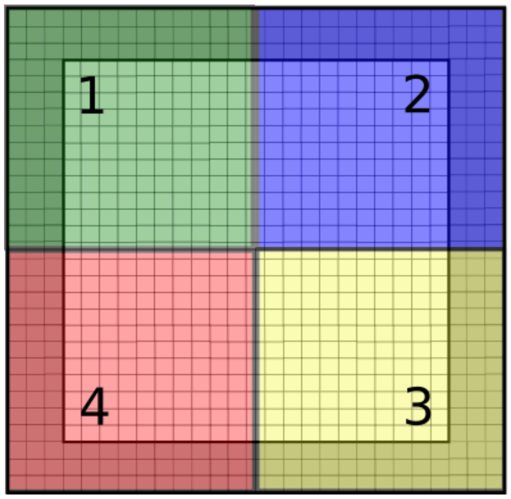

FIGURE 3. Typical mesh partition. Mesh nodes under each color are owned by the respective computer.

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 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.

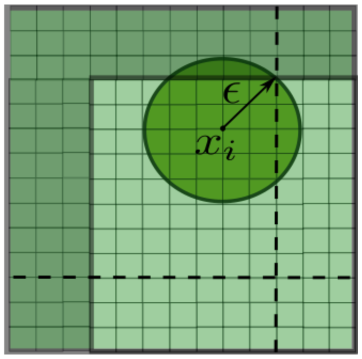

FIGURE 4. Mesh nodes of green partition which are within dashed line do not interact with partition owned by other computers.

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.

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

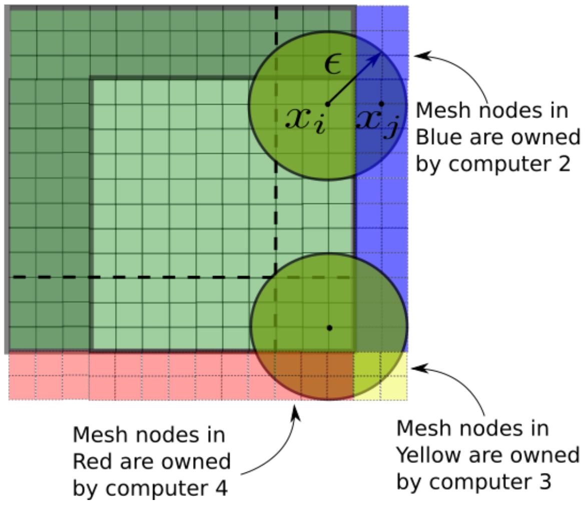
ϵ
x_i
x_j
Mesh nodes in Blue are owned by computer 2
Mesh nodes in Red are owned by computer 4
Mesh nodes in Yellow are owned by computer 3

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.

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

```
% mesh_file is the file containing mesh data
% N is the number of computational nodes

% get the id of this computational node
my_id = get_id()

% read mesh data and create mesh partition
myMeshNodes = create_mesh_partition(mesh_file)

% create neighbor list.
neighborList = create_neighbor_list(mesh_file)

% next task is to create a list of mesh
% nodes, associated to other computational
% nodes, which interact with mesh nodes
% owned by this computational node
for each integer $0 \le i \le N$ do
  if $i ==$ my_$id$ then
    % skip this $i$
  else
    % create a list of interacting nodes
    % owned by comp. node $i$
    neighborsOutside[i]
      = create_list_interacting_nodes(my_id, i)
  end if
end for

% $U$ is the vector of temperatures of
% mesh nodes owned by this comp. node

% integrate in time
for each integer $0 \le k \le T/\Delta t$ do
  % time step $k$

  % get data from other computer
  % before integrating in time
  for each integer $0 \le i \le N$ do
    if $i ==$ my_$id$ then
      % skip this $i$
    else
      % request data from comp. node $i$
      getData[i] = request_data(my_id, i)

      % share data to comp. node $i$
      % as comp. node $i$ may also have
      % mesh nodes which have neighbors
      % in this comp. node
      share_data(my_id, i)
    end if
  end for
  % we now have all the relevant data so
  % we can solve for temperature
  % at next time step
  hpx::parallel::for_loop $i \in$ myMeshNodes do
    % Loop over neighbors of $i$
    $val\_i = 0$
    for each integer $j \in$ neighborList[$i$] do
      % check if $j$ is in myMeshNode
      if $j \in$ myMeshNode then
        % this mesh node is within dashed line

        % compute contribution
        $val\_i = val\_i$
          $+ \frac{1}{\epsilon^4} J(|x_j - x_i|/\epsilon)$
          $(U[j] - U[i])V_j$
      else
        % this mesh node is outside dashed line
        % search and find which comp. node
        % owns mesh node $j$
        get_comp_id =
          search_for_mesh_node(my_id, j)

        % Look for $j$ in getData[get_comp_id]
        % and get the temperature of mesh node $j$
        t_j = find_in(j, getData[get_comp_id])

        % compute contribution
        $val\_i = val\_i$
          $+ \frac{1}{\epsilon^4} J(|x_j - x_i|/\epsilon)$
          $(t\_j - U[i])V_j$
      end if
    end for
    % Update temperature
    $U[i] = U[i] + \Delta t \times val\_i$
  end parallel for
  % Output temperature at time $t^k = k\Delta t$
  Output($U$)
end for
```