Commun. Comput. Phys.
doi: 10.4208/cicp.OA-2018-0288

Vol. **27**, No. 1, pp. 274-291
January 2020

A Fast State-Based Peridynamic Numerical Model

Ning Du[1], Xu Guo[2,*] and Hong Wang[3]

[1] *School of Mathematics, Shandong University, Jinan 250100, China.*
[2] *Department of Mathematics, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong.*
[3] *Department of Mathematics, University of South Carolina, SC, USA.*

Received 1 November 2018; Accepted (in revised version) 7 March 2019

Abstract. The peridynamic (PD) theory is a reformulation of the classical theory of continuum solid mechanics and is particularly suitable for the representation of discontinuities in displacement fields and the description of cracks and their evolution in materials, which the classical partial differential equation (PDE) models tend to fail to apply. However, the PD models yield numerical methods with dense stiffness matrices which requires $\mathcal{O}(N^2)$ memory and $\mathcal{O}(N^3)$ computational complexity where N is the number of spatial unknowns. Consequently, the PD models are deemed to be computationally very expensive especially for problems in multiple space dimensions.

State-based PD models, which were developed lately, can be treated as a great improvement of the previous bond-based PD models. The state-based PD models have more complicated structures than the bond-based PD models. In this paper we develop a fast collocation method for a state-based linear PD model by exploring the structure of the stiffness matrix of the numerical method. The method has an $\mathcal{O}(N)$ memory requirement and computational complexity of $\mathcal{O}(N\log N)$ per Krylov subspace iteration. Numerical methods are presented to show the utility of the method.

AMS subject classifications: 65F10, 65M70, 65T50, 74B15

Key words: State-based peridynamic model, fast algorithm, collocation method, stiffness matrix.

1 Introduction

The classical theory of continuum solid mechanics assumes that all internal forces act locally and yields mathematical models that are expressed in terms of partial differential equations (PDEs). While they have been very successful for problems with smooth displacement fields, these models tend to have difficulties to describe problems with evolving discontinuities, due to their differentiability assumption on displacement fields. The

*Corresponding author. *Email addresses:* duning@sdu.edu.cn (N. Du), guoxu5861@gmail.com (X. Guo), hwang@math.sc.edu (H. Wang)

peridynamic (PD) theory [17] is a reformulation of the classical theory of continuum solid mechanics, and yields nonlocal mathematical formulations that are based on long-range interactions. Constitutive models in the PD theory depend on finite deformation vectors, instead of deformation gradients in classical constitutive models. Consequently, PD models are particularly suitable for the representation of discontinuities in displacement fields and the description of cracks and their evolution in materials. They have been successfully applied to applications such as failure and damage in composite laminates [13], crack propagation and branching [10], crack nucleation [20], phase transformations in solids [4], impact damage [16,18], polycrystal fracture [9], crystal plasticity [21], damage in concrete [8], and geomaterial fragmentation [11].

Because the PD models provide better modeling capability than classical integer-order PDE models do especially for problems with discontinuous displacement fields, different numerical methods have been developed for these models with different computational expenses, memory requirements, and implementation effort as well as accuracy, convergence, and stability [6]. For instance, collocation methods and meshfree methods apply directly to the strong form of the PD models and are relatively simple to implement [14,15,18], while Galerkin finite element methods apply to the weak form of the PD models and enjoy high convergence rates [2,13,23,24]. It is worth mentioning that in [24] the authors established a framework to develop robust asymptotically compatible schemes for nonlocal models and their local limits.

However, the PD models present numerical difficulties that were not encountered in classical integer-order PDE models. Recall that the numerical methods for integer-order PDE models generate sparse stiffness matrices which have an optimal-order memory requirement of $\mathcal{O}(N)$ where N is the number of spatial unknowns. The matrix-vector multiplication has a computational complexity of $\mathcal{O}(N)$. Hence, the computational complexity of any Krylov subspace iterative solver depends only on the number of iterations. In contrast, the numerical methods for the PD models yield dense stiffness matrices. Consequently, the numerical methods for PD models have $\mathcal{O}(N^2)$ memory requirement. Direct solvers have been widely used in the numerical simulation of PD models, which have $\mathcal{O}(N^3)$ computational complexity to invert the coefficient matrices. On the other hand, the matrix-vector multiplication by the stiffness matrices have $\mathcal{O}(N^2)$ computational complexity. Furthermore, the evaluation and assembly of the stiffness matrices require the evaluation of $\mathcal{O}(N^2)$ entries, which can be very expensive and often constitute a very large portion of simulation times! In summary, the significantly increased computational complexity and memory requirement of the PD models over those for the classical integer-order PDE models render the numerical simulation of the PD models computationally very expensive, especially for problems in multiple space dimensions.

We previously developed fast and accurate numerical methods for bond-based linear PD models in one and two space dimensions. These fast methods reduce the memory requirement from $\mathcal{O}(N^2)$ to the optimal-order memory requirement of $\mathcal{O}(N)$ and computational complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N\log N)$ per Krylov subspace iteration, which were achieved without resorting to any lossy compression, but rather by exploring the

structure of the stiffness matrices of the numerical methods [22, 25–28].

However, it was later realized that the bond-based PD models may sometimes oversimplify the problem and so limit their applications. Subsequently, state-based PD models were developed to improve the original bond-based PD models [7, 12, 19]. Mathematically, state-based PD models have more complicated structures and consequently, the computational complexity of numerical discretizations of the state-based PD models is further increased and the previous developments of the fast numerical methods for the bond-based PD models fail to work.

In this paper we develop a fast collocation method for a state-based linear PD model in one space dimension by exploring the structure of the stiffness matrix of the numerical method. The rest of the paper is organized as follows. In Section 2 we discuss the state-based peridynamic model to be considered in this paper. In Section 3 we carry out a collocation discretization of the state-based peridynamic model. In Section 4 we analyze the structure of the stiffness matrix of the collocation scheme and develop a fast conjugate gradient algorithm with efficient storage. In Section 5 we carry out numerical experiments to investigate the performance of the fast numerical method which shows the utility of the method. Finally, we summarize the conclusion and future extension of this work in Section 6.

2 A state-based linear peridynamic model

In this section we go over a state-based linear PD model in one space dimension, which is to be considered in this paper. We follow the presentations in [7, 12, 14, 15, 19] in the exposition of the model.

2.1 Bond-based and state-based PD models

The original bond-based PD formulation replaces the divergence of the stress field term in the classical conservation of momentum equation with a weakly singular integral operator. The integral operators relates pairwise forces or bonds between material particles in a continuum body possibly with discontinuous displacement fields. A bond-based PD equation of motion is of the form

$$\rho \mathbf{u}_{tt} = \int_H \mathbf{f}\big(\mathbf{u}(\mathbf{x}',t)-\mathbf{u}(\mathbf{x},t),\mathbf{x}'-\mathbf{x}\big)d\mathbf{x}'+\mathbf{b}(\mathbf{x},t), \tag{2.1}$$

where ρ is the mass density, $\mathbf{u}$ is the displacement field, $\mathbf{x}$ is the position vector in the reference configuration, $\mathbf{x}'$ is the position vector of some neighboring material location with respect to $\mathbf{x}$, and $\mathbf{b}$ is the body force density field. H is the horizon which is typically a ball of radius δ with center $\mathbf{x}$.

It was subsequently discovered that bond-based PD models may sometimes oversimplify the problem by assuming that any pair of particles interacts with each other only

through a central force potential that is independent of all other local conditions. Consequently, for an isotropic linear material there will always be an effective Poisson's ratio of 1/4.

The subsequently developed state-based PD models have improved upon the original bond-based PD models through the use of peridynamic force-vector states. These force-vector states handle the constitutive behavior of the material allowing neighboring material points to interact with each other in any fashion desired, not just pairwise [7, 12, 19]. The resulting state-based PD equation of motion can be expressed as

$$\rho\mathbf{u}_{tt}=\int_H\Big\{\mathbf{T}[\mathbf{x},t]\langle\mathbf{x}'-\mathbf{x}\rangle-\mathbf{T}[\mathbf{x}',t]\langle\mathbf{x}-\mathbf{x}'\rangle\Big\}d\mathbf{x}'+\mathbf{b}(\mathbf{x},t), \tag{2.2}$$

where all the definitions for (2.1) hold, and $\mathbf{T}$ denotes the force state field, which maps vectors to vectors but is not necessarily linear or continuous. The angle brackets $\langle\cdot\rangle$ indicate the vector on which the state operates. $\mathbf{T}$ maps a deformation-vector state into a force-vector state for each material point within the horizon H. $\mathbf{T}[\mathbf{x},t]\langle\,_{\bullet}\rangle$ denotes the force state at $\mathbf{x}$ and t, mapping the bond $\xi=\mathbf{x}'-\mathbf{x}$ to force per unit volume squared.

2.2 A one-dimensional state-based linear PD model

In this paper we consider the one-dimensional state-based linear PD model. As the goal of this paper is to develop a fast numerical method through the study of the structure of its stiffness matrix, we consider the following PD model

$$\rho u_{tt}-\frac{1}{\delta^2}\int_{x-\delta}^{x+\delta}\Big\{T[x',t]\langle x-x'\rangle-T[x,t]\langle x'-x\rangle\Big\}dx'=f(x,t),\quad x\in(a,b). \tag{2.3}$$

A linearized linear PD solid constitutive model with force state is of the form

$$\begin{aligned}T[x,t]\langle x'-x\rangle:=&\alpha\,\omega(|x'-x|)(x'-x)\theta^{\rm lin}(x,t)\\&+\frac{\beta}{2}\,\omega(|x'-x|)\big(u(x',t)-u(x,t)\big),\end{aligned} \tag{2.4}$$

where $\alpha=\alpha(G,K,m)$ and $\beta=\beta(G,m)$ are constants that depend on the shear modulus G, the bulk modulus K, and the weighted volume m which is defined by

$$m:=\int_{-\delta}^{\delta}\omega(|\zeta|)|\zeta|^2d\zeta, \tag{2.5}$$

with ω being an influence function. $\theta^{\rm lin}(x,t)$ is the linearized nonlocal dilatation which is defined by

$$\theta^{\rm lin}(x,t):=\frac{1}{m}\int_{x-\delta}^{x+\delta}\omega(|x'-x|)(x'-x)(u(x',t)-u(x,t))dx', \tag{2.6}$$

for $x\in[a-\delta,b+\delta]$.

We use the expression (2.4) and its symmetric analogue with the order of x and x' interchanged to obtain

$$\begin{aligned} &T[x',t]\langle x-x'\rangle - T[x,t]\langle x'-x\rangle \\ =&-\alpha\,\omega(|x'-x|)(x'-x)\big(\theta^{\rm lin}(x,t)+\theta^{\rm lin}(x',t)\big) \\ &-\beta\omega(|x'-x|)\big(u(x',t)-u(x,t)\big). \end{aligned} \tag{2.7}$$

We substitute this expression for the integrand in (2.3) to rewrite the one-dimensional state-based model (2.3) as follows

$$\begin{cases} \rho u_{tt}-\dfrac{1}{\delta^2}\displaystyle\int_{x-\delta}^{x+\delta}\Big[\alpha\,\omega(|x'-x|)(x'-x)\big(\theta^{\rm lin}(x,t)+\theta^{\rm lin}(x',t)\big) \\ \qquad\qquad +\beta\omega(|x'-x|)\big(u(x',t)-u(x,t)\big)\Big]dx'=f(x,t), & x\in[a,b], \\ \theta^{\rm lin}(x,t)=\dfrac{1}{m}\displaystyle\int_{x-\delta}^{x+\delta}\omega(|x'-x|)(x'-x)(u(x',t)-u(x,t))dx', & x\in[a-\delta,b+\delta], \\ u(x,0)=u_0(x),\quad u_t(x,0)=u_1(x), & x\in[a,b], \\ u(x,t)=g(x,t), & x\in[a-2\delta,a]\cup[b,b+2\delta], \end{cases} \tag{2.8}$$

where g is the prescribed nonlocal Dirichlet boundary data on the nonlocal boundary $[a-2\delta,a]\cup[b,b+2\delta]$.

3 A collocation discretization of the state-based linear PD model

We previously developed fast Galerkin and collocation methods that are discretized on uniform or gridded meshes for the PD models in one space dimension by directly proving that the stiffness matrix of the numerical schemes are of Toeplitz or Toeplitz-like structures [22, 26]. We proved that the stiffness matrix of the collocation method of the two-dimensional bond-based linear PD model is of block-Toeplitz-Toeplitz-block type by exploring the tensor product and translation invariance property of the PD model [28]. Consequently, we develop lossless fast numerical methods, without resorting to any lossy compression, but rather by utilize the matrix structures of these fast methods. Numerical experiments show the strength of these fast methods.

However, the integral operator in the state-based PD model (2.8) involves an internal integral operator $\theta^{\rm lin}$. Consequently, the computational complexity of numerical discretizations of the PD model (2.8) is further increased and the previous developments of the fast numerical methods for the bond-based PD model fail to work. To simplify our analysis and focus on the key points in developing a fast method, we consider the

following static state-state PD model instead of (2.8) as

$$\begin{cases} -\displaystyle\int_{x-\delta}^{x+\delta}\Big[\alpha\,\omega(|x'-x|)(x'-x)\theta^{\rm lin}(x') \\ \qquad\qquad +\beta\omega(|x'-x|)\big(u(x')-u(x)\big)\Big]dx'=f(x), & x\in[a,b], \\ \theta^{\rm lin}(x)=\dfrac{1}{m}\displaystyle\int_{x-\delta}^{x+\delta}\omega(|x'-x|)(x'-x)u(x')dx', & x\in[a-\delta,b+\delta], \\ u(x)=g(x), & x\in[a-2\delta,a]\cup[b,b+2\delta]. \end{cases} \tag{3.1}$$

Here we have used the relations

$$\begin{aligned} &\int_{x-\delta}^{x+\delta}\omega(|x'-x|)(x'-x)\theta^{\rm lin}(x)dx'\equiv 0, \\ &\int_{x-\delta}^{x+\delta}\omega(|x'-x|)(x'-x)u(x)dx'\equiv 0. \end{aligned} \tag{3.2}$$

Let $h:=(b-a)/(N+1)$ be the mesh size for some positive integer N and

$$l:=\left\lceil\frac{\delta}{h}\right\rceil, \tag{3.3}$$

where $\lceil\cdot\rceil$ is the ceiling of δ/h. Then we define an equidistant spatial partition over the interval $[a-2\delta,b+2\delta]$

$$x_i:=a+ih,\quad i=-2l,-l+1,\cdots,N+2l+1. \tag{3.4}$$

We note that the nodes $\{x_i\}_{i=1}^N$ are the interior nodes where we need to seek the numerical approximations u_i to $u(x_i)$ for $i=1,2,\cdots,N$. On the other hand, the nodes $\{x_i\}_{i=-2l}^0$ and $\{x_i\}_{i=N+1}^{N+2l+1}$ are the boundary nodes on $[a-2\delta,a]\cup[b,b+2\delta]$ where we enforce $u_i=g(x_i)$ for $i=-2l,-2l+1,\cdots,0$ or $i=N+1,\cdots,N+2l+1$.

Let ψ be the standard hat function that is defined by

$$\psi(\zeta)=\begin{cases} 1-|\zeta|, & \zeta\in[-1,1], \\ 0, & \text{elsewhere.} \end{cases} \tag{3.5}$$

Let $\{\phi_i\}_{i=-2l}^{N+2l+1}$ be the set of hat functions defined by

$$\phi_i(x)=\psi\Big(\Big|\frac{x-x_i}{h}\Big|\Big). \tag{3.6}$$

Then the true solution u can be approximated by the trial function u_h that is expressed in the form of

$$u_h(x)=\sum_{j=1}^N u_j\phi_j(x)+\left(\sum_{j=-2l}^0+\sum_{j=N+1}^{N+2l+1}\right)g(x_j)\phi_j(x). \tag{3.7}$$

If we replace u by u_h in the second equation of (3.1) and enforce the resulting equation on the collocation points $\{x_i\}_{i=-l+1}^{N+l}$ to obtain the following set of equations

$$\begin{aligned}
&\theta_i := \theta^{\text{lin}}(x_i)\\
=&\frac{1}{m}\int_{x_i-\delta}^{x_i+\delta}\omega(|x'-x_i|)(x'-x_i)\left(\sum_{j=1}^{N}u_j\phi_j(x')+\left(\sum_{j=-2l}^{0}+\sum_{j=N+1}^{N+2l+1}\right)g(x_j)\phi_j(x')\right)dx'\\
=&\frac{1}{m}\sum_{j=1}^{N}u_j\int_{x_i-\delta}^{x_i+\delta}\omega(|x'-x_i|)(x'-x_i)\phi_j(x')dx'\\
&+\frac{1}{m}\left(\sum_{j=-2l}^{0}+\sum_{j=N+1}^{N+2l+1}\right)g(x_j)\int_{x_i-\delta}^{x_i+\delta}\omega(|x'-x_i|)(x'-x_i)\phi_j(x')dx'.
\end{aligned}\tag{3.8}$$

Let $u := [u_1,\cdots,u_N]^T$, $\Theta := [\theta_{-l+1},\cdots,\theta_{N+l}]^T$ and $G_{N+2l} := [\mathcal{G}_{-l+1},\cdots,\mathcal{G}_{N+l}]^T$ be N-dimensional and $(N+2l)$-dimensional vectors, respectively, and $A_{N+2l,N} = [a_{i,j}]_{-l+1\le i\le N+l,1\le j\le N}$ be an $(N+2l)$-by-N matrix where $\mathcal{G}_i$ and $a_{i,j}$ are defined by

$$\begin{aligned}
a_{i,j}=&\int_{x_i-\delta}^{x_i+\delta}\omega(|x'-x_i|)(x'-x_i)\phi_j(x')dx',\\
\mathcal{G}_i=&\left(\sum_{j=-2l}^{0}+\sum_{j=N+1}^{N+l+1}\right)g(x_j)\int_{x_i-\delta}^{x_i+\delta}\omega(|x'-x_i|)(x'-x_i)\phi_j(x')dx',\\
&-l+1\le i\le N+l;\quad 1\le j\le N.
\end{aligned}\tag{3.9}$$

Then the collocation scheme (3.8) can be written into a matrix form

$$\frac{1}{m}A_{N+2l,N}u+\frac{1}{m}G_{N+2l}=\Theta.\tag{3.10}$$

Next we turn to the first equation in the PD model (3.1). Let θ_h^{lin} be a piecewise linear approximation of θ^{lin} defined by

$$\theta_h^{\text{lin}}(x) := \sum_{j=-l+1}^{N+l}\theta_j\phi_j(x).\tag{3.11}$$

We then replace $\theta^{\text{lin}}(x)$ by $\theta_h^{\text{lin}}(x)$ and $u(x)$ by $u_h(x)$ in the first equation in the PD model (3.1), and enforce the resulting equation at the collocation points $\{x_i\}_{i=1}^{N}$ to arrive at the following set of equations for $1\le i\le N$

$$\begin{aligned}
-\int_{x_i-\delta}^{x_i+\delta}&\left[\alpha\omega(|x'-x_i|)(x'-x_i)\sum_{j=-l+1}^{N+l}\theta_j\phi_j(x')\right.\\
&\left.+\beta\omega(|x'-x_i|)\left(\sum_{j=1}^{N}u_j\phi_j(x')+\left(\sum_{j=-2l}^{0}+\sum_{j=N+1}^{N+2l+1}\right)g(x_j)\phi_j(x')-u_i\right)\right]dx'
\end{aligned}$$

$$
\begin{aligned}
&=-\alpha\sum_{j=-l+1}^{N+l}\theta_j\Big[\int_{x_i-\delta}^{x_i+\delta}\omega(|x'-x_i|)(x'-x_i)\phi_j(x')dx'\Big]\\
&\quad+\beta\sum_{j=1}^{N}u_j\Big[\int_{x_i-\delta}^{x_i+\delta}\omega(|x'-x_i|)\big(\delta_{i,j}-\phi_j(x')\big)dx'\Big]\\
&\quad-\beta\left(\sum_{j=-2l}^{0}+\sum_{j=N+1}^{N+2l+1}\right)g(x_j)\int_{x_i-\delta}^{x_i+\delta}\omega(|x'-x_i|)\phi_j(x')dx'\\
&=f_i:=f(x_i).
\end{aligned}\tag{3.12}
$$

Let $A_{N,N+2l}=[a_{i,j}]_{1\le i\le N,-l+1\le j\le N+l}$ be an N-by-$(N+2l)$ matrix, in which all the entries $a_{i,j}$ are still defined by (3.9) with $1\le i\le N$ and $-l+1\le j\le N+l$. Furthermore, we let $f:=[f_1,\cdots,f_N]^T$, $\hat{G}_N:=[\hat{\mathcal{G}}_1,\cdots,\hat{\mathcal{G}}_N]^T$ be two N-dimensional vectors, $B=[b_{i,j}]$ be an N-by-N matrix, where $\hat{\mathcal{G}}_i$ and $b_{i,j}$ are defined by

$$
\begin{aligned}
\hat{\mathcal{G}}_i&=\left(\sum_{j=-2l}^{0}+\sum_{j=N+1}^{N+l+1}\right)g(x_j)\int_{x_i-\delta}^{x_i+\delta}\omega(|x'-x_i|)\phi_j(x')dx',\\
b_{i,j}&=\int_{x_i-\delta}^{x_i+\delta}\omega(|x'-x_i|)\big(\delta_{i,j}-\phi_j(x')\big)dx',\quad 1\le i,j\le N.
\end{aligned}\tag{3.13}
$$

We can express the set of equations in (3.12) in the following matrix form

$$
-\alpha A_{N,N+2l}\Theta+\beta Bu-\beta\hat{G}_N=f.\tag{3.14}
$$

We insert (3.10) into (3.14) to obtain the following linear system for u

$$
\mathcal{A}u=f+\beta\hat{G}_N+\frac{\alpha}{m}A_{N,N+2l}G_{N+2l},\tag{3.15}
$$

where

$$
\mathcal{A}:=\beta B-\frac{\alpha}{m}A_{N,N+2l}A_{N+2l,N}.\tag{3.16}
$$

It is clear that the linear system (3.15) has a full stiffness matrix, which require $\mathcal{O}(N^2)$ of storage and $\mathcal{O}(N^3)$ of computations to invert. To develop a fast numerical method with efficient storage we will analyze the structure of the stiffness matrix $\mathcal{A}$ in the next section.

4 A fast Krylov subspace method

In this section we develop a fast Krylov subspace method for the linear system (3.15).

4.1 Structure of the stiffness matrix $\mathcal{A}$

The main result of this subsection is the following theorem

Theorem 4.1. *The stiffness matrix $\mathcal{A}$ in* (3.16) *can be expressed as*

$$\mathcal{A}:=\beta B+\frac{\alpha}{m}\left(A_{N+2l,N}\right)^T A_{N+2l,N}. \tag{4.1}$$

Here the matrix B is a symmetric and positive-definite, irreducible and diagonally dominant, banded Toeplitz matrix with half band-width $l+1$, the matrix $A_{N+2l,N}$ is a banded Toeplitz matrix with half band-width $l+1$, and so $\mathcal{A}$ is a symmetric and positive-definite matrix.

We will prove the theorem by first proving several lemmas. To do so, we begin by introducing the following expanded matrix

$$A_{N+2l,N+2l}:=[a_{i,j}]_{-l+1\le i,j\le N+l}, \tag{4.2}$$

with $a_{i,j}$ being defined in (3.9) for $-l+1\le i,j\le N+l$.

Lemma 4.1. *The matrices $A_{N+2l,N}$ and $A_{N,N+2l}$ are the submatrices of the matrix $A_{N+2l,N+2l}$ with the following embedding*

$$A_{N+2l,N+2l}=\left[A^{(L)}_{N+2l,l},A_{N+2l,N},A^{(R)}_{N+2l,l}\right]=\begin{bmatrix} A^{(T)}_{l,N+2l} \\ A_{N,N+2l} \\ A^{(B)}_{l,N+2l} \end{bmatrix}, \tag{4.3}$$

where the submatrices $A^{(L)}_{N+2l,l}$, $A^{(R)}_{N+2l,l}$, $A^{(T)}_{l,N+2l}$ and $A^{(B)}_{l,N+2l}$ are defined by

$$\begin{aligned} A^{(L)}_{N+2l,l}&:=[a_{i,j}]_{-l+1\le i\le N+l,-l+1\le j\le 0}, \\ A^{(R)}_{N+2l,l}&:=[a_{i,j}]_{-l+1\le i\le N+l,N+1\le j\le N+l}, \\ A^{(T)}_{l,N+2l}&:=[a_{i,j}]_{-l+1\le i\le 0,-l+1\le j\le N+l}, \\ A^{(B)}_{l,N+2l}&:=[a_{i,j}]_{N+1\le i\le N+l,-l+1\le j\le N+l}. \end{aligned} \tag{4.4}$$

In particular, the matrix product $A_{N,N+2l}A_{N+2l,N}$ is a submatrix of $\left(A_{N+2l,N+2l}\right)^2$.

Proof. (4.3) can be verified directly. In addition, we get

$$\begin{aligned} \left(A_{N+2l,N+2l}\right)^2=&\begin{bmatrix} A^{(T)}_{l,N+2l} \\ A_{N,N+2l} \\ A^{(B)}_{l,N+2l} \end{bmatrix}\left[A^{(L)}_{N+2l,l},A_{N+2l,N},A^{(R)}_{N+2l,l}\right] \\ &\begin{bmatrix} A^{(T)}_{l,N+2l}A^{(L)}_{N+2l,l} & A^{(T)}_{l,N+2l}A_{N+2l,N} & A^{(T)}_{l,N+2l}A^{(R)}_{N+2l,l} \\ A_{N,N+2l}A^{(L)}_{N+2l,l} & A_{N,N+2l}A_{N+2l,N} & A_{N,N+2l}A^{(R)}_{N+2l,l} \\ A^{(B)}_{l,N+2l}A^{(L)}_{N+2l,l} & A^{(B)}_{l,N+2l}A_{N+2l,N} & A^{(B)}_{l,N+2l}A^{(R)}_{N+2l,l} \end{bmatrix}, \end{aligned} \tag{4.5}$$

where the matrix block in the center is the matrix product of $A_{N,N+2l}$ and $A_{N+2l,N}$. □

Lemma 4.2. *The matrix $A_{N+2l,N+2l}$ is a skew symmetric, banded Toeplitz matrix with half band-width $l+1$.*

Proof. We prove the lemma in three steps.

1. We prove that $a_{i,j}=a_{i',j'}$ for any $-l+1\le i,j,i',j'\le N+l$ satisfying $i'-i=j'-j$. In this case, we have $x_{i'}-x_i=x_{j'}-x_j$. Then we let $\xi=x'-x_i$ and conclude from (3.6) and (3.9) that

$$\begin{aligned}
a_{i,j}&=\int_{x_i-\delta}^{x_i+\delta}\omega(|x'-x_i|)(x'-x_i)\phi_j(x')dx'\\
&=\int_{-\delta}^{\delta}\omega(|\xi|)\ \xi\ \phi_j(x_i+\xi)d\xi\\
&=\int_{-\delta}^{\delta}\omega(|\xi|)\ \xi\ \psi\Big(1-\Big|\frac{x_i+\xi-x_j}{h}\Big|\Big)d\xi\\
&=\int_{-\delta}^{\delta}\omega(|\xi|)\ \xi\ \psi\Big(1-\Big|\frac{x_{i'}+\xi-x_{j'}}{h}\Big|\Big)d\xi\\
&=\int_{-\delta}^{\delta}\omega(|\xi|)\ \xi\ \phi_{j'}(x_{i'}+\xi)d\xi=a_{i',j'}.
\end{aligned}\tag{4.6}$$

   We conclude that the matrix $A_{N+2l,N+2l}$ is a Toeplitz matrix.

2. We prove that $a_{i,j}=0$ for $|i-j|>l$. In fact, by (3.9) and the fact that the support of ϕ_j is (x_j-h,x_j+h), we clearly have $a_{i,j}=0$ if $|i-j|>l$. Therefore, the matrix $A_{N+2l,N+2l}$ is a band matrix with half band-width $l+1$.

3. We prove that $a_{i,i+k}=-a_{i,i-k}$, and then $a_{i,j}=-a_{j,i}$. As a matter of fact, by Step 2, if $k>l$, we have $a_{i,i+k}=-a_{i,i-k}=0$. If $k\le l$, we let $\xi=x'-x_i$ and $\eta=-\xi$ in (3.9) to get

$$\begin{aligned}
a_{i,i+k}&=\int_{x_i-\delta}^{x_i+\delta}\omega(|x'-x_i|)(x'-x_i)\phi_{i+k}(x')dx'\\
&=\int_{-\delta}^{\delta}\omega(|\xi|)\ \xi\ \phi_{i+k}(x_i+\xi)d\xi\\
&=\int_{-\delta}^{\delta}\omega(|\xi|)\ \xi\ \psi\Big(1-\Big|\frac{\xi}{h}-k\Big|\Big)d\xi\\
&=-\int_{-\delta}^{\delta}\omega(|\eta|)\ \eta\ \psi\Big(1-\Big|\frac{\eta}{h}+k\Big|\Big)d\eta\\
&=-\int_{-\delta}^{\delta}\omega(|\eta|)\ \eta\ \phi_{i-k}(x_i+\eta)d\eta=-a_{i,i-k}.
\end{aligned}\tag{4.7}$$

   Without loss of generality, we let $j=i+k$. As the matrix $A_{N+2l,N+2l}$ is Toeplitz by Step 1, we have

$$a_{i,j}=a_{i,i+k}=-a_{i,i-k}=-a_{i+k,i}=-a_{j,i},\tag{4.8}$$

   which shows that the matrix $A_{N+2l,N+2l}$ is skew symmetric.

The proof is completed. □

We conclude from the skew symmetric property of the matrix $A_{N+2l,N+2l}$ that

$$A_{N,N+2l}=-A_{N+2l,N}^T, \tag{4.9}$$

thus we insert this equation into (3.16) to obtain (4.1).

It remains to prove the properties of B, which will be carried out in the following lemma.

Lemma 4.3. *The matrix B is a symmetric and positive-definite, irreducible and diagonally dominant, banded Toeplitz matrix with half band-width $l+1$.*

Proof. We can prove that the matrix B is a symmetric and positive-definite, banded Toeplitz matrix with half band-width $l+1$ as we did in Lemma 4.2. It remains to show that B is irreducible weak diagonal dominant. From (3.13) we have

$$\begin{aligned} b_{i,i}&=\int_{x_i-\delta}^{x_i+\delta}\omega(|x'-x_i|)\big(1-\phi_i(x')\big)dx',\\ b_{i,j}&=-\int_{x_i-\delta}^{x_i+\delta}\omega(|x'-x_i|)\phi_j(x')dx',\quad j\neq i. \end{aligned} \tag{4.10}$$

It is clear that $b_{i,i}>0$ and $b_{i,j}\leq 0$ for $i\neq j$, so that

$$\begin{aligned} b_{i,i}-\sum_{j=1,j\neq i}^{N}|b_{i,j}|&=b_{i,i}+\sum_{j=1,j\neq i}^{N}b_{i,j}\\ &=\int_{x_i-\delta}^{x_i+\delta}\omega(|x'-x_i|)\big(1-\phi_i(x')\big)dx'\\ &\quad-\sum_{j=1,j\neq i}^{N}\int_{x_i-\delta}^{x_i+\delta}\omega(|x'-x_i|)\phi_j(x')dx'\\ &=\int_{x_i-\delta}^{x_i+\delta}\omega(|x'-x_i|)dx'-\sum_{j=1}^{N}\int_{x_i-\delta}^{x_i+\delta}\omega(|x'-x_i|)\phi_j(x')dx'\\ &=\int_{x_i-\delta}^{x_i+\delta}\omega(|x'-x_i|)\Big(1-\sum_{j=1}^{N}\phi_j(x')\Big)dx'. \end{aligned} \tag{4.11}$$

Note that the set of all the hat functions form a partition of unity. In particular, we have

$$\begin{aligned} &\sum_{|j-i|\leq l}\phi_j(x)=1,\quad x\in[x_1,x_N],\\ &\sum_{|j-i|\leq l}\phi_j(x)<1,\quad x\in[x_0,x_1]\bigcup[x_N,x_{N+1}]. \end{aligned} \tag{4.12}$$

We combine (4.11) and (4.12) to get

$$b_{i,i}-\sum_{j=1,j\neq i}^{N}|b_{i,j}|\begin{cases}>0, & 1\leq i\leq l \text{ or } N-l\leq i\leq N,\\ =0, & l<i<N-l.\end{cases} \tag{4.13}$$

We conclude from (4.10) that $b_{i,i+1}$ is nonzero for $i=1,2,\cdots,N-1$. As B is symmetric, we conclude that B is irreducible. □

We combine Lemmas 4.1, 4.2 and 4.3 to finish the proof of the theorem.

4.2 Efficient storage and fast matrix-vector multiplication

The main result of this subsection is the following theorem

Theorem 4.2. *The stiffness matrix $\mathcal{A}$ can be stored in $\mathcal{O}(N)$ memories. Furthermore, For any $v\in\mathbb{R}^N$, the matrix-vector multiplication $\mathcal{A}v$ can be carried out in $\mathcal{O}(N\log N)$ computations.*

Proof. By the decomposition (4.1) in Theorem 4.1, to store $\mathcal{A}$ we need only to store the matrices B and $A_{N+2l,N}$ which are banded Toeplitz matrices with half bandwidth $l+1$ again by Theorem 4.1. Hence, we need only to store $a_k:=a_{i,i+k}$ and $b_k:=b_{i,i+k}$ for $k=0,1,\cdots,l$, as B is a symmetric Toeplitz matrix with half bandwidth $l+1$ and $A_{N+2l,N+2l}$ is a skew-symmetric Toeplitz matrix with half bandwidth $l+1$ while $A_{N+2l,N}$ is a submatrix of $A_{N+2l,N+2l}$. Hence, we need to store $2(l+1)+3$ (for α, β and m) $=2l+5=\mathcal{O}(N)$ memories asymptotically.

We now turn to the computational complexity of $\mathcal{A}v$. By the decomposition (4.1) in Theorem 4.1, it suffices to show that both Bv and $\left(A_{N+2l,N}\right)^T A_{N+2l,N}v$ can be performed in $\mathcal{O}(N\log N)$ computations.

We begin by proving the former. Note that the Toeplitz matrix B can be embedded into a $2N\times 2N$ circulant matrix C [1]

$$C=\begin{bmatrix}B & \tilde{B}\\ \tilde{B} & B\end{bmatrix}, \tag{4.14}$$

where $\tilde{B}$ is an auxiliary matrix. It is known that the circulant matrix C can be decomposed in terms of discrete Fourier transform [1,3]

$$C=F_{2N}^{-1}\,\mathrm{diag}(F_{2N}c)\,F_{2N}, \tag{4.15}$$

where c is the first column vector of C and F_{2N} is the $2N\times 2N$ discrete Fourier transform matrix. It is well known that the matrix-vector multiplication $F_{2N}v_{2N}$ for any $v_{2N}\in\mathbb{R}^{2N}$ can be carried out in $\mathcal{O}(2N\log(2N))=\mathcal{O}(N\log N)$ operations via fast Fourier transform (FFT). (4.15) shows that Cv_{2N} can be evaluated in $\mathcal{O}(N\log N)$ operations.

On the other hand, for any $v\in\mathbb{R}^N$, we define $v_{2N}:=\left[v^T,0^T\right]^T\in\mathbb{R}^{2N}$ to get

$$Cv_{2N}=\begin{bmatrix}B & \tilde{B}\\ \tilde{B} & B\end{bmatrix}\begin{bmatrix}v\\ 0\end{bmatrix}=\begin{bmatrix}Bv\\ \tilde{B}v\end{bmatrix}. \tag{4.16}$$

Bu can be obtained by collecting the top half vector of Cv_{2N} and so can be evaluated in $\mathcal{O}(N\log N)$ computations.

We are now in a position to consider an efficient evaluation of $A_{N+2l,N}^T A_{N+2l,N}v$, which we can accomplish in the following steps:

1. Expand the vector $v\in\mathbb{R}^N$ into a vector $v_{N+2l}\in\mathbb{R}^{N+2l}$

$$v_{N+2l}:=\begin{bmatrix} 0 \\ -v \\ 0 \end{bmatrix}, \tag{4.17}$$

   where each 0 is an l-dimensional zero vector.

2. By (4.5) and (4.9) we get

$$\begin{aligned}
&\left(A_{N+2l,N+2l}\right)^2 v_{N+2l} \\
&=\begin{bmatrix} A_{l,N+2l}^{(T)}A_{N+2l,l}^{(L)} & A_{l,N+2l}^{(T)}A_{N+2l,N} & A_{l,N+2l}^{(T)}A_{N+2l,l}^{(R)} \\ A_{N,N+2l}A_{N+2l,l}^{(L)} & A_{N,N+2l}A_{N+2l,N} & A_{N,N+2l}A_{N+2l,l}^{(R)} \\ A_{l,N+2l}^{(B)}A_{N+2l,l}^{(L)} & A_{l,N+2l}^{(B)}A_{N+2l,N} & A_{l,N+2l}^{(B)}A_{N+2l,l}^{(R)} \end{bmatrix}\begin{bmatrix} 0 \\ -v \\ 0 \end{bmatrix} \\
&=\begin{bmatrix} -A_{l,N+2l}^{(T)}A_{N+2l,N}v \\ -A_{N,N+2l}A_{N+2l,N}v \\ -A_{l,N+2l}^{(B)}A_{N+2l,N}v \end{bmatrix}=\begin{bmatrix} -A_{l,N+2l}^{(T)}A_{N+2l,N}v \\ A_{N+2l,N}^T A_{N+2l,N}v \\ -A_{l,N+2l}^{(B)}A_{N+2l,N}v \end{bmatrix}.
\end{aligned} \tag{4.18}$$

3. In other words, to compute $A_{N+2l,N}^T A_{N+2l,N}v$ efficiently, we need only to compute $\left(A_{N+2l,N+2l}\right)^2 v_{N+2l}$. As $A_{N+2l,N+2l}$ is a banded Toeplitz matrix, we can calculate $A_{N+2l,N+2l}v_{N+2l}$ in $\mathcal{O}((N+2l)\log(N+2l))=\mathcal{O}(N\log N)$ operations via FFT as we did for Bv.

4. Then we compute $\left(A_{N+2l,N+2l}\right)^2 v_{N+2l}=A_{N+2l,N+2l}\left(A_{N+2l,N+2l}v_{N+2l}\right)$ in $\mathcal{O}(N\log N)$ operations via FFT.

5. Remove the first and last l entries of the vector $\left(A_{N+2l,N+2l}\right)^2 v_{N+2l}$ to obtain the vector $A_{N+2l,N}^T A_{N+2l,N}v$.

We thus finish the proof of the theorem. □

5 Numerical experiments

We conduct several numerical experiments in this Section to study the performance of the fast conjugate gradient (FCG) algorithm based on the fast matrix-vector multiplication

developed in Theorem 4.2. The Gaussian elimination (Gauss) method (3.15) is adopted as a comparison with our fast collocation scheme. For the FCG solver, we set the stopping criterion

$$\frac{\left\|r_{(k)}\right\|_2}{\left\|r_{(0)}\right\|_2} \leq 10^{-8}. \tag{5.1}$$

These methods were implemented using Intel Visual Fortran 2013 on a ThinkPad T440p Laptop.

5.1 Example 1: A static problem with manufactured solution

We consider a static state-based PD model (3.1), and the manufactured solution is set to be $u(x)=x^2(1-x)^2$ when $x\in(0,1)$ and $u(x)=0$ elsewhere. In addition, we let $(a,b)=(0,1)$, $\delta=1/16$, $\alpha=\beta=1$, the influence function $\omega(x)\equiv 1$, and the loading term $f(x)$ is calculated accordingly.

We present the CPU time consumption (CPU), l^2 errors, and convergence rates (Rate) of Gauss and FCG solvers in Table 1. For the FCG solver, we also present the number of iterations (Itr).

Table 1: Numerical results of Example 1.

	Gauss			FCG			
N	CPU	$\|u-u_h\|_{l^2}$	Rate	CPU	Itr	$\|u-u_h\|_{l^2}$	Rate
2^6	0.000s	3.88e-4	-	0.000s	14	3.88e-4	-
2^7	0.016s	9.81e-5	1.984	0.000s	14	9.81e-5	1.984
2^8	0.016s	2.46e-5	1.996	0.000s	14	2.46e-5	1.996
2^9	0.140s	6.16e-6	1.998	0.016s	14	6.16e-6	1.998
2^{10}	4.087s	1.54e-6	2.000	0.031s	14	1.54e-6	2.000
2^{11}	67.97s	3.86e-7	1.996	0.047s	14	3.86e-7	1.996
2^{12}	747.2s	9.64e-8	2.001	0.109s	14	9.64e-8	2.001
2^{13}	6769s	2.41e-8	2.000	0.328s	14	2.41e-8	2.000
2^{14}	-	-	-	0.796s	14	6.02e-9	2.001

We first note that the FCG and Gauss solvers of the collocation scheme (3.15) generate numerical solutions with almost identical accuracy and the same second-order convergence rate. We then note that the FCG solver has a significantly reduced CPU time over Gauss solver. For instance, as the mesh size h is refined from 2^{-12} to 2^{-13}, the CPU time consumed by Gauss increases from 747.2 seconds to around 6769 seconds. This coincides with the fact that each time h is refined by half, the number of grid points is doubled and the overall computational work for solving the linear system (3.15) by Gaussian increases 8 times asymptotically. Gaussian elimination at $h=2^{-14}$ would be on the order of several hours of CPU time. Hence we stopped the Gauss simulation at $h=2^{-13}$. Meanwhile,

we continue to push the FCG solver to finer mesh $h=2^{-14}$, which consumes only 0.796 second to perform the computation.

5.2 Example 2: A static problem without an analytical solution

We then manage to solve a static state-based PD model (3.1) with a different influence function $\omega(x)=\frac{1}{1+x^2}$. We choose the same data $(a,b)=(0,1)$, $\delta=1/16$, and $\alpha=\beta=1$ as in Example 1. The source term is chosen to be $f(x)=1$.

As the true solution to this example is not known, we use the numerical solution solved by Gauss on the fine mesh $N=2^{13}=8{,}192$ as a reference solution to compute the l^2 errors and the convergence rates. We present the CPU time consumption (CPU) of the two solvers, the l^2 errors and convergence rates (Rate), the number of iterations (Itr) of FCG solver in Table 2.

Table 2: Numerical results for Example 2.

	Gauss	FCG			
N	CPU	CPU	Itr	$\|u-u_h\|_{l^2}$	Rate
2^6	0.000s	0.000s	28	1.56e-4	-
2^7	0.000s	0.000s	34	3.63e-5	2.104
2^8	0.016s	0.016s	32	1.34e-5	1.438
2^9	0.140s	0.016s	27	7.50e-6	0.837
2^{10}	4.087s	0.047s	26	4.01e-6	0.903
2^{11}	67.53s	0.062s	24	1.85e-6	1.116
2^{12}	709.7s	0.203s	22	6.44e-7	1.522
2^{13}	6602s	0.437s	18	-	-
2^{14}	-	0.967s	17	-	-
2^{15}	-	2.075s	17	-	-
2^{16}	-	4.586s	16	-	-
2^{17}	-	12.50s	16	-	-
2^{18}	-	34.48s	15	-	-
2^{19}	-	75.74s	14	-	-
2^{20}	-	160.2s	13	-	-

Again we stopped the Gauss simulation at $h=2^{-13}$ and continue to push the FCG solver to finer meshes. We notice that in this example, the FCG solver succeeds to converge by acceptable CPU time consumption, for instance, using only 160.2 second for the finest mesh with $N=2^{20}=1{,}048{,}576$ nodes. We also observed that the Gauss and FCG solvers generate almost the same numerical solutions and the FCG solver shows great efficiency over Gauss, as in Example 1. We also find roughly the first-order convergence rate from Table 2. The drop of convergence rate to first order is most possibly due to the lack of regularity of the exact solution nearby the boundaries resulting from not carefully

chosen Dirichlet boundary condition. Further theoretical analysis and research will be considered in a future work.

5.3 Example 3: A dynamic state-based PD problem

Finally we consider to solve a dynamic state-based PD model (2.8), and we apply the Crank–Nicolson scheme for this non-static model.

Let $\rho=1$, the influence function $\omega(x)\equiv 1$, and the exact solution here is set to be

$$u(x,t)=x^2(1-x)^2\sin t.$$

Then in the domain $0<x<1$, $0<t\leq 1$, the initial condition can be calculated as

$$u(x,0)=0,\quad u_t(x,0)=x^2(1-x)^2,$$

and the source term has the form can be fixed accordingly. For this showing example, we let the number of time steps M be the same as the number of spacial meshes N, and $\delta=1/16$, $\alpha=\beta=1$, The CPU time consumption, l^2 errors, and the convergence rates of the Gauss and FCG solvers are listed in Table 3. It can be seen clearly that the FCG method maintains the same convergence rate as the traditional Gaussian method, but saves a great quantity of computation time. For example, when $N=M=10$, the Gauss method takes 3853 seconds, while the FCG requires only 4.531 seconds, which is 1/850 of the former. It is worth noting that this advantage will become more apparent as N and M increase.

Table 3: Numerical results of Example 3.

	Gauss			FCG		
$N=M$	CPU	$\|u-u_h\|_{l^2}$	Rate	CPU	$\|u-u_h\|_{l^2}$	Rate
2^5	0.000s	2.00e-4	-	0.000s	2.00e-4	-
2^6	0.016s	5.02e-5	1.994	0.016s	5.02e-5	1.994
2^7	0.234s	1.26e-5	1.996	0.063s	1.23e-5	2.027
2^8	3.391s	3.15e-6	1.997	0.234s	3.09e-6	1.997
2^9	71.16s	7.89e-7	1.999	0.969s	7.72e-7	1.998
2^{10}	3853 s	1.97e-7	1.999	4.531s	1.93e-7	1.999
2^{11}	-	-	-	21.53s	4.83e-8	2.000
2^{12}	-	-	-	100.5s	1.21e-8	2.000

6 Conclusion

In the present work, we develop a fast numerical algorithm for solving a state-based peridynamic (PD) model. Physically, this kind of model describes realistic cracks and

materials singularities precisely as many more number of degrees of freedom are used within the horizon. Conventionally people restrict the number of degrees of freedom to control the computational cost. Since it has significantly reduced the computational complexity and memory requirement, our fast algorithm allows the use of more degrees of freedom within the horizon. Numerically, the present fast algorithm can deal with much more nodes in the horizon efficiently, which are usually expensive in computation for other algorithms.

In principle we anticipate that the fast algorithm should work for more general numerical schemes. However, the development requires quite a bit of research work, which is under way. The extensions of the fast algorithm to more numerical schemes, the asymptotic compatibility, and more realistic peridynamic models in multi-dimensions will be considered in our future work.

Acknowledgments

We express our sincere thanks to the referees for their very helpful comments and suggestions, which greatly improved the quality of this paper. This work was supported in part by the OSD/ARO MURI Grant W911NF-15-1-0562, by the National Natural Science Foundation of China under Grants 11831010, 11571026, 91630207, and 11571115, by the National Science Foundation under Grant DMS-1620194, and by Taishan Scholars Program of Shandong Province of China.

References

[1] R.H. Chan and M.K. Ng, Conjugate gradient methods for Toeplitz systems, *SIAM Review*, 38 (1996), 427–482.
[2] X. Chen, M. Gunzburger, Continuous and discontinuous finite element methods for a peridynamics model of mechanics, *Comput. Methods Appl. Mech. Engrg.*, 200 (2011) 1237–1250.
[3] P.J. Davis, *Circulant Matrices*, Wiley-Intersciences, New York, 1979.
[4] K. Dayal and K. Bhattacharya, Kinetics of phase transformations in the peridynamic formulation of continuum mechanics, *J. Mech. Phys. Solids*, 54 (2006), 1811–1842.
[5] W. Deng, Finite Element Method for the Space and Time Fractional Fokker-Planck Equation, *SIAM J. Numer. Anal.*, 47 (2008), 204-226.
[6] E. Emmrich and O. Weckner, The peridynamic equation and its spatial discretisation, *Math. Model. Anal.*, 12 (2007) 17–27.
[7] J.T. Foster, S.A. Sillings, and W.W. Chen, State based peridynamic modeling of dynamic fracture, Proceedings of the SEM Annual Conference, June 1-4, 2009 Albuquerque, New Mexico, USA.
[8] W. Gerstle, N. Sau, and S. Silling, Peridynamic modeling of concrete structures, *Nucl. Eng. Des.*, 237 (2007), 1250–1258.
[9] M. Ghajari, L. Iannucci, and P. Curtis, A peridynamic material model for the analysis of dynamic crack propagation in orthotropic media, *Comput. Methods Appl. Mech. Engrg.*, 276 (2014) 431–452.

[10] Y. D. Ha and F. Bobaru, Characteristics of dynamic brittle fracture captured with peridynamics, *Eng. Fract. Mech.*, 78 (2011), 1156–1168.
[11] X. Lai, B. Ren, H. Fan, S. Li, C. T. Wu, R. A. Regueiro, and L. Liu, Peridynamics simulations of geomaterial fragmentation by impulse loads, *Int. J. Numer. Anal. Meth. Geomech*, 39 (2015), 1304–1330.
[12] J.A. Mitchell, A nonlocal, ordinary, state-based plasticity model for peridynamics, Sandia Report SAND 2011-3166, May 2011.
[13] E. Oterkus, E. Madenci, O.Weckner, S. Silling, P. Bogert, Combined finite element and peridynamic analyses for predicting failure in a stiffened composite curved panel with a central slot, *Compos. Struct.*, 94 (2012) 839–850.
[14] P. Seleson, Q. Du, and M. L. Parks, On the consistency between nearest-neighbor peridynamic discretizations and discretized classical elasticity models, *Computer Methods Appl. Mech. Engrg.*, 311 (2016), 698–722.
[15] P. Seleson and D. Littlewood, Convergence studies in meshfree peridynamic simulations, *Computers Math. Appl.*, 71 (2016), 2432–2448.
[16] P. Seleson and M.L. Parks, On the role of the infuence function in the peridynamic theory, *Int'l J. Multiscale Comput. Engrg.*, 9 (2011), 689–706.
[17] S. Silling, Reformulation of elasticity theory for discontinuous and long-range forces, *J. Mech. Phys. Solids*, 48 (2000) 175–209.
[18] S. Silling and E. Askari, A meshfree method based on the peridynamic model of solid mechanics, *Comput. Struct.*, 83 (2005) 1526–1535.
[19] S. Silling, M. Epton, O. Wecker, J. Xu, and E. Askari, Peridynamic states and constitutive modeling, *J. Elast.*, 88 (2007) 151–184.
[20] S. Silling, O. Weckner, E. Askari, and F. Bobaru, Crack nucleation in a peridynamic solid, *Int. J. Fract.*, 162 (2010), 219–227.
[21] S. Sun and V. Sundararaghavan, A peridynamic implementation of crystal plasticity, *Int'l J. Solids and Structures*, 51 (2014) 3350–3360.
[22] H. Tian, H. Wang, and W. Wang, An efficient collocation method for a non-local diffusion model, *Int'l J. Numer. Anal. Modeling*, 10 (2013), 815–825.
[23] X. Tian and Q. Du, Analysis and comparison of different approximations to nonlocal diffusion and linear peridynamic equations, *SIAM J. Numer. Anal.*, 51 (2013), 3458–3482.
[24] X. Tian and Q. Du, Asymptotically compatible schemes and applications to robust discretization of nonlocal models, *SIAM J. Numer. Anal.*, 52 (2014), 1641–1665.
[25] C. Wang and H. Wang, A fast collocation method for a variable-coefficient nonlocal diffusion model, *J. Comput. Phys.*, 330 (2017), 114–126.
[26] H. Wang, H. Tian, A fast Galerkin method with efficient matrix assembly and storage for a peridynamic model, *J. Comput. Phys.*, 231 (2012) 7730-7738.
[27] H. Wang, H. Tian, A fast and faithful collocation method with efficient matrix assembly for a two-dimensional nonlocal diffusion model, *Comput. Methods Appl. Mech. Eng.*, 273 (2014) 19-36.
[28] X. Zhang and H. Wang, A fast method for a steady-state bond-based peridynamic model, *Comput. Methods Appl. Mech. Engrg.*, 311 (2016), 280–303.