Robotica (2024), 1–14
doi:10.1017/S0263574724002066

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

A material point-based simulation method for soft robots with free boundary interactions

Siwei He[1], Jie Shen[2], Beijia Zhang[2], Faizan Ahmad[1], Hao Deng[1], Xiaohui Li[1], Jing Xiong[1] and Zeyang Xia[2]

[1]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
[2]School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai, China
Corresponding author: Jing Xiong; Email: jing.xiong@siat.ac.cn

Received: 5 January 2024; **Revised:** 29 October 2024; **Accepted:** 1 November 2024

Keywords: sensor or actuator design; soft robot; material point method; free boundary interaction; deformation

Abstract

Soft robots show an advantage when conducting tasks in complex environments due to their enormous flexibility and adaptability. However, soft robots suffer interactions and nonlinear deformation when interacting with soft and fluid materials. The reason behind is the free boundary interactions, which refers to undetermined contact between soft materials, specifically containing nonlinear deformation in air and nonlinear interactions in fluid for soft robot simulation. Therefore, we propose a new approach using material point method (MPM), which can solve the free boundary interactions problem, to simulate soft robots under such environments. The proposed approach can autonomously predict the flexible and versatile behaviors of soft robots. Our approach entails incorporating automatic differentiation into the algorithm of MPM to simplify the computation and implement an efficient implicit time integration algorithm. We perform two groups of experiments with an ordinary pneumatic soft finger in different free boundary interactions. The results indicate that it is possible to simulate soft robots with nonlinear interactions and deformation, and such environmental effects on soft robots can be restored.

1. Introduction

Soft robotics has become a popular research field. Compared to traditional robots, soft robots can achieve large deformations while ensuring interaction safety, making them advantageous in areas such as flexible grasping [1, 2], medical rehabilitation [3–5], underactuated finger [6, 7], and operations in unstructured environments [8, 9].

However, since the actuation of soft robots is often intrinsic to their structure, their kinematics result from a coupling of geometric relationships and complex mechanical mechanisms, making it difficult to directly solve for design parameters based on the desired motion. During the design process of soft robots, the researchers need to use simulations to evaluate the performance and behavior of the prototype, so that the soft robot works appropriately in the consistency of expectation. As soft materials such as rubber and hydrogel provide soft robots with enormous flexibility and adaptability, the behaviors of soft robots exhibit nonlinear interactions and deformation in complex environments and are hard to predict. Thus, the physical simulation based on continuum mechanics is adopted. Simulation of soft robots can be regarded as mathematics of physics. It contains continuum mechanical, geometrical, discrete, and surrogate models [10]. Continuum body assumption is made that material is continuously distributed in the occupied space of an object and can be divided into infinitesimal elements of identical properties [11]. Based on continuum body assumption, solving the partial differential equation derived from constitutive equations such as conservation of mass and energy leads to physical simulation based on continuum mechanics. Finite element analysis (FEA) is a widely used Galerkin method for solving PDEs numerically and has been applied to continuum mechanics successfully for a long time. A soft

finger for hand rehabilitation was developed, and its deformation was evaluated with FEA [11]. Another work Comber et al. [12] shows that a soft robot can drive the surgical needle, and they inspected the actuator displacement with FEA. An iterative algorithm was developed to optimize the structural design of pneumatic soft robots [13], and FEA was used to quantify the deformation of the intermediate optimized result. These approaches show that the physical simulation based on continuum mechanics supports and promotes the development of soft robots.

However, these approaches do not consider the effect of complex environments. The shortcoming of FEA, which is the incapability to handle free boundary interactions, does not support such consideration. For FEA, the two-step contact formulation procedure [14] and slave–master body technique [15] are used to handle naive fixed boundary coupling by fixing elements position and applying contact force on the contact boundary. A two-step scheme is adopted to solve solid and fluid systems separately and alternately [16] when the contact problem comes to a scenario with both fluid and solid materials. Aforementioned methods fail in such interactions and deformation scenarios because the contact cannot be determinant before simulation. Thus, FEA is incapable of simulating the scenario of soft robots in complex environments, and a new approach is required.

Meshless methods, that is, material point method (MPM) [17] and smoothed particle hydrodynamics (SPH) [18], are significantly different from FEA because they can autonomously solve the coupling of free boundary. The reason is how the shape function is supported. In FEA, a shape function is supported by a set of grouped nodes (which is an element). When boundary contact happens, two elements will go across as they will not contribute to the shape functions of each other. While in the meshless methods, all nodes (in favor of using the terminology of particles) within the support range will contribute to the same support function, which facilitates the information exchange between different phases and objects of materials. Though the shape function of SPH and MPM are supported similarly, SPH requires extra computation to search supporting particles compared to MPM. Research shows that MPM takes advantage of SPH in both computational efficiency and accuracy [19]. For such reasons, we use MPM to develop our approach for the simulation of soft robots. The proposed approach can handle nonlinear interactions and nonlinear deformation of soft robots.

Recently, many methods [20] and frameworks have been proposed to address the interaction problems between free boundaries and the environment. Differentiable simulators have also been integrated with gradient-based optimization algorithms specifically developed for soft robotics applications. DiffPD [21], a differentiable soft-body simulator based on finite element method (FEM), is designed for efficient soft-body learning and control and can be used to simulate an underwater starfish robot [22]. SoftCon [23], another differentiable soft-body simulator based on FEM, serves as a versatile framework for designing and controlling underwater soft-bodied organisms. ChainQueen [24, 25] and its follow-up work DiffTaichi [26] introduce differentiable physics-based soft-body simulators using the MPM. There are also other differentiable frameworks, such as Elastica [27]. In addition, many researchers use fluid–structure interaction (FSI) to simulate the deformation of soft robots in fluid environments. For instance, Xavier et al [28] employ bidirectional FSI to study the deformation of soft robots underwater impact, while Soomro et al [29] use FSI to investigate the interaction between a soft biomimetic frog robot and water.

These methods have made significant contributions to soft robot simulation with free boundary interactions. However, MPM has advantages over other methods in handling free boundary interactions in soft robotics. MPM naturally handles large deformations and interactions, which are common in soft robotics but are often not modeled in mesh-based approaches due to computational expense. As mentioned earlier, by utilizing a mesh-free approach, MPM handles complex boundary problems more effectively compared to mesh-based methods like FEM [21, 23] and Coupled Eulerian–Lagrangian (CEL). Compared to other mesh-free methods like SPH, MPM offers higher accuracy and computational.

Although ChainQueen has already applied MPM to soft robot simulation, our method offers greater stability. The explicit time integration in their method requires small timesteps to maintain numerical stability and leads to significant numerical errors when the timestep is large. In contrast, our MPM framework uses implicit time integration, which ensures numerical stability and guarantees higher

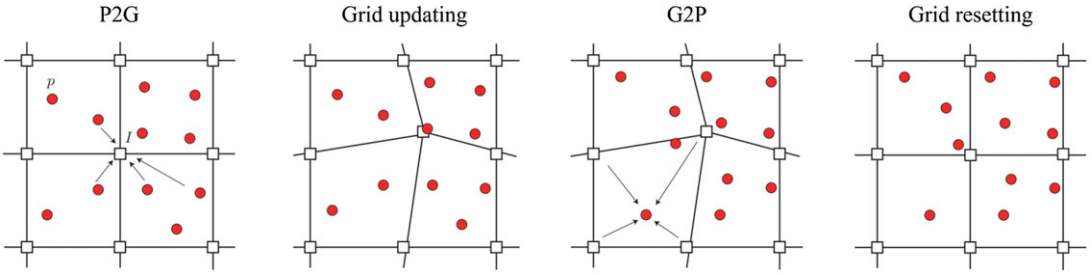

Figure 1. *An illustration of MPM update framework. From left to right: particle to grid, grid update, grid to particle and reset grid state. [17]*

numerical accuracy. Additionally, we have introduced automatic differentiation to relatively improve computational efficiency and accuracy.

In this study, the main contribution in this article can be described in two aspects:

- We address the open problem of simulation of soft finger with free boundary interactions. The proposed approach can be applied automatically without complicated algorithm design, especially in nonlinear deformation and interaction cases.
- We integrate automatic differentiation and implicit time integration into the modeling framework, which makes it more computationally efficient and stable, respectively.

Our key strengths include the ability to manage large deformation problems and complex boundary conditions with excellent stability. Although this process involves several simplifications when extending to multiphysical phenomena, such as Rayleigh damping and Coulomb friction, it can be theoretically proven that their impact is negligible in most cases. We have also experimentally verified that the simulation results remain highly accurate even after disregarding these factors.

The rest of the paper is organized as follows. In section 2, we will discuss the dynamic model principles for simulation of soft robot. It contains MPM framework and our modification in detail. Then, we illustrate the implemented algorithm in section 3. The simulation and experiment results will be shown in section 4.

2. Dynamic modeling for simulation of soft robot

2.1. MPM based modeling framework

When conducting simulation via MPM, the space is discretized by background grids, while the object is discretized by particles. The physical information is stored on particles, but grids enable such physical information to be updated. The MPM can perform both dynamic and static simulations, but for the simulation of soft robots, the dynamic simulation is more important since inertia affects the robot's behavior. Each time step of a dynamic MPM simulation involves three steps [24]. In the first stage, the physical properties of particles are transferred to the adjacent grids (spatial points at which the shape functions are centered). Mass, momentum, and external forces are transferred from particles to grids by $m_i = \sum_p \alpha_{ip} m_p$, $m_i \boldsymbol{v}_i^n = \sum_p \alpha_{ip} m_p \boldsymbol{v}_p^n$, and $\boldsymbol{f}_i^{ext} = \sum_p \alpha_{ip} \boldsymbol{f}_p^{ext}$, where variables with subscript p is the physical information on particles and variables with subscript i is the intermediate variables on grids, α_{ip} is the weight between specific particle p and grid i. In the second stage, the internal force $\boldsymbol{f}_i^{int^n}$ is computed and intermediate variables on grids are to be updated using explicit time integration, $\boldsymbol{v}_i^{n+1} = \boldsymbol{v}_i^n + \Delta t \boldsymbol{a}_i^n$, where $\boldsymbol{a}_i^n = (\boldsymbol{f}_i^{int^n} + \boldsymbol{f}_i^{ext})/m_i$. In the third stage, the intermediate variables on grids are transferred back to update the particles as $\boldsymbol{v}_p^{n+1} = \sum_i \alpha_{ip} \boldsymbol{v}_i^{n+1}$ and $\boldsymbol{x}_p^{n+1} = \boldsymbol{x}_p^n + \Delta t \boldsymbol{v}_p^{n+1}$. These stages are combined and referred as update framework of MPM. After these main stages, intermediate variables on grids are to be reset. An illustration is shown in Figure 1.

Table 1. *Constitutive equations for solids.*

Type	Strain Energy Density Function (SEDF)	Strain Range
Mooney-Rivlin	$\Psi = C_{01}\left(\bar{I}_1 - d\right) + C_{10}\left(\bar{I}_2 - d\right) + \frac{1}{D_1}(J-1)^2$	<200%
Yeoh	$\Psi = \sum_{i=1}^{n} C_{i0}(\bar{I}_1 - d)^i + C_{11}(J-1)^2$	<400%
Ogden	$\Psi = \sum_{i=1}^{n} \frac{\mu_p}{\alpha_p}\left(\lambda_1^{\alpha_p} + \lambda_2^{\alpha_p} + \lambda_1^{-\alpha_p}\lambda_2^{-\alpha_p} - 3\right)$	<400%

However, numerous MPM implementations have modified those above equations. Not all MPM frameworks are suitable for soft robot simulation. The most representative frameworks of MPM are fluid implicit particle (FLIP) and particle in cell (PIC). However, PIC and FLIP have flaws that cannot be applied directly in the simulation of soft robots. The PIC technique does not guarantee the conservation of angular momentum because it overlooks the local rotational motion, which severely dissipates energy. As soft robots often suffer from nonlinear deformation of shear and rotation, PIC method cannot handle the simulation correctly. FLIP was proposed to solve such a flaw in the simulation of fluids, but angular momentum is not conserved unless using a full mass matrix [30]. Thus, FLIP is not a proper framework for the simulation of soft robots.

Affine particle in cell (APIC) introduced affine velocity field tensor to store the angular momentum on particles, making APIC free from energy dissipation when simulating deformable objects. As the affine velocity field is added into the framework, it needs to be computed and updated for numerical computation. In the third stage, the updated affine velocity field $\boldsymbol{C}_p^{n+1}$ is computed as $\boldsymbol{C}_p^{n+1} = 4\sum_i \boldsymbol{v}_i^{n+1}(\boldsymbol{x}_i - \boldsymbol{x}_p)^T \alpha_{ip}/h^2$, it is used to update the deformation gradient $\boldsymbol{F}_p^{n+1} = (\boldsymbol{I} + \Delta t \boldsymbol{C}_p^{n+1})\boldsymbol{F}_p^n$, where $\boldsymbol{I}$ represents the identity matrix and which will be used to compute internal force. The proof of exact angular momentum conservation has been reported in a previous work [31]. APIC is more stable when compared with PIC and FLIP [32]. Moving least squares material point method (MLS-MPM) [24, 33] simplifies the computation of APIC framework by introducing the moving least squares function into interpolation. Our MPM soft robot simulation approach is based on APIC and MLS-MPM.

2.2. The nonlinear material models with automatic differentiation

Automatic differentiation is a technique to compute the derivatives of functions specified by a program [34]. When a mathematical function is computed, it executes a consecutive series of basic operation and functions. By using chain rule, the overall derivative of the given function is the product of all derivatives during each operation. Moreover, this idea makes sure the result is purely analytical when compared to the finite difference method.

We mainly focus on soft solids and Newtonian fluids as they are ubiquitous in soft robotic systems. Soft robots are often fabricated by materials like hydrogel, silicon-based rubber, and some sorts of polymers. Hyperelastic models can restore the nonlinear stress-strain relationship exhibited by those materials, which fails in linear elastic models. We use Mooney-Rivlin, Ogden, and Yeoh [35] model to exhibit the simulation of soft robot. Their constitutive equations are listed in Table 1 (In the table, d refers to the dimensionality of the simulation). The simulation results based on three material models are shown in Figure 2 (visualized using Matplotlib [36]).

For Newtonian fluids [37], the authors mentioned that using fixed corotational model (which is a SEDF) is capable to simulate Newtonian fluid as $\Psi = \lambda/2(J-1)^2$, where λ is bulk modulus for the fluid and J is the determinant of the deformation gradient. Furthermore, taking the derivative of this function yields a relation identical to the equation of state (EOS) used in SPH simulations [38]. For fluids simulation, the deformation gradient needs to be reset as $\boldsymbol{F} = J^{\frac{1}{d}}\boldsymbol{I}$.

To incorporate these material models into the MPM framework. The relation exists in MLS-MPM [33] as

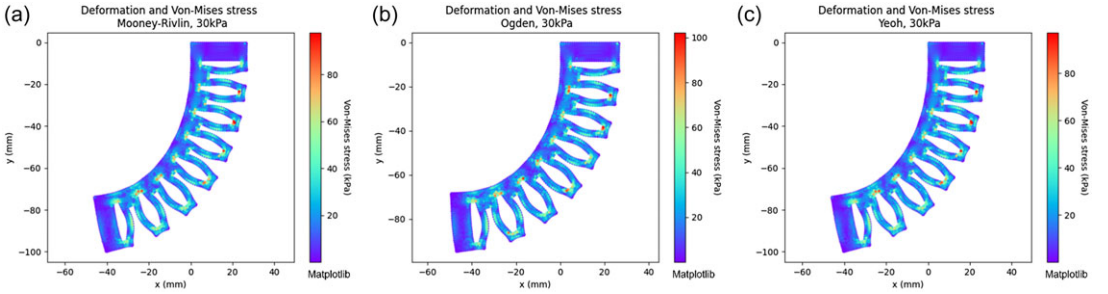

Figure 2. *The simulation results of a bending soft finger based on three material models: (a) Monney-Rivlin. (b) Ogden. (c) Yeoh.*

$$\boldsymbol{f}_i^{int^n} = -4/h^2 \sum_p V_p^0 \partial \Psi / \partial \boldsymbol{F}_p^n \left(\boldsymbol{F}_p^n\right)^T \left(\boldsymbol{x}_i - \boldsymbol{x}_p\right) \alpha_{ip} \tag{1}$$

Therefore, the derivative $\partial \Psi / \partial \boldsymbol{F}$ is required. A manual calculation is usually performed to obtain this value. However, for some complex SEDF, such a calculation is not feasible. In these cases, the strain energy of the local volume of the deformed soft robot can be computed over the entire volume of the soft robot to obtain the total strain energy.

$$e = \int_{\Omega_0} \Psi(\boldsymbol{F}(\boldsymbol{X}))d\boldsymbol{X} = \sum_p V_p^0 \Psi\left(\boldsymbol{F}_p\right) \tag{2}$$

Accordingly, the strain-based internal force is the derivative of total strain energy to the location of space grid $\boldsymbol{x}_i$:

$$\boldsymbol{f}_i^{int^n}\left(\hat{\boldsymbol{x}}_i\right) = -\partial e / \partial \boldsymbol{x}_i \tag{3}$$

where the location of space grid $\boldsymbol{x}_i$ is the function of space grid velocity $\boldsymbol{v}_i$, which is

$$\boldsymbol{f}_i^{int^n}\left(\hat{\boldsymbol{x}}_i\left(\boldsymbol{v}_i\right)\right) = -\left(\partial \boldsymbol{v}_i / \partial \boldsymbol{x}_i\right)\left(\partial e / \partial \boldsymbol{v}_i\right) = -\left(1/\Delta t\right)\left(\partial e / \partial \boldsymbol{v}_i\right) \tag{4}$$

As we measure the internal force before the grid updating, thus the grid velocity is $\mathbf{0}$, as it is an intermediate variable. We can get the internal force by computing e and its derivative using automatic differentiation. This modification is used in explicit time integration framework.

2.3. The implicit time integration

As mentioned previously, the computation of MPM framework relies only on the current states of the system (known as the explicit time integration). As a result of explicit time integration, numerical errors accumulate, reducing the simulation's stability and accuracy in three aspects:

1) the material parameters are significant in numbers (from MPa to GPa for common materials), which increases this error.

2) the interval between time steps needs to be small enough to compensate the accumulation error.

3) there is always a tradeoff between material parameters and accuracy to get a robust simulation.

A solution to these problems is implicit time integration. Unlike the explicit framework, implicit time integration requires solving an equation involving both the current time step of the system and the next time step. Usually, it is solved by an optimization algorithm with finite difference method providing gradient information at the cost of accuracy and running time. We incorporate automatic differentiation to overcome the flaws of finite difference method using implicit time integration. The difference in stability between explicit and implicit time integration is illustrated using a simulation case of a cube freely falling under gravity with a large time step. The results, shown in Figure. 3, demonstrate that MPM

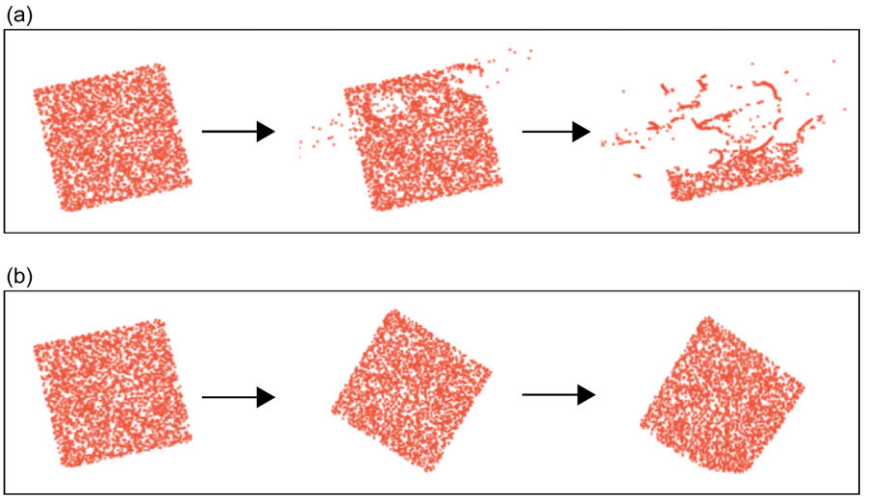

Figure 3. *The difference of explicit and implicit time frameworks' simulation results: (a) MPM using explicit framework fails when the simulation with large time step and large material parameters. (b) MPM using implicit framework and automatic differentiation improves the robustness of simulation.*

with an explicit framework fails when using a large time step, whereas MPM with an implicit framework and automatic differentiation proves to be more robust, improving the stability of the simulation.

A previous work [39] describes the implicit time integration scheme of APIC. However, our approach is innovative in three aspects. First, we use grid velocity as independent variable rather than intermediate grid position, which is more accurate and precise. Second, our method is based on MLS-MPM. Lastly, the adoption of automatic differentiation makes our approach to compute derivatives much faster and accurate.

The only difference between implicit and explicit time integration is the update of grid velocity. For an implicit time integration, the update of grid intermediate variable $\boldsymbol{v}_i$ becomes [40]:

$$\boldsymbol{v}_i^{n+1} = \boldsymbol{v}_i^n + \Delta t \boldsymbol{a}_i^{n+1} \tag{5}$$

According to kinematics, $\boldsymbol{a}_i^{n+1} = (\boldsymbol{v}_i^{n+1} - \boldsymbol{v}_i^n)/\Delta t$, and dynamics, $\boldsymbol{a}_i^{n+1} = (\boldsymbol{f}_i^{int^{n+1}} + \boldsymbol{f}_i^{ext})/m_i$. The two expressions are equal:

$$\boldsymbol{h} = \left(\boldsymbol{v}_i^{n+1} - \boldsymbol{v}_i^n\right)/\Delta t - \left(\boldsymbol{f}_i^{int^{n+1}} + \boldsymbol{f}_i^{ext}\right)/m_i \tag{6}$$

The roots which satisfy $\boldsymbol{h} = \boldsymbol{0}$ is going to be calculated. By solving this equation, the grid velocity is then updated. In Eq. 6, there are two variables of next time step $n+1$, $\boldsymbol{v}_i^{n+1}$ and $\boldsymbol{f}_i^{int^{n+1}}$, but the internal force is dependent on grid velocity. To represent the internal force in terms of grid velocity, we begin with the total strain energy:

$$e(\boldsymbol{x}) = \int_{\Omega_0} \Psi(\boldsymbol{F}(\boldsymbol{X}))d\boldsymbol{X} = \sum_p V_p^0 \Psi\left(\boldsymbol{F}_p\right) \tag{7}$$

The updated intermediate grid position is $\hat{\boldsymbol{x}}_i = \boldsymbol{x}_i + \Delta t \boldsymbol{v}_i$, and the stress-based force on grid $\boldsymbol{f}_i^{int}(\hat{\boldsymbol{x}}_i)$ is

$$-\frac{\partial e}{\partial \hat{\boldsymbol{x}}_i} = -\partial \sum_p V_p^0 \Psi\left(\hat{\boldsymbol{F}}_p\left(\hat{\boldsymbol{x}}_i\left(\boldsymbol{v}_i\right)\right)\right)/\partial \hat{\boldsymbol{x}}_i \tag{8}$$

The internal force can be written in terms of grid velocity by taking a second derivative:

$$-\frac{\partial e}{\partial \hat{\boldsymbol{x}}_i} = -\frac{\partial e}{\partial \boldsymbol{v}_i}\frac{\partial \boldsymbol{v}_i}{\partial \hat{\boldsymbol{x}}_i} = -\frac{1}{\Delta t}\frac{\partial e}{\partial \boldsymbol{v}_i} \tag{9}$$

In Eq. 8, the intermediate deformation gradient $\hat{\boldsymbol{F}}_p$ can be calculated by

$$\hat{\boldsymbol{F}}_p\left(\hat{\boldsymbol{x}}_i\right) = \left(\boldsymbol{I} + \Delta t \boldsymbol{C}\left(\boldsymbol{v}_i\right)\right)\boldsymbol{F}_p \tag{10}$$

For implicit time integration, by taking $\boldsymbol{v}_i = \boldsymbol{v}_i^{n+1}$ and Eq. 6 for $\boldsymbol{h}(\boldsymbol{v}_i^{n+1})$ becomes

$$m_i\left(\boldsymbol{v}_i^{n+1} - \boldsymbol{v}_i^n\right)/\Delta t - \boldsymbol{f}_i^{ext} + 1/\Delta t\left(\partial e/\partial \boldsymbol{v}_i^{n+1}\right) \tag{11}$$

Eq. 11 has only one unknown variable $\boldsymbol{v}_i^{n+1}$. Also, at $\boldsymbol{h} = \boldsymbol{0}$ it is a large set of nonlinear equations. In order to simplify the process of finding the roots, we perform an integration of Eq. 11 over $\boldsymbol{v}_i^{n+1}$, to construct an objective function for optimization $E(\boldsymbol{v}_i^{n+1})$.

$$\sum_i \frac{1}{2\Delta t} m_i \left\| \boldsymbol{v}_i^{n+1} - \boldsymbol{v}_i^n - \Delta t \frac{\boldsymbol{f}_i^{ext}}{m_i} \right\|^2 + \frac{1}{\Delta t} e\left(\boldsymbol{v}_i^{n+1}\right) \tag{12}$$

By using automatic differentiation, the derivative $\partial E(\boldsymbol{v}_i^{n+1})/\partial \boldsymbol{v}_i^{n+1}$ can be easily obtained. In our implementation, we use chain rule to decouple contribution from different grids, which fastens the computation.

3. Numerical implementation for simulation of soft robot

There are three stages to be taken in a time step of dynamic analysis. For the first stage, the physical properties of particles are transferred to the adjacent grids (spatial points at which the shape functions are centered). Mass, momentum, and external forces are transferred from particles to grids by weight factor. For weight α_{ip}, we choose quadratic B-spline function [41] as it takes less time to compute.

$$N(x) = \begin{cases} 3/4 - |x|^2, & 0 \le |x| < 1/2 \\ 1/2\left(3/2 - |x|\right)^2, & 1/2 \le |x| < 3/2 \\ 0, & 3/2 \le |x| \end{cases} \tag{13}$$

The external force is applied on grids, indicating that the driven sources can be simulated by analyzing and modeling the mechanics of driven sources and then computing their direction and magnitude. There are many driven sources, for example, pneumatic pressure, hydraulic pressure, and heat driven. We use pneumatic soft robots [24, 25] to demonstrate how to apply driven sources to soft robot simulation. Air drives the soft robots using pressure. The driven pressure is required to simulate the behavior of pneumatic soft robots. For every particle located at the driven loading surface, there exists a force proportional to the input pressure and parallel to the normal direction of local surface. Thus, by estimating the particle's normal direction and area, the driven pressure can be applied in simulation. Nanson's relation [26] can be used to compute. The formula is given as follows:

$$\boldsymbol{n}\mathrm{d}a = J\boldsymbol{F}^{-T}\boldsymbol{N}\mathrm{d}A \tag{14}$$

where $\mathrm{d}a$ and $\mathrm{d}A$ are the area of the particle located at driven loading surface in current and reference configuration, respectively, and $\boldsymbol{n}$, $\boldsymbol{N}$ are the normal direction pointing to the interior of the material of $\mathrm{d}a$ and $\mathrm{d}A$, respectively. When generating the geometric model, particles on the driven loading surface are marked, and normal directions $\boldsymbol{N}$ are estimated. The area A is obtained when meshing the geometric model. For any time step after starting the simulation, the forces on particles located at the driven loading surface can be computed.

$$\boldsymbol{f}_p^{ext} = p^{load}\boldsymbol{n}\mathrm{d}a \tag{15}$$

where p^{load} is the magnitude of the input pressure. Then, the forces on particles are interpolated to grids:

$$\boldsymbol{f}_i^{ext} = \sum_p \alpha_{ip} p^{load}\boldsymbol{n}\mathrm{d}a \tag{16}$$

Table 2. *The implicit framework with automatic differentiation.*

Algorithm 1: Implicit time integration framework of MPM based on automatic differentiation

procedure: The procedure for a time step update
1: Reset all variables on grid (with subscript of i)
#Particle to Grid: map physical information on particles to grids
3: **for all** p **do**
4: **for all** i in adjacency of p **do**
5: $m_i, m_i \boldsymbol{v}_i \overset{\text{update}}{\leftarrow} \alpha_{ip}, m_p, \boldsymbol{v}_p, \boldsymbol{x}_p$
6: $\boldsymbol{f}_i^{ext} \overset{\text{update}}{\leftarrow} \boldsymbol{f}_p^{ext}$
7: **end for**
8: **end for**
Grid updating
9: Set $\boldsymbol{v}_i^{n+1_0} = \boldsymbol{v}_i^n$ as original value, set $k = 0$
10: **While** not converge **do**
11: $\hat{\boldsymbol{F}}_p \overset{\text{update}}{\leftarrow} \boldsymbol{v}_i = \boldsymbol{v}_i^{n+1}$
12: $e = \sum_p \sum_p V_p^0 \Psi(\hat{\boldsymbol{F}}_p)$
13: Compute $E(\boldsymbol{v}_i^{n+1_k})$
14: $\boldsymbol{v}_i^{n+1_{k+1}} \overset{\text{update}}{\leftarrow} \frac{\partial E}{\partial \boldsymbol{v}_i^{n+1_k}}$ # $\frac{\partial E}{\partial \boldsymbol{v}_i^{n+1_k}}$ by automatic differentiation
15: $k = k + 1$
16: **end while**
#Grid to particle: map intermediate variables on grids to particles
17: **for all** $\boldsymbol{p}$ **do**
18: **for all** i in adjacency of p **do**
19: $\boldsymbol{x}_p^{n+1}, \boldsymbol{v}_p^{n+1}, \boldsymbol{C}_p^{n+1}, \boldsymbol{F}_p^{n+1} \overset{\text{update}}{\leftarrow} \boldsymbol{v}_i^{n+1}$
20: **end for**
21: **end for**
22: **end procedure**

By using Eq. 16, we can incorporate the pneumatic pressure into our simulation without having to compute the normal direction and magnitude of local force for each time step separately. For the second stage, the internal force $\boldsymbol{f}_i^{int^n}$ is computed using automatic differentiation, and intermediate grid variables are updated using implicit time integration. To utilize automatic differentiation, JAX [42], a programing tool developed by Google, is used. It supports both forward and backward mode of automatic differentiation. As simulation requires extensive computation, so we used JAX which provides parallel computation on GPU. In the last stage, the intermediate variables on grids are transferred back to update the particles as $\boldsymbol{v}_p^{n+1} = \sum_i \alpha_{ip} \boldsymbol{v}_i^{n+1}$ and $\boldsymbol{x}_p^{n+1} = \boldsymbol{x}_p^n + \Delta t \boldsymbol{v}_p^{n+1}$. Algorithm 1 summarizes the implemented numerical simulation approach to clarify shooting interloop (Table 2).

4. Validation and results

4.1. Setup of experiments

Soft finger is the fundamental component of soft robotic systems, as they are responsible for moving or controlling wearables, haptic, and medical devices [43]. We chose soft finger to validate our simulation because it possesses a high degree of flexibility and has been applied to a variety of tasks. We use DragonSkin-30 silicone rubber to construct the soft finger's body and RTV silicone glue as the adhesive. The material parameters for DragonSkin-30 silicone rubber in Mooney-Rivlin is set as $C_1 = 1.19$kPa,

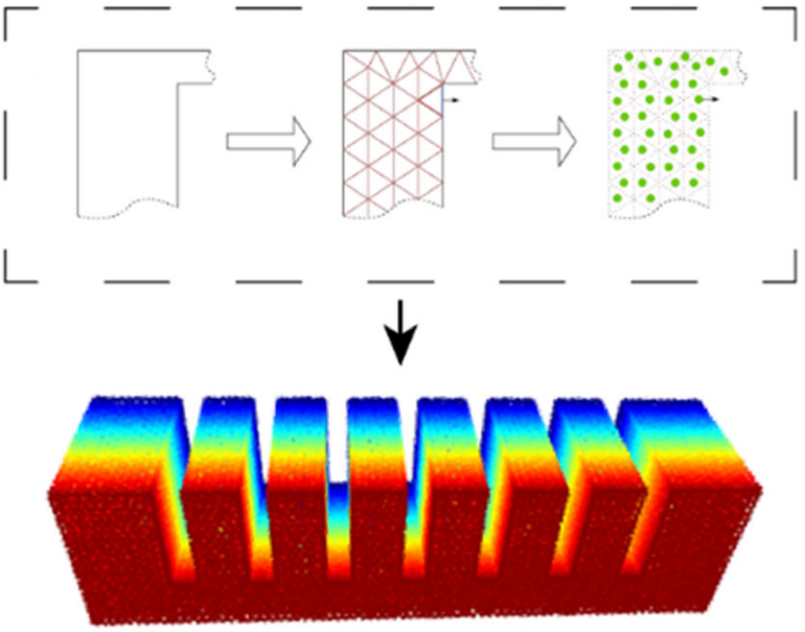

Figure 4. *Particle based geometry model from the meshed geometry model.*

$C_2 = 23.028\text{kPa}$, $1/D_1 = k/2 = 1.75\text{GPa}/2$, where k is the initial bulk modulus. For fluid simulation, its bulk modulus λ is set to less than 2.1GPa.

Our geometric model is generated using two steps shown in Figure 4. Firstly, we use Gmsh [44], an opensource software for FEA meshing, to generate tetrahedral mesh. Then, we make the center and volume of each element as the position and volume of the particle, respectively. In order to simulate the pressure acting on the inside chambers of the soft finger, a set of particles on chamber walls are specially marked out and assigned two extra properties. One is the area da on the wall surface, and other is the normal direction of area $\boldsymbol{n}$. These settings make it possible to apply the Nanson's relation to compute the input pressure.

4.2. Validation of simulation's accuracy

In our simulation's object, the free boundary condition contains two cases: specifically nonlinear deformation and nonlinear interactions with nonsolid materials. Therefore, in order to validate the algorithm's accuracy, we test two cases, respectively, which are soft finger simulation with nonlinear deformation and interaction. The figures and the error rate of experiments show that our framework can model the soft finger effectively to predict its behavior in complex environment.

Experiment I: *Validation of simulation with nonlinear deformation*

In this experiment, we validated the accuracy of our algorithm based on implicit time integration by using nonlinear deformation in air. We use von-Mises stress to demonstrate the concentration of stresses, as shown in Figure 5 (Visualization using Open3D [45]). The simulation results are consistent with experimental results. Moreover, our computation indicates that the stress concentration occurs simultaneously with severe local deformation at key locations. The results indicate that our method can accurately predict the deformation and stress distribution of soft robots.

When the deformation is nonlinear, the boundary coupling becomes complex. In FEA, handling such complex contact of free boundary requires extra computation. However, our approach solves such a problem autonomously.

Experiment II: *Validation of simulation with nonlinear interactions*

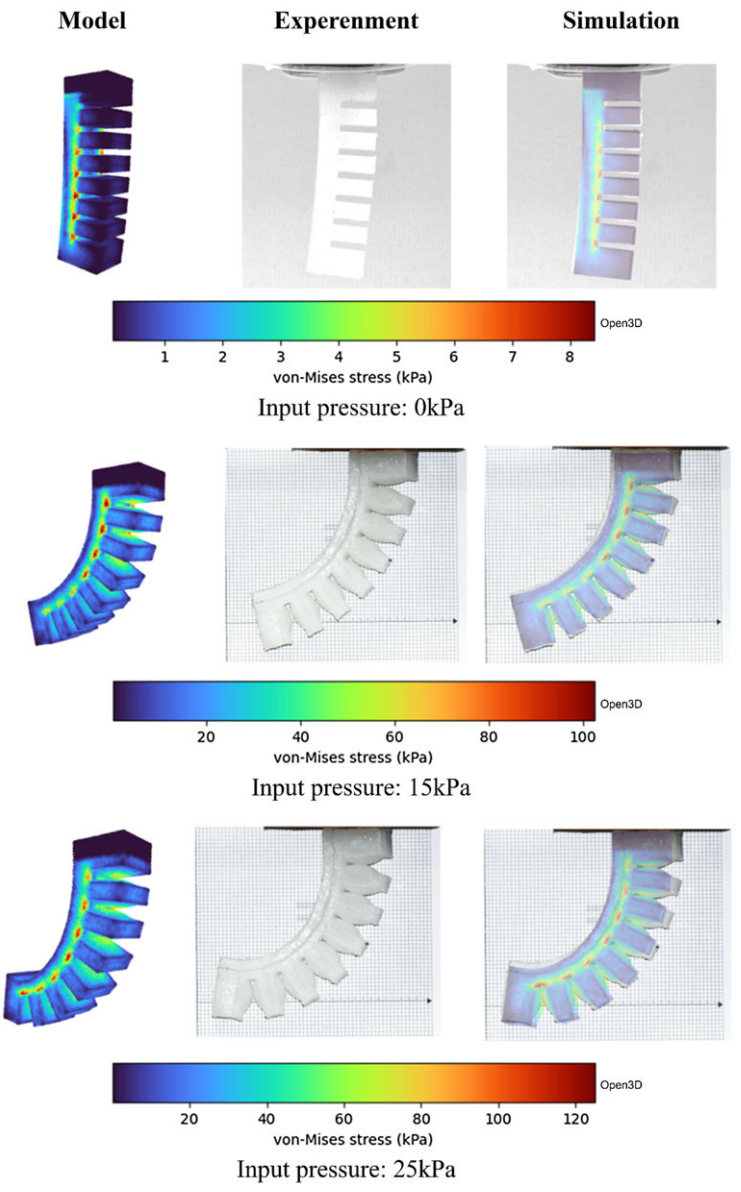

Figure 5. *Soft finger experiment and simulation results at different input pressures.*

We choose a fluid environment as the representative of nonlinear interactions, and we compare our results with those obtained using the Coupled Eulerian–Lagrangian (CEL) in Abaqus with C3D10M elements for dynamic explicit analysis. The results are shown in Figure 6 and Figure 7. In Figure 6, the y-axis indicates the Euclidean distance of the fingertip from its original position, and the x-axis is the input pressure. In Figure 7, the tracked point on the soft finger is marked as red dot, and the comparison is made between simulation and experiment. The results show that our approach can recover the deformation effects from the environment of the soft robot. Furthermore, the comparison reveals that our proposed method is more accurate, demonstrating the superiority of our method in simulation accuracy.

In Table 3, we computed the error ratio γ/β, where γ is euclidean distance between simulation and results of experiment in water, while β is the euclidean distance between simulation and results of

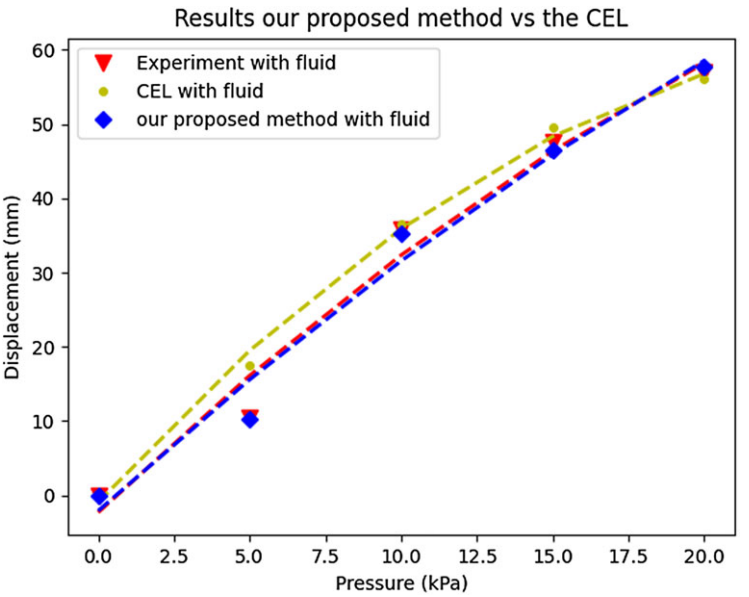

Figure 6. *Comparison of fingertip Euclidean distance under varying input pressure between our method and CEL simulation.*

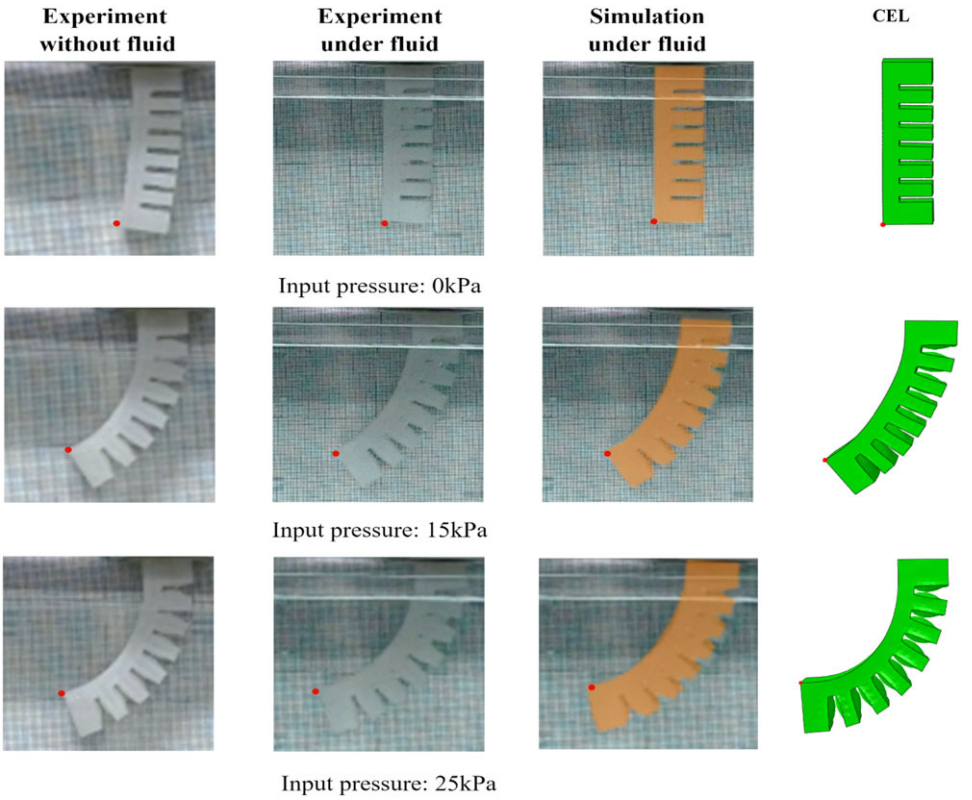

Figure 7. *Comparative analysis of deformation of soft finger in fluid environment between simulation and experimental results.*

Table 3. *Error ratio of simulation.*

Pressure (kPa)	**0**	**5**	**10**	**15**	**20**
Ratio (%)	0.6	13.7	16.7	15.1	8.7

experiment in air. During the initial pressure at 0 (kPa), we achieved an error rate of 0.6%. Similarly, for 5, 10, 15, and 20 (kPa), we achieved error rate of 13.7%, 16.7%, 15.1%, and 8.7%, respectively, which indicates that the computation results are nearly 10 times closer in fluid than in air. This indicates that our model meets the requirement of setting up a virtual environment where the interactions are fully exhibited.

5. Discussion

We proposed a novel method for soft robot simulation using MPM by employing automatic differentiation and implicit integration to address the challenges associated with free boundary interactions. We also designed experiments to validate our approach and performed comparisons to demonstrate the advantages of our proposed method in solving free boundary condition problems.

Moreover, there are two major aspects that will significantly increase the applicability of the proposed approach.

In order to emphasize the advantages of free contact simulation and accurate deformation prediction in complex environments, many possibilities for extending the approach to multiphysical phenomena are simplified, including Rayleigh damping and Coulomb friction (both of which dissipate energy in a nonconservation system), turbulent fluid models (which are effective in facilitating bidirectional interactions between fluids and solids), and magnetic and thermal effects in soft robots. In addition, applying different types of soft robot driven sources to simulation is an important topic.

This approach eliminates the need to explicitly determine contact surfaces within the soft robot itself or with other objects in its environment, so it can be used in exploring the adaptability of different soft robot systems to their working environments, which serves as the backbone for the design of optimal topological structures for soft robots.

Although we use a single example for a single type of soft robot, our approach has the potential to be applied to different kinds of soft robots. In practice, there are several complex soft robots with complex chambers, complex structures (even composed of multiple materials), and more diverse functions (not limited to bending or gripping). However, their simulation methods do not differ in principle from ours. The reason is that these soft robots will deform largely and have complex interactions with the environment, which causes the free boundary coupling and makes the simulation difficult. At the same time, our approach can handle these soft robots' nonlinear interactions and deformations.

Many driven sources other than pneumatic pressure need to be modeled. For pneumatic pressure, we find out that the force obeys Nanson's relation and then integrate it into our simulation. Nevertheless, the forces applied to soft robots are different for other driven sources. As the framework expressed, the external force is applied on grids, which means the driven sources can be simulated by analyzing and modeling the mechanics behind other driven sources, then computing their direction and magnitude.

This article primarily uses pneumatic bending soft robots as an example for simulation and experimentation. In future research, we will conduct more comprehensive simulations and experiments on other types of soft robots to analyze the impact of various parameters, including but not limited to pneumatic pressure, on the performance of soft robots.

6. Conclusion

In this article, we propose an approach for soft robot simulation using MPM, in order to overcome the free boundary interactions problem, particularly nonlinear deformation and interactions, so that a variety of soft robot scenarios (i.e., subsea bio-sampling, in-vivo surgery) of multiple phases and objects can be

simulated. In the proposed approach, a vivid virtual environment can be set up easily to accurately predict the behavior of soft robot, which fastens and promotes the design of soft robots. Also, we explored the application of automatic differentiation and implicit integration, which achieve efficient and stable simulation, respectively. The error rate of experiments shows that our framework can simulate the soft robot effectively to predict its behavior in complex environment. In the future, we will extend our work based on soft finger for more complex morphological soft robots.

Author contributions. Siwei He performed the experiments, developed the model, and drafted the manuscript. Faizan Ahmad contributed to the manuscript revision. Hao Deng and Xiaohui Li contributed to the experimental analysis. Jing Xiong contributed to the conceptualization of the study, the design of the research framework, the collection of materials, and the manuscript revision. Zeyang Xia motivated the manuscript and contributed to the framework design, materials collection, and manuscript preparation and revision. All authors gave final approval and agreed to be accountable for all aspects of the work.

Financial support. This work was supported by the National Natural Science Foundation of China (U2013205, 62073309), in part by Chinese Academy of Sciences Youth Innovation Promotion Association Excellent Member Program (Y201968), in part by Guangdong Basic and Applied Basic Research Foundation (2022B1515020042), and Shenzhen Science and Technology Program (JCYJ20220818101603008, JCYJ20210324115606018).

Competing interests. All authors disclosed no relevant relationships. No potential competing interests was reported by the authors.

Data availability statement. None.

References

[1] J. Shintake, V. Cacucciolo, D. Floreano and H. Shea, "Soft robotic grippers," *Adv Mater* **30**(29), 1707035 (2018).
[2] C.-J. Wang and B. Cheng, "Design of a robotic gripper for casting sorting robots with rigid-flexible coupling structures," *Robotica*, **42**, 1–19 (2024).
[3] N. Cheng, K. S. Phua, H. S. Lai, P. K. Tam, K. Y. Tang, K. K. Cheng, R. C. H. Yeow, K. K. Ang, C. Guan and J. H. Lim, "Brain-computer interface-based soft robotic glove rehabilitation for stroke," *IEEE Trans BioMed Eng* **67**(12), 3339–3351 (2020).
[4] M. Malvezzi, Z. Iqbal, M. C. Valigi, M. Pozzi, D. Prattichizzo and G. Salvietti, "Design of multiple wearable robotic extra fingers for human hand augmentation," *Robotics* **8**(4), 102 (2019).
[5] N. Nikafrooz and A. Leonessa, "A single-actuated, cable-driven, and self-contained robotic hand designed for adaptive grasps," *Robotics* **10**(4), 109 (2021).
[6] C. Gosselin, F. Pelletier and T. Laliberte, "An Anthropomorphic Underactuated Robotic Hand with 15 Dofs and a Single Actuator," **In:** *2008 IEEE International Conference on Robotics and Automation*, (2008) pp. 749–754.
[7] V. Niola, C. Rossi and S. Savino, "Influence of the Tendon Design on the Behavior of an Under-Actuated Finger," **In:** *Advances in Service and Industrial Robotics: Proceedings of the 26th International Conference on Robotics in Alpe-Adria-Danube Region, RAAD 2017*, (2018) pp. 1033–1042.
[8] Y. Di, Y. Zhang, Y. Wen and Y. Ren, "Modeling and optimization of motion for inchworm-inspired magnetically driven soft robot," *Robotica* **42**(1), 72–86 (2024).
[9] Z. Liu, Y. Wang, J. Wang, Y. Fei and Q. Du, "An obstacle-avoiding and stiffness-tunable modular bionic soft robot," *Robotica* **40**(8), 2651–2665 (2022).
[10] C. Armanini, F. Boyer, A. T. Mathew, C. Duriez and F. Renda, "Soft robots modeling: A structured overview," *IEEE Trans Robot* **39**(3), 1728–1748 (2023).
[11] P. Polygerinos, S. Lyne, Z. Wang, L. F. Nicolini, B. Mosadegh, G. M. Whitesides and C. J. Walsh, "Towards a Soft Pneumatic Glove for Hand Rehabilitation," **In:** *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (2013) pp. 1512–1517.
[12] D. B. Comber, J. E. Slightam, V. R. Gervasi, J. S. Neimat and E. J. Barth, "Design, additive manufacture, and control of a pneumatic MR-compatible needle driver," *IEEE Trans Robot* **32**(1), 138–149 (2016).
[13] Y. Chen, Z. Xia and Q. Zhao, "Optimal design of soft pneumatic bending actuators subjected to design-dependent pressure loads," *IEEE/ASME Trans Mechatron* **24**(6), 2873–2884 (2019).
[14] R. Wang, X. Zhang, B. Zhu, H. Zhang, B. Chen and H. Wang, "Topology optimization of a cable-driven soft robotic gripper," *Struct Multidiscip Optim* **62**(5), 2749–2763 (2020).
[15] K.-J. Bathe and A. Chaudhary, "A solution method for planar and axisymmetric contact problems," *Int J Numer Meth Eng* **21**(1), 65–88 (1985).
[16] S. Mitra and K. P. Sinhamahapatra, "2D simulation of fluid-structure interaction using finite element method," *Finite Elem Anal Des* **45**(1), 52–59 (2008).
[17] A. de Vaucorbeil, V. P. Nguyen, S. Sinaie and J. Y. Wu, "Material point method after 25 years: Theory, implementation, and applications," *Adv Appl Mech* **53**, 185–398 (2020).

[18] S. J. Lind, B. D. Rogers and P. K. Stansby, "Review of smoothed particle hydrodynamics: Towards converged Lagrangian flow modelling," *Proc Royal Soc A* **476**(2241), 20190801 (2020).
[19] Z. Sun, H. Li, Y. Gan, H. Liu, Z. Huang and L. He, "Material point method and smoothed particle hydrodynamics simulations of fluid flow problems: A comparative study," *Prog Comput Fluid Dyn Int J* **18**(1), 1–18 (2018).
[20] V. Ferrandy, F. F. Indrawanto, A. Sugiharto, E. Franco, A. Garriga-Casanovas, A. I. Mahyuddin, F. Rodriguez, S. M. Baena and V. Virdyawan, "Modeling of a two-degree-of-freedom fiber-reinforced soft pneumatic actuator," *Robotica* **41**(12), 3608–3626 (2023).
[21] T. Du, K. Wu, P. Ma, S. Wah, A. Spielberg, D. Rus and W. Matusik, "Diffpd: Differentiable projective dynamics," *ACM Trans Graph (TOG)* **41**(2), 1–21 (2021).
[22] T. Du, J. Hughes, S. Wah, W. Matusik and D. Rus, "Underwater soft robot modeling and control with differentiable simulation," *IEEE Robot Autom Lett* **6**(3), 4994–5001 (2021).
[23] S. Min, J. Won, S. Lee, J. Park and J. Lee, "SoftCon: Simulation and control of soft-bodied animals with biomimetic actuators," *ACM Trans Graph* **38**(6), 208 (2019).
[24] Y. Hu, J. Liu, A. Spielberg, J. B. Tenenbaum, W. T. Freeman, J. Wu, D. Rus and W. Matusik, "ChainQueen: A Real-Time Differentiable Physical Simulator for Soft Robotics," **In:** *IEEE International Conference on Robotics and Automation (ICRA)*, (2019) pp.6265–6271.
[25] A. Spielberg, T. Du, Y. Hu, D. Rus and W. Matusik, "Advanced soft robot modeling in chainQueen," *Robotica* **41**(1), 74–104 (2023).
[26] Y. Hu, L. Anderson, T.-M. Li, Q. Sun, N. Carr, J. Ragan-Kelley and F. Durand, "Difftaichi: Differentiable programming for physical simulation," arXiv preprint arXiv: 1910.00935, (2019).
[27] N. Naughton, J. Sun, A. Tekinalp, T. Parthasarathy, G. Chowdhary and M. Gazzola, "Elastica: A compliant mechanics environment for soft robotic control," *IEEE Robot Autom Lett* **6**(2), 3389–3396 (2021).
[28] M. S. Xavier, S. M. Harrison, D. Howard, Y. K. Yong and A. J. Fleming, "Modeling of soft fluidic actuators using fluid-structure interaction simulations with underwater applications," *Int J Mech Sci* **255**, 108437 (2023).
[29] A. M. Soomro, F. H. Memon, J.-W. Lee, F. Ahmed, K. H. Kim, Y. S. Kim and K. H. Choi, "Fully 3D printed multi-material soft bio-inspired frog for underwater synchronous swimming," *Int J Mech Sci* **210**, 106725 (2021).
[30] J. Brackbill, D. Kothe and H. M. Ruppel, "FLIP: A low-dissipation, particle-in-cell method for fluid flow," *Comput Phys Commun* **48**(1), 25–38 (1988).
[31] C. Jiang, C. Schroeder and J. Teran, "An angular momentum conserving affine-particle-in-cell method," *J Comput Phys* **338**, 137–164 (2017).
[32] C. Jiang, C. Schroeder, A. Selle, J. Teran and A. Stomakhin, The affine particle-in-cell method, (2015).
[33] Y. Hu, Y. Fang, Z. Ge, Z. Qu, Y. Zhu, A. Pradhana and C. Jiang, "A moving least squares material point method with displacement discontinuity and two-way rigid body coupling," *ACM Trans Graph* **37**(4), 1–14 (2018).
[34] J. N. a. S. J. Wright, "Numerical optimization," (2006).
[35] J. Bergström, "Mechanics of Solid Polymers: Theory and Computational Modeling," **In:** *Mechanics of Solid Polymers: Theory and Computational Modeling* (William Andrew, New York, United States, 2015) pp. 1–509.
[36] A. Pajankar, *Hands-on Matplotlib: Learn Plotting and Visualizations with Python 3* (Apress, Berkeley, CA, U.S., 2022).
[37] A. Stomakhin, C. Schroeder, C. Jiang, L. Chai, J. Teran and A. Selle, "Augmented MPM for phase-change and varied materials," *ACM Trans Graph* **33**(4), 1–11 (2014).
[38] D. Koschier, J. Bender, B. Solenthaler and M. Teschner, Smoothed particle hydrodynamics techniques for the physics based simulation of fluids and solids, (2020).
[39] C. Jiang, The material point method for the physics-based simulation of solids and fluids (University of California, Los Angeles, 2015).
[40] W. Press, B. Flannery, S. Teukolsky and W. Vetterling, "Numerical recipe in C the art of scientific computing," (2nd edn. Cambridge University Press, Cambridge, UK, 1991).
[41] F. Cheng, X. Wang and B. A. Barsky, "Quadratic B-spline curve interpolation," *Comput Math Appl* **41**(1), 39–50 (2001).
[42] R. Frostig, M. J. Johnson and C. Leary, "Compiling machine learning programs via high-level tracing," *Syst Mach Learn* **4**(9), (2018). https://cs.stanford.edu/~rfrostig/pubs/jax-mlsys2018.pdf.
[43] M. Li, A. Pal, A. Aghakhani, A. Pena-Francesch and M. Sitti, "Soft actuators for real-world applications," *Nat Rev Mater* **7**(3), 235–249 (2021).
[44] C. Geuzaine and J.-F. Remacle, "Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities," *Int J Num Meth Eng* **79**(11), 1309–1331 (2009).
[45] Q.-Y. Zhou, J. Park and V. Koltun, "Open3D: A modern library for 3D data processing," arXiv preprint arXiv: 1801.09847, (2018).

Cite this article: S. He, J. Shen, B. Zhang, F. Ahmad, H. Deng, X. Li, J. Xiong and Z. Xia, "A material point-based simulation method for soft robots with free boundary interactions", Robotica. https://doi.org/10.1017/S0263574724002066