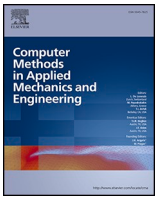
Computer Methods in Applied Mechanics and Engineering 429 (2024) 117159

Contents lists available at ScienceDirect

Computer Methods in Applied Mechanics and Engineering

journal homepage: www.elsevier.com/locate/cma



A robust radial point interpolation method empowered with neural network solvers (RPIM-NNS) for nonlinear solid mechanics

Jinshuai Bai [a,b,c], Gui-Rong Liu [d], Timon Rabczuk [e], Yizheng Wang [c,e], Xi-Qiao Feng [c], YuanTong Gu [a,b,*]

[a] School of Mechanical, Medical and Process Engineering, Queensland University of Technology, Brisbane, QLD 4000, Australia
[b] ARC Industrial Transformation Training Centre–Joint Biomechanics, Queensland University of Technology, Brisbane, QLD 4000, Australia
[c] Institute of Biomechanics and Medical Engineering, AML, Department of Engineering Mechanics, Tsinghua University, Beijing 100084, China
[d] Department of Aerospace Engineering and Engineering Mechanics, University of Cincinnati, OH 45221, USA
[e] Institute of Structural Mechanics, Bauhaus Universität Weimar, Weimar 99423, Germany

ARTICLE INFO

Keywords:
Radial point interpolation method
Neural network
Radial basis function
Nonlinear computational mechanics

ABSTRACT

In this work, we proposed a robust radial point interpolation method empowered with neural network solvers (RPIM-NNS) for solving highly nonlinear solid mechanics problems. It is enabled by neural network solvers via minimizing an energy-based functional loss. The RPIM-NNS has the following key ingredients: (1) It uses radial basis functions (RBFs) for displacement interpolation at arbitrary points in the problem domain, permitting irregular node distributions. (2) Nodes are placed also beyond the domain boundary, allowing the convenient implementation of boundary conditions of both Dirichlet and Neumann types. (3) It uses strain energy in an integral form as a part of the loss function, ensuring solution stability. (4) A well-developed gradient descendant algorithm in machine learning is employed to find the optimal solution, enabling robustness and ease in handling material and geometrical nonlinearities. (5) The proposed RPIM-NNS is compatible with parallel computing schemes. The performance of this method is tested using nonlinear problems including Cook's membrane and 3D twisting rubber problems, demonstrating its remarkable stability and robustness. This work, which seamlessly integrates the neural network solvers with mechanics governing equations and computational mechanics techniques, offers an excellent alternative for nonlinear solid mechanics problems. MATLAB codes are made available at https://github.com/JinshuaiBai/RPIM_NNS for free downloading.

1. Introduction

Nonlinear material and large deformation problems are the most prevailing nonlinear problems in solid mechanics. Due to their complexity, most problems are prohibitive to be analysed theoretically, and therefore numerical modelling becomes the major way to solve them. The well-known finite element method (FEM) [1] and its advanced version of the smoothed finite element method (S-FEM) [2] are the most popular numerical methods for solid mechanics modelling. It divides substances into small elements via meshes. However, the use of meshes needs to follow a basic rule; that is, each element should be convex without any excessive distortion. For

* Corresponding author at: School of Mechanical, Medical and Process Engineering, Queensland University of Technology, Brisbane, QLD 4000, Australia.
E-mail address: yuantong.gu@qut.edu.au (Y. Gu).

https://doi.org/10.1016/j.cma.2024.117159
Received 20 January 2024; Received in revised form 15 May 2024; Accepted 12 June 2024
Available online 26 June 2024

example, the interior angles of a 4-node isoparametric element should be less than 180°. This is because the positivity of the Jacobian determinant of each element must be ensured. The extreme mesh distortion caused by large deformation will induce severe numerical errors. Besides, the nonlinearities arising from material properties and large deformations are treated separately during numerical implementations. They are decoupled with each other and then solved by iterative algorithms, such as the Newton-Raphson algorithm [3], the Riks method [4], and the arc length method [5], etc.

Besides the FEM, the meshfree methods [6] have been proven to be effective as alternative approaches. Instead of using meshes, those meshfree methods utilise discrete nodes or points to represent the continuum substance, and thus they can avoid the mesh distortion for scenarios under large deformation. To date, various meshfree methods have been developed, including the element free Galerkin method (EFGM) [7], the smoothed particle hydrodynamics (SPH) [8], the meshless local Petrov-Galerkin (MLPG) method [9], and the radial point interpolation method (RPIM) [10]. The RPIM is a weak form meshfree method originally proposed by Liu et al. [11]. In the RPIM, a series of radial basis functions (RBFs) are applied to construct the displacement fields of the mechanics problems [12,13]. The shape function constructed by RPIM enjoys the Kronecker delta properties and is robust for irregular node distributions [14]. It has been demonstrated to be effective for various applications, such as functionally graded plates [15], acoustic problems [16], and magnetoelectric structures [17].

In the recent years, deep learning (DL) [18] has gained extensive attention, and great successes have been witnessed in various fields [19–21]. In DL models, the powerful artificial neural network, a bionic nonlinear computing system, which can approximate any continuous function, is harnessed to capture and excavate complex relations between raw data [22,23]. Furthermore, neural networks trained using the fundamental equations of mechanics can provide a superior accuracy in making predictions [24–28]. Consequently, there has been a surge of interest directed towards this particular field [29–32]. The success of DL should also be attributed to the well-developed neural network solvers that work for highly nonlinear loss functions [33]. In those solvers, a loss function is first formulated to quantify the performance of neural networks. Then, optimisers, mostly the gradient descendant algorithms, are applied to minimise the loss function by repetitively modifying the trainable parameters in the nonlinear computing systems [34].

Since the neural network solvers have shown remarkable capability to optimise nonlinear computing systems, a question naturally emerges. Can those powerful neural network solvers be embedded into traditional meshfree methods to deal with solid mechanics modelling, particularly for nonlinear problems? By doing so, the meshfree methods secure the stability and robustness of the interpolation and energy quadrature, while the powerful neural network solvers simplify the solving process. In this paper, therefore, we implement this idea and propose a robust RPIM empowered by neural network solvers (RPIM-NNS) for highly nonlinear solid mechanics. The proposed RPIM-NNS integrates powerful neural network solvers with mechanics governing equations and techniques of traditional computational mechanics. It has the following features:

(1) The radial basis functions (RBFs) are applied to interpolate displacement fields by arbitrary nodes in the computational domain. Thereby, the RPIM-NNS is robust even for irregularly distributed nodes. Furthermore, it is straightforward to calculate the derivatives, leading to seamless implementation with neural network solvers.
(2) The RPIM-NNS allows nodes to be placed at the out-domain area near the boundary. Therefore, the RPIM-NNS can achieve excellent accuracy for implementing boundary conditions of both Dirichlet and Neumann types.
(3) Only a functional (loss function) that includes integral potential energy and essential boundary conditions is required as the solving target. Therefore, the RPIM-NNS is simple to implement. Moreover, the use of the weak form ensures the stability and convergence of solutions.
(4) Well-developed gradient descendant optimisers are harnessed to solve problems via minimising the functional loss. Empowered by the optimisers, the RPIM-NNS can easily handle the nonlinearities arising from material and large deformation.
(5) The proposed RPIM-NNS is compatible with parallel computing schemes, and thus is computationally efficient for large-scale computing scenarios.

These features are well demonstrated by intensive numerical examples, including 2D Cook's membrane and 3D twisting rubber problems. The good results indicate the effectiveness and the great potential of the proposed RPIM-NNS for nonlinear solid mechanics modelling.

The remainder of the paper is organised as follows: In Section 2, a brief introduction regarding the RPIM is presented. In Section 3, the RPIM-NNS is proposed in detail. In Section 4, nonlinear mechanics examples are conducted to examine the performance of the proposed RPIM-NNS. In Section 5, the conclusions of this work and future perspectives are summarised.

2. Radial point interpolation method

In the RPIM, the displacement solution of a mechanics problem is constructed by using multiple RBFs $R(\mathbf{x})$ and polynomial basis functions $P(\mathbf{x})$ inside the computational domain [6,14]

$$\begin{aligned} u(\mathbf{x}) &= \sum_{i}^{n} R_i(\mathbf{x}) a_i + \sum_{j}^{m} P_j(\mathbf{x}) d_j \\ &= \mathbf{R}^{\mathrm{T}}(\mathbf{x})\mathbf{a} + \mathbf{P}^{\mathrm{T}}(\mathbf{x})\mathbf{d}, \end{aligned} \tag{1}$$

where a and d are the unknown vectors that remain to be solved, n is the number of RBFs, and m is the number of polynomial basis functions. The polynomial terms are not always necessary. Hence, for sake of simplicity, pure RBFs are used in this work

$$u(\mathbf{x}) = \sum_{i}^{n} R_i(\mathbf{x}) a_i = \mathbf{R}^{\mathrm{T}}(\mathbf{x})\mathbf{a}. \tag{2}$$

The RBF is expressed as

$$R_i(\mathbf{x}) = R(\| \mathbf{x} - \mathbf{c}_i \|) = R(r_i), \tag{3}$$

where c is the centre of the RBF. In this work, the Gaussian type RBF and the fifth-order piecewise function proposed by Morris [35] are used. The Gaussian type RBF is given as

$$R(\mathbf{x}) = e^{-\chi_i \|\mathbf{x}-\mathbf{c}_i\|^2} = e^{-\chi_i r_i^2}, \tag{4}$$

where χ is the positive shape parameter that controls the shape of the RBF [36,37]. The fifth-order piecewise function is written as

$$R(\mathbf{x}) = 0.01 \times \begin{cases} \left(3 - \frac{3}{2h}r\right)^5 - 6\left(2 - \frac{3}{2h}r\right)^5 + 15\left(1 - \frac{3}{2h}r\right)^5 & \left(r < \frac{2}{3}h\right) \\ \left(3 - \frac{3}{2h}r\right)^5 - 6\left(2 - \frac{3}{2h}r\right)^5 & \left(\frac{2}{3}h \le r < \frac{4}{3}h\right) \\ \left(3 - \frac{3}{2h}r\right)^5 & \left(\frac{4}{3}h \le r < 2h\right) \\ 0 & (r \ge 2h) \end{cases} \tag{5}$$

where h is the support domain length.

To solve the unknown vector $\mathbf{a}$, nodes are enforced into the constructed displacement field within the computational domain

$$\mathbf{u}^{\mathrm{e}} = \mathbf{R}_{\mathrm{c}}\mathbf{a}, \tag{6}$$

where $\mathbf{u}^{\mathrm{e}} = [u_1 \quad u_2 \quad \cdots \quad u_n]^{\mathrm{T}}$ and $\mathbf{R}_{\mathrm{c}}$ is the moment matrix expressed as

$$\mathbf{R}_{\mathrm{c}} = \begin{bmatrix} R_1(\mathbf{x}_1) & R_1(\mathbf{x}_2) & \cdots & R_1(\mathbf{x}_n) \\ R_2(\mathbf{x}_1) & R_2(\mathbf{x}_2) & \cdots & R_2(\mathbf{x}_n) \\ \vdots & \vdots & & \vdots \\ R_n(\mathbf{x}_1) & R_n(\mathbf{x}_2) & \cdots & R_n(\mathbf{x}_n) \end{bmatrix}. \tag{7}$$

Thus, the unknown vector a can be calculated by

$$\mathbf{a} = \mathbf{R}_{\mathrm{c}}^{-1}\mathbf{u}^{\mathrm{e}}. \tag{8}$$

By substituting Eq. (8) into (1), the shape function of the RPIM can be obtained as

$$\begin{aligned} u(\mathbf{x}) &= \mathbf{R}^{\mathrm{T}}(\mathbf{x})\mathbf{R}_{\mathrm{c}}^{-1}\mathbf{U}_{\mathrm{s}} \\ &= \mathbf{\Phi}^{\mathrm{T}}(\mathbf{x})\mathbf{U}_{\mathrm{s}}. \end{aligned} \tag{9}$$

It has been proved that, for both the infinitely smooth RBFs and the compactly supported RBFs, the inverse of the moment matrix $\mathbf{R}_{\mathrm{c}}$ exists for arbitrarily distributed nodes [38]. The derivatives of the displacement can be calculated by

$$\begin{aligned} \nabla u(\mathbf{x}) &= \nabla\mathbf{\Phi}^{\mathrm{T}}(\mathbf{x})\mathbf{U}_{\mathrm{s}}, \\ \nabla^2 u(\mathbf{x}) &= \nabla^2\mathbf{\Phi}^{\mathrm{T}}(\mathbf{x})\mathbf{U}_{\mathrm{s}}. \end{aligned} \tag{10}$$

3. RPIM empowered with neural network solvers

3.1. Constructing displacement field

The RPIM-NNS starts from constructing the displacement field. In this part, all the formulations are derived in 2D scenarios. They can be easily extended to 3D cases and degraded to 1D scenarios. Instead of using the constructed shape function presented as Eq. (6), Eq. (2) is directly used in the proposed RPIM-NNS. In this manner, the displacements u and v can be written as

$$\begin{aligned} u &= \sum_{i}^{n} R_i(\mathbf{x}) a_i = \mathbf{R}^{\mathrm{T}}(\mathbf{x})\mathbf{a}, \\ v &= \sum_{i}^{n} R_i(\mathbf{x}) b_i = \mathbf{R}^{\mathrm{T}}(\mathbf{x})\mathbf{b}, \end{aligned} \tag{11}$$

where $\mathbf{a}$ and $\mathbf{b}$ are unknown vectors to be solved. The Einstein summation convention is not applied in the following contents. For

simplicity, we let the i^{th} RBF at the j^{th} sample point as

$$R_i^j = R_i(\mathbf{x}^j). \tag{12}$$

To better distinguish the indexes for RBF centres and sample points, we note that the subscript i denotes the i^{th} RBF centre and the superscript j denotes the j^{th} sample point. The sample points are used to numerically integral the strain energy in the mechanics system. An example of the constructed displacement field is shown in Fig. 1. It is worth highlighting that in the RPIM-NNS, nodes are not only placed at the in-domain area, but also initialised at the out-domain area, which is distinguished to the original RPIM. These out-domain nodes can bring more flexibilities and a higher accuracy for imposing boundary conditions.

The spatial derivatives of the constructed displacement field can be straightforwardly obtained by

$$\begin{aligned} \nabla u(\mathbf{x}) &= \sum_i^n a_i \nabla R_i(\mathbf{x}), \\ \nabla v(\mathbf{x}) &= \sum_i^n b_i \nabla R_i(\mathbf{x}), \end{aligned} \tag{13}$$

Using Eq. (13), the strains and stresses can be straightforwardly derived. The infinitesimal strain under small deformation can be calculated as

$$\begin{aligned} \varepsilon_{xx}^j &= \frac{\partial u^j}{\partial x} = \sum_{i=1}^n a_i \frac{\partial}{\partial x} R_i^j, \\ \varepsilon_{yy}^j &= \frac{\partial v^j}{\partial y} = \sum_{i=1}^n b_i \frac{\partial}{\partial y} R_i^j, \\ \varepsilon_{xy}^j &= \frac{1}{2}\left(\frac{\partial v^j}{\partial x} + \frac{\partial u^j}{\partial y}\right) = \frac{1}{2}\left(\sum_{i=1}^n b_i \frac{\partial}{\partial x} R_i^j + \sum_{i=1}^n a_i \frac{\partial}{\partial y} R_i^j\right). \end{aligned} \tag{14}$$

For linear elastic materials, the stresses can be calculated by

$$\begin{aligned} \sigma_{xx}^j &= (\lambda + 2\mu)\varepsilon_{xx}^j + \lambda \varepsilon_{yy}^j \\ &= (\lambda + 2\mu) \sum_{i=1}^{n_i} a_i \frac{\partial}{\partial x} R_i^j + \lambda \sum_{i=1}^{n_i} b_i \frac{\partial}{\partial y} R_i^j, \\ \sigma_{yy}^j &= (\lambda + 2\mu)\varepsilon_{yy}^j + \lambda \varepsilon_{xx}^j \\ &= (\lambda + 2\mu) \sum_{i=1}^{n_i} b_i \frac{\partial}{\partial y} R_i^j + \lambda \sum_{i=1}^{n_i} a_i \frac{\partial}{\partial x} R_i^j, \\ \tau_{xy}^j &= 2\mu \varepsilon_{xy}^j = \mu\left(\sum_{i=1}^{n_i} b_i \frac{\partial}{\partial x} R_i^j + \sum_{i=1}^{n_i} a_i \frac{\partial}{\partial y} R_i^j\right), \end{aligned} \tag{15}$$

where λ and μ are the Lame constants. For large deformation scenarios, the total Lagrangian formulation is applied, and the deformation gradient tensor $\mathbf{F}$ is given by

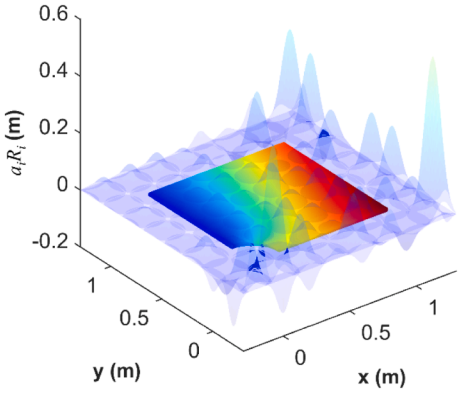


Fig. 1. An example of constructed displacement field by the proposed RPIM-NNS. Note that the nodes in the RPIM-NNS is also placed out of the computational domain, which can more accurately achieve boundary conditions compared to the original RPIM.

$$\begin{aligned}\mathbf{F}^j &= \begin{bmatrix} 1+u_x{}^j & u_y{}^j \\ v_x{}^j & 1+v_y{}^j \end{bmatrix} \\ &= \begin{bmatrix} 1+\sum_{i=1}^{n} a_i \frac{\partial}{\partial x} R_i^j & \sum_{i=1}^{n} a_i \frac{\partial}{\partial y} R_i^j \\ \sum_{i=1}^{n} b_i \frac{\partial}{\partial x} R_i^j & 1+\sum_{i=1}^{n} b_i \frac{\partial}{\partial y} R_i^j \end{bmatrix},\end{aligned} \tag{16}$$

where $\mathbf{x}$ denotes the coordinate of initial(reference) configuration. Thus, the right Cauchy-Green deformation tensor C can be computed by

$$\mathbf{C}^j = \left(\mathbf{F}^j\right)^T \mathbf{F}^j. \tag{17}$$

For large deformation problems, the neo-Hookean constitutive model is considered. In this manner, the strain energy density can be calculated by [39]

$$\Psi^j = \frac{1}{2}\lambda\left[\ln\left(J^j\right)\right]^2 - \mu\ln\left(J^j\right) + \frac{1}{2}\mu\left(I^j - 2\right), \tag{18}$$

where J and I are given by

$$\begin{aligned} J^j &= \det\left(\mathbf{F}^j\right), \\ I^j &= \mathrm{tr}\left(\mathbf{C}^j\right). \end{aligned} \tag{19}$$

It is worth highlighting that, instead of using the conjugated pair of strains and stresses, the strain energy density for large deformation problems is calculated by the invariant of the deformation gradient tensor.

3.2. Neighbour sample points searching

If local RBFs are selected, a neighbour sample points searching process is necessary. An example is shown in Fig. 2. In this work, we apply the direct searching algorithm [40], which is easy to implement but computationally expensive. It is noted that more advanced searching algorithms have been developed to speed up the searching process, such as the link-list algorithm [41], the tree searching algorithm [42], and the hashing algorithm [8], to name but a few. More importantly, some of them are compatible with GPU, which can further improve computational efficiency.

3.3. Formulating functional loss via the weak form

To solve the problem using neural network solvers, a functional loss is necessary as the training target. Considering that computational methods based on the weak form are generally more stable than the strong form methods [43], the weak form formulation is applied in the functional loss. The use of the energy form governing equations can ensure the stability and convergence of the RPIM-NNS. According to the principle of minimum potential energy, the solution of the mechanics problem can be obtained by finding the displacement functions that minimise the overall potential energy functional

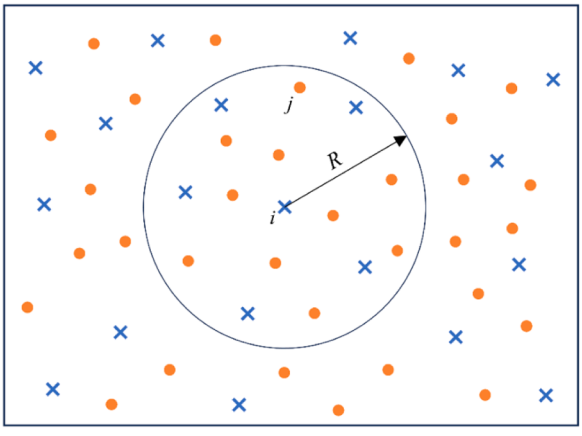


Fig. 2. An example of neighbour sample point searching. The blue crosses denote the centres of RBFs and the circle denotes the influence area of the corresponding RBF. The orange dots denote the sample points. The RBF only has impact to the sample points inside its influence area.

$$\underset{u(\mathbf{x})}{\operatorname{argmin}}\Pi, \tag{20}$$

where Π is the overall potential energy integrated over the entire problem domain, and can be calculated via two parts:

$$\Pi = E_{\text{in}} - E_{\text{ex}}, \tag{21}$$

where E_{in} and E_{ex} are the strain energy and the potential energy of external forces, respectively. The strain energy can be obtained via integrating the strain energy density as

$$E_{\text{in}} = \int_{\Omega_0} \psi \mathrm{d}V, \tag{22}$$

where Ω_0 denotes the initial(reference) configuration and ψ is the strain energy density. The potential energy of external forces can be obtained by

$$E_{\text{ex}} = \int_{\Gamma_0} \mathbf{u}^{\mathrm{T}} \mathbf{f}_0 \mathrm{d}S, \tag{23}$$

where Γ_0 denotes the natural boundary at the initial configuration, $\mathbf{u} = [u, v]^{\mathrm{T}}$ and $\mathbf{f}_0 = \left[f_x, f_x\right]^{\mathrm{T}}$ are the displacement vector and the prescribed traction force vector at the initial configuration, respectively.

3.4. Imposing essential boundary conditions

As shown in Eq. (23), natural (traction) boundary conditions are already embedded when calculating the overall potential energy of the mechanics system. In this case, only essential (displacement) boundary conditions are required to be additionally imposed. To this end, we apply the penalty function method to the potential energy functional. Therefore, a modified functional loss can be written as

$$\mathscr{L} = \kappa \mathscr{L}_{\text{BC}} + \Pi, \tag{24}$$

where κ is the manually determined penalty value, $\mathscr{L}_{\text{BC}}$ denotes the residuals from the essential boundary and can be calculated by

$$\mathscr{L}_{\text{BC}} = \frac{1}{n_{\text{e}}} \sum_{j}^{n_{\text{e}}} \parallel \mathbf{u}\left(\mathbf{x}^j\right) - \overline{\mathbf{u}}\left(\mathbf{x}^j\right) \parallel, \tag{25}$$

where n_{e} is the number of sample points on the essential boundary and $\overline{\mathbf{u}}$ is the ground truth displacement on the natural boundary.

3.5. Solving the problem by neural network solvers

To solve the problem, the well-developed neural network solvers [44] can be used to find the solution of the unknown vectors that minimise the functional loss shown in Eq. (24). In this study, we implement the Adam optimiser, which is the most popular solver in neural network training. The Adam optimiser is a stochastic gradient descent algorithm that searches a local minimum of target differentiable functions. Its basic idea is to repetitively modify the trainable parameters via the direction of steepest descent; that is, in the opposite direction of the gradient of the target functions [45]. Besides, the moment method is combined to overcome the low convergence issue due to the appearance of zig-zag training pathologies [34]. Apart from neural network training, the Adam algorithm has been widely applied for optimising complex nonlinear systems [18]. It is also worth noting that other neural network solvers, such as the quasi-Newton methods, the Levenberg-Marquardt algorithm and the conjugated gradient methods, can also be used.

Before training, we initialise **a** and **b** to be 0. By doing so, the initialised displacement fields denote the undeformed state. Because RPIM is used, the gradient of the final loss with respect to a and b can be calculated in the following explicit forms:

$$\begin{aligned} \frac{\partial}{\partial \mathbf{a}} \mathscr{L} &= \kappa \frac{\partial}{\partial \mathbf{a}} \mathscr{L}_{\text{BC}} + \frac{\partial}{\partial \mathbf{a}} \Pi, \\ \frac{\partial}{\partial \mathbf{b}} \mathscr{L} &= \kappa \frac{\partial}{\partial \mathbf{b}} \mathscr{L}_{\text{BC}} + \frac{\partial}{\partial \mathbf{b}} \Pi. \end{aligned} \tag{26}$$

In Eq. (26), the gradient of essential boundary condition is given as

$$\begin{aligned}\frac{\partial}{\partial a_i}\mathscr{L}_{\mathrm{BC}} &= \frac{1}{n_{\mathrm{e}}}\frac{\partial}{\partial a_i}\sum_{j=1}^{n_{\mathrm{e}}} \parallel \mathbf{u}(\mathbf{x}^j) - \overline{\mathbf{u}}(\mathbf{x}^j) \parallel \\ &= \frac{2}{n_{\mathrm{e}}}\sum_{j=1}^{n_{\mathrm{e}}} (u^j - \overline{u}^j) R_i^j, \\ \frac{\partial}{\partial b_i}\mathscr{L}_{\mathrm{BC}} &= \frac{1}{n_{\mathrm{e}}}\frac{\partial}{\partial b_i}\sum_{j=1}^{n_{\mathrm{e}}} \parallel \mathbf{u}(\mathbf{x}^j) - \overline{\mathbf{u}}(\mathbf{x}^j) \parallel \\ &= \frac{2}{n_{\mathrm{e}}}\sum_{j=1}^{n_{\mathrm{e}}} (v^j - \overline{v}^j) R_i^j,\end{aligned} \tag{27}$$

where n_{e} is the number of sample points on the essential boundary.

The gradient of the natural boundary condition is given as

$$\begin{aligned}\frac{\partial}{\partial a_i}E_{\mathrm{ex}} &= \frac{\partial}{\partial a_i}\int_{\Gamma_0} \mathbf{u}^{\mathrm{T}}\mathbf{f}_0 \mathrm{d}S_0 \\ &= \sum_{j=1}^{n_{\mathrm{n}}} l^j f_x{}^j R_i^j, \\ \frac{\partial}{\partial b_i}E_{\mathrm{ex}} &= \frac{\partial}{\partial b_i}\int_{\Gamma_0} \mathbf{u}^{\mathrm{T}}\mathbf{f}_0 \mathrm{d}S_0 \\ &= \sum_{j=1}^{n_{\mathrm{n}}} l^j f_y{}^j R_i^j,\end{aligned} \tag{28}$$

where n_{n} is the number of sample points on the natural boundary, and l^j is the weight for the sample point $\mathbf{x}^j$ according to the numerical quadrature method.

In addition, the gradient of the strain energy density with respect to a_i under small deformation scenarios can be obtained as

$$\begin{aligned}\frac{\partial}{\partial a_i}\Psi^j &= \frac{1}{2}\frac{\partial}{\partial a_i}\left(\sigma_{xx}^j \varepsilon_{xx}^j + \sigma_{yy}^j \varepsilon_{yy}^j + \tau_{xy}^j \gamma_{xy}^j\right) \\ &= \frac{1}{2}\left\{\left[\sigma_{xx}^j + (2\mu+\lambda)\varepsilon_{xx}^j + \lambda\varepsilon_{yy}^j\right]\frac{\partial}{\partial x}R_i^j + 2\mu\gamma_{xy}^j\frac{\partial}{\partial y}R_i^j\right\}.\end{aligned} \tag{29}$$

Similarly, we obtain the gradient of the elastic strain energy density with respect to b_i

$$\frac{\partial}{\partial b_i}\Psi^j = \frac{1}{2}\left\{\left[\lambda\varepsilon_{xx}^j + \sigma_{yy}^j + (2\mu+\lambda)\varepsilon_{yy}^j\right]\frac{\partial}{\partial y}R_i^j + 2\mu\gamma_{xy}^j\frac{\partial}{\partial x}R_i^j\right\}. \tag{30}$$

For neo-Hookean materials under large deformation, the gradient of the strain energy density can be computed by

$$\begin{aligned}\frac{\partial}{\partial a_i}\Psi^j &= \frac{\partial}{\partial a_i}\left(\frac{1}{2}\lambda\left[\ln(J^j)\right]^2 - \mu\ln(J^j) + \frac{1}{2}\mu(I^j - 2)\right) \\ &= \frac{\lambda\ln(J^j) - \mu}{J^j}\frac{\partial J^j}{\partial a_i} + \frac{\mu}{2}\frac{\partial I^j}{\partial a_i},\end{aligned} \tag{31}$$

where

$$\begin{aligned}\frac{\partial J^j}{\partial a_i} &= (1+v_y{}^j)\frac{\partial}{\partial x}R_i^j - v_x{}^j\frac{\partial}{\partial y}R_i^j \\ &= \left(1 + \sum_{i=1}^{n} b_i\frac{\partial}{\partial y}R_i^j\right)\frac{\partial}{\partial x}R_i^j - \left(\sum_{i=1}^{n} b_i\frac{\partial}{\partial x}R_i^j\right)\frac{\partial}{\partial y}R_i^j, \\ \frac{\partial I^j}{\partial a_i} &= 2(1+u_x{}^j)\frac{\partial}{\partial x}R_i^j + 2u_y{}^j\frac{\partial}{\partial y}R_i^j \\ &= 2\left(1 + \sum_{i=1}^{n} a_i\frac{\partial}{\partial x}R_i^j\right)\frac{\partial}{\partial x}R_i^j + \left(\sum_{i=1}^{n} a_i\frac{\partial}{\partial y}\vartheta_i^j\right)\frac{\partial}{\partial y}R_i^j.\end{aligned} \tag{32}$$

Analogously, we can obtain the gradient information of the strain energy density with respect to b_i as

$$\frac{\partial}{\partial b_i}\Psi^j = \frac{\lambda \ln(J^j) - \mu}{J^j}\frac{\partial J^j}{\partial b_i} + \frac{\mu}{2}\frac{\partial I^j}{\partial b_i}, \tag{33}$$

where

$$\begin{aligned}
\frac{\partial J^j}{\partial b_i} &= \left(1 + \sum_{i=1}^{n} a_i \frac{\partial}{\partial x}\vartheta_i^j\right)\frac{\partial}{\partial x}R_i^j - \left(\sum_{i=1}^{n} a_i \frac{\partial}{\partial y}\vartheta_i^j\right)\frac{\partial}{\partial y}R_i^j, \\
\frac{\partial I^j}{\partial b_i} &= 2\left(1 + \sum_{i=1}^{n} b_i \frac{\partial}{\partial y}R_i^j\right)\frac{\partial}{\partial x}R_i^j + 2\left(\sum_{i=1}^{n} b_i \frac{\partial}{\partial x}R_i^j\right)\frac{\partial}{\partial y}R_i^j.
\end{aligned} \tag{34}$$

Therefore, by summarising all the gradients terms and substituting them into Eq. (26), the gradient of the final loss with respect to a_i and b_i can be calculated as

$$\begin{aligned}
\frac{\partial}{\partial a_i}\mathscr{L} &= \frac{\partial}{\partial a_i}\Pi + \kappa \sum_{j=1}^{n_e} \frac{\partial}{\partial a_i}\mathscr{L}_{\mathrm{BC}}{}^j \\
&= \sum_{j=1}^{n_j} s^j \frac{\partial}{\partial a_i}\Psi^j - \sum_{j=1}^{n_n} l^j \overline{F}_x{}^j R_i^j + \frac{2\kappa}{n_e}\sum_{j=1}^{n_e}(u^j - \overline{u}^j)R_i^j, \\
\frac{\partial}{\partial b_i}\mathscr{L} &= \frac{\partial}{\partial b_i}\Pi + \kappa \sum_{j=1}^{n_e} \frac{\partial}{\partial b_i}\mathscr{L}_{\mathrm{BC}}{}^j \\
&= \sum_{j=1}^{n_j} s^j \frac{\partial}{\partial b_i}\Psi^j - \sum_{j=1}^{n_n} l^j \overline{F}_y{}^j R_i^j + \frac{2\kappa}{n_e}\sum_{j=1}^{n_e}(v^j - \overline{v}^j)R_i^j,
\end{aligned} \tag{35}$$

where s and l are the weights for the in-domain and boundary integrations, κ is the penalty value for imposing essential boundary conditions, and n_j is the number of in-domain sample points.

Eventually, the gradients of the energy loss are fed into the Adam solver, and the solver updates the a and b for the next iteration. The flowchart of the RPIM-NNS is summarised in Table 1. It is also worth noting that, since the strain energy density is obtained by the invariant of the deformation gradient tensor, the conjugated strain and stress pair are not needed. Therefore, the solving process is done without using increment loading process, where the conjugated strain and stress require to be updated at every increment.

To test the performance of the proposed RPIM-NNS, a perforated plate problem is considered. The configuration is shown in Fig. 3, where both the uniformly and arbitrarily distributed RBFs are used. Besides, the out-domain RBFs are placed to enhance the accuracy of the solution near the boundaries. The penalty value for the essential boundary condition is $\kappa = 1 \times 10^2$. The Adam optimiser is applied with a learning rate of 1×10^{-3}. Both the global (Gaussian) and local (5^{th} order spline) RBFs are applied in RPIM-NNS. The shape parameter for the global RBF is $\chi = 0.1333$ and the support domain length for the local RBF is $h = 0.2$ m. The code for this problem is built up with MATLAB. Fig. 4(a)–(e) shows the displacement results obtained by the RPIM-NNS and FEM. Both the RPIM-NNS with global (Gaussian) and local (5^{th} order spline) RBFs agree well with the reference results. Furthermore, the results also demonstrate the stability of the RPIM-NNS with highly arbitrary RBF distributions. Fig. 4(f) depicts the history plots of the potential energy of the plate by using the RPIM-NNS. It is worth highlighting that the training of all the cases using RPIM-NNS can stably converge, although the RPIM-NNS with uniformly distributed RBFs can converge faster than it with arbitrarily distributed RBFs. Moreover, despite both the global and local RPIM-NNS can effectively solve the problem, their computational efficiencies are significantly different. Table 2 lists the CPU time used by the global and the local RPIM-NNS with uniform or arbitrary RBF

Table 1
The flowchart of the RPIM-NNS. Note that n_j is the number of in-domain sample points.

Radial point interpolation method with neural network implementation (RPIM-NNS)
• Neighbour sample points searching:
$i = 1, 2, \ldots, n$ do
for $j = 1, 2, \ldots, n_j$ do
Calculate $R_i(\mathbf{x}^j)$ and $\nabla R_i(\mathbf{x}^j)$;
end for
end for
• Start training:
for $iter = 1, \ldots, max\_iter$ do
Obtain the gradient of essential boundary condition with respect to a: $\kappa\partial\mathscr{L}_{\mathrm{BC}}/\partial\mathbf{a}$;
Obtain the gradient of natural boundary condition with respect to a: $\partial E_{\mathrm{ex}}/\partial\mathbf{a}$;
Calculate the gradient of strain energy density with respect to a: $\partial\psi/\partial\mathbf{a}$;
Assemble the gradient of final loss: $\partial\mathscr{L}/\partial\mathbf{a}$;
Feed the gradient of final loss to the Adam optimiser;
Update a by the Adam optimiser;
end for

distributions. As expected, the computational efficiency of the local RPIM-NNS is better than the global RPIM. The computational expense of the global RPIM-NNS will suffer exponential growth with increasing numbers of RBF centres and sample points. That makes the global RPIM-NNS computationally expensive or even prohibited when dealing with large problems. Therefore, the local RPIM-NNS is suggested rather than the global RPIM-NNS when using large numbers of RBF and sample points.

3.6. Parallel computing

It is also worth highlighting that the proposed RPIM-NNS can be accelerated by parallel computing. The parallel schemes can be implemented in the neighbour sample points searching process as well as in calculating the gradient information. In the neighbour sample points searching process, the RBF centres can be assigned to different CPU threads. Each thread searches the neighbouring sample points of the assigned RBF centres. In this manner, with the increasing number of threads, the overall CPU time for searching can be effectively decreased.

Apart from searching, the gradient information can be obtained through parallel computing. As the equations derived above, the gradient information for each trainable parameter is the sum of RBFs' values from the surrounding sample points. With the neighbouring information obtained by searching, updating the gradient information for each trainable parameter can be assigned to different CPU threads. With an increasing number of CPU threads, a decreasing number of trainable parameters is assigned for each CPU thread, where less CPU time is required for one gradient information updating process. Eventually, the overall gradient information can be assembled and fed into the Adam optimiser for training.

Herein, we implement the parallel computing scheme for solving a 2D cantilever beam problem with linear elastic material under the small deformation assumption. The configuration of the problem is given in Fig. 5(a). A parabolic force $P = 1$ N is applied on the right boundary of the beam. The point A is located at the centre of the right boundary of the beam. The local RPIM-NNS with 5^{th} order spline RBF is used. Five different sets of sample points and RBF centres are tested, where the spacing of the RBF centres are respectively $dx_c = 6$ m, 3 m, 1.5 m, 0.75 m and 0.375 m, as shown in Fig. 5(b)–(f). Based on our numerical experiments, the support domain length is suggested to take 2 or 3 times the spacing of the RBF centres. Therefore, the corresponding support domain lengths are respectively $h = 12$ m, 6 m, 3 m, 1.5 m and 0.75 m. The Young's module $E = 300$ Pa and the Poisson's ratio $\nu = 0.3$. The penalty value for the essential boundary condition is $\kappa = 1 \times 10^4$ and the learning rate is 1×10^{-4}. Referring to the solution given by Timoshenko [46], the deflection in the y direction at the point A, V_A, is 0.89 m.

Table 3 lists the CPU time and the deflection V_A from RPIM-NNS by using single thread and eight-thread parallel computing schemes. Fig. 6 shows the overall CPU time vs. RBF spacings plots and the history plots of the potential energy of the beam by using different RBF centre spacings. Fig. 7 shows the displacement and absolute error contours from the RPIM-NNS with RBF spacing $dx_c = 0.375$ m. As observed, with the same discretisation, the deflection V_A from both schemes agree well with each other, suggesting the effectiveness of the parallel computing scheme. Besides, when dealing with large numbers of RBF centres and sample points, the RPIM-NNS with an eight-thread parallel computing scheme is computationally more efficient than the RPIM-NNS with single thread. However, the parallel computing scheme costs more CPU time when the RBF spacing $dx_c = 6$ m. This is because passing data through threads also requires CPU time. For cases with small numbers of RBF centres and sample points, the CPU time spent for data transmission between threads is more than the calculation. Therefore, a parallel computing scheme is not recommended for simple cases with small numbers of RBF centres and sample points. Meanwhile, the deflection V_A converges to the reference solution with the decreasing RBF spacing. Furthermore, according to the history plots of the potential energy, all the cases stably converge without suffering energy drop. Nevertheless, the converged potential energy obtained by $dx_c = 6$ m and 3 m are higher than the other cases.

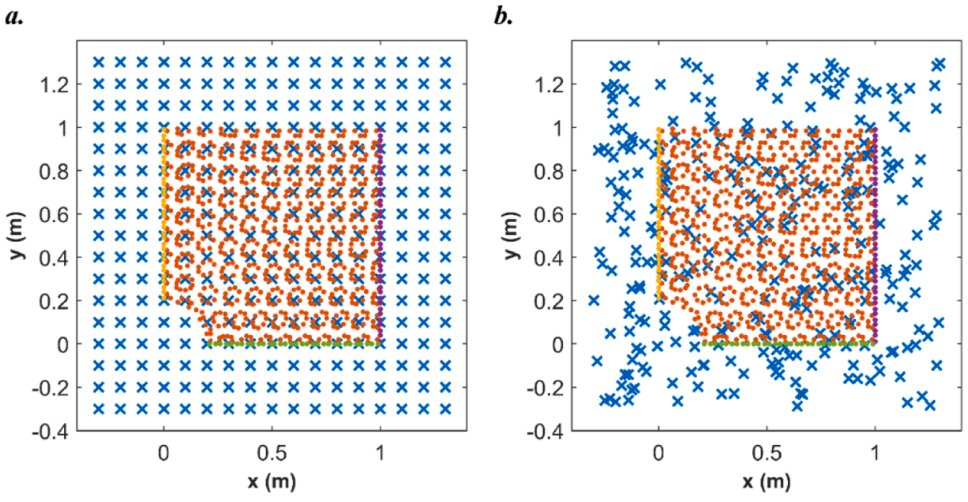
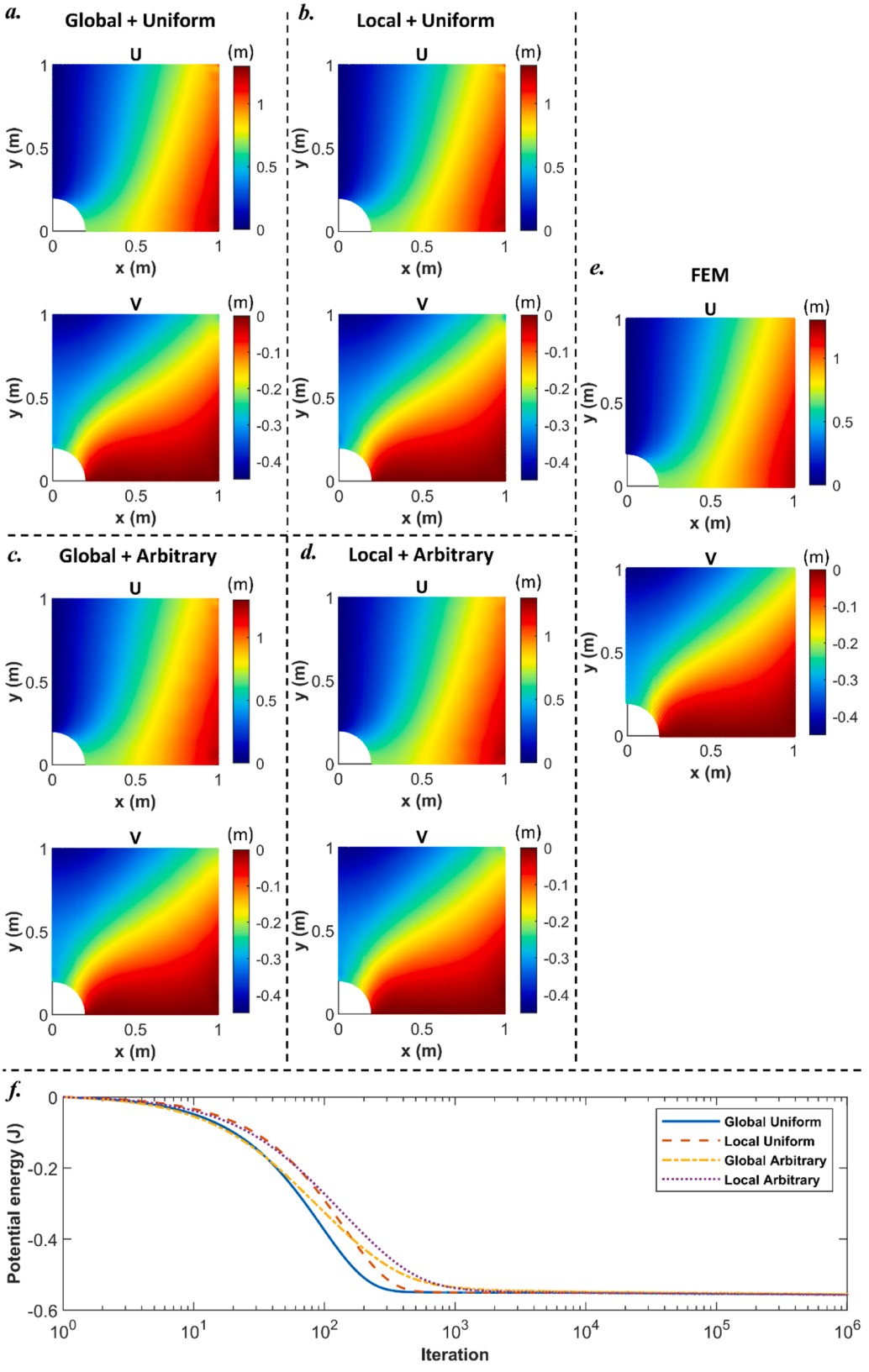


Fig. 3. Configuration of the 720 sample points and 289 RBF centres in the computational domain for the RPIM-NNS. (a) Uniformly distributed RBFs. (b) Arbitrarily distributed RBFs.



(caption on next page)

Fig. 4. Displacement results obtained with (a) uniform distributed global (Gaussian) RBFs, (b) uniform distributed local (5th order spline) RBFs, (c) arbitrary distributed global (Gaussian) RBFs, (d) arbitrary distributed local (5th order spline) RBFs and (e) FEM. (f) The history plots of the potential energy of the plate from the RPIM-NNS.

Table 2
Comparisons of CPU time and mean absolute error between the RPIM-NNS using a global (Gaussian) RBF and a local (5^{th} order spline) RBF. The code is built up with MATLAB.

	Uniform RBF distribution		Arbitrary RBF distribution	
	Global	Local	Global	Local
CPU time (s)	3667.03	697.147	3897.23	728.748

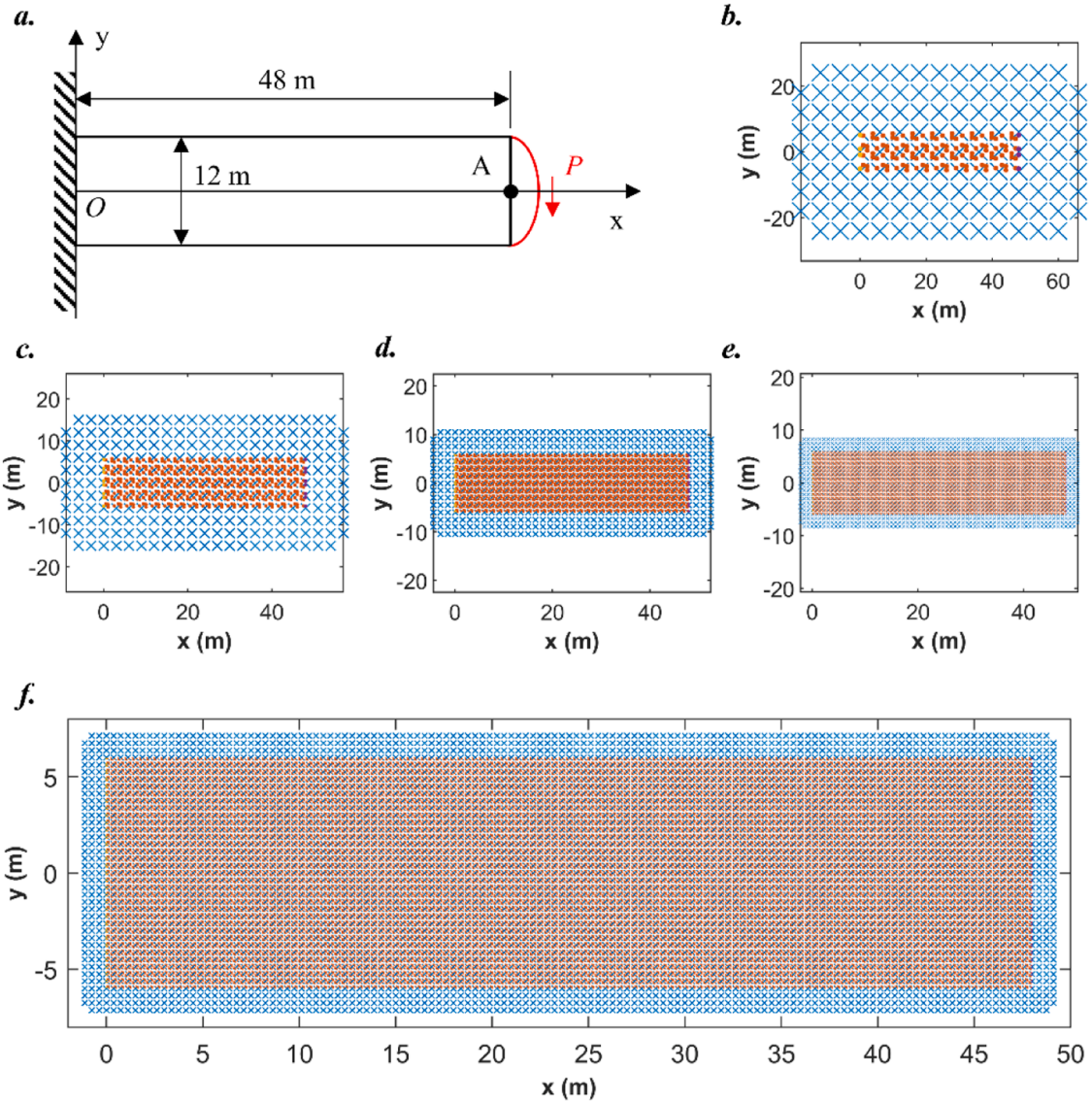


Fig. 5. (a) Configuration of the cantilever beam problem. Different discretisation for the cantilever beam problem, where the spacing of the RBF centres are (b) $dx_c = 6$ m; (c) $dx_c = 3$ m; (d) $dx_c = 1.5$ m; (e) $dx_c = 0.75$ m; (f) $dx_c = 0.375$ m. Note that the blue crosses represent the centres of the radial basis functions, the orange, yellow and purple dots represent the in-domain sample points, the left-boundary sample points and the right-boundary sample points, respectively.

Hence, the deflections V_A from those two cases have relatively larger errors compared to the reference result.

3.7. Comparison between the RPIM-NNS and the original RPIM

Compared to the original RPIM, the proposed RPIM-NNS can place nodes at the out-domain area. In the original RPIM, since nodes represent discretised substance, they are only distributed at the in-domain area. By applying the out-domain nodes, the proposed RPIM-NNS can enjoy more flexibility to construct the displacement field. In this manner, it can more accurately impose boundary conditions than the original RPIM, especially the natural boundary conditions. Herein, the perforated plate problem is revisited as an example. Fig. 8 shows convergence plots of the displacement U at points A (1, 0) and B (1, 1), respectively. As observed, the displacement results of the traditional RPIM can be gradually improved by adopting an increasing h. When $h \leq 0.2$, only insufficient surrounding nodes are within the support domain of the boundary nodes, resulting in poor displacement solutions from the traditional RPIM. On the contrary, the displacements at both points A and B from the proposed RPIM-NNS are very stable with respect to different h. The use of the out-domain nodes in the RPIM-NNS offers the boundary nodes additional RBF basis functions, which greatly improves the approximating ability near the problem boundaries. This can be further proved by the displacement contours from both the traditional RPIM and the RPIM-NNS, as shown in Fig. 9(a)–(d). When $h = 0.2$, the displacement results obtained from the traditional RPIM exhibit relatively larger errors than results from the RPIM-NNS. We also highlight that the computational time of the proposed RPIM-NNS will increase with an increasing h. Therefore, under the consideration of computational accuracy and efficiency, taking 2 or 3 times the average node spacing for h is suggested.

Moreover, the target equations of the RPIM-NNS and the original RPIM are different. The original RPIM seeks for the stational point of the energy functional, while the RPIM-NNS directly applies optimisers to find the minimum value of overall potential energy. Therefore, during the numerical implementation, the original RPIM can eventually build up a linear system

$$\mathbf{K}\mathbf{u}_n = \mathbf{f}, \tag{36}$$

where $\mathbf{K}$, $\mathbf{u}_n$, and $\mathbf{f}$ are the stiffness matrix, nodal displacement vector, and force vector, respectively. As for the RPIM-NNS, the solving process only relies on iterative neural network solver algorithms. It is obvious that, for linear problems, the original RPIM are more effective and efficient. However, for nonlinear problems, the RPIM-NNS can be more effective and simpler to be implemented. This is because additional algorithms for dealing with material nonlinearity and geometrical nonlinearity are required when using traditional computational mechanics methods. As for the RPIM-NNS, those additional algorithms are no longer required, and treatments accounting for nonlinearities are all integrated into a gradient descendent process. Detailed examples of nonlinear mechanics problems solved by the RPIM-NNS are presented in Section 4.

4. Numerical examples

In this section, two nonlinear problems are conducted to test the performance of the RPIM-NNS. In those examples, the 5^{th} order spline RBF is applied. All the results are produced with a 12th Gen Intel(R) Core(TM) i5–12490F CPU (3.0 GHz). Furthermore, the code is written by C++ and the parallel computing is applied with eight threads.

4.1. Cook's membrane

We test the performance of the proposed RPIM-NNS by the well-known 2D Cook's membrane problem. The configuration of this problem is given in Fig. 10(a). The point A is located at the top-right of the membrane. Five different sets of sample points and RBF centres are tested, where the spacing of the RBF centres are respectively $dx_c = 6$ m, 3 m, 1.5 m, 0.75 m and 0.375 m, as shown in Fig. 10(b)–(f). The corresponding support domain lengths are respectively $h = 12$ m, 6 m, 3 m, 1.5 m and 0.75 m. The Neo-Hookean model is considered, where the Young's modulus $E = 500$ MPa and the Poisson's ratio $\nu = 0.35$. The membrane is subjected to a vertically distributed force $P = 20$ MPa on the right boundary. The penalty value for the essential boundary condition is $\kappa = 1 \times 10^6$ and the learning rate of the Adam optimiser is 10^{-4}. All the cases are trained by 5×10^4 iterations.

Fig. 11 presents the displacement and Cauchy stress contours from the proposed RPIM-NNS and FEM. All the displacement results obtained by the RPIM-NNS align well with the FEM results. Besides, in the Cauchy stress contours, only slight discrepancies can be observed at the top-left corner of the membrane from the RPIM-NNS when $dx_c = 6$ m. Table 4 lists the deflections in the y direction at the point A with respect to different RBF centre spacings. The reference deflection results [47] with respect to the numbers of degrees of

Table 3
CPU time and the deflection result at the point A by using a single thread and the eight-thread parallel computing scheme.

	Single thread			Eight threads		
dx_c (m)	Searching (s)	Training (s)	V_A (m)	Searching (s)	Training (s)	V_A (m)
6	0.0030129	6.82116	−0.857137	0.003308	22.347	−0.857137
3	0.0188555	24.1093	−0.885793	0.0091486	24.6554	−0.885793
1.5	0.0933294	92.6103	−0.889014	0.0412543	49.6681	−0.889014
0.75	0.688504	365.286	−0.889873	0.281767	153.621	−0.889873
0.375	7.83161	1451.43	−0.889725	3.38183	540.795	−0.889725

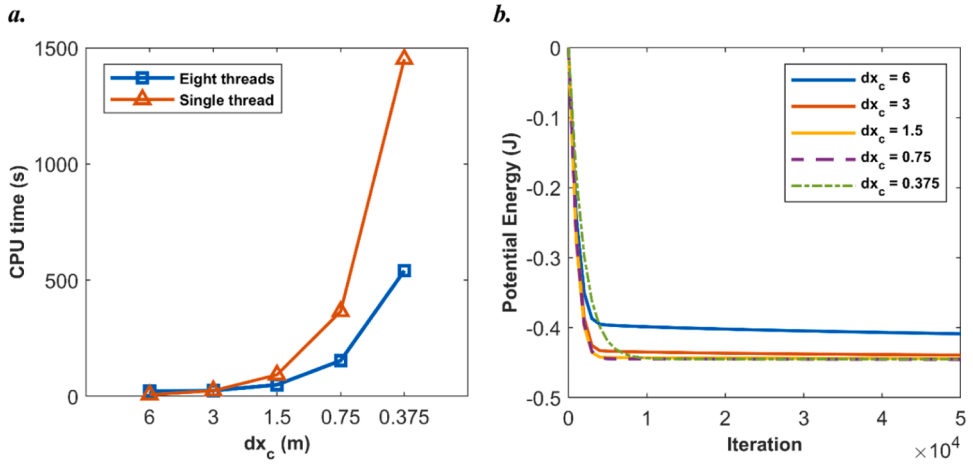


Fig. 6. (a) Overall CPU time for solving the cantilever beam problem by using a single thread and the eight-thread parallel computing scheme. (b) The history plots of the potential energy of the beam by using different RBF centre spacings.

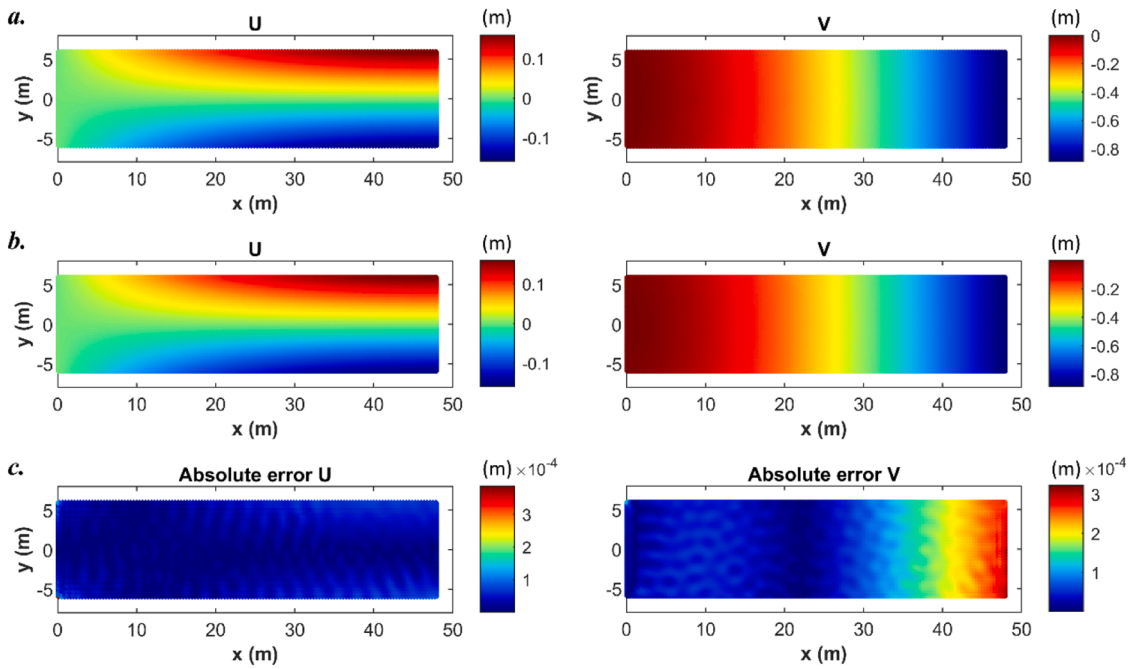


Fig. 7. (a) Displacement contours of the Cantilever beam problems obtained by the local RPIM-NNS when $dx_c = 0.375$ m and (b) Analytical solutions.

freedom (DOF) are also provided for comparison. Note that the pure displacement-based FEM is denoted as 'T_2' and the mixed FEM with an additional pressure field is denoted as 'T_2P_0'. Fig. 12 shows the potential energy history plots of the membrane with respect to different RBF centre spacings and the convergence plots of the deflection V_A. In Fig. 12(a), the horizontal axis matches the RBF centre spacings with the average length of the FEM element. It can be found that the deflection V_A from the RPIM-NNS can stably converge to -10.595 m. More importantly, the RPIM-NNS solution converges faster than the FEM results. As shown in Fig. 12(b), the overall energy of the membrane with different discretisation converges after roughly 380,000 training iterations.

4.2. Twisting rubber

A 3D twisting rubber cuboid problem is applied to test the proposed RPIM-NNS. The configuration of the rubber cuboid is given in Fig. 13, where the spacing of the RBF centres is $dx_c = 0.05$ m and the support domain length is $h = 0.15$ m. A nearly incompressible neoHookean material is applied, where the Young's modulus $E = 100$ Pa and the Poisson ratio $\nu = 0.499$. The U and V of the central line

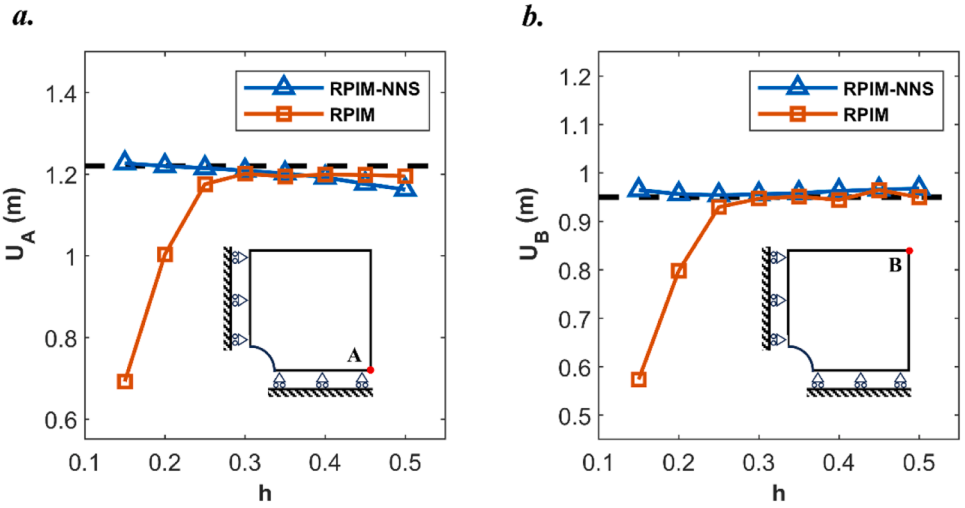


Fig. 8. (a) The displacement convergence plot at point A (1, 0) by using different *h*. (b) The displacement convergence plot at point B (0, 1) by using different *h*.

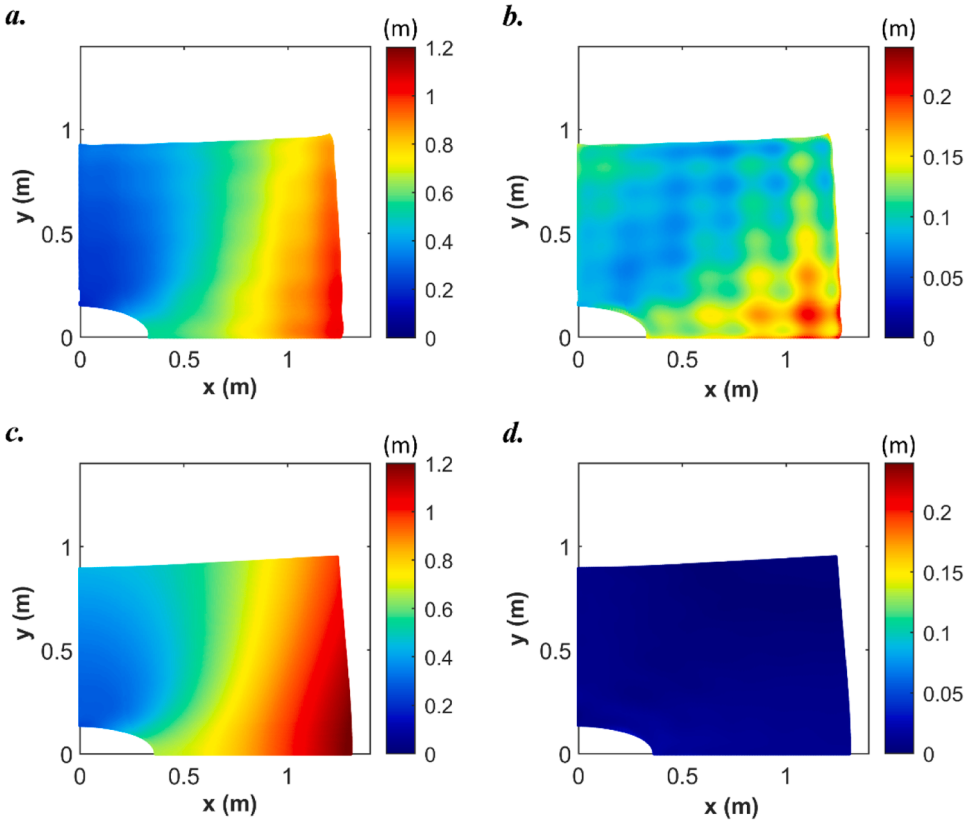


Fig. 9. Comparison between the original RPIM and the proposed RPIM-NNS for solving the perforated plate problem. (a) The absolute displacement contour obtained by the PRIM. (b) The absolute error contour of the absolute displacement field from the RPIM with respect to the FEM reference result. (c) The absolute displacement contour obtained by the RPIM-NNS using the settings in Section 3.4. (d) The absolute error contour of the absolute displacement field for the RPIM-NNS with respect to the FEM reference result. Note that both the RPIM-NNS and RPIM use the same RBFs. All the displacements are scaled with a factor of 0.25.

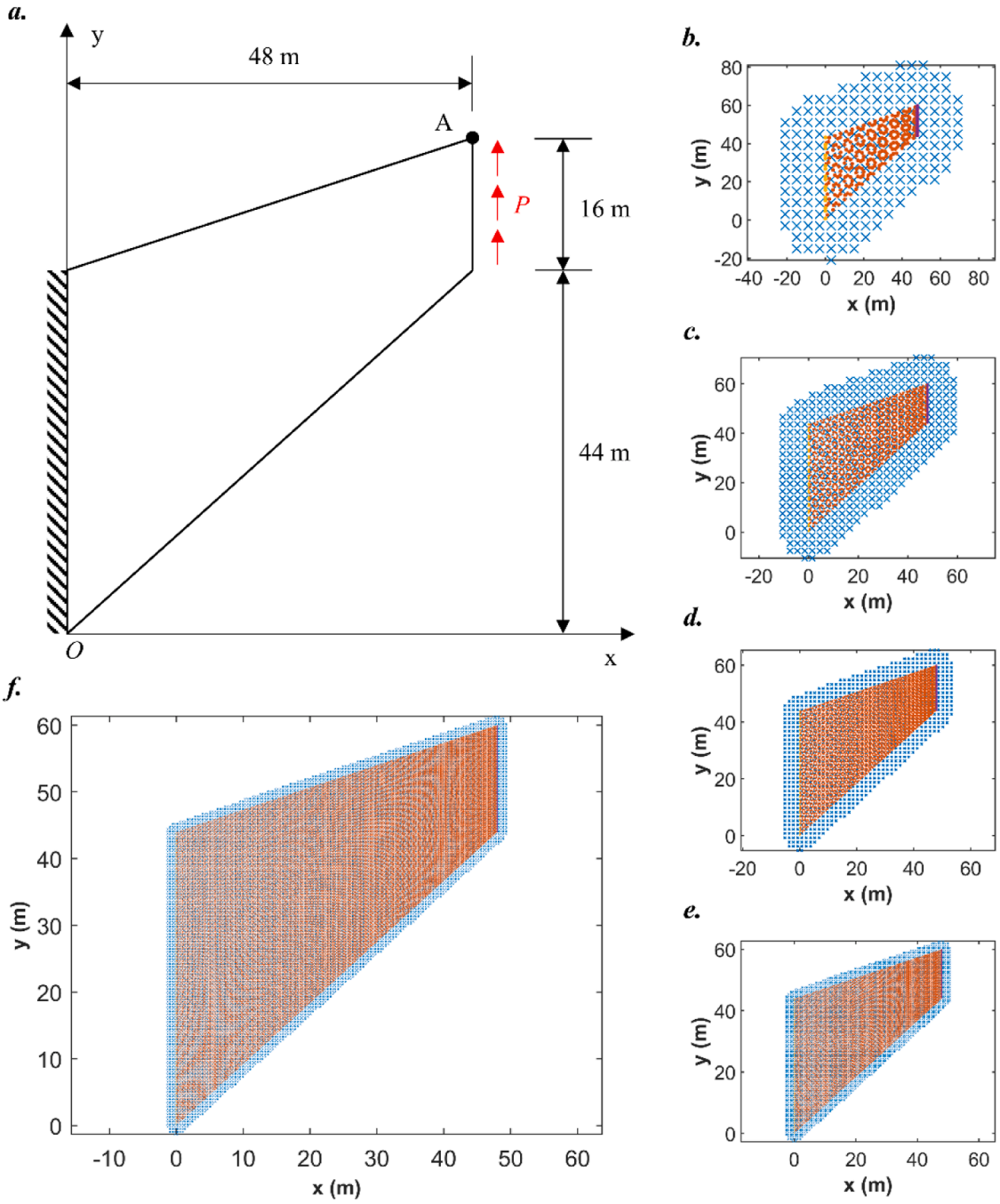


Fig. 10. (a) The configuration of the Cook's membrane problem. Different discretisation for the cantilever beam problem, where the spacing of the RBF centres are (b) $dx_c = 6$ m; (c) $dx_c = 3$ m; (d) $dx_c = 1.5$ m; (e) $dx_c = 0.75$ m; (f) $dx_c = 0.375$ m. Note that the blue crosses represent the centres of the radial basis functions, the orange, yellow and purple dots represent the in-domain sample points, the left-boundary sample points and the right-boundary sample points, respectively.

of the rubber are fixed. The whole loading process is divided into 24 steps

$$\theta = \frac{\pi \cdot n}{8}, (n = 1, ..., 24), \tag{37}$$

where θ is the twisting angle. The penalty value for essential boundary condition is $\kappa = 1 \times 10^2$. As for the optimiser settings, the RPIM-NNS is trained by 5×10^4 iterations with the learning rate of 1×10^{-5} for each loading step. Fig. 14 shows the deformations of the

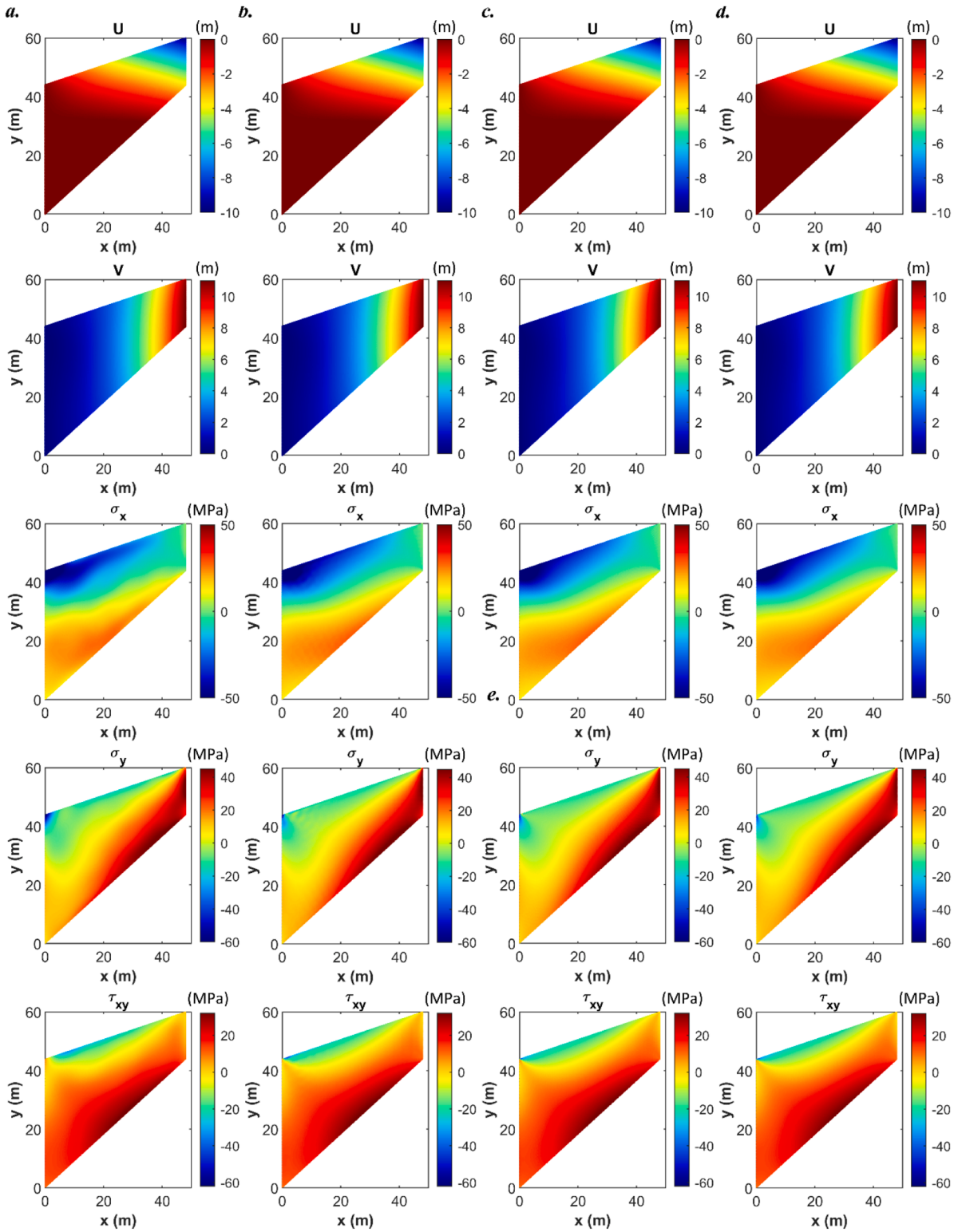


Fig. 11. Displacement and Cauchy stress contours of RPIM-NNS by using different RBF centre spacings. (a) $dx_c = 6$ m; (b) $dx_c = 1.5$ m; (c) $dx_c = 0.375$ m; (d) The FEM results obtained from ABAQUS with triangle elements and the global seed size of 0.1 m.

Table 4
The deflection in the y direction at the point A with respect to RBF centre spacing. The FEM results with respect to the degree of freedoms (DOFs) is also listed for comparison.

dx_c (m)	RPIM-NNS (m)	DOFs	FEM(T_2) [47] (m)	FEM(T_2P_0) [47] (m)
6	10.57999731	144	10.33773586	10.82618213
3	10.58761307	544	10.50672306	10.68358929
1.5	10.59487369	3280	10.56978840	10.61771743
0.75	10.59558335	12,960	10.58432907	10.60250393
0.375	10.59547446	51,520	10.59020842	10.59707501
		80,400	10.59124759	10.59625734
		115,680	10.59191221	10.59578301

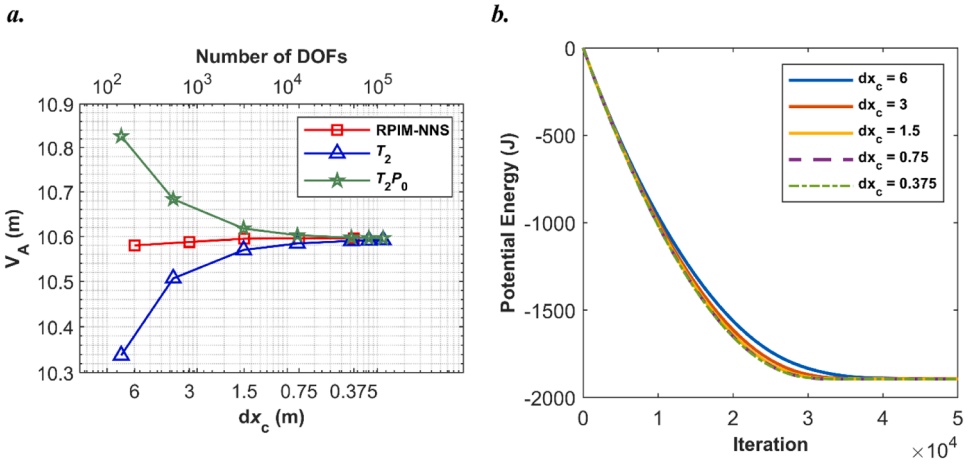


Fig. 12. (a) The potential energy history plots of the Cook's membrane with respect to different RBF centre spacings. (b) Convergence plots of the deflection in the y direction at point A (V_A). The reference results are from [47]. Note that the pure displacement-based FEM is denoted as 'T_2' and the mixed FEM with an additional pressure field is denoted as 'T_2P_0'. The bottom horizontal axis is for the RBF centre spacing and the top horizontal axis is for the number of DOFs.

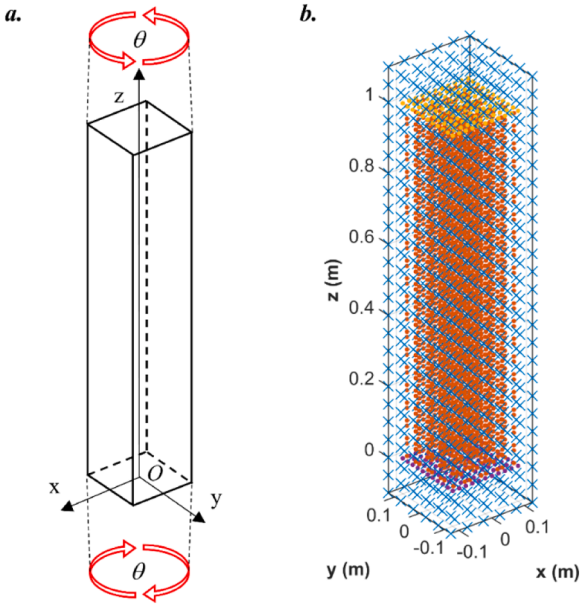


Fig. 13. (a) Configuration of the twisting rubber cuboid problem. (b) Sample points and RBF centres for the twisting rubber cuboid problem.

twisting cuboid at $\theta = \pi$, 2π and 3π rad. It is clear to observe that the RPIM-NNS is locking-free for such extreme torsion. Besides, the RPIM-NNS results exhibit a uniform torsional deformation along with the z direction.

Next, a 3D twisting rubber cylinder problem is also tested by the proposed RPIM-NNS. The configuration of rubber is given in Fig. 15. Again, a nearly incompressible neoHookean material is used, where the Young's modulus $E = 1.1572$ MPa and the Poisson ratio $\nu = 0.499$ are from the Ref. [48]. This time, the central line of rubber cylinder is no longer fixed. The penalty value for the essential boundary condition is $\kappa = 1 \times 10^6$ and the learning rate of the Adam optimiser is 10^{-5}. Under torsions, the rubber cylinder can exhibit torsional instability; that is, the central line of the rubber will also be twisted. According to Gent and Hua [49], the critical twisting angle θ_c, which starts the torsional instability, is defined as

$$\theta_c = 2\pi N \frac{R_0}{L_0}, \tag{38}$$

where N is the number of rotations of the rubber cylinder that starts the torsional instability, R_0 is the radius of the rubber cylinder and L_0 is the original length of the rubber cylinder. The deformation and von Mises stress contours from the proposed RPIM-NNS at different rotations are shown in Fig. 16. With increasing torsion, torsional instability is observed. Such instability starts when $\theta \in [1.25\pi, 1.75\pi]$, where the centre line of the rubber starts to be twisted. In addition, it is found that the rubber gradually forms a knot at the middle of the cylinder with further rotations. Finally, an obvious stress concentration is observed at the inner surface of the knot.

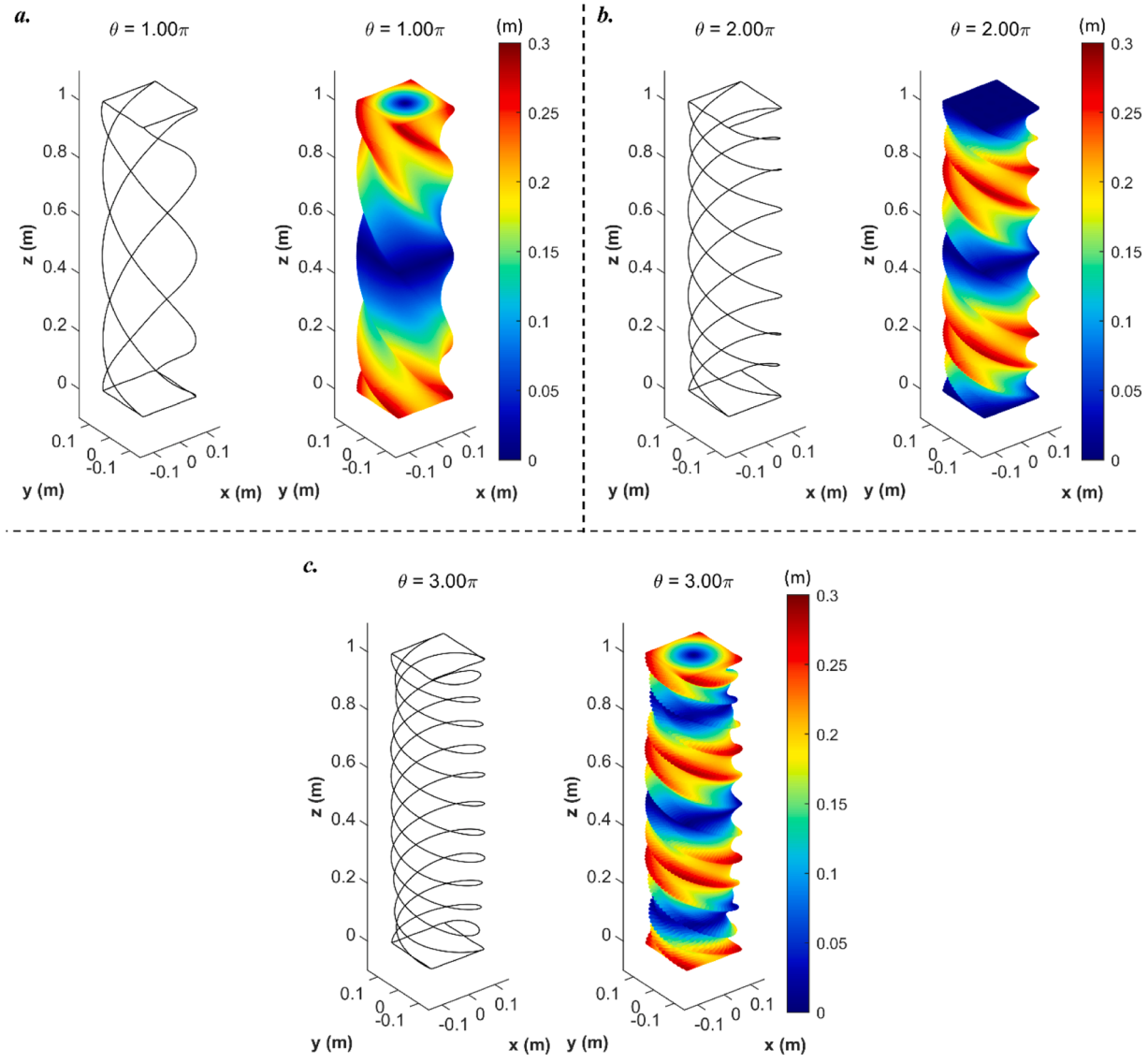


Fig. 14. Deformations of the rubber under different rotations by using the proposed RPIM-NNS. Note that the contours represent the absolute displacement.

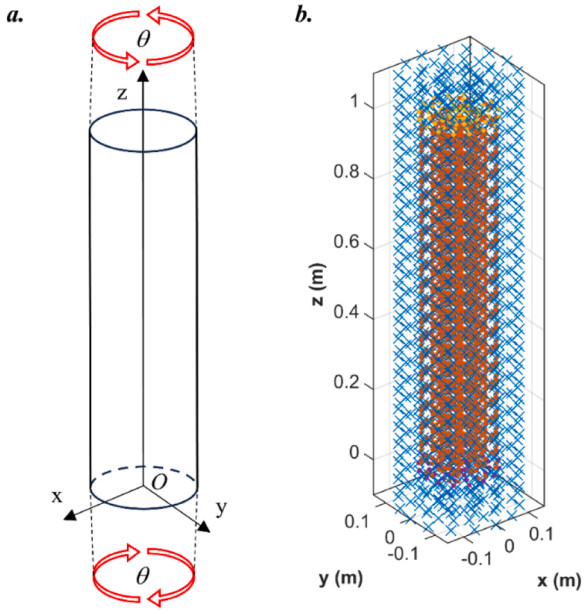


Fig. 15. (a) Configuration of the rubber cylinder. (b) Sample points and RBF centres for the rubber cylinder.

Moreover, it has been reported that the critical torsion angle can be delayed by stretching the rubber cylinder [50]. Therefore, we further simulate the torsional instability of the rubber cylinder under different stretches. The stretch ratio, ζ, is defined as

$$\zeta = \frac{L}{L_0}, \tag{39}$$

where L and L_0 are the stretched length and original length of the rubber cylinder. Fig. 17 presents the offset distance contour of the rubber cylinders under different stretches. It can be found that, with the increasing stretch ratio, the critical torsion of the rubber increases. Fig. 18 shows the comparisons of the critical twisting angle obtained by RPIM-NNS and references. The results obtained by the RPIM-NNS are in good agreement with the experimental data [49,51] and the theoretical solution [48].

5. Conclusions

In this work, we have proposed a robust radial point interpolation method empowered with neural network solvers (RPIM-NNS) for solving nonlinear solid mechanics problems. In this method, the displacement is interpolated by a series of arbitrarily settled RBFs, which guarantees the robustness of the RPIM-NNS with respect to irregular node distributions. Besides, the use of RBFs makes the RPIM-NNS easy to compute derivatives, resulting in seamless implementation with neural network solvers. In addition, the proposed RPIM-NNS enables the out-domain RBFs. Therefore, it can easily achieve higher accuracy for imposing boundary conditions than the original RPIM. Most importantly, neural network solvers are harnessed to solve the mechanics problems via minimising the total potential energy in the mechanics system. In this case, no additional numerical techniques are required for treating nonlinearities arising from material and large deformation, making the proposed RPIM-NNS much easier to implement for nonlinear problems. It has also been demonstrated that the proposed RPIM-NNS is compatible with parallel computing schemes.

In numerical examples, two nonlinear problems have been conducted to test the performance of the RPIM-NNS. The 2D Cook's membrane cases, it has been demonstrated that the RPIM-NNS can produce reliable displacement and stress solutions to nonlinear problems. Besides, the convergence rate of the RPIM-NNS is faster than the FEM solutions. In the 3D twisting rubber cases, it has been shown that the RPIM-NNS is locking-free under extreme torsion even when the material is nearly incompressible. Moreover, the RPIM-NNS successfully captures the torsional instability of twisting rubber cylinders. The torsional instability modelled by the RPIM-NNS matches well with theoretical and experimental results.

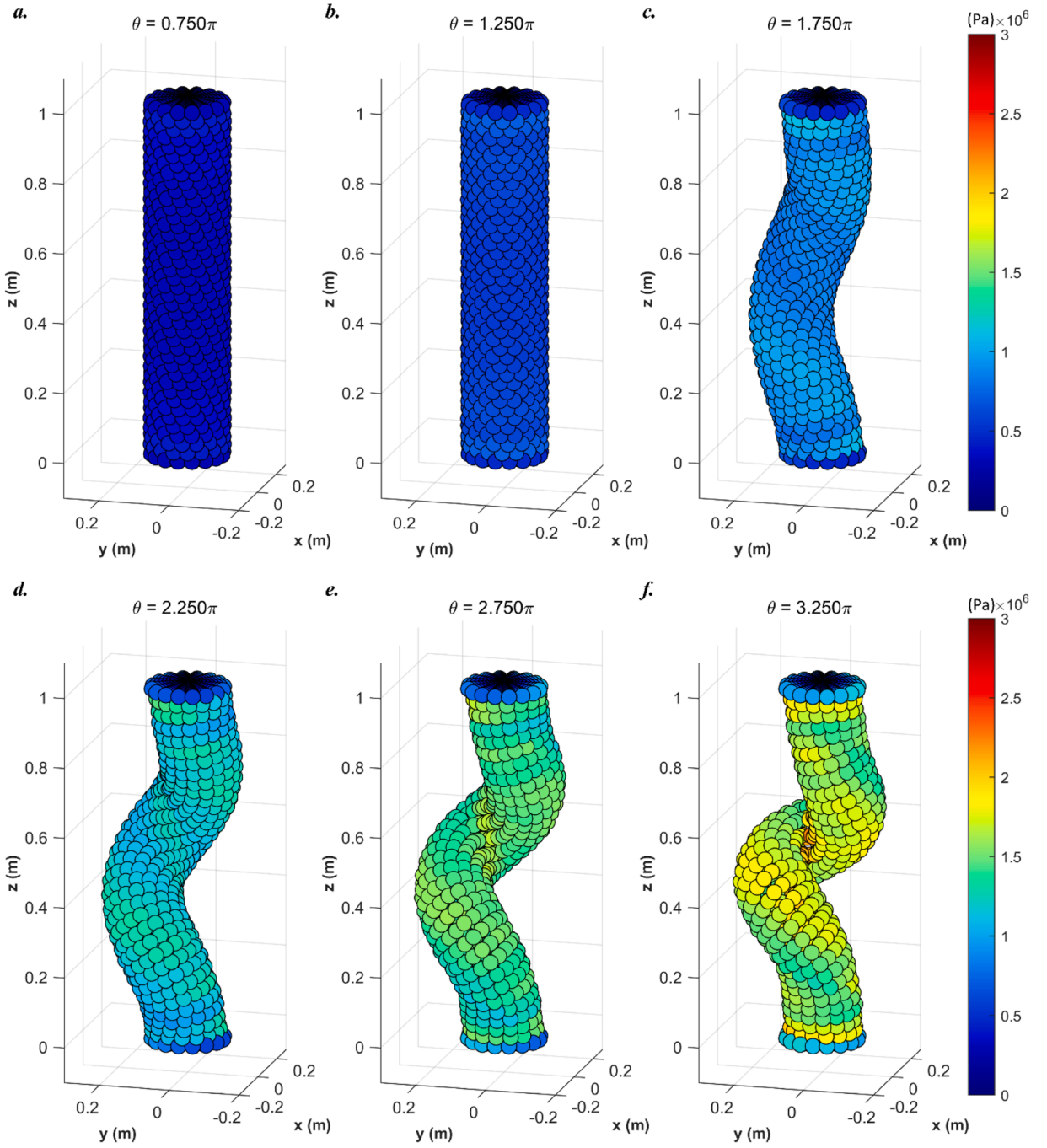


Fig. 16. Deformations and von Mises stress contours of the rubber cylinder under different rotations. (a) $\theta = 0.75\pi$; (b) $\theta = 1.25\pi$; (c) $\theta = 1.75\pi$; (d) $\theta = 2.25\pi$; (e) $\theta = 2.75\pi$; (f) $\theta = 3.25\pi$.

Though only limited numerical examples are presented, there exist no technical difficulties that barrier the application of the proposed RPIM-NNS to more complex and nonlinear problems, for example, elastoplastic and contact problems. Enhanced by powerful neural network solvers, the RPIM-NNS can simply and stably deal with problems as long as the functional loss can be obtained. Consequently, it is concluded that the proposed RPIM-NNS offers an effective way and shows great potential for dealing with nonlinear mechanics problems. This work also suggests that the neural network solvers, which comprise of training algorithms and a loss function, can help development not only machine learning methods but also to law-based models.

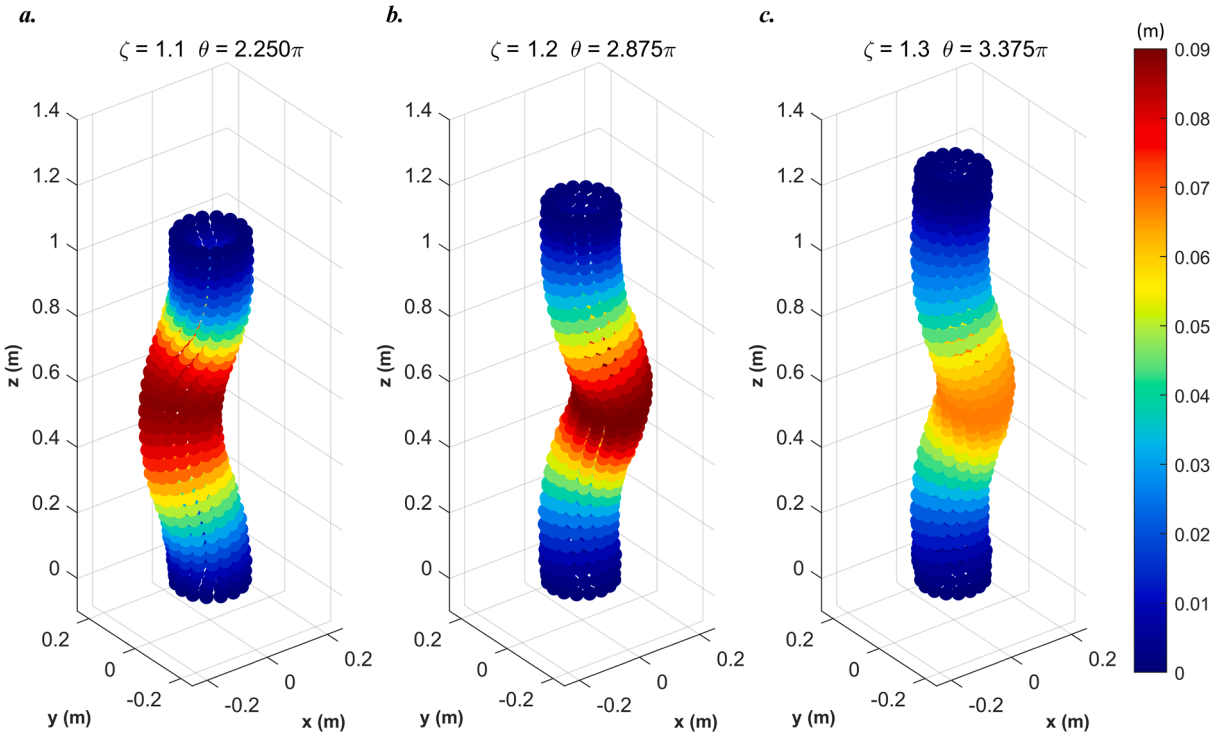


Fig. 17. Offset distance contours of the twisted rubber cylinder under different stretches (a) $\zeta = 1.1$; (b) $\zeta = 1.2$; (c) $\zeta = 1.3$.

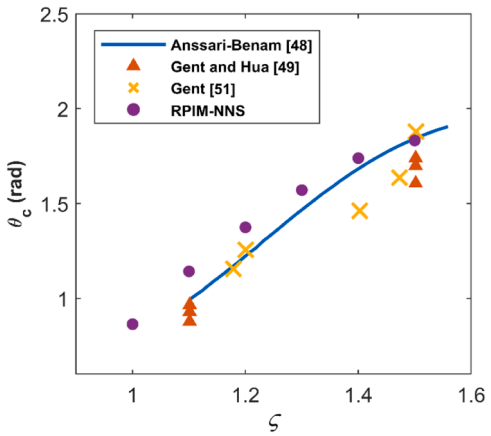


Fig. 18. Comparisons of the critical twisting angle θ_c at different stretch ratios ζ. The blue line is the theoretical solution for long slender rubber rods from [48]. The red triangles and yellow crosses are experimental data from [49] and [51], respectively.

CRediT authorship contribution statement

Jinshuai Bai: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Gui-Rong Liu:** Writing – review & editing, Methodology, Formal analysis. **Timon Rabczuk:** Writing – review & editing. **Yizheng Wang:** Writing – review & editing. **Xi-Qiao Feng:** Writing – review & editing. **YuanTong Gu:** Writing – review & editing, Supervision, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare no competing interests.

Data availability

We have uploaded our data and code on Github. Anyone can freely access to them once the paper is accepted.

Acknowledgements

Support from the Australian Research Council research grants (IC190100020 and DP 200102546) is gratefully acknowledged (J. Bai and Y.T. Gu). Support from the Key Project of the National Natural Science Foundation of China (12332005) is also gratefully acknowledged (Y. Liu).

References

[1] G.R. Liu, S.S. Quek, The Finite Element Method: A Practical Course, Butterworth-Heinemann, 2013.
[2] G.R. Liu, N. Trung, Smoothed Finite Element Methods, CRC Press, 2016.
[3] T.J. Ypma, Historical development of the Newton–Raphson method, SIAM Rev. 37 (1995) 531–551.
[4] E. Riks, An incremental approach to the solution of snapping and buckling problems, Int. J. Solids Struct. 15 (1979) 529–551.
[5] M. Crisfield, An arc-length method including line searches and accelerations, Int. J. Numer. Methods Eng. 19 (1983) 1269–1289.
[6] G.R. Liu, D. Karamanlidis, Mesh free methods: moving beyond the finite element method, Appl. Mech. Rev. 56 (2003) B17–B18.
[7] T. Belytschko, Y.Y. Lu, L. Gu, Element-free Galerkin methods, Int. J. Numer. Methods Eng. 37 (1994) 229–256.
[8] J. Bai, Y. Zhou, C.M. Rathnayaka, H. Zhan, E. Sauret, Y. Gu, A data-driven smoothed particle hydrodynamics method for fluids, Eng. Anal. Bound. Elem. 132 (2021) 12–32.
[9] S.N. Atluri, T. Zhu, A new meshless local Petrov-Galerkin (MLPG) approach in computational mechanics, Comput. Mech. 22 (1998) 117–127.
[10] G. Liu, Y. Gu, A local radial point interpolation method (LRPIM) for free vibration analyses of 2-D solids, J. Sound Vib. 246 (2001) 29–46.
[11] G. Liu, L. Yan, J. Wang, Y. Gu, Point interpolation method based on local residual formulation using radial basis functions, Struct. Eng. Mech. 14 (2002) 713–732.
[12] J. Wang, G. Liu, A point interpolation meshless method based on radial basis functions, Int. J. Numer. Methods Eng. 54 (2002) 1623–1648.
[13] J. Wang, G. Liu, On the optimal shape parameters of radial basis functions used for 2-D meshless methods, Comput. Methods Appl. Mech. Eng. 191 (2002) 2611–2630.
[14] G.R. Liu, Y. Gu, An Introduction to Meshfree Methods and Their Programming, Springer Science & Business Media, 2005.
[15] T.T. Truong, V.S. Lo, M.N. Nguyen, N.T. Nguyen, K.D. Nguyen, A novel meshfree radial point interpolation method with discrete shear gap for nonlinear static analysis of functionally graded plates, Eng. Comput. 39 (2023) 2989–3009.
[16] H. Ma, Y. Zhang, An improved cell-based smoothed radial point interpolation method using condensed shape functions for 3D interior acoustic problems, Comput. Methods Appl. Mech. Eng. 380 (2021) 113805.
[17] S. Ren, G. Meng, B. Nie, L. Zhou, H. Zhao, A novel stabilized node-based smoothed radial point interpolation method (SNS-RPIM) for coupling analysis of magneto-electro-elastic structures in hygrothermal environment, Comput. Methods Appl. Mech. Eng. 365 (2020) 112975.
[18] G.R. Liu, Machine Learning with Python: Theory and Applications, World Scientific, 2022.
[19] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nat. Rev. Phys. 3 (2021) 422–440.
[20] X. Zhuang, H. Guo, N. Alajlan, H. Zhu, T. Rabczuk, Deep autoencoder based energy method for the bending, vibration, and buckling analysis of Kirchhoff plates with transfer learning, Eur. J. Mech. A/Solids (2021) 87.
[21] S. Goswami, C. Anitescu, S. Chakraborty, T. Rabczuk, Transfer learning enhanced physics informed neural network for phase-field modeling of fracture, Theor. Appl. Fract. Mech. (2020) 106.
[22] I. Jeong, M. Cho, H. Chung, D.N. Kim, Data-driven nonparametric identification of material behavior based on physics-informed neural network with full-field data, Comput. Methods Appl. Mech. Eng. 418 (2024) 116569.
[23] K. Linka, E. Kuhl, A new family of Constitutive Artificial Neural Networks towards automated model discovery, Comput. Methods Appl. Mech. Eng. 403 (2023) 115731.
[24] J. Bai, T. Rabczuk, A. Gupta, L. Alzubaidi, Y. Gu, A physics-informed neural network technique based on a modified loss function for computational 2D and 3D solid mechanics, Comput. Mech. (2022).
[25] Y. Wang, J. Sun, W. Li, Z. Lu, Y. Liu, CENN: conservative energy method based on neural networks with subdomains for solving variational problems involving heterogeneous and complex geometries, Comput. Methods Appl. Mech. Eng. 400 (2022) 115491.
[26] J. Sun, Y. Liu, Y. Wang, Z. Yao, X. Zheng, BINN: a deep learning approach for computational mechanics problems based on boundary integral equations, Comput. Methods Appl. Mech. Eng. 410 (2023) 116012.
[27] J. Bai, Y. Zhou, Y. Ma, H. Jeong, H. Zhan, C. Rathnayaka, E. Sauret, Y. Gu, A general Neural Particle Method for hydrodynamics modeling, Comput. Methods Appl. Mech. Eng. (2022) 393.
[28] E. Samaniego, C. Anitescu, S. Goswami, V.M. Nguyen-Thanh, H. Guo, K. Hamdia, X. Zhuang, T. Rabczuk, An energy approach to the solution of partial differential equations in computational mechanics via machine learning: concepts, implementation and applications, Comput. Methods Appl. Mech. Eng. (2020) 362.
[29] J. Bai, G.R. Liu, A. Gupta, L. Alzubaidi, X.Q. Feng, Y. Gu, Physics-informed radial basis network (PIRBN): a local approximating neural network for solving nonlinear partial differential equations, Comput. Methods Appl. Mech. Eng. 415 (2023) 116290.
[30] E. Haghighat, M. Raissi, A. Moure, H. Gomez, R. Juanes, A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics, Comput. Methods Appl. Mech. Eng. (2021) 379.
[31] H. Wessels, C. Weißenfels, P. Wriggers, The neural particle method – An updated Lagrangian physics informed neural network for computational fluid dynamics, Comput. Methods Appl. Mech. Eng. (2020) 368.
[32] Y. Ghaffari Motlagh, P.K. Jimack, R. de Borst, Deep learning phase-field model for brittle fractures, Int. J. Numer. Methods Eng. 124 (2023) 620–638.
[33] S. Ruder, An overview of gradient descent optimization algorithms, arXiv preprint arXiv:1609.04747, (2016).
[34] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv preprint arXiv:1412.6980, (2014).
[35] J.P. Morris, P.J. Fox, Y. Zhu, Modeling low Reynolds number incompressible flows using SPH, J. Comput. Phys. 136 (1997) 214–226.
[36] H.M. Gutmann, A radial basis function method for global optimization, J. Glob. Optim. 19 (2001) 201–227.
[37] S. Wong, Y. Hon, M.A. Golberg, Compactly supported radial basis functions for shallow water equations, Appl. Math. Comput. 127 (2002) 79–101.
[38] H. Wendland, Error estimates for interpolation by compactly supported radial basis functions of minimal degree, J. Approx. Theory 93 (1998) 258–272.
[39] V.M. Nguyen-Thanh, X. Zhuang, T. Rabczuk, A deep energy method for finite deformation hyperelasticity, Eur. J. Mech. A/Solids 80 (2020).
[40] G.R. Liu, M.B. Liu, Smoothed Particle Hydrodynamics: A Meshfree Particle Method, World Scientific, Singapore, SINGAPORE, 2003.
[41] Z.X. Zhao, H. Liu, Z.X. Gong, A high-efficiency smoothed particle hydrodynamics model with multi-cell linked list and adaptive particle refinement for two-phase flows, Phys. Fluids 33 (2021).
[42] X. Xia, Q. Liang, A GPU-accelerated smoothed particle hydrodynamics (SPH) model for the shallow water equations, Environ. Model. Softw. 75 (2016) 28–43.

[43] J. Bai, H. Jeong, C.P. Batuwatta-Gamage, S. Xiao, Q. Wang, C.M. Rathnayaka, L. Alzubaidi, G.R. Liu, Y. Gu, An introduction to programming Physics-Informed Neural Network-based computational solid mechanics, Int. J. Comput. Methods (2023).
[44] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444.
[45] M. Zinkevich, M. Weimer, L. Li, A. Smola, Parallelized stochastic gradient descent, in: Adv. Neural Inf. Process. Syst., 23, 2010.
[46] S. Timoshenko, Theory of Elasticity, Oxford, 1951.
[47] J. Schröder, T. Wick, S. Reese, P. Wriggers, R. Müller, S. Kollmannsberger, M. Kästner, A. Schwarz, M. Igelbüscher, N. Viebahn, H.R. Bayat, S. Wulfinghoff, K. Mang, E. Rank, T. Bog, D. D'Angella, M. Elhaddad, P. Hennig, A. Düster, W. Garhuom, S. Hubrich, M. Walloth, W. Wollner, C. Kuhn, T. Heister, A selection of benchmark problems in solid mechanics and applied mathematics, Arch. Comput. Methods Eng. 28 (2020) 713–751.
[48] A. Anssari-Benam, C.O. Horgan, Torsional instability of incompressible hyperelastic rubber-like solid circular cylinders with limiting chain extensibility, Int. J. Solids Struct. (2022) 238.
[49] A.N. Gent, K.C. Hua, Torsional instability of stretched rubber cylinders, Int. J. Non-Linear Mech. 39 (2004) 483–489.
[50] J.G. Murphy, The stability of thin, stretched and twisted elastic rods, Int. J. Non-Linear Mech. 68 (2015) 96–100.
[51] A.N. Gent, Elastic instabilities in rubber, Int. J. Non-Linear Mech. 40 (2005) 165–175.