Modelling of Eulerian incompressible fluid flows by using peridynamic differential operator

Cong Tien Nguyen[1], Selda Oterkus[1,*], Erkan Oterkus[1], Islam Amin[1,2], Murat Ozdemir[1], Abdel-Hameed El-Aassar[3], Hosam Shawky[3]

[1]PeriDynamics Research Center (PDRC), Department of Naval Architecture, Ocean and Marine Engineering, University of Strathclyde, Glasgow, United Kingdom

[2]Department of Naval Architecture and Marine Engineering, Port Said University, Port Said 42511, Egypt

[3]Egypt Desalination Research Centre of Excellence (EDRC) and Hydrogeochemistry Department, Desert Research Centre, Cairo 11753, Egypt

Abstract

This study presents a novel method for modelling of Eulerian incompressible fluid flow by using peridynamic differential operator. The peridynamic differential operator is used to calculate partial derivatives in the Navier-Stokes equations. The pressure Poisson equation is used to obtain the pressure field whereas the velocity field is obtained by solving momentum equations. The numerical procedure to solve Navier-Stokes equations in peridynamics for the incompressible fluid is also provided. The capability of the proposed peridynamic incompressible fluid model is demonstrated by considering problems of two-dimensional cavitation, a flow inside an open channel, and a flow over a cylinder. Moreover, to demonstrate the capabilities of the proposed model, the problems of two-dimensional cavitation and a flow over a cylinder are investigated for different Reynolds numbers. The vortex shedding is also captured for the problem of a flow over a cylinder at Reynolds number of $Re = 100$. For verification purposes, the peridynamic results are compared with the results obtained by ANSYS Fluent, a commercial fluid dynamics software.

Keywords: Peridynamics; Peridynamic differential operators; incompressible fluid; Eulerian approach; vortex shedding.

*Corresponding author: selda.oterkus@strath.ac.uk

Nomenclature

Latin Letters

$\mathbf{a}$	Matrix stores coefficients $a_1, a_2, \cdots, a_{25}$ that are used to calculate peridynamic orthogonal functions.
$a_1, a_2, \cdots, a_{25}$	coefficients that are used to calculate peridynamic orthogonal functions.
$A_{(j)}$	Area of node j
$\mathbf{A}$	Matrix that is used on the left-hand side of equation $\mathbf{Aa} = \mathbf{b}$ to obtain $\mathbf{a}$
$\mathbf{b}$	Matrix that is used on the right-hand side of equation $\mathbf{Aa} = \mathbf{b}$ to obtain $\mathbf{a}$
D	Diameter of the cylinder.
$\mathbf{d}^{(n-1)}$	Dot product of $\mathbf{u}^{(n-1)}$ and $\nabla\mathbf{u}^{(n-1)}$: $\mathbf{d}^{(n-1)} = \mathbf{u}^{(n-1)} \cdot \nabla\mathbf{u}^{(n-1)}$
$\mathbf{F}$	Force vector acting on the circular cylinder.
F_D	Drag force acting on the circular cylinder.
F_L	Lift force acting on the circular cylinder.
$f_{1(k)}, f_{2(k)}$	Components of vector $\mathbf{f}_{(k)}$ in x and y directions, respectively.
$f_{,x(k)}$	First order derivative at node k of f with respect to x coordinate.
$f_{,y(k)}$	First order derivative at node k of f with respect to y coordinate.
$f_{,xx(k)}$	Second order derivative at node k of f with respect to x coordinate.
$f_{,yy(k)}$	Second order derivative at node k of f with respect to y coordinate.
$f_{,xy(k)}$	Second order derivative at node k of f with respect to x, y coordinates.
$g^{10}_{2(k)(j)}$	Peridynamic orthogonal function for first order derivative $f_{,x(k)}$
$g^{01}_{2(k)(j)}$	Peridynamic orthogonal function for first order derivative $f_{,y(k)}$
$g^{20}_{2(k)(j)}$	Peridynamic orthogonal function for second order derivative $f_{,xx(k)}$
$g^{02}_{2(k)(j)}$	Peridynamic orthogonal function for second order derivative $f_{,yy(k)}$
$g^{11}_{2(k)(j)}$	Peridynamic orthogonal function for second order derivative $f_{,xy(k)}$
$\mathbf{K}_L$	Matrix that represents the Laplace operator $\nabla^2\mathbf{P}^{(n)} = \mathbf{K}_L\mathbf{P}^{(n)}$
k	Node k
j	Node j
$\mathbf{L}^{(n-1)}$	Laplace operator of the velocity field: $\mathbf{L}^{(n-1)} = \nabla^2\mathbf{u}^{(n-1)}$
N_p	Total number nodes in the discretized model.
N_t	Total number of time steps.
$\mathbf{n}$	Normal unit vector of a solid boundary.
$\mathbf{n}_L$	Normal unit vector of a left boundary of the fluid domain.
$\mathbf{n}_R$	Normal unit vector of a right boundary of the fluid domain.
p	Fluid pressure
$p^{(n)}$	Fluid pressure at n^{th} time step.
$\mathbf{P}^{(n)}$	Pressure vector for the fluid domain at the n^{th} time step.
$p^{(n)}_{,x(k)}$	First order derivative of velocity $p^{(n-1)}_{(k)}$ with respect to x coordinate.
$p^{(n)}_{,y(k)}$	First order derivative of velocity $p^{(n-1)}_{(k)}$ with respect to y coordinate.
$p^{(n)}_{,xx(k)}$	Second order derivative of velocity $p^{(n-1)}_{(k)}$ with respect to x coordinate.
$p^{(n)}_{,yy(k)}$	Second order derivative of velocity $p^{(n-1)}_{(k)}$ with respect to y coordinate.
$\mathbf{R}^{(n)}$	Vector on right-hand side of pressure Poisson equation.
$r^{(n)}_{(k)}$	Component k^{th} of vector $\mathbf{R}^{(n)}$ which corresponds to node k.

Re	Reynolds number.
$\mathbf{u}^{(n-1)}$	Velocity vector at $(n-1)^{th}$time step.
$\mathbf{u}^{(n)}$	Velocity vector at n^{th}time step.
u	Velocity component in x direction.
$u_{,x(k)}^{(n-1)}$	First order derivative of velocity $u_{(k)}^{(n-1)}$ with respect to x coordinate.
$u_{,y(k)}^{(n-1)}$	First order derivative of velocity $u_{(k)}^{(n-1)}$ with respect to y coordinate.
$u_{,xx(k)}^{(n-1)}$	Second order derivative of velocity $u_{(k)}^{(n-1)}$ with respect to x coordinate.
$u_{,yy(k)}^{(n-1)}$	Second order derivative of velocity $u_{(k)}^{(n-1)}$ with respect to y coordinate.
v	Velocity component in y direction.
$v_{,x(k)}^{(n-1)}$	First order derivative of velocity $v_{(k)}^{(n-1)}$ with respect to x coordinate.
$v_{,y(k)}^{(n-1)}$	First order derivative of velocity $v_{(k)}^{(n-1)}$ with respect to y coordinate.
$v_{,xx(k)}^{(n-1)}$	Second order derivative of velocity $v_{(k)}^{(n-1)}$ with respect to x coordinate.
$v_{,yy(k)}^{(n-1)}$	Second order derivative of velocity $v_{(k)}^{(n-1)}$ with respect to y coordinate.
x and y	x and y coordinates.

Greek Letters

Γ	Solid boundary.
δ	Horizon size.
Δt	Time step size.
Δx	Mesh size in x direction.
Δy	Mesh size in y direction.
Δr	Mesh size in the radial coordinate.
$\Delta\theta$	Mesh size in the angular coordinate.
μ	Dynamic viscosity of the fluid.
$\boldsymbol{\xi}$	Vector of relative coordinates between two nodes.
ξ_1	Component of vector $\boldsymbol{\xi}$ in x direction.
ξ_2	Component of vector $\boldsymbol{\xi}$ in y direction.
$\lvert\boldsymbol{\xi}\rvert$	Bond length or the distance between two nodes.
ρ	Fluid density.
$\boldsymbol{\sigma}$	Stress tensor acting on surface S
$\sigma_{xx}, \sigma_{xy}, \sigma_{yy}$	Components of stress tensor $\boldsymbol{\sigma}$.
$\omega(\lvert\boldsymbol{\xi}\rvert)$	Peridynamic weight function.
$\nabla \cdot ()$	Divergence operator.
$\nabla()$	Gradient operator.
$\nabla^2()$	Laplace operator.

Acronyms

CFD	Computational fluid dynamics
SPH	Smoothed particle hydrodynamics
MPS	Moving particle semi-implicit method
PD	Peridynamics/Peridynamic.
PDDO	Peridynamic differential operator(s)
FSI	Fluid-structure interaction
WSPH	Weakly compressible smoothed particle hydrodynamics
ISPH	Incompressible smoothed particle hydrodynamics

1. Introduction

Computational fluid dynamics (CFD) is the analysis of systems involving fluid flow, heat transfer and associated phenomena such as chemical reactions utilizing computer-based simulation. With a long-term history of development, the technique has become a very powerful tool with a wide range of industrial and non-industrial application areas. To perform CFD simulations, either mesh-based or meshfree methods can be used. The traditional mesh-based methods prefer the Eulerian approach which is widely used in many fluid dynamics related areas. Meanwhile, the meshfree methods such as smoothed particle hydrodynamics (SPH) or moving particle semi-implicit method (MPS) often use the Lagrangian approach in which the locations of fluid particles need to be updated during simulations (Benz, 1990; Meister et al., 2014; Monaghan, 1992; Morris, 1996; Shakibaeinia and Jin, 2011, 2012; Xie et al., 2005). Recently, fluid flow problems have also been simulated by using peridynamic differential operator (PDDO) (Gao and Oterkus, 2019b, 2020, 2021), which is a mathematical technique to represent field variables and their spatial derivatives in nonlocal form (Madenci et al., 2019a; Madenci et al., 2016; Madenci et al., 2017; Madenci et al., 2019b). PDDO can represent the original peridynamic equations of motion by recasting Navier's displacement equilibrium equations into their nonlocal form.

The original peridynamic theory was first introduced by Silling (2000) and the theory has been mainly used for problems in solid mechanics. To represent equations of motion of a node, peridynamics uses integro-differential equations instead of partial differential equations (Madenci and Oterkus, 2014; Silling, 2000; Silling and Askari, 2005; Silling et al., 2007; Silling and Lehoucq, 2010). Therefore, peridynamics (PD) is highly suitable for predicting progressive damages. Over 20 years of development, peridynamics has become applicable for many fields such as elastic and plastic deformations (Foster et al., 2010; Huang et al., 2019; Kružík et al., 2018; Madenci and Oterkus, 2016; Mitchell, 2011; Nguyen and Oterkus, 2019a), beam and shell structures (Chowdhury et al., 2016; Diyaroglu et al., 2019; Diyaroglu et al., 2015; Nguyen and Oterkus, 2019b, c, 2020, 2021; O'Grady and Foster, 2014a, b; Tien Nguyen and Oterkus, 2020; Yang et al., 2019), corrosion and fatigue (Chen and Bobaru, 2015; Chen et al., 2021; De Meo et al., 2016a; De Meo and Oterkus, 2017; Nguyen et al., 2020; Rokkam et al., 2019; Shojaei et al., 2020; Silling and Askari, 2014; Zhang et al., 2016), multiphysics and multiscale modeling (Askari et al., 2008; Bobaru and Ha, 2011; Gao and Oterkus, 2019c; Oterkus, 2015; Oterkus et al., 2017), composite and polycrystalline materials (De Meo et al., 2016b; Gao and Oterkus, 2019a; Ghajari et al., 2014; Hu et al., 2012; Hu et al., 2015; Oterkus, 2010; Oterkus et al., 2012). Recently, peridynamic differential operators (PDDO) for the approximation of field variables and their spatial derivatives were developed (Madenci et al., 2019a; Madenci et al., 2016; Madenci et al., 2017; Madenci et al., 2019b). The development of PDDO opened a wide range of applications in engineering (Haghighat et al., 2020; Madenci et al., 2019a; Madenci et al., 2016; Madenci et al., 2017). Shojaei et al. (2019) developed a generalized finite difference method based on the PDDO for the solution of problems in bounded and unbounded domains. Moreover, PDDO can also be used to represent Navier-Stokes equations in a nonlocal form Gao and Oterkus (2019b, 2020, 2021). Therefore, using peridynamics and peridynamic differential operators to simulate both structural dynamics and fluid dynamics problems will become highly possible. This will be a very important step that can allow simulating complex fluid-structure interaction (FSI) problems including structural damages that current local techniques are still facing significant difficulties to deal with.

To date, the development of PDDO in the field of fluid dynamics is very limited. Bazazzadeh et al. (2018) applied PDDO to incompressible inviscid fluid flow with moving boundaries for the solution of sloshing problems in tanks. Recently, Gao and Oterkus (2019b, 2020, 2021) represented Navier-Stokes equations in the nonlocal form by using peridynamic differential operators. The fluid model developed by Gao and Oterkus (2019b); (Gao and Oterkus, 2020, 2021) is the weakly compressible fluid model which is similar to the weakly compressible smoothed particle hydrodynamics (WSPH) (Benz, 1990; Monaghan, 1992; Morris, 1996). The weakly compressible fluid model employs an artificial equation of state that specifies pressure as an algebraic function of density. Therefore, a high speed of sound needs to be used to maintain an acceptable density variation (Cummins and Rudman, 1999). As a result, small time step sizes need to be used to ensure a stable solution in weakly compressible fluid models. In addition, due to the use of the artificial equation of state, the pressure can have large and non-physical fluctuations which can cause numerical instability (Hosseini and Feng, 2011). Therefore, the weakly compressible fluid models often face difficulties when dealing with problems where pressure is of physical interest such as FSI problems (Hosseini and Feng, 2011; Lee et al., 2008).

In smoothed particle hydrodynamics, to overcome difficulties faced by the WSPH, Cummins and Rudman (1999) introduced truly incompressible SPH (ISPH) algorithms based on the projection scheme. The truly incompressible fluid model produces more accurate predictions of velocity and forces on solids, and it is more efficient than weakly compressible fluid models since larger time step sizes can be used (Cummins and Rudman, 1999; Hosseini and Feng, 2011; Lee et al., 2008; Yildiz et al., 2009). Therefore, in this study, a peridynamic model for truly incompressible fluid based on the Eulerian approach is proposed. Specifically, the Navier-Stokes equations for incompressible fluid based on the Eulerian approach are represented in a nonlocal form by using the peridynamic differential operator. The pressure Poisson equation is used to obtain the pressure field, meanwhile, the velocity field is obtained by solving momentum equations. In this study, the proposed model is also called the Eulerian incompressible PDDO model.

This paper is organized as follows. Section 2 presents Navier-Stokes equations and the pressure Poisson equation for incompressible fluid based on the Eulerian approach. Section 3 presents the proposed Eulerian incompressible fluid PDDO model. The numerical procedure for the proposed Eulerian incompressible fluid PDDO model is also provided in this section. Section 4 presents various boundary conditions that are implemented in this study. Sections 5 and 6 presents numerical results and the conclusion.

2. Navier-Stokes equations for a two-dimensional incompressible fluid

According to Versteeg and Malalasekera (2007), the Navier-Stokes equations for an incompressible fluid in the 2D domain can be written by using the Eulerian approach as

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0 \tag{1a}$$

$$\rho\frac{\partial u}{\partial t}+\rho\left(u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}\right)=-\frac{\partial p}{\partial x}+\mu\left[\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right] \tag{1b}$$

$$\rho\frac{\partial v}{\partial t}+\rho\left(u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}\right)=-\frac{\partial p}{\partial y}+\mu\left[\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right] \tag{1c}$$

where x and y represent two components of the Cartesian coordinates, and t represents time. The terms u and v represent the velocity components in x and y directions, respectively. The parameter p represents the fluid pressure. The parameters ρ and μ represent the constant density and dynamic viscosity of the fluid. In the above equations, Eq. (1a) is known as the continuity equation or mass conservation equation. Meanwhile, Eq. (1b) and Eq. (1c) are the momentum equations in x and y directions, respectively.

The Navier-Stokes equations for incompressible fluid given in Eq. (1) can be written in a compact form as

$$\nabla \cdot \mathbf{u} = 0 \tag{2a}$$

$$\partial \mathbf{u}/\partial t = -\mathbf{u} \cdot \nabla \mathbf{u} - \frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\nabla^2 \mathbf{u} \tag{2b}$$

with

$$\mathbf{u} = [u \quad v]^T \tag{2c}$$

where $\mathbf{u}$ is the 2D velocity vector.

Pressure Poisson equation

In many particle-based methods, the pressure field can be obtained by solving the pressure Poisson equation. This pressure field is then used to correct the velocity field to make it divergence-free (Fourtakas et al., 2018; Hosseini and Feng, 2011; Lind and Stansby, 2016; Nasar et al., 2020; Shadloo et al., 2012; Shao and Lo, 2003; Solenthaler and Pajarola, 2009). The pressure Poisson equation is obtained by taking the divergence of momentum equation given in Eq. (2b) as

$$\nabla \cdot \left(\frac{\partial \mathbf{u}}{\partial t}\right) = -\nabla \cdot (\mathbf{u} \cdot \nabla \mathbf{u}) - \frac{1}{\rho}\nabla^2 p + \frac{\mu}{\rho}\nabla \cdot (\nabla^2 \mathbf{u}) \tag{3a}$$

which can be rearranged as

$$\nabla^2 p = -\rho\nabla \cdot (\partial \mathbf{u}/\partial t) - \rho\nabla \cdot (\mathbf{u} \cdot \nabla \mathbf{u}) - +\mu\nabla \cdot (\nabla^2 \mathbf{u}) \tag{3b}$$

Eq. (3b) is known as the pressure Poisson equation in the continuum level. In a discretized model, to obtain the Eulerian form of the pressure Poisson equation, first, the continuity and momentum equations given in Eq. (2) for incompressible fluid is written in the numerical form as (Fourtakas et al., 2018; Lind and Stansby, 2016; Nasar et al., 2020)

$$\nabla \cdot \mathbf{u}^{(n)} = 0 \tag{4a}$$

$$(\mathbf{u}^{(n)} - \mathbf{u}^{(n-1)})/\Delta t = \left(-\mathbf{u}^{(n-1)} \cdot \nabla \mathbf{u}^{(n-1)} - \frac{1}{\rho}\nabla p^{(n)} + \frac{\mu}{\rho}\nabla^2 \mathbf{u}^{(n-1)}\right) \tag{4b}$$

The momentum equation in Eq. (4b) can be rewritten as

$$\mathbf{u}^{(n)} = \mathbf{u}^{(n-1)} + \Delta t\left(-\mathbf{u}^{(n-1)} \cdot \nabla \mathbf{u}^{(n-1)} - \frac{1}{\rho}\nabla p^{(n)} + \frac{\mu}{\rho}\nabla^2 \mathbf{u}^{(n-1)}\right) \tag{5}$$

where $\mathbf{u}^{(n-1)}$ represents the known velocity field at the previous time step, and $\mathbf{u}^{(n)}$ represents the unknown velocity field at the current time step. The term $p^{(n)}$ represents the unknown pressure field at the current time step and Δt represents time step size.

Taking the divergence of momentum equation given in Eq. (5) results in

$$\nabla \cdot \mathbf{u}^{(n)} = \nabla \cdot \mathbf{u}^{(n-1)} + \Delta t \left(-\nabla \cdot \left(\mathbf{u}^{(n-1)} \cdot \nabla \mathbf{u}^{(n-1)} \right) - \frac{1}{\rho} \nabla^2 p^{(n)} + \frac{\mu}{\rho} \nabla \cdot \left(\nabla^2 \mathbf{u}^{(n-1)} \right) \right) \tag{6}$$

By enforcing $\nabla \cdot \mathbf{u}^{(n)} = 0$ due to the continuity equation given in Eq. (4a), Eq. (6) reduces to

$$\nabla^2 p^{(n)} = \frac{\rho}{\Delta t} \nabla \cdot \mathbf{u}^{(n-1)} - \rho \nabla \cdot \left(\mathbf{u}^{(n-1)} \cdot \nabla \mathbf{u}^{(n-1)} \right) + \mu \nabla \cdot \left(\nabla^2 \mathbf{u}^{(n-1)} \right) \tag{7}$$

Eq. (7) is known as the pressure Poisson equation which can be used to obtain the pressure field (Matsuno et al., 2003; Shirokoff, 2011; Sohn and Heinrich, 1990). Note that compared to the pressure Poisson equation for the Lagrangian approach as presented in (Hosseini and Feng, 2011; Shadloo et al., 2012; Shao and Lo, 2003; Solenthaler and Pajarola, 2009), Eq. (7) has two additional terms, $\rho \nabla \cdot \left(\mathbf{u}^{(n-1)} \cdot \nabla \mathbf{u}^{(n-1)} \right)$ and $\mu \nabla \cdot \left(\nabla^2 \mathbf{u}^{(n-1)} \right)$, which arise from Eulerian approach.

In this study, the Navier-Stokes equations are solved by two steps. First, the pressure field at time t is obtained by solving the pressure Poisson equation given in Eq. (7) (Fourtakas et al., 2018; Hosseini and Feng, 2011; Lind and Stansby, 2016; Nasar et al., 2020; Shadloo et al., 2012; Shao and Lo, 2003; Solenthaler and Pajarola, 2009). The pressure is subjected to boundary conditions described in Section 4.

Next, by using the obtained pressure at current time t, the velocity field at the current time t can be calculated as

$$\mathbf{u}^{(n)} = \mathbf{u}^{(n-1)} + \Delta t \left(-\mathbf{u}^{(n-1)} \cdot \nabla \mathbf{u}^{(n-1)} - \frac{1}{\rho} \nabla p^{(n)} + \frac{\mu}{\rho} \nabla^2 \mathbf{u}^{(n-1)} \right) \tag{8}$$

The velocity boundary conditions are described in Section 4. The above two steps are repeated for every time step until the final solution is obtained.

3. Eulerian incompressible fluid model using peridynamic differential operators

In this section, first, the formulations of peridynamic differential operators introduced by (Madenci et al., 2019a); Madenci et al. (2016) are summarized in Section 3.1. Second, the peridynamic differential operators for momentum equations and pressure Poisson equation as given in Eqs. (6-7) are presented in Section 3.2. Next, an implicit solution for the pressure Poisson equation in the Eulerian incompressible PDDO model is presented in Section 3.3. Finally, the numerical procedure to solve Navier-Stokes equations using the proposed Eulerian incompressible PDDO model is presented in Section 3.4.

3.1. Peridynamic differential operators in two-dimensional space

In this section, the peridynamic differential operators for the first and second-order derivatives in a two-dimensional (2D) space are presented. First, let f be a function in a 2D space as

$$f = f(x, y) \tag{9}$$

According to (Madenci et al., 2019a); Madenci et al. (2016), the first and second order derivatives of f at point k can be presented as

$$f_{,x(k)} = \sum_{j=1}^{N}\big(f_{(j)} - f_{(k)}\big)g_{2(k)(j)}^{10}A_{(j)} \tag{10a}$$

$$f_{,y(k)} = \sum_{j=1}^{N}\big(f_{(j)} - f_{(k)}\big)g_{2(k)(j)}^{01}A_{(j)} \tag{10b}$$

and

$$f_{,xx(k)} = \sum_{j=1}^{N}\big(f_{(j)} - f_{(k)}\big)g_{2(k)(j)}^{20}A_{(j)} \tag{10c}$$

$$f_{,yy(k)} = \sum_{j=1}^{N}\big(f_{(j)} - f_{(k)}\big)g_{2(k)(j)}^{02}A_{(j)} \tag{10d}$$

$$f_{,xy(k)} = \sum_{j=1}^{N}\big(f_{(j)} - f_{(k)}\big)g_{2(k)(j)}^{11}A_{(j)} \tag{10e}$$

where $f_{,x(k)}$ and $f_{,y(k)}$ are first-order derivatives of f with respect to coordinates x and y, respectively. In Eq. (10), j is a family member of particle (node) k. The term $A_{(j)}$ represents the area of node j. The terms $f_{(j)}$ and $f_{(k)}$ represent the values of function f at the coordinates of nodes k and j, respectively, as

$$f_{(k)} = f\big(x_{(k)}, y_{(k)}\big) \tag{11a}$$

$$f_{(j)} = f\big(x_{(j)}, y_{(j)}\big) \tag{11b}$$

In Eqs. (10a, b), the terms $g_{2(k)(j)}^{10}$ and $g_{2(k)(j)}^{01}$ represent the peridynamic orthogonal functions for first-order derivatives. Meanwhile, the terms $g_{2(k)(j)}^{20}$, $g_{2(k)(j)}^{02}$, and $g_{2(k)(j)}^{11}$ in Eqs. (10c-e) represent the peridynamic orthogonal functions for second-order derivatives. These peridynamic orthogonal functions can be expressed as (Madenci et al., 2019a; Madenci et al., 2016)

$$g_{2(k)(j)}^{10}(\boldsymbol{\xi}) = \omega(|\boldsymbol{\xi}|)(a_1\xi_1 + a_2\xi_2 + a_3\xi_1^2 + a_4\xi_2^2 + a_5\xi_1\xi_2) \tag{12a}$$

$$g_{2(k)(j)}^{01}(\boldsymbol{\xi}) = \omega(|\boldsymbol{\xi}|)(a_6\xi_1 + a_7\xi_2 + a_8\xi_1^2 + a_9\xi_2^2 + a_{10}\xi_1\xi_2) \tag{12b}$$

$$g_{2(k)(j)}^{20}(\boldsymbol{\xi}) = \omega(|\boldsymbol{\xi}|)(a_{11}\xi_1 + a_{12}\xi_2 + a_{13}\xi_1^2 + a_{14}\xi_2^2 + a_{15}\xi_1\xi_2) \tag{12c}$$

$$g_{2(k)(j)}^{02}(\boldsymbol{\xi}) = \omega(|\boldsymbol{\xi}|)(a_{16}\xi_1 + a_{17}\xi_2 + a_{18}\xi_1^2 + a_{19}\xi_2^2 + a_{20}\xi_1\xi_2) \tag{12d}$$

$$g_{2(k)(j)}^{11}(\boldsymbol{\xi}) = \omega(|\boldsymbol{\xi}|)(a_{21}\xi_1 + a_{22}\xi_2 + a_{23}\xi_1^2 + a_{24}\xi_2^2 + a_{25}\xi_1\xi_2) \tag{12e}$$

with

$$\boldsymbol{\xi} = \mathbf{x}_{(j)} - \mathbf{x}_{(k)} = \big(x_{(j)} - x_{(k)}, y_{(j)} - y_{(k)}\big) \tag{12f}$$

$$\xi_1 = x_{(j)} - x_{(k)} \tag{12g}$$

$$\xi_2 = y_{(j)} - y_{(k)} \tag{12h}$$

$$|\boldsymbol{\xi}| = \sqrt{\xi_1^2 + \xi_2^2} \tag{12i}$$

$$\omega(|\boldsymbol{\xi}|) = e^{-\left(\frac{2|\boldsymbol{\xi}|}{\delta}\right)^2} \tag{12j}$$

where ξ_1 and ξ_2 are the relative coordinates between nodes k and j, and $\omega(|\boldsymbol{\xi}|)$ represents the peridynamic weight function (Madenci et al., 2019a; Madenci et al., 2016).

In Eqs. (12a-e), the terms, $a_1, a_2, \cdots, a_{25}$, represent the components of matrix **a** that can be defined as (Madenci et al., 2019a; Madenci et al., 2016)

$$\mathbf{a} = \begin{bmatrix} a_1 & a_6 & a_{11} & a_{16} & a_{21} \\ a_2 & a_7 & a_{12} & a_{17} & a_{22} \\ a_3 & a_8 & a_{13} & a_{18} & a_{23} \\ a_4 & a_9 & a_{14} & a_{19} & a_{24} \\ a_5 & a_{10} & a_{15} & a_{20} & a_{25} \end{bmatrix} \tag{13}$$

The matrix **a** in Eq. (13) can be obtained by solving the following equation as (Madenci et al., 2019a; Madenci et al., 2016)

$$\mathbf{Aa} = \mathbf{b} \tag{14a}$$

with

$$\mathbf{A} = \sum_{j=1}^{N} \omega(|\boldsymbol{\xi}|) \begin{bmatrix} \xi_1^2 & \xi_1\xi_2 & \xi_1^3 & \xi_1\xi_2^2 & \xi_1^2\xi_2 \\ \xi_1\xi_2 & \xi_2^2 & \xi_1^2\xi_2 & \xi_2^3 & \xi_1\xi_2^2 \\ \xi_1^3 & \xi_1^2\xi_2 & \xi_1^4 & \xi_1^2\xi_2^2 & \xi_1^3\xi_2 \\ \xi_1\xi_2^2 & \xi_2^3 & \xi_1^2\xi_2^2 & \xi_2^4 & \xi_1\xi_2^3 \\ \xi_1^2\xi_2 & \xi_1\xi_2^2 & \xi_1^3\xi_2 & \xi_1\xi_2^3 & \xi_1^2\xi_2^2 \end{bmatrix} A_{(j)} \tag{14b}$$

and

$$\mathbf{b} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{14c}$$

By using the first-order derivatives given in Eqs. (10a, b), the peridynamic gradient operator for function f can be represented as

$$\nabla f_{(k)} = \begin{bmatrix} f_{,x(k)} \\ f_{,y(k)} \end{bmatrix} = \sum_{j=1}^{N} A_{(j)} \begin{bmatrix} -g_{2(k)(j)}^{10} & g_{2(k)(j)}^{10} \\ -g_{2(k)(j)}^{01} & g_{2(k)(j)}^{01} \end{bmatrix} \begin{bmatrix} f_{(k)} \\ f_{(j)} \end{bmatrix} \tag{15}$$

By using the second-order derivatives given in Eqs. (10c, d), the peridynamic Laplace operator for function f can be represented as

$$\nabla^2 f_{(k)} = f_{,xx(k)} + f_{,yy(k)} = \sum_{j=1}^{N} \left(g_{2(k)(j)}^{20} + g_{2(k)(j)}^{02}\right) A_{(j)} \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} f_{(k)} \\ f_{(j)} \end{bmatrix} \tag{16}$$

Similarly, by using the first-order derivatives given in Eqs. (10a-b), the peridynamic gradient operator and peridynamic divergence operator for a vector $\mathbf{f} = [f_1 \quad f_2]^T$ can be represented as

$$\nabla \mathbf{f}_{(k)} = \begin{bmatrix} f_{1,x(k)} & f_{2,x(k)} \\ f_{1,y(k)} & f_{2,y(k)} \end{bmatrix} = \sum_{j=1}^{N} A_{(j)} \begin{bmatrix} -g_{2(k)(j)}^{10} & g_{2(k)(j)}^{10} \\ -g_{2(k)(j)}^{01} & g_{2(k)(j)}^{01} \end{bmatrix} \begin{bmatrix} f_{1(k)} & f_{2(k)} \\ f_{1(j)} & f_{2(j)} \end{bmatrix} \tag{17}$$

$$\nabla \cdot \mathbf{f}_{(k)} = f_{1,x(k)} + f_{2,y(k)} = \sum_{j=1}^{N} A_{(j)} \begin{bmatrix} -g_{2(k)(j)}^{10} & g_{2(k)(j)}^{10} & -g_{2(k)(j)}^{01} & g_{2(k)(j)}^{01} \end{bmatrix} \begin{bmatrix} f_{1(k)} \\ f_{1(j)} \\ f_{2(k)} \\ f_{2(j)} \end{bmatrix} \tag{18}$$

By using the second-order derivatives given in Eqs. (10c-d), the PD Laplace operator for a vector $\mathbf{f} = [f_1 \quad f_2]^T$ can be represented as

$$\nabla^2 \mathbf{f}_{(k)} = \begin{bmatrix} f_{1,xx(k)} + f_{1,yy(k)} \\ f_{2,xx(k)} + f_{2,yy(k)} \end{bmatrix} = \sum_{j=1}^{N} \left(g^{20}_{2(k)(j)} + g^{02}_{2(k)(j)} \right) A_{(j)} \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} f_{1(k)} \\ f_{1(j)} \\ f_{2(k)} \\ f_{2(j)} \end{bmatrix} \tag{19}$$

As given in Eq. (14b) and (12j), the formulations of $\mathbf{A}$ and weight function $\omega(|\xi|)$ include length-scale, δ, which is the horizon size in PDDO. Therefore, the terms, $a_1, a_2, \cdots, a_{25}$ given in Eq. (13) and Eq. (14a), and the PDDO orthogonal functions, $g^{10}_{2(k)(j)}$, $g^{01}_{2(k)(j)}$, $g^{20}_{2(k)(j)}$, $g^{02}_{2(k)(j)}$, $g^{11}_{2(k)(j)}$ given in Eq. (12) include length-scale, δ. Thus, the length-scale, δ, still exists in the PDDO for the first order and second order derivatives given in Eq. (10). A discussion about the effects of horizon size δ on the accuracy of PDDO estimations was also presented in (Madenci et al., 2019a; Madenci et al., 2016).

3.2. Peridynamic differential operators for Navier-Stokes equations

In this section, the PDDO for the calculations of the terms, $\nabla \mathbf{u}^{(n-1)}$, $\nabla^2 \mathbf{u}^{(n-1)}$, $\nabla^2 p^{(n)}$, $\nabla p^{(n)}$ in Eqs. (6-7) are presented. First, by using the operators given in Eq. (17) and Eq. (19), the gradient, $\nabla \mathbf{u}^{(n-1)}$ and Laplace, $\nabla^2 \mathbf{u}^{(n-1)}$, of the velocity $\mathbf{u}^{(n-1)}$ in Eqs. (6-7) can be calculated as

$$\nabla \mathbf{u}^{(n-1)}_{(k)} = \begin{bmatrix} u^{(n-1)}_{,x(k)} & v^{(n-1)}_{,x(k)} \\ u^{(n-1)}_{,y(k)} & v^{(n-1)}_{,y(k)} \end{bmatrix} = \sum_{j=1}^{N} A_{(j)} \begin{bmatrix} -g^{10}_{2(k)(j)} & g^{10}_{2(k)(j)} \\ -g^{01}_{2(k)(j)} & g^{01}_{2(k)(j)} \end{bmatrix} \begin{bmatrix} u^{(n-1)}_{(k)} & v^{(n-1)}_{(k)} \\ u^{(n-1)}_{(j)} & v^{(n-1)}_{(j)} \end{bmatrix} \tag{20}$$

$$\nabla^2 \mathbf{u}^{(n-1)}_{(k)} = \begin{bmatrix} u^{(n-1)}_{,xx(k)} + u^{(n-1)}_{,yy(k)} \\ v^{(n-1)}_{,xx(k)} + v^{(n-1)}_{,yy(k)} \end{bmatrix} = \sum_{j=1}^{N} \left(g^{20}_{2(k)(j)} + g^{02}_{2(k)(j)} \right) A_{(j)} \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} u^{(n-1)}_{(k)} \\ u^{(n-1)}_{(j)} \\ v^{(n-1)}_{(k)} \\ v^{(n-1)}_{(j)} \end{bmatrix} \tag{21}$$

Second, by using Eq. (18), the divergence of the velocity vector, $\nabla \cdot \mathbf{u}^{(n-1)}$, in Eq. (7) can be represented as

$$\nabla \cdot \mathbf{u}^{(n-1)}_{(k)} = u^{(n-1)}_{,x(k)} + v^{(n-1)}_{,y(k)} = \sum_{j=1}^{N} A_{(j)} \begin{bmatrix} -g^{10}_{2(k)(j)} & g^{10}_{2(k)(j)} & -g^{01}_{2(k)(j)} & g^{01}_{2(k)(j)} \end{bmatrix} \begin{bmatrix} u^{(n-1)}_{(k)} \\ u^{(n-1)}_{(j)} \\ v^{(n-1)}_{(k)} \\ v^{(n-1)}_{(j)} \end{bmatrix} \tag{22}$$

To calculate the divergence $\nabla \cdot \left(\mathbf{u}^{(n-1)} \cdot \nabla \mathbf{u}^{(n-1)} \right)$ in Eq. (7), the term $\mathbf{u}^{(n-1)} \cdot \nabla \mathbf{u}^{(n-1)}$ can be represented as

$$\mathbf{d}^{(n-1)} = \mathbf{u}^{(n-1)} \cdot \nabla \mathbf{u}^{(n-1)} = \begin{bmatrix} u^{(n-1)} u^{(n-1)}_{,x} + v^{(n-1)} u^{(n-1)}_{,y} \\ u^{(n-1)} v^{(n-1)}_{,x} + v^{(n-1)} v^{(n-1)}_{,y} \end{bmatrix} \tag{23}$$

Therefore, the divergence $\nabla \cdot \left(\mathbf{u}^{(n-1)} \cdot \nabla \mathbf{u}^{(n-1)} \right)$ in Eq. (7) can be represented as

$$\nabla \cdot \left(\mathbf{u}^{(n-1)} \cdot \nabla \mathbf{u}^{(n-1)}\right) = \nabla \cdot \mathbf{d}^{(n-1)} = d_{1,x}^{(n-1)} + d_{2,y}^{(n-1)} \tag{24a}$$

with

$$d_{1}^{(n-1)} = u^{(n-1)} u_{,x}^{(n-1)} + v^{(n-1)} u_{,y}^{(n-1)} \tag{24b}$$

$$d_{2(k)}^{(n-1)} = u^{(n-1)} v_{,x}^{(n-1)} + v^{(n-1)} v_{,y}^{(n-1)} \tag{24c}$$

In the discretized model, by using Eq. (18), the divergence $\nabla \cdot \left(\mathbf{u}^{(n-1)} \cdot \nabla \mathbf{u}^{(n-1)}\right)$ in Eq. (7) can be represented as

$$\nabla \cdot \left(\mathbf{u}_{(k)}^{(n-1)} \cdot \nabla \mathbf{u}_{(k)}^{(n-1)}\right) = \nabla \cdot \mathbf{d}_{(k)}^{(n-1)} = d_{1,x(k)}^{(n-1)} + d_{2,y(k)}^{(n-1)} =$$

$$\sum_{j=1}^{N} A_{(j)} \begin{bmatrix} -g_{2(k)(j)}^{10} & g_{2(k)(j)}^{10} & -g_{2(k)(j)}^{01} & g_{2(k)(j)}^{01} \end{bmatrix} \begin{bmatrix} d_{1(k)}^{(n-1)} \\ d_{1(j)}^{(n-1)} \\ d_{2(k)}^{(n-1)} \\ d_{2(j)}^{(n-1)} \end{bmatrix} \tag{25a}$$

with

$$d_{1(k)}^{(n-1)} = u_{(k)}^{(n-1)} u_{,x(k)}^{(n-1)} + v_{(k)}^{(n-1)} u_{,y(k)}^{(n-1)} \tag{25b}$$

$$d_{2(k)}^{(n-1)} = u_{(k)}^{(n-1)} v_{,x(k)}^{(n-1)} + v_{(k)}^{(n-1)} v_{,y(k)}^{(n-1)} \tag{25c}$$

$$d_{1(j)}^{(n-1)} = u_{(j)}^{(n-1)} u_{,x(j)}^{(n-1)} + v_{(j)}^{(n-1)} u_{,y(j)}^{(n-1)} \tag{25d}$$

$$d_{2(j)}^{(n-1)} = u_{(j)}^{(n-1)} v_{,x(j)}^{(n-1)} + v_{(j)}^{(n-1)} v_{,y(j)}^{(n-1)} \tag{25e}$$

Similarly, to calculate the divergence $\nabla \cdot \left(\nabla^2 \mathbf{u}^{(n-1)}\right)$ in Eq. (7), first, the Laplace operator of the velocity field, $\nabla^2 \mathbf{u}^{(n-1)}$ can be calculated by using Eq. (21) as

$$\mathbf{L}^{(n-1)} = \nabla^2 \mathbf{u}^{(n-1)} = \begin{bmatrix} u_{,xx}^{(n-1)} + u_{,yy}^{(n-1)} \\ v_{,xx}^{(n-1)} + v_{,yy}^{(n-1)} \end{bmatrix} \tag{26}$$

Therefore, the divergence $\nabla \cdot \left(\nabla^2 \mathbf{u}^{(n-1)}\right)$ in Eq. (7) can be represented as

$$\nabla \cdot \left(\nabla^2 \mathbf{u}^{(n-1)}\right) = \nabla \cdot \mathbf{L}^{(n-1)} = L_{1,x}^{(n-1)} + L_{2,y}^{(n-1)} \tag{27a}$$

with

$$L_{1}^{(n-1)} = u_{,xx}^{(n-1)} + u_{,yy}^{(n-1)} \tag{27b}$$

$$L_{2}^{(n-1)} = v_{,xx}^{(n-1)} + v_{,yy}^{(n-1)} \tag{27c}$$

In the discretized model, by using Eq. (18), the divergence $\nabla \cdot \left(\nabla^2 \mathbf{u}^{(n-1)}\right)$ in Eq. (7) can be represented as

$$\nabla \cdot \left(\nabla^2 \mathbf{u}_{(k)}^{(n-1)}\right) = \nabla \cdot \mathbf{L}_{(k)}^{(n-1)} = L_{1,x(k)}^{(n-1)} + L_{2,y(k)}^{(n-1)} =$$

$$\sum_{j=1}^{N} A_{(j)} \begin{bmatrix} -g_{2(k)(j)}^{10} & g_{2(k)(j)}^{10} & -g_{2(k)(j)}^{01} & g_{2(k)(j)}^{01} \end{bmatrix} \begin{bmatrix} L_{1(k)}^{(n-1)} \\ L_{1(j)}^{(n-1)} \\ L_{2(k)}^{(n-1)} \\ L_{2(j)}^{(n-1)} \end{bmatrix} \tag{28a}$$

with

$$L_{1(k)}^{(n-1)} = u_{,xx(k)}^{(n-1)} + u_{,yy(k)}^{(n-1)} \tag{28b}$$

$$L_{2(k)}^{(n-1)} = v_{,xx(k)}^{(n-1)} + v_{,yy(k)}^{(n-1)} \tag{28c}$$

$$L_{1(j)}^{(n-1)} = u_{,xx(j)}^{(n-1)} + u_{,yy(j)}^{(n-1)} \tag{28d}$$

$$L_{2(j)}^{(n-1)} = v_{,xx(j)}^{(n-1)} + v_{,yy(j)}^{(n-1)} \tag{28e}$$

Finally, by using Eqs. (15-16), the gradient and Laplace of pressure, $\nabla p^{(n)}$ and $\nabla^2 p^{(n)}$, in Eqs. (6-7) can be represented as

$$\nabla p_{(k)}^{(n)} = \begin{bmatrix} p_{,x(k)}^{(n)} \\ p_{,y(k)}^{(n)} \end{bmatrix} = \sum_{j=1}^{N} A_{(j)} \begin{bmatrix} -g_{2(k)(j)}^{10} & g_{2(k)(j)}^{10} \\ -g_{2(k)(j)}^{01} & g_{2(k)(j)}^{01} \end{bmatrix} \begin{bmatrix} p_{(k)}^{(n)} \\ p_{(j)}^{(n)} \end{bmatrix} \tag{29}$$

$$\nabla^2 p_{(k)}^{(n)} = p_{,xx(k)}^{(n)} + p_{,yy(k)}^{(n)} = \sum_{j=1}^{N} A_{(j)} \left(g_{2(k)(j)}^{20} + g_{2(k)(j)}^{02}\right) \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} p_{(k)}^{(n)} \\ p_{(j)}^{(n)} \end{bmatrix} \tag{30}$$

3.3. Implicit solution for pressure Poisson equation

In this study, the pressure Poisson equation given in Eq. (7) is solved by using the implicit solution. First, Eq. (7) can be rewritten in a matrix form for all the particles in the fluid domain as

$$\mathbf{K}_L \mathbf{P}^{(n)} = \mathbf{R}^{(n)} \tag{31a}$$

with

$$\mathbf{P}^{(n)} = \begin{bmatrix} p_{(1)}^{(n)} & \cdots & p_{(k)}^{(n)} & \cdots & p_{(N_p)}^{(n)} \end{bmatrix}^T \tag{31b}$$

and

$$\mathbf{K}_L = \sum_{k=1}^{N_p} \left(\sum_{j=1}^{N} A_{(j)} \left(g_{2(k)(j)}^{20} + g_{2(k)(j)}^{02}\right) \begin{bmatrix} -1 & 1 \end{bmatrix} \right) \tag{31c}$$

where $\mathbf{P}^{(n)}$ represents the pressure vector for the fluid domain at the current time step, n. This vector has a size of $(N_p \times 1)$, where N_p is the total number of particles (nodes) in the fluid domain. In Eqs. (31a, c), $\mathbf{K}_L$ is a matrix that can be used to represent the Laplace operator as

$$\nabla^2 \mathbf{P}^{(n)} = \mathbf{K}_L \mathbf{P}^{(n)} \tag{32}$$

Matrix $\mathbf{K}_L$ is a square matrix with a size of $(N_p \times N_p)$. The numerical procedure for the calculation of $\mathbf{K}_L$ is presented in Fig. 4.

In Eq. (31a), the parameter $\mathbf{R}^{(n)}$ is a vector with a size of $(N_p \times 1)$. This vector represents the right-hand side terms in Eq. (7) as

$$\mathbf{R}^{(n)} = \begin{bmatrix} r_{(1)}^{(n)} & \cdots & r_{(k)}^{(n)} & \cdots & r_{N_p}^{(n)} \end{bmatrix}^T \tag{33a}$$

with

$$r_{(k)}^{(n)} = \frac{\rho}{\Delta t}\nabla \cdot \mathbf{u}_{(k)}^{(n-1)} - \rho\nabla \cdot \left(\mathbf{u}_{(k)}^{(n-1)} \cdot \nabla\mathbf{u}_{(k)}^{(n-1)}\right) + \mu\nabla \cdot \left(\nabla^2\mathbf{u}_{(k)}^{(n-1)}\right) \tag{33b}$$

where $\mathbf{u}_{(k)}^{(n-1)}$ is the velocity vector of particle k at the previous time step $(n-1)$. The calculation of $\mathbf{R}^{(n)}$ at each time step is presented in Fig. 5.

The pressure Poisson equation in Eq. (31a) is subjected to the boundary conditions presented in Section 4. In this study, the numerical simulations are implemented in MATLAB version R2018b and the pressure field, $\mathbf{P}^{(n)}$, is obtained by using the backslash (\) operator as

$$\mathbf{P}^{(n)} = \mathbf{K}_L \backslash \mathbf{R}^{(n)} \tag{34}$$

3.4. Numerical procedure

First, the fluid domain is discretized into particles and the family member search for each particle is conducted with a horizon size of δ. Next, the orthogonal functions $g_{2(k)(j)}^{10}$, $g_{2(k)(j)}^{01}$, $g_{2(k)(j)}^{20}$, $g_{2(k)(j)}^{02}$, and $g_{2(k)(j)}^{11}$ for each interaction are calculated using Eq. (12). Details of the calculation for these orthogonal functions are presented in Fig. 2. Next, the matrix $\mathbf{K}_L$ given in Eq. (31c) is calculated as shown in Fig. 4.

At each time step, the right-hand side term, $\mathbf{R}^{(n)}$, of the pressure Poisson equation given in Eq. (31a) are calculated by using Eq. (33). Details of the calculation for these terms are presented in Fig. 5. Next, the pressure Poisson equation given in Eq. (31a) is solved by using the implicit method as presented in Eq. (34) to obtain the pressure field, $\mathbf{P}^{(n)}$. Finally, the velocity field at the current time step, $\mathbf{u}^{(n)}$ is calculated by using Eq. (8). Details of this step are presented in Fig. 6.

The numerical procedure in numerical simulations includes the following steps:

Step 1: Create geometry for the fluid domain and provide the fluid parameters including fluid density, ρ, and fluid viscosity, μ.

Step 2: Discretize the fluid domain.

Step 3: Find family members for each particle within a horizon size of δ.

Step 4: Calculate orthogonal functions $g_{2(k)(j)}^{10}$, $g_{2(k)(j)}^{01}$, $g_{2(k)(j)}^{20}$, $g_{2(k)(j)}^{02}$, $g_{2(k)(j)}^{11}$ for each interaction by using Eq. (12). Details of this step are presented in Fig. 2.

Step 5: Calculate matrix $\mathbf{K}_L$ which represents the Laplace operator for the pressure field (see Fig. 4).

Step 6: Initialize velocity and pressure vectors.

Step 7: Loop over time steps: $n = 1,2, \dots, N_t$, where N_t is the total number of time steps.

Step 7.1: Apply boundary conditions for the inlet, outlet, and no-slip boundary conditions for stationary walls.

Step 7.2: Using the known velocity field at the previous time step, $\mathbf{u}^{(n-1)}$, to calculate the terms on the right-hand side of the Pressure Poisson equation given in Eq. (7) and Eq. (31a): $\nabla\mathbf{u}_{(k)}^{(n-1)}$, $\nabla^2\mathbf{u}_{(k)}^{(n-1)}$, $\nabla\cdot\mathbf{u}_{(k)}^{(n-1)}$, $\mathbf{d}_{(k)}^{(n-1)}$, $\nabla\cdot\left(\mathbf{u}_{(k)}^{(n-1)}\cdot\nabla\mathbf{u}_{(k)}^{(n-1)}\right)$, $\nabla\cdot\left(\nabla^2\mathbf{u}_{(k)}^{(n-1)}\right)$, $\mathbf{R}^{(n)}$ (see Fig. 5)

Step 7.3: Solving Eq. (31a) to obtain pressure field at current time step, $\mathbf{P}^{(n)}$ by using Eq. (34): $\mathbf{P}^{(n)} = \mathbf{K}_L \backslash \mathbf{R}^{(n)}$

Step 7.4: Calculate velocity at current time step, $\mathbf{u}^{(n)}$, by using Eq. (8) (see Fig. 6):

$$\mathbf{u}_{(k)}^{(n)} = \mathbf{u}_{(k)}^{(n-1)} + \Delta t\left(-\mathbf{u}_{(k)}^{(n-1)}\cdot\nabla\mathbf{u}_{(k)}^{(n-1)} - \frac{1}{\rho}\nabla p_{(k)}^{(n)} + \frac{\mu}{\rho}\nabla^2\mathbf{u}_{(k)}^{(n-1)}\right) \tag{35}$$

Step 7.5: Go to next time step if $n < N_t$.

Step 8: Output results for pressure and velocity field.

The detailed numerical procedure in numerical simulations for 2D incompressible fluid is shown in Fig. 1.

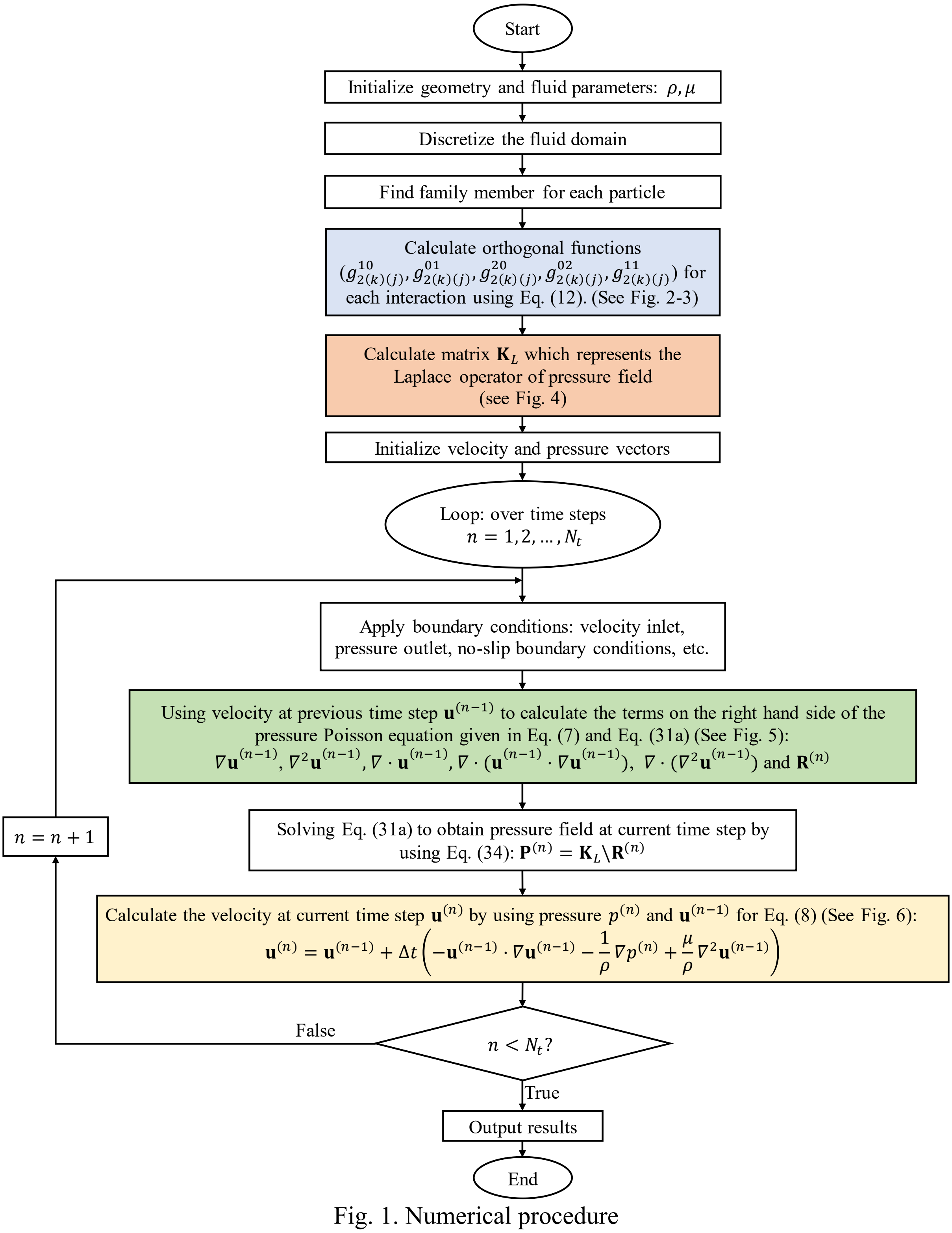
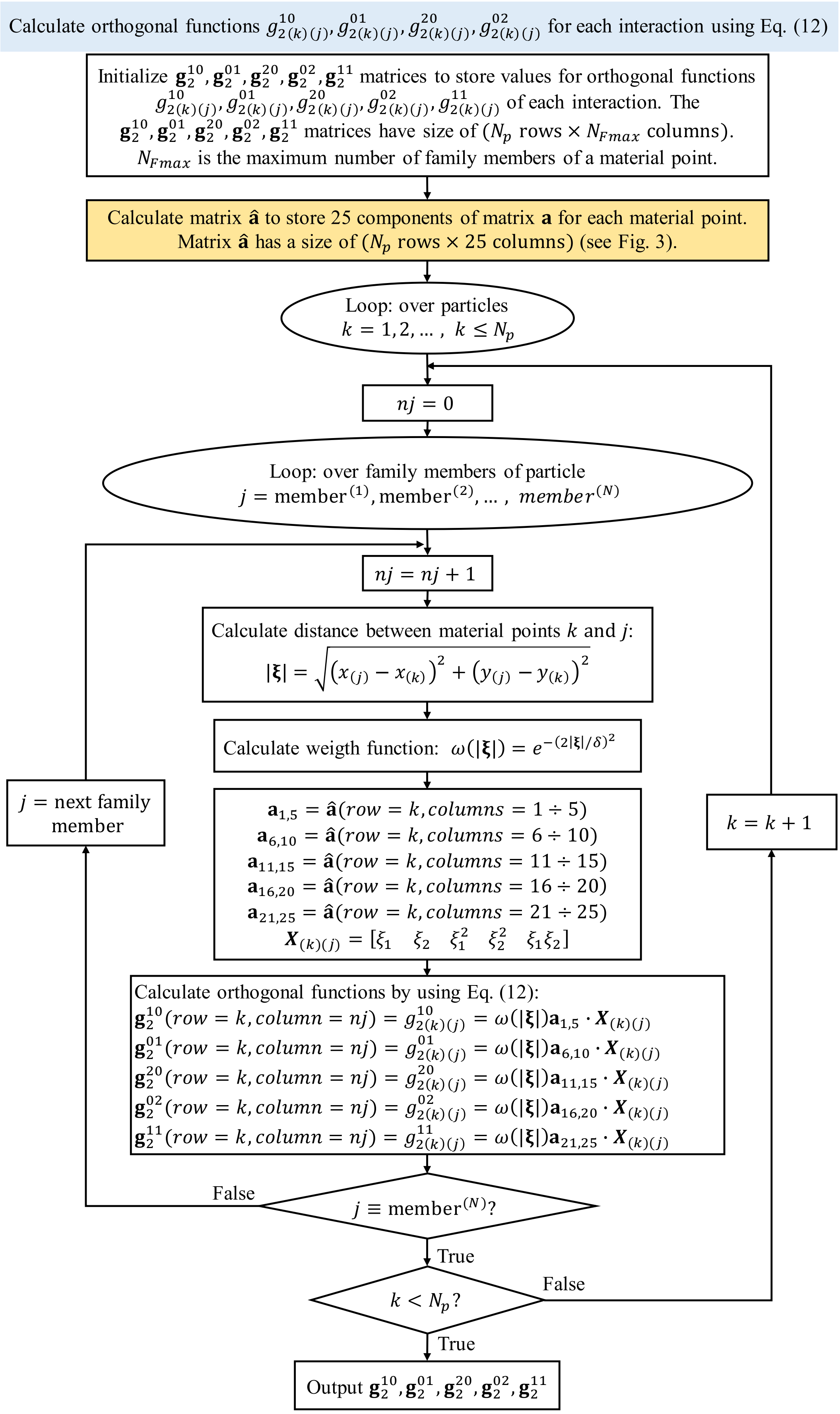

Start
Initialize geometry and fluid parameters: ρ, μ
Discretize the fluid domain
Find family member for each particle
Calculate orthogonal functions $(g^{10}_{2(k)(j)}, g^{01}_{2(k)(j)}, g^{20}_{2(k)(j)}, g^{02}_{2(k)(j)}, g^{11}_{2(k)(j)})$ for each interaction using Eq. (12). (See Fig. 2-3)
Calculate matrix $\mathbf{K}_L$ which represents the Laplace operator of pressure field (see Fig. 4)
Initialize velocity and pressure vectors
Loop: over time steps $n = 1, 2, \dots, N_t$
Apply boundary conditions: velocity inlet, pressure outlet, no-slip boundary conditions, etc.
Using velocity at previous time step $\mathbf{u}^{(n-1)}$ to calculate the terms on the right hand side of the pressure Poisson equation given in Eq. (7) and Eq. (31a) (See Fig. 5): $\nabla \mathbf{u}^{(n-1)}, \nabla^2 \mathbf{u}^{(n-1)}, \nabla \cdot \mathbf{u}^{(n-1)}, \nabla \cdot (\mathbf{u}^{(n-1)} \cdot \nabla \mathbf{u}^{(n-1)}),\ \nabla \cdot (\nabla^2 \mathbf{u}^{(n-1)})$ and $\mathbf{R}^{(n)}$
$n = n + 1$
Solving Eq. (31a) to obtain pressure field at current time step by using Eq. (34): $\mathbf{P}^{(n)} = \mathbf{K}_L \backslash \mathbf{R}^{(n)}$
Calculate the velocity at current time step $\mathbf{u}^{(n)}$ by using pressure $p^{(n)}$ and $\mathbf{u}^{(n-1)}$ for Eq. (8) (See Fig. 6):
$\mathbf{u}^{(n)} = \mathbf{u}^{(n-1)} + \Delta t \left(-\mathbf{u}^{(n-1)} \cdot \nabla \mathbf{u}^{(n-1)} - \frac{1}{\rho} \nabla p^{(n)} + \frac{\mu}{\rho} \nabla^2 \mathbf{u}^{(n-1)} \right)$
False
$n < N_t$?
True
Output results
End

Fig. 1. Numerical procedure



Calculate orthogonal functions $g^{10}_{2(k)(j)}, g^{01}_{2(k)(j)}, g^{20}_{2(k)(j)}, g^{02}_{2(k)(j)}$ for each interaction using Eq. (12)

Initialize $\mathbf{g}^{10}_2, \mathbf{g}^{01}_2, \mathbf{g}^{20}_2, \mathbf{g}^{02}_2, \mathbf{g}^{11}_2$ matrices to store values for orthogonal functions $g^{10}_{2(k)(j)}, g^{01}_{2(k)(j)}, g^{20}_{2(k)(j)}, g^{02}_{2(k)(j)}, g^{11}_{2(k)(j)}$ of each interaction. The $\mathbf{g}^{10}_2, \mathbf{g}^{01}_2, \mathbf{g}^{20}_2, \mathbf{g}^{02}_2, \mathbf{g}^{11}_2$ matrices have size of (N_p rows × N_{Fmax} columns). N_{Fmax} is the maximum number of family members of a material point.

Calculate matrix $\hat{\mathbf{a}}$ to store 25 components of matrix $\mathbf{a}$ for each material point. Matrix $\hat{\mathbf{a}}$ has a size of (N_p rows × 25 columns) (see Fig. 3).

Loop: over particles $k = 1, 2, \ldots, \; k \leq N_p$

$nj = 0$

Loop: over family members of particle $j = \text{member}^{(1)}, \text{member}^{(2)}, \ldots, \; member^{(N)}$

$nj = nj + 1$

Calculate distance between material points k and j:

$$|\boldsymbol{\xi}| = \sqrt{\left(x_{(j)} - x_{(k)}\right)^2 + \left(y_{(j)} - y_{(k)}\right)^2}$$

Calculate weigth function: $\omega(|\boldsymbol{\xi}|) = e^{-(2|\boldsymbol{\xi}|/\delta)^2}$

$$\begin{aligned}
\mathbf{a}_{1,5} &= \hat{\mathbf{a}}(row = k, columns = 1 \div 5)\\
\mathbf{a}_{6,10} &= \hat{\mathbf{a}}(row = k, columns = 6 \div 10)\\
\mathbf{a}_{11,15} &= \hat{\mathbf{a}}(row = k, columns = 11 \div 15)\\
\mathbf{a}_{16,20} &= \hat{\mathbf{a}}(row = k, columns = 16 \div 20)\\
\mathbf{a}_{21,25} &= \hat{\mathbf{a}}(row = k, columns = 21 \div 25)\\
\boldsymbol{X}_{(k)(j)} &= [\xi_1 \quad \xi_2 \quad \xi_1^2 \quad \xi_2^2 \quad \xi_1\xi_2]
\end{aligned}$$

Calculate orthogonal functions by using Eq. (12):

$$\begin{aligned}
\mathbf{g}^{10}_2(row = k, column = nj) &= g^{10}_{2(k)(j)} = \omega(|\boldsymbol{\xi}|)\mathbf{a}_{1,5} \cdot \boldsymbol{X}_{(k)(j)}\\
\mathbf{g}^{01}_2(row = k, column = nj) &= g^{01}_{2(k)(j)} = \omega(|\boldsymbol{\xi}|)\mathbf{a}_{6,10} \cdot \boldsymbol{X}_{(k)(j)}\\
\mathbf{g}^{20}_2(row = k, column = nj) &= g^{20}_{2(k)(j)} = \omega(|\boldsymbol{\xi}|)\mathbf{a}_{11,15} \cdot \boldsymbol{X}_{(k)(j)}\\
\mathbf{g}^{02}_2(row = k, column = nj) &= g^{02}_{2(k)(j)} = \omega(|\boldsymbol{\xi}|)\mathbf{a}_{16,20} \cdot \boldsymbol{X}_{(k)(j)}\\
\mathbf{g}^{11}_2(row = k, column = nj) &= g^{11}_{2(k)(j)} = \omega(|\boldsymbol{\xi}|)\mathbf{a}_{21,25} \cdot \boldsymbol{X}_{(k)(j)}
\end{aligned}$$

$j \equiv \text{member}^{(N)}$?

False

j = next family member

True

$k < N_p$?

False

$k = k + 1$

True

Output $\mathbf{g}^{10}_2, \mathbf{g}^{01}_2, \mathbf{g}^{20}_2, \mathbf{g}^{02}_2, \mathbf{g}^{11}_2$

Fig. 2. Calculate orthogonal functions $g^{10}_{2(k)(j)}$, $g^{01}_{2(k)(j)}$, $g^{20}_{2(k)(j)}$, $g^{02}_{2(k)(j)}$, $g^{11}_{2(k)(j)}$ for each interaction.

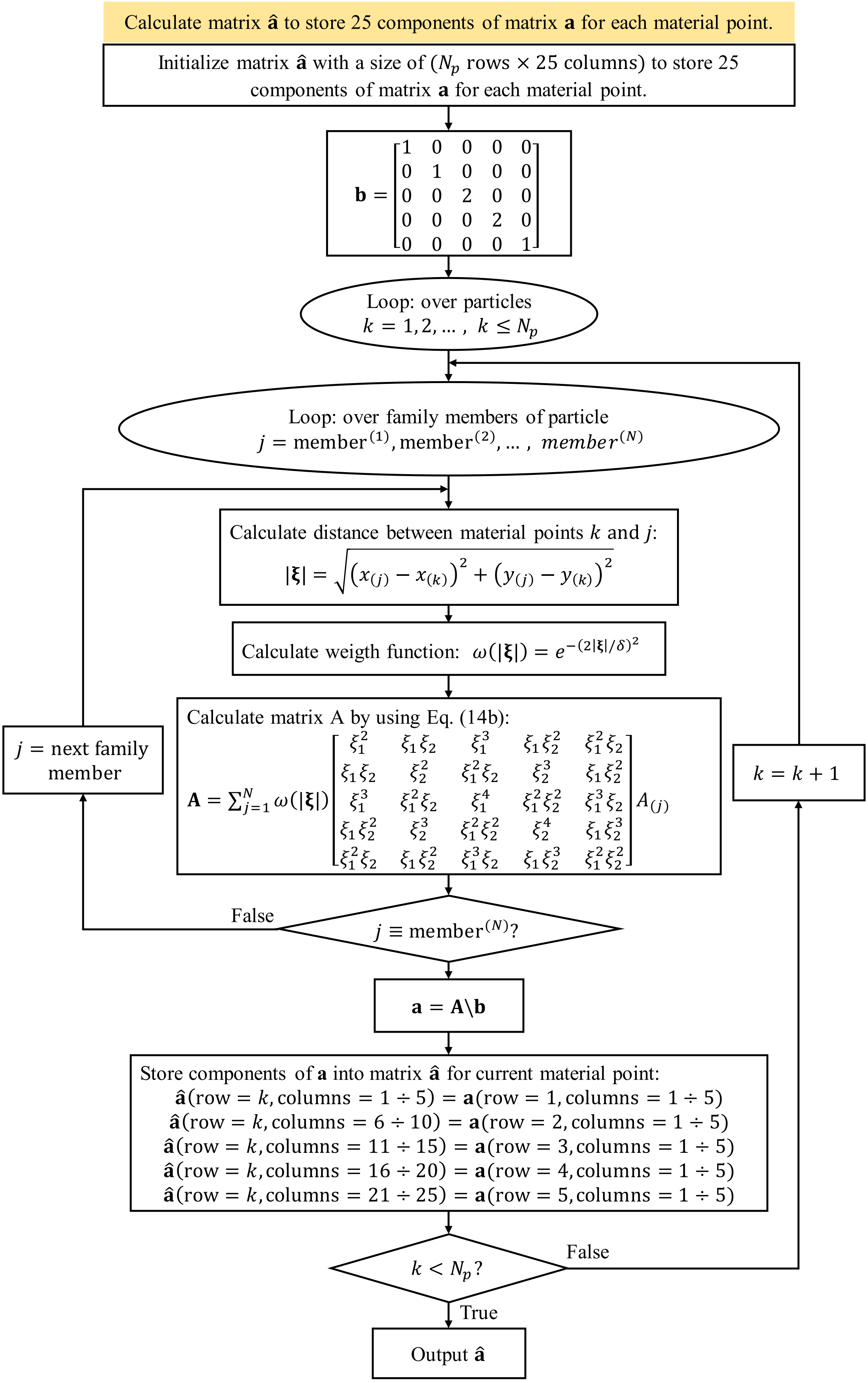

Calculate matrix $\hat{\mathbf{a}}$ to store 25 components of matrix $\mathbf{a}$ for each material point.
Initialize matrix $\hat{\mathbf{a}}$ with a size of (N_p rows × 25 columns) to store 25 components of matrix $\mathbf{a}$ for each material point.
$\mathbf{b} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$
Loop: over particles
$k = 1, 2, \ldots,\ k \le N_p$
Loop: over family members of particle
$j = \mathrm{member}^{(1)}, \mathrm{member}^{(2)}, \ldots,\ member^{(N)}$
Calculate distance between material points k and j:
$|\boldsymbol{\xi}| = \sqrt{\left(x_{(j)} - x_{(k)}\right)^2 + \left(y_{(j)} - y_{(k)}\right)^2}$
Calculate weigth function: $\omega(|\boldsymbol{\xi}|) = e^{-(2|\boldsymbol{\xi}|/\delta)^2}$
Calculate matrix A by using Eq. (14b):
$\mathbf{A} = \sum_{j=1}^{N} \omega(|\boldsymbol{\xi}|) \begin{bmatrix} \xi_1^2 & \xi_1\xi_2 & \xi_1^3 & \xi_1\xi_2^2 & \xi_1^2\xi_2 \\ \xi_1\xi_2 & \xi_2^2 & \xi_1^2\xi_2 & \xi_2^3 & \xi_1\xi_2^2 \\ \xi_1^3 & \xi_1^2\xi_2 & \xi_1^4 & \xi_1^2\xi_2^2 & \xi_1^3\xi_2 \\ \xi_1\xi_2^2 & \xi_2^3 & \xi_1^2\xi_2^2 & \xi_2^4 & \xi_1\xi_2^3 \\ \xi_1^2\xi_2 & \xi_1\xi_2^2 & \xi_1^3\xi_2 & \xi_1\xi_2^3 & \xi_1^2\xi_2^2 \end{bmatrix} A_{(j)}$
j = next family member
$k = k + 1$
False
$j \equiv \mathrm{member}^{(N)}$?
$\mathbf{a} = \mathbf{A} \backslash \mathbf{b}$
Store components of $\mathbf{a}$ into matrix $\hat{\mathbf{a}}$ for current material point:
$\hat{\mathbf{a}}(\mathrm{row} = k, \mathrm{columns} = 1 \div 5) = \mathbf{a}(\mathrm{row} = 1, \mathrm{columns} = 1 \div 5)$
$\hat{\mathbf{a}}(\mathrm{row} = k, \mathrm{columns} = 6 \div 10) = \mathbf{a}(\mathrm{row} = 2, \mathrm{columns} = 1 \div 5)$
$\hat{\mathbf{a}}(\mathrm{row} = k, \mathrm{columns} = 11 \div 15) = \mathbf{a}(\mathrm{row} = 3, \mathrm{columns} = 1 \div 5)$
$\hat{\mathbf{a}}(\mathrm{row} = k, \mathrm{columns} = 16 \div 20) = \mathbf{a}(\mathrm{row} = 4, \mathrm{columns} = 1 \div 5)$
$\hat{\mathbf{a}}(\mathrm{row} = k, \mathrm{columns} = 21 \div 25) = \mathbf{a}(\mathrm{row} = 5, \mathrm{columns} = 1 \div 5)$
False
$k < N_p$?
True
Output $\hat{\mathbf{a}}$

Fig. 3. Calculate matrix $\hat{\mathbf{a}}$ to store 25 components of matrix $\mathbf{a}$ for each node.

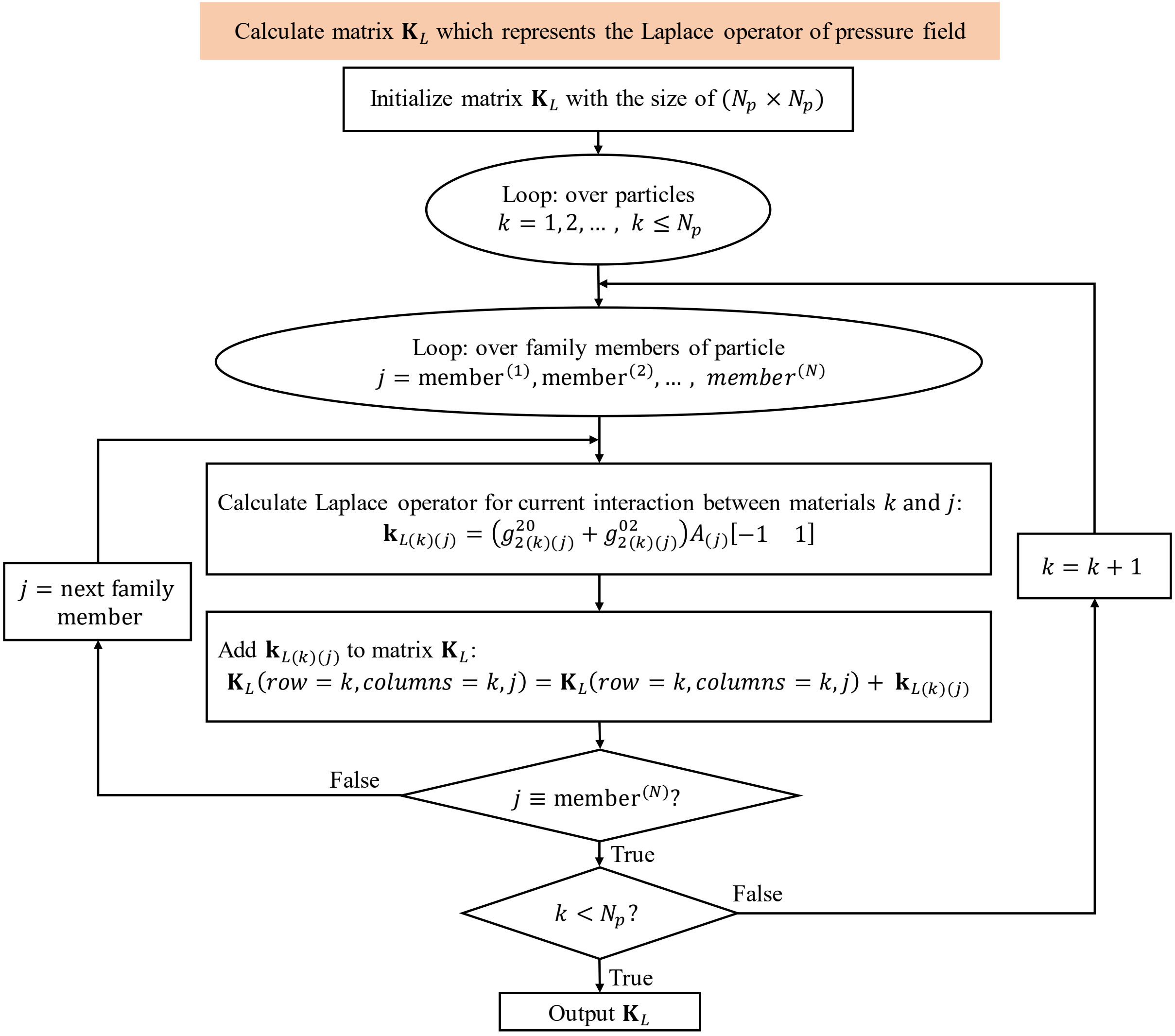

Calculate matrix $\mathbf{K}_L$ which represents the Laplace operator of pressure field
Initialize matrix $\mathbf{K}_L$ with the size of $(N_p \times N_p)$
Loop: over particles $k = 1, 2, \dots, \ k \le N_p$
Loop: over family members of particle $j = \text{member}^{(1)}, \text{member}^{(2)}, \dots, \ member^{(N)}$
Calculate Laplace operator for current interaction between materials k and j: $\mathbf{k}_{L(k)(j)} = \left(g_{2(k)(j)}^{20} + g_{2(k)(j)}^{02}\right) A_{(j)} [-1 \quad 1]$
Add $\mathbf{k}_{L(k)(j)}$ to matrix $\mathbf{K}_L$: $\mathbf{K}_L(row = k, columns = k, j) = \mathbf{K}_L(row = k, columns = k, j) + \mathbf{k}_{L(k)(j)}$
$j \equiv \text{member}^{(N)}$?
False
$j = $ next family member
True
$k < N_p$?
False
$k = k + 1$
True
Output $\mathbf{K}_L$

Fig. 4. Calculate matrix $\mathbf{K}_L$ to represent Laplace operator of the pressure field.

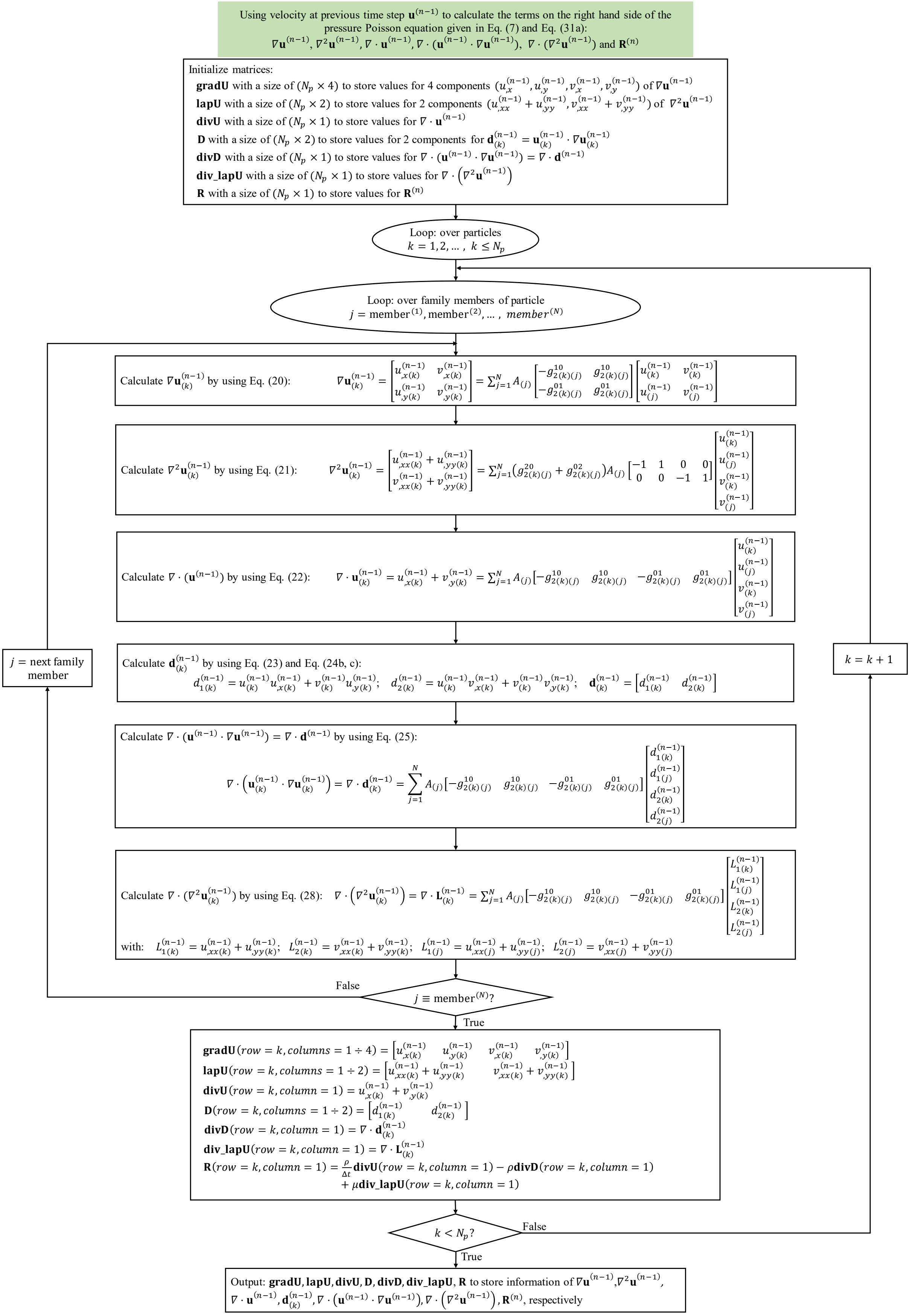

Using velocity at previous time step $\mathbf{u}^{(n-1)}$ to calculate the terms on the right hand side of the pressure Poisson equation given in Eq. (7) and Eq. (31a):
$\nabla\mathbf{u}^{(n-1)}, \nabla^2\mathbf{u}^{(n-1)}, \nabla\cdot\mathbf{u}^{(n-1)}, \nabla\cdot(\mathbf{u}^{(n-1)}\cdot\nabla\mathbf{u}^{(n-1)}),\ \nabla\cdot(\nabla^2\mathbf{u}^{(n-1)})$ and $\mathbf{R}^{(n)}$
Initialize matrices:
gradU with a size of $(N_p \times 4)$ to store values for 4 components $(u_{,x}^{(n-1)}, u_{,y}^{(n-1)}, v_{,x}^{(n-1)}, v_{,y}^{(n-1)})$ of $\nabla\mathbf{u}^{(n-1)}$
lapU with a size of $(N_p \times 2)$ to store values for 2 components $(u_{,xx}^{(n-1)} + u_{,yy}^{(n-1)}, v_{,xx}^{(n-1)} + v_{,yy}^{(n-1)})$ of $\nabla^2\mathbf{u}^{(n-1)}$
divU with a size of $(N_p \times 1)$ to store values for $\nabla\cdot\mathbf{u}^{(n-1)}$
D with a size of $(N_p \times 2)$ to store values for 2 components for $\mathbf{d}_{(k)}^{(n-1)} = \mathbf{u}_{(k)}^{(n-1)}\cdot\nabla\mathbf{u}_{(k)}^{(n-1)}$
divD with a size of $(N_p \times 1)$ to store values for $\nabla\cdot(\mathbf{u}^{(n-1)}\cdot\nabla\mathbf{u}^{(n-1)}) = \nabla\cdot\mathbf{d}^{(n-1)}$
div_lapU with a size of $(N_p \times 1)$ to store values for $\nabla\cdot\left(\nabla^2\mathbf{u}^{(n-1)}\right)$
R with a size of $(N_p \times 1)$ to store values for $\mathbf{R}^{(n)}$
Loop: over particles
$k = 1, 2, \dots,\ k \le N_p$
Loop: over family members of particle
$j = \text{member}^{(1)}, \text{member}^{(2)}, \dots,\ member^{(N)}$
Calculate $\nabla\mathbf{u}_{(k)}^{(n-1)}$ by using Eq. (20): $\nabla\mathbf{u}_{(k)}^{(n-1)} = \begin{bmatrix} u_{,x(k)}^{(n-1)} & v_{,x(k)}^{(n-1)} \\ u_{,y(k)}^{(n-1)} & v_{,y(k)}^{(n-1)} \end{bmatrix} = \sum_{j=1}^{N} A_{(j)} \begin{bmatrix} -g_{2(k)(j)}^{10} & g_{2(k)(j)}^{10} \\ -g_{2(k)(j)}^{01} & g_{2(k)(j)}^{01} \end{bmatrix} \begin{bmatrix} u_{(k)}^{(n-1)} & v_{(k)}^{(n-1)} \\ u_{(j)}^{(n-1)} & v_{(j)}^{(n-1)} \end{bmatrix}$
Calculate $\nabla^2\mathbf{u}_{(k)}^{(n-1)}$ by using Eq. (21): $\nabla^2\mathbf{u}_{(k)}^{(n-1)} = \begin{bmatrix} u_{,xx(k)}^{(n-1)} + u_{,yy(k)}^{(n-1)} \\ v_{,xx(k)}^{(n-1)} + v_{,yy(k)}^{(n-1)} \end{bmatrix} = \sum_{j=1}^{N} \left(g_{2(k)(j)}^{20} + g_{2(k)(j)}^{02}\right) A_{(j)} \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} u_{(k)}^{(n-1)} \\ u_{(j)}^{(n-1)} \\ v_{(k)}^{(n-1)} \\ v_{(j)}^{(n-1)} \end{bmatrix}$
Calculate $\nabla\cdot(\mathbf{u}^{(n-1)})$ by using Eq. (22): $\nabla\cdot\mathbf{u}_{(k)}^{(n-1)} = u_{,x(k)}^{(n-1)} + v_{,y(k)}^{(n-1)} = \sum_{j=1}^{N} A_{(j)} \begin{bmatrix} -g_{2(k)(j)}^{10} & g_{2(k)(j)}^{10} & -g_{2(k)(j)}^{01} & g_{2(k)(j)}^{01} \end{bmatrix} \begin{bmatrix} u_{(k)}^{(n-1)} \\ u_{(j)}^{(n-1)} \\ v_{(k)}^{(n-1)} \\ v_{(j)}^{(n-1)} \end{bmatrix}$
Calculate $\mathbf{d}_{(k)}^{(n-1)}$ by using Eq. (23) and Eq. (24b, c):
$d_{1(k)}^{(n-1)} = u_{(k)}^{(n-1)} u_{,x(k)}^{(n-1)} + v_{(k)}^{(n-1)} u_{,y(k)}^{(n-1)};\quad d_{2(k)}^{(n-1)} = u_{(k)}^{(n-1)} v_{,x(k)}^{(n-1)} + v_{(k)}^{(n-1)} v_{,y(k)}^{(n-1)};\quad \mathbf{d}_{(k)}^{(n-1)} = \begin{bmatrix} d_{1(k)}^{(n-1)} & d_{2(k)}^{(n-1)} \end{bmatrix}$
Calculate $\nabla\cdot(\mathbf{u}^{(n-1)}\cdot\nabla\mathbf{u}^{(n-1)}) = \nabla\cdot\mathbf{d}^{(n-1)}$ by using Eq. (25):
$\nabla\cdot\left(\mathbf{u}_{(k)}^{(n-1)}\cdot\nabla\mathbf{u}_{(k)}^{(n-1)}\right) = \nabla\cdot\mathbf{d}_{(k)}^{(n-1)} = \sum_{j=1}^{N} A_{(j)} \begin{bmatrix} -g_{2(k)(j)}^{10} & g_{2(k)(j)}^{10} & -g_{2(k)(j)}^{01} & g_{2(k)(j)}^{01} \end{bmatrix} \begin{bmatrix} d_{1(k)}^{(n-1)} \\ d_{1(j)}^{(n-1)} \\ d_{2(k)}^{(n-1)} \\ d_{2(j)}^{(n-1)} \end{bmatrix}$
Calculate $\nabla\cdot(\nabla^2\mathbf{u}_{(k)}^{(n-1)})$ by using Eq. (28): $\nabla\cdot\left(\nabla^2\mathbf{u}_{(k)}^{(n-1)}\right) = \nabla\cdot\mathbf{L}_{(k)}^{(n-1)} = \sum_{j=1}^{N} A_{(j)} \begin{bmatrix} -g_{2(k)(j)}^{10} & g_{2(k)(j)}^{10} & -g_{2(k)(j)}^{01} & g_{2(k)(j)}^{01} \end{bmatrix} \begin{bmatrix} L_{1(k)}^{(n-1)} \\ L_{1(j)}^{(n-1)} \\ L_{2(k)}^{(n-1)} \\ L_{2(j)}^{(n-1)} \end{bmatrix}$
with: $L_{1(k)}^{(n-1)} = u_{,xx(k)}^{(n-1)} + u_{,yy(k)}^{(n-1)};\ L_{2(k)}^{(n-1)} = v_{,xx(k)}^{(n-1)} + v_{,yy(k)}^{(n-1)};\ L_{1(j)}^{(n-1)} = u_{,xx(j)}^{(n-1)} + u_{,yy(j)}^{(n-1)};\ L_{2(j)}^{(n-1)} = v_{,xx(j)}^{(n-1)} + v_{,yy(j)}^{(n-1)}$
False
$j \equiv \text{member}^{(N)}$?
True
j = next family member
gradU$(row = k, columns = 1 \div 4) = \begin{bmatrix} u_{,x(k)}^{(n-1)} & u_{,y(k)}^{(n-1)} & v_{,x(k)}^{(n-1)} & v_{,y(k)}^{(n-1)} \end{bmatrix}$
lapU$(row = k, columns = 1 \div 2) = \begin{bmatrix} u_{,xx(k)}^{(n-1)} + u_{,yy(k)}^{(n-1)} & v_{,xx(k)}^{(n-1)} + v_{,yy(k)}^{(n-1)} \end{bmatrix}$
divU$(row = k, column = 1) = u_{,x(k)}^{(n-1)} + v_{,y(k)}^{(n-1)}$
D$(row = k, columns = 1 \div 2) = \begin{bmatrix} d_{1(k)}^{(n-1)} & d_{2(k)}^{(n-1)} \end{bmatrix}$
divD$(row = k, column = 1) = \nabla\cdot\mathbf{d}_{(k)}^{(n-1)}$
div_lapU$(row = k, column = 1) = \nabla\cdot\mathbf{L}_{(k)}^{(n-1)}$
R$(row = k, column = 1) = \frac{\rho}{\Delta t}$**divU**$(row = k, column = 1) - \rho$**divD**$(row = k, column = 1) + \mu$**div_lapU**$(row = k, column = 1)$
$k < N_p$?
False
True
$k = k + 1$
Output: **gradU**, **lapU**, **divU**, **D**, **divD**, **div_lapU**, **R** to store information of $\nabla\mathbf{u}^{(n-1)}, \nabla^2\mathbf{u}^{(n-1)}, \nabla\cdot\mathbf{u}^{(n-1)}, \mathbf{d}_{(k)}^{(n-1)}, \nabla\cdot\left(\mathbf{u}^{(n-1)}\cdot\nabla\mathbf{u}^{(n-1)}\right), \nabla\cdot\left(\nabla^2\mathbf{u}^{(n-1)}\right), \mathbf{R}^{(n)}$, respectively

Fig. 5. Calculate the terms on the right-hand side of the pressure Poisson equation.

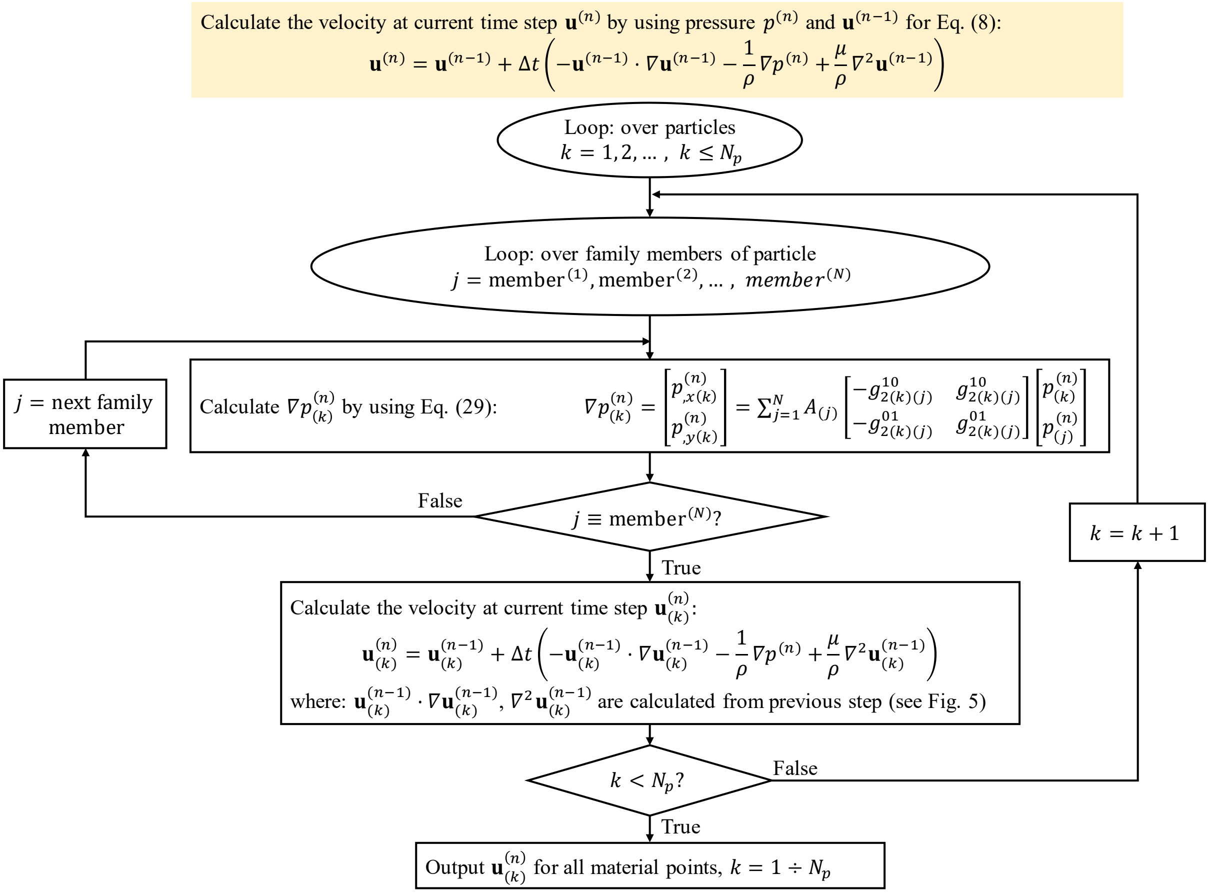

Calculate the velocity at current time step $\mathbf{u}^{(n)}$ by using pressure $p^{(n)}$ and $\mathbf{u}^{(n-1)}$ for Eq. (8):
$\mathbf{u}^{(n)} = \mathbf{u}^{(n-1)} + \Delta t\left(-\mathbf{u}^{(n-1)} \cdot \nabla\mathbf{u}^{(n-1)} - \frac{1}{\rho}\nabla p^{(n)} + \frac{\mu}{\rho}\nabla^2\mathbf{u}^{(n-1)}\right)$
Loop: over particles
$k = 1,2,\dots,\ k \le N_p$
Loop: over family members of particle
$j = \text{member}^{(1)}, \text{member}^{(2)}, \dots,\ member^{(N)}$
j = next family member
Calculate $\nabla p_{(k)}^{(n)}$ by using Eq. (29):
$\nabla p_{(k)}^{(n)} = \begin{bmatrix} p_{,x(k)}^{(n)} \\ p_{,y(k)}^{(n)} \end{bmatrix} = \sum_{j=1}^{N} A_{(j)} \begin{bmatrix} -g_{2(k)(j)}^{10} & g_{2(k)(j)}^{10} \\ -g_{2(k)(j)}^{01} & g_{2(k)(j)}^{01} \end{bmatrix} \begin{bmatrix} p_{(k)}^{(n)} \\ p_{(j)}^{(n)} \end{bmatrix}$
False
$j \equiv \text{member}^{(N)}$?
True
$k = k + 1$
Calculate the velocity at current time step $\mathbf{u}_{(k)}^{(n)}$:
$\mathbf{u}_{(k)}^{(n)} = \mathbf{u}_{(k)}^{(n-1)} + \Delta t\left(-\mathbf{u}_{(k)}^{(n-1)} \cdot \nabla\mathbf{u}_{(k)}^{(n-1)} - \frac{1}{\rho}\nabla p^{(n)} + \frac{\mu}{\rho}\nabla^2\mathbf{u}_{(k)}^{(n-1)}\right)$
where: $\mathbf{u}_{(k)}^{(n-1)} \cdot \nabla\mathbf{u}_{(k)}^{(n-1)}$, $\nabla^2\mathbf{u}_{(k)}^{(n-1)}$ are calculated from previous step (see Fig. 5)
$k < N_p$?
False
True
Output $\mathbf{u}_{(k)}^{(n)}$ for all material points, $k = 1 \div N_p$

Fig. 6. Calculate the velocity at current time step $\mathbf{u}^{(n)}$

4. Boundary conditions

In this study, for solid boundaries, no-slip boundary conditions are used. Details of the no-slip boundary conditions are described in Section 4.1. The inlet and outlet boundary conditions are described in Section 4.2 and Section 4.3, respectively.

4.1. No-slip boundary conditions

To apply no-slip boundary conditions for solid walls, three layers of fictitious particles are used as shown in Fig. 7. In Fig. 7, the fictitious solid particles are shown in blue. Particles located on the boundary are shown in red. Meanwhile, the fluid particles inside the fluid domain are shown in pink.

To apply no-slip boundary conditions for a stationary wall, the velocities of particles on the boundary and the velocities of solid particles are set equal to zero as

$$\mathbf{u}^{(n)}(\mathbf{x}, t)\big|_{\Gamma} = 0 \tag{36}$$

where Γ represents the solid boundary as shown in Fig. 7.

For the pressure Poisson equation given in Eq. (7), the homogeneous Neumann boundary condition is applied for the solid walls as (Hosseini and Feng, 2011)

$$\mathbf{n} \cdot \nabla p^{(n)}\big|_{\Gamma} = 0 \tag{37}$$

where **n** represents the normal vector of the solid boundary Γ as shown in Fig. 7.

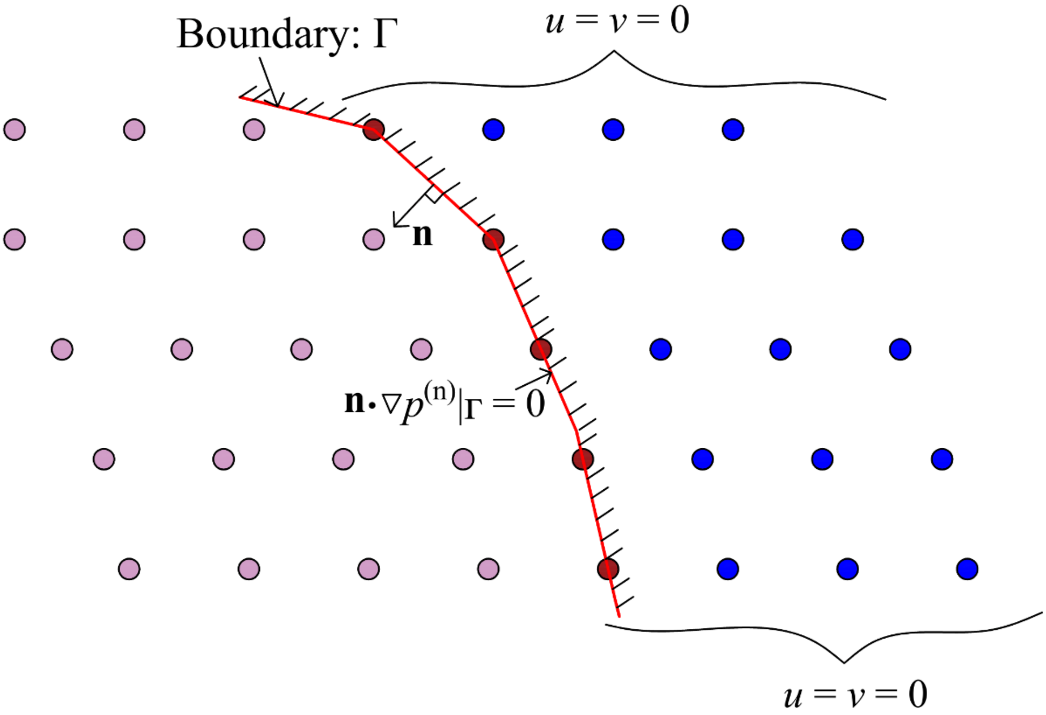


Fig. 7. No-slip boundary condition for a solid boundary.

Details of implementing no-slip boundary conditions for solid boundaries as well as the obstacle cylinder used in this study are described in Sections 4.1.1 and 4.1.2, respectively.

4.1.1. Implementation of no-slip boundary conditions for vertical and horizontal boundaries in the discretized model

In this study, the considered solid boundaries used in Section 5 are either vertical or horizontal walls as shown in Fig. 8. In Fig. 8, blue points represent fictitious particles in solid regions, red points represent the fluid particle along the solid walls, and pink points represent fluid particles inside the fluid domain. For the vertical walls, the normal vector can be either $\mathbf{n} = [1 \quad 0]^T$ or $\mathbf{n} = [-1 \quad 0]^T$ as shown in Fig. 8(a). Therefore, the no-slip velocity boundary conditions given in Eq. (36) can be written for these vertical walls as

$$\mathbf{u}^{(n)}\big|_L = \mathbf{u}^{(n)}\big|_{\text{fictitious particles on the left}} = 0 \tag{38a}$$

and

$$\mathbf{u}^{(n)}\big|_R = \mathbf{u}^{(n)}\big|_{\text{fictitious particles on the right}} = 0 \tag{38b}$$

where L and R represent solid boundaries on the left and right edges of the fluid domain as shown in Fig. 8(a).

For the horizontal walls, the normal vector can be either $\mathbf{n} = [0 \quad 1]^T$ or $\mathbf{n} = [0 \quad -1]^T$ as shown in Fig. 8(b). Therefore, the no-slip velocity boundary conditions given in Eq. (36) can be written for these horizontal walls as

$$\mathbf{u}^{(n)}\big|_T = \mathbf{u}^{(n)}\big|_{\text{fictitious particles on the top}} = 0 \tag{39a}$$

and

$$\mathbf{u}^{(n)}\big|_B = \mathbf{u}^{(n)}\big|_{\text{fictitious particles on the bottom}} = 0 \tag{39b}$$

where T and B represent solid boundaries on the top and bottom edges of the fluid domain as shown in Fig. 8(b).

In this study, the pressure boundary conditions in Eq. (37) for the vertical walls are applied by setting pressures of the solid particles equal to the pressures of their image particles inside the fluid domain with the same y coordinates. Specifically, the pressure boundary conditions in Eq. (37) for the left vertical boundary can be implemented as

$$p(x_L - \Delta x, y, t) = p(x_L + \Delta x, y, t) \tag{40a}$$

$$p(x_L - 2\Delta x, y, t) = p(x_L + 2\Delta x, y, t) \tag{40b}$$

$$p(x_L - 3\Delta x, y, t) = p(x_L + 3\Delta x, y, t) \tag{40c}$$

where x_L is the location in x direction of the left vertical boundary and Δx represents the mesh size in x direction.

Similarly, the pressure boundary conditions in Eq. (37) for the right vertical boundary can be implemented as

$$p(x_R + \Delta x, y, t) = p(x_R - \Delta x, y, t) \tag{41a}$$

$$p(x_R + 2\Delta x, y, t) = p(x_R - 2\Delta x, y, t) \tag{41b}$$

$$p(x_R + 3\Delta x, y, t) = p(x_R - 3\Delta x, y, t) \tag{41c}$$

where x_R is the location in x direction of the right vertical boundary.

On the other hand, the pressure boundary conditions in Eq. (37) for the horizontal walls are applied by setting pressures of the solid particles equal to the pressures of their image particles inside the fluid domain with the same x coordinates. Specifically, the pressure boundary conditions in Eq. (37) for the top horizontal boundary can be implemented as

$$p(x, y_T + \Delta y, t) = p(x, y_T - \Delta y, t) \tag{42a}$$

$$p(x, y_T + 2\Delta y, t) = p(x, y_T - 2\Delta y, t) \tag{42b}$$

$$p(x, y_T + 3\Delta y, t) = p(x, y_T - 3\Delta y, t) \tag{42c}$$

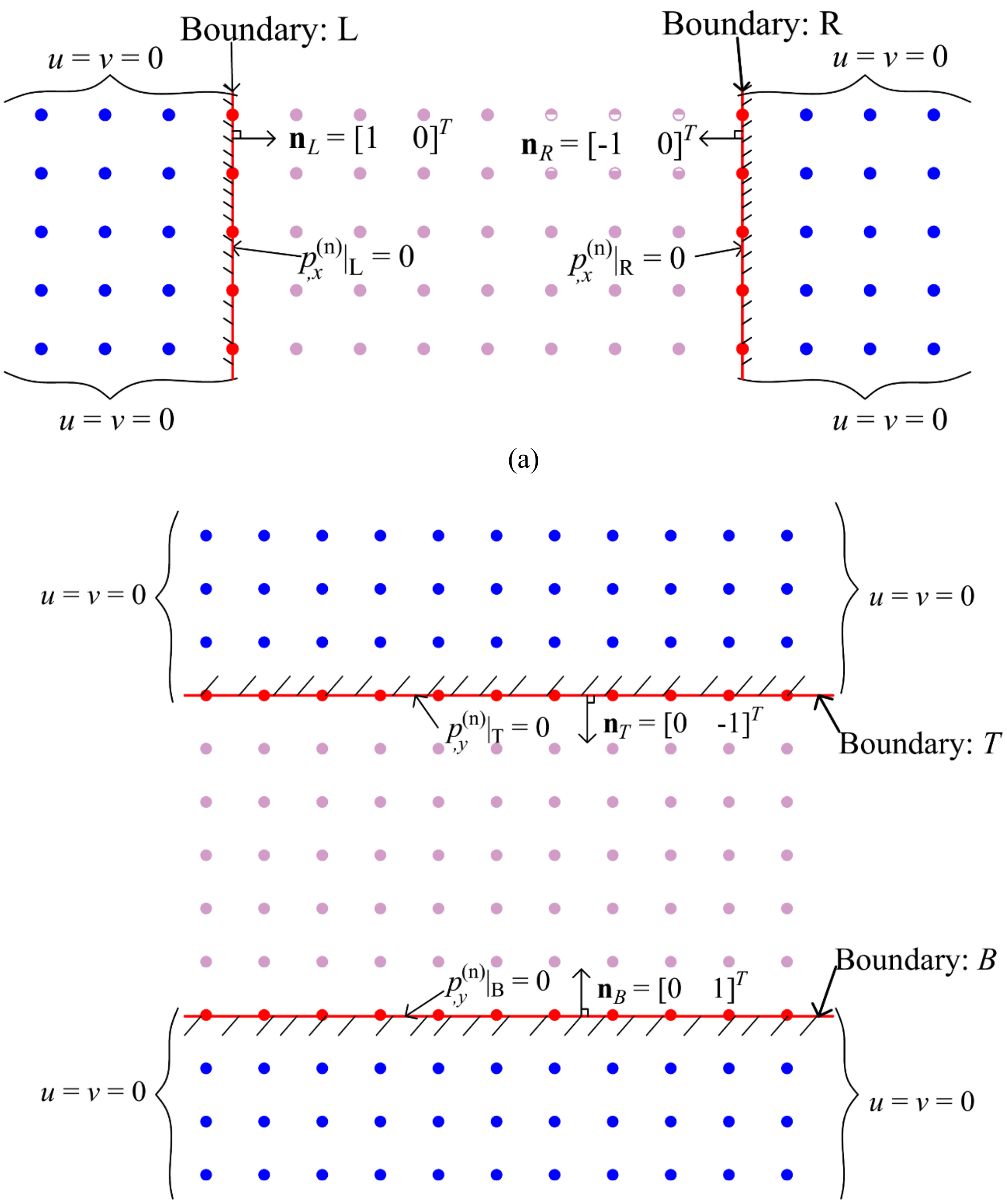
where y_T is the location in y direction of the top horizontal boundary and Δy represents the mesh size in y direction. Similarly, the pressure boundary conditions in Eq. (37) for the bottom horizontal boundary can be implemented as

$$p(x, y_B - \Delta y, t) = p(x, y_B + \Delta y, t) \tag{43a}$$

$$p(x, y_B - 2\Delta y, t) = p(x, y_B + 2\Delta y, t) \tag{43b}$$

$$p(x, y_B - 3\Delta y, t) = p(x, y_B + 3\Delta y, t) \tag{43c}$$

where y_B is the location in y direction of the bottom horizontal boundary.



(a)

(b)

Fig. 8. No-slip boundary conditions for (a): vertical walls, (b): horizontal walls.

4.1.2. Implementation of no-slip boundary conditions for circular stationary walls in the discretized model

To investigate the problem of flow over the obstacle circular cylinder as presented in Section 5.3, the nodes surrounding the cylinder are arranged in the cylindrical coordinates as shown in Fig. 9. In this figure, blue points represent fictitious particles in solid regions, red points represent the fluid particle along the solid walls, and pink points represent fluid particles inside the fluid domain. To apply no-slip boundary conditions and pressure boundary conditions for the cylinder wall, Eqs. (36-37) can be rewritten with respect to the cylindrical coordinate system attached with the cylinder as

$$\mathbf{u}^{(n)}\Big|_{\text{cylinder wall}} = 0 \tag{44}$$

$$\left(n_r \frac{\partial p^{(n)}}{\partial r} + n_\tau \frac{\partial p^{(n)}}{\partial \tau}\right)\Bigg|_{\text{cylinder wall}} = 0 \tag{45}$$

where r represents the radius coordinate. Meanwhile, τ represents the tangential coordinate.

Since the tangential component of the normal vector is zero, $n_\tau = 0$, Eq. (45) can be simplified as

$$\frac{\partial p^{(n)}}{\partial r}\Bigg|_{\text{cylinder wall}} = 0 \tag{46}$$

In the discretized model, the zero-velocity condition given in Eq. (44) can be implemented as

$$\mathbf{u}_{(s_1)}^{(n)} = \mathbf{u}_{(s_2)}^{(n)} = \mathbf{u}_{(s_3)}^{(n)} = \mathbf{u}_{(f_0)}^{(n)} = 0 \tag{47}$$

where s_1, s_2, s_3 are fictitious particles in the solid domain and f_0 is a node located on the cylinder wall as shown in Fig. 9.

Meanwhile, the pressure boundary condition given in Eq. (46) can be applied by setting pressures of the solid particles equal to the pressures of their image particles inside the fluid domain as

$$p_{(\mathrm{S}_1)}^{(n)} = p_{(\mathrm{f}_1)}^{(n)} \tag{48a}$$

$$p_{(\mathrm{S}_2)}^{(n)} = p_{(\mathrm{f}_2)}^{(n)} \tag{48b}$$

$$p_{(\mathrm{S}_3)}^{(n)} = p_{(\mathrm{f}_3)}^{(n)} \tag{48c}$$

where f_1, f_2, f_3 are the fluid particles that are respectively images of s_1, s_2, s_3, as shown in Fig. 9.

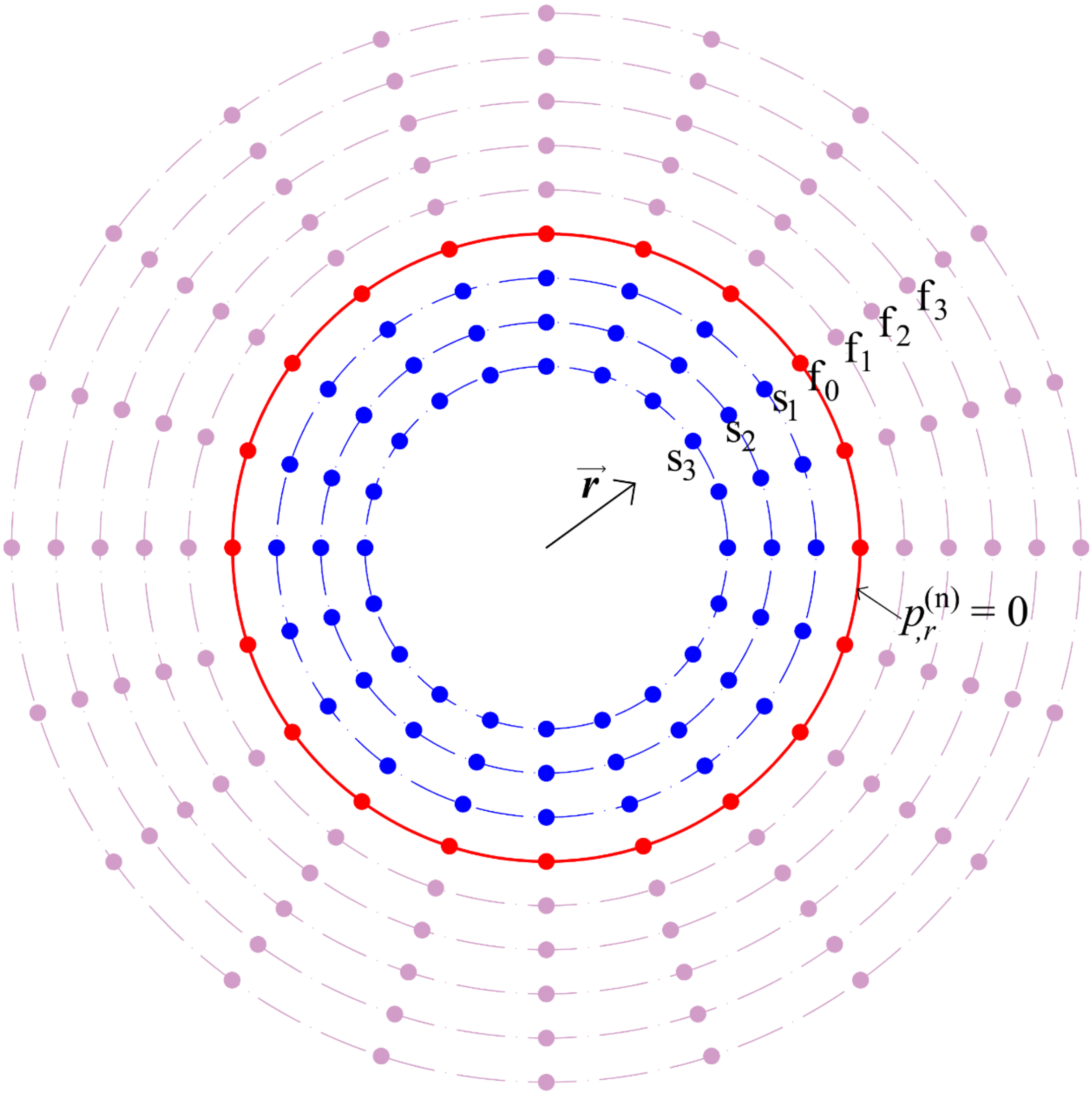


Fig. 9. No-slip boundary conditions for obstacle circular cylinder.

4.2. Boundary conditions at the inlet

In Sections 5.2 and 5.3, the constant velocity is applied for the inlet boundary at the location of $x = 0$ as

$$u(x = 0, y, t) = u_0 \tag{49a}$$

$$v(x = 0, y, t) = v_0 \tag{49b}$$

In the discretized model, three fictitious layers of nodes are added for the inlet. The velocities in x direction of these fictitious particles and the particles located at $x = 0$ are set equal to u_0. Meanwhile, the velocities in y direction of these fictitious particles and the particles located at $x = 0$ are set equal to v_0. In addition, the pressures of these fictitious particles are set equal to the pressure of the particles at $x = 0$ with the same y coordinates (Hosseini and Feng, 2011; Lee et al., 2008; Shao and Lo, 2003).

4.3. Boundary condition at the outlet

In Sections 5.2 and 5.3, the zero-pressure condition is applied for the outlet boundary at the location of $x = L_x$ as

$$p(x = L_x, y, t) = 0 \tag{50}$$

In the discretized model, the pressure of the exit layer, shown in black in Fig. 10, is set equal to zero. Meanwhile, the pressures of the fictitious particles, shown in blue in Fig. 10, are defined as

$$p_{(i)}^{(n)} = 0 \tag{51a}$$

$$p_{(j)}^{(n)} = -p_{(r)}^{(n)} \tag{51b}$$

$$p_{(k)}^{(n)} = -p_{(q)}^{(n)} \tag{51c}$$

$$p_{(l)}^{(n)} = -p_{(p)}^{(n)} \tag{51d}$$

where i in Eq. (51a) is a particle located on the exit layer as shown in Fig. 10. In Eqs. (51b-d), j, k, l are three fictitious particles that have the same y coordinate with particle i. Meanwhile, r, q, p are three fluid particles that are images of fictitious particles j, k, l, respectively, as shown in Fig. 10.

Moreover, the velocities of the fictitious (dummy) particles downstream are set equal to the velocity of the particle with the same y coordinate located on the exit layer as shown in Fig. 10 as

$$\mathbf{u}_{(j)}^{(n)} = \mathbf{u}_{(k)}^{(n)} = \mathbf{u}_{(l)}^{(n)} = \mathbf{u}_{(i)}^{(n)} \tag{52}$$

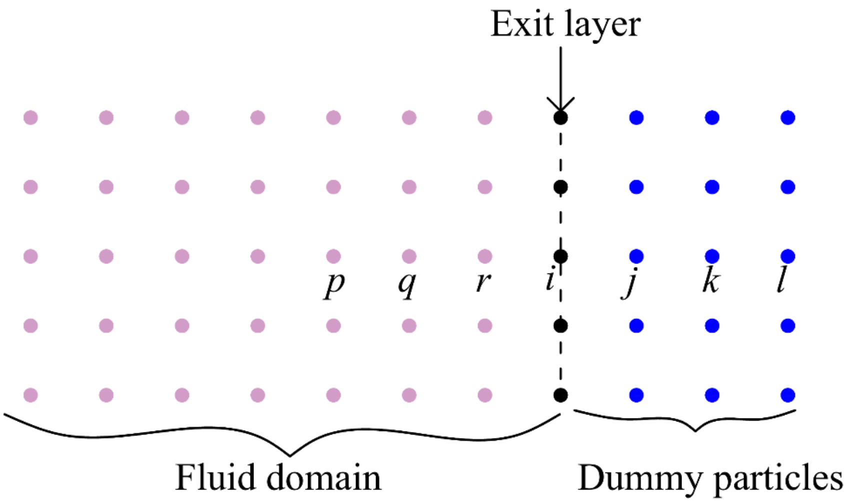


Fig. 10. The zero-pressure boundary condition for the outlet

5. Numerical results

In this section, first, a 2D cavitation problem for Reynolds numbers, $Re = 1$ and $Re = 200$, is investigated in Section 5.1. Next, in Section 5.2, a 2D fluid flow inside an open channel is investigated for $Re = 56.02$. Finally, a problem of a fluid flow over an obstacle cylinder for Reynolds numbers, $Re = 10$ and $Re = 100$, is investigated in Section 5.3. In these examples, the horizon size of $\delta = 3\Delta x$ is used. To verify the accuracy of the Eulerian incompressible

PDDO model, the results predicted by the Eulerian incompressible PDDO model are compared with those obtained by ANSYS Fluent.

5.1. 2D cavitation

In this section, a 2D square fluid domain with a size of $L = 0.001$ m is investigated as shown in Fig. 11(a). The fluid has a density of $\rho = 1000\ \mathrm{kg/m^3}$ and viscosity of $\mu = 0.001$ Pa. s. On the top boundary at $y = L$, the fluid is subjected to velocity and pressure boundary conditions as

$$u(x, y = L, t) = u_0 \tag{53a}$$

$$v(x, y = L, t) = 0 \tag{53b}$$

$$\frac{\partial p}{\partial y}(x, y = L, t) = 0 \tag{53c}$$

The fluid is subjected to no-slip boundary conditions on the three boundaries as shown in red in Fig. 11(a) as

$$u(x = 0, y, t) = v(x = 0, y, t) = 0 \tag{54a}$$

$$u(x = L, y, t) = v(x = L, y, t) = 0 \tag{54b}$$

$$u(x, y = 0, t) = v(x, y = 0, t) = 0 \tag{54c}$$

The fluid domain is uniformly discretized with a mesh size of $\Delta x = 10^{-5}$ m. To implement boundary conditions, three layers of fictitious particles are used as shown in red and black in Fig. 11(b). The implementation of no-slip boundary conditions for the left and right walls, as shown in red in Fig. 11(b), is presented in Section 4.1.1 for a vertical wall. The implementation of no-slip boundary conditions for the bottom wall, as shown in red in Fig. 11(b), is presented in Section 4.1.1 for the horizontal wall. Meanwhile, the velocity conditions on the top boundary, shown in black in Fig. 11(b), are represented as

$$u(x, y \geq L, t) = u_0 \tag{55a}$$

$$v(x, y \geq L, t) = 0 \tag{55b}$$

Moreover, as shown in Fig. 11(c), the pressures of the fictitious particles on the top boundary are set equal to the pressures of their image particles inside the fluid domain as

$$p_{(j)} = p_{(r)} \tag{56a}$$

$$p_{(k)} = p_{(q)} \tag{56b}$$

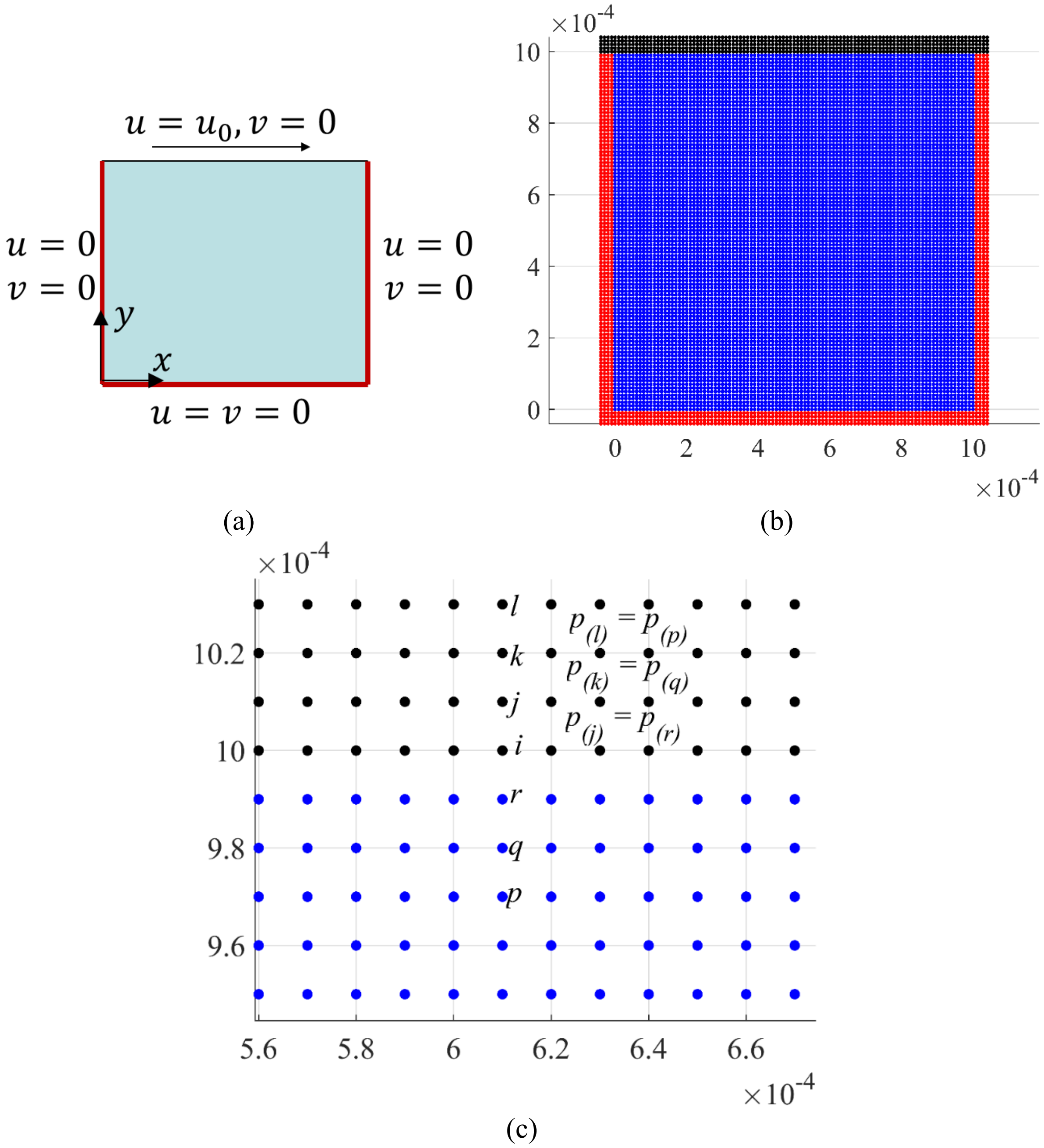
$$p_{(l)} = p_{(p)} \tag{56c}$$

where j, k, l are three particles located on three fictitious boundary layers on top as shown in black in Fig. 11(c). Meanwhile, r, q, p are three image particles inside the fluid domain of particles j, k, l, respectively, as shown in Fig. 11(c).

The problem is investigated in two cases of different Reynolds numbers as

- Case 1: $u_0 = 0.001$ (m/s), $Re = \rho u_0 L/\mu$ =1
- Case 2: $u_0 = 0.2$ (m/s), $Re = \rho u_0 L/\mu$ =200

For verification purposes, the results predicted by the Eulerian incompressible PDDO model are compared with the results obtained by using ANSYS Fluent. In ANSYS Fluent, the mesh size of $\Delta x = 10^{-5}$ is used. The problem is solved using a steady solution with the SIMPLE (Semi-Implicit Method for Pressure Linked Equations) scheme. In the Eulerian incompressible PDDO model, the transient analysis using the numerical procedure presented in Section 3.3 is used. The time step size of $\Delta t = 10^{-4}$ is used for Case 1 and $\Delta t = 10^{-5}$ is used for Case 2.



(a)

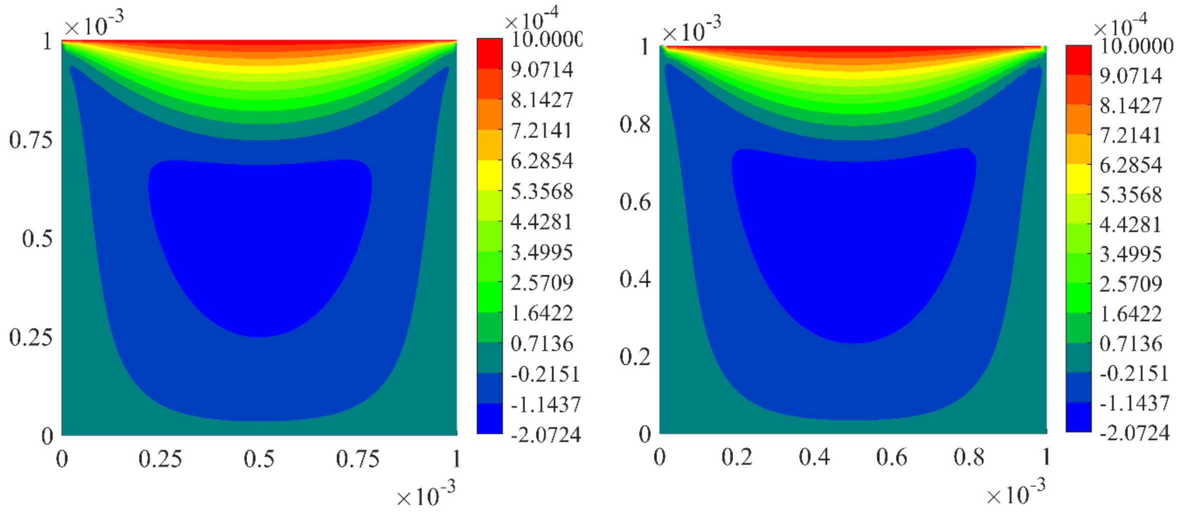
(b)

(c)

Fig. 11. Two-dimensional cavitation (a): geometry, (b): the discretized model, (c): pressure boundary conditions for the fictitious layers on the top boundary

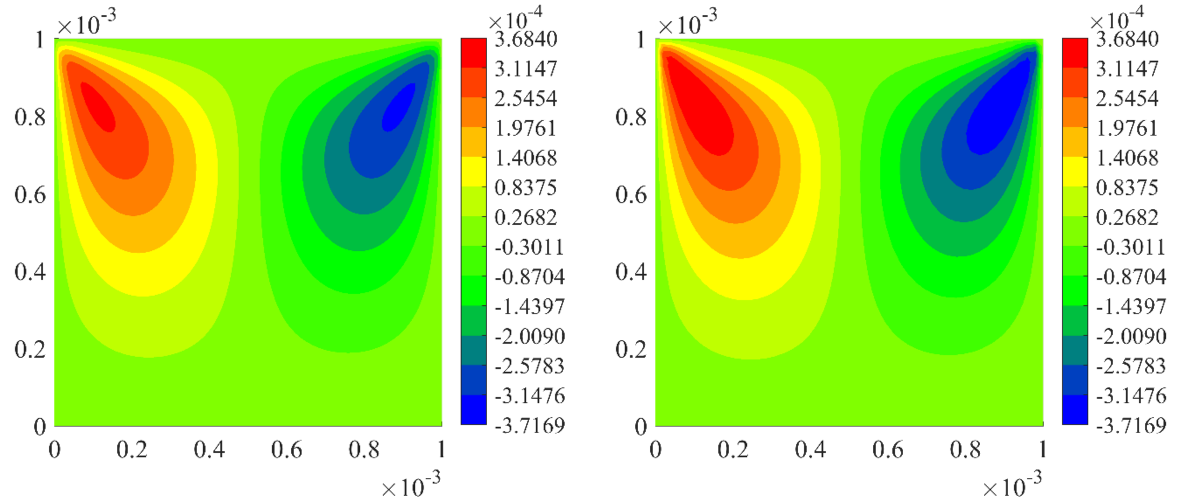
Case 1: $u_0 = 0.001\ (m/s),\ Re = 1$

Fig. 12-Fig. 15 show the converged solution results of velocity and pressure fields for case 1 with Re = 1. It can be observed from Fig. 12 and Fig. 13 that the distributions of velocities u and v captured by the Eulerian incompressible PDDO model agree very well with those captured by ANSYS Fluent. Fig. 14(a) shows the variation of the relative velocity u/u_0 versus the relative coordinate y/L along the central line $x = L/2$. Fig. 14(b) shows the variation of the relative velocity v/u_0 versus the relative coordinate x/L along the central line $y = L/2$. As can be observed from these figures, the results obtained by the Eulerian incompressible PDDO model and ANSYS Fluent agree very well. Moreover, as shown in Fig. 15, the pressure fields predicted by the Eulerian incompressible PDDO model and ANSYS Fluent have also a good agreement.



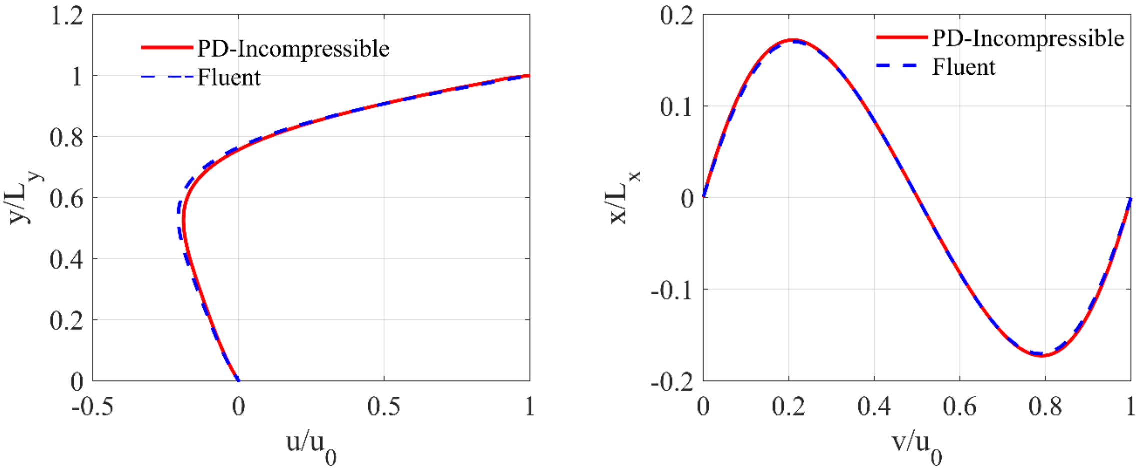
(a) (b)

Fig. 12. Variations of velocity u (m/s) captured by (a): Eulerian incompressible PDDO model, (b): ANSYS Fluent for Case 1 with $Re = 1$



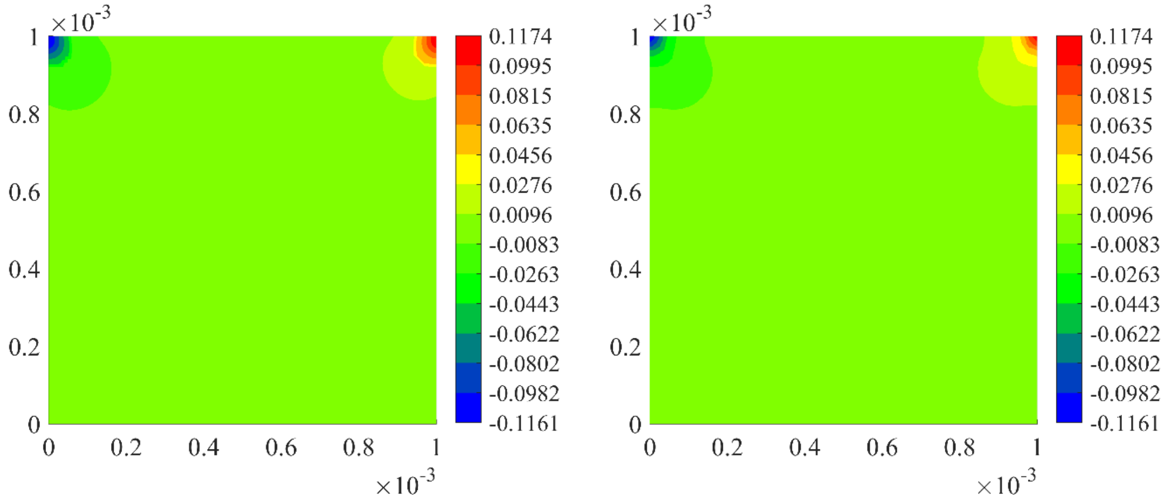
(a) (b)

Fig. 13. Variations of velocity v (m/s) captured by (a): Eulerian incompressible PDDO model, (b): ANSYS Fluent for Case 1 with $Re = 1$



(a) (b)

Fig. 14. Variations of relative velocities along central lines captured by the Eulerian incompressible PDDO model and ANSYS Fluent (a): u/u_0 versus y/L at $x = L/2$, (b): v/u_0 versus x/L at $y = L/2$ for Case 1 with $Re = 1$

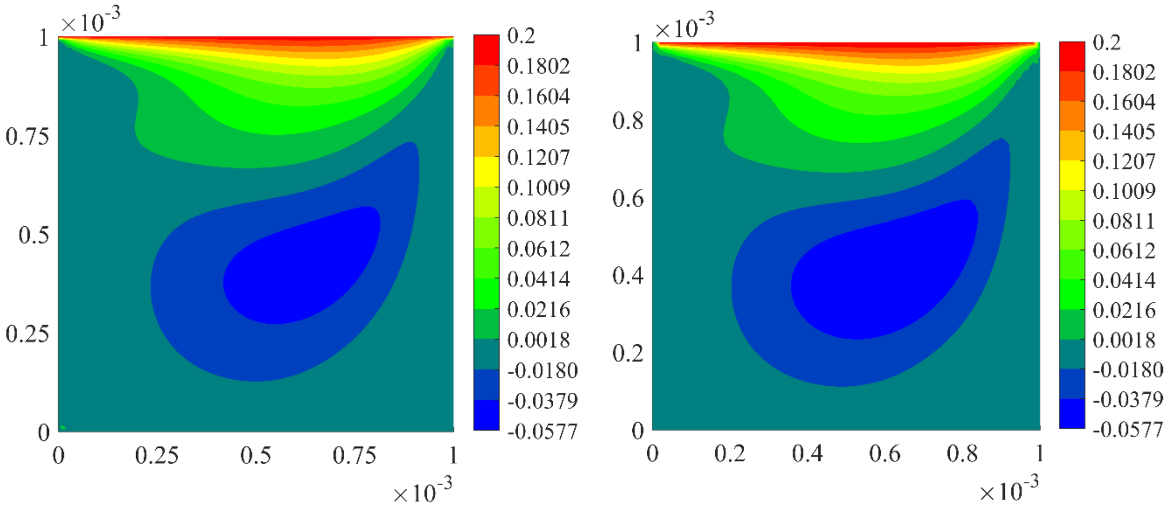


(a) (b)

Fig. 15. Variations of pressure p (Pa) captured by (a): Eulerian incompressible PDDO model, (b): ANSYS Fluent for Case 1 with $Re = 1$

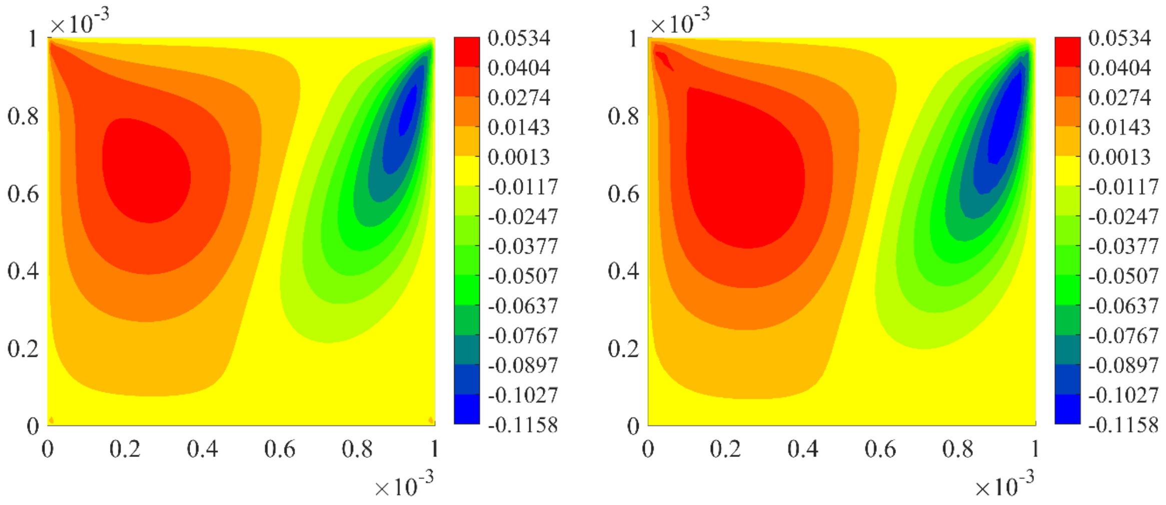
Case 2: $u_0 = 0.2\ (m/s)$, $Re = 200$

Fig. 16-Fig. 18 show the converged results of velocity and pressure fields for Case 2 with $Re = 200$. As can be seen from Fig. 16-Fig. 18, the velocities predicted by the Eulerian incompressible PDDO model agree well with those in ANSYS Fluent. Similarly, as shown in Fig. 19, the pressure fields captured by the Eulerian incompressible PDDO model and ANSYS Fluent have a very good agreement. Both results show two critical regions on the top-left and top-right corners of the fluid domain.



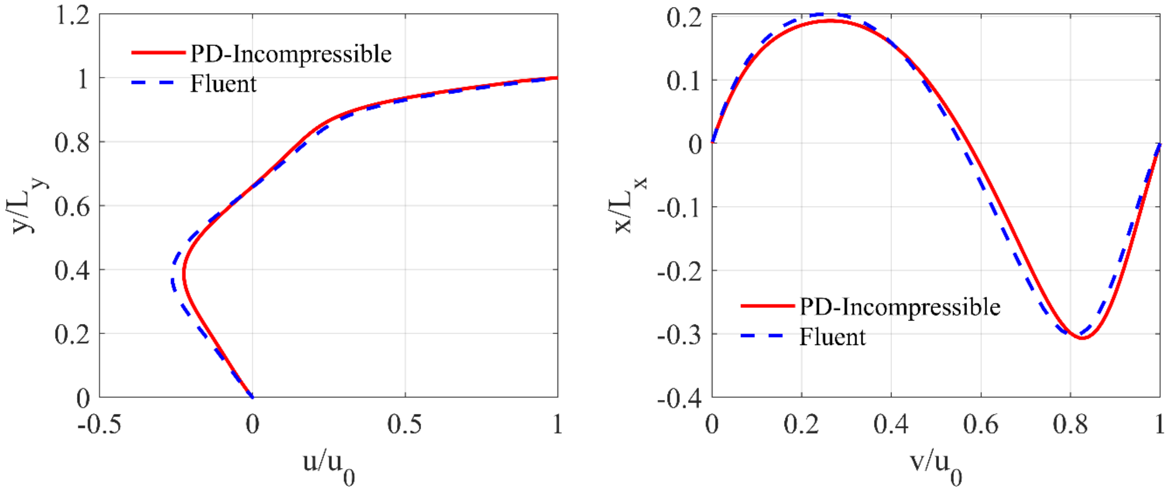
(a) (b)

Fig. 16. Variations of velocity u (m/s) captured by (a): Eulerian incompressible PDDO model, (b): ANSYS Fluent for Case 2 with $Re = 200$



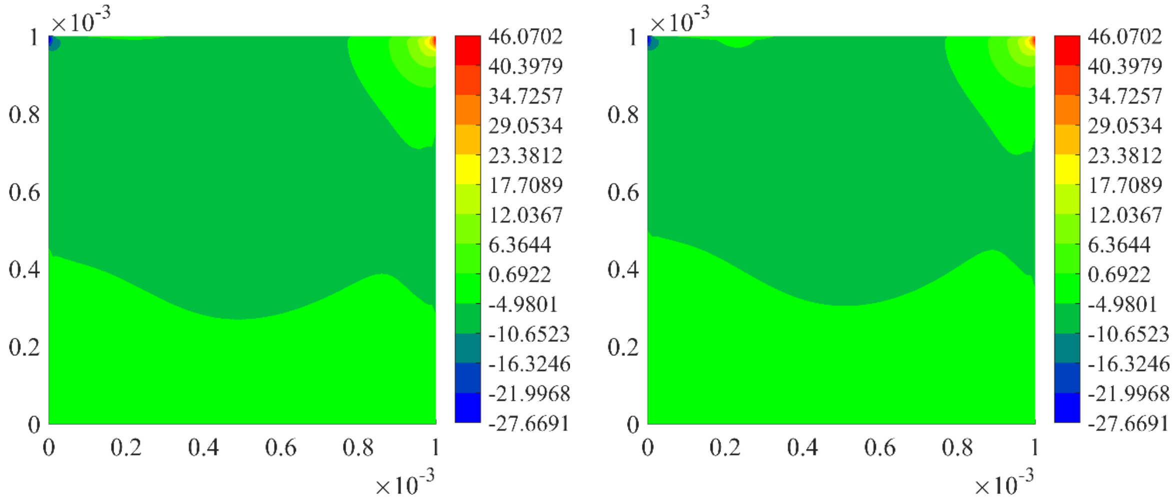
(a) (b)

Fig. 17. Variations of velocity v (m/s) captured by (a): Eulerian incompressible PDDO model, (b): ANSYS Fluent for Case 2 with $Re = 200$



(a) (b)

Fig. 18. Variations of relative velocities along central lines captured by the Eulerian incompressible PDDO model and ANSYS Fluent (a): u/u_0 versus y/L at $x = L/2$, (b): v/u_0 versus x/L at $y = L/2$ for Case 2 with $Re = 200$



(a) (b)

Fig. 19. Variations of pressure p (Pa) captured by (a): Eulerian incompressible PDDO model, (b): ANSYS Fluent for Case 2 with $Re = 200$

5.2. Flow inside an open channel

In this section, the proposed Eulerian incompressible PDDO model is used to investigate a 2D fluid flow inside an open channel as shown in Fig. 20(a). The open channel has a length of $L_x = 0.005$ m and a width of $L_y = D = 0.001$ m. The fluid has density of $\rho = 997.1 \text{ kg/m}^3$ and viscosity of $\mu = 0.00089$ Pa. s.

The fluid is subjected to velocity inlet boundary condition on the left side as

$$u(x = 0, y, t) = u_0 \tag{57a}$$

$$v(x = 0, y, t) = 0 \tag{57b}$$

where $u_0 = 0.05\ \mathrm{m/s}$.

Thus, the Reynolds number for this problem can be estimated as

$$Re = \frac{\rho u_0 D}{\mu} = 56.02 \tag{58}$$

On the top and bottom boundaries, the fluid is subjected to no-slip boundary conditions as

$$u(x, y = L_y, t) = v(x, y = L_y, t) = 0 \tag{59a}$$

$$u(x, y = 0, t) = v(x, y = 0, t) = 0 \tag{59b}$$

Meanwhile, on the right edge, the fluid is subjected to the zero-pressure boundary condition as

$$p(x = L_x, y, t) = 0 \tag{60}$$

The domain is uniformly discretized with a mesh size of $\Delta x = L_x/200$ and the time step size of $\Delta t = 10^{-5}$ is used. To apply boundary conditions, three layers of fictitious particles are added on four edges of the fluid domain as shown in red and black in Fig. 20(b). The implementation of the zero-pressure boundary condition for the outlet is presented in Section 4.2. The implementation of no-slip boundary conditions for the top and bottom boundaries is presented in Section 4.1.1 for the horizontal walls.

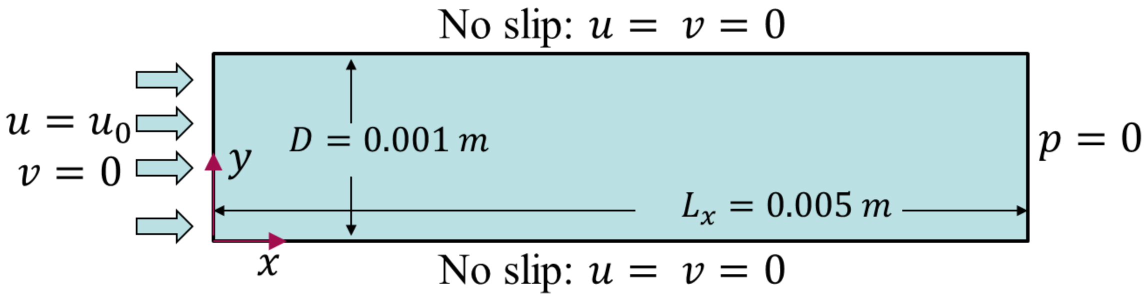
To apply the velocity conditions for the inlet as given in Eq. (57), three layers of fictitious particles are used as shown in black in Fig. 20(b). The velocities of these fictitious particles and the particles located at $x = 0$ are set as

$$u(x \leq 0, 0 \leq y \leq L_y, t) = u_0 \tag{61a}$$

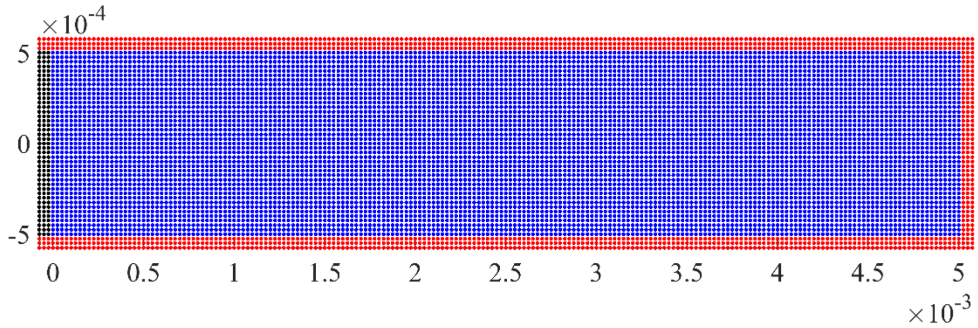
$$v(x \leq 0, 0 \leq y \leq L_y, t) = 0 \tag{61b}$$

Meanwhile, the pressures of these fictitious inlet particles are set equal to the pressure of the particles at $x = 0$ with the same y coordinates as (Hosseini and Feng, 2011; Lee et al., 2008; Shao and Lo, 2003)

$$p(x = -3\Delta x, y, t) = p(x = -2\Delta x, y, t) = p(x = -\Delta x, y, t) = p(x = 0, y, t) \tag{62}$$



(a)



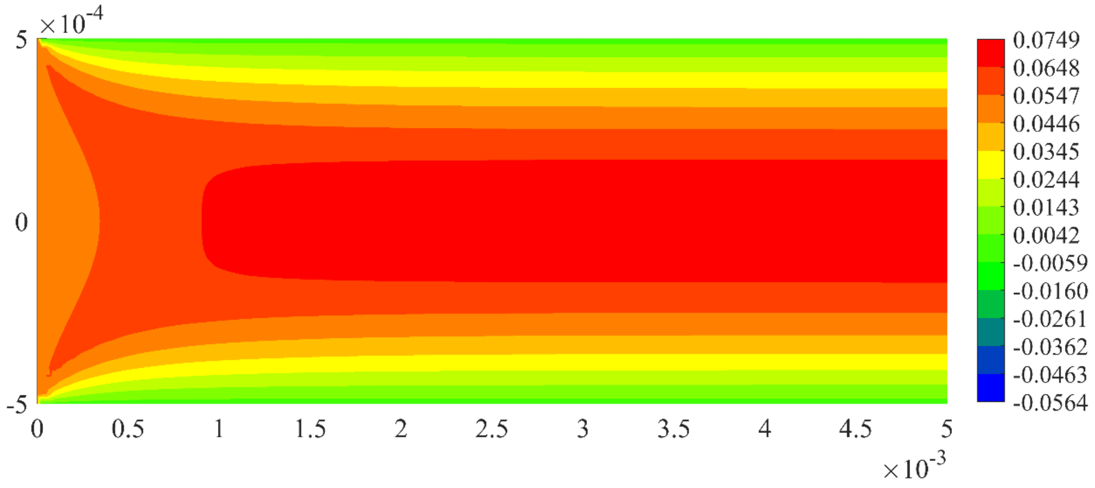
(b)

Fig. 20. Flow inside an open channel (a): geometry, (b) discretized model

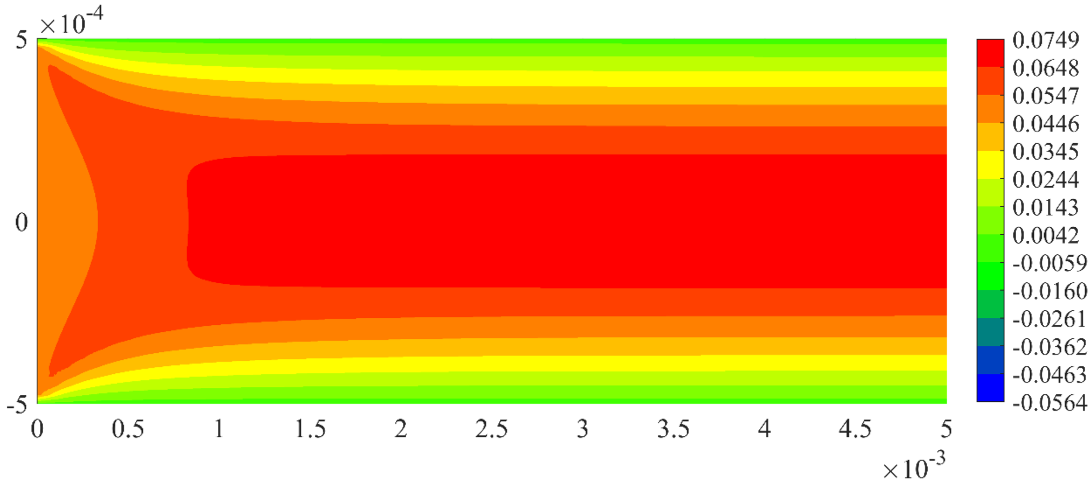
Similar to the previous example, the results predicted by the Eulerian incompressible PDDO model are compared with the results obtained by using ANSYS Fluent. In ANSYS Fluent, the mesh size of $\Delta x = 10^{-5}$ and the steady solution with the SIMPLE scheme is used.

Fig. 21-Fig. 24 show the converged results of velocity and pressure fields obtained by using the Eulerian incompressible PDDO model and ANSYS Fluent. As can be seen from Fig. 21 and Fig. 22, the velocities predicted by the Eulerian incompressible PDDO model agree very well with those in ANSYS Fluent. For further comparison, the relative velocity u/u_0 versus the relative coordinate y/L_y for different cross sections of the channel are shown in Fig. 23. As can be observed from Fig. 23, the results obtained from the Eulerian incompressible PDDO model and ANSYS Fluent have a good agreement.

Fig. 24 shows the variation of the pressure field obtained by using the Eulerian incompressible PDDO model and ANSYS Fluent. As can be seen from the figure, the pressure fields in the Eulerian incompressible PDDO model and ANSYS Fluent have a good agreement. Both results show peak values of pressure at two corners on the left side of the channel. In the Eulerian incompressible PDDO model, the highest value of pressure at two corners on the left side of the channel is 7.19 Pa. Meanwhile, the highest value of pressure at two corners on the left side of the channel in ANSYS Fluent is 7.58. Pa.

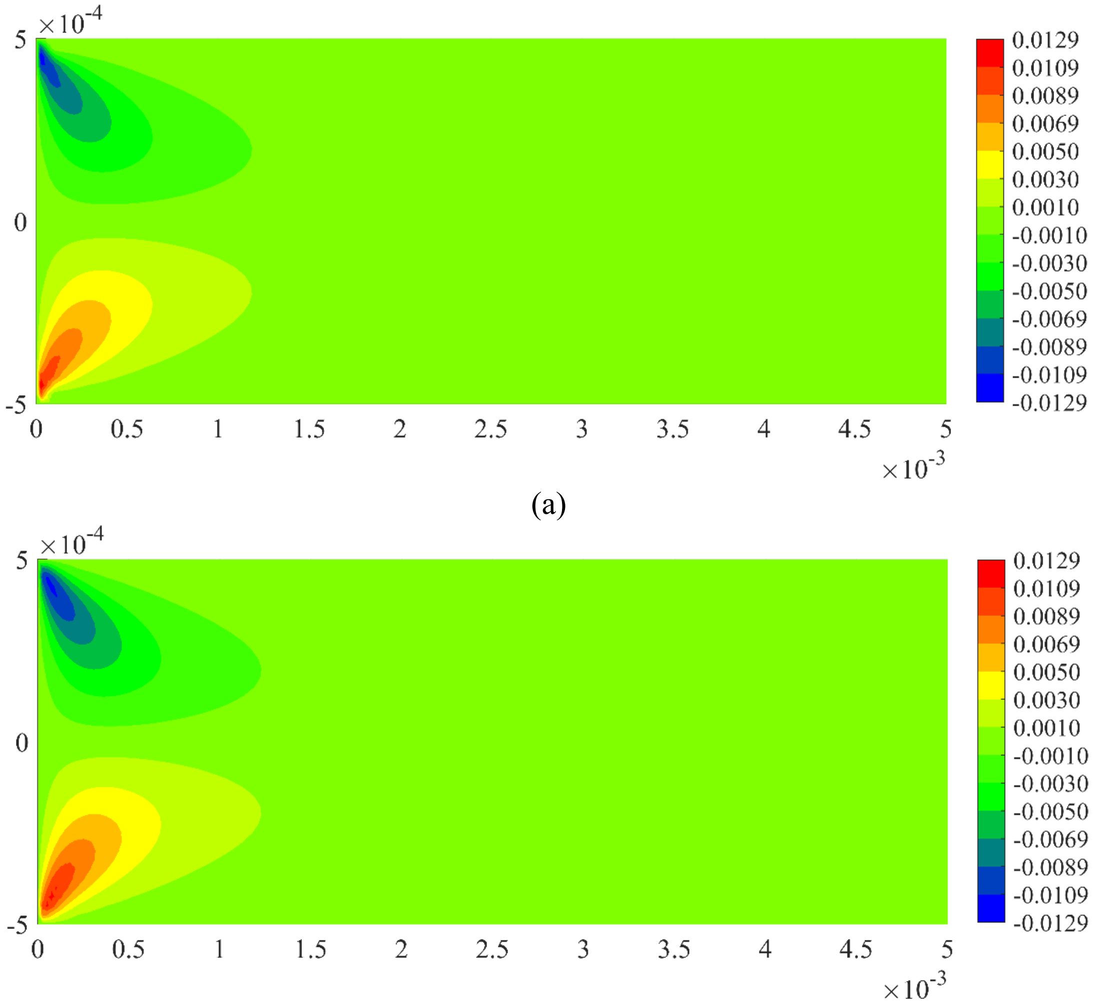


(a)



(b)

Fig. 21. Variations of velocity u (m/s) captured by (a): Eulerian incompressible PDDO model, (b): ANSYS Fluent



(a)

(b)

Fig. 22. Variations of velocity v (m/s) captured by (a): Eulerian incompressible PDDO model, (b): ANSYS Fluent

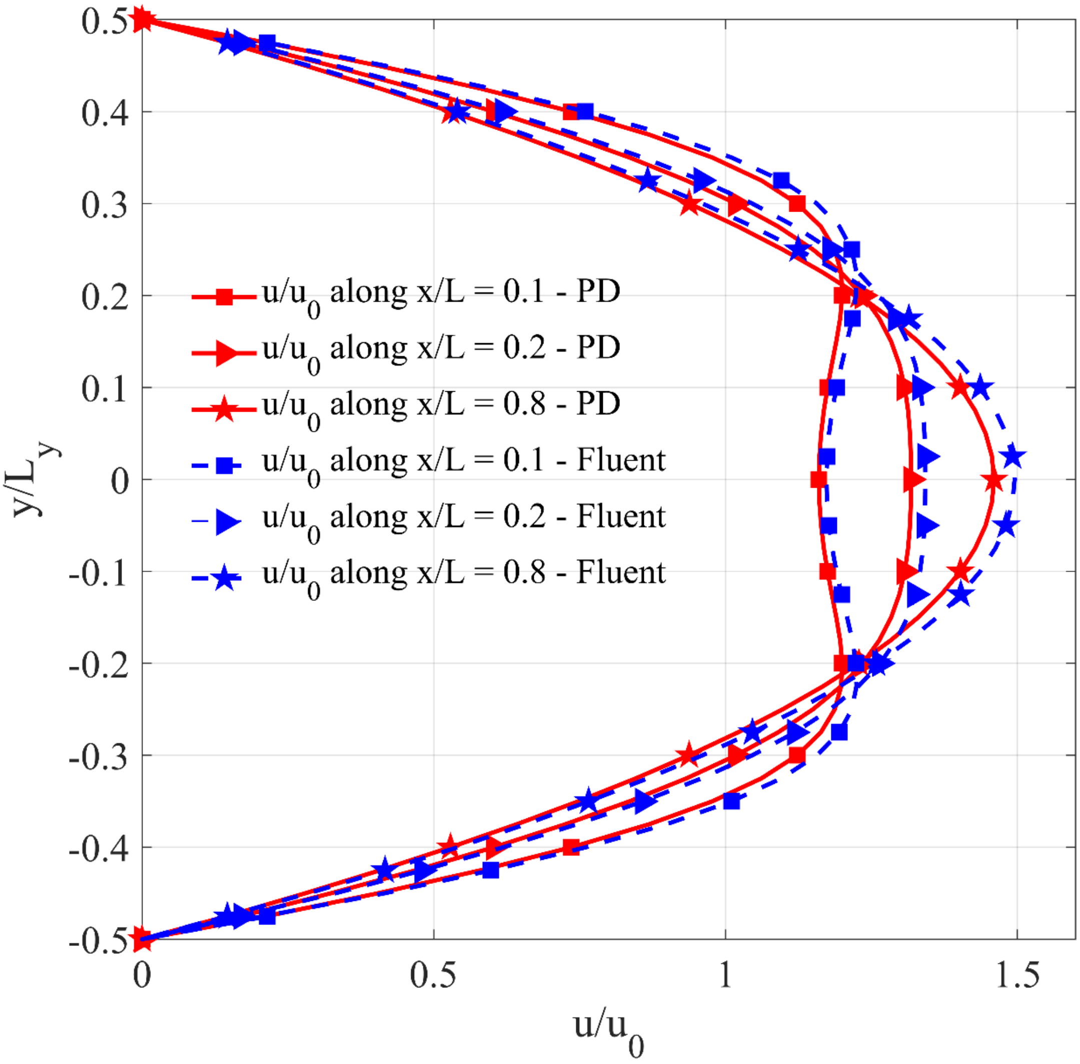
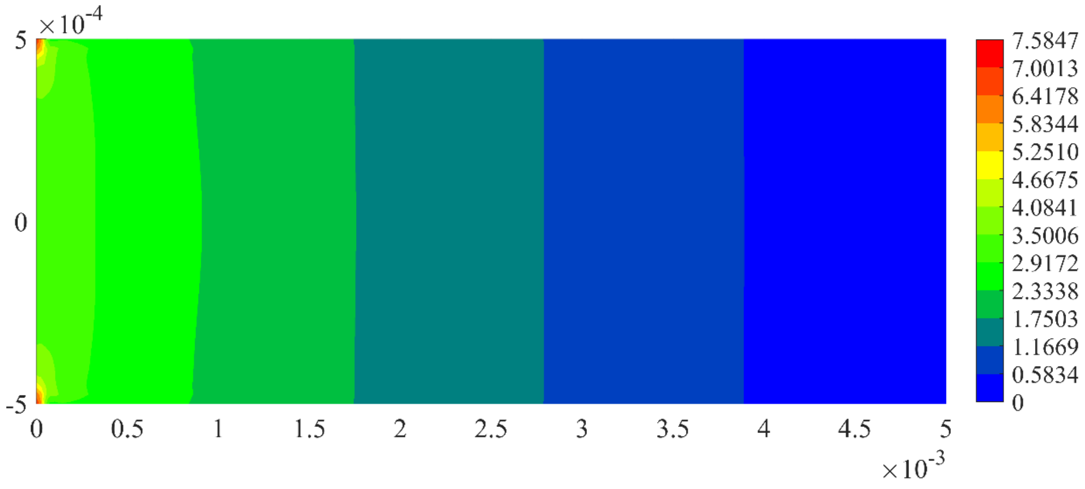
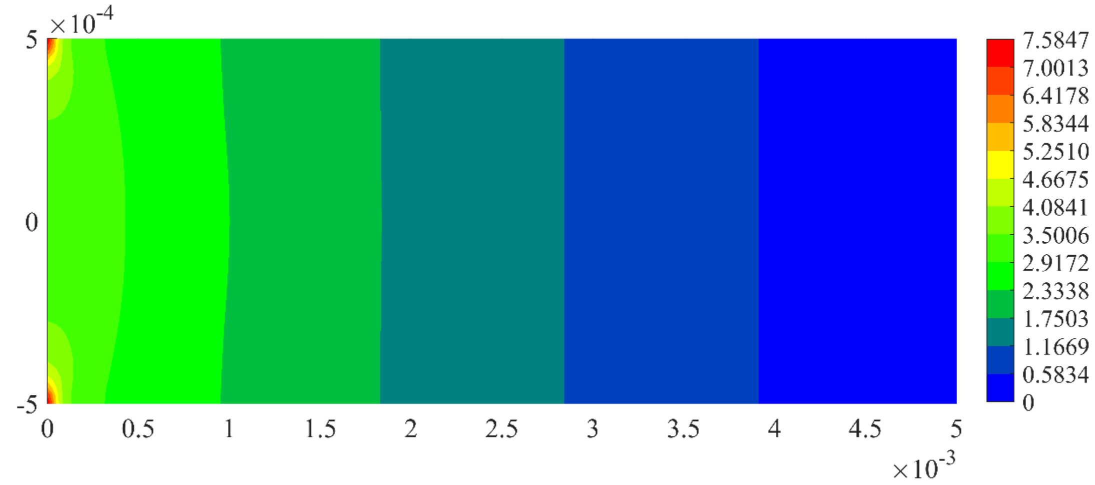


Fig. 23. Relative velocity u/u_0 versus relative coordinate y/L_y at different cross sections of the channel



(a)



(b)

Fig. 24. Variations of pressure p (Pa) captured by (a): Eulerian incompressible PDDO model, (b): ANSYS Fluent

5.3. Flow over a circular cylinder

In this section, a 2D fluid domain with $L_x = 15\text{ m}, L_y = 6\text{ m}$ passing over a cylinder is investigated as shown in Fig. 25. The diameter of the obstacle cylinder is $D = 1$ m. The cylinder is located at $x_c = 5\text{ m}, y_c = 3\text{ m}$ as shown in Fig. 25. The fluid density and viscosity are chosen as $\rho = 1\text{ kg/m}^3$ and $\mu = 1\text{ Pa.s}$.

Similar to the previous example, the fluid is subjected to velocity inlet boundary conditions at the left edge as

$$u(x = 0, y, t) = u_0 \tag{63a}$$

$$v(x = 0, y, t) = 0 \tag{63b}$$

where u_0 is a constant velocity. In this example, two different values of u_0 are investigated as

$$u_0 = \begin{cases} 10 \text{ for Case 1} \\ 100 \text{ for Case 2} \end{cases} \tag{64}$$

Thus, the Reynolds number for this problem can be estimated as

$$Re = \frac{\rho u_0 D}{\mu} = \begin{cases} 10\text{ m/s for Case 1} \\ 100\text{ m/s for Case 2} \end{cases} \tag{65}$$

where $D = 1$ m is the diameter of the cylinder.

On the right side, the fluid is subjected to the zero pressure boundary condition as

$$p(x = L_x, y, t) = 0 \tag{66}$$

On the top and bottom boundaries, the fluid is subjected to no-slip boundary conditions as

$$u(x, y = L_y, t) = 0,\ v(x, y = L_y, t) = 0 \tag{67a}$$

$$u(x, y = 0, t) = 0,\ v(x, y = 0, t) = 0 \tag{67b}$$

On the cylinder boundary, the fluid is also subjected to no-slip boundary conditions as

$$\mathbf{u}^{(n)}\Big|_{\text{cylinder wall}} = 0 \tag{68}$$

$$\frac{\partial p^{(n)}}{\partial r}\Big|_{\text{cylinder wall}} = 0 \tag{69}$$

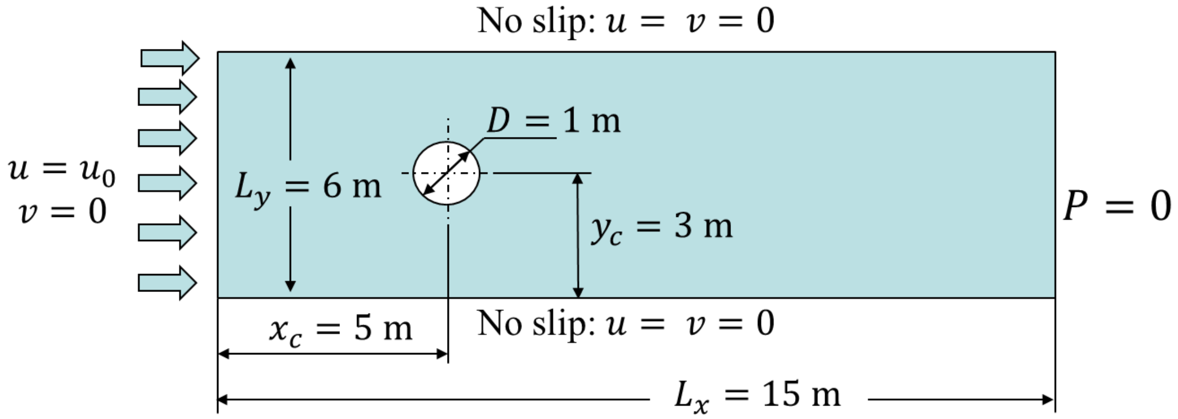


Fig. 25. Flow over a cylinder

In the Eulerian incompressible PDDO model, the fluid domain is discretized using two different mesh sizes for two regions as shown in Fig. 26(a). Specifically, the fluid domain far from the obstacle cylinder is uniformly discretized in Cartesian coordinates with a mesh size of $\Delta x = L_x/500$. Meanwhile, the fluid domain surrounding the obstacle cylinder is discretized in cylindrical coordinates, (r, θ), with the mesh size of $\Delta\theta = 0.0149$ rad and $\Delta r = 0.69\Delta x$.

To apply boundary conditions, three layers of fictitious particles are added on four sides of the fluid domain as well as inside the circular cylinder as shown in red and black in Fig. 26(a). Therefore, there are 137156 particles in the PD discretized model including fictitious particles.

In ANSYS Fluent, the fluid domain is also discretized using different mesh sizes for two regions as shown in Fig. 26(b). Specifically, the fluid domain far from the cylinder is discretized with an average mesh size of 0.1 m. Meanwhile, the fluid domain surrounding the cylinder is discretized in cylindrical coordinates, (r, θ), with the mesh size of $\Delta\theta = 0.0321$ rad and $\Delta r = 1.6 \times 10^{-2}$ m. Therefore, there are 16233 nodes in the discretized model in the Fluent model.

In the Eulerian incompressible PDDO model, the implementation of the no-slip boundary conditions in Eq. (67) for the top and bottom walls is presented in Section 4.1.1. Meanwhile, the zero-pressure condition in Eq. (66) for the right side is presented in Section 4.3.

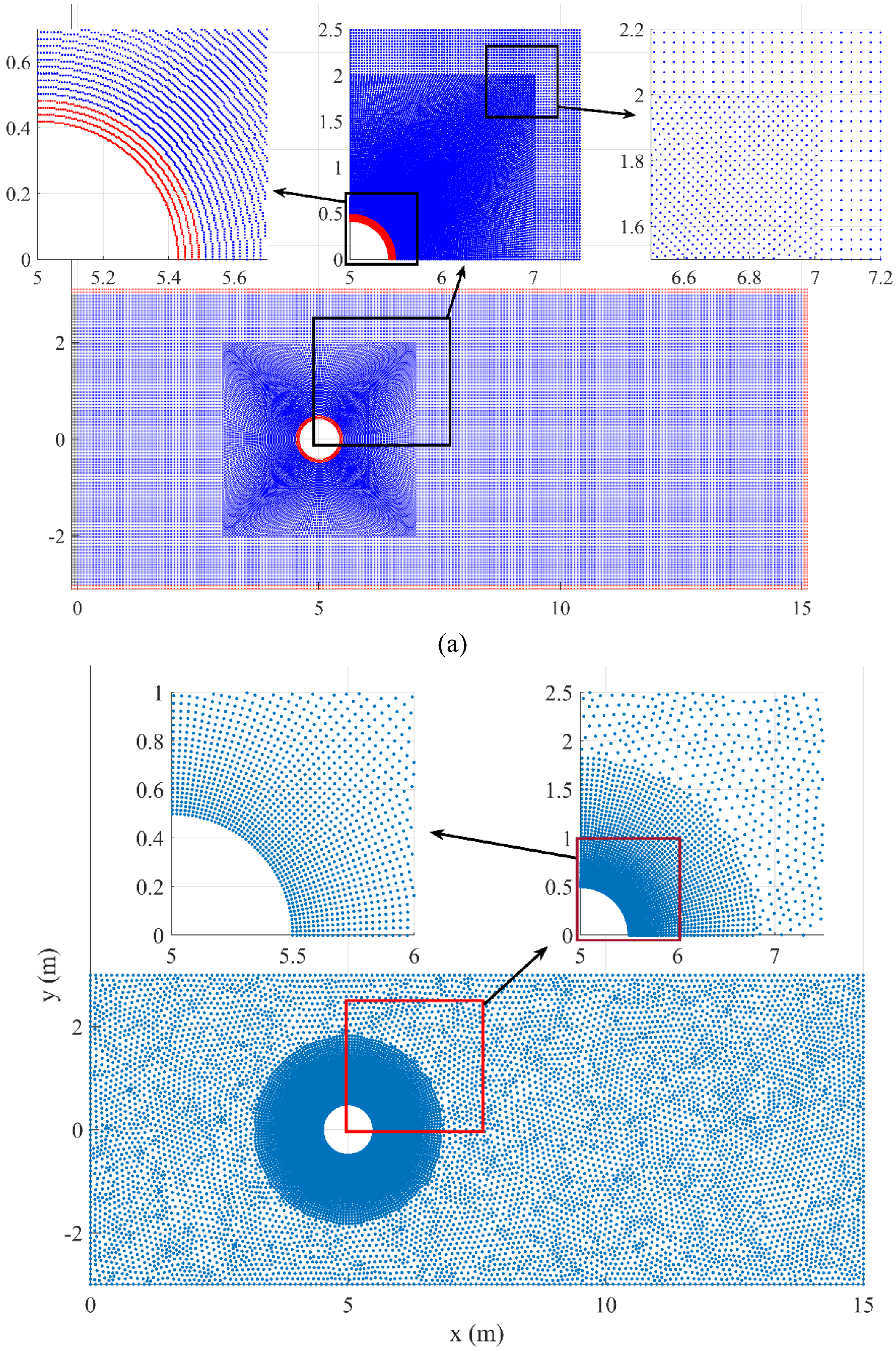
To apply velocity conditions given in Eq. (63) for the inlet boundary, three layers of fictitious particles are used as shown in black in Fig. 26(a). The velocities of these fictitious particles are defined as

$$u(x \le 0, 0 \le y \le L_y, t) = u_0 \tag{70a}$$

$$v(x \le 0, 0 \le y \le L_y, t) = 0 \tag{70b}$$

Meanwhile, the pressures of these fictitious particles are set equal to the pressure of the particles at $x = 0$ with the same y coordinates as

$$p(x=-3\Delta x,y,t)=p(x=-2\Delta x,y,t)=p(x=-\Delta x,y,t)=p(x=0,y,t) \tag{71}$$



(a)

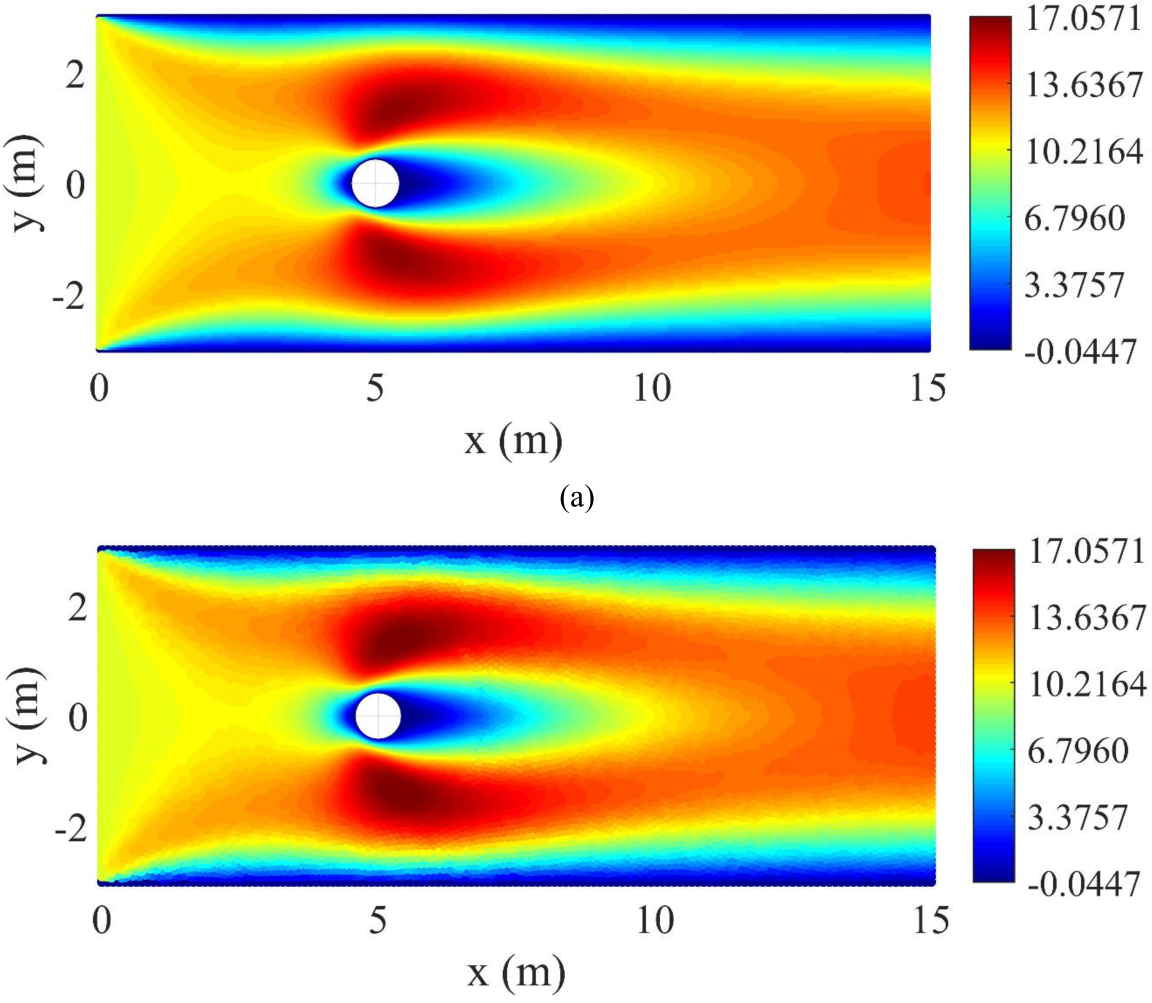
(b)

Fig. 26. Model discretization (a): particles in the Eulerian incompressible PDDO model, (b): nodes in ANSYS Fluent

Case 1: $u_0 = 10\ m/s, Re = 10$

For case 1 with $Re = 10$, the time step size used in the Eulerian incompressible PDDO model is $\Delta t = 10^{-4}$ s. In FLUENT, the transient analysis with the same time step is used. The transient analyses in both the Eulerian incompressible PDDO model and ANSYS Fluent are run for 40000 time steps, and both simulations in the Eulerian incompressible PDDO model and ANSYS Fluent are converged.

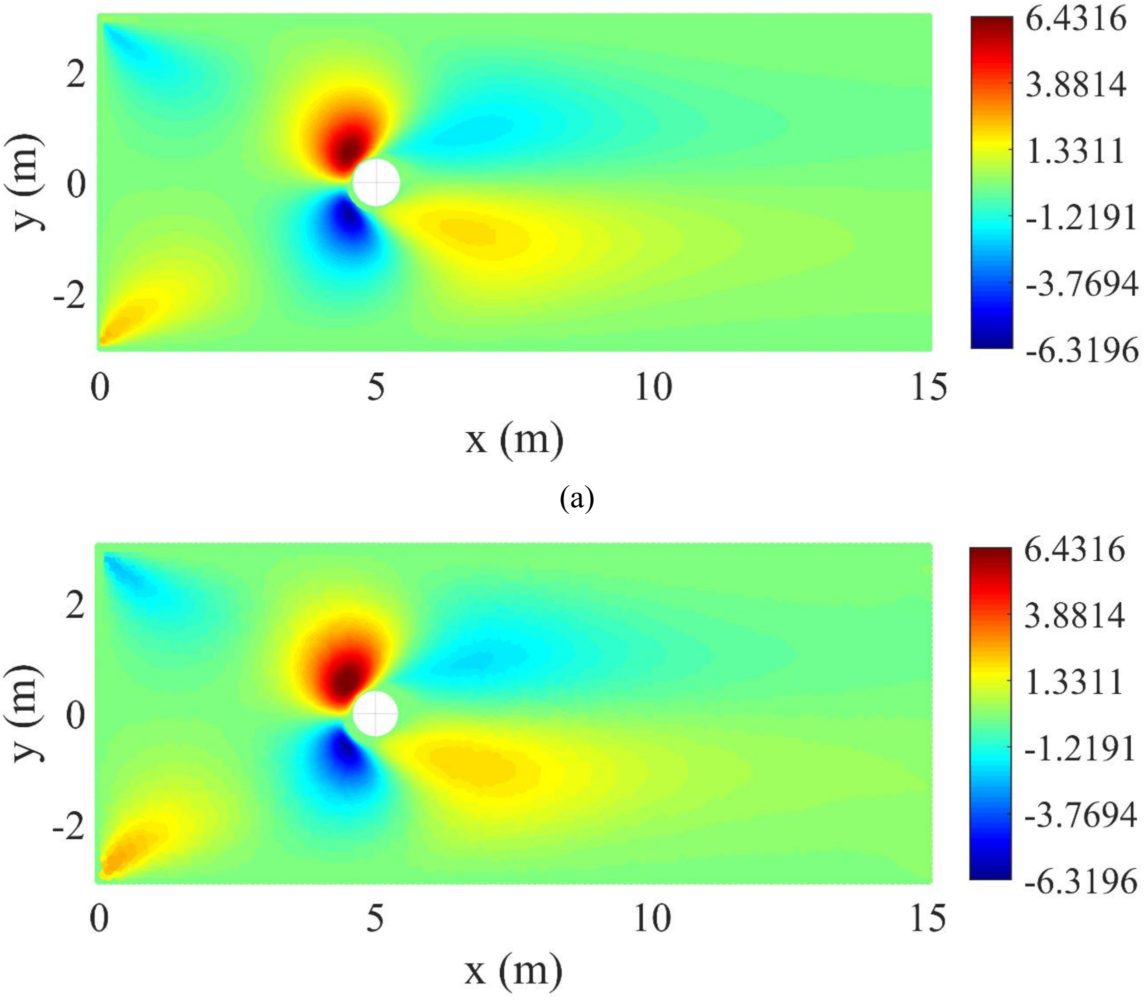
Fig. 27-Fig. 28 show the variation of velocity components, u and v, for the converged solution results in the Eulerian incompressible PDDO model and ANSYS Fluent at time of $t = 4$ s. As can be seen from the figures, the results obtained by the Eulerian incompressible PDDO model and ANSYS Fluent match very well. Fig. 29 shows the variation of pressure in the fluid domain for the converged solution results at time of $t = 4$ s. As can be seen from Fig. 29, the pressure values obtained by the Eulerian incompressible PDDO model and ANSYS Fluent agree very well. Both results show the critical locations of the pressure field are at two corners on the left side as well as the location in front of the cylinder.



(a)

(b)

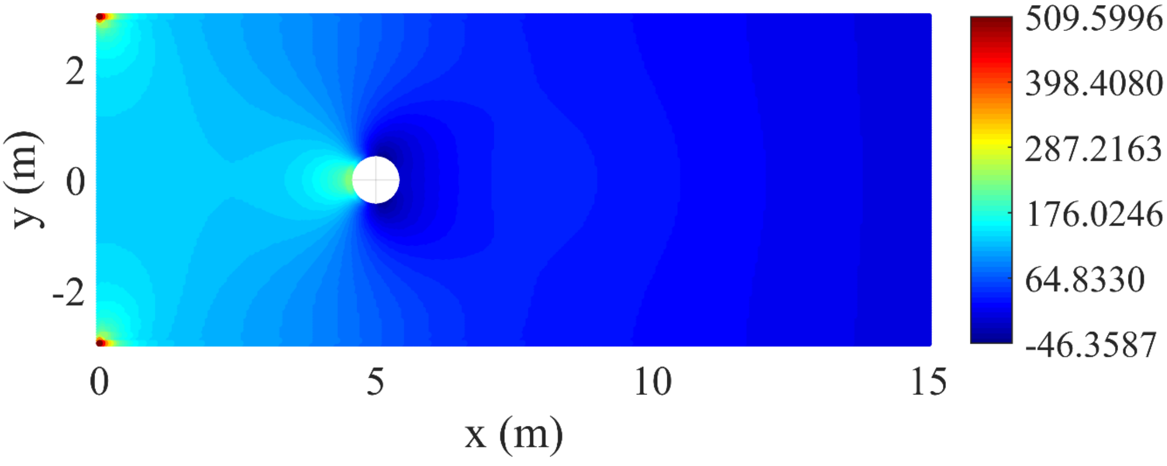
Fig. 27. Variations of velocity u (m/s) captured by (a): Eulerian incompressible PDDO model, (b): ANSYS Fluent for Case 1 with $Re = 10$



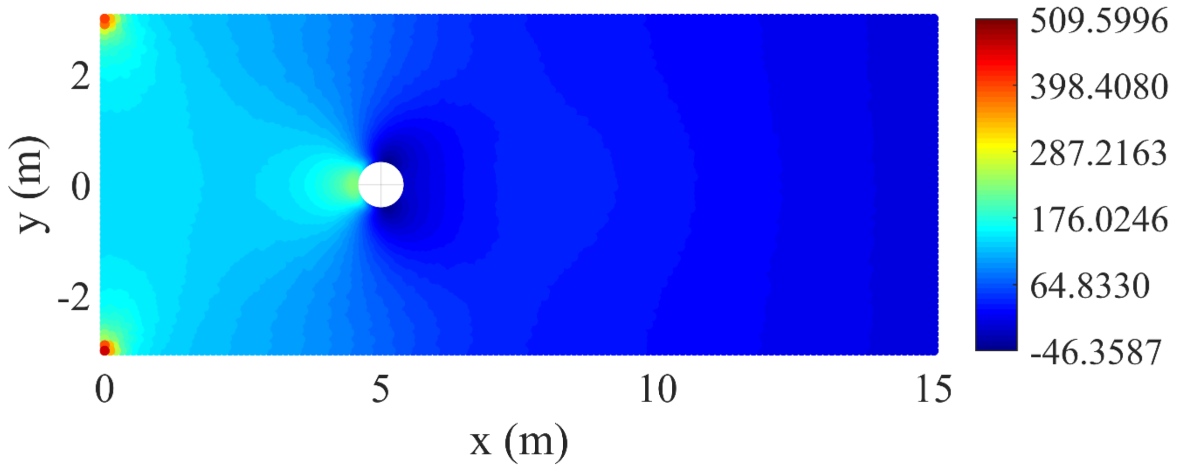
(a)

(b)

Fig. 28. Variations of velocity v (m/s) captured by (a): Eulerian incompressible PDDO model, (b): ANSYS Fluent for Case 1 with $Re = 10$



(a)

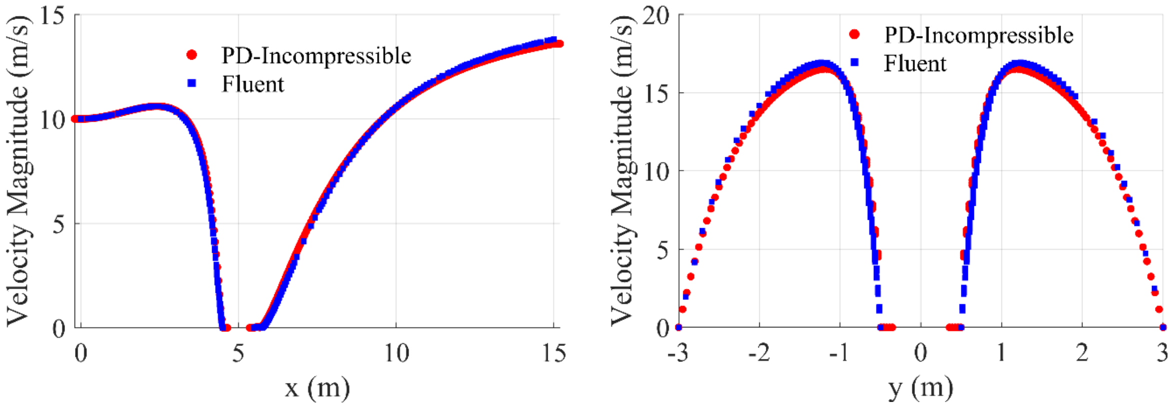


(b)

Fig. 29. Variations of pressure p (Pa) captured by (a): Eulerian incompressible PDDO model, (b): ANSYS Fluent for Case 1 with $Re = 10$

For further comparison, the variations of velocity magnitude and pressure along the lines at $y = 0$ and $x = 5$ m are presented as shown in Fig. 30 and Fig. 31. As shown in Fig. 30(a), because no-slip boundary condition is applied at the circular cylinder, the velocity magnitude reduces to zero at the locations right before and right after the cylinder. At the right end of the fluid domain, the velocity magnitude reaches the fully developed value at around 13.81 m/s. Similarly, as shown in Fig. 30(b), because of no-slip boundary conditions, the velocity magnitude reduces to zero at the locations adjacent to the circular cylinder, the top and bottom boundaries. Meanwhile, at the locations of ($x = 5$ m, $y = \pm 1.2$ m), the velocity magnitude is around 16.9 m/s. As can be seen from Fig. 30, the variations of velocity magnitude along the lines at $y = 0$ and $x = 5$ m captured by the Eulerian incompressible PDDO model and ANSYS Fluent agree very well.

As shown in Fig. 31(a), both results show that at the location right before the cylinder, the pressure is around 220 Pa. Meanwhile, at the location right after the cylinder, the pressure is zero. As shown in Fig. 31(b), both results give nearly the same value of pressure at around −37 Pa at the locations next to the cylinder. As can be observed from the figures, both results show a good agreement for the variation of pressure.



(a) (b)

Fig. 30. Variation of velocity magnitude along the lines at (a): $y = 0$, (b): $x = 5$ m

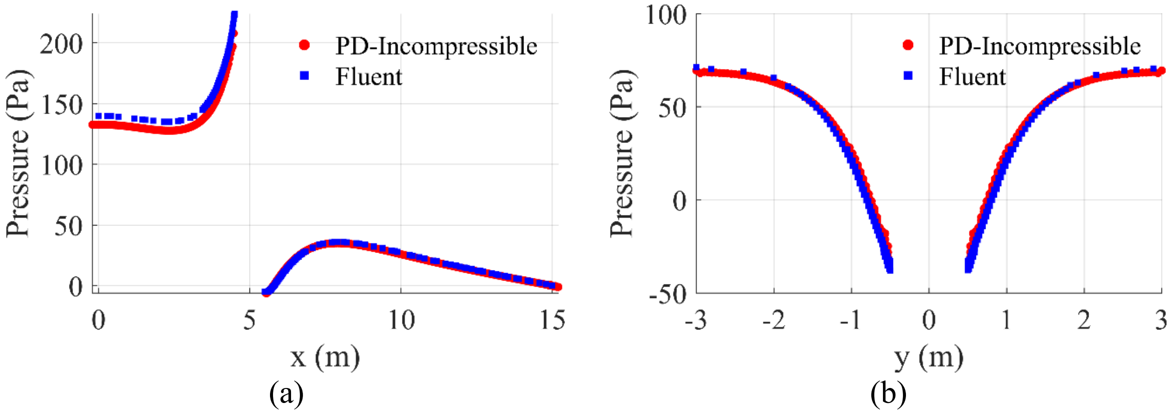
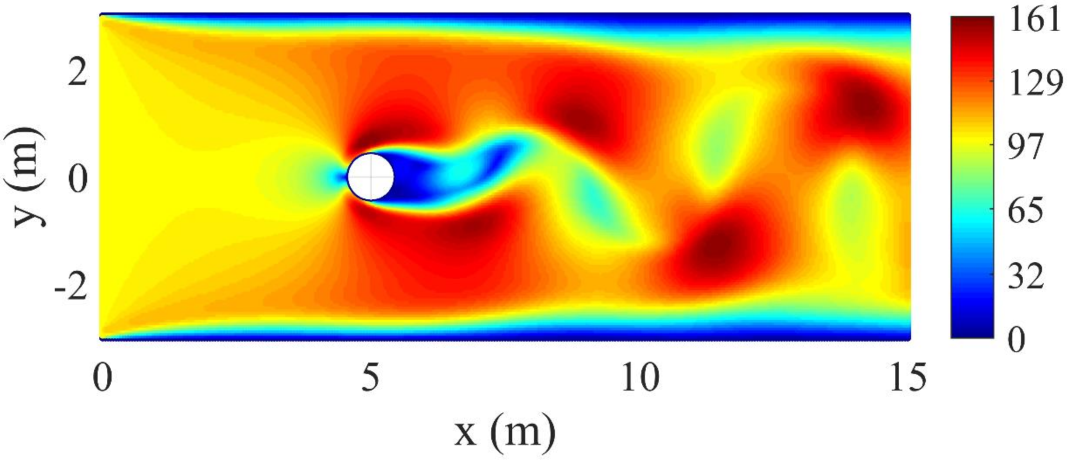


Fig. 31. Variation of pressure along the lines at (a): $y = 0$, (b): $x = 5$ m

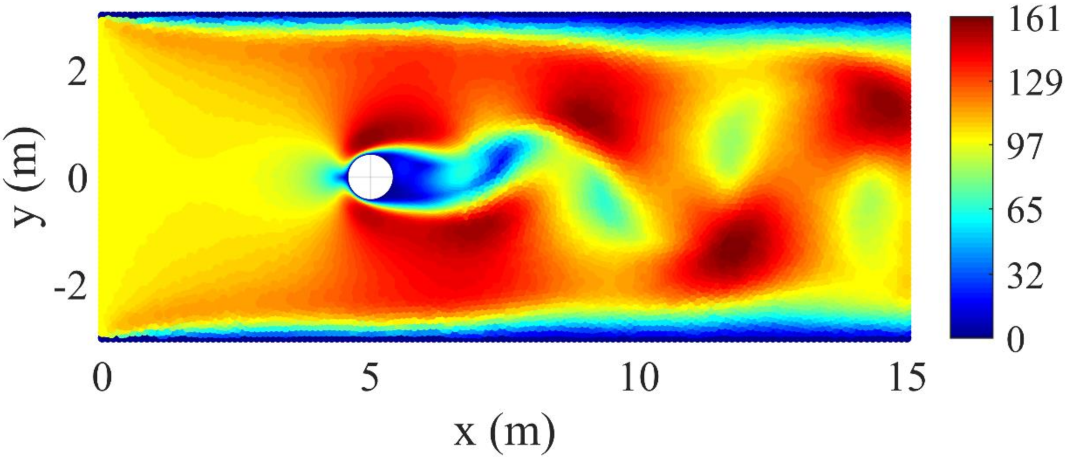
Case 2: $u_0 = 100\ m/s, Re = 100$

For case 2 with $Re = 100$, the time step size used in the Eulerian incompressible PDDO model is $\Delta t = 5 \times 10^{-5}$ s. In ANSYS Fluent, the transient analysis with the same time step is used. The transient analyses in both the Eulerian incompressible PDDO model and ANSYS Fluent are run for 40000 time steps to capture vortex shedding phenomena.

Fig. 32-Fig. 33 show the vortex shedding contours of velocity magnitude and pressure for the case of vortex tail up in the Eulerian incompressible PDDO model and ANSYS Fluent. As can be seen from the figures, the velocity magnitude and pressure fields obtained by the Eulerian incompressible PDDO model and ANSYS Fluent match very well. For further comparison, the variations of velocity magnitude and pressure along the lines at $y = 0$ and $x = 5$ m are investigated as shown in Fig. 34 and Fig. 35. As shown in Fig. 34(a) and Fig. 35(a), because of the vortex, the values of the velocity magnitude and pressure fluctuate along the region behind the cylinder. As shown in Fig. 34(b), in the case of vortex tail up, the values of the velocity magnitude on the positive y direction are slightly higher than those on the negative y direction. On the other hand, as shown in Fig. 35(b), the values of pressure on the positive y direction are slightly lower than those on the negative y direction. Moreover, as can be seen from Fig. 34 and Fig. 35, the results obtained by the Eulerian incompressible PDDO model and ANSYS Fluent agree very well.

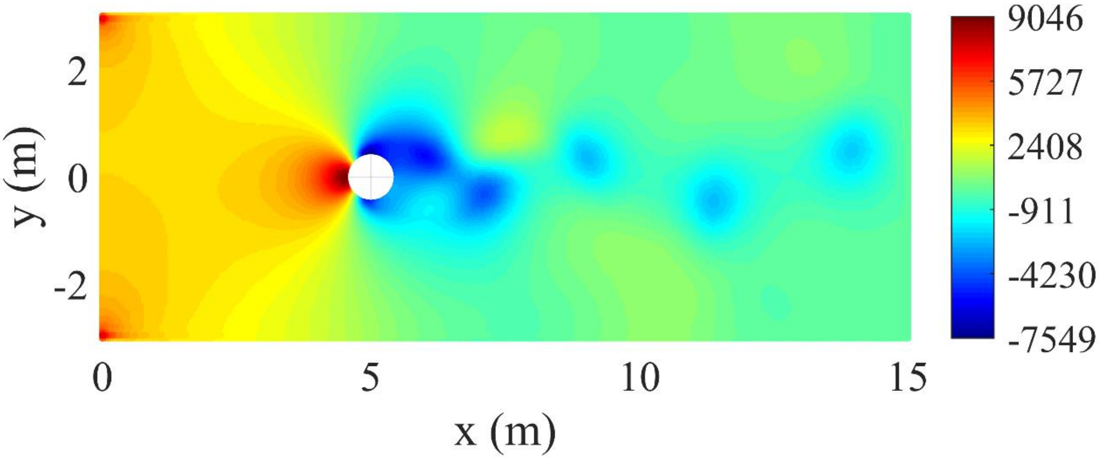


(a)

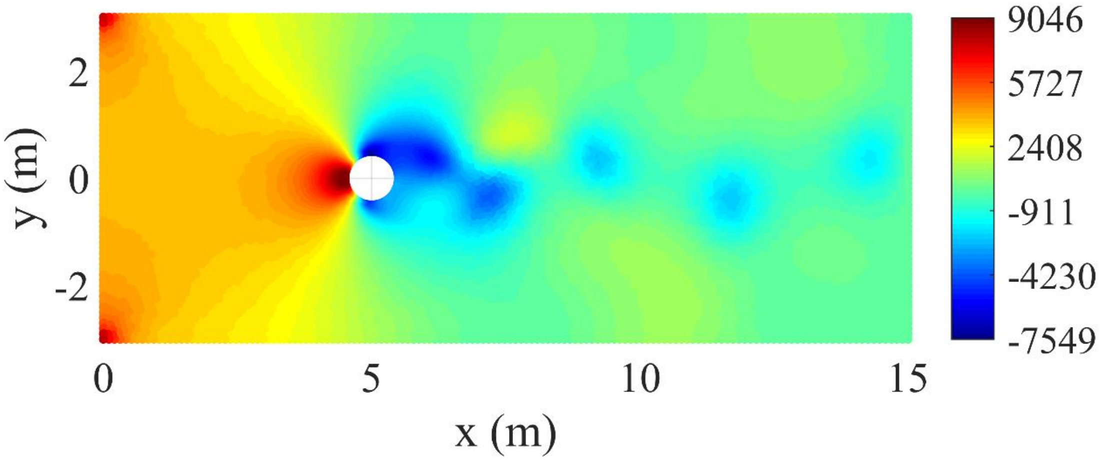


(b)

Fig. 32. The comparison of velocity magnitude for the case of vortex tail up obtained by (a): Eulerian incompressible PDDO model, (b): ANSYS Fluent

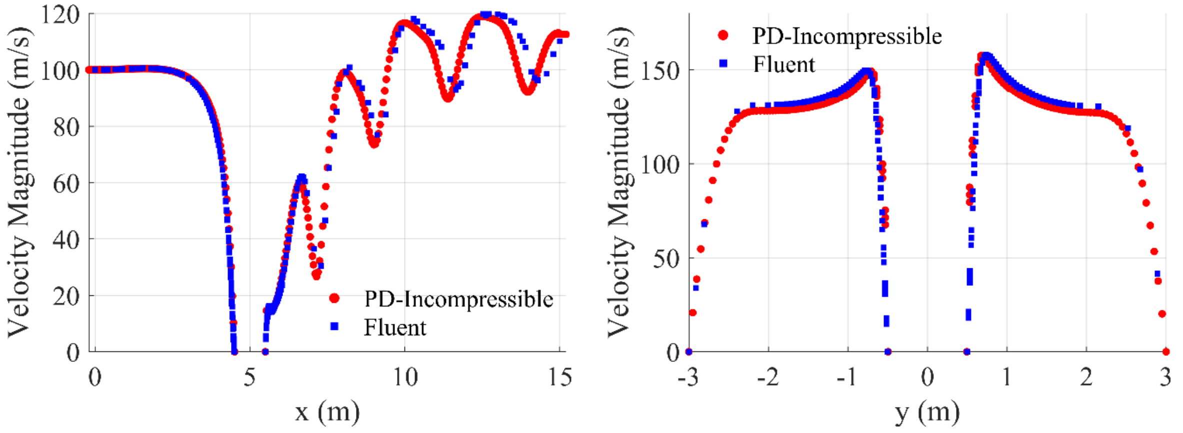


(a)



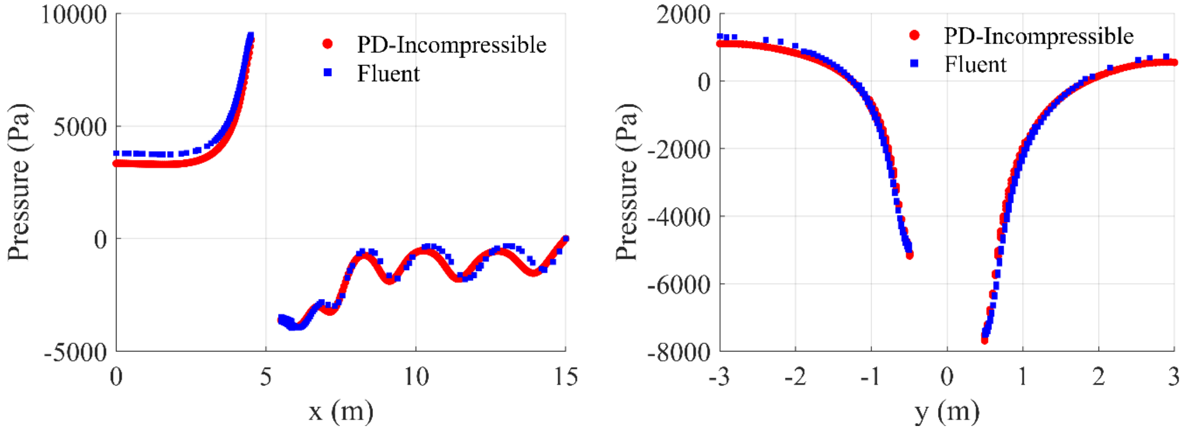
(b)

Fig. 33. The comparison of pressure values for the case of vortex tail up obtained by (a): Eulerian incompressible PDDO model, (b): ANSYS Fluent



(a) (b)

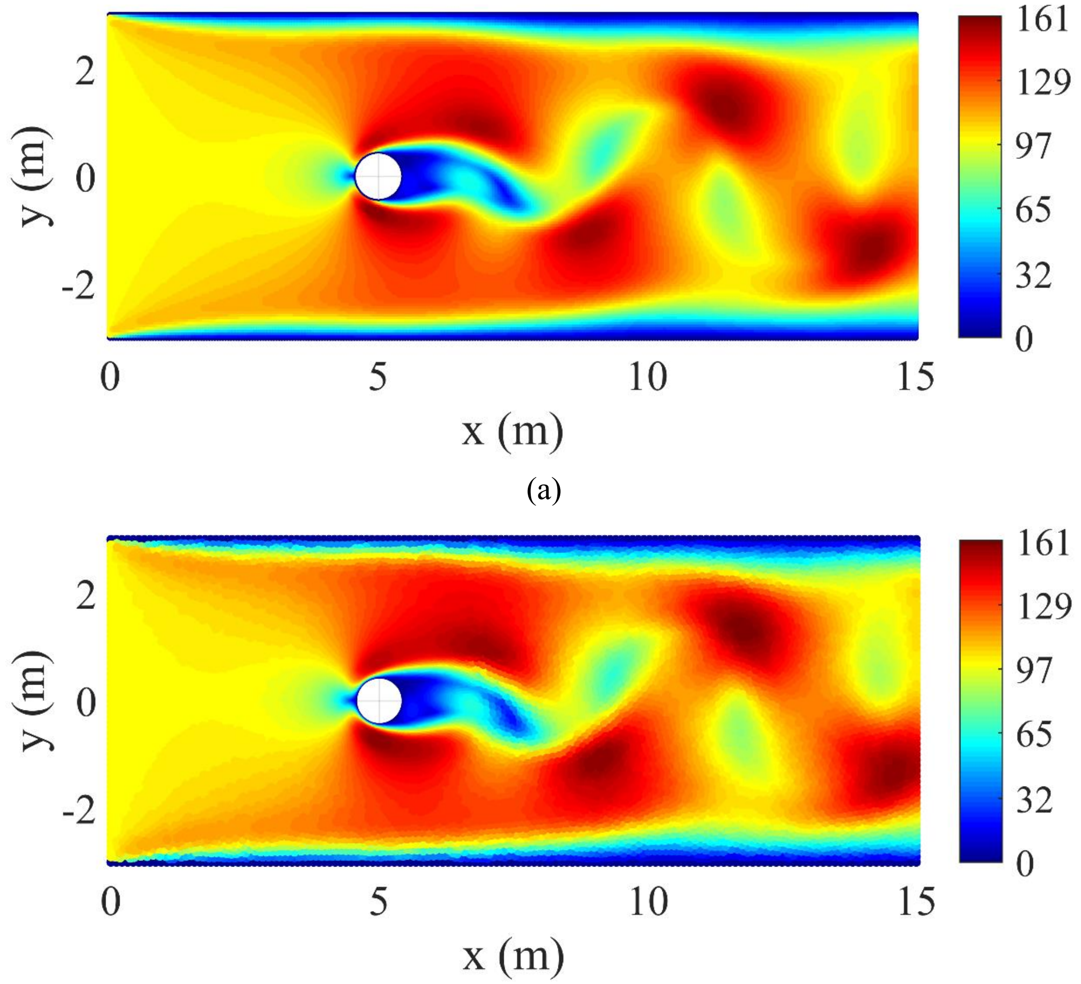
Fig. 34. Variations of velocity magnitudes at (a): $y = 0$, (b): $x = 5$ m for the case of vortex tail up captured by the Eulerian incompressible PDDO model and ANSYS Fluent



(a) (b)

Fig. 35. Variations of pressure at (a): $y = 0$, (b): $x = 5$ m for the case of vortex tail up captured by the Eulerian incompressible PDDO model and ANSYS Fluent

Fig. 36 and Fig. 37 show the vortex shedding contours of velocity magnitude and pressure for the case of vortex tail down in the Eulerian incompressible PDDO model and ANSYS Fluent. As can be seen from the figures, the velocity magnitude and pressure fields obtained by the Eulerian incompressible PDDO model and ANSYS Fluent match very well. Similar to the case of vortex tail up, the variations of velocity magnitude and pressure along the lines at $y = 0$ and $x = 5$ m are investigated as shown in Fig. 38 and Fig. 39. Similar to the case of vortex tail up, the values of the velocity magnitude and pressure fluctuate along the region behind the cylinder as shown in Fig. 38(a) and Fig. 39(a). As shown in Fig. 38(b), in the case of vortex tail down, the values of the velocity magnitude on the positive y direction are slightly lower than those on the negative y direction. Meanwhile, as shown in Fig. 35(b), the values of pressure on the positive y direction are slightly higher than those on the negative y direction. As can be seen from these figures, the results obtained by the Eulerian incompressible PDDO model and ANSYS Fluent also agree very well.



(a)

(b)

Fig. 36. The comparison of velocity magnitude for the case of vortex tail down obtained by (a): Eulerian incompressible PDDO model, (b): ANSYS Fluent

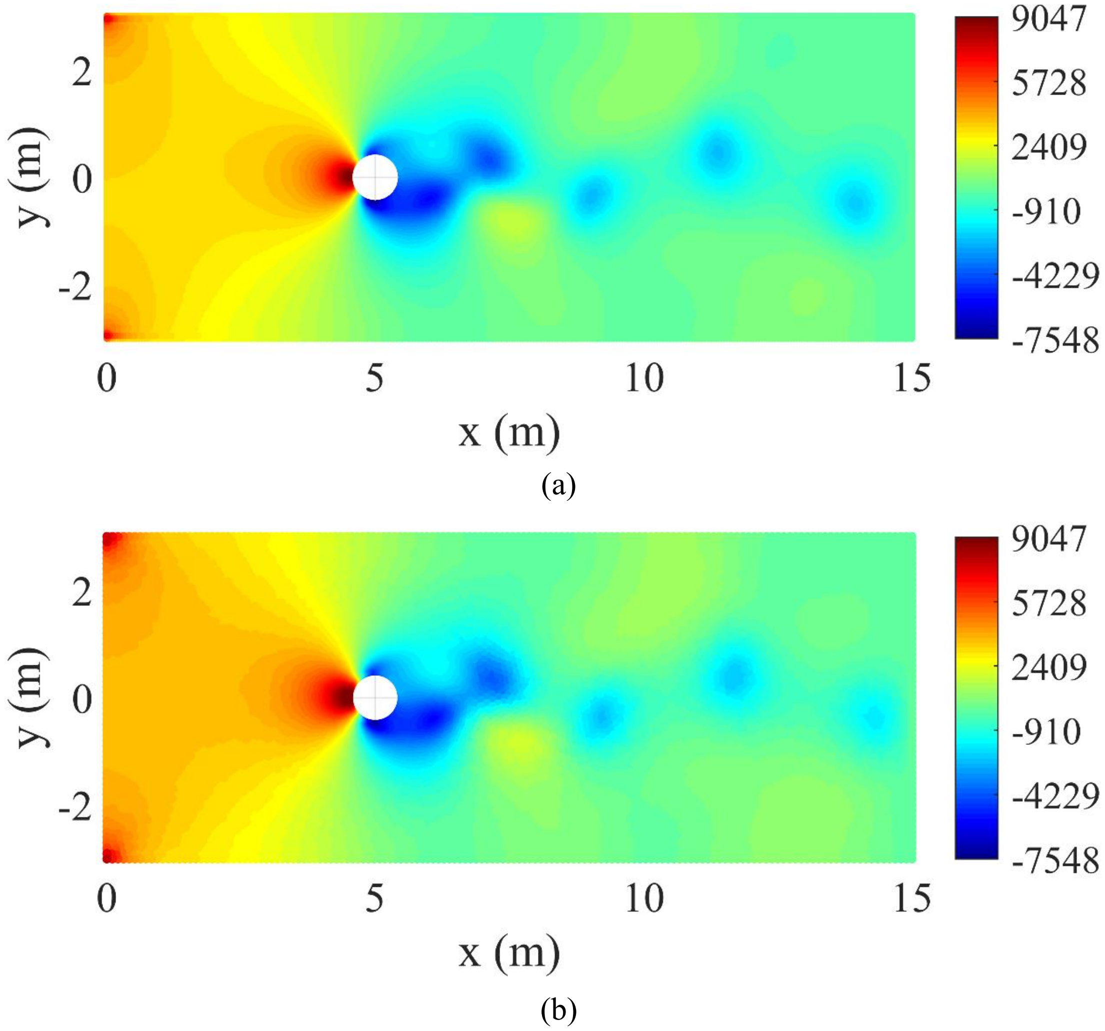


Fig. 37. The comparison of pressure values for the case of vortex tail down obtained by (a): Eulerian incompressible PDDO model, (b): ANSYS Fluent

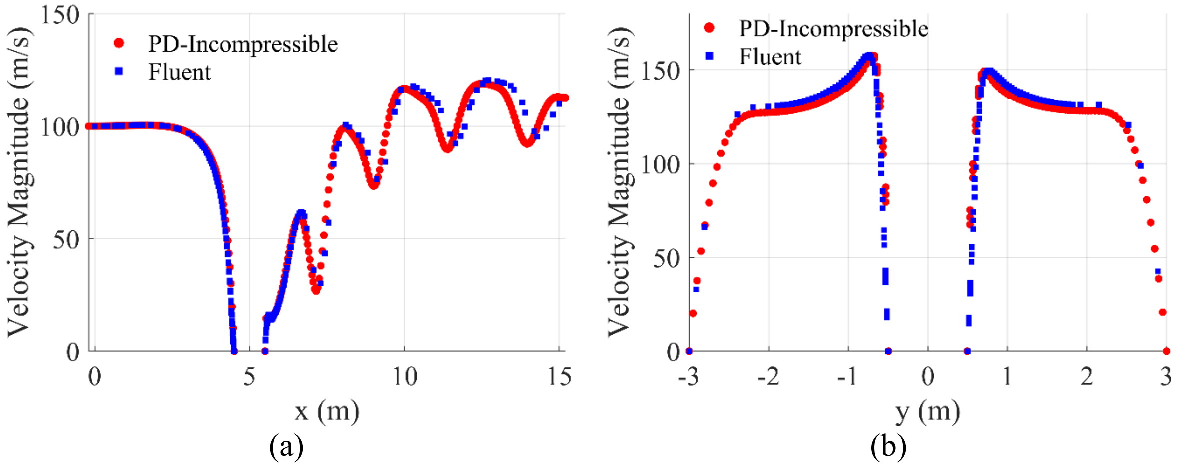


Fig. 38. Variations of velocity magnitudes at (a): $y = 0$, (b): $x = 5$ m for the case of vortex tail down captured by the Eulerian incompressible PDDO model and ANSYS Fluent

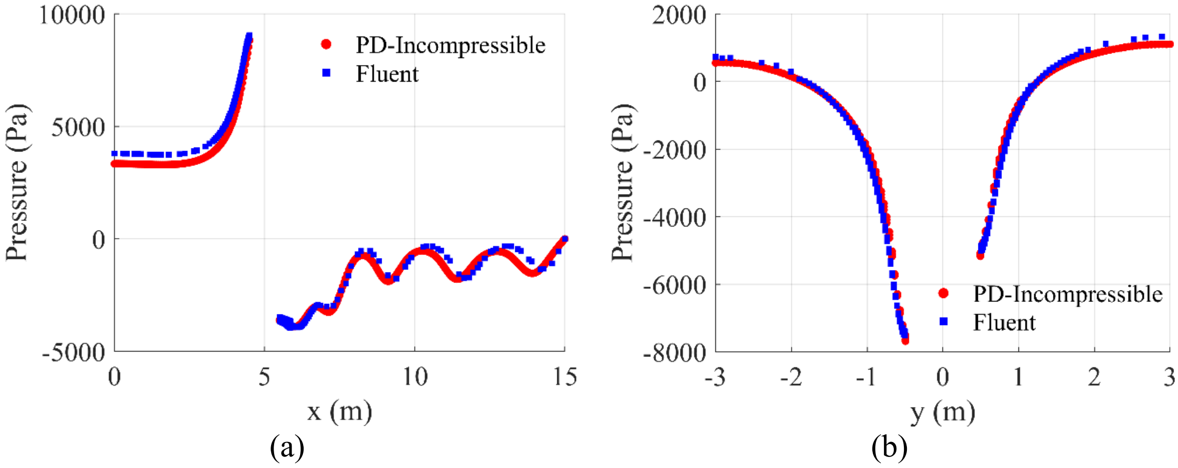


Fig. 39. Variations of pressure at (a): $y = 0$, (b): $x = 5$ m for the case of vortex tail down captured by the Eulerian incompressible PDDO model and ANSYS Fluent

For further validation, the drag and lift forces acting on the circular cylinder are calculated and compared with the results obtained in ANSYS Fluent. In 2D space, the drag and lift forces acting on the circular cylinder can be calculated as (Hosseini and Feng, 2011)

$$\mathbf{F} = \begin{bmatrix} F_D \\ F_L \end{bmatrix} = \int_S \boldsymbol{\sigma}\mathbf{n}ds \tag{72a}$$

with

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{yy} \end{bmatrix} = \mu[\nabla\otimes\mathbf{u} \ + (\nabla\otimes\mathbf{u})^{\mathrm{T}}] - p\mathbf{I} \tag{72b}$$

$$\mathbf{n} = \begin{bmatrix} n_x \\ n_y \end{bmatrix} \tag{72c}$$

where F_D and F_L represent the drag force and lift force, respectively. The parameter S represents the circle which is the cylinder boundary, $\mathbf{n}$ represents the normal vector on S, $\boldsymbol{\sigma}$ represents the stress tensor acting on S. The stress tensor $\boldsymbol{\sigma}$ given in Eq. (72b) can be rewritten as

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{yy} \end{bmatrix} = \begin{bmatrix} 2\mu u_{,x} - p & \mu(u_{,y} + v_{,x}) \\ \mu(u_{,y} + v_{,x}) & 2\mu v_{,y} - p \end{bmatrix} \tag{73}$$

Therefore, by substituting Eq. (73) and Eq. (72c) into Eq. (72a), the drag and lift forces acting on the circular cylinder can be calculated as

$$F_D = \int_S \ [(2\mu u_{,x} - p)n_x + \mu(u_{,y} + v_{,x})n_y]ds \tag{74a}$$

$$F_L = \int_S \ [\mu(u_{,y} + v_{,x})n_x + (2\mu v_{,y} - p)n_y]ds \tag{74b}$$

In the Eulerian incompressible PDDO model, to calculate the drag and lift forces using Eq. (74), all the particles located along the boundary of the circular cylinder are determined. The normal vector $\mathbf{n} = [n_x \quad n_y]^T$ associated with each particle is determined based on its relative position with the center of the cylinder. Therefore, the drag and lift forces acting on the circular cylinder given in Eq. (74) can be rewritten in the discretized form as

$$F_D = \sum_{i=1}^{m}\left[\left(2\mu u_{,x(i)} - p_{(i)}\right)n_{x(i)} + \mu\left(u_{,y(i)} + v_{,x(i)}\right)n_{y(i)}\right]\Delta s_{(i)} \tag{75a}$$

$$F_L = \sum_{i=1}^{m}\left[\mu\left(u_{,y(i)} + v_{,x(i)}\right)n_{x(i)} + \left(2\mu v_{,y(i)} - p_{(i)}\right)n_{y(i)}\right]\Delta s_{(i)} \tag{75b}$$

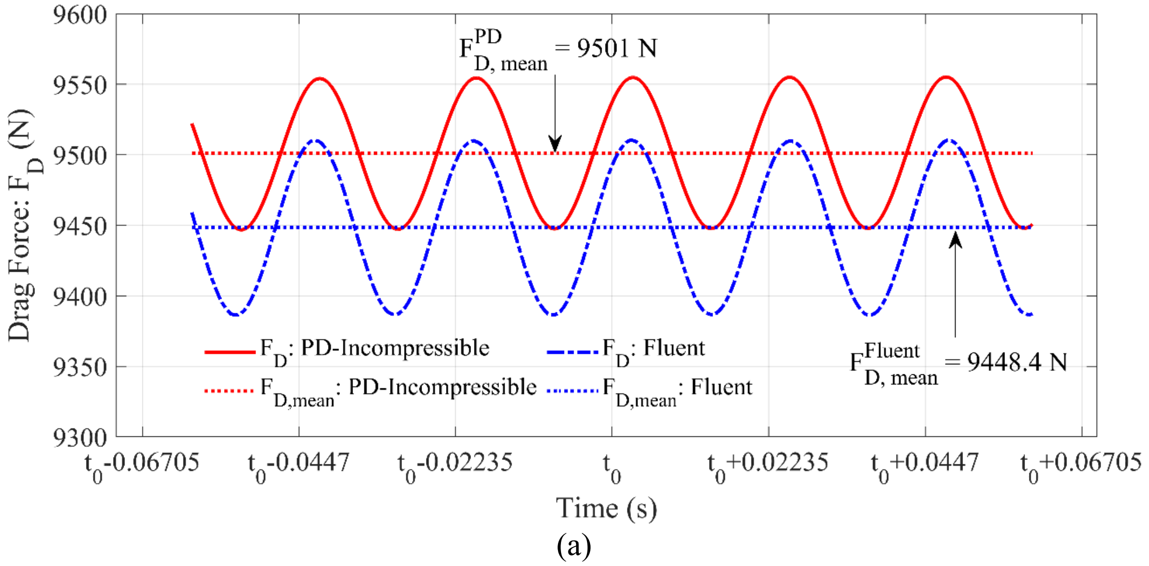
where i represents a particle located on the circle S and m represents the total number of particles located on circle S. The parameter $p_{(i)}$ represents the pressure at particle i. The parameter $\Delta s_{(i)}$ represents the length of the circular segment associated with particle i which can be calculated as

$$\Delta s_{(i)} = (D/2)\Delta\theta \tag{76}$$

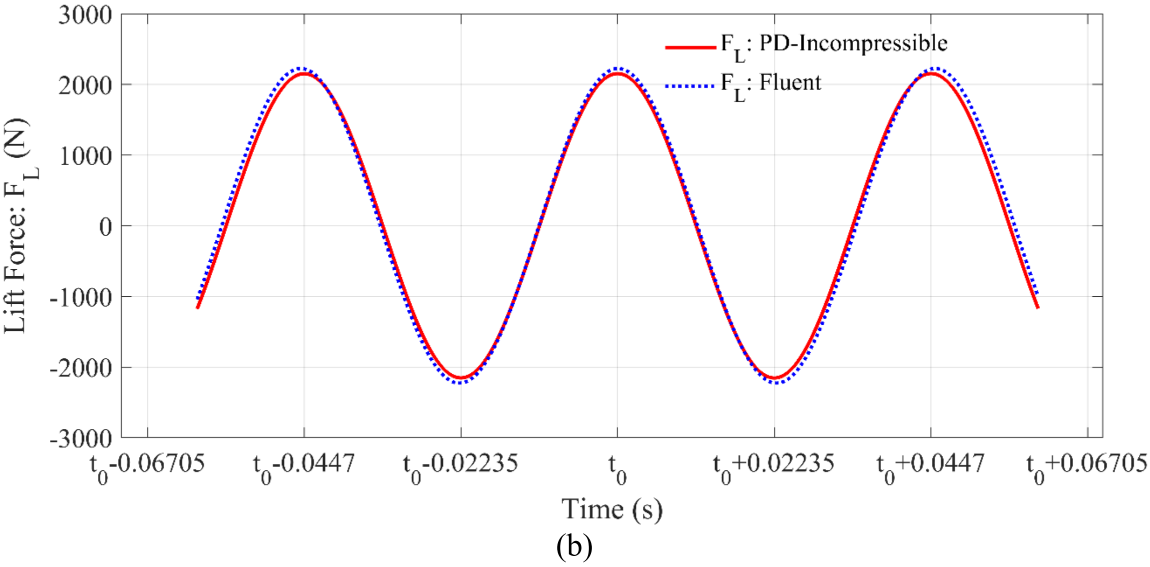
The terms $u_{,x(i)}, u_{,y(i)}, v_{,x(i)}, v_{,y(i)}$ are the derivatives of velocity components of node i with respect to x and y coordinates. These derivatives are attracted from the calculation of $\nabla\mathbf{u}_{(k)}^{(n-1)}$ given in Eq. (20).

Fig. 40 shows the variations of drag force and lift force captured by the Eulerian incompressible PDDO model and ANSYS Fluent. As shown in Fig. 40(a), the mean value of drag force captured by ANSYS Fluent is 9448.4 N. Meanwhile, the mean value of drag force captured by the Eulerian incompressible PDDO model is 9501 N, which is 0.56% different than the result obtained by ANSYS Fluent. However, due to the use of different numerical algorithms, it is observed that the simulation in ANSYS Fluent captures the fully periodic vortex shedding sooner than the simulation in the Eulerian incompressible PDDO model. In ANSYS Fluent, the SIMPLE scheme with the under-relaxation factors of 0.7 for momentum and 0.3 for pressure are chosen based on the recommendation of the software. Meanwhile, in the Eulerian incompressible PDDO model, the numerical procedure presented in Section 3.4 is used without the under-relaxation factor. As a result, the time points that the lift forces reach the maximum values and minimum values captured by ANSYS Fluent and the Eulerian incompressible PDDO model are slightly different. Therefore, to have a better comparison, the time points that the lift forces reach the peak values captured by ANSYS Fluent and the Eulerian incompressible PDDO model are shifted to the same time point t_0 as shown in Fig. 40(b).

As can be observed from Fig. 40(b), the magnitude of lift force captured by ANSYS Fluent is 2224 N. Meanwhile, the magnitude of lift force captured by the Eulerian incompressible PDDO model is 2153 N, which is 3.2% different than the value obtained by ANSYS Fluent. Moreover, as shown in Fig. 40(b), both the Eulerian incompressible PDDO model and ANSYS Fluent results give nearly the same value of 0.0447 s for the period of the vortex shedding, which is also the frequency of the lift force.



(a)



(b)

Fig. 40. Variations of (a): drag force, F_D (N), (b): lift force F_L (N) captured by the Eulerian incompressible PDDO model and ANSYS Fluent

6. Conclusion

In this study, a novel PDDO model for incompressible fluid based on the Eulerian approach is developed. The peridynamic differential operators are used to represent Navier-Stokes equations in the nonlocal forms. The numerical procedure for solving Navier-Stokes equations using the Eulerian incompressible PDDO model is presented. The pressure is obtained by solving the Poisson equation and is then used to update the velocity field to ensure that it is divergence-free. Details of implementing the pressure boundary conditions, no-slip and zero-pressure boundary conditions are also presented.

The accuracy of the developed Eulerian incompressible PDDO model is verified by considering various 2D fluid flow problems with different Reynolds numbers. The results predicted by the Eulerian incompressible PDDO model are compared with the results obtained by ANSYS Fluent with very good agreement. The capability of the developed Eulerian incompressible PDDO model is further demonstrated by capturing vortex shedding phenomena for a flow over a cylinder with a Reynolds number of $Re = 100$. The results in terms of velocity and pressure fields, drag and lift forces predicted by the Eulerian incompressible PDDO model and ANSYS Fluent show a good agreement. The developed Eulerian incompressible PDDO model can be further extended for 3D formulations to consider 3D fluid flows.

Acknowledgement

This work was supported by an Institutional Links grant, ID 527426826, under the Egypt-Newton-Mosharafa Fund partnership. The grant is funded by the UK Department for Business, Energy and Industrial Strategy and Science, Technology and Innovation Funding Authority (STIFA) - project NO. 42717 (An Integrated Smart System of Ultrafiltration, Photocatalysis, Thermal Desalination for Wastewater Treatment) and delivered by the British Council. Results were obtained using the ARCHIE-WeSt High-Performance Computer (www.archie-west.ac.uk) based at the University of Strathclyde.

References

Askari, E., Bobaru, F., Lehoucq, R., Parks, M., Silling, S., Weckner, O., 2008. Peridynamics for multiscale materials modeling, Journal of Physics: Conference Series. IOP Publishing, p. 012078.

Bazazzadeh, S., Shojaei, A., Zaccariotto, M., Galvanetto, U., 2018. Application of the peridynamic differential operator to the solution of sloshing problems in tanks. Engineering computations.

Benz, W., 1990. Smooth particle hydrodynamics: a review. The numerical modelling of nonlinear stellar pulsations, 269-288.

Bobaru, F., Ha, Y.D., 2011. Adaptive refinement and multiscale modeling in 2D peridynamics. Journal for Multiscale Computational Engineering, 635–659.

Chen, Z., Bobaru, F., 2015. Peridynamic modeling of pitting corrosion damage. Journal of the Mechanics and Physics of Solids 78, 352-381.

Chen, Z., Jafarzadeh, S., Zhao, J., Bobaru, F., 2021. A coupled mechano-chemical peridynamic model for pit-to-crack transition in stress-corrosion cracking. Journal of the Mechanics and Physics of Solids 146, 104203.

Chowdhury, S.R., Roy, P., Roy, D., Reddy, J., 2016. A peridynamic theory for linear elastic shells. International Journal of Solids and Structures 84, 110-132.

Cummins, S.J., Rudman, M., 1999. An SPH projection method. Journal of Computational Physics 152 (2), 584-607.

De Meo, D., Diyaroglu, C., Zhu, N., Oterkus, E., Siddiq, M.A., 2016a. Modelling of stress-corrosion cracking by using peridynamics. International Journal of Hydrogen Energy 41 (15), 6593-6609.

De Meo, D., Oterkus, E., 2017. Finite element implementation of a peridynamic pitting corrosion damage model. Ocean Engineering 135, 76-83.

De Meo, D., Zhu, N., Oterkus, E., 2016b. Peridynamic modeling of granular fracture in polycrystalline materials. Journal of Engineering Materials Technology 138 (4), 041008.

Diyaroglu, C., Oterkus, E., Oterkus, S., 2019. An Euler–Bernoulli beam formulation in an ordinary state-based peridynamic framework. Mathematics Mechanics of Solids 24 (2), 361-376.

Diyaroglu, C., Oterkus, E., Oterkus, S., Madenci, E., 2015. Peridynamics for bending of beams and plates with transverse shear deformation. International Journal of Solids Structures 69, 152-168.

Foster, J.T., Silling, S.A., Chen, W.W., 2010. Viscoplasticity using peridynamics. International Journal for Numerical Methods in Engineering 81 (10), 1242-1258.

Fourtakas, G., Stansby, P., Rogers, B., Lind, S., 2018. An Eulerian–Lagrangian incompressible SPH formulation (ELI-SPH) connected with a sharp interface. Computer Methods in Applied Mechanics and Engineering 329, 532-552.

Gao, Y., Oterkus, S., 2019a. Fully coupled thermomechanical analysis of laminated composites by using ordinary state based peridynamic theory. Composite Structures 207, 397-424.

Gao, Y., Oterkus, S., 2019b. Non-local modeling for fluid flow coupled with heat transfer by using peridynamic differential operator. Engineering Analysis with Boundary Elements 105, 104-121.

Gao, Y., Oterkus, S., 2019c. Ordinary state-based peridynamic modelling for fully coupled thermoelastic problems. Continuum Mechanics and Thermodynamics 31 (4), 907-937.

Gao, Y., Oterkus, S., 2020. Multi-phase fluid flow simulation by using peridynamic differential operator. Ocean Engineering 216, 108081.

Gao, Y., Oterkus, S., 2021. Coupled thermo-fluid-mechanical peridynamic model for analysing composite under fire scenarios. Composite Structures 255, 113006.
Ghajari, M., Iannucci, L., Curtis, P., 2014. A peridynamic material model for the analysis of dynamic crack propagation in orthotropic media. Computer Methods in Applied Mechanics and Engineering 276, 431-452.
Haghighat, E., Bekar, A.C., Madenci, E., Juanes, R., 2020. A nonlocal physics-informed deep learning framework using the peridynamic differential operator. arXiv preprint arXiv:2006.00446.
Hosseini, S.M., Feng, J.J., 2011. Pressure boundary conditions for computing incompressible flows with SPH. Journal of Computational Physics 230 (19), 7473-7487.
Hu, W., Ha, Y.D., Bobaru, F., 2012. Peridynamic model for dynamic fracture in unidirectional fiber-reinforced composites. Computer Methods in Applied Mechanics Engineering 217, 247-261.
Hu, Y., De Carvalho, N., Madenci, E., 2015. Peridynamic modeling of delamination growth in composite laminates. Composite Structures 132, 610-620.
Huang, Y., Oterkus, S., Hou, H., Oterkus, E., Wei, Z., Zhang, S., 2019. Peridynamic model for visco-hyperelastic material deformation in different strain rates. Continuum Mechanics Thermodynamics, 1-35.
Kružík, M., Mora-Corral, C., Stefanelli, U., 2018. Quasistatic elastoplasticity via Peridynamics: existence and localization. Continuum Mechanics and Thermodynamics 30 (5), 1155-1184.
Lee, E.-S., Moulinec, C., Xu, R., Violeau, D., Laurence, D., Stansby, P., 2008. Comparisons of weakly compressible and truly incompressible algorithms for the SPH mesh free particle method. Journal of Computational Physics 227 (18), 8417-8436.
Lind, S.J., Stansby, P., 2016. High-order Eulerian incompressible smoothed particle hydrodynamics with transition to Lagrangian free-surface motion. Journal of Computational Physics 326, 290-311.
Madenci, E., Barut, A., Dorduncu, M., 2019a. Peridynamic differential operator for numerical analysis. Springer.
Madenci, E., Barut, A., Futch, M., 2016. Peridynamic differential operator and its applications. Computer Methods in Applied Mechanics and Engineering 304, 408-451.
Madenci, E., Dorduncu, M., Barut, A., Futch, M., 2017. Numerical solution of linear and nonlinear partial differential equations using the peridynamic differential operator. Numerical Methods for Partial Differential Equations 33 (5), 1726-1753.
Madenci, E., Dorduncu, M., Gu, X., 2019b. Peridynamic least squares minimization. Computer Methods in Applied Mechanics and Engineering 348, 846-874.
Madenci, E., Oterkus, E., 2014. Peridynamic Theory and Its Applications. Springer, New York.
Madenci, E., Oterkus, S., 2016. Ordinary state-based peridynamics for plastic deformation according to von Mises yield criteria with isotropic hardening. Journal of the Mechanics Physics of Solids 86, 192-219.
Matsuno, K., Ecer, A., Periaux, J., Satofuka, N., Fox, P., 2003. A Substepping Navier-Stokes Splitting Scheme for Spectral/hp Element. Parallel Computational Fluid Dynamics 2002: New Frontiers and Multi-Disciplinary Applications, 43.
Meister, M., Burger, G., Rauch, W., 2014. On the Reynolds number sensitivity of smoothed particle hydrodynamics. Journal of hydraulic research 52 (6), 824-835.
Mitchell, J.A., 2011. A non-local, ordinary-state-based viscoelasticity model for peridynamics. Sandia National Lab Report 8064, 1-28.
Monaghan, J.J., 1992. Smoothed particle hydrodynamics. Annual review of astronomy and astrophysics 30 (1), 543-574.

Morris, J.P., 1996. A study of the stability properties of smooth particle hydrodynamics. Publications of the Astronomical Society of Australia 13, 97-102.

Nasar, A., Fourtakas, G., Lind, S., Rogers, B., Stansby, P., King, J., 2020. High-order velocity and pressure wall boundary conditions in Eulerian incompressible SPH. Journal of Computational Physics, 109793.

Nguyen, C.T., Oterkus, S., 2019a. Ordinary state-based peridynamic model for geometrically nonlinear analysis. Engineering Fracture Mechanics.

Nguyen, C.T., Oterkus, S., 2019b. Peridynamics for the thermomechanical behavior of shell structures. Engineering Fracture Mechanics, 106623.

Nguyen, C.T., Oterkus, S., 2019c. Peridynamics formulation for beam structures to predict damage in offshore structures. Ocean Engineering 173, 244-267.

Nguyen, C.T., Oterkus, S., 2020. Investigating the effect of brittle crack propagation on the strength of ship structures by using peridynamics. Ocean Engineering Accepted/In press.

Nguyen, C.T., Oterkus, S., 2021. Peridynamics for geometrically nonlinear analysis of three-dimensional beam structures. Engineering Analysis with Boundary Elements 126, 68-92.

Nguyen, C.T., Oterkus, S., Oterkus, E., 2020. An energy-based peridynamic model for fatigue cracking. Engineering Fracture Mechanics 241, 107373.

O'Grady, J., Foster, J., 2014a. Peridynamic beams: a non-ordinary, state-based model. International Journal of Solids Structures 51 (18), 3177-3183.

O'Grady, J., Foster, J., 2014b. Peridynamic plates and flat shells: A non-ordinary, state-based model. International Journal of Solids Structures 51 (25-26), 4572-4579.

Oterkus, E., 2010. Peridynamic theory for modeling three-dimensional damage growth in metallic and composite structures. The University of Arizona, The University of Arizona.

Oterkus, E., Madenci, E., Weckner, O., Silling, S., Bogert, P., Tessler, A., 2012. Combined finite element and peridynamic analyses for predicting failure in a stiffened composite curved panel with a central slot. Composite Structures 94 (3), 839-850.

Oterkus, S., 2015. Peridynamics for the solution of multiphysics problems, Aerospace and Mechanical Engineering Faculty. The University of Arizona.

Oterkus, S., Madenci, E., Oterkus, E., 2017. Fully coupled poroelastic peridynamic formulation for fluid-filled fractures. Engineering geology 225, 19-28.

Rokkam, S., Gunzburger, M., Brothers, M., Phan, N., Goel, K., 2019. A nonlocal peridynamics modeling approach for corrosion damage and crack propagation. Theoretical and Applied Fracture Mechanics 101, 373-387.

Shadloo, M.S., Zainali, A., Yildiz, M., Suleman, A., 2012. A robust weakly compressible SPH method and its comparison with an incompressible SPH. International Journal for Numerical Methods in Engineering 89 (8), 939-956.

Shakibaeinia, A., Jin, Y.-C., 2011. MPS-based mesh-free particle method for modeling open-channel flows. Journal of Hydraulic Engineering 137 (11), 1375-1384.

Shakibaeinia, A., Jin, Y.-C., 2012. MPS mesh-free particle method for multiphase flows. Computer Methods in Applied Mechanics and Engineering 229, 13-26.

Shao, S., Lo, E.Y., 2003. Incompressible SPH method for simulating Newtonian and non-Newtonian flows with a free surface. Advances in Water Resources 26 (7), 787-800.

Shirokoff, D.D.G., 2011. I. A pressure Poisson method for the incompressible Navier-Stokes equations: II. Long time behavior of the Klein-Gordon equations. Massachusetts Institute of Technology.

Shojaei, A., Galvanetto, U., Rabczuk, T., Jenabi, A., Zaccariotto, M., 2019. A generalized finite difference method based on the Peridynamic differential operator for the solution of problems in bounded and unbounded domains. Computer Methods in Applied Mechanics and Engineering 343, 100-126.

Shojaei, A., Hermann, A., Seleson, P., Cyron, C.J., 2020. Dirichlet absorbing boundary conditions for classical and peridynamic diffusion-type models. Computational Mechanics 66 (4), 773-793.
Silling, S.A., 2000. Reformulation of elasticity theory for discontinuities and long-range forces. Journal of the Mechanics and Physics of Solids 48 (1), 175-209.
Silling, S.A., Askari, A., 2014. Peridynamic model for fatigue cracking. SAND-18590. Albuquerque: Sandia National Laboratories.
Silling, S.A., Askari, E., 2005. A meshfree method based on the peridynamic model of solid mechanics. Computers & structures 83 (17-18), 1526-1535.
Silling, S.A., Epton, M., Weckner, O., Xu, J., Askari, E., 2007. Peridynamic states and constitutive modeling. Journal of Elasticity 88 (2), 151-184.
Silling, S.A., Lehoucq, R., 2010. Peridynamic theory of solid mechanics, Advances in applied mechanics. Elsevier, pp. 73-168.
Sohn, J., Heinrich, J., 1990. A Poisson equation formulation for pressure calculations in penalty finite element models for viscous incompressible flows. International Journal for Numerical Methods in Engineering 30 (2), 349-361.
Solenthaler, B., Pajarola, R., 2009. Predictive-corrective incompressible SPH, ACM SIGGRAPH 2009 papers, pp. 1-6.
Tien Nguyen, C., Oterkus, S., 2020. Ordinary state-based peridynamics for geometrically nonlinear analysis of plates. Theoretical and Applied Fracture Mechanics.
Versteeg, H.K., Malalasekera, W., 2007. An introduction to computational fluid dynamics: the finite volume method. Pearson education.
Xie, H., Koshizuka, S., Oka, Y., 2005. Simulation of drop deposition process in annular mist flow using three-dimensional particle method. Nuclear engineering and design 235 (16), 1687-1697.
Yang, Z., Oterkus, E., Nguyen, C.T., Oterkus, S., 2019. Implementation of peridynamic beam and plate formulations in finite element framework. Continuum Mechanics Thermodynamics 31 (1), 301-315.
Yildiz, M., Rook, R., Suleman, A., 2009. SPH with the multiple boundary tangent method. International Journal for Numerical Methods in Engineering 77 (10), 1416-1438.
Zhang, G., Le, Q., Loghin, A., Subramaniyan, A., Bobaru, F., 2016. Validation of a peridynamic model for fatigue cracking. Engineering Fracture Mechanics 162, 76-94.