SAND2014-1530P

Chapter 18

Roadmap for Software Implementation

David Littlewood

Sandia National Laboratories

CONTENTS

18.1 Introduction

Peridynamics has advanced the state of the art for modeling material failure and fragmentation within a computational simulation code. To be effective in modeling real-world systems, the underlying theory must be accompanied by an effective software implementation strategy. The goal of this chapter is to provide a roadmap for implementing peridynamics within an analysis code. This is motivated in part by the gaps that exist between much of the available literature on peridynamics, which is often focused on theoretical considerations, and implementation challenges that can arise in practice.

SAND 2014-XXXX
Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

This chapter focuses exclusively on the meshfree approach of Silling and Askari [17]. To date, the vast majority of peridynamic simulations have utilized this approach. The decision to focus on this particular numerical strategy is not meant to imply that it is superior to alternative approaches. Applying the finite-element technique to peridynamics, for example, may provide superior results in some cases. Nonetheless, the meshfree approach of Silling and Askari has proven to be a reliable and efficient strategy for addressing problems in solid mechanics with pervasive material failure and fracture.

The goal of a peridynamics code that implements the meshfree approach of Silling and Askari is to solve the following semidiscrete equation of motion,

$$\rho(\mathbf{x})\,\ddot{\mathbf{y}}(\mathbf{x},t) = \sum_{\mathcal{B}} \left(\underline{\mathbf{T}}[\mathbf{x}]\,\langle \mathbf{q}-\mathbf{x}\rangle - \underline{\mathbf{T}}[\mathbf{q}]\,\langle \mathbf{x}-\mathbf{q}\rangle\right) dV_{\mathbf{q}} + \mathbf{b}(\mathbf{x}), \tag{18.1}$$

where ρ is the material density, $\ddot{\mathbf{y}}$ is acceleration, t is time, $\mathbf{x}$ and $\mathbf{q}$ are material points in the body $\mathcal{B}$, $\underline{\mathbf{T}}$ is a peridynamic force state, V denotes volume, and $\mathbf{b}$ is a body force density. In general, simulations may involve multiple bodies, multiple materials, and any number of specified initial and boundary conditions. Equation 18.1 holds for both bond-based and state-based constitutive models, as bond-based models can be considered a special case of the more general family of state-based models.

At a high level, a computational peridynamics code contains several key components: a time integrator, a method for evaluating the internal force, routines for applying initial and boundary conditions, a means to evaluate and apply contact forces, and input/output routines for reading and writing information. For transient dynamic simulations, the critical time step must be estimated. Quasi-static simulations require the implementation of a nonlinear solver. One standard approach for solving a nonlinear system of equations is Newton's method. Newton's method requires the construction of a tangent stiffness matrix and a method for solving a linear system of equations.

Peridynamics has been implemented in standalone software, and also as a component of a larger, more comprehensive simulation codes. Peridynamics was originally implemented in the *EMU* code by Stewart Silling [17]. Examples of preexisting software packages in which peridynamics has been implemented include *LAMMPS* and *Sierra/SolidMechanics* [12, 15]. It has also been demonstrated that peridynamics can be implemented in the commercial finite-element code *Abaqus* [11]. A more recent implementation of peridynamics is the open-source code *Peridigm* [13, 2].

The following sections address the pimary components of a computational peridynamics code. Where it is illustrative to do so, the *Peridigm* code is used as an example. The majority of the discussion focuses on serial code. While implementation of a parallel code adds complexity, there are no fundamental changes required to the core peridynamic algorithms. Software implementation of a peridynamic constitutive model is discussed, followed by sections covering the tangent stiffness matrix, contact modeling, meshfree discretizations, the proximity search for identifying peridynamics bonds, and time integration. Finally, example simulations demonstrating explicit transient dynamics and quasi-statics are presented.

18.2 Evaluating the internal force density

The core kernels of a peridynamic code are the routines for computing the internal force density. The internal force density is determined via constitutive laws that compute pairwise force densities based on kinematic data and, optionally, material state variables. Core constitutive model routines include initialization, calculation of the internal force density, and, optionally, calculation of the tangent stiffness matrix. Each of these routines has access to discretization-related data (initial positions and element volumes), neighbor lists, values for the current time and the time step size, and basic kinematic data that are updated throughout the simulation by the time integration routine (*e.g.*, velocities and current positions). In addition, constitutive models may define material-specific data fields that are tracked and updated over the course of a simulation.

Constitutive model routines are intimately tied to the time integrator. The goal of a time integrator is to advance a simulation from a given time step or load step, n, to the next step, $n+1$. For this reason, values corresponding to both step n and step $n+1$ are tracked for many fields. Tracking the current coordinates at steps n and $n+1$, for example, enables a constitutive model to compute rates of deformation. The storing of solutions at both step n and $n+1$ also facilitates the evauation of trial configurations in implicit time integration. In this case, the solution at step n must be available for reuse in the evaluation of multiple trial configurations.

The initialization and internal force density routines for a linear peridynamic solid constitutive model with a Gaussian influence function are presented in Algorithms 1 and 2, respectively. Note that in the internal force density routine, the force state corresponding to each material point is is computed only once. Once the force state for a given point, $\mathbf{x}$, is computed, it is immediately applied to the bonds $\langle \mathbf{q}_i - \mathbf{x} \rangle$ and $\langle \mathbf{x} - \mathbf{q}_i \rangle$ that correspond to each neighbor, $\mathbf{q}_i$, of $\mathbf{x}$. The resulting pairwise force density vectors are summed directly into the global force density vector for both $\mathbf{x}$ and $\mathbf{q}$ in accordance with Equation (18.1).

18.3 The tangent stiffness matrix

The tangent stiffness matrix is required when applying many nonlinear solver strategies, for example Newton's method, for implicit time integration. It is comprised of the derivatives of nodal forces with respect to nodal displacements. Alternative formulations are also possible, for example one in which the nodal velocities are treated as cardinal unknowns. A common approach for construction of the tangent stiffness matrix is to first consider the internal force (*i.e.*, the constitutive model response), after which the matrix is modified to reflect boundary conditions. For general nonlinear problems, the tangent stiffness matrix is a function of the chosen linearization

Algorithm 1 The initialization routine for a linear peridynamic solid

procedure LINEAR PERIDYNAMIC SOLID INITIALIZATION
  ▷ Compute the weighted volume for each element
  for each element i **do**
    $m_i \leftarrow 0$
    for each element j in neighbor list for element i **do**
      $\zeta \leftarrow \|\mathbf{x}_j - \mathbf{x}_i\|$
      $\underline{\omega} \leftarrow \exp\left(-\frac{\zeta^2}{\delta^2}\right)$
      $m_i \leftarrow m_i + \underline{\omega}\,\zeta\,\zeta$
    end for
  end for
end procedure

point and must be re-evaluated whenever the nodal displacements are updated. It is important to note that several so-called linear peridynamic constitutive models impose a nonlinear relationship between force and displacement, and therefore require the application of a nonlinear solver. An example is the linear peridynamic solid, in which pairwise forces are linearly related to bond stretch but nonlinearly related to the nodal displacements.

The tangent stiffness matrix is defined as

$$K_{ij} = \frac{\partial f_i^{int}(\mathbf{u})}{\partial u_j}, \tag{18.2}$$

where f_i^{int} is a component of the internal force vector, $\mathbf{u}$ is the displacement vector, and u_j is a component of $\mathbf{u}$. The tangent stiffness matrix relates an infinitesimal perturbation of a displacement degree of freedom to the resulting change in the global internal force vector. The tangent stiffness matrix for a nonlocal model is inherently more dense than that of a local model. The tangent stiffness matrix associated with a state-based peridynamic model, for example, has nonzero entries for each pair of nodes that interact directly. This interactin can take two forms: interactions between nodes that are bonded to each other, and interactions between nodes that share a common neighbor.

Finite-difference approximation is one approach to constructing the tangent stiffness matrix. The partial derivatives in Equation (18.2) may be approximated, for example, using a central-difference scheme,

$$K_{ij} \approx \frac{f_i^{int}(\mathbf{u}+\varepsilon_j) - f_i^{int}(\mathbf{u}-\varepsilon_j)}{2\varepsilon}. \tag{18.3}$$

Here, ε_j is a vector containing a single nonzero entry, ε, at the position corresponding to the j^{th} degree of freedom in the discretization. For an accurate approximation, the magnitude of ε should be chosen to be small relative to the discretization, but not so small that the limits of machine precision become a significant factor. The default

Algorithm 2 The internal force routine for a linear peridynamic solid

procedure LINEAR PERIDYNAMIC SOLID INTERNAL FORCE
  ▷ Initialize global force density vector to zero
  for each element i **do**
    $\mathbf{f}_i \leftarrow 0$
  end for
  ▷ Compute the dilatation for each element
  for each element i **do**
    $\theta_i \leftarrow 0$
    for each element j in neighbor list for element i **do**
      $\zeta \leftarrow \|\mathbf{x}_j - \mathbf{x}_i\|$
      $\eta \leftarrow \|\mathbf{y}_j - \mathbf{y}_i\|$
      $\underline{\omega} \leftarrow \exp\left(-\frac{\zeta^2}{\delta^2}\right)$
      $\underline{e} \leftarrow \eta - \zeta$
      $\theta_i \leftarrow \theta_i + \frac{3}{m_i}\,\underline{\omega}\,\zeta\,\underline{e}\,V_j$
    end for
  end for
  ▷ Compute pairwise contribution to global force density vector
  for each element i **do**
    for each element j in neighbor list for element i **do**
      $\zeta \leftarrow \|\mathbf{x}_j - \mathbf{x}_i\|$
      $\eta \leftarrow \|\mathbf{y}_j - \mathbf{y}_i\|$
      $\omega \leftarrow \exp\left(-\frac{\zeta^2}{\delta^2}\right)$
      $\underline{e}^d \leftarrow \eta - \zeta - \frac{\theta\,\zeta}{3}$
      $\underline{t} \leftarrow \frac{3}{m_i}\,k\,\theta\,\underline{\omega}_i\,\underline{\zeta} + \frac{15\mu}{m}\,\underline{\omega}\,\underline{e}^d$
      $\mathbf{M} \leftarrow \frac{\mathbf{y}_j - \mathbf{y}_i}{\|\mathbf{y}_j - \mathbf{y}_i\|}$
      $\mathbf{f}_i \leftarrow \mathbf{f}_i + \underline{t}\mathbf{M}V_j$
      $\mathbf{f}_j \leftarrow \mathbf{f}_j - \underline{t}\mathbf{M}V_i$
    end for
  end for
end procedure

perturbation size in *Peridigm*, for example, is $1.0e^{-6}$ times the characteristic element size.

The central difference scheme for constructing the tangent stiffness matrix involves perturbing individual displacement degrees of freedom and examining the resulting change in each component of the global force vector. To minimize computational expense, the evaluation of the internal force under a perturbation of a single displacement degree of freedom should be restricted to include only those components of the internal force vector that are directly affected. As described above, this subset of the internal force vector contains, at most, the components associated with the neighbor list of the perturbed node and the neighbor lists of each neighbor of the perturbed node.

The *Peridigm* algorithm for evaluating the tangent stiffness matrix by finite difference operates by traversing each node in the discretization and evaluating the force state at that node under a perturbation of each degree of freedom in that node's neighbor list. Each time the force state is evaluated, it is applied to each bond in the neighbor list and multiplied by the volumes connected by that bond. The resulting force value is then divided by the perturbation length and assembled into the proper locations in the tangent stiffness matrix. This approach results in the minimum number of force state evaluations for construction of the tangent stiffness matrix. An assumption implicit to this approach is that the peridynamic force state at a given material point can be determined based only on the constitutive model data (and history) at that point, plus basic kinematic data (*e.g.*, position, velocity) at each neighbor. The corresponding pseudocode is presented in Figure 3.

18.4 Modeling contact

Modeling contact is important for many applications of peridynamics. Impact, penetration, and fragmentation simulations, for example, require the ability to capture multi-body contact interactions. This includes bodies that are disconnected at the onset of a simulation, and also those that become disconnected as a result of material failure. Contact modeling in peridynamics has received little attention relative to contact modeling for classical finite-element analysis, which has been addressed extensively in the literature (see, for example, Laursen [8]). To date, the most common approach to modeling contact in peridynamics is the short-range force approach of Silling and Askari [17]. The short-range force approach models contact interactions between bodies using methods similar to those of molecular dynamics. At each step in the simulation, spring-like repulsive forces are applied between elements that are in close proximity to one another. This approach produces acceptable results in many cases. Alternatively, it has been demonstrated that a classical (local) contact algorithm may be applied within a peridynamic simulation [9]. Here, an iterative penalty approach was applied between elements to minimize contact gaps and disallow interpenetration.

Algorithm 3 Construction of the tangent stiffness matrix by finite-difference.

procedure TANGENT STIFFNESS MATRIX
  ▷ Initialize the tangent stiffness matrix to zero
  $\mathbf{K} \leftarrow \mathbf{0}$
  ▷ Traverse each element in the discretization
  for each element i **do**
    $\{traversal\ list\} \leftarrow$ element i and all neighbors of element i
    ▷ Evaluate the force state at i under perturbations of $\mathbf{u}$
    for each element j in $\{traversal\ list\}$ **do**
      for each displacement degree of freedom r at element j **do**
        $\underline{\mathbf{T}}^{\varepsilon+} \leftarrow \underline{\mathbf{T}}[\mathbf{x}_i](\mathbf{u} + \varepsilon_r)$
        $\underline{\mathbf{T}}^{\varepsilon-} \leftarrow \underline{\mathbf{T}}[\mathbf{x}_i](\mathbf{u} - \varepsilon_r)$
        for each element k in neighbor list of element i **do**
          $\mathbf{g}^{\varepsilon+} \leftarrow \underline{\mathbf{T}}^{\varepsilon+} \langle \mathbf{x}_k - \mathbf{x}_i \rangle \, \Delta V_i \, \Delta V_j$
          $\mathbf{g}^{\varepsilon-} \leftarrow \underline{\mathbf{T}}^{\varepsilon-} \langle \mathbf{x}_k - \mathbf{x}_i \rangle \, \Delta V_i \, \Delta V_j$
          $\mathbf{g} \leftarrow \mathbf{g}^{\varepsilon+} - \mathbf{g}^{\varepsilon-}$
          for each degree of freedom s at element k **do**
            $\mathrm{K}_{sr} \leftarrow \mathrm{K}_{sr} + \frac{g_s}{2\varepsilon}$
          end for
        end for
      end for
    end for
  end for
end procedure

The basic components of a contact model are the detection algorithm and the enforcement algorithm. The detection algorithm performs a proximity search in the current (deformed) configuration. The proximity search required for contact is not different in principle from the proximity search required for the creation of neighbor lists. The *Peridigm* code executes the proximity search for contact in three phases: geometry-based parallel decomposition, identification and communication of off-processor data, and execution of an on-processor proximity search. The first two steps are required only for parallel simulations. Their goal is to reduce the parallel search to a set of serial searches. As described in Section 18.5, this is achieved by creating geometry-based parallel partitions and expanding the partitions by a distance equal to the horizon, thus collecting on each partition the necessary data to identify neighbors for the on-processor elements. The serial proximity search may then be carried out using a number of approaches, such as those based on quadtree or k-d tree data structures. Because elements travel a finite distance over a time step, it is occasionally necessary to consider element paths as opposed to simply examining the element positions at the end of the step (*e.g.*, in hypervelocity simulations).

The short-range force algorithm consists of pairwise repulsive forces that increase in magnitude as the distance between elements decreases. The distance between elements is typically determined as the distance between the element centroids. As illustrated in Algorithm 4, the short-range force calculation is extremely

Algorithm 4 The short-range force contact algorithm applies a pairwise repulsive force that resists interpenetration.

procedure SHORT-RANGE FORCE CONTACT ALGORITHM
  ▷ Initialize the contact forces to zero.
  for each element i **do**
    $\mathbf{f}_i^{\text{contact}} \leftarrow \mathbf{0}$
  end for
  ▷ For each element, examine the results of the global proximity search.
  for each element i **do**
    for each element j in proximity of element i **do**
      $d^2 \leftarrow (\mathbf{y}_j - \mathbf{y}_i) \cdot (\mathbf{y}_j - \mathbf{y}_i)$
      if $d^2 < d_0^2$ **then**
        $f_c \leftarrow \left(\frac{9C}{\pi \delta^4}\right)\left(\frac{d_0 - \sqrt{d^2}}{\delta}\right)$
        $\mathbf{M} \leftarrow \frac{\mathbf{y}_j - \mathbf{y}_i}{\|\mathbf{y}_j - \mathbf{y}_i\|}$
        $\mathbf{f}_i^{\text{contact}} \leftarrow \mathbf{f}_i^{\text{contact}} - f_c V_j \mathbf{M}$
        $\mathbf{f}_j^{\text{contact}} \leftarrow \mathbf{f}_j^{\text{contact}} + f_c V_i \mathbf{M}$
      end if
    end for
  end for
end procedure

simple. Strengths of the short-range force approach include ease of implementation and relatively low computational cost.

Friction effects may be included through extensions to the short-range force model. The kinematic information required for a Coulomb friction model, for example, may be determined by examining the relative motion of any two elements interacting via contact.

An additional extension of the short-range force model concerns peridynamic bonds. In its most simple form, the short-range force model does not take peridynamic bonds into account. This results in the possibility that elements may interact through the material model and the contact model simultaneously, for example when a body is subjected to large hydrostatic compression. This effectively results in rapid material stiffening, often well beyond the stiffness prescribed by the constitutive model. This scenario can be avoided by considering neighbor list information in the contact model and disallowing contact forces between bonded elements.

The computational expense of detecting and enforcing contact interactions often comprises a significant portion of the overall simulation cost. One strategy of reducing this expense is to limit the range of possible contact interactions to include only pairs of material blocks or node sets. Further, so-called self contact may be disabled by designating that contact interactions may not occur between any two elements that are within the same material block.

The short-range force algorithm does not perform well under all conditions. One source of difficulty is the tracking of surfaces. The short-range force model represents surfaces as a collection of undeformable spheres. This approach breaks down as gaps appear between the spheres under large deformation. This is exacerbated in cases where the peridynamic discretization is created from a nonuniform hexahedron or tetrahedron mesh. In this case, slender elements are replaced by spheres that yield poor representations of continuous surfaces. Poor surface approximations may lead to nonphysical interpenetration and subsequent numerical difficulties. A possible solution is the construction of explicit surface representations, for example using a level-set approach.

Contact modeling in peridynamic simulations remains an open area of research. To date, contact has been applied in peridynamic simulations only for explicit transient dynamic simulations. A penalty- or constraint-based approach is more likely to be successful with implicit time integration. Furthermore, general issues related to nonlocality in contact have not been explored. The interaction distance for the short-range force model is typically assigned a value approximately equal to the characteristic element size, whereas the interaction distance for the constitutive model is equal to the horizon. In this scenario, the contact model is effectively a local model, in contrast to the constitutive law, which is nonlocal.

18.5 Meshfree discretizations for peridynamics

The meshfree approach of Silling and Askari requires that the model domain be discretized into a set of elements, each of which is defined by a single point having initial coordinates, (x, y, z), and volume, V. Elements may be grouped into a set of element blocks, each tied to a specific material model and set of constitutive parameters. *Peridigm* defines a discretization using tuples of the form $(x, y, z, V, \text{block\_id})$. Discretizations of this type may be created internally within the code, read from a file, or created by processing a standard hexahedron or tetrahedron mesh (*e.g.*, a mesh created for use in a classical finite-element code). Each element in a discretization may be assigned a unique global ID. This facilitates the definition of node sets, to which initial and boundary conditions are applied. In the case of a parallel code, local (on-processor) IDs may also be assigned, and a mapping maintained between the local and global IDs.

The use of material blocks facilitates simulations involving multiple materials and enables the application of specialized constitutive laws to those bonds that cross material boundaries. Direct application of Equation (18.1) results in an averaging of constitutive responses when computing net pairwise interactions across material boundaries. Alternatively, a constitutive law that is specific to a material interface may be applied, for example a bond failure law that governs material damage across grain boundaries in a polycrystalline material.

Mesh generation tools developed for finite-element analysis can be applied to generate discretizations for peridynamic simulations. Mesh generation software typically operates on a geometric description of the domain and produces a mesh of solid hexahedron or tetrahedron elements. Converting a mesh of this type into a meshfree discretization for a peridynamic simulation is straightforward: solid elements are converted to a $(x, y, z, V, \text{block\_id})$ tuple, where (x, y, z) is the centroid of the original element and V is the volume of the original element. This approach yields a one-to-one relationship between the elements in the original finite-element mesh and the meshfree peridynamic discretization. The nodes in the original mesh are not preserved, however, and this complicates the transfer of node sets defined on the original mesh. One approach is to transfer node sets to the peridynamic discretization as follows: for each element in the original mesh, if any node belonging to that element is in a node set, then the single node in the peridynamic mesh corresponding to that element is in the node set.

It is important to recognize several differences between the requirements for a viable peridynamic discretization and those for a classical finite-element mesh. Boundary conditions in the classical local theory are applied over a two-dimensional surface, whereas nonlocal simulations require that constraints be applied over a three-dimensional volume. This complicates the definition of node sets for peridynamic simulations due to the fact that mesh-generation tools do not typically include functionality for defining three-dimensional layers along domain boundaries. The use of a highly-graded mesh, which is common in classical finite element analyses, is an

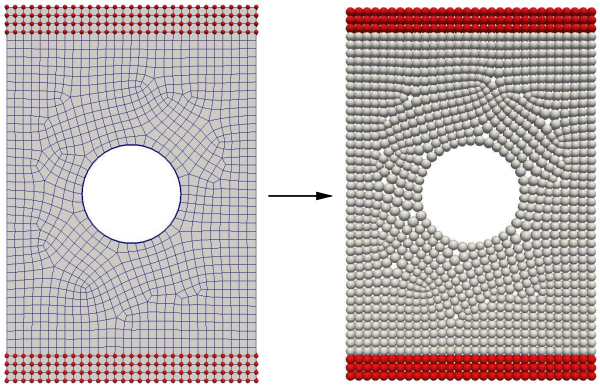


Figure 18.1
Conversion of a standard finite-element mesh to a meshfree discretization. Nodes belonging to a node set are shown in red.

additional source of difficulty. The use of a highly-graded discretization in peridynamic simulations is problematic because the ratio of the characteristic element size to the horizon varies wildly over the domain. Experience has shown that employing a horizon that is roughly three times the characteristic element size produces satisfactory results in may cases. Computational expense can increase dramatically if the element size is reduced relative to the horizon size, which occurs in the highly-refined regions of a graded mesh. A final point on the requirements for a viable peridynamic discretization concerns the chosen length scale. In many peridynamic constitutive models, the value of the horizon is raised to the fourth power, for example in the prototype microelastic brittle material [17]. For this reason, the unit of length should be chosen such that the value of the horizon is of order one to avoid potential numerical difficulties.

18.6 Proximity search for identification of peridynamic bonds

A proximity search is required to identify the neighbors of any given material point in a peridynamic body. A so-called neighbor list is a record of these points, for example a list of element IDs for all the elements that comprise a given family. Because the meshfree approach of Silling and Askari is a Lagrangian approach, neighbor lists do not change as a result of deformation and may be created at the onset of a simulation.

The creation of neighbor lists, while straightforward in theory, presents several

challenges in practice. For a general unstructured discretization, a global proximity search must be executed to find all elements within the horizon, δ, that define the family of each material point in the model. Typically, a single value of the peridynamic horizon is assigned to the entire computational domain. The use of a variable horizon within a simulation, while not disallowed by the peridynamic theory, can produce nonphysical results and should be avoided in general. The *Peridigm* code utilizes the *Zoltan* software package for the creation of neighbor lists [5]. The *Zoltan* package contains routines with a variety of applications to parallel partitioning, load balancing, and data management, a number of which facilitate efficient proximity searches. Search algorithms based on quadtree or *k*-d tree data structures provide additional alternatives. In the case of a parallel code, proximity searches require an initial global decomposition phase. One strategy for this initial step is to create a geometrically-based parallel decomposition, for example using the Recursive Coordinate Bisection (RCB) algorithm [4]. Expanding the bounding box for a given partition by the length δ in each dimension allows for the identification of all potential off-processor neighbors to the elements in that partition.

The creation of neighbor lists is complicated by elements that lie partially inside and partially outside a given element's horizon. Consider the case where an element, $\mathbf{q}$, may lie within the horizon of an element, $\mathbf{x}$. The most straightforward approach is to simply examine the centroids of the elements. Under this approach, the element $\mathbf{q}$ is added to the neighbor list of $\mathbf{x}$ if and only if the distance between centroids is less than δ. While extremely simple to apply, this approach results in a crude approximation of the family of $\mathbf{x}$ and has been shown to produce unsatisfactory simulation results in some cases. Improvement can be made by applying a scalar correction factor that reduces the pairwise forces between $\mathbf{q}$ and $\mathbf{x}$ when only part of the volume of $\mathbf{q}$ lies within the horizon of $\mathbf{x}$. A more accurate approach is to explicitly compute the portion of the volume of $\mathbf{q}$ that falls within the horizon of $\mathbf{x}$ [14]. Further, the bond between $\mathbf{x}$ and $\mathbf{q}$ may be computed using the centroid of this partial volume of $\mathbf{q}$. While advantageous from an accuracy standpoint, this technique adds significant complexity to the proximity search. A final point regarding the intersection of a given element's horizon with neighboring elements is the potentially mitigating effect of the influence function for state-based models. An influence function that reduces smoothly to zero as bond length approaches the horizon may reduce the impact of discretization errors at horizon boundaries [14].

Care must be taken when constructing neighbor lists in the vicinity of small geometric features. The danger is that bonds may pass across features that are small relative to the horizon size, for example small holes or notches, in a way that is nonphysical. One possible solution is to include in the proximity search process a set of geometric entities through which bonds may not pass. For example, a finite plane may be used to define a pre-crack. Any bond that would pass through the pre-crack plane is excluded from the model. This approach can be made quite general in the case of a discretization created from a standard finite-element mesh. Shell elements, for instance, provide a means to define planes that can be used in the proximity search process. A second, simpler approach to controlling the construction of neighbor lists is to explicitly allow or disallow bond creation between specific material blocks.

18.7 Time integration

Numerical time integration plays a central role in engineering analysis codes. Conceptually, it is the time integrator that drives the simulation and orchestrates the passing of information to and from various computational kernels. Time integration schemes for peridynamic models do not differ significantly from those of classical, local models, for which a vast literature exists. Standard approaches for engineering codes can be found in the excellent books by Hughes [7] and by Belytschko, Liu, and Moran [3]. The time integration strategies for peridynamic models presented below were adopted directly from these texts.

Time integration schemes fall into two broad categories: explicit and implicit. Explicit time integration is typically utilized for transient simulations in which inertial effects play an important role. It is extremely robust, and as such is well suited for peridynamic simulations involving large deformations and pervasive material failure. The great drawback of explicit time integration is that it is only conditionally stable, which limits the maximum allowable size of the time step. Implicit methods constitute a second class of time integration schemes. They are unconditionally stable, allowing for much larger time steps than those permitted by explicit approaches. Implicit time integration, however, requires the solution of a system of equations involving both the current state and a future state of the system. In some cases it may be advantageous to apply multiple time integrators, in sequence, within a single simulation. An example is mechanical or thermal preloading, which may be simulated as a quasi-static process, followed by a dynamic process such as impact.

18.7.1 Explicit time integration for transient dynamics

An explicit time integration scheme well suited for application to peridynamics is outlined below. It is a central difference scheme in which velocities are defined at the midpoints of the time intervals, as detailed in [3].

1. Initialize $n = 0$, $t = 0$, $\mathbf{d} = \mathbf{0}$, and $\mathbf{a} = \mathbf{0}$. Set initial conditions for $\mathbf{v}$ and initialize material state variables (if any).
2. Estimate the maximum stable time step, Δt_{crit}, and determine the simulation time step, Δt.
3. Update the times t^{n+1} and $t^{n+\frac{1}{2}}$: $\quad t^{n+1} = t^n + \Delta t, \quad t^{n+\frac{1}{2}} = \frac{1}{2}\left(t^n + t^{n+1}\right)$.
4. First partial velocity update, $\mathbf{v}^{n+\frac{1}{2}} = \mathbf{v}^n + \left(t^{n+\frac{1}{2}} - t^n\right)\mathbf{a}^n$.
5. Enforce velocity boundary conditions by setting $\mathbf{v}^{n+\frac{1}{2}}$ for nodes with velocity boundary conditions. Displacement boundary conditions can be converted to velocities and applied here, or they can be applied directly after step 7.
6. Update nodal displacements, $\mathbf{d}^{n+1} = \mathbf{d}^n + \mathbf{v}^{n+\frac{1}{2}}\Delta t$.

7. Evaluate the internal force by means of the damage model and constitutive model for $\mathbf{d}^{n+1}$ and $\mathbf{v}^{n+\frac{1}{2}}$. Update material state variables accordingly. Optionally, recompute the estimate for the critical time step, Δt_{crit}, and update the simulation time step, Δt.

8. Evaluate contact forces for $\mathbf{d}^{n+1}$ and $\mathbf{v}^{n+\frac{1}{2}}$.

9. Compute acceleration $\mathbf{a}^{n+1} = \mathbf{M}^{-1}\mathbf{f}^{n+1}$, where $\mathbf{f}^{n+1} = \mathbf{f}_{int}^{n+1} + \mathbf{f}_{ext}^{n+1} + \mathbf{f}_{contact}^{n+1}$.

10. Second partial velocity update, $\mathbf{v}^{n+1} = \mathbf{v}^{n+\frac{1}{2}} + \left(t^{n+1} - t^{n+\frac{1}{2}}\right)\mathbf{a}^{n+1}$.

11. Check energy balance as a check on stability.

12. If simulation is not complete, update $n = n+1$, go to Step 3.

The first step in the time integration scheme is initialization. The time, t, and time step, n, as well as nodal values for displacement, $\mathbf{d}$, and acceleration, $\mathbf{a}$, are set to zero. The nodal velocities, $\mathbf{v}$, are be set to reflect initial conditions on the velocity field. Variables for representing material states within a constitutive model must also be initialized in Step 1. The maximum stable time step, commonly referred to as the critical time step, Δt_{crit}, is determined in Step 2, as described below. The time step, Δt, is typically determined by multiplying the critical time step by a safety factor. To guard against numerical instability, the safety factor must be less than one; typical values fall in the range of 0.7 to 0.9. A constant value of Δt may be applied throughout the course of the simulation, or, alternatively, the time step may be recomputed as part of Step 7 to reflect changes in model geometry and material state.

The time-stepping process begins with Step 3, in which both the time at end of the step, t^{n+1}, and the mid-step time, $t^{n+\frac{1}{2}}$, are computed. The midstep velocity is then determined in Step 4 using the current values for velocity and acceleration. Step 5 concerns the application of prescribed displacements and velocities. Prescribed velocities may be applied directly. Prescribed displacements may be converted to velocities through the relation

$$\mathbf{v}^{n+\frac{1}{2}} = \frac{1}{\Delta t}\left(\mathbf{d}^{n+1} - \mathbf{d}^{n}\right). \tag{18.4}$$

The mid-step velocities, $\mathbf{v}^{n+\frac{1}{2}}$, are then applied to determine the displacements at the end of the time step, $\mathbf{d}^{n+1}$, in Step 6.

The constitutive model, damage model, and contact model are invoked in Step 7 for evaluation of the nodal forces, $\mathbf{f}^{n+1}$. The inputs to these models are the displacements at the end of the time step, $\mathbf{d}^{n+1}$, and the mid-step velocities, $\mathbf{v}^{n+\frac{1}{2}}$, as well as the material state variables stored at step n. The damage law may be applied first, to determine the current state of damage for each bond, followed by evaluation of the constitutive model. Contact forces, if any, may generally be determined independently from the material-model response. Step 7 also includes the application of

prescribed, external body forces, $\mathbf{f}_{ext}^{n+1}$. Contact forces, constitutive model forces, and prescribed body forces are summed together to determine the net force on each node.

Following determination of nodal forces, the nodal accelerations, $\mathbf{a}^{n+1}$, are computed by dividing by the mass associated with each node. The resulting acceleration values are then applied to determine the velocities at the end of the time step, $\mathbf{v}^{n+1}$, in Step 10. The time stepping process then repeats by returning to Step 3 until the simulation is complete.

Prior to incrementing to the next time step, it may be advantageous to carry out a check on global energies to ensure that the simulation remains numerically stable (Step 12). Instability, for example resulting from an inaccurate estimate of the critical time step, often results in a large, nonphysical increase in the total energy of the system.

18.7.2 Estimating the maximum stable time step

A means for estimating the critical time step for the prototype microelastic brittle model was published by Silling and Askari [17],

$$\Delta t_{\text{crit}} = \sqrt{\frac{2\rho}{\sum_p V_p C_{ip}}}, \tag{18.5}$$

where ρ is the density, p iterates over all bonds at node i, V_p is the volume associated with neighbor p, and C_{ip} is the micromodulus between nodes i and p. Equation (18.5) was derived for the one-dimensional case. Its application in two- and three-dimensional simulations results in a conservative estimate of the time step. (Effectively, it assumes that all bonds for a given node are colinear.)

It has been demonstrated that Equation (18.5) may provide a reasonable estimate of the maximum stable time step for other constitutive models, for example the state-based linear peridynamic solid [10]. In these case, the micromodulus, C_{ip}, appearing in the denominator of Equation (18.5), is replaced by an effective micromodulus,

$$C_{ip}^{\text{eff}} = \frac{1}{|\xi|}\,\frac{18\,k}{\pi\delta^4}, \tag{18.6}$$

where $|\xi|$ is the distance between nodes i and p in the reference configuration, k is the bulk modulus, and δ is the horizon. The second term in Equation 18.6 is taken from Silling and Askari [17]; it is the spring constant for a prototype microelastic brittle material having bulk modulus k.

Another option for estimating the maximum critical time step is the well-known Courant-Friedrichs-Lewy (CFL) approach [6],

$$\Delta t_{\text{crit}} = \frac{h}{c}, \tag{18.7}$$

where h is the characteristic mesh size and c is the wave speed,

$$c = \sqrt{\frac{k}{\rho}}. \tag{18.8}$$

Equation (18.7) generally yields a very conservative estimate of the critical time step for peridynamic models. This is because wave propagation in peridynamic models is largely dictated by the size of the horizon. The CFL limit is commonly applied in classical finite-element analyses, where nodes interact directly only when connected by a common element. In contrast, nodes in a peridynamic simulation may interact directly when separated by as much as twice the horizon. A natural question regarding Equation 18.7 is, can the horizon be used as the value for h? In preliminary numerical experiments, this approach yielded a nonconservative critical time step [10].

18.7.3 Implicit time integration for quasi-statics

A time integration scheme is presented below for nonlinear quasi-static peridynamic simulations. Nonlinear quasi-statics is one modeling strategy within a family of implicit approaches. Other formulations include the well-known Newmark-β methods, which include intertial terms, and varients that are specific to linear problems. All implicit methods require the solution of a system of equations. For this reason, they are significantly more difficult to implement and are more computationally demanding than explicit methods.

The governing equation for quasi-static simulations is obtained by setting the acceleration term to zero in Equation (18.1),

$$\sum_{\mathcal{B}} \left(\underline{\mathbf{T}}[\mathbf{x}] \langle \mathbf{q} - \mathbf{x} \rangle - \underline{\mathbf{T}}[\mathbf{q}] \langle \mathbf{x} - \mathbf{q} \rangle \right) dV_{\mathbf{q}} + \mathbf{b}(\mathbf{x}) = \mathbf{0}. \tag{18.9}$$

It is assumed that the system is in static equilibrium at each load step in a quasi-static simulation. Dynamic (inertial) effects, such as wave propagation, are neglected. The simulation progresses through a series of updates to the boundary conditions. Following each update of the boundary conditions, a nonlinear solver is applied to minimize a global residual. The residual, r, is generally defined as a norm of the residual vector, $\mathbf{r}$. The residual vector is defined as the nodal force vector with the degrees of freedom subject to kinematic boundary conditions omitted. The unconstrained displacement degrees of freedom are treated as unknowns. A load step is complete when the residual has dropped below a specified threshold value.

The solution process for quasi-static peridynamic simulations using Newton's method as the nonlinear solver may be written as follows:

1. Initialize $n = 0$, $\mathbf{d} = \mathbf{0}$, and $\mathbf{v} = \mathbf{0}$. Initialize material state variables.
2. Update the load step $n = n + 1$ and pseudo-time t. Update the boundary conditions.
3. Evaluate the residual vector, $\mathbf{r}$, and residual r. Determine the convergence criterion for the load step.
4. Set the trial displacement $\mathbf{d}_{\text{trial}} = \mathbf{d}^n$.
5. Apply Newton's method to minimize the residual.

(a) Construct the tangent stiffness matrix, $\mathbf{K}$, for the configuration $\mathbf{d}_{\text{trial}}$ (recompute, reuse, or modify).

(b) Solve linear system $\mathbf{K}\,\Delta\mathbf{d} = -\mathbf{r}$ for the Newton step, $\Delta\mathbf{d}$.

(c) Set $\mathbf{d}_{\text{trial}} = \mathbf{d}_{\text{trial}} + \Delta\mathbf{d}$.

(d) Evaluate the residual vector, $\mathbf{r}$, and the residual, r, for the configuration $\mathbf{d}_{\text{trial}}$.

(e) If the convergence criterion is not met, return to 5a.

6. Set $\mathbf{d}^{n+1} = \mathbf{d}_{\text{trial}}$.

7. If simulation not complete, go to 2.

The simulation begins with initialization of all relevant fields, as well as all material state variables associated with the constitutive models. Here, we assume that the system is in equilibrium at the onset of the simulation. Load stepping begins with Step 2, in which an increment of the boundary conditions is applied to the system. This may take the form of updated displacement values, in the case of kinematic boundary conditions, or updated external force values in the case of prescribed body forces or pressures. The extent to which the system is no longer in equilibrium is determined through the residual evaluation in Step 3. The residual is a function (*e.g.*, L2-norm) of the forces corresponding to degrees of freedom not subjected to kinematic boundary conditions. (The forces corresponding to degrees of freedom at which kinematic boundary conditions are applied are generally nonzero and are referred to as reaction forces.) The convergence criterion for the load step is typically computed relative to the initial residual value, although an absolute criterion may also be used.

Solution of the nonlinear problem begins with Step 4. The goal is to determine an increment to the displacement vector, $\Delta\mathbf{d}$, that will reduce the residual such that the convergence criterion is met. While many techniques exist for solving nonlinear problems, Newton's method is perhaps the most straightforward approach. Here, the residual is minimized by solving a series of linear problems. The linear problems are defined as

$$\mathbf{K}\,\Delta\mathbf{d} = -\mathbf{r} \tag{18.10}$$

where $\mathbf{K}$ is the tangent stiffness matrix, $\mathbf{r}$ is the residual vector, and $\Delta\mathbf{d}$ is an (unknown) increment in the nodal displacements. The tangent matrix is a linearization of the system about the current trial displacement, $\mathbf{d}_{\text{trial}}$, as described in Section 18.3. Prior to solving the linear system in Equation 18.10, the tangent stiffness matrix and right-hand side vector are typically modified to reflect kinematic boundary conditions. One strategy is to set the rows and colums of the stiffness matrix corresponding to kinematic boundary conditions to zero, with a scalar on the diagonal (*e.g.*, the L2 norm of the unmodified diagonal). The entries in the residual vector corresponding to kinematic boundary conditions are also set to zero. The net result is that the increment in displacement, Δd, for degrees of freedom corresponding to kinematic boundary conditions is zero.

A quasi-static load step is deemed to be converged with the residual drops below a specified threshold, at which time the solution procedure advances to Step 6. Here,

both the field variables and the material state variables are set equal to their trial values. The load stepping process is then repeated until the simulation is complete.

18.8 Example simulations

Example simulations carried out with the explicit and implicit time integration schemes described in Section 18.7 are presented below. The first is an explicit transient dynamics simulation of fracture that results from a projectile impacting a brittle disk. It is an intial value problem in which an initial velocity is assigned to the projectile. The second is a quasi-static simulation of a tensile test. In this case, the simulation is driven by prescribed displacement boundary conditions applied to the end portions of the tensile specimen.

18.8.1 Fragmentation of a brittle disk resulting from impact

A peridynamic model of a fragmenting brittle disk is illustrated in Figure 18.2. It is a modified version of the brittle fragmentation simulation presented by Silling and Askari [17]. The simulation was carried out using the *Peridigm* code via explicit time integration. Constitutive model parameters for the disk and projectile, which were assigned different material blocks, are given in Tables 18.1 and 18.2, respectively. The geometry of the disk is a cylinder of radius 37.0 *mm* and height 2.5 *mm*. The discretization for the disk was created by converting an unstructured hexahedron mesh, created with the *Cubit* mesh generation code [1], into a meshfree discretization containing approximately 180,000 elements. The geometry of the projectile is a sphere with radius 5.0 *mm*. The discretization for the projectile was created by converting a tetrahedron mesh, created with *Cubit*, into a meshfree representation containing approximately 18,000 elements. The peridynamic horizon for both the disk and the projectile was set to roughly three times the node spacing. Contact in the fragmenting disk simulation was modeled with the short-range force contact model. The contact radius was set to 0.775 *mm* and the spring constant to 1000.00 *GPa*. The disk was initially at rest, and the projectile was assigned an initial velocity of 100.0 m/s. The simulation was run to a final simulation time of 400.0 μs. A constant time step of 0.0416 μs (0.7 times the estimated critical time step) was used throughout the simulation. The fragmenting disk simulation was run in parallel using ten 3.0 GHz Intel Xeon processors, requiring a total run time of approximately 2.8 hours.

18.8.2 Quasi-static simulation of a tensile test

A peridynamic simulation of a tensile bar experiment is illustrated in Figure 18.3. The simulation was carried out in *Peridigm* using quasi-static time integration. Constitutive model parameters for the bar are given in Table 18.3. The bar was modeled using an elastic correspondence constitutive model [16]. An additional stabilization

Table 18.1
Constitutive parameters for the disk, modeled using the linear peridynamic solid constitutive model and the critical stretch bond failure law.

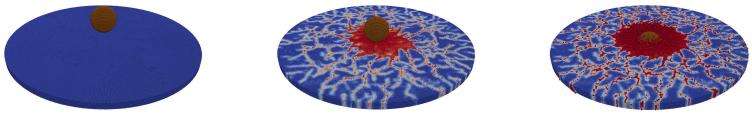
Parameter	Value
Horizon	1.17 mm
Density	2.20 g/cm^3
Bulk Modulus	14.90 GPa
Shear Modulus	8.94 GPa
Critical Stretch	0.0005

Table 18.2
Constitutive parameters for the projectile, modeled using the linear peridynamic solid constitutive model. No bond failure law was assigned to the projectile (*i.e.*, bond failure was disallowed in the projectile).

Parameter	Value
Horizon	1.17 mm
Density	7.70 g/cm^3
Bulk Modulus	160.00 GPa
Shear Modulus	78.30 GPa

term was applied to mitigate the impact of low-energy modes of deformation on the solution [9]. The length of the tensile bar is 10.16 *cm*, with a maximum width of 1.27 *cm*, a minimum width of 0.635 *cm*, and a thickness of 0.315 *cm*. The discretization was created by converting an unstructured hexahedron mesh, created using *Cubit*, to a meshfree representation containing approximately 99,000 elements.

Boundary conditions were applied in the form of prescribed displacements over volumes at the ends of the bar. In each case, the volume over which prescribed displacements were applied was defined as spanning from the end of the bar to a distance equal to two times the horizon inward from the end of the bar. Nodes within these volumes were assigned an initial displacement of zero and a final displacement equal to $0.005x$, where x is the distance from the center of the bar. The result is a linear dis-



(a) 0 μs. (b) 44 μs. (c) 131 μs.

Figure 18.2
Explicit dynamic simulation of brittle fracture resulting from impact.

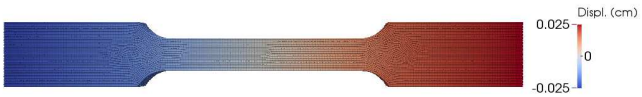
Table 18.3
Constitutive parameters for tensile bar. The projectile is modeled using a linear elastic correspondence constitutive law.

Parameter	Value
Horizon	0.09525 cm
Density	8.00 g/cm^3
Young's Modulus	180.0 GPa
Poisson's Ratio	0.30
Stabilization Coefficient	0.02

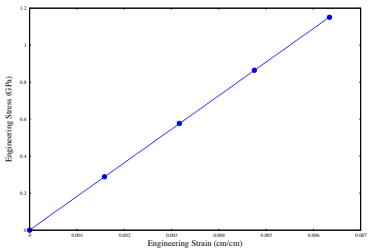
placement field corresponding to an engineering strain of 0.5%. This displacement field is an approximation of the true displacement field that would occur in a physical experiment. It is expected that, due to the approximate nature of the displacement boundary conditions, nonphysical artifacts will be present in and around the end segments of the bar. These artifacts are are not expected to signifcantly pollute the solution in the narrow section of the bar.

The tensile bar simulation was divided into four load steps and solved using the general approach for nonlinear quasi-statics described in Section 18.7.3. A convergence criterion was defined in terms of both the L2 norm and the infinity norm of the residual vector, $20\|\mathbf{r}\|_\infty + \|\mathbf{r}\|_{L_2} \leq 1.0 dyne$. Here, the infinity norm is used in addition to the L2 norm to avoid the acceptance of solutions in which the residual is large in highly-localized areas, but the overall L2 norm of the residual is small. The residual value at the onset of each load step was approximately nine orders of magnitude larger than the convergence criterion. The tensile bar simulation was run in parallel using ten 3.0 GHz Intel Xeon processors, requiring a total run time of approximately 1.3 hours.

The tensile bar simulation was designed to mimic a standard engineering tensile test experiment. One purpose of a tensile test experiment is to determine a material's elastic modulus. This is typically done by monitoring the elongation of a small segment of the bar using a strain gauge, and a load cell that tracks the total force applied to the bar. These data are then used to compute the engineering strain (final length of the strain gauge divided by its initial length) and the engineering stress (total force divided by the initial cross-sectional area of the narrow section of the bar). The elastic modulus is computed as the ratio of the engineering stress to the engineering strain. This procedure was modeled within the peridynamic simulation by tracking the displacement of a pair of nodes as well as the net reaction forces in the end segments of the bar. The pair of nodes was chosen to approximate the positions at which the ends of a one-inch (2.54 *cm*) strain gauge would be located, assuming the strain gauge was centered along the length of the bar. The resulting engineering stress-strain curve is presented in Figure 18.3(b). The value of Young's modulus calculated in this manner was 181.5 *GPa*, which differs from the prescribed value of 180.0 *GPa* by 0.83%.



(a) Displacement in the loading direction.



(b) Computed engineering stress-strain curve. The recovered elastic modulus is 181.5 GPa.

Figure 18.3
Quasi-static simulation of a tensile test.

18.9 Summary

The application of peridynamics for computational simulation depends critically on an efficient and robust software implementation. In the preceding sections, a roadmap was presented for implementation of the meshfree approach of Silling and Askari, which is the discretization technique used for the vast majority of peridynamic simulations to date. Codes developed at Sandia National Laboratories, including *EMU*, *Peridigm*, *LAMMPS*, and *Sierra/SolidMechanics*, contain peridynamics functionality based on this approach.

Nonlocality pervades many aspects of a peridynamic simulation code. This is apparent within the internal force evaluation, where constitutive model routines must traverse a neighbor list data structure, and in the tangent stiffness matrix, which is significantly more dense that its counterpart in the local theory. The material in the preceding sections draws from experience implementing both standalone peridynamics codes and peridynamics functionality within larger software packages. It is hoped that the material presented will provide clarity on the implementation of peridynamic theory, as presented in the literature, in a computational simulation code.

References

[1] Cubit mesh generation code. `http://cubit.sandia.gov`.

[2] Peridigm peridynamics code. `http://peridigm.sandia.gov`.

[3] Ted Belytschko, Wing Kam Liu, and Brian Moran. *Nonlinear Finite Elements for Continua and Structures*. John Wiley & Sons, Ltd., Chichester, England, 2000.

[4] M. Berger and S. Bokhari. A partitioning strategy for nonuniform problems on multiprocessors. *IEEE Transactions on Computers*, C-36(5):570–580, 1987.

[5] E. G. Boman, U. V. Catalyurek, C. Chevalier, and K. D. Devine. The Zoltan and Isorropia parallel toolkits for combinatorial scientific computing: partitioning, ordering, and coloring. *Scientific Programming*, 20(2), 2012.

[6] R. Courant, K.O. Friedrichs, and H. Lewy. Über die partiellen differenzensleichungen der mathematischen physik. *Mathematische Annalen*, 100(1):32–74, 1928.

[7] Thomas J.R. Hughes. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1987.

[8] Tod A. Laursen. *Computational Contact and Impact Mechanics: Fundamentals of Modeling Interfacial Phenomena in Nonlinear Finite Element Analysis*. Springer-Verlag, Heidelberg, Germany, 2002.

[9] David Littlewood. Simulation of dynamic fracture using peridynamics, finite element modeling, and contact. In *Proceedings of the ASME 2010 International Mechanical Engineering Congress and Exposition (IMECE)*, Vancouver, British Columbia, Canada, 2010.

[10] David Littlewood, Jesse Thomas, and Timothy Shelton. Estimation of the critical time step for peridynamic models. In *Presented at the SIAM Conference on Mathematical Aspects of Materials Science*, Philadelphia, Pennsylvania, 2013.

[11] Richard W. Macek and Stewart A. Silling. Peridynamics via finite element analysis. *Finite Elements in Analysis and Design*, 43(15):1169–1178, 2007.

[12] Michael L. Parks, Richard B. Lehoucq, Steven J. Plimpton, and Stewart A. Silling. Implementing peridynamics within a molecular dynamics code. *Computer Physics Communications*, 179(11):777–783, 2008.

[13] Michael L. Parks, David J. Littlewood, John A. Mitchell, and Stewart A. Silling. Peridigm users' guide v1.0.0. SAND Report 2012-7800, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, 2012.

[14] Pablo Seleson. Improved one-point quadrature algorithms for two-dimensional peridynamic models based on analytical calculations. *Computer Methods in Applied Mechanics and Engineering*, 2014. Accepted for publication.

[15] SIERRA Solid Mechanics Team. Sierra/SolidMechanics 4.22 user's guide. SAND Report 2011-7597, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, 2011.

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

[17] 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.