Accelerating Peridynamics Simulations on the GPU

Steven F. Henke and Sachin Shanbhag

Scientific Computing, Florida State University, Tallahassee, FL 32306, USA

1. Introduction

Peridynamics [2] is a recently developed non-local theory of continuum mechanics that is useful in simulating multi-scale phenomena. Its formulation is based upon an integral equation of motion so that discontinuities may spontaneously form and propagate without special treatment, thus it is naturally well-suited to modeling fracture, dislocations, and phase boundaries.

The governing equations of peridynamics permit a "mesh-free" solution method [3] that has the same computational structure as a molecular dynamics simulation. Molecular dynamics [1] and other similar particle-based simulations have been seen large speedups and linear scaling when implemented on graphics processing units (GPUs), which excel at such data-parallel computational tasks.

In this work, we review a particle-based solution method for the governing equations of peridynamics, and detail our efforts to extend this method to the GPU using NVIDIA's CUDA parallel computing architecture.

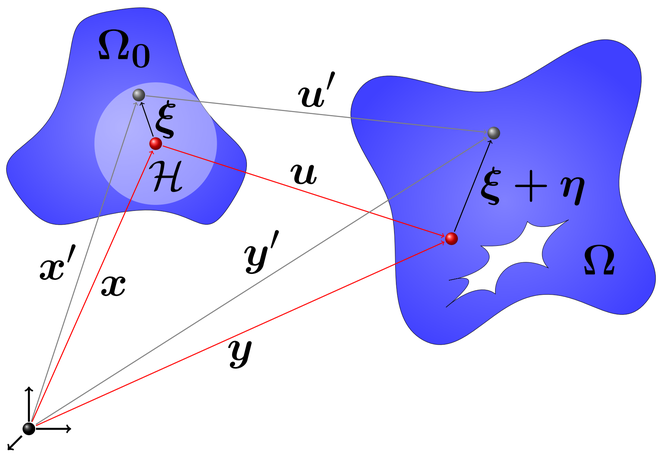

Figure 1: *Kinematic quantities that describe a peridynamic continuum body in its reference (Ω_0) and current (Ω) configurations.*

2. Peridynamics

Peridynamic theory is a reformulation of continuum mechanics that employs a non-local force model to account for long-range material interactions. It is governed by an integro-differential equation of motion that avoids spatial derivatives.

$$\rho \frac{\partial^2 \boldsymbol{u}}{\partial t^2}(\boldsymbol{x},t) = \int_{\mathcal{H}_x} \boldsymbol{f}\left(\boldsymbol{u}'-\boldsymbol{u},\boldsymbol{x}'-\boldsymbol{x}\right) dV_{\boldsymbol{x}'} + \boldsymbol{b}(\boldsymbol{x},t) \tag{1}$$

From here on, the relative position between particles in the reference configuration is denoted by $\boldsymbol{\xi}$ and in the current configuration by $\boldsymbol{\xi}+\boldsymbol{\eta}$, where $\boldsymbol{\eta}$ is the relative displacement. The pairwise internal force function $\boldsymbol{f}(\boldsymbol{\eta},\boldsymbol{\xi})$ contains all of a material body's constitutive information and the force term $\boldsymbol{b}(\boldsymbol{x},t)$ accounts for all external forces acting upon the body.

Various numerical integration techniques have been useful in approximating the peridynamic equation of motion, including Gaussian quadrature, finite elements, and spectral methods. Our solution scheme uses the so-called mesh-free "EMU" method [3] which discretizes spatial quantities using the composite midpoint rule,

$$\rho \frac{\partial^2 \boldsymbol{u}_i^n}{\partial t^2} = \sum_p \boldsymbol{f}\left(\boldsymbol{u}_p^n - \boldsymbol{u}_i^n, \boldsymbol{x}_p - \boldsymbol{x}_i\right) V_p + \boldsymbol{b}_i^n, \tag{2}$$

and temporal quantities using a central difference (Verlet) method,

$$\frac{\partial^2 \boldsymbol{u}_i^n}{\partial t^2} \approx \frac{\boldsymbol{u}_i^{n+1} - 2\boldsymbol{u}_i^n + \boldsymbol{u}_i^{n-1}}{(\Delta t)^2}, \tag{3}$$

In the above equations, superscripts indicate the time step number during which a quantity is evaluated, and subscripts represent the node number.

In this work, we focus solely on the *bond-based* peridynamic theory, where it is assumed that the collection of bonds associated with a particle do not interact with each other. Furthermore, we restrict our study to micro-elastic materials, in which the pairwise force function is conservative, so $\boldsymbol{f}(\boldsymbol{\eta},\boldsymbol{\xi})$ can be written as the gradient of a scalar micro-potential.

$$\boldsymbol{f}(\boldsymbol{\eta},\boldsymbol{\xi}) = \frac{\partial w}{\partial \boldsymbol{\eta}}(\boldsymbol{\eta},\boldsymbol{\xi}) \tag{4}$$

We continue by defining a particular choice for the micropotential that is appropriate for a brittle elastic material.

3. Prototype Micro-elastic Brittle (PMB) Material

Two particles in a PMB material are initially bonded if they are positioned within some neighborhood $\|\boldsymbol{\xi}\| \leq \delta$. Bonded particles exert a force on each other that is analogous to that of an elastic spring.

$$w = \frac{1}{2}\frac{c}{\|\boldsymbol{\xi}\|}\mu\left(\|\boldsymbol{\eta}+\boldsymbol{\xi}\| - \|\boldsymbol{\xi}\|\right)^2 \tag{5}$$

$$\boldsymbol{f} = cs\mu\frac{\boldsymbol{\eta}+\boldsymbol{\xi}}{\|\boldsymbol{\eta}+\boldsymbol{\xi}\|} \tag{6}$$

The stretch of a bond is defined as the relative difference of the particle's relative separation distance in the reference and current configuration,

$$s = \frac{\|\boldsymbol{\eta}+\boldsymbol{\xi}\| - \|\boldsymbol{\xi}\|}{\|\boldsymbol{\xi}\|}. \tag{7}$$

In a brittle damage model, bonds stretched beyond a certain critical extension are broken irreversibly so that the involved particles no longer interact.

$$\mu(t,\xi) = \begin{cases} 1 & s(t',\xi) < s_0(t') \;\forall\, t' \in (0,t) \\ 0 & \text{otherwise} \end{cases}. \tag{8}$$

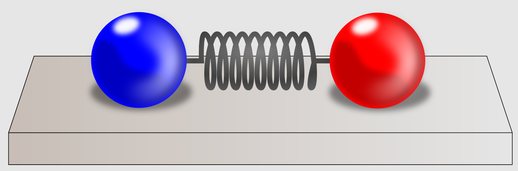

Figure 2: *In a PMB material, bonded continuum particles attract and repel in a spring-mass system until their bond is broken.*

A short-range force is introduced in the current configuration to prevent the overlap of moving material. Thus, for each particle pair such that $\|\boldsymbol{\eta}+\boldsymbol{\xi}\| \leq d^s$, we include an additional repulsive force.

$$w^s = \frac{1}{2}\frac{c^s}{\delta}\left(\|\boldsymbol{\eta}+\boldsymbol{\xi}\| - d^s\right)^2 \tag{9}$$

$$\boldsymbol{f}^s = \frac{c^s}{\delta}\left(\|\boldsymbol{\eta}+\boldsymbol{\xi}\| - d^s\right)\frac{\boldsymbol{\eta}+\boldsymbol{\xi}}{\|\boldsymbol{\eta}+\boldsymbol{\xi}\|} \tag{10}$$

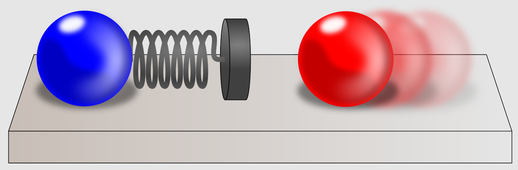

Figure 3: *Repulsive short range forces are necessary to prevent the overlap of nearby continuum particles in the current configuration.*

4. Graphics Processing Units

Graphics Processing Units (GPUs) are powerful parallel number crunching engines readily available in many computer workstations. Relative to their CPU counterparts, such GPUs dedicate more transistors to arithmetic circuitry while minimizing control and caching units, thus are specialized for processing compute-intensive, highly parallel algorithms.

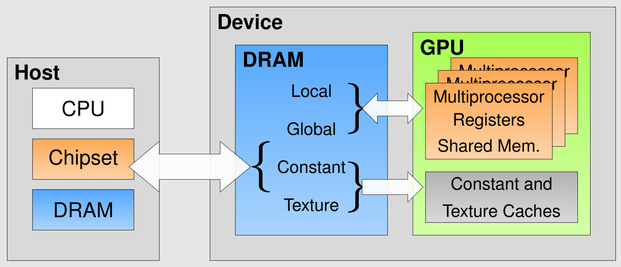

Figure 4: *Depiction of various memory spaces on a CUDA Device. Adapted from the NVIDIA CUDA C Best Practices Guide.*

Adapting computer programs designed for CPUs to a GPU architecture can be non-trivial and often requires traditional algorithms and data structures to be rethought to achieve good performance. Attaining maximum speedups for an application revolves around three basic strategies: maximizing parallel execution, optimizing memory usage to achieve maximum memory bandwidth, and optimizing instruction usage to achieve maximum instruction throughput.

5. Algorithm

For each time-step:

1. Update positions and half-step velocities of all particles
2. Sort current positions based on location along space-filling curve
3. Update computation of bond forces
4. Compute short range forces; neighbor query can be terminated early due to structure of space-filling curve
5. Finish velocity update of all particles
6. When needed, reduce energy and scalar damage of each particle.

6. Space-Filling Curves

A space-filling curve is as a continuous function that maps points in n-dimensional space to the unit interval $[0,1]$. Their self-similar structure leads to many useful applications. Space-filling curves are an essential ingredient to enabling parallelism and scalability of our algorithm.

(a) Raster (b) Lebesgue (c) Hilbert

Figure 5: *Examples of space-filling curves in two dimensions (order=4)*

In our scheme, we work with the Lebesgue (also known as a Morton order or Z-order) curve for its good spatial locality and simple implementation. We encode a Morton order key by interleaving the bits of a position vector.

$$Z = \sum_{d=0}^{n-1}\sum_{k=0}^{w-1} b_k^{(d)} 2^{nk+d} \tag{11}$$

This formula groups bits of equal significance into n-tuples within the key without altering their order. This observation allows us to compare the relative position of any two points on the curve without the expense of explicit bit interleaving.

$i \leftarrow 1$
for $j \leftarrow 2$ **to** n **do**
  if $\lfloor \log_2(\mathtt{lhs}_i \oplus \mathtt{rhs}_i) \rfloor < \lfloor \log_2(\mathtt{lhs}_j \oplus \mathtt{rhs}_j) \rfloor$
    $i \leftarrow j$
end
return $(\mathtt{lhs}_i < \mathtt{rhs}_i)$

Shortcuts are available for evaluating the $\log_2$ comparison of both integer and floating point valued coordinates. This enables a sparse data structure for storing the history of inter-particle bonds, and an efficient algorithm for ball neighbor queries.

7. Summary and Future Work

- The theory described here has been implemented in a C++ code using NVIDIA's CUDA parallel computing architecture and the Thrust library.
- Spatial orderings are used to increase memory throughput when querying the neighbor lists that record bond histories.
- The neighbor query required in computing the short-range forces has been reduced in complexity using space-filling curves.
- We demonstrated this method's use in impact simulations (see below).
- We are working to extend this code to study composite structures.

References

[1] Joshua A. Anderson, Chris D. Lorenz, and A. Travesset. General purpose molecular dynamics simulations fully implemented on graphics processing units. *Journal of Computational Physics*, 227(10):5342 – 5359, 2008.

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

[3] S.A. Silling and E. Askari. A meshfree method based on the peridynamic model of solid mechanics. *Computers and Structures*, 83(17-18):1526 – 1535, 2005. Advances in Meshfree Methods.

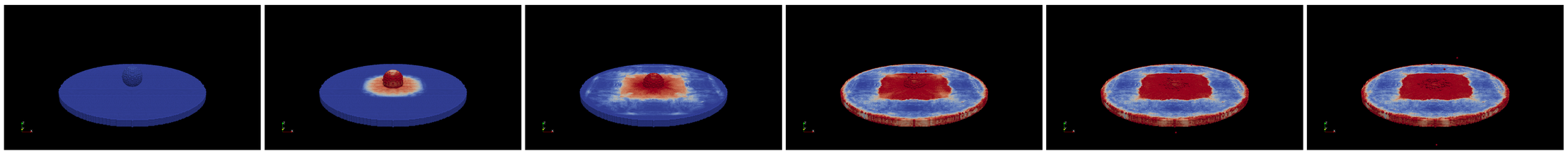

Figure 6: *We implemented the above solution scheme on the GPU and simulated the high speed impact of a spherical projectile into a cylindrical target.*