Noname manuscript No.
(will be inserted by the editor)

Coupled Peridynamics Least Square Minimization with Finite Element Method in 3D and implicit solutions by Message Passing Interface

Qibang Liu · X.J. Xin · Jeff Ma

Received: date / Accepted: date

Abstract In this work, we present a new framework to couple peridynamics least squares minimization with finite element method (PDLSM-FEM) for three-dimensional (3D) problems based on the weighted residual method (WRM). In PDLSM-FEM, the problem domain is divided into a PD domain and a FEM domain, and both domains are discretized into uniform or non-uniform elements. Within this framework, the coupling between the PD domain and the FEM domain is straightforward, and the formulation can be converted conveniently between PDLSM, FEM, and PDLSM-FEM. To directly and implicitly solve for displacements, we present an algorithm to assemble the global stiffness matrix in a compressed sparse row (CSR) format by the message passing interface (MPI) technique. The system equations are solved by the parallel direct sparse solver (PDSS) of the Intel Math Kernel Library (MKL). Three static examples are performed and the results show the proposed PDLSM-FEM can substantially reduce computational cost in terms of model efficiency and memory usage. Comparison between PDLSM-FEM results and FEM results using ANSYS demonstrates the good accuracy of the proposed PDLSM-FEM.

Keywords Peridynamics · Least square minimization · Finite element method · Implicit method · MPI

Q. Liu
Department of Mechanical and Nuclear Engineering, Kansas State University, Manhattan, KS 66506, USA
E-mail: qibangliu@ksu.edu

X.J. Xin
Department of Mechanical and Nuclear Engineering, Kansas State University, Manhattan, KS 66506, USA
E-mail: xin@ksu.edu

Jeff Ma
Parks College of Engineering, Aviation, and Technology, Saint Louis University, MO 63103, USA
E-mail: jeff.ma@slu.edu

1 Introduction

Many problems of fundamental importance in solid mechanics involve pre-existing and propagating discontinuities such as cracks. As classical continuum theory employs spatial derivatives in its formulation and assumes the material is continuous as it deforms, it is inherently difficult to predict discontinuous behaviors such as failure of materials and structures. In light of the inadequacies of the classical continuum theory, Peridynamics (PD), which is based on non-local interactions and employs spatial integrals, was first introduced by Silling [1]. PD theory is better suited for failure analysis of materials and structures because its governing equations are defined by integral equations and remain valid at fracture surfaces.

The first introduced PD theory is the bond-based PD (BBPD) [1] in which a single PD material parameter, the bond constants c, is employed to analyze isotropic materials, resulting in a restriction of a fixed Poisson's ratio of 1/4 [2]. Subsequently, Silling et al. [3] introduced the ordinary state-based PD (OSBPD) and non-ordinary state-based PD (NOSBPD) to remove the restriction. Both bond-based and state-based PD theories were derived by equating the classical strain energy at a material point to that of PD with a complete circle or sphere interaction domain [2]. These formulas in PD theory, however, are not valid for a point located near a surface because its interaction domain is not a complete circular or spherical [4]. Furthermore, most published work on PD to date employs a simple meshless method with a one-point quadrature for cells. Such one-point quadrature requires a volume correction procedure to improve integration accuracy [5]. To remove these two drawbacks, Madenci et al. [6] proposed a PD least squares minimization (PDLSM), and Liu et al [7] proposed a revised non-ordinary state-based PD (RNOSBPD). Both methods were derived by Taylor series expansion and the concept of non-local interactions of PD. To date, the PD theories have been not only applied to various structural discontinuous problems, such as damage and failure analysis of composite materials [8], metallic structures [9], ice craters [10], ceramics tiles [11], and glasses [12–14], but also extended to the analysis of heat conduction [15, 16], fluid flow [17, 18], electro-mechanical behavior [19], thermal and electrical transport processes of thermoelectric materials [20], and many others.

Compared with the finite element method (FEM), PD is computationally expensive. Thus, coupling PD with FEM is an appealing choice for taking advantage of these two methods. Kilic et al. [21] introduced a coupling method using overlapping regions in which both BBPD and FE equations are utilized. Agwai et al. [22] coupled BBPD with FEM using a sub-domain method, where the global modeling is performed using FEM while the PD theory is employed for the sub-modeling and failure prediction. A morphing strategy to couple BBPD with FEM based on energy equivalence was proposed and was employed for 1D and 2D problems by Lubineau et al. [23]. Liu et al. [24] introduced interface elements and coupling forces to bridge the BBPD and FEM. Some adaptive approaches which can transform FEM nodes into peridynamic

nodes for coupling BBPD or OSBPD with FEM were developed in [25–27]. Bie et al. [28] proposed an approach to couple OSBPD with node-based smoothed finite element which requires no transition region. Sun et al. [29] proposed a coupling approach based on a partial superposition of FEM and PD solutions for static and quasi-static problems. To couple OSBPD with FEM, the Arlequin method [30] is used in [31] to impose the mechanical compatibility between finite element and peridynamic sub-regions. Dong et al. [32] proposed a stability-enhanced PD element to couple NOSBPD with FEM, in which the stiffness matrix of the PD element is derived to establish the global stiffness matrix. Pagni et al. [33] proposed a technique to couple 3D peridynamics with 1D high-order finite elements using Lagrange multipliers. Shen et al. [34] introduced truss elements to bridge finite element (FE) sub-regions and PD sub-regions to couple PD with FEM. The authors [35] proposed a straightforward framework to couple PDLSM with FEM for 2D problems based on weighted residual method.

In general, PD is computationally expensive due to a large number of non-local interactions, much-refined mesh, and PD-based codes written in terms of dynamic equilibrium even for static problems. Typically, static problems in PD are solved by adopting an explicit adaptive dynamic relaxation (ADR) method [36] that determines steady-state solutions for a dynamic system by introducing fictitious mass and damping matrices. The explicit dynamic relaxation method is commonly preferred because it does not require large-matrix operations. Generally, the explicit dynamic relaxation method requires a large number of iterations for each time step to find a convergent solution. An alternative way would be to build the system equations $\mathbf{Ku} = \mathbf{F}$, then directly and implicitly solve the system equations for displacement solutions. Comparing to the explicit ADR method, the implicit method does not need a large number of iterations for solutions, and equilibrium is guaranteed if convergence is reached, thus it is more efficient for static problems. Sun et al. [37] used an implicit scheme to find the solution to a NOSBPD model of crystal plasticity with Newton-Raphson method. Breitenfeld et al. [38] implicitly implemented NOSBPD formulation for quasi-static linearly elastic solids problems. Zaccariotto et al. [39] implemented BBPD within an implicit code for static crack propagation phenomena based on the Newton-Raphson method. Prakash et al. [40] presented an algorithm that used coordinate and neighbor information to directly generate the sparse matrices based on BBPD formulas and solve the system equations for 2D static or quasi-static problems, which was shown much faster than the ADR method. Although the implicit method requires matrix inversion, the stiffness matrix of a FEM model is symmetric and sparse and its inversion is relatively easier and fast computationally. In comparison, mesh refinement in PD increases the size of the stiffness matrix $\mathbf{K}$ and a large number of non-local interactions leads to a denser $\mathbf{K}$, both of which significantly increase the computational cost to solve for $\mathbf{K}^{-1}$. Coupling PD with FEM makes it easier to employ the implicit method to solve fracture problems since FEM nodes interact only with nodes in the connecting elements, making the number of interactions much smaller than that of a full PD model.

Furthermore, FEM enables a much coarser mesh at regions far from the crack tips. Ni et al. [41] presented a static solution of crack propagation problems by implicitly solving a coupled BBPD with FEM. In some of the works for coupling PD with FEM quoted above, the implicit method was used to solve static problems for taking advantage of these two methods [23, 26, 27, 32–35].

In this work, we present a new framework to couple PDLSM with FEM (PDLSM-FEM) for 3D problems based on the weighted residual method (WRM), which was used to derive the weak form of PDLSM in [42] for 2D problems. In the paper [42], it discretized the domain by elements and took the center of the elements as the integral points to derive the weak form of pure PDLSM based on WRM and meshless method. In the current work (PDLSM-FEM), the problem domain is divided into a PD domain and a FEM domain, and both domains are discretized into uniform or non-uniform elements. The elements' nodes in the PD domain are treated as integral points, named as PD nodes, whose volumes are calculated from the volume of the connecting elements. To derive the governing equation, the PDLSM formulas and the meshless method are employed in the PD domain, while the FEM formulas and the mesh-based method are employed in the FEM domain. With this framework, the coupling between the PD domain and the FEM domain is straightforward, and the formulation can be converted conveniently between PDLSM, FEM, and PDLSM-FEM. To directly and implicitly solve for displacements, we present an algorithm to assemble the global stiffness matrix $\mathbf{K}$ in a compressed sparse row (CSR) format by message passing interface (MPI) technique. Within the CSR format, only non-zero coefficients of the stiffness matrix and their row and column indices are stored, rather than the full matrix. The system equations $\mathbf{Ku} = \mathbf{F}$ are solved by the parallel direct sparse solver (PDSS) of Intel Math Kernel Library (MKL). Three examples are performed to demonstrate the computational cost reduction and accuracy of the proposed PDLSM-FEM.

The remainder of this paper is outlined as follows. First, the theory of PDLSM is reviewed in Section 2. A new technique for coupling PDLSM with FEM for 3D problems based on the weighted residual method is developed and presented in Section 3. Next, the algorithm for assembling the global stiffness matrix in a CSR format is illustrated in Section 4. Three applications are performed to validate the computational efficiency and accuracy of PDLSM-FEM in Section 5. Finally, concluding remarks are summarized in Section 6.

2 Peridynamics least square minimization

In this section, the peridynamics least square minimization (PDLSM) first developed in [6] is briefly reviewed. PDLSM employs the concept of PD interactions and LSM in conjunction with the Taylor series expansion (TSE). Relevant formulas essential for understanding PDLSM can be found in [6]. As described in [6], the variation between $f(\mathbf{x}')$ and $f(\mathbf{x})$ can be approximated

based on TSE in a 3 dimensional space as

$$f(\mathbf{x}') = \sum_{n=0}^{N} B^n f(\mathbf{x}) + R(N, \mathbf{x}). \tag{1}$$

in which $R(N, \mathbf{x})$ is the remainder and B^n is defined as

$$B^n = \frac{1}{n!}\left(\xi_1 \frac{\partial}{\partial x_1} + \xi_2 \frac{\partial}{\partial x_2} + \xi_3 \frac{\partial}{\partial x_3}\right)^n, \tag{2}$$

and $\mathbf{x}' = \mathbf{x} + \boldsymbol{\xi}$ represents the neighbours of point $\mathbf{x}$ within its interaction domain H_x. The weighted error E_x from the approximation in Eq. (1) within the interaction domain can be evaluated based on least square minimization as

$$E_x = \int_{H_x} \omega(|\boldsymbol{\xi}|) R^2 dV_{x'} = \int_{H_x} \omega(|\boldsymbol{\xi}|) \left[f(\mathbf{x}') - \sum_{n=0}^{N} B^n f(\mathbf{x})\right]^2 dV_{x'}. \tag{3}$$

This error can be minimized by requiring its first variation to vanish as $\delta E_x = 0$. For 2nd-order TSE ($N = 2$), this requirement leads to the derivatives of $f(\mathbf{x})$ in a integral form as

$$\begin{aligned} &\left[\frac{\partial}{\partial x_1} \ \frac{\partial}{\partial x_2} \ \frac{\partial}{\partial x_3} \ \frac{\partial^2}{\partial x_1^2} \ \frac{\partial^2}{\partial x_2^2} \ \frac{\partial^2}{\partial x_3^2} \ \frac{\partial^2}{\partial x_1 x_2} \ \frac{\partial^2}{\partial x_2 x_3} \ \frac{\partial^2}{\partial x_3 x_1}\right]^T f(\mathbf{x}) = \\ &\int_{H_x} \omega(|\boldsymbol{\xi}|) \begin{bmatrix}\mathbf{g} \\ \mathbf{d}\end{bmatrix} (f(\mathbf{x}') - f(\mathbf{x}))\, dV'. \end{aligned} \tag{4}$$

The vector $\mathbf{g} = [g_1 \quad g_2 \quad g_3]^T$ and $\mathbf{d} = [d_1 \quad d_2 \quad d_3 \quad d_4 \quad d_5 \quad d_6]^T$ are defined as

$$\begin{bmatrix}\mathbf{g} \\ \mathbf{d}\end{bmatrix} = \mathbf{A}^{-1}\hat{\boldsymbol{\xi}}, \tag{5}$$

where

$$\hat{\boldsymbol{\xi}} = [\xi_1 \quad \xi_2 \quad \xi_3 \quad \xi_1^2 \quad \xi_2^2 \quad \xi_3^2 \quad \xi_1\xi_2 \quad \xi_2\xi_3 \quad \xi_1\xi_3]^T, \tag{6}$$

$$\mathbf{A} = \begin{bmatrix}\mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22}\end{bmatrix}, \tag{7}$$

$$\mathbf{A}_{11} = \int_{H_x} \omega(|\boldsymbol{\xi}|) \begin{bmatrix} \xi_1^2 & \xi_1\xi_2 & \xi_1\xi_3 \\ \xi_1\xi_2 & \xi_2^2 & \xi_2\xi_3 \\ \xi_1\xi_3 & \xi_2\xi_3 & \xi_3^2 \end{bmatrix} dV_{x'}, \tag{8}$$

$$\mathbf{A}_{12} = \int_{H_x} \omega(|\boldsymbol{\xi}|) \begin{bmatrix} \frac{\xi_1^3}{2} & \frac{\xi_1\xi_2^2}{2} & \frac{\xi_1\xi_3^2}{2} & \xi_1^2\xi_2 & \xi_1^2\xi_3 & \xi_1\xi_2\xi_3 \\ \frac{\xi_1^2\xi_2}{2} & \frac{\xi_2^3}{2} & \frac{\xi_2\xi_3^2}{2} & \xi_1\xi_2^2 & \xi_1\xi_2\xi_3 & \xi_2^2\xi_3 \\ \frac{\xi_1^2\xi_3}{2} & \frac{\xi_2^2\xi_3}{2} & \frac{\xi_3^3}{2} & \xi_1\xi_2\xi_3 & \xi_1\xi_3^2 & \xi_2\xi_3^2 \end{bmatrix} dV_{x'}, \tag{9}$$

$$\mathbf{A}_{21} = \int_{H_x} \omega(|\boldsymbol{\xi}|) \begin{bmatrix} \xi_1^3 & \xi_1^2\xi_2 & \xi_1^2\xi_3 \\ \xi_1\xi_2^2 & \xi_2^3 & \xi_2^2\xi_3 \\ \xi_1\xi_3^2 & \xi_2\xi_3^2 & \xi_3^3 \\ \xi_1^2\xi_2 & \xi_1\xi_2^2 & \xi_1\xi_2\xi_3 \\ \xi_1\xi_2\xi_3 & \xi_2^2\xi_3 & \xi_2\xi_3^2 \\ \xi_1^2\xi_3 & \xi_1\xi_2\xi_3 & \xi_1\xi_3^2 \end{bmatrix} dV_{x'}, \tag{10}$$

$$\mathbf{A}_{22} = \int_{H_x} \omega(|\boldsymbol{\xi}|) \begin{bmatrix} \frac{\xi_1^4}{2} & \frac{\xi_1^2\xi_2^2}{2} & \frac{\xi_1^2\xi_3^2}{2} & \xi_1^3\xi_2 & \xi_1^3\xi_3 & \xi_1^2\xi_2\xi_3 \\ \frac{\xi_1^2\xi_2^2}{2} & \frac{\xi_2^4}{2} & \frac{\xi_2^2\xi_3^2}{2} & \xi_1\xi_2^3 & \xi_1\xi_2^2\xi_3 & \xi_2^3\xi_3 \\ \frac{\xi_1^2\xi_3^2}{2} & \frac{\xi_2^2\xi_3^2}{2} & \frac{\xi_3^4}{2} & \xi_1\xi_2\xi_3^2 & \xi_1\xi_3^3 & \xi_2\xi_3^3 \\ \frac{\xi_1^3\xi_2}{2} & \frac{\xi_1\xi_2^3}{2} & \frac{\xi_1\xi_2\xi_3^2}{2} & \xi_1^2\xi_2^2 & \xi_1^2\xi_2\xi_3 & \xi_1\xi_2^2\xi_3 \\ \frac{\xi_1^2\xi_2\xi_3}{2} & \frac{\xi_2^3\xi_3}{2} & \frac{\xi_2\xi_3^3}{2} & \xi_1\xi_2^2\xi_3 & \xi_1\xi_2\xi_3^2 & \xi_2^2\xi_3^2 \\ \frac{\xi_1^3\xi_3}{2} & \frac{\xi_1\xi_2^2\xi_3}{2} & \frac{\xi_1\xi_3^3}{2} & \xi_1^2\xi_2\xi_3 & \xi_1^2\xi_3^2 & \xi_1\xi_2\xi_3^2 \end{bmatrix} dV_{x'}. \tag{11}$$

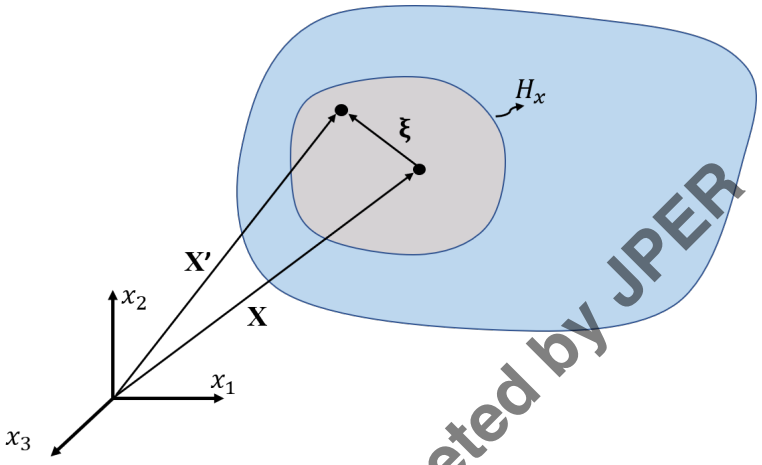


Fig. 1: Bond between points $\mathbf{x}'$ and $\mathbf{x}$ in the interaction domain H_x.

The differential operator of Eq. (4) can be used to derive the equation of motion in engineering problems by replacing the $f(\mathbf{x})$ as displacement components $u_i(\mathbf{x})$ $(i = 1,\ 2,\ 3)$, as shown in [42]. In PD, the equation of motion is given by

$$\rho\ddot{\mathbf{u}} = \mathbf{L}^{pd}(\mathbf{x}, t) + \mathbf{b}(\mathbf{x}, t), \tag{12}$$

where ρ is the mass density, $\ddot{\mathbf{u}}$ is the acceleration, $\mathbf{b}$ is the external force vector, and $\mathbf{L}$ is the internal force vector. The internal force at point $\mathbf{x}$ is evaluated by the integration of bond force between material points $\mathbf{x}$ and $\mathbf{x}'$ over the interaction domain H_x of the material point $\mathbf{x}$, as shown in Fig. 1. According to the classical continuum mechanics (CCM), the internal force vector can be expressed as

$$\mathbf{L}^{pd}(\mathbf{x}, t) = \nabla \cdot \boldsymbol{\sigma}^{pd}(\mathbf{x}, t), \tag{13}$$

in which $\boldsymbol{\sigma}^{pd}$ is Cauchy's stress. For a linear isotropic material, the stress tensor can be expressed as

$$\boldsymbol{\sigma}^{pd} = \lambda \mathbf{tr}(\nabla \mathbf{u}^{pd})\mathbf{I} + \mathcal{G}\left[\nabla \mathbf{u}^{pd} + (\nabla \mathbf{u}^{pd})^T\right], \tag{14}$$

with λ representing Lame's material constant and $\mathcal{G}$ representing shear modulus. Replacing the $f(\mathbf{x})$ of Eq. (4) by displacement components $u_i(\mathbf{x})$ leads to the non-local displacement gradient as

$$\nabla \mathbf{u}^{pd} = \int_{H_x} \omega(|\boldsymbol{\xi}|)\boldsymbol{\eta} \otimes \mathbf{g} dV_{x'}, \tag{15}$$

and the non-local internal force vector as

$$\mathbf{L}^{pd} = \int_{H_x} \omega(|\boldsymbol{\xi}|)\mathbf{G}\boldsymbol{\eta} dV_{x'}, \tag{16}$$

where $\boldsymbol{\xi} = \mathbf{x}' - \mathbf{x}$ and $\boldsymbol{\eta} = \mathbf{u}(\mathbf{x}') - \mathbf{u}(\mathbf{x})$ are the relative position and relative displacement of points $\mathbf{x}'$ and $\mathbf{x}$, respectively, and $\omega(|\boldsymbol{\xi}|)$ is the weight function. The matrix $\mathbf{G}$ in Eq. (16) is defined as

$$\mathbf{G} = \begin{bmatrix} (\lambda+\mathcal{G})d_1 + \mathcal{G}(d_1+d_2+d_3) & (\lambda+\mathcal{G})d_4 & (\lambda+\mathcal{G})d_6 \\ (\lambda+\mathcal{G})d_4 & (\lambda+\mathcal{G})d_2 + \mathcal{G}(d_1+d_2+d_3) & (\lambda+\mathcal{G})d_5 \\ (\lambda+\mathcal{G})d_6 & (\lambda+\mathcal{G})d_5 & (\lambda+\mathcal{G})d_3 + \mathcal{G}(d_1+d_2+d_3) \end{bmatrix}, \tag{17}$$

Note that the derivation of Eq. (15) and Eq. (16) does not require the interaction domain H_x to be a sphere, as many published PD models required. In fact, the interaction domain H_x can be of arbitrary shape. Thus, the surface effect [4] can be ignored and the volume correction [5] is not required.

3 A coupling framework of PDLSM and FEM for 3D problems

A new framework for coupling PDLSM with FEM is presented in this section. In this framework, the problem domain is divided into a PD domain and a FEM domain, both of which are discretized into uniform or non-uniform elements. Shared nodes on the interface between the PD domain and the FEM domain play a dual role, governed by the theory of PD on one side and by the theory of elasticity on the other. The weighted residual method is applied to PD equations of motion over the PD domain and elasticity equations of motion over the FEM domain, and the resultant governing equations combine PD and theory of elasticity in a unified way. When the FEM domain shrinks to zero, the formulation fully recovers the PDLSM and when the PD domain shrinks to zero, the formulation becomes FEM. When the PD domain and the FEM domain both exist, the formulation represents a coupling between PDLSM and FEM. We refer to the numerical model developed in this work as PDLSM-FEM.

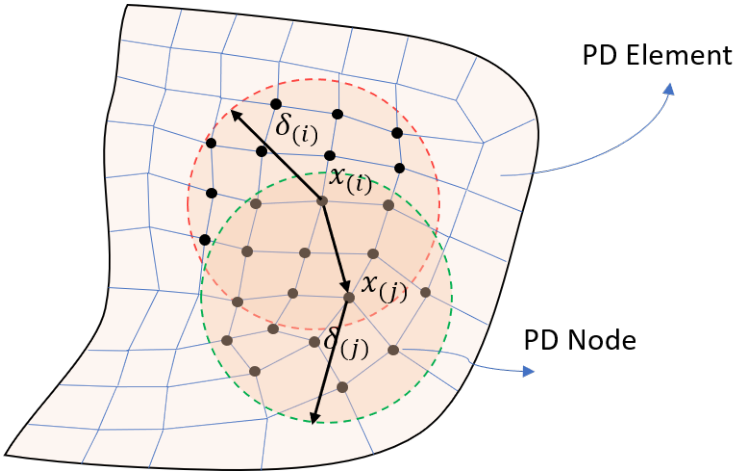


Fig. 2: 2D schematic PD domain discretization and the interactions between node $\mathbf{x}_{(i)}$ and its family members.

3.1 Discretization of PD domain

The PDLSM model of a domain can be discretized with elements (named as PD element herein) using commercial software, such as ANSYS due to its robust meshing ability, as shown in Fig. 2. For 3D cases, the PD element can be tetrahedron, hexahedron, pyramid, and prism, etc., which are compatible with the elements definition in commercial FEM codes. Each PD element is divided into N_n parts with equal volume V_e/N_n, in which N_n is the number of nodes of the PD element and V_e is the volume of the PD element. Then volume V_e/N_n will be ascribed to each node of the PD element. It is worth noting that the PD node may not locate at the center of its volume, especially for these nodes locating on the boundary of the domain. Although we have the PD element concept here, the numerical implementation of PDLSM in the PD domain is a meshless method, rather than an element-based method.

As shown in Fig. 2, the internal force vector at PD node $\mathbf{x}_{(i)}$ is achieved by including the effect of its interactions with other PD nodes (family members), $\mathbf{x}_{(j)}$, in the interaction domain. The interaction domain of node $\mathbf{x}_{(i)}$ depends on its family size $\delta_{(i)}$ and can be of arbitrary shape for PDLSM. The family size $\delta_{(i)}$ of node $\mathbf{x}_{(i)}$ is specified as $\delta_{(i)} = m\Delta_{(i)}$, with the m-ratio being of constant and the characteristic length $\Delta_{(i)}$ depends on volume $V_{(i)}$ occupied by PD node $\mathbf{x}_{(i)}$. $\Delta_{(i)}$ is defined as [43]

$$\Delta_{(i)} = \sqrt[3]{V_{(i)}} \tag{18}$$

Thus, each PD node has its own family size and family members. For non-uniform discretization, the rule to determine family members of node $\mathbf{x}_{(i)}$ is

suggested in [43] as

$$H_{\mathbf{x}_{(i)}} = \left\{ \mathbf{x}_{(j)} \in H_{\mathbf{x}_{(i)}} : \left|\mathbf{x}_{(j)} - \mathbf{x}_{(i)}\right| \leq \delta_{(i)} \cup \left|\mathbf{x}_{(j)} - \mathbf{x}_{(i)}\right| \leq \delta_{(j)} \right\}, \tag{19}$$

which ensures that the bond between nodes $\mathbf{x}_{(i)}$ and $\mathbf{x}_{(j)}$ is a paired interaction for either interaction domain. For node $\mathbf{x}_{(i)}$, there are $N_{(i)}$ family members within its interaction domain and the displacement set of these nodes is

$$\mathbf{u}^{(i)} = \left[u_{1(i)} \; u_{2(i)} \; u_{3(i)} \; u_{1(2)} \; u_{2(2)} \; u_{3(2)} \; \cdots \; u_{1(m)} \; u_{2(m)} \; u_{3(m)} \; \cdots \; u_{1(N_{(i)})} \; u_{2(N_{(i)})} \; u_{3(N_{(i)})} \right]^T, \tag{20}$$

here, the subscript (m) represents the m-th family member node $\mathbf{x}_{(m)}$ within the interaction domain of node $\mathbf{x}_{(i)}$ and its first member is itself $\mathbf{x}_{(i)}$. From Eq. (15), we obtain the discrete form of strain vector at node $\mathbf{x}_{(i)}$ as

$$\{\varepsilon\}_{(i)}^{pd} = [\varepsilon_x \quad \varepsilon_y \quad \varepsilon_z \quad \gamma_{xy} \quad \gamma_{yz} \quad \gamma_{zx}]^T = \mathbf{C}^{(i)}\mathbf{u}^{(i)}, \tag{21}$$

in which the matrix $\mathbf{C}$ is defined as below,

$$\mathbf{C}^{(i)} = \begin{bmatrix}
-\sum_{m=2}^{N_{(i)}} C_{1(im)} & 0 & 0 & C_{1(i2)} & 0 & 0 & \cdots \\
0 & -\sum_{m=2}^{N_{(i)}} C_{2(im)} & 0 & 0 & C_{2(i2)} & 0 & \cdots \\
0 & 0 & -\sum_{m=2}^{N_{(i)}} C_{3(im)} & 0 & 0 & C_{3(i2)} & \cdots \\
-\sum_{m=2}^{N_{(i)}} C_{2(im)} & -\sum_{m=2}^{N_{(i)}} C_{1(im)} & 0 & C_{2(i2)} & C_{1(i2)} & 0 & \cdots \\
0 & -\sum_{m=2}^{N_{(i)}} C_{3(im)} & -\sum_{m=2}^{N_{(i)}} C_{2(im)} & 0 & C_{3(i2)} & C_{2(i2)} & \cdots \\
-\sum_{m=2}^{N_{(i)}} C_{3(im)} & 0 & -\sum_{m=2}^{N_{(i)}} C_{1(im)} & C_{3(i2)} & 0 & C_{1(i2)} & \cdots
\end{bmatrix} \tag{22}$$

$$\begin{bmatrix}
\cdots & C_{1(im)} & 0 & 0 & \cdots & C_{1(iN_{(i)})} & 0 & 0 \\
\cdots & 0 & C_{2(im)} & 0 & \cdots & 0 & C_{2(iN_{(i)})} & 0 \\
\cdots & 0 & 0 & C_{3(im)} & \cdots & 0 & 0 & C_{3(iN_{(i)})} \\
\cdots & C_{2(im)} & C_{1(im)} & 0 & \cdots & C_{2(iN_{(i)})} & C_{1(iN_{(i)})} & 0 \\
\cdots & 0 & C_{3(im)} & C_{2(im)} & \cdots & 0 & C_{3(iN_{(i)})} & C_{2(iN_{(i)})} \\
\cdots & C_{3(im)} & 0 & C_{1(im)} & \cdots & C_{3(iN_{(i)})} & 0 & C_{1(iN_{(i)})}
\end{bmatrix},$$

where the subscript (im) represents the bond between node $\mathbf{x}_{(i)}$ and its m-th family member, and $C_{k(im)}(k = 1, 2, 3)$ are defined as

$$C_{k(im)} = \mu_{(im)}\omega_{(im)}g_{k(im)}V_{(m)}, \tag{23}$$

here $\mu_{(im)}$ and $\omega_{(im)}$ are the bond status parameter and the influence function, respectively, and $g_{k(im)}$ is defined in Eq. (5). From Eq. (21), the discrete form of stress vector at node $\mathbf{x}_{(i)}$ is

$$\{\sigma\}_{(i)}^{pd} = \mathbf{D}\mathbf{C}^{(i)}\mathbf{u}^{(i)}, \tag{24}$$

here, $\mathbf{D}$ is material stiffness matrix of elasticity material and is defined as

$$\mathbf{D} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} (1-\nu) & \nu & \nu & 0 & 0 & 0 \\ \nu & (1-\nu) & \nu & 0 & 0 & 0 \\ \nu & \nu & (1-\nu) & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix}, \tag{25}$$

Also, from Eq. (16), we obtain the discrete form of the internal force vector at node $\mathbf{x}_{(i)}$ as

$$\mathbf{L}_{(i)}^{pd} = \mathbf{H}^{(i)}\mathbf{u}^{(i)}, \tag{26}$$

$$\mathbf{H}^{(i)} = \left[-\sum_{m=2}^{N_{(i)}} \mu_{(im)}\omega_{(im)}\mathbf{G}_{(im)}V_{(m)} \;\; \mu_{(i2)}\omega_{(i2)}\mathbf{G}_{(i2)}V_{(2)} \;\cdots\; \mu_{(iN_{(i)})}\omega_{(iN_{(i)})}\mathbf{G}_{(iN_{(i)})}V_{(N_{(i)})} \right]. \tag{27}$$

The bond status parameter $\mu_{(im)}$ in Eq. (23) and Eq. (27) is equal to 1 for unbroken bonds or equal to 0 for broken bonds. The bond status can be determined by the failure criteria which are flexible and some of the published failure criteria can be found in [43–46].

3.2 Discretization of PDLSM-FEM domain

To couple PDLSM with FEM, the whole problem domain is divided into a FEM domain and a PD domain, and both domains are discretized into 3D elements, as shown in Fig. 3. For clarification, the elements in the PD domain are called PD elements and elements in the FEM domain are still called finite elements. Note that PD nodes on the PD-FEM interface are also nodes of the finite element, and the volumes of these PD nodes are calculated only from the PD elements. As describe in Section 2, PDLSM works for arbitrary interaction domain shape, and does not require surface correction. Thus, the FEM nodes can be either counted or not counted as family members of these PD nodes near the PD-FEM interface. In this work, the interaction domains of PD nodes are only determined from the PD domain but do not count FEM nodes as PD family members. That means, for these PD nodes near PD domain boundary, the interaction domain is truncated by PD domain boundary.

Consider a whole 3D discrete domain that contains N_g nodes, of which N^{pd} are PD nodes. For the i-th PD node $\mathbf{x}_{(i)}$, its displacement vector is $\mathbf{u}_{(i)} = [u_{1(i)} \quad u_{2(i)} \quad u_{3(i)}]^T$. The displacement set of all nodes (global displacement vector) is expressed as

$$\mathbf{u}_g = \begin{bmatrix} \mathbf{u}_u \\ \mathbf{u}_p \end{bmatrix}, \tag{28}$$

here $\mathbf{u}_u$ is the set of all unknown displacement components and $\mathbf{u}_p$ is the set of prescribed displacement components. The displacement vector of the PD node $\mathbf{x}_{(i)}$ can be expressed as

$$\mathbf{u}_{(i)} = \mathbf{M}_{(i)}\mathbf{u}_g, \tag{29}$$

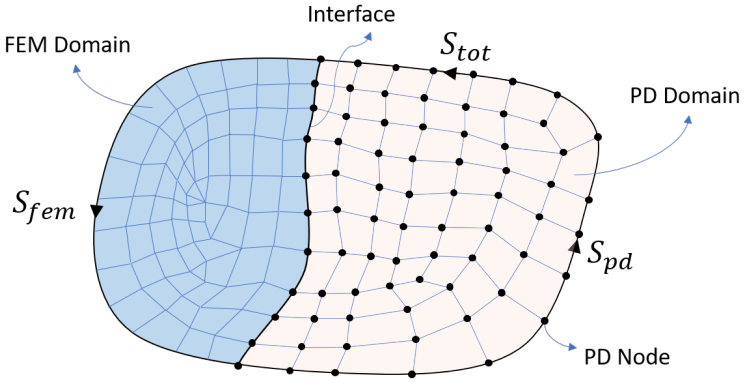


Fig. 3: 2D schematic of the coupling PD with FEM.

here, $\mathbf{M}_{(i)}$ is the mapping matrix from $\mathbf{u}_g$ to $\mathbf{u}_{(i)}$, with size $3 \times 3N_g$, and subscript (i) represents the PD node $\mathbf{x}_{(i)}$.

The displacement set of the family members of the PD node $\mathbf{x}_{(i)}$, $\mathbf{u}^{(i)}$, can be expressed by global displacement vector as

$$\mathbf{u}^{(i)} = \mathbf{M}^{(i)}\mathbf{u}_g, \tag{30}$$

here, superscript (i) represents the family memebers of the PD node $\mathbf{x}_{(i)}$, and $\mathbf{M}^{(i)}$ is the mapping matrix from $\mathbf{u}_g$ to $\mathbf{u}^{(i)}$, with size $3N_{(i)} \times 3N_g$ and it can be specified as follows: if the k-th component of $\mathbf{u}^{(i)}$ and the j-th component of $\mathbf{u}_g$ are the same, $M^{(i)}_{kj} = 1$, otherwise $M^{(i)}_{kj} = 0$. Thus, there is one and only one component equal to 1 in each row of $\mathbf{M}^{(i)}$. Also, $\mathbf{M}_{(i)}$ is determined by a similar way.

The mapping matrices are used for the matrix assembling procedure. For example, if $\mathbf{B} = \mathbf{M}^T_{(p)}\mathbf{A}\mathbf{M}^{(i)}$, the assembling procedure to obtain matrix $\mathbf{B}$ from $\mathbf{A}$ is as follows: we first initialize all components of $\mathbf{B}$ as zero, then go through all components of $\mathbf{M}_{(p)}$ and $\mathbf{M}^{(i)}$, if $M^{(i)}_{kj} = 1$ and $M_{mn(p)} = 1$, then $B_{nj} = A_{mk}$ $(n, j \in 1, 2, 3, \cdots 3N_g, m \in 1, 2$, and $k \in 1, 2, 3, \cdots 3N_{(i)})$.

Substituting Eq. (30) into Eq. (24) and Eq. (26), leads to the non-local stress vector and internal force vector by global displacement vector as below,

$$\{\sigma\}^{pd}_{(i)} = \mathbf{D}\mathbf{C}^{(i)}\mathbf{M}^{(i)}\mathbf{u}_g, \tag{31}$$

$$\mathbf{L}^{pd}_{(i)} = \mathbf{H}^{(i)}\mathbf{M}^{(i)}\mathbf{u}_g. \tag{32}$$

3.3 Governing Equations of PDLSM-FEM

For a coupled model, both the PD domain and the FEM domain exist. Applying the WRM on the equilibrium equation and the boundary condition over

the whole problem domain leads to

$$\int_{V_{tot}} \delta\mathbf{u}^T(\mathbf{L}+\mathbf{b}-\rho\ddot{\mathbf{u}})dV - \int_{S_{tot}} \delta\mathbf{u}^T(\boldsymbol{\sigma}\mathbf{n}-\mathbf{T})dS = 0, \tag{33}$$

where $\mathbf{T}$ is the external traction force on the surface boundary, $\mathbf{n}$ is the unit normal vector of the boundary, V_{tot} and S_{tot} are the volume and surface of the whole domain respectively, and $\delta(\bullet)$ represents a virtual value. The virtual work of inner volume forces and of the boundary surface forces are split into PD and FEM contributions as:

$$\int_{V_{tot}} \delta\mathbf{u}^T(\rho\ddot{\mathbf{u}}-\mathbf{L})dV = \int_{V_{pd}} \delta\mathbf{u}^T(\rho\ddot{\mathbf{u}}-\mathbf{L}^{pd})dV + \int_{V_{fem}} \delta\mathbf{u}^T(\rho\ddot{\mathbf{u}}-\mathbf{L}^{fem})dV, \tag{34}$$

and

$$\int_{S_{tot}} \delta\mathbf{u}^T\boldsymbol{\sigma}\mathbf{n}dS = \int_{S_{pd}} \delta\mathbf{u}^T\boldsymbol{\sigma}^{pd}\mathbf{n}dS + \int_{S_{fem}} \delta\mathbf{u}^T\boldsymbol{\sigma}^{fem}\mathbf{n}dS, \tag{35}$$

where V_{fem} and V_{pd} represent the volume of the FEM domain and PD domain respectively, and S_{pd} and S_{fem} represent the boundary of the PD domain and FEM domain respectively. For the PD domain V_{pd}, the internal force vector, $\mathbf{L}^{pd}$, takes on the integral form of Eq. (32) which is based on meshless method. While for the FEM domain V_{fem}, the Gauss integration and element-based method are used, and the internal force vector $\mathbf{L}^{fem}$, takes on the differential form of

$$\mathbf{L}^{fem} = \nabla\cdot\boldsymbol{\sigma}^{fem}. \tag{36}$$

For the surface of the PD domain, S_{pd}, the stress in the item of $\delta\mathbf{u}^T\boldsymbol{\sigma}^{pd}\mathbf{n}$ is derived based on the integral form of Eq. (31). For the surface of the FEM domain, S_{fem}, the item of $\delta\mathbf{u}^T\boldsymbol{\sigma}^{fem}\mathbf{n}$ is canceled by applying Gauss theorem as

$$\begin{aligned}\int_{V_{fem}} \delta\mathbf{u}^T\mathbf{L}^{fem}dV &= \int_{V_{fem}} \left\{\nabla\cdot\left(\delta\mathbf{u}^T\boldsymbol{\sigma}^{fem}\right) - \nabla\delta\mathbf{u}:\boldsymbol{\sigma}^{fem}\right\}dV \\ &= \int_{S_{fem}} \delta\mathbf{u}^T\boldsymbol{\sigma}^{fem}\mathbf{n}dS - \int_{V_{fem}} \nabla\delta\mathbf{u}:\boldsymbol{\sigma}^{fem}dV.\end{aligned} \tag{37}$$

Therefore, Eq. (33) becomes to

$$\begin{aligned}&\int_{V_{fem}} \left[\delta\mathbf{u}^T\rho\ddot{\mathbf{u}} + \delta(\{\varepsilon^{fem}\}^T)\{\sigma^{fem}\}\right]dV + \int_{V_{pd}} \delta\mathbf{u}^T(\rho\ddot{\mathbf{u}}-\mathbf{L}^{pd})dV + \int_{S_{pd}} \delta\mathbf{u}^T\boldsymbol{\sigma}^{pd}\mathbf{n}dS = \\ &\int_{S_{tot}} \delta\mathbf{u}^T\mathbf{T}dS + \int_{V_{fem}} \delta\mathbf{u}^T\mathbf{b}dV + \int_{V_{pd}} \delta\mathbf{u}^T\mathbf{b}dV.\end{aligned} \tag{38}$$

The virtual internal work by the internal force of the PD domain, δU_B^{pd}, is expressed as

$$\delta U_B^{pd} = -\int_{V_{pd}} \delta\mathbf{u}^T\mathbf{L}^{pd}dV = -\sum_{i=1}^{N^{pd}} \delta\mathbf{u}_{(i)}^T\mathbf{L}_{(i)}^{pd}V_{(i)} \tag{39}$$

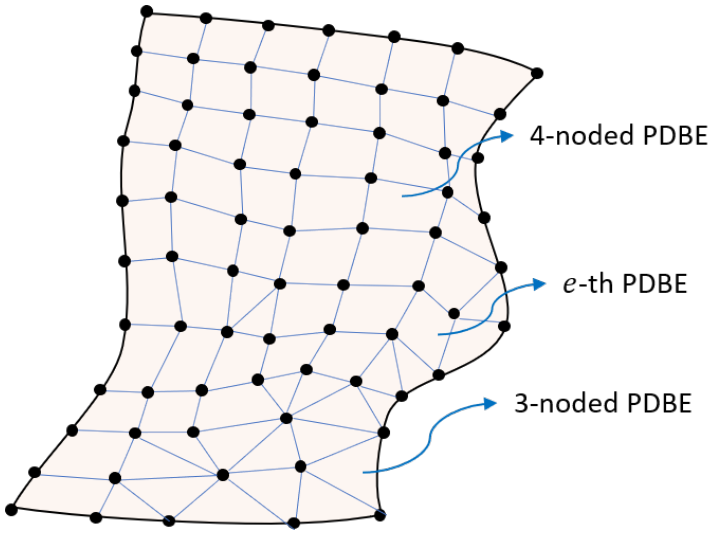


Fig. 4: PD boundary element.

Substituting Eq. (29) and Eq. (32) into Eq. (39) leads to

$$\delta U_B^{pd} = \delta \mathbf{u}_g^T \mathbf{K}_b^{pd} \mathbf{u}_g, \tag{40}$$

where $\mathbf{K}_b^{pd} = -\sum_{i=1}^{N^{pd}} \mathbf{M}_{(i)}^T \mathbf{H}_{(i)} \mathbf{M}^{(i)} V_{(i)}$. The boundary of the PD domain S_{pd} is discretized as 4-noded or 3-noded boundary elements, named as PD boundary element (PDBE), by the discretization of the PD domain, as shown in Fig. 4. The virtual internal work associated with PD domain boundary, δU_S^{pd}, is evaluated as

$$\delta U_S^{pd} = \int_{S_{pd}} \delta \mathbf{u}^T \boldsymbol{\sigma}^{pd} \mathbf{n} dS = \sum_{e=1}^{N_S^{pd}} \left(\delta U_{S_e}^{pd} \right), \tag{41}$$

in which N_S^{pd} is the number of PDBEs, and $\delta U_{S_e}^{pd}$ is the virtual internal work associated with the e-th PDBE and is defined as

$$\delta U_{S_e}^{pd} = \int_{S_e} \delta \mathbf{u}^T \boldsymbol{\sigma}^{pd} \mathbf{n} dS \tag{42}$$

The stress on the S_e is approximately expressed as

$$\boldsymbol{\sigma}^{pd} = \sum_{i=1} \mathcal{N}_i \boldsymbol{\sigma}_{(e_i)}^{pd}, \tag{43}$$

the displacement of S_e is approximately expressed as

$$\mathbf{u}^{pd} = \sum_{i=1} \mathcal{N}_i \mathbf{u}_{(e_i)}^{pd} = \boldsymbol{\mathcal{N}} \mathbf{u}_e, \tag{44}$$

and the coordinate of S_e is expressed as

$$\mathbf{x} = \sum_{i=1} \mathcal{N}_i \mathbf{x}_{(e_i)}, \tag{45}$$

in which e_i represents the i-th node of the e-th PDBE, and $\mathcal{N}_i$ represent the shape functions, which are defined as follows. For 4-noded PDBE,

$$\begin{cases} \mathcal{N}_1 = 0.25(1-p)(1-q) \\ \mathcal{N}_2 = 0.25(1+p)(1-q) \\ \mathcal{N}_3 = 0.25(1+p)(1+q) \\ \mathcal{N}_4 = 0.25(1-p)(1+q) \end{cases} \tag{46}$$

where $p,\ q \in [-1, 1]$, are local coordinates. The matrix $\boldsymbol{\mathcal{N}}$ is defined as

$$\boldsymbol{\mathcal{N}} = [\mathcal{N}_1\mathbf{I} \quad \mathcal{N}_2\mathbf{I} \quad \mathcal{N}_3\mathbf{I} \quad \mathcal{N}_4\mathbf{I}] \tag{47}$$

in which $\mathbf{I}$ is a identical matrix with size 3×3. $\mathbf{u}_e$ is the nodal displacement set of the e-th PDBE and is defined as

$$\mathbf{u}_e = \begin{bmatrix} u_{1(e_1)} & u_{2(e_1)} & u_{3(e_1)} \cdots & u_{1(e_4)} & u_{2(e_4)} & u_{3(e_4)} \end{bmatrix}^T \tag{48}$$

$\mathbf{u}_e$ is also expressed by the global displacement vector as

$$\mathbf{u}_e = \mathbf{M}_e \mathbf{u}_g \tag{49}$$

where $\mathbf{M}_e$ is the mapping matrix from $\mathbf{u}_g$ to $\mathbf{u}_e$ and can be obtained as described previously. The traction applied on the infinitesimal area of the boundary of 4-noded PDBE can be expressed by the local coordinates as

$$\boldsymbol{\sigma}^{pd}\mathbf{n}dS = \boldsymbol{\sigma}^{pd}\left(\frac{\partial \mathbf{x}}{\partial p} \times \frac{\partial \mathbf{x}}{\partial q}\right) dpdq = \overline{\mathbf{N}}\{\sigma\}^{pd} dpdq \tag{50}$$

in which $\overline{\mathbf{N}}$ is defined as

$$\overline{\mathbf{N}} = \begin{bmatrix} \overline{N}_1 & 0 & 0 & \overline{N}_2 & 0 & \overline{N}_3 \\ 0 & \overline{N}_2 & 0 & \overline{N}_1 & \overline{N}_3 & 0 \\ 0 & 0 & \overline{N}_3 & 0 & \overline{N}_2 & \overline{N}_1 \end{bmatrix}. \tag{51}$$

here $\overline{N}_1$, $\overline{N}_2$ and $\overline{N}_3$ are defined as

$$\begin{cases} \overline{N}_1 = \frac{\partial x_2}{\partial p}\frac{\partial x_3}{\partial q} - \frac{\partial x_3}{\partial p}\frac{\partial x_2}{\partial q} \\ \overline{N}_2 = \frac{\partial x_3}{\partial p}\frac{\partial x_1}{\partial q} - \frac{\partial x_1}{\partial p}\frac{\partial x_3}{\partial q} \\ \overline{N}_3 = \frac{\partial x_1}{\partial p}\frac{\partial x_2}{\partial q} - \frac{\partial x_2}{\partial p}\frac{\partial x_1}{\partial q} \end{cases} \tag{52}$$

For 4-noded PDBE, $\delta U_{S_e}^{pd}$ can be evaluated by Gauss integration as below,

$$\begin{aligned}
\delta U_{S_e}^{pd} &= \int_{-1}^{1}\int_{-1}^{1} \delta \mathbf{u}^T \overline{\mathbf{N}} \{\sigma\}^{pd} dpdq \\
&= \int_{-1}^{1}\int_{-1}^{1} \delta \mathbf{u}_e^T \boldsymbol{\mathcal{N}}^T \overline{\mathbf{N}} \sum_{i=1}^{4} \mathcal{N}_{e_i} \{\sigma\}_{(e_i)}^{pd} dpdq \\
&= \sum_p \sum_q \sum_{i=1}^{4} \delta \mathbf{u}_e^T \boldsymbol{\mathcal{N}}^T \overline{\mathbf{N}} \mathcal{N}_{e_i} \{\sigma\}_{(e_i)}^{pd} w_p w_q \\
&= \sum_p \sum_q \sum_{i=1}^{4} \delta \mathbf{u}_g^T \mathbf{M}_e^T \boldsymbol{\mathcal{N}}^T \overline{\mathbf{N}} \mathcal{N}_{e_i} \mathbf{D}\mathbf{C}^{(e_i)} \mathbf{M}^{(e_i)} \mathbf{u}_g w_p w_q \\
&= \delta \mathbf{u}_g^T \left(\sum_p \sum_q \sum_{i=1}^{4} \mathbf{M}_e^T w_p w_q \boldsymbol{\mathcal{N}}^T \overline{\mathbf{N}} \mathcal{N}_i \mathbf{D}\mathbf{C}^{(e_i)} \mathbf{M}^{(e_i)} \right) \mathbf{u}_g \\
&= \delta \mathbf{u}_g^T \mathbf{K}_{S_e}^{pd} \mathbf{u}_g
\end{aligned} \tag{53}$$

where w_p and w_q are the weights of Gauss integration, and $\mathbf{K}_{S_e}^{pd}$ is defined as

$$\mathbf{K}_{S_e}^{pd} = \sum_p \sum_q \sum_{i=1}^{4} \mathbf{M}_e^T w_p w_q \boldsymbol{\mathcal{N}}^T \overline{\mathbf{N}} \mathcal{N}_i \mathbf{D}\mathbf{C}^{(e_i)} \mathbf{M}^{(e_i)} \tag{54}$$

For 3-noded PDBE, a similar way can be employed to evaluate $\mathbf{K}_{S_e}^{pd}$ as

$$\mathbf{K}_{S_e}^{pd} = \mathbf{M}_e^T \sum_{i=1}^{3} \left[\left(\int_{S_e} \boldsymbol{\mathcal{N}}^T \mathcal{N}_i dS \right) \overline{\mathbf{N}} \mathbf{D}\mathbf{C}^{(e_i)} \mathbf{M}^{(e_i)} \right] \tag{55}$$

in which $\boldsymbol{\mathcal{N}}$ is defined as

$$\boldsymbol{\mathcal{N}} = [\mathcal{N}_1 \mathbf{I} \quad \mathcal{N}_2 \mathbf{I} \quad \mathcal{N}_3 \mathbf{I}] \tag{56}$$

where $\mathbf{I}$ is an identical matrix with size 3×3, $\mathcal{N}_i$ is the area coordinates, and the matrix $\overline{\mathbf{N}}$ is defined as Eq. (51) , in which $\overline{N}_i$ $(i = 1, 2, 3)$, are replaced by the unit normal vector components of the PDBE S_e. The integration of $\int_{S_e} \boldsymbol{\mathcal{N}}^T \mathcal{N}_i dS$ can be evaluated by the following exact integration expression as

$$\int_{S_e} \mathcal{N}_1^a \mathcal{N}_2^b \mathcal{N}_3^c dS = \frac{a!\ b!\ c!}{(a+b+c+2)!} 2A_{S_e} \tag{57}$$

here A_{S_e} is the area of the triangle.

With the representation of $\mathbf{K}_{S_e}^{pd}$, the virtual internal work by the internal force of the PD domain is evaluated as

$$\delta U_S^{pd} = \delta \mathbf{u}_g^T \mathbf{K}_s^{pd} \mathbf{u}_g, \tag{58}$$

where $\mathbf{K}_s^{pd}$ is defined as

$$\mathbf{K}_s^{pd} = \sum_{e=1}^{N_S^{pd}} \mathbf{K}_{S_e}^{pd} \tag{59}$$

in which N_S^{pd} is number of PDBEs. The virtual work by the inertia force in the PD domain, δU_I^{pd}, can be evaluated as

$$\delta U_I^{pd} = \int_{V_{pd}} \delta \mathbf{u}^T \rho \ddot{\mathbf{u}} dV = \delta \mathbf{u}_g^T \mathbf{M}^{pd} \ddot{\mathbf{u}}_g, \tag{60}$$

where $\mathbf{M}^{pd} = \sum_{i=1}^{N^{pd}} \rho \mathbf{M}_{(i)}^T \mathbf{M}_{(i)} V_{(i)}$. Similarly, the virtual work by the external body force $\mathbf{b}$ in the PD domain, δW_B^{pd}, can be evaluated as

$$\delta W_B^{pd} = \int_{V_{pd}} \delta \mathbf{u}^T \mathbf{b} dV = \delta \mathbf{u}_g^T \mathbf{F}_b^{pd}, \tag{61}$$

where $\mathbf{F}_b^{pd} = \sum_{i=1}^{N^{pd}} \mathbf{M}_{(i)}^T \mathbf{b}_{(i)} V_{(i)}$.

The virtual strain energy of the FEM domain, δU^{fem}, can be evaluated by the FEM as

$$\delta U^{fem} = \int_{V_{fem}} \delta(\{\varepsilon^{fem}\}^T)\{\sigma^{fem}\} dV = \delta \mathbf{u}_g^T \mathbf{K}^{fem} \mathbf{u}_g. \tag{62}$$

Similarly, the virtual work by the inertial force in the FEM domain, δU_I^{fem}, the virtual work by the external body force in the FEM domain, δW_B^{fem}, and the virtual work by the external traction force of the whole domain, δW_S, can be evaluated by FEM as follows:

$$\delta U_I^{fem} = \int_{V_{fem}} \delta \mathbf{u}^T \rho \ddot{\mathbf{u}} dV = \delta \mathbf{u}_g^T \mathbf{M}^{fem} \ddot{\mathbf{u}}_g, \tag{63}$$

$$\delta W_B^{fem} = \int_{V_{fem}} \delta \mathbf{u}^T \mathbf{b} dV = \delta \mathbf{u}_g^T \mathbf{F}_b^{fem}, \tag{64}$$

$$\delta W_S = \int_{S_{tot}} \delta \mathbf{u}^T \mathbf{T} dS = \delta \mathbf{u}_g^T \mathbf{F}_s. \tag{65}$$

As the FEM is a mature technology, the details to obtain the $\mathbf{K}^{fem}$, $\mathbf{M}^{fem}$, $\mathbf{F}_b^{fem}$ and $\mathbf{F}_s$ are not shown here for conciseness. Substituting Eqs. (40), (58), (60), (61), (62), (63), (64) and (65) into Eq. (38) leads to the governing equation of this coupling model,

$$\mathbf{M}\ddot{\mathbf{u}}_g + \mathbf{K}\mathbf{u}_g = \mathbf{F}, \tag{66}$$

where the global equivalent nodal mass $\mathbf{M}$ is defined as

$$\mathbf{M} = \mathbf{M}^{pd} + \mathbf{M}^{fem}, \tag{67}$$

the global stiffness matrix $\mathbf{K}$ is defined as

$$\mathbf{K} = \mathbf{K}_b^{pd} + \mathbf{K}_s^{pd} + \mathbf{K}^{fem} \tag{68}$$

and the global equivalent nodal force $\mathbf{F}$ is defined as

$$\mathbf{F} = \mathbf{F}_b^{pd} + \mathbf{F}_b^{fem} + \mathbf{F}_s, \tag{69}$$

here, $\mathbf{M}$, $\mathbf{K}$ and $\mathbf{F}$ can be obtained by the assembling procedure as described previously. Note that $\mathbf{M}^{pd}$ is a lumped diagonal mass matrix, while $\mathbf{M}^{fem}$ can be either lumped diagonal or consistent mass matrix. For explicit dynamic solutions, using lumped diagonal mass matrix $\mathbf{M}^{fem}$ may be more efficient since it is easier to solve for $\mathbf{M}^{-1}$. When $V_{pd} = 0$, Eq. (66) represents a pure FEM model and when $V_{fem} = 0$, it represents a pure PD model. Thus, the formulation converts easily between PDLSM, FEM, and PDLSM-FEM. Besides, Eq. (66) can be decomposed as

$$\begin{bmatrix} \mathbf{M}_{uu} & \mathbf{M}_{up} \\ \mathbf{M}_{pu} & \mathbf{M}_{pp} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{u}}_u \\ \ddot{\mathbf{u}}_p \end{bmatrix} + \begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{up} \\ \mathbf{K}_{pu} & \mathbf{K}_{pp} \end{bmatrix} \begin{bmatrix} \mathbf{u}_u \\ \mathbf{u}_p \end{bmatrix} = \begin{bmatrix} \mathbf{F}_u \\ \mathbf{F}_p \end{bmatrix}. \tag{70}$$

As $\mathbf{u}_p$ is the prescribed displacement constraint, Eq. (70) reduces to

$$\mathbf{M}_{uu}\ddot{\mathbf{u}}_u + \mathbf{K}_{uu}\mathbf{u}_u = \mathbf{F}_u - \mathbf{M}_{up}\ddot{\mathbf{u}}_p - \mathbf{K}_{up}\mathbf{u}_p \tag{71}$$

For static problem, Eq. (70) reduces to

$$\mathbf{K}_{uu}\mathbf{u}_u = \mathbf{F}_u - \mathbf{K}_{up}\mathbf{u}_p \tag{72}$$

4 Assembling stiffness matrix

In this work, the large linear systems of equations are solved by the Parallel Direct Sparse Solver (PDSS) of Intel Math Kernel Library (MKL). PDSS is for cluster interface and is based on message passing interface (MPI) technique. The large sparse matrix is stored in a compressed sparse row (CSR) format which is one of inputs of PDSS. In this section, we present how to assemble the global stiffness matrix in a CSR format by MPI technique. Note that zero-based indexing is employed for arrays in this coding and the program is written in C++ environment.

4.1 Initialize data model

As described in Section 3, the problem domain is divided into a PD domain and a FEM domain, both of which can be discretized as elements by commercial finite element codes, such as ANSYS. Then the nodes' coordinates, elements' connectivity, essential and natural boundary conditions are read by the program from the input file. The volume of each PD node is calculated from the volume of PD elements. The family members of the PD node $\mathbf{x}_{(i)}$ are

stored in a container of `std::vector`, in which the first member is itself $\mathbf{x}_{(i)}$. Each node has 3 degrees of freedom (DOF) and the total number of DOFs is $N_t = 3N_g$, of which N_r are unknown DOFs, and N_p are constrained DOFs. Because constrained DOFs need not be solved, the problem is reduced to Eq. (72). To build the Eq. (72), a link between the DOFs of each node and the index of $\mathbf{u}_u$ is created as `idx[i][j]`, in which i represents the i-th node, and $j = 0, 1, 2$ represents the DOF. If the DOF is unknown, `idx[i][j]` represents position of the DOF in vector $\mathbf{u}_u$, otherwise `idx[i][j]` $= -1$. Determination of the 2D array `idx[][]` is shown in Algorithm 1.

Algorithm 1: Determine the array idx[ ][ ].

```
count ← 0;
for i ← 0 to < N_g do
    for j ← 0 to 3 do
        if j-th DOF of i-th node is constrained then
            idx[i][j] ← −1;
        else
            idx[i][j] ← count;
            count++;
```

Note that `idx[][]` is an alternative to the mapping matrices in Eqs. (29), (30) and (49). For example, in Eq. (29),

$$\mathbf{u}_{(i)} = [u_{1(i)} \quad u_{2(i)} \quad u_{3(i)}]^T = \mathbf{M}_{(i)}\mathbf{u}_g, \tag{73}$$

if DOF $u_{j(i)}$, $(j = 1, 2, 3)$ is unknown, the j-th row and idx[i][j-1]-th column coefficient of $\mathbf{M}_{(i)}$ is $\mathbf{M}_{(i)j,\text{idx[i][j-1]}} = 1$ and other coefficients of $\mathbf{M}_{(i)}$ are zero. Thus, during assembling the stiffness matrix, there is no need to build the mapping matrices.

4.2 Create interactive list and determine the non-zero coefficients' indices

It is impractical to store the full stiffness matrix. For example, a 3D domain with 10^5 nodes requires 670.5 gigabytes to store the full stiffness matrix using double precision. Thus, the stiffness matrix is stored in a CSR format, which only stores the non-zero coefficients and their column and row indices information by 3 arrays, `K[]`, `ja[]`, and `ia[]`, respectively. The array `K[]` contains the non-zero coefficients of the matrix corresponding to the indices in `ja[]`, which stores the column indices of each non-zero coefficient. The array `K[]` and `ja[]` have the same size. The array `ia[]` has a size of $N_r + 1$, in which N_r is the total number of rows of the matrix. `ia[i]` $(0 \leq i < N_r)$ points to the first column index of row i in the `ja[]`. That is, `ia[i]` gives the index of the coefficient in array `K[]` that is the first non-zero coefficient from row i of the matrix. `(ia[i + 1]-ia[i])` gives the number of non-zero

Algorithm 2: Create interaction list.

```
   /* PD node interacting with its family members.                    */
 1 for n ← 0 to < N^pd do
       /* For each PD node.                                           */
 2     std::vector fami ← family member of n-th PD node;
 3     i ← fami.at(0) ;                    // The 1st family member is itself.
 4     iN[i].assign(fami.begin(), fami.end());
   /* PD node of the PDBE interacting with the nodes of the same PDBE and
      their family members.                                           */
 5 for e ← 0 to < N_S^pd do
       /* For each PDBE.                                              */
 6     for n_1 ← node of the e-th PDBE do
 7         std::vector fami ← the family members of the node n_1;
 8         for n_2 ← node of the e-th PDBE do
 9             if n_1 != n_2 then
10                 iN[n_2].insert(iN[n_2].end(), fami.begin(), fami.end());
   /* Node of the FE interacting with the node of the same FE.        */
11 for e ← 0 to < N^fe do
       /* For each FE.                                                */
12     for n_1 ← node of the e-th FE do
13         for n_2 ← node of the e-th FE do
14             iN[n_1].push_back(n_2);
   /* Delete the duplicated nodes and sort the interaction list in
      ascending order.                                                */
15 unordered_set<int> inteSet ;                  // declare a unordered_set
16 for n ← 0 to < N_g do
       /* For each node.                                              */
17     for nid ← node in list iN[n] do
18         inteSet.insert(nid) ;                  // store the unique nodes
19     iN[n].assign(inteSet.begin(), inteSet.end());
20     sort(iN[n].begin(), iN[n].end()) ;        // sort in a ascending order
21     inteSet.clear();
```


coefficients of the row i of the matrix, and the last element `ia[`N_r`]` is taken to be equal to the number of non-zero coefficient in the matrix. To assemble the stiffness matrix, we need to determine the indices of rows and columns of all the non-zero coefficients and store them in the arrays `ia[]` and `ja[]`. In this subsection, we present how to obtain the array `ia[]` and `ja[]`.

Create interaction list The row `idx[i][j]` of the stiffness matrix is corresponding to the j-th DOF of i-th node. The non-zero coefficients of row `idx[i][j]` are determined by other nodes which are interacting with the i-th node. Thus, the interaction list `iN[i]` is created to store the nodes which are interacting with the i-th node. `iN[i]` is a standard sequence container `std::vector`, whose size can be changed dynamically. There are 3 interaction types: (1) PD node in the PD domain interacting with its family members; (2) PD node of PDBE interacting with the nodes of the same PDBE and their

family members; and (3) FE's nodes interacting with the nodes of the same FE. Note that interaction list `iN[i]` of node i contains itself. The pseudo-code to obtain the interaction list is shown in Algorithm 2.

Algorithm 3: Obtain arrays ia[ ] and ja[ ]

```
1  Initialize ia[ ]=0 ;                                  // size of ia is N_r + 1.
2  std::vector<int> vja ;                                // declare a vector for ja[ ].
3  for n_1 ← 0 to < N_g do
       /* For each node.                                                        */
4      for j_1 ← 0 to < 3 do
5          if idx[n_1][j_1] != −1 then
6              for n_2 ← node in list iN[n_1] do
7                  for j_2 ← 0 to < 3 do
8                      if idx[n_2][j_2]!=-1 then
9                          vja.push_back(idx[n_2][j_2]);
10                         ia[idx[n_1][j_1]+1]++;
11 for i ← 0 to < N_r do
12     ia[i+1] = ia[i] + ia[i+1];
13 for i ← 0 to <ia[N_r] do
14     ja[i] = vja.at(i);
```

Determine the non-zero coefficients' indices With the interaction list, `iN`, of each node, the row and column information of all the non-zero coefficients of the stiffness matrix $\mathbf{K}_{uu}$ can be determined. The column and row indices information of the non-zero coefficients are stored in the arrays `ja[]` and `ia[]`. The pseudo-code for obtaining the `ia[]` and `ja[]` is shown in Algorithm 3.

4.3 Assembling the stiffness matrix by MPI

With the arrays `ia[]` and `ja[]`, the positions of all non-zero coefficients of the stiffness matrix are known. To find the corresponding position in the array `K[]` of the *row*-th row and *col*-th column coefficient of the stiffness matrix $\mathbf{K}_{uu}$, a function `findCSRIndexOfMat(row, col)` is first built. A binary searching method is employed in this function to take advantage of ascending order of `ja[]` for each row. The assembling procedure of the stiffness matrix is time-consuming, but is independent between each PD node, each PDBE, and each FE. Thus, the MPI technique is applied to assemble the stiffness matrix. Note that, during assembling the stiffness matrix $\mathbf{K}_{uu}$, the item $-\mathbf{K}_{up}\mathbf{u}_p$ is obtained simultaneously and stored in the array `local_F[]`. With the stiffness matrix and equivalent nodal force, the displacements are solved by the PDSS, which is also based on the MPI technique. The pseudo-code for assembling the stiffness matrix is presented in Algorithm 4. For the matrix $\mathbf{K}_s^{pd}$, we only illustrate

Algorithm 4: Assembling the stiffness matrix

```
   /* create a function to find the index of the non-zero coefficient
      K_uu[row][col] in the array K[ ], using a binary method.          */
 1 Function findCSRIndexOfMat(row, col):
 2     low = ia[row];
 3     up = ia[row+1];
 4     while low<up do
 5         mid = (low+up)/2;
 6         if ja[mid] == col then
 7             return mid;
 8         else if col<ja[mid] then
 9             up = mid;
10         else
11             low = mid + 1;
12 Initialize local_K[ ] = 0 ;                  // size of local_K[ ] is ia[N_r].
13 Initialize local_F[ ] = 0 ;                      // size of local_F[ ] is N_r.
14 rank ← rank ID;
15 numProc ← number of cores;
   /* Assemble the matrix K_b^pd                                          */
16 startP = rank * N^pd / numProc;
17 endP = (rank + 1) * N^pd / numProc;
18 for i ← startP to <endP do
19     std::vector fami ← family member of i-th PD node;
20     n_1 = fami.at(0);
21     H[ ][ ] ← matrix (−V_(i) H_(i)) of the i-th PD node n_1;
       /* Start Assembling                                                */
22     for j_1 ← 0 to < 3 do
23         if idx[n_1][j_1] != -1 then
24             for m ← 0 to <fami.size() do
25                 n_2 = fami.at(m);
26                 for j_2 ← 0 to < 3 do
27                     if idx[n_2][j_2]!=-1 then
28                         t = findCSRIndexOfMat(idx[n_1][j_1], idx[n_2][j_2]);
29                         local_K[t] = local_K[t] + H[j_1][3m + j_2];
30                     else
31                         local_F[idx[i][j_1]] = local_F[idx[i][j_1]] −
                             H[j_1][3m + j_2]*u[n_2][j_2];
   /* Assemble the matrix K^fem.                                          */
32 startP = rank * N^fe / numProc;
33 endP = (rank + 1) * N^fe / numProc;
34 for e ← startP to <endP do
35     K_e ← Element stiffness matrix;
       /* Start Assembling. N^fe is number of FE, N_e is number of node in
          FE.                                                             */
36     for i_1 ← 0 to < N_e do
37         n_1 ← the i_1-th node of the FE;
38         for j_1 ← 0 to < 3 do
39             if idx[n_1][j_1] != -1 then
40                 for i_2 ← 0 to < N_e do
41                     n_2 ← the i_2-th node of the FE;
42                     for j_2 ← 0 to < 3 do
43                         if idx[n_2][j_2] != -1 then
44                             t = findCSRIndexOfMat(idx[n_1][j_1], idx[n_2][j_2]);
45                             local_K[t] = local_K[t] + K_e[3i_1 + j_1][3i_2 + j_2];
46                         else
47                             local_F[idx[n_1][j_1]] = local_F[idx[n_1][j_1]] −
                                 K_e[3i_1 + j_1][3i_2 + j_2]*u[n_2][j_2];
```

```
   /* Assemble the matrix K_s^pd.                                          */
48 startP = rank * N_S^pd / numProc;
49 endP = (rank + 1) * N_S^pd / numProc;
50 for e ← startP to <endP do
51     for p ← Gauss points do
52         for q ← Gauss points do
53             for i ← to 0 < 4 do
54                 e_i ← the i-th node of the PDBE;
55                 std::vector fami ← family members of the node e_i;
56                 eK[ ][ ] ← matrix of (w_p w_q N^T N̄ N_i DC^(e_i));
                   /* Start Assembling.                                    */
57                 for i_1 ← to 0 < 4 do
58                     n_1 ← the i_1-th node of the PDBE;
59                     for j_1 ← to 0 < 3 do
60                         if idx[n_1][j_1] != -1 then
61                             for m ← 0 to < fami.size() do
62                                 n_2 = fami.at(m);
63                                 for j_2 ← to 0 < 3 do
64                                     if idx[n_2][j_2] != -1 then
65                                         t = findCSRIndexOfMat(idx[n_1][j_1],
                                            idx[n_2][j_2]);
66                                         local_K[t] = local_K[t] +
                                            eK[3i_1 + j_1][3m + j_2];
67                                     else
68                                         local_F[idx[n_1][j_1]]=local_F[idx[n_1][j_1]] −
                                            eK[3i_1 + j_1][3m + j_2] * u[n_2][j_2];
69 Get the nodal forces from external forces and add to local_F[ ];
   /* Get K[ ] and nodal force F[ ] by the MPI_Allreduce routine of MPI.   */
70 K[ ] ← summation of local_K[ ] by MPI_Allreduce routine;
71 F[ ] ← summation of local_F[ ] by MPI_Allreduce routine;
```


the assembling procedure for 4-noded PDBE for conciseness, and for 3-noded PDBE, a similar procedure can be used.

5 Numerical results and performance evaluation

To demonstrate the capability of this proposed PDLSM-FEM, three static examples concerning displacements and stresses are conducted in this section. In the first example, a block under uniaxial uniform tension at various PD volume fraction, m-ratio, and number of cores is analyzed in this section, to better quantify the improvement of the computational cost of the proposed coupling method. The second example concerns a block with a hole at its center under displacement loading, and the third example is a block with a pre-existing crack under displacement loading. For the first and third examples, the material constants are Young's modulus $E = 70$ GPa and Poisson's ratio

$\nu = 0.33$. For the second example, Young's modulus and Poisson's ratio are specified as $E = 200$ GPa and $\nu = 0.3$, respectively.

In these simulations, the nodal stresses in the FEM domain and the PD domain are calculated by the FEM formulation and the PD equation, Eq. (24), respectively. For the PD-FEM interface, the nodal stresses are calculated by the average of FEM and PD formulations. All PD bonds that pass through the pre-existing crack are broken. We also analyze the second and third examples by ANSYS. The comparison of the displacement and stress by PDLSM-FEM and ANSYS shows good agreements between the two methods. Note that the influence function $\omega(|\boldsymbol{\xi}|)$ may affect the accuracy in numerical implementation. In this paper, we specify the influence function as a Gaussian distribution in the form of

$$\omega(|\boldsymbol{\xi}|) = e^{-(\frac{|\boldsymbol{\xi}|}{c\delta_{(i)}})^2}, \tag{74}$$

here, c is a constant and we define it as a *non-local factor*. In this paper, we take $c = 1/4$.

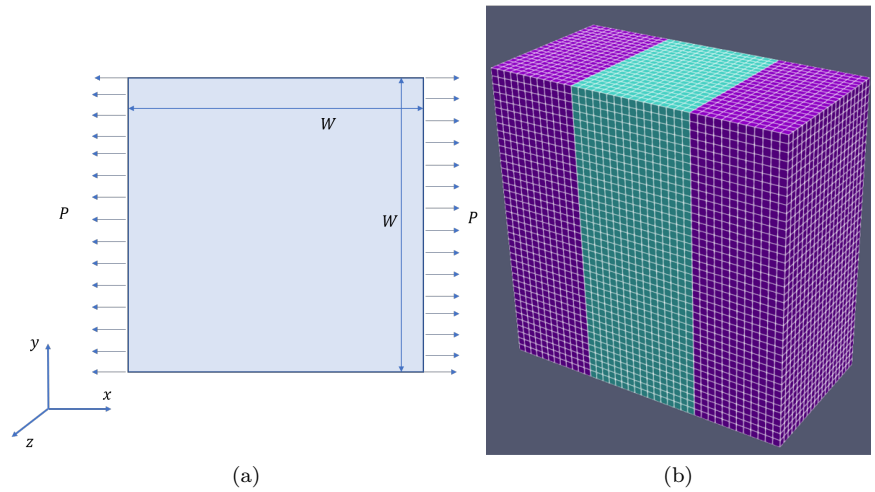
All the PDLSM-FEM simulations are run on the "Moles" computing nodes of the "Beocat" machine at Kansas State University. The "Beocat" machine has 120 "Moles" computing nodes and each node has 32 GB RAM, 2 "Xeon E5-2630 v4" CPU processors with a total of 20 cores, and 2 "Intel I350" network interface controllers.

5.1 A block under uniaxial uniform tension

As shown in Fig. 5a, a block with size 1 m × 1 m × 0.5 m is under uniaxial uniform tension $P = 0.7$ MPa. The middle part of the block is specified as PD domain, and the left and right parts are specified as FEM domain, as shown in Fig. 5b. The whole domain is discretized with uniform cube mesh with mesh size 25 mm, and it contains a total of 32000 elements and a total of 35301 nodes. To better evaluate the performance of the coupling method, we simulate the example with various volume fraction of PD domain $v_r = \frac{V_{pd}}{V_{tot}}$. Fig. 5b illustrates the meshes for $v_r = 0.4$ with blue PD domain and purple FEM domain. Note that $v_r = 0$ represents a pure FEM model, and $v_r = 1.0$ represents a pure PDLSM model.

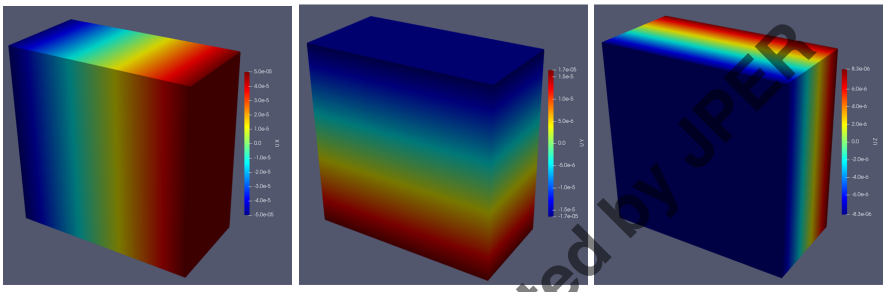
Fig. 6 shows the displacement results for $v_r = 0.4$. As expected, the displacements u_x, u_y, and u_z vary linearly along the x-, y-, and z-direction, respectively. Although not shown here, for all the simulations of this example, the magnitude of the relative errors of the displacement results of each node is up to machine precision, compared to the analytical results of $u_x = \frac{P}{E}x$, $u_y = -\nu\frac{P}{E}y$, and $u_z = -\nu\frac{P}{E}z$, as both PDLSM and FEM can predict exactly the mechanical behaviors of this example.

Fig. 7 illustrates the wall time and speedup of the coupling model for assembling the stiffness matrix and solving the system equations, with constant $m = 4$, yet various v_r and various cores. The speedup is defined as the wall time ratio of the coupling model to the pure PD model ($v_r = 1.0$). As Fig.



(a) (b)

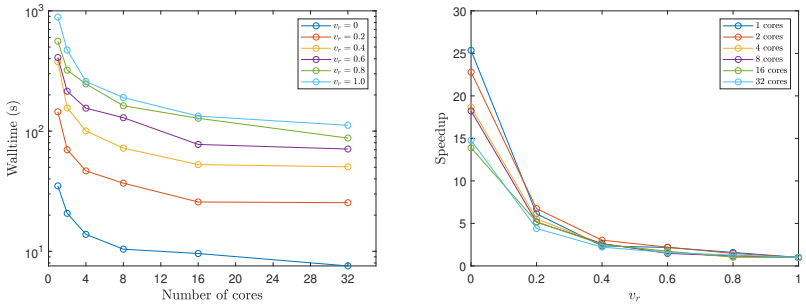
Fig. 5: The block with uniaxial uniform tension: (a) Geometry and loading, (b) illustrated uniform meshes for $v_r = 0.4$.



(a) (b) (c)

Fig. 6: Displacement results of the block under uniaxial uniform tension: (a) u_x, (b) u_y, (c) u_z.

7a shows, the coupling model scales well from 1 to 4 cores, almost linearly, after which, the performance is not improved substantially as the number of cores increases further due to communication between cores. After the number of cores reaches 16, the wall time is almost an constant. Thus, for saving computing resources, we may run the program on up to 16 cores. As shown in Fig. 7b, this coupling model improves computational efficiency significantly. The pure FEM model ($v_r = 0$) is $14 - 25$ times faster than the pure PD model ($v_r = 1$) and a larger volume of FEM domain saves more computational cost. When $v_r = 0.2$, the coupling model is $4.4 - 6.7$ times faster than the pure PD model. Note that we use uniform mesh in this example, but for practical applications of PDLSM-FEM, we can use very coarse mesh in the



(a) (b)

Fig. 7: Wall time and speedup of the coupling model for assembling the stiffness matrix and solving the system equations with constant $m = 4$, yet various v_r and various cores.

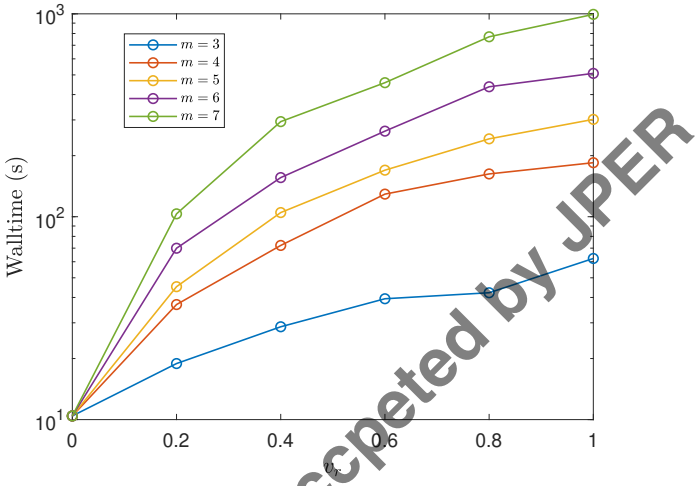


Fig. 8: Wall time for assembling the stiffness matrix and solving the system equations with constant 8 cores, yet various v_r and various m-ratio.

FEM domain, which is far from the discontinuous region, and relative finer mesh in the PD domain, where discontinuities occur. This treatment may save much more computational cost compared to treating the whole domain as a full PD domain with fine mesh. In general and by comparison, to get results with similar accuracy, PDLSM requires finer mesh than FEM does, due to two reasons: (a) PDLSM assumes that the displacements are in 2nd-order form (see Section 2) within the interaction domain H_x, which contains many elements inside, while FEM assumes the displacements are in a 2nd-order form (for 8-noded brick elements) or even higher-order form (depends

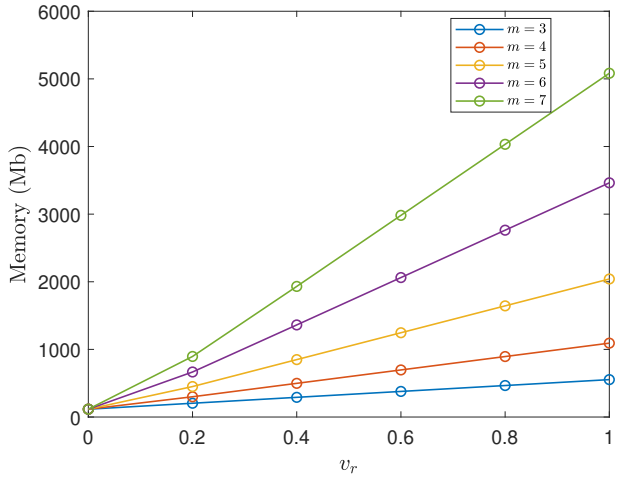


Fig. 9: Memories for storing the stiffness matrix in CSR format in each core with various v_r and various m-ratio.

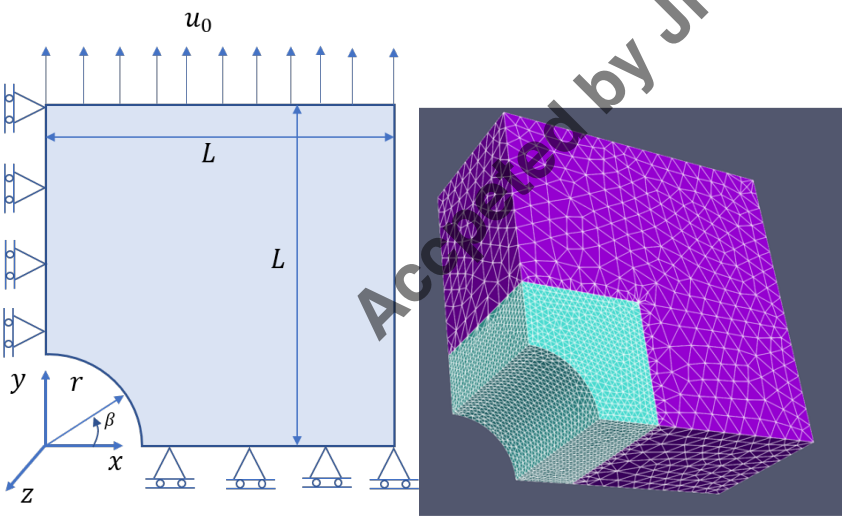
on element type) within every single element, which is much smaller than the interaction domain of PD; (b) numerical implementation of PD employs the simple meshless method with a one-point quadrature for cells, while FEM employs the element-based method with Gauss integration which is much more accurate than the one-point quadrature method.

Fig. 8 shows the wall time for assembling the stiffness matrix and solving the system equations, with constant 8 cores, yet various v_r and various m-ratio. As it shows, increasing m-ratio makes the consuming time increase sharply. When $m = 7$ and $v_r = 1.0$, the run time is 994 s, which is about 96 times more than the pure FEM model with run time 10.4 s. Fig. 9 shows the memories for storing the stiffness matrix in CSR format for each core with various v_r and various m-ratio. Note that the data types of the arrays `ia[]` and `ja[]` are `long long int`, and non-zero coefficients are stored by double precision. As it shows, the memory increases almost linearly when increasing PD volume fraction. Increasing the m-ratio makes the memory increase sharply and when m-ratio is larger, slightly increasing m-ratio leads to an enormous increase in the memory usage due to the enormous increase of interaction domain volumn. When $m = 7$ and $v_r = 1.0$, the memory usage to store the stiffness matrix is 5083 MB, which is about 43.8 times more than that of the pure FEM model, with a memory usage of 116 MB. In comparison, if the full stiffness matrix was stored for this example, it would have required memory of 83.4 GB for each core. Thus, in general, the proposed coupling PDLSM-FEM greatly reduces computational cost in terms of program run time and memory usage.

5.2 A block with a hole under displacement loading

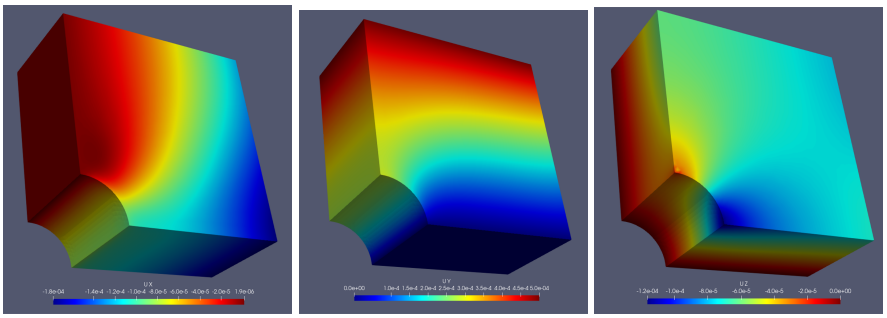
A block with a central hole, as depicted in Fig. 10, is under a displacement loading $u_0 = 5 \times 10^{-5}$ m. Due to symmetry, only one-eighth of the box is modeled in this simulation. The symmetrical boundary conditions are applied on the faces $x = 0$, $y = 0$ and $z = 0$. The radius of the hole is $r = 0.1$ m. The length of the model is $L = 0.5$ m and the thickness of the model is $B = 0.25$ m. The whole model utilizes non-uniform discretization with a total of 16480 nodes and 90811 tetrahedron elements. The region $0 \leq x \leq 0.2$ m $\cap\, 0 \leq y \leq 0.2$ m is discretized with 12009 PD nodes and 66741 tetrahedron PD elements. The interaction domain of each PD node is determined by $\delta_{(i)} = 4\Delta_{(i)}$. Fig. 10b illustrates the tetrahedron meshes with blue PD domain and purple FEM domain.

Fig. 11 presents the displacement results of the block with a hole by PDLSM-FEM. Fig. 12a demonstrates the comparison of u_x and u_y along the edge of the hole at $z = 0$ obtained from PDLSM-FEM and ANSYS. It is clear that u_x and u_y obtained from the proposed method and ANSYS are in excellent agreement. Fig. 12b illustrates the comparison of σ_y and σ_z along the edge of the hole at $z = 0$ obtained from the proposed method and ANSYS. It reveals that stresses values obtained from these two methods are in excellent agreement while the σ_y values predicted using the proposed method is slightly lower than that obtained using ANSYS.



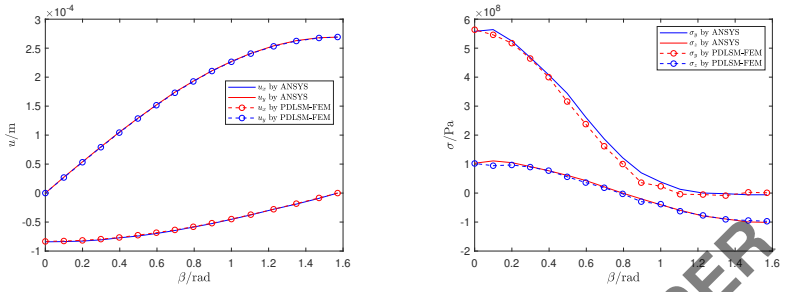
(a) (b)

Fig. 10: The block with a hole under displacement loading: (a) Geometry and loading, (b) Non-uniform tetrahedron meshes.



(a) (b) (c)

Fig. 11: Displacements of the block with a hole: (a) u_x, (b) u_y, (c) u_z.



(a) (b)

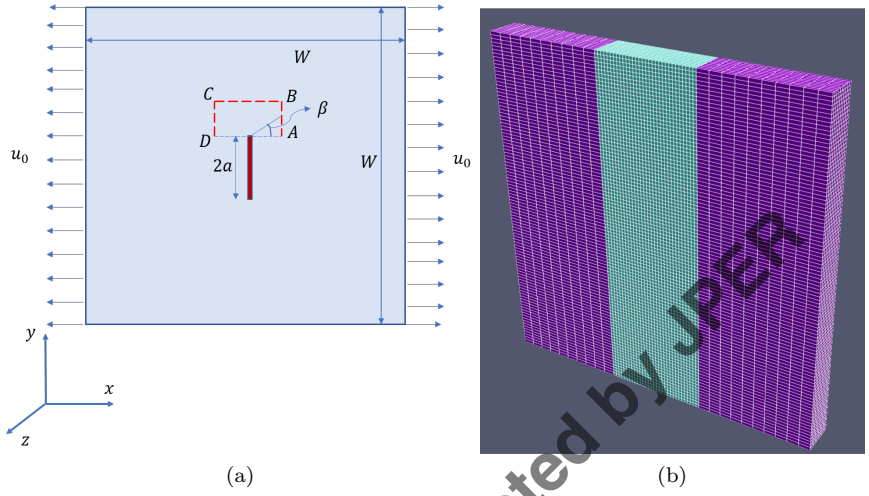
Fig. 12: Comparison of PDLSM-FEM with ANSYS for displacement and stress solutions along with the edge of the hole at $z = 0$: (a) Comparison of displacements u_x and u_y, (b) comparison of stresses σ_y and σ_z.

5.3 A block with a pre-existing crack under displacement loading

As shown in Fig. 13a, a block with a pre-existing crack is subjected to displacement loading $u_0 = 5 \times 10^{-5}$ m. The edge length of the block is $L = 1$ m and the thickness of block is $B = 0.1$ m. The length of the crack is $2a = 0.2$ m. A fictitious plane is used to represents the crack and all PD bonds crossing the plane are broken. The whole domain is constructed by non-uniform discretization with a total of 57772 nodes and 51000 hexahedron elements. The region -0.15 m $\leq x \leq 0.15$m is specified as the PD domain and is discretized with 35552 PD nodes and 31000 hexahedron elements. The family size is specified as $\delta_{(i)} = 4\Delta_{(i)}$. Fig. 13b illustrates the hexahedron meshes with blue PD domain and purple FEM domain.

Fig. 14 demonstrates the displacements from PDLSM-FEM and Fig. 15 presents the deformed shape and stress results of the block. Note that the deformation is magnified 2000 times in Fig. 15. As shown in the figures, an obvious discontinuity occurs at the crack position and stresses are concentrated

at the crack front. Fig. 16 elucidates the displacements of u_x and u_y, the normal stresses, σ_x, and the shear stresses, σ_{xy}, from PDLSM-FEM and ANSYS, along with path $\overline{ABCD}$ as seen in Fig. 13a, where the coordinates of the key points of the path are A $(0.15, 0.1, 0)$ m, B $(0.15, 0.25, 0)$ m, C $(-0.15, 0.25, 0)$ m and D $(-0.15, 0.1, 0)$ m. It is observed that for u_x, the proposed method and ANSYS generate very close results for the horizontal segment $\overline{BC}$; for the vertical segments $\overline{AB}$ and $\overline{CD}$, PDLSM-FEM generates slightly larger value than ANSYS. Very interestingly, for u_y, PDLSM-FEM and ANSYS generates very close results on the vertical segments $\overline{AB}$ and $\overline{CD}$; on the horizontal segments $\overline{BC}$, PDLSM-FEM generates slightly larger results. It is very clear that for both stress and displacement, the results by the two methods are in good agreement.

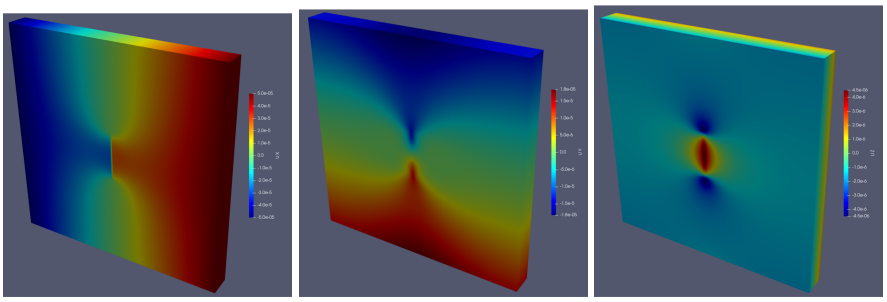


(a) (b)

Fig. 13: The block with a pre-existing crack under displacement loading: (a) Geometry and loading, (b) Non-uniform hexahedron meshes.

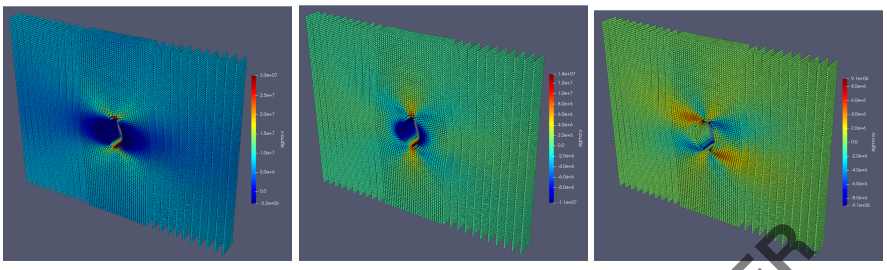
6 Conclusions

In this work, we present a new framework to couple PDLSM with FEM (PDLSM-FEM) for 3D problems based on the weighted residual method. In PDLSM-FEM, the problem domain is divided into a PD domain and a FEM domain, and both domains are discretized into uniform or non-uniform elements. The coupling method is straightforward, and the formulation can be converted conveniently between PDLSM, FEM, and PDLSM-FEM. To directly and implicitly solve the 3D problems, we present an algorithm to assemble the global stiffness matrix in a compressed sparse row (CSR) format by message



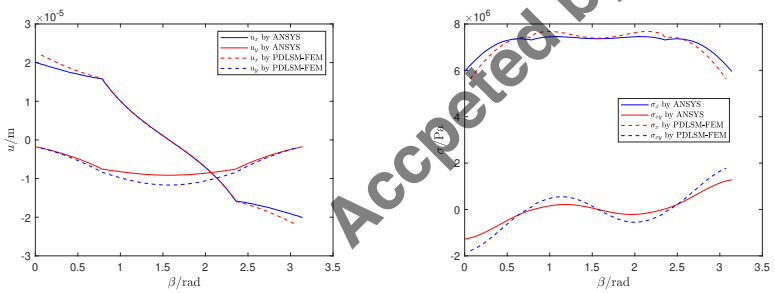
(a) (b) (c)

Fig. 14: Displacement results from the block with a pre-existing crack: (a) u_x, (b) u_y, (b) u_z.



(a) (b) (c)

Fig. 15: Deformed shape and stress results of the block a with pre-existing crack: (a) σ_x, (b) σ_y, (c) σ_{xy}



(a) (b)

Fig. 16: Comparison of PDLSM-FEM with ANSYS for displacement and stress solutions along with the path $\overline{ABCD}$: (a) Comparison of displacements u_x and u_y; (b) comparison of stresses σ_x and σ_{xy}.

passing interface technique (MPI). Within the CSR format, only non-zero coefficients of the stiffness matrix, and their row and column indices information are stored. The system equations are solved by the parallel direct sparse solver which is also based on MPI. Three static examples are performed to demonstrate the capability of this proposed PDLSM-FEM. For the first example, the magnitude of the relative errors of the displacement results of each node is up to machine precision, compared to the analytical results. The performance evaluation of the first example shows the MPI technique improves the computational efficiency generally and the maximum number of cores is suggested as 16. The performance evaluation also demonstrates that the proposed PDLSM-FEM significantly improves the computational efficiency, which is up to 96 times faster than the pure PDLSM, and significantly saves memory usage, which is up to 43 times less than the pure PDLSM. For the second and third examples, the displacements and stresses are obtained by PDLSM-FEM, and comparison with results using ANSYS demonstrates the proposed PDLSM-FEM has good accuracy.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. Silling, S.A.: Reformulation of elasticity theory for discontinuities and long-range forces. *Journal of the Mechanics and Physics of Solids* **48**(1), 175–209 (2000)
2. Madenci, E., Oterkus, E.: *Peridynamic theory and its applications*. Springer New York, New York, NY (2014). DOI 10.1007/978-1-4614-8465-3
3. Silling, S.A., Epton, M., Weckner, O., Xu, J., Askari, E.: Peridynamic states and constitutive modeling. *Journal of Elasticity* **88**(2), 151–184 (2007). DOI 10.1007/s10659-007-9125-1
4. Le, Q.V., Bobaru, F.: Surface corrections for peridynamic models in elasticity and fracture. *Computational Mechanics* **61**(4), 499–518 (2018). DOI 10.1007/s00466-017-1469-1
5. Seleson, P.: Improved one-point quadrature algorithms for two-dimensional peridynamic models based on analytical calculations. *Computer Methods in Applied Mechanics and Engineering* **282**, 184–217 (2014). DOI 10.1016/j.cma.2014.06.016
6. Madenci, E., Dorduncu, M., Gu, X.: Peridynamic least squares minimization. *Computer Methods in Applied Mechanics and Engineering* **348**, 846–874 (2019). DOI 10.1016/j.cma.2019.01.032
7. Liu, Q., Xin, X.: Revised non-ordinary state-based peridynamics and a new framework for coupling with finite element method. *Engineering Fracture Mechanics* **242**, 107483 (2021). DOI 10.1016/j.engfracmech.2020.107483
8. Oterkus, E., Madenci, E.: Peridynamic analysis of fiber-reinforced composite materials. *Journal of Mechanics of Materials and Structures* **7**(1), 45–84 (2012). DOI 10.2140/jomms.2012.7.45
9. Wu, C., Ren, B.: A stabilized non-ordinary state-based peridynamics for the nonlocal ductile material failure analysis in metal machining process. *Computer Methods in Applied Mechanics and Engineering* **291**, 197–215 (2015). DOI 10.1016/j.cma.2015.03.003
10. Song, Y., Yan, J., Li, S., Kang, Z.: Peridynamic modeling and simulation of ice craters by impact. *Computer Modeling in Engineering and Sciences* **121**(2), 465–492 (2019). DOI 10.32604/cmes.2019.07190

11. Chu, B., Liu, Q., Liu, L., Lai, X., Mei, H.: A rate-dependent peridynamic model for the dynamic behavior of ceramicmaterials. *Computer Modeling in Engineering and Sciences* **124**(1), 151–178 (2020). DOI 10.32604/cmes.2020.010115
12. Waxman, R., Guven, I.: An experimental and peridynamic study of the erosion of optical glass targets due to sand and sphere microparticles. *Wear* **428-429**, 340–355 (2019). DOI 10.1016/j.wear.2019.04.003
13. Rivera, J., Berjikian, J., Ravinder, R., Kodamana, H., Das, S., Bhatnagar, N., Bauchy, M., Krishnan, N.M.A.: Glass Fracture Upon Ballistic Impact: New Insights From Peridynamics Simulations. *Frontiers in Materials* **6**, 239 (2019). DOI 10.3389/fmats.2019.00239
14. Wu, L., Wang, L., Huang, D., Xu, Y.: An ordinary state-based peridynamic modeling for dynamic fracture of laminated glass under low-velocity impact. *Composite Structures* **234**, 111722 (2020). DOI 10.1016/j.compstruct.2019.111722
15. Wang, L., Xu, J., Wang, J.: A peridynamic framework and simulation of non-Fourier and nonlocal heat conduction. *International Journal of Heat and Mass Transfer* **118**, 1284–1292 (2018). DOI 10.1016/j.ijheatmasstransfer.2017.11.074
16. Gu, X., Zhang, Q., Madenci, E.: Refined bond-based peridynamics for thermal diffusion. *Engineering Computations* **36**(8), 2557–2587 (2019). DOI 10.1108/EC-09-2018-0433
17. Gao, Y., Oterkus, S.: Non-local modeling for fluid flow coupled with heat transfer by using peridynamic differential operator. *Engineering Analysis with Boundary Elements* **105**, 104–121 (2019). DOI 10.1016/j.enganabound.2019.04.007
18. Katiyar, A., Agrawal, S., Ouchi, H., Seleson, P., Foster, J.T., Sharma, M.M.: A general peridynamics model for multiphase transport of non-Newtonian compressible fluids in porous media. *Journal of Computational Physics* **402**, 109075 (2020). DOI 10.1016/j.jcp.2019.109075
19. Diana, V., Carvelli, V.: An electromechanical micropolar peridynamic model. *Computer Methods in Applied Mechanics and Engineering* **365**, 112998 (2020). DOI 10.1016/j.cma.2020.112998
20. Zeleke, M.A., Lai, X., Liu, L.: A peridynamic computational scheme for thermoelectric fields. *Materials* **13**(11), 2546 (2020). DOI 10.3390/ma13112546
21. Kilic, B., Madenci, E.: Coupling of peridynamic theory and the finite element method. *Journal of Mechanics of Materials and Structures* **5**(5), 707–733 (2010). DOI 10.2140/jomms.2010.5.707
22. Agwai, A., Guven, I., Madenci, E.: Damage prediction for electronic package drop test using finite element method and peridynamic theory. In: 2009 59th Electronic Components and Technology Conference, pp. 565–569. IEEE, San Diego, CA, USA (2009). DOI 10.1109/ECTC.2009.5074069
23. Lubineau, G., Azdoud, Y., Han, F., Rey, C., Askari, A.: A morphing strategy to couple non-local to local continuum mechanics. *Journal of the Mechanics and Physics of Solids* **60**(6), 1088–1102 (2012). DOI 10.1016/j.jmps.2012.02.009
24. Liu, W., Hong, J.W.: A coupling approach of discretized peridynamics with finite element method. *Computer Methods in Applied Mechanics and Engineering* **245-246**, 163–175 (2012). DOI 10.1016/j.cma.2012.07.006
25. Zaccariotto, M., Mudric, T., Tomasi, D., Shojaei, A., Galvanetto, U.: Coupling of FEM meshes with Peridynamic grids. *Computer Methods in Applied Mechanics and Engineering* **330**, 471–497 (2018). DOI 10.1016/j.cma.2017.11.011
26. Wang, Yongwei, Han, F., Lubineau, G.: A hybrid local/nonlocal continuum mechanics modeling and simulation of fracture in brittle materials. *Computer Modeling in Engineering and Sciences* **121**(2), 399–423 (2019). DOI 10.32604/cmes.2019.07192
27. Tong, Y., Shen, W.Q., Shao, J.F.: An adaptive coupling method of state-based peridynamics theory and finite element method for modeling progressive failure process in cohesive materials. *Computer Methods in Applied Mechanics and Engineering* **370**, 113248 (2020). DOI 10.1016/j.cma.2020.113248
28. Bie, Y., Cui, X., Li, Z.: A coupling approach of state-based peridynamics with node-based smoothed finite element method. *Computer Methods in Applied Mechanics and Engineering* **331**, 675–700 (2018). DOI 10.1016/j.cma.2017.11.022
29. Sun, W., Fish, J.: Superposition-based coupling of peridynamics and finite element method. *Computational Mechanics* **64**(1), 231–248 (2019). DOI 10.1007/s00466-019-01668-5

30. Dhia, H.B.: Multiscale mechanical problems: the arlequin method. *Comptes Rendus de l'Academie des Sciences Series IIB Mechanics Physics Astronomy* **12**(326), 899–904 (1998)
31. Wang, X., Kulkarni, S.S., Tabarraei, A.: Concurrent coupling of peridynamics and classical elasticity for elastodynamic problems. *Computer Methods in Applied Mechanics and Engineering* **344**, 251–275 (2019). DOI 10.1016/j.cma.2018.09.019
32. Dong, Y., Su, C., Qiao, P.: A stability-enhanced peridynamic element to couple non-ordinary state-based peridynamics with finite element method for fracture analysis. *Finite Elements in Analysis and Design* **181**, 103480 (2020). DOI 10.1016/j.finel.2020.103480
33. Pagani, A., Carrera, E.: Coupling three-dimensional peridynamics and high-order one-dimensional finite elements based on local elasticity for the linear static analysis of solid beams and thin-walled reinforced structures. *International Journal for Numerical Methods in Engineering* **121**(22), 5066–5081 (2020). DOI https://doi.org/10.1002/nme.6510. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.6510
34. Shen, F., Yu, Y., Zhang, Q., Gu, X.: Hybrid model of peridynamics and finite element method for static elastic deformation and brittle fracture analysis. *Engineering Analysis with Boundary Elements* **113**, 17–25 (2020). DOI 10.1016/j.enganabound.2019.12.016
35. Liu, Q., Xin, X., Ma, J., Wang, Y.: Simulating quasi-static crack propagation by coupled peridynamics least square minimization with finite element method. *Engineering Fracture Mechanics* **252**, 107862 (2021). DOI 10.1016/j.engfracmech.2021.107862
36. Kilic, B., Madenci, E.: An adaptive dynamic relaxation method for quasi-static simulations using the peridynamic theory. *Theoretical and Applied Fracture Mechanics* **53**(3), 194–204 (2010). DOI 10.1016/j.tafmec.2010.08.001
37. Sun, S., Sundararaghavan, V.: A peridynamic implementation of crystal plasticity. *International Journal of Solids and Structures* **51**(19-20), 3350–3360 (2014). DOI 10.1016/j.ijsolstr.2014.05.027
38. Breitenfeld, M., Geubelle, P., Weckner, O., Silling, S.: Non-ordinary state-based peridynamic analysis of stationary crack problems. *Computer Methods in Applied Mechanics and Engineering* **272**, 233–250 (2014). DOI 10.1016/j.cma.2014.01.002
39. Zaccariotto, M., Luongo, F., sarego, G., Galvanetto, U.: Examples of applications of the peridynamic theory to the solution of static equilibrium problems. *The Aeronautical Journal* **119**(1216), 677–700 (2015). DOI 10.1017/S0001924000010770
40. Prakash, N., Stewart, R.J.: A multi-threaded method to assemble a sparse stiffness matrix for quasi-static solutions of linearized bond-based peridynamics. *Journal of Peridynamics and Nonlocal Modeling* (2020). DOI 10.1007/s42102-020-00041-y
41. Ni, T., Zaccariotto, M., Zhu, Q.Z., Galvanetto, U.: Static solution of crack propagation problems in Peridynamics. *Computer Methods in Applied Mechanics and Engineering* **346**, 126–151 (2019). DOI 10.1016/j.cma.2018.11.028
42. Madenci, E., Dorduncu, M., Barut, A., Phan, N.: Weak form of peridynamics for nonlocal essential and natural boundary conditions. *Computer Methods in Applied Mechanics and Engineering* **337**, 598–631 (2018). DOI 10.1016/j.cma.2018.03.038
43. Hu, Y., Chen, H., Spencer, B.W., Madenci, E.: Thermomechanical peridynamic analysis with irregular non-uniform domain discretization. *Engineering Fracture Mechanics* **197**, 92–113 (2018). DOI 10.1016/j.engfracmech.2018.02.006
44. Silling, S., Askari, E.: A meshfree method based on the peridynamic model of solid mechanics. *Computers and Structures* **83**(17-18), 1526–1535 (2005). DOI 10.1016/j.compstruc.2004.11.026
45. Warren, T.L., Silling, S.A., Askari, A., Weckner, O., Epton, M.A., Xu, J.: A non-ordinary state-based peridynamic method to model solid material deformation and fracture. *International Journal of Solids and Structures* **46**(5), 1186–1195 (2009). DOI 10.1016/j.ijsolstr.2008.10.029
46. Madenci, E., Dorduncu, M., Barut, A., Phan, N.: A state-based peridynamic analysis in a finite element framework. *Engineering Fracture Mechanics* **195**, 104–128 (2018). DOI 10.1016/j.engfracmech.2018.03.033