



A MATLAB topology optimization code to control the trajectory of particle in fluid

Young Hun Choi¹ · Gil Ho Yoon¹

Received: 18 May 2022 / Revised: 27 February 2023 / Accepted: 6 March 2023 / Published online: 1 April 2023
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2023

Abstract

This paper presents an educational code for topology optimization controlling the trajectory of particle in steady-state laminar fluid. To control the trajectory of particle, fluid motion is optimized by the fluid topology optimization. The one-way forward analysis between fluid and particle and the adjoint sensitivity analysis are formulated and implemented in the framework of MATLAB. The Navier–Stokes equation is solved by the finite element method with Newton–Raphson iteration and Newton’s equation for the analysis of transient particle motion is solved by the Newmark scheme. In the present paper, the educational code is attached in the supplementary material. Throughout the code, the optimization problems considering particle trajectory can be solved. The code consists of the finite element analysis of a fluid, transient analysis of a particle suspended in fluid, and computation of the adjoint sensitivity analysis. This code can be easily expanded for complex particle fluid problems. Several benchmark problems are presented that control the velocity and position of a particle and separate multiple particles suspended in a fluid.

Keywords Topology optimization · MATLAB · Particle separation · Particle–fluid interaction

1 Introduction

This study presents an educational code for topology optimization to control the trajectory of a particle in a steady-state fluid. The trajectory of the particle, as well as the fluid motion, is controlled by designing a pseudo-rigid domain. The one-way forward analysis between the fluid and particle and the adjoint sensitivity analysis are formulated and implemented in MATLAB. The Navier–Stokes equation is solved using Newton–Raphson iteration, and Newton’s equation is solved using the Newmark scheme. To demonstrate the validity of the code, several optimizations are solved.

A topology optimization scheme was developed to determine an optimized layout without an initial priority layout.

In Bendsøe and Kikuchi (1988), topology optimization using a homogenization method was proposed. The material properties were estimated using a homogenization method for given microstructures. In Bendsøe and Sigmund (2003), a topology optimization method for compliance minimization and heat conduction was proposed. Unlike density-based topology optimization, topology optimization has also been attempted using explicit curves or surfaces. For example, level-set-based topology optimization methods have been developed and the recent development of level-set methods were reviewed in van Dijk et al. (2013). A recent review about the education code can be found in Wang et al. (2021). In addition to the structural topology optimization scheme, many relevant studies have also been conducted for fluids. A recent review of the fluid topology optimization can be found in Alexandersen and Andreasen (2020). In Borrvall and Petersson (2003), topology optimization for Stokes flow was developed. The pseudo-rigid domain was modelled by modifying the fluid governing equation, i.e., the Navier–Stokes equation. The Darcy force has been used for many years as a pseudo-rigid domain in fluid topology optimization, e.g., fluid topology optimization for steady laminar viscous flow in Gersborg-Hansen et al. (2005), and

Responsible Editor: Casper Schousboe Andreasen

Topical Collection: Flow-driven Multiphysics
Guest Editors: J Alexandersen, C S Andreasen, K Giannakoglou,
K Maute, K Yaji

✉ Gil Ho Yoon
ghy@hanyang.ac.kr

¹ School of Mechanical Engineering, Hanyang University,
Seoul, South Korea

more recently, unsteady laminar viscous flows in Deng et al. (2011).

Unlike the structural topology optimization problem, which primarily utilizes the finite element procedure, adopting different analysis procedures is also viable. For example, fluid analysis can be performed using the lattice Boltzmann method for fluid topology optimization (Makhija et al. 2012). It is impossible to review all relevant research in fluid topology optimization; a review related to fluid-related problems can be found in Alexandersen and Andreassen (2020). Multi-physical systems with fluids are also important. Conjugated heat and fluid problems were investigated in Yoon (2010). A fluid–structure interaction system was also considered in Yoon (2012); Andreassen and Sigmund (2013); Yoon (2014); Picelli et al. (2017). Turbulent flow was considered by Papoutsis-Kiachagias et al. (2011); Papoutsis-Kiachagias and Giannakoglou (2016); Yoon (2016); Dilgen et al. (2018). Recently, topology optimization of particles suspended in a fluid has also been considered. Control of particles in fluids has been studied over the years. In Lee and Balachandar (2010); Hood et al. (2016), the force exerted on a particle by a fluid was investigated. A numerical method for the simulation of particle–fluid interaction can be found in Hu et al. (1992); Hashemi et al. (2016); Wu and Yang (2018). The application of a microfluid device to manipulate particles can be found in An et al. (2021); Dinler and Okumus (2018); Madadelahi et al. (2020); Prohm et al. (2013). Topology optimization of a structure in a fluid for particle control has recently been studied in Yoon (2020); Yoon and So (2021); Andreassen (2020); Yoon (2022). These relevant studies show the importance of fluid and particle problems.

Many studies with introductory codes have been published to help disseminate ideas and help researchers in the field of structural optimization. Educational papers that contain MATLAB codes for topology optimization have been proposed. A topology optimization MATLAB code for compliance minimization and heat conduction was implemented in Sigmund (2001); Bendsøe and Sigmund (2003).

In this study, a MATLAB code for the control of particles suspended in a steady-state laminar flow is implemented. The MATLAB code consists of three parts in the main program, `ParticleInFluid2D.m`. First, the generation of the

geometrical model, boundary condition, and material properties are contained in `Makemodel.m`; then, initialization of the optimization parameters and allocation of additional parameters for each example contained in `optExample.m` are implemented in `ParticleInFluid2D.m`. After the assignment of the initial variables, a finite element analysis of the fluid and particle motions are conducted in `Steady_Navier_Stokes_2D.m` and `Particle_2D.m`, respectively. The numerical method for the interaction between the particles and fluid used in this study is shown in Fig. 1. With these analyses, it is possible to conduct a fluid topology optimization. After the analysis, an optimization process is implemented. The objective, constraints, and their sensitivities are computed in `ObjectiveFunction.m`, `Constraint.m`, and `Sensitivity.m`, respectively. The remainder of the MATLAB code includes visualization of the optimized layout and storage of the resulting data.

The remainder of this article is arranged as follows: Sect. 2 provides an overview of the topology optimization formulation to control the trajectory of particles suspended in a fluid. The Navier–Stokes equation and Newton’s equations are formulated and presented. Details of the implementation are explained in Sect. 3. In Sect. 4, several examples are solved to demonstrate the validity of the present code. Finally, the conclusions and findings are summarized in Sect. 5.

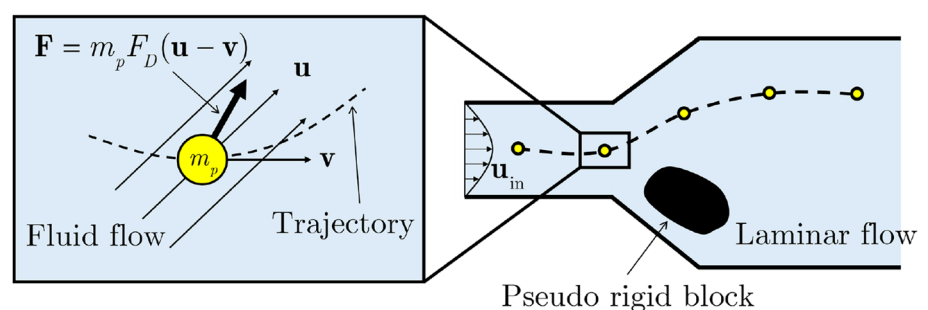
2 Topology optimization for particle motion in steady-state fluid

Before presenting the code, this section summarizes the theories of the topology optimization framework for the coupled analysis of fluid and particle.

2.1 Governing equations: Navier–Stokes equation and Newton’s equation

The coupled analysis of particles suspended in laminar flow requires consideration of hydrodynamic interactions. By assuming sufficiently small particles, the mutual coupling

Fig. 1 Interaction between particles and fluid in topology optimization of particles in laminar flow



effect of particles in fluid motion can be neglected. For the computation of an incompressible laminar fluid with Darcy's force $\alpha \mathbf{u}$, the Navier–Stokes equation for the fluid domain is modified as follows:

$$\begin{aligned} \rho(\mathbf{u} \cdot \nabla) \mathbf{u} &= \nabla \cdot [-p \mathbf{I} + \mu(\nabla \mathbf{u} + \nabla \mathbf{u}^T)] - \alpha \mathbf{u} & \text{on } \Omega \\ \nabla \cdot (\rho \mathbf{u}) &= 0 & \text{on } \Omega \end{aligned} \quad (1)$$

$$\alpha = \alpha_{\max} \gamma^n \quad (2)$$

Compared with the original Navier–Stokes equation, the Darcy force $\alpha \mathbf{u}$ is also included. The maximum inverse permeability is α_{\max} , and the coefficient is interpolated with respect to the design variables γ with penalization n . Although many relevant studies utilize the RAMP interpolation function, however, this study intentionally employs the SIMP interpolation function to show that it is also possible to use another interpolation function by changing the physical interpretation of the design variable and the constraint. The following boundary conditions are considered.

$$\begin{aligned} \text{No-slip b.c.: } \mathbf{u} &= \mathbf{0} \text{ on } \Gamma_{u^0} \\ \text{Inflow/outflow b.c.: } \mathbf{u} &= \mathbf{u}^* \text{ on } \Gamma_{u^*} \\ \text{Zero normal stress b.c.:} & \\ [-p \mathbf{I} + \mu(\nabla \mathbf{u} + \nabla \mathbf{u}^T)] \cdot \mathbf{n} &= \mathbf{0} \text{ on } \Gamma_{\sigma} \end{aligned} \quad (3)$$

The above Navier–Stokes equation is defined as the control volume Ω . The velocity and pressure are denoted as \mathbf{u} and p , respectively. The fluid density and dynamic viscosity are denoted as ρ and μ , respectively. The wall (no-slip) boundary condition and inflow/outflow boundary conditions of \mathbf{u}^* for the prescribed fluid velocities are defined along Γ_{u^0} and Γ_{u^*} , respectively. The Neumann boundary condition (zero normal stress boundary condition) is defined along Γ_{σ} with $\mathbf{0}$ as the zero vector. Figure 2 is presented to help the clear understanding of the above conditions.

Without loss of generality, the above partial differential equation can be discretized in the framework of the finite element procedure. The domain Ω is discretized by the

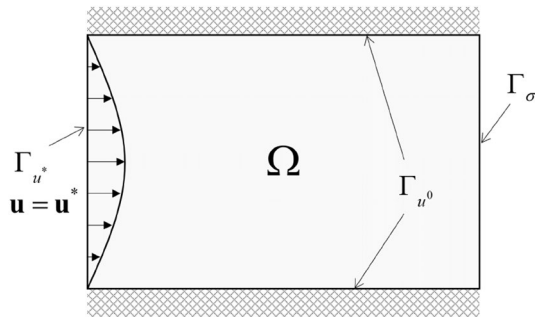


Fig. 2 A design domain and boundary conditions

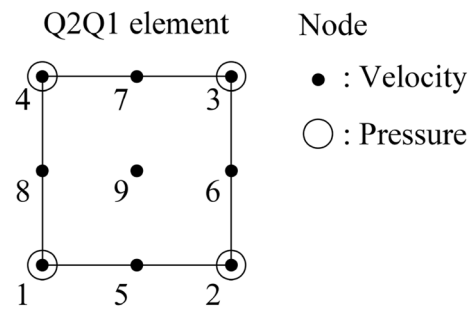


Fig. 3 Q2Q1 element and node number

Q2Q1 elements with 9 velocity nodes and 4 pressure nodes as shown in Fig. 3. The Q2Q1 element is one of the elements used in the mixed finite element discretization and satisfies the Ladyzhenskaya–Babuška–Brezzi condition (Donea and Huerta 2003). By following the notation in Donea and Huerta (2003), the following nonlinear equations can be obtained:

$$\begin{bmatrix} \mathbf{K} + \mathbf{C}(\mathbf{U}) & \mathbf{G} \\ \mathbf{G}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ \mathbf{P} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{h} \end{bmatrix}, \quad (4)$$

where \mathbf{K} , \mathbf{C} , and \mathbf{G} are the viscosity matrix, convection matrix, and discrete gradient operators, respectively. The velocity and pressure vectors are denoted by \mathbf{U} and \mathbf{P} , respectively. The body force and boundary conditions (Neumann and Dirichlet boundary conditions) are formulated as \mathbf{f} and \mathbf{h} , respectively. By considering the Darcy force term, an additional matrix \mathbf{A} is added as follows:

$$\begin{bmatrix} \mathbf{K} + \mathbf{C}(\mathbf{U}) + \mathbf{A} & \mathbf{G} \\ \mathbf{G}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ \mathbf{P} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{h} \end{bmatrix}, \quad (5)$$

where the Newton–Raphson method can be used to solve the above equations. The Newton–Raphson method can be written as follows:

$$\mathbf{r}_{(i)} = \frac{\partial \mathbf{r}}{\partial \mathbf{s}} \bigg|_{(i)} (\mathbf{s}_{(i+1)} - \mathbf{s}_{(i)}), \quad (6)$$

where the residuals of the nonlinear equations and the variable vector of the equations \mathbf{U} , \mathbf{P} are denoted by \mathbf{r}_i and \mathbf{s}_i at the i th iteration, respectively. To improve the numerical efficiency, any advanced implementation with another element or stabilization approaches (the SUPG (Streamline Upwind Petrov–Galerkin) and PSPG (Pressure-Stabilizing Petrov–Galerkin)) can be incorporated. However, to illustrate the concept, the laminar flow is assumed. With the Q2Q1 element and the laminar flow, we do not experience severe oscillations or nonconvergence at least for the present examples.

Newton's law, which ignores the rotational motion of a particle, can be numerically formulated as follows:

$$\frac{d}{dt}(m_p \mathbf{v}) = m_p F_D(\mathbf{u} - \mathbf{v}), \quad \mathbf{v} = \frac{d\mathbf{X}}{dt} = \dot{\mathbf{X}}, \quad (7)$$

where the coefficient F_D is set as in Walsh (1976),

$$F_D = \frac{18\mu}{\rho_p d_p^2} \quad (8)$$

where the mass, diameter, density, and velocity of the particles are denoted by m_p , d_p , ρ_p , and \mathbf{v} , respectively. The difference between the velocity of the particle and the velocity of the fluid determines the drag force, whose direction is opposite to the relative motion of the particle. The absolute coordinates of the particles are denoted by \mathbf{X} . The coefficient F_D of the drag force can be formulated with the equation (8). The trajectory of a particle with the larger drag force comparably larger than the inertial force becomes similar to the streamline. Thus, it may be possible to consider the topology optimization scheme with the streamline control. However, the trajectories of particles with different masses become different to the streamline.

For the numerical simulation of the particle, the Newmark scheme is employed as follows:

$$\begin{aligned} \mathbf{X}_{i+1} &= \mathbf{X}_i + (\Delta t) \dot{\mathbf{X}}_i + (\Delta t)^2 \left[\left(\frac{1}{2} - \beta \right) \ddot{\mathbf{X}}_i + \beta \ddot{\mathbf{X}}_{i+1} \right] \\ \dot{\mathbf{X}}_{i+1} &= \dot{\mathbf{X}}_i + (\Delta t) \left[(1 - \gamma) \ddot{\mathbf{X}}_i + \gamma \ddot{\mathbf{X}}_{i+1} \right] \\ \mathbf{M} \ddot{\mathbf{X}}_{i+1} &= F_D \mathbf{M} (\mathbf{u}_i - \dot{\mathbf{X}}_{i+1}) \end{aligned} \quad (9)$$

where \mathbf{M} is $m_p \mathbf{I}$ and the velocity and acceleration of the particle are denoted at the i th iteration by $\dot{\mathbf{X}}_i$ and $\ddot{\mathbf{X}}_i$, respectively. To simulate the trajectory of a particle, the fluid velocity is computed using finite elements. In the first iteration of the Newmark scheme, the initial fluid velocity \mathbf{u}_0 at the initial position of the particle \mathbf{X}_0 is interpolated by finite element. Then, the difference between the initial particle velocity $\dot{\mathbf{X}}_0$ and fluid velocity \mathbf{u}_0 is computed and multiplying it by F_D gives $\ddot{\mathbf{X}}_0$. By applying the Newmark scheme (9), the trajectory of the particle can be obtained.

2.2 Transient sensitivity analysis of particle motion

To allow topology optimization considering the motion of the particle resulting from the drag force, design variables γ are assigned to each fluid finite element. The general objective function is defined as follows:

$$\text{Objective function: } \int_0^{t_f} c(\dot{\mathbf{X}}) dt \quad (10)$$

where the simulation time is t_f and the function of the velocity of the particle is denoted by $c(\dot{\mathbf{X}})$. For example, the following functions can be considered for the objective value or constraint:

$$\int_0^{t_f} \|\mathbf{v}\|^2 dt, \quad \mathbf{v} = \dot{\mathbf{X}} \quad (11)$$

$$\mathbf{p}_0 + \int_0^{t_f} \mathbf{v} dt, \quad \mathbf{v} = \dot{\mathbf{X}} \quad (12)$$

where the initial position of the particle is denoted as \mathbf{p}_0 . Although not described here, the viscosity energy dissipation, $\int_{\Omega} \left(\frac{\mu}{2} \|\nabla \mathbf{u} + \nabla \mathbf{u}^T\|^2 + \alpha \|\mathbf{u}\|^2 \right) d\Omega$, is also implemented as an objective function.

For the sensitivity analyses of the above objective functions, two Lagrange multipliers, i.e., λ for Newton's law and ψ for Navier–Stokes equation, are introduced for the Lagrangian of the objective function in (10) and (13).

$$\begin{aligned} L &= \int_0^{t_f} c(\dot{\mathbf{X}}) dt + \int_0^{t_f} \lambda^T (\mathbf{M} \ddot{\mathbf{X}} - \mathbf{F}(\dot{\mathbf{X}}, \mathbf{u})) dt \\ &\quad + \int_0^{t_f} \psi^T \mathbf{R} dt, \quad (13) \\ &\text{with } \mathbf{F}(\dot{\mathbf{X}}, \mathbf{u}) = \mathbf{M} F_D(\mathbf{u} - \dot{\mathbf{X}}) \end{aligned}$$

where the mass matrix and force vector of the particle, primarily for the drag force, are denoted by \mathbf{M} and \mathbf{F} , respectively.

Newton's equation is a transient analysis, and the second term of the equation in (13) is integrated twice over the time interval. Note that the third term in (13) for the residual equation \mathbf{R} for the steady-state condition is also formulated as an integration form. The derivative of the Lagrange function with respect to the design variable γ_e can be summarized as follows:

$$\begin{aligned} \frac{\partial L}{\partial \gamma_e} &= \left[\left(\left(\frac{\partial c}{\partial \dot{\mathbf{X}}} \right) - \lambda^T \mathbf{M} \right) \frac{\partial \mathbf{X}}{\partial \gamma_e} + \lambda^T \left(\mathbf{M} \frac{\partial \dot{\mathbf{X}}}{\partial \gamma_e} - \frac{\partial \mathbf{F}}{\partial \dot{\mathbf{X}}} \frac{\partial \mathbf{X}}{\partial \gamma_e} \right) \right]_0^{t_f} \\ &\quad + \int_0^{t_f} \left(-\frac{d}{dt} \left(\frac{\partial c}{\partial \dot{\mathbf{X}}} \right) + \ddot{\mathbf{X}}^T \mathbf{M} + \lambda^T \frac{\partial \mathbf{F}}{\partial \dot{\mathbf{X}}} \right) \frac{\partial \mathbf{X}}{\partial \gamma_e} dt \\ &\quad + \int_0^{t_f} -\lambda^T \frac{\partial \mathbf{F}}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \gamma_e} dt + \int_0^{t_f} \psi^T \left(\frac{\partial \mathbf{R}}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \gamma_e} + \frac{\partial \mathbf{R}}{\partial \mathbf{P}} \frac{\partial \mathbf{P}}{\partial \gamma_e} \right) dt \\ &\quad + \int_0^{t_f} \psi^T \frac{\partial \mathbf{R}}{\partial \gamma_e} dt \end{aligned} \quad (14)$$

Here, it is an essential step to obtain the fluid velocities \mathbf{u} of the particles as follows:

$$\mathbf{u} = \mathbf{N}^T \mathbf{U}, \quad (15)$$

where the shape function is denoted as \mathbf{N} . The drag force is then computed accordingly.

$$\mathbf{F} = F_D \mathbf{M}(\mathbf{N}^T \mathbf{U} - \mathbf{v}) \quad (16)$$

Using the Jacobian matrix of the Navier–Stokes equation, $\mathbf{J} = \left[\frac{\partial \mathbf{R}}{\partial \mathbf{U}}, \frac{\partial \mathbf{R}}{\partial \mathbf{P}} \right]$, the differentiation of the Lagrangian function can be expressed as follows:

$$\begin{aligned} \frac{\partial L}{\partial \gamma_e} &= \int_0^{t_f} \left(-\frac{d}{dt} \left(\frac{\partial c}{\partial \dot{\mathbf{x}}} \right) + \mathbf{M} \left(\dot{\lambda}^T - F_D \dot{\lambda}^T - F_D \lambda^T \frac{\partial \mathbf{u}}{\partial \mathbf{x}} \right) \right) \frac{\partial \mathbf{X}}{\partial \gamma_e} dt \\ &+ \int_0^{t_f} \left(-F_D \lambda^T \mathbf{M} \left[\mathbf{N}^T \mathbf{0} \right] + \psi^T \mathbf{J} \right) \frac{\partial}{\partial \gamma_e} \begin{bmatrix} \mathbf{U} \\ \mathbf{P} \end{bmatrix} dt \\ &+ \int_0^{t_f} \psi^T \frac{\partial \mathbf{R}}{\partial \gamma_e} dt \end{aligned} \quad (17)$$

Using the adjoint sensitivity analysis method, the two adjoint systems can be formulated as follows:

Adjoint system 1:

$$\begin{aligned} \ddot{\lambda} - F_D \dot{\lambda} - F_D \left(\frac{\partial \mathbf{u}}{\partial \mathbf{x}} \right)^T \lambda &= \mathbf{M}^{-1} \frac{d}{dt} \left(\frac{\partial c}{\partial \dot{\mathbf{x}}} \right) \\ \text{with } \lambda &= \mathbf{0} \text{ and } \dot{\lambda} = \mathbf{M}^{-1} \left(\frac{\partial c}{\partial \dot{\mathbf{x}}} \right) \text{ at } t = t_f \end{aligned} \quad (18)$$

$$\text{Adjoint system 2: } \psi = F_D (\mathbf{J}^T)^{-1} \mathbf{M} \begin{bmatrix} \mathbf{N} \\ \mathbf{0} \end{bmatrix} \lambda \quad (19)$$

With the reverse time integration scheme, the above adjoint equations can be solved. After solving the adjoint sensitivity equation in (18) for λ , it is substituted into the adjoint sensitivity equation in (19) for ψ .

In the simulation, calculation of the fluid velocity \mathbf{u} at the current particle position is intricate. Assuming the same shape

elements, efficient calculations of the natural coordinates are performed as follows:

$$\begin{aligned} \xi &= (x_{\text{current}} - \text{floor}(\frac{x_{\text{current}}}{\Delta x}) \times \Delta x) \times \frac{2}{\Delta x} - 1 \\ \eta &= (y_{\text{current}} - \text{floor}(\frac{y_{\text{current}}}{\Delta y}) \times \Delta y) \times \frac{2}{\Delta y} - 1 \end{aligned} \quad (20)$$

Assuming the same geometries for the employed finite elements, the natural coordinates ξ and η are computed as shown in Fig. 4. This implies that computation becomes complicated with a general shape mesh. The current position of the particle is denoted by $(x_{\text{current}}, y_{\text{current}})$, with Δx and Δy for the element sizes in the x and y directions, respectively.

3 Implementation

This section presents an introductory implementation and explanations in connection with the theory of fluid topology optimization in a two-dimensional domain in the framework of MATLAB without an optimizer code. The main code for topology optimization is presented in `ParticleInFluid2D.m` with auxiliary codes `MakeModel.m`, `Steady_Navier_Stokes_2D.m`, `Particle_2D.m`, and `Sensitivity.m` for the definition and generation of the model, laminar flow solver, particle solver, and adjoint sensitivity analyses, respectively, as shown in Fig. 5. Without loss of generality, the default objective function is set to the integration of the norm of particle velocity. Table 1 summarizes the input parameters, and Table 2 summarizes the variables and their meanings generated by `MakeModel.m`.

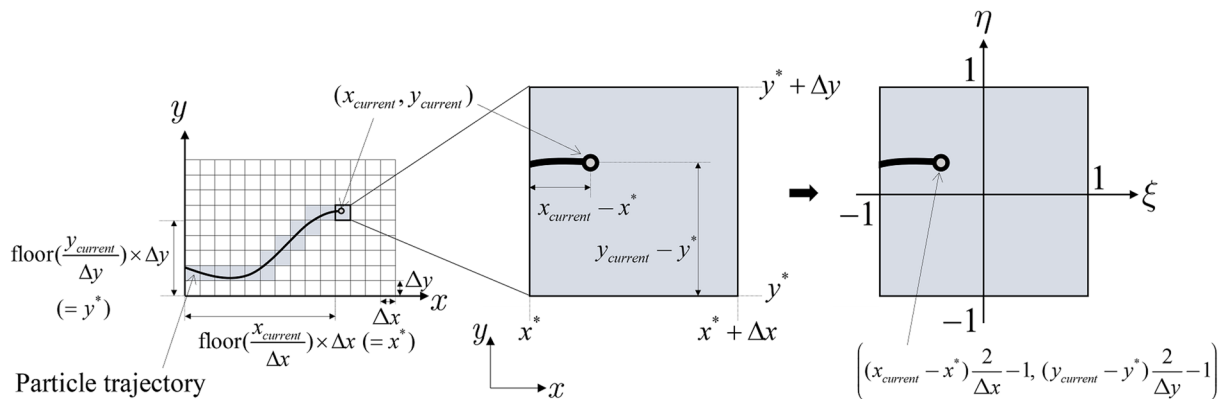


Fig. 4 Natural coordinates at the current particle position in the same rectangular finite elements

Table 1 List of the input parameters for `ParticleInFluid2D.m`

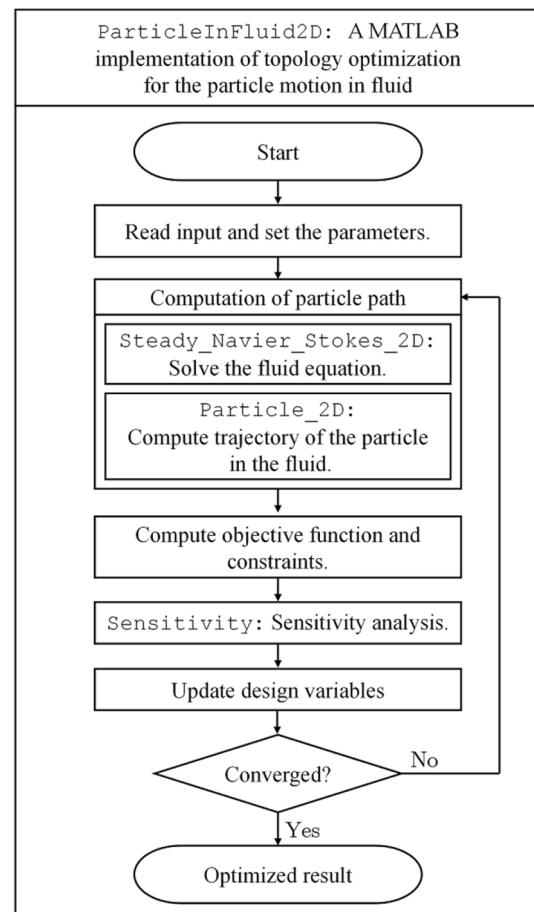
Variables	Description
<code>nelx</code>	Number of elements in x-direction
<code>nely</code>	Number of elements in y-direction
<code>l_x</code>	x length of design domain
<code>l_y</code>	y length of design domain
<code>u_max</code>	Maximum of parabolic velocity of inlet
<code>t_f</code>	Integration time
<code>dt</code>	Time step of Newmark method
<code>volfrac</code>	Volume fraction of volume constraint
<code>initial_x</code>	Initial design variable value
<code>rmin</code>	Radius of the filter
<code>penal</code>	SIMP penalization power
<code>EX_N</code>	Example number
<code>opcase</code>	Case number of the example

Table 2 Variables and list of input data generated by `MakeModel.m`

Variables	Description
<code>X_p</code>	Coordinates of the nodes of the Q4 elements
<code>X_v</code>	Coordinates of the nodes of the Q9 elements
<code>IX_p</code>	List of nodes of each of the Q4 elements (Pressure)
<code>IX_v</code>	List of nodes of each of the Q9 elements (Velocity)
<code>edofList_v</code>	List of dof of each of the Q9 elements
<code>ndof_p</code>	Number of dof of the pressure
<code>ndof_v</code>	Number of dof of the velocity
<code>body_force</code>	Body force applied to fluid
<code>bc_p</code>	Pressure boundary condition [node, dof, value]
<code>bc_v</code>	Velocity boundary condition [node, dof, value]
<code>mprop</code>	Material properties of the fluid ($\mu/\rho, \rho, \alpha_{\max}$)
<code>particle_F_D</code>	Drag coefficients of particles (F_D)
<code>particle_X0</code>	Initial positions of particles
<code>particle_V0</code>	Initial velocities of particles

3.1 ParticleInFluid2D: Topology optimization for particle motion in fluid

File `ParticleInFluid2D.m` is the main script for topology optimization. In lines 25–27, the finite element model of interest and problem definitions (boundary conditions, material properties, and the initial position and velocity of the particle) are generated. In lines 28–52,

**Fig. 5** Flowchart of the topology optimization for particle motion in fluid

some important parameters are defined and computed. The matrices and information regarding the fluid topology optimization are computed in lines 53 to 56. The optimization process is implemented in lines 57 to 108. Each optimization process requires fluid analysis (lines 65–66 with `Steady_Navier_Stokes_2D.m`) and particle analysis (lines 67–68 with `Particle_2D.m`), computations of the objective function, and sensitivity analysis (lines 69–70 with `ObjectiveFunction.m` and lines 73–74 with `Sensitivity.m`, lines 71–72 with `Constraint.m`). As mentioned, the various optimization formulations can be defined. Plotting of the optimized layout and updating of the design variables are implemented in lines 75–108.

The main code, `ParticleInFluid2D.m`, is read as follows:


```

25 function ParticleInFluid2D(nelx,
    nely, l_x, l_y, u_max, t_f, dt,
    volfrac, initial_x, rmin,
    penal, EX_N, opcase)
26 %% Read input data
27 [X_p, X_v, IX_p, IX_v, edofList_v,
    ndof_p, ndof_v, body_force,
    bc_p, bc_v, mprop, particle_F_D,
    particle_X0, particle_V0] =
    MakeModel(nelx, nely, [l_x, l_y],
    u_max, EX_N);

```

Basic information is created using `MakeModel.m`. The necessary environmental parameters are saved as a struct array as follows:

```

28 %% Newmark, time
29 newmark = [1/6, 1/3];
30 t = [0, t_f, dt, 0];
31 t(4) = round((t(2)-t(1))/t(3))
    + 1;
32 %% Optimization parameters
33 opt = struct;
34 opt.Idx = (1:nelx*nely)';
35 opt.n = size(opt.Idx,1);
36 xval = zeros(size(IX_p
    ,1),1);
37 xval(opt.Idx) = initial_x * ones
    (opt.n,1);
38 opt.min_density = 0;
39 opt.volfrac = volfrac;
40 opt.interpScheme = {'SIMP', penal};
41 opt.movelimit = 0.05;
42 opt.low = zeros(opt.n, 1);
43 opt.upp = ones(opt.n, 1);
44 opt.xold1 = xval;
45 opt.xold2 = xval;
46 opt.rmin = rmin * (l_x /
    nelx);
47 opt.H = CalDist(X_p,
    IX_p, opt.rmin);
48 opt.convergence = 1e-4;
49 opt = OptExample(opt,
    EX_N, opcase, [l_x, l_y]);
50 %% Result folder
51 result_folder = [sprintf('EX%d_%
    d_result', EX_N, opcase) char(
    datetime('now','Format','
    _yyyyMMdd','_','HH_mm_ss'))];
52 mkdir(result_folder);

```

In line 34, the number of the elements to be used as design variables are saved in `opt.Idx`. And then, in lines 36–37, the topological density, `xval`, is generated. Because the `xval` consists of the design variables and non-design variables, the minimum density 0 is allocated first in line 36 and initial density is set to the design variables in line 37.

For every optimization iteration, fluid analysis, particle analysis, and calculation of the objective function and sensitivity are carried out using `Steady_Navier_Stokes_2D.m`, `Particle_2D.m`, `ObjectiveFunction.m`, and `Sensitivity.m`. In the present study, the method of moving asymptotes is used as the default optimizer (Svanberg 1987).

3.2 Finite element analysis of the Navier–Stokes equation

Code `Steady_Navier_Stokes_2D.m` solves the Navier–Stokes equation using the Newton–Raphson method. The Newton–Raphson equation (6) can be rewritten as follows:

$$\begin{bmatrix} \mathbf{U}_{(i+1)} \\ \mathbf{P}_{(i+1)} \end{bmatrix} = \begin{bmatrix} \mathbf{U}_{(i)} \\ \mathbf{P}_{(i)} \end{bmatrix} - \mathbf{J}_{(i)}^{-1} \mathbf{R}_{(i)}, \quad (21)$$

where the subscripts indicate the iteration number of the Newton–Raphson method. The i th residual and inverse of the Jacobian are denoted by $\mathbf{R}_{(i)}$ and $\mathbf{J}_{(i)}^{-1}$, respectively. The pseudo-rigid domain is modelled with the design variables `xval`, with the `A0` matrix (see `InitialMatrix.m`).

3.2.1 InitialMatrix

Utilizing the information in `example.mat` generated by `MakeModel.m`, the basic matrices for the fluid simulations are computed with `InitialMatrix.m`. The essential matrices, i.e., $\mathbf{K}, \mathbf{G}, \mathbf{f}, \mathbf{h}, \mathbf{A}$, and zero matrices, defined in (4) and (5), are computed by the `InitialMatrix.m` sub-routine as follows:

```

53 %% FE matrices for fluid (K, G, f,
    h, A, ...)
54 [gauss, matrix_K, matrix_G,
    matrix_G_T, matrix_f, matrix_h,
    matrix_A0, matrix_Sparse,
    matrix_A_mu] = InitialMatrix(
    X_v, IX_v, IX_p, edofList_v,
    ndof_v, ndof_p, bc_v, bc_p,
    mprop, body_force);
55 U = zeros(ndof_v,1);
56 P = zeros(ndof_p,1);

```

The above matrices are formulated using a standard finite element formula. The pressure and velocity are interpolated with four nodes and nine nodes as shown in Fig. 3, respectively.

$$\mathbf{N}_p = \begin{bmatrix} n_{p1} \\ n_{p2} \\ n_{p3} \\ n_{p4} \end{bmatrix} = \begin{bmatrix} (1-\xi)(1-\eta)/4 \\ (1+\xi)(1-\eta)/4 \\ (1+\xi)(1+\eta)/4 \\ (1-\xi)(1+\eta)/4 \end{bmatrix},$$

$$\mathbf{N}_v = \begin{bmatrix} n_{v1} \\ n_{v2} \\ n_{v3} \\ n_{v4} \\ n_{v5} \\ n_{v6} \\ n_{v7} \\ n_{v8} \\ n_{v9} \end{bmatrix} = \begin{bmatrix} (1-\xi)(1-\eta)\xi\eta/4 \\ -(1+\xi)(1-\eta)\xi\eta/4 \\ (1+\xi)(1+\eta)\xi\eta/4 \\ -(1-\xi)(1+\eta)\xi\eta/4 \\ -(1-\eta)\eta(1-\xi^2)/2 \\ (1+\xi)\xi(1-\eta^2)/2 \\ (1+\eta)\eta(1-\xi^2)/2 \\ -(1-\xi)\xi(1-\eta^2)/2 \\ (1-\xi^2)(1-\eta^2) \end{bmatrix}, \quad (22)$$

where the shape function vectors for the pressure and velocities are denoted by \mathbf{N}_p and \mathbf{N}_v , where ξ and η are the natural coordinates. The subscripts of the shape functions indicate the corresponding nodes for the pressure and velocity. With the above interpolation function, the following matrices are defined for the finite element formulation.

$$\mathbf{N} = [\mathbf{N}_1 \ \mathbf{N}_2 \ \cdots \ \mathbf{N}_9]^T$$

$$\mathbf{N}_i = \begin{bmatrix} n_{vi} & 0 \\ 0 & n_{vi} \end{bmatrix}$$

$$\mathbf{B}_1 = [\mathbf{B}_{11} \ \mathbf{B}_{12} \ \cdots \ \mathbf{B}_{19}]^T$$

$$\mathbf{B}_2 = [\mathbf{B}_{21} \ \mathbf{B}_{22} \ \cdots \ \mathbf{B}_{29}]^T \quad (23)$$

$$\mathbf{B}_{1i} = \begin{bmatrix} \frac{\partial n_{vi}}{\partial x} & \frac{\partial n_{vi}}{\partial y} & 0 & 0 \\ 0 & 0 & \frac{\partial n_{vi}}{\partial x} & \frac{\partial n_{vi}}{\partial y} \end{bmatrix}^T$$

$$\mathbf{B}_{2i} = \begin{bmatrix} \frac{\partial n_{vi}}{\partial x} & \frac{\partial n_{vi}}{\partial y} \end{bmatrix}$$

The element matrices of the e th element at Ω_e are defined as follows:

$$\mathbf{K}^e = \frac{\mu}{\rho} \iint_{\Omega_e} \mathbf{B}_1 \mathbf{B}_1^T |J| d\xi d\eta$$

$$\mathbf{G}^e = - \iint_{\Omega_e} \mathbf{B}_2 \mathbf{N}_p^T |J| d\xi d\eta, \quad (24)$$

$$\mathbf{A}^e = \frac{\alpha}{\rho} \iint_{\Omega_e} \mathbf{N} \mathbf{N}^T |J| d\xi d\eta$$

where the determinant of the e th element is $|J|$. For the implementation, the governing equation (1) is divided by the fluid density.

3.2.2 matrix_C_J

Unlike constant matrices, matrices $\mathbf{A}(\boldsymbol{\gamma})$, $\mathbf{C}(\mathbf{U})$, and $\mathbf{J}(\mathbf{U}, \boldsymbol{\gamma})$, are subject to change with every iteration. The pseudo-rigid body is modelled in $\mathbf{A}(\boldsymbol{\gamma})$, which is a function of the

design variable, i.e., $\sum_e (\mathbf{A} \mathbf{0}_e \gamma_e^{penal})$. The convection matrix and Jacobian matrix are formulated using the following subroutine:

```
17 function [C, J] = matrix_C_J(
    edofList_v, U, gauss, ndof_v,
    ndof_p)
```

where matrices are generated as follows:

$$\mathbf{C}^e = \iint_{\Omega_e} \mathbf{N} \begin{bmatrix} u_x & u_y & 0 & 0 \\ 0 & 0 & u_x & u_y \end{bmatrix} \mathbf{B}_1^T |J| d\xi d\eta \quad (25)$$

Then, the Jacobian matrix is computed as follows:

$$\mathbf{J} = \begin{bmatrix} \mathbf{A} + \mathbf{K} + \mathbf{C} + \begin{bmatrix} \frac{\partial \mathbf{C}}{\partial u_1} \mathbf{U} & \cdots & \frac{\partial \mathbf{C}}{\partial u_n} \mathbf{U} \end{bmatrix} & \mathbf{G} \\ \mathbf{G}^T & \mathbf{0} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial \mathbf{C}}{\partial u_1} \mathbf{U} & \cdots & \frac{\partial \mathbf{C}}{\partial u_n} \mathbf{U} \end{bmatrix} = \mathbf{A}_e \begin{bmatrix} \frac{\partial \mathbf{C}^e}{\partial u_1^e} \mathbf{U}^e & \cdots & \frac{\partial \mathbf{C}^e}{\partial u_{18}^e} \mathbf{U}^e \end{bmatrix} \quad (26)$$

$$\frac{\partial \mathbf{C}^e}{\partial u_{xi}^e} \mathbf{U}^e = \left(\iint_{\Omega_e} \mathbf{N} \begin{bmatrix} n_{vi} & 0 & 0 & 0 \\ 0 & 0 & n_{vi} & 0 \end{bmatrix} \mathbf{B}_1^T dx dy \right) \mathbf{U}^e$$

$$\frac{\partial \mathbf{C}^e}{\partial u_{yi}^e} \mathbf{U}^e = \left(\iint_{\Omega_e} \mathbf{N} \begin{bmatrix} 0 & n_{vi} & 0 & 0 \\ 0 & 0 & 0 & n_{vi} \end{bmatrix} \mathbf{B}_1^T dx dy \right) \mathbf{U}^e$$

3.3 Particle motion in fluid

After solving the Navier–Stokes equation, the acceleration, velocity, and displacement of the particle are computed using the following function:

```
29 function [particle_X, particle_V,
    particle_A, path_Element,
    particle_ShapeFunction,
    particle_gradU] = Particle_2D(
    X_v, IX_v, edofList_v, U,
    particle_X0, particle_V0,
    particle_F_D, newmark, t,
    l_x_l_y, nelx, nely, EX_N)
```

In addition to the particle information, the fluid velocity and gradient velocity ($\nabla \mathbf{u}$) are also computed. In the current implementation, the same rectangular elements are assumed. One unique function used here is to determine the natural coordinates inside an element. In the present study, `CurrentPositionO`, where O denotes the number of examples. For example, the first example calls `CurrentPosition1`. The computations of the particle position and the corresponding element containing particle depend on the geometry of the design domain. Because the geometries of the examples are different from each other, each computation code should be implemented separately.


```

112 function [csi, eta, element] =
    CurrentPosition1(x, y, dx, dy,
        nelx)
113 nx = floor(x / dx);
114 ny = floor(y / dy);
115 element = ny * nelx + nx + 1;
116 csi = (x-nx*dx)/dx*2-1;
117 eta = (y-ny*dy)/dy*2-1;
118 end

```

3.4 Sensitivity analysis of the particle trajectory

Without loss of generality, the integration of the square of the particle velocities, $\int_0^{t_f} (v_x^2 + v_y^2) dt$, is set to the objective function for the sensitivity analysis. Sensitivity analysis is performed using `Sensitivity.m`. As the objective function varies with respect to the examples, two codes, i.e., `SensitivityVelocitySquare.m` and `SensitivityPosition.m`, are implemented. For example, the sensitivity of the square of the velocity is computed as

```

31 function dfdx =
    SensitivityVelocitySquare(
        matrix_A0, matrix_J, particle_V
        , particle_A, path_Element,
        particle_ShapeFunction,
        particle_gradU, particle_F_D,
        mprop, U, edofList_v, ndof_v,
        ndof_p, bc_v, newmark, t, xval,
        interpScheme, nelx, nely)

```

First, the differentiation of the objective function with respect to the velocities is computed.

$$c = v_x^2 + v_y^2, \quad \frac{\partial c}{\partial \mathbf{v}} = 2 \begin{bmatrix} v_x \\ v_y \end{bmatrix}, \quad \frac{d}{dt} \frac{\partial c}{\partial \mathbf{v}} = 2 \begin{bmatrix} a_x \\ a_y \end{bmatrix} \quad (27)$$

Then, the two adjoint systems are solved with the time-reversal integration scheme. The first Lagrange multiplier, λ , is solved using (18), which is the time-reversal integration scheme ($\Delta t \rightarrow -\Delta t, i+1 \rightarrow i-1$).

$$\begin{aligned}
 \dot{\lambda}^{(i-1)} &= \dot{\lambda}^{(i)} - (\Delta t)(1 - \gamma)\ddot{\lambda}^{(i)} - \gamma(\Delta t)\ddot{\lambda}^{(i-1)} \\
 \lambda^{(i-1)} &= \lambda^{(i)} - (\Delta t)\dot{\lambda}^{(i)} + (\Delta t)^2 \left(\left(\frac{1}{2} - \beta \right) \ddot{\lambda}^{(i)} + \beta \ddot{\lambda}^{(i-1)} \right) \\
 \ddot{\lambda}^{(i-1)} &= F_D \dot{\lambda}^{(i-1)} + F_D (\nabla \mathbf{u})^{(i-1)} \lambda^{(i-1)} + \frac{1}{m_p} \left(\frac{d}{dt} \left(\frac{\partial c}{\partial \mathbf{v}} \right) \right)^{(i-1)}
 \end{aligned} \quad (28)$$

The above equations are implemented in lines 43–60 in `SensitivityVelocitySquare.m`. The second adjoint system is formulated as follows:

$$\int_0^{t_f} \psi dt = (m_p F_D) (\mathbf{J}^T)^{-1} \left(\int_0^{t_f} \mathbf{N}^T \lambda dt \right) \quad (29)$$

The second adjoint system can be computed in lines 61–68 in `SensitivityVelocitySquare.m`. The final sensitivity can be obtained as follows:

$$\begin{aligned}
 \frac{\partial L}{\partial \gamma_e} &= \int_0^{t_f} \psi^T \begin{bmatrix} \frac{\partial}{\partial \gamma_e} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ \mathbf{P} \end{bmatrix} dt \\
 \frac{\partial}{\partial \gamma_e} \mathbf{A} &= \alpha_{\max} p \gamma_e^{p-1} \mathbf{A}^e
 \end{aligned} \quad (30)$$

The above implementation is implemented in lines 69–80 in `SensitivityVelocitySquare.m`.

4 Numerical examples

To illustrate the present implementation of the fluid topology optimization for the particle–fluid interaction system, this section presents some numerical examples. The method of moving asymptotes is considered (Svanberg 1987).

4.1 Sensitivity analysis

For the first example, the sensitivity analysis of the objective function $\int_0^{t_f} \|\mathbf{v}\|^2 dt$ is shown in Fig. 6. The parabolic fluid input is considered along the left side of the design domain (0.05 m by 0.03 m). With the distribution of the design variables shown in Fig. 6, sensitivity evaluations for the 10 random design variables are performed. To investigate the effect of the numerical integration time Δt and the perturbation of the design variable $\Delta \gamma$, two different values are considered in Fig. 6. As illustrated, an accurate and efficient sensitivity analysis can be obtained. The estimated computing time for the 10 variables using the Finite Difference Method (FDM) is 299.2 s; the average FDM time of 10 variables is 29.92 s and the computed FDM time for all the design variables will be $29.92 \times 10201 = 3.052 \times 10^5$ s when the adjoint sensitivity analysis only takes 14.5 s, which is about one twenty thousandth. This example shows that the accurate sensitivity is obtained efficiently using the present adjoint sensitivity analysis. These results can be replicated using the present code. After the sensitivity is obtained by the adjoint method, the filtering scheme is applied to the sensitivity. In this study, the sensitivity filtering and the p -norm density filtering with the S-shape function in Yoon and Yi (2019) are used. The sensitivity filtering is used in example 1 and 2 and the p -norm density filtering with the S-shape function (Yoon and Kim 2003) is used in example 3.

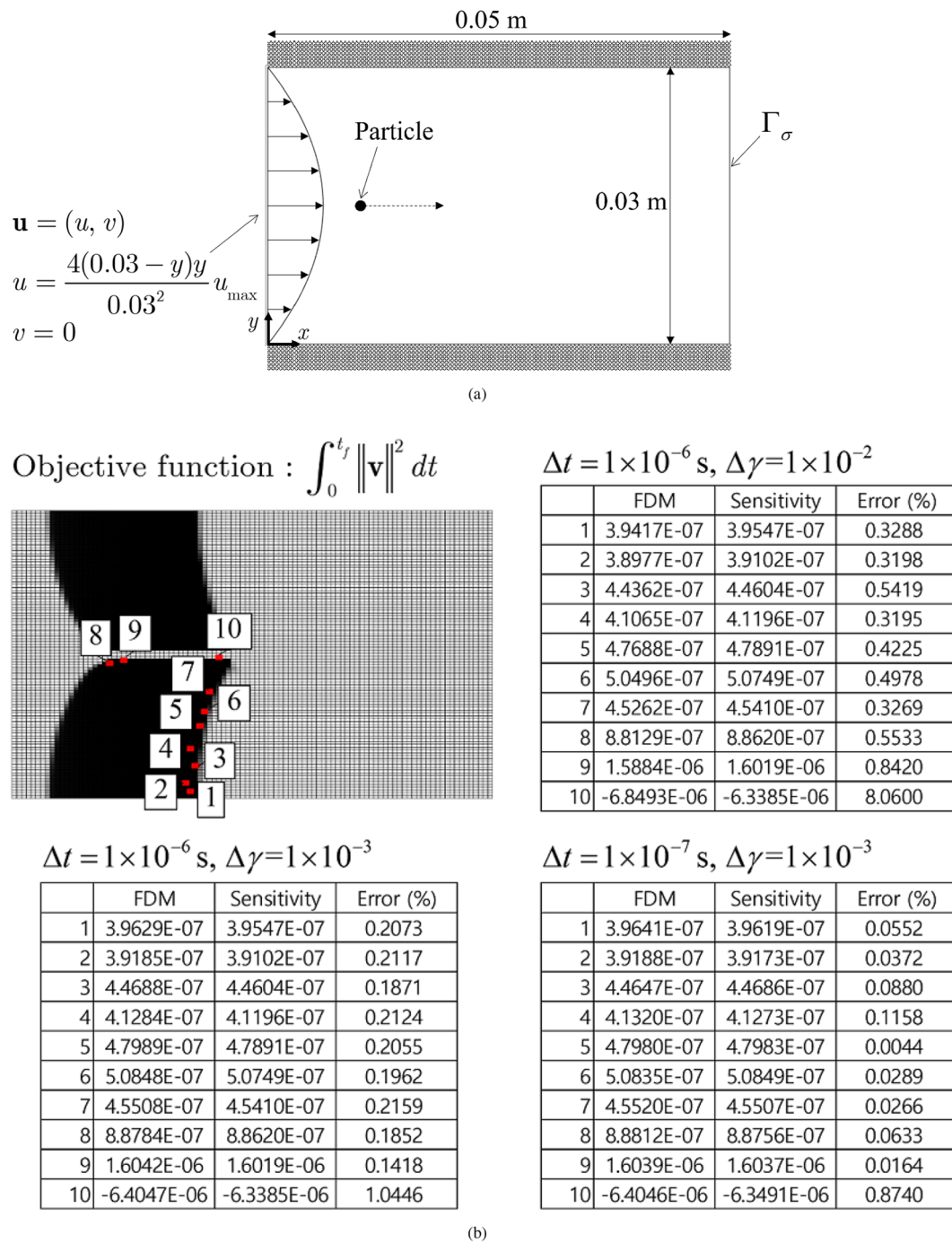


Fig. 6 a Problem definition ($\alpha_{\max} = 10^7$, $\text{mass}_0 = 30\%$ of the design domain, Input velocity: $u_{\max} = 2.5 \text{ mm/s}$, Fluid: density = 1000 kg/m^3 , viscosity = $5 \times 10^{-3} \text{ Pa} \cdot \text{s}$, Particle: density = 1900 kg/m^3 ,

diameter = $4.78 \times 10^{-5} \text{ m}$), **b** comparison of errors between the finite difference method (FDM) and sensitivity ($t_f = 0.2 \text{ s}$)

4.2 Example 1: maximizing particle velocity

For the first optimization example, the following optimization formulation in (31) is considered:

$$\begin{aligned} & \text{Maximize } \int_0^{t_f} \|\mathbf{v}\|^2 dt \\ & \text{Subject to } \text{mass} \leq \text{mass}_0 \\ & \quad \gamma_{\min} \leq \gamma \leq 1 \end{aligned} \quad (31)$$

where the mass limit is denoted by mass_0 (30 % in the examples). The parabolic input is assumed to be on the left side, and the pressure output condition is imposed along the right side. The objective of this example is to maximize the integration of the velocity of a particle whose initial location is set to (0.01 m, 0.015 m). The domain is discretized as 101×101 , and $\gamma_{\min} = 0.0001$. The results with the different t_f in Fig. 7 can be obtained by running `ParticleInFluid2D(101, 101, 0.05, 0.03, 0.0025, 0.2, 1e-6, 0.3, 0.1, 1.5, 4, 1, 1)` in Fig. 7a and

Fig. 7 Example 1. Topology optimization to maximize the particle velocity for the design in Fig. 6a. **a** An optimized layout with $t_f = 0.2$ s with 200,000 time steps, **b** an optimized layout with $t_f = 0.5$ s with 500,000 time steps, and (c and d) fluid velocity with $t_f = 0.2$ s and $t_f = 0.5$ s

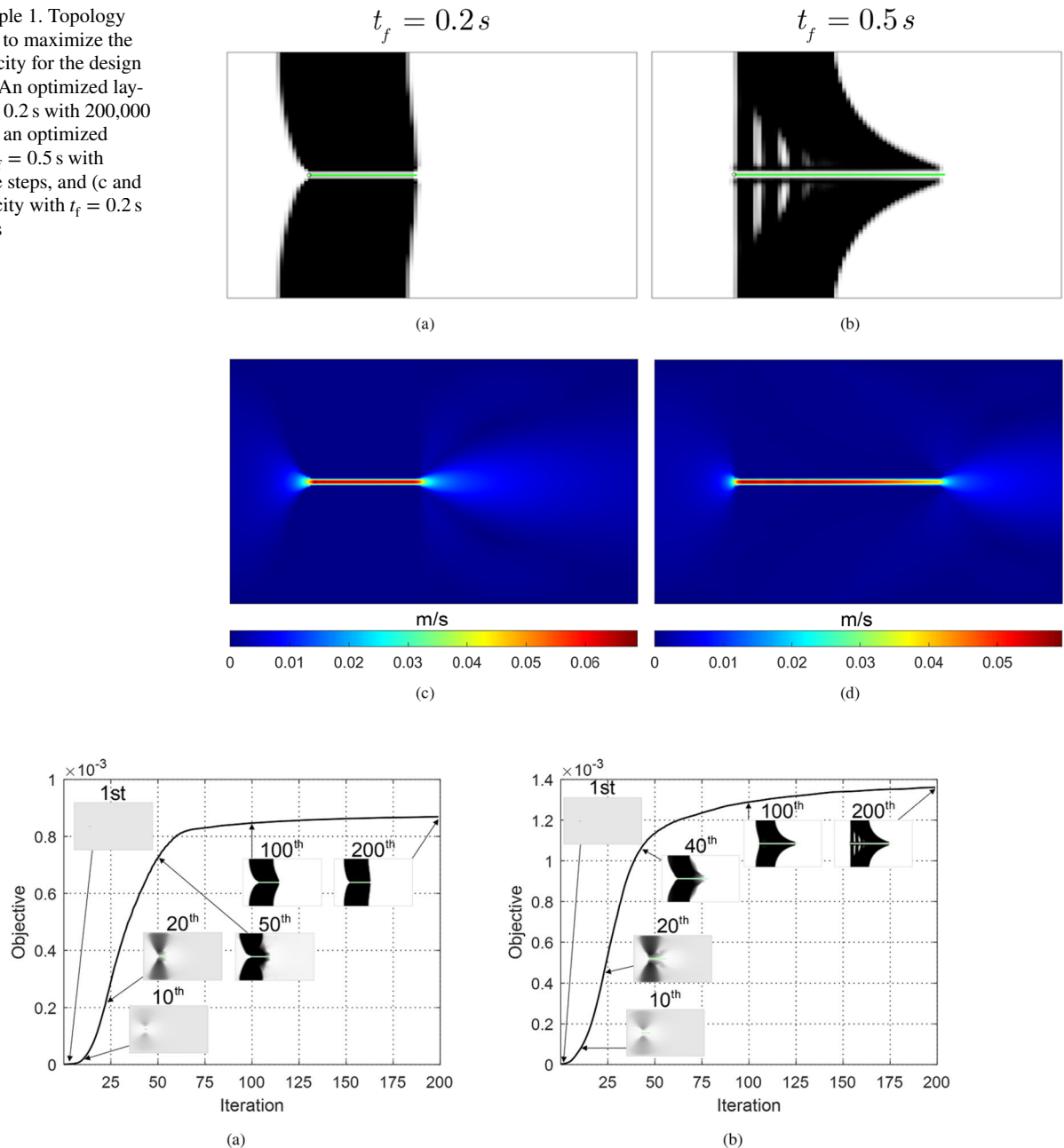


Fig. 8 **a** The optimization history with $t_f = 0.2$ s and **b** the optimization history with $t_f = 0.5$ s

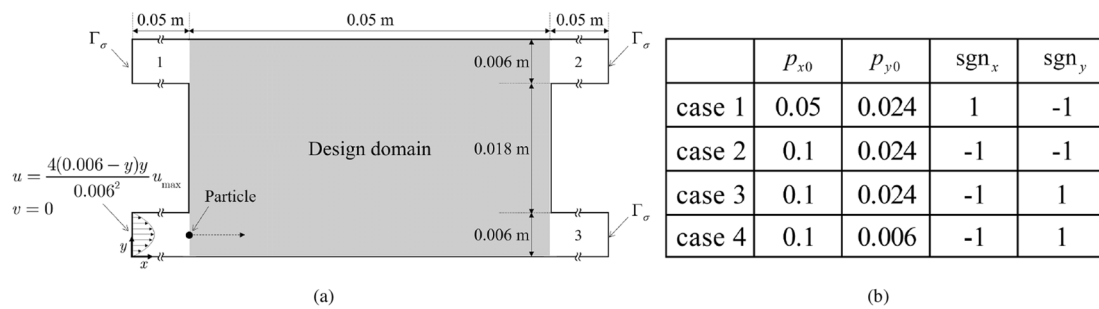


Fig. 9 Example 2. Topology optimization to control the position of the particle. **a** Problem definition ($\alpha_{\max} = 10^7$, $n = 4$, $\text{mass}_0 = 80\%$ of the design domain, Input velocity: $u_{\max} = 2.5$ mm/s, Fluid:

$\text{density} = 1000$ kg/m³, viscosity = 5×10^{-3} Pa s, Particle: $F_D = 500$ N s/kg m), **b** table of the position constraints

ParticleInFluid2D(101, 101, 0.05, 0.03, 0.0025, 0.5, 1e-6, 0.3, 0.1, 1.5, 4, 1, 2) for Fig. 7b, which solves the following optimization example in (31): The optimization histories are shown in Fig. 8. Using the present code, narrow channels are obtained by maximizing the integration of the norm of the velocity. Because the particle velocity increases with the increment of the fluid velocity around the particle, the following channel designs increasing the fluid velocity are obtained. Different channels are obtained depending on the simulation time t_f .

4.3 Example 2: particle position

In the next example, the particle position problem shown in Fig. 9a is considered. The outlet channels are set long enough for the particle to do not go outside of the channels. If the particle goes outside of the channels, the particle trajectory cannot be computed. The objective is to minimize the fluid dissipation energy subject to the constraints for the particle position and the mass constraint as the following optimization formulation.

$$\begin{aligned} & \text{Minimize} \quad \int_{\Omega} \left(\frac{\mu}{2} \|\nabla \mathbf{u} + \nabla \mathbf{u}^T\|^2 + \alpha \|\mathbf{u}\|^2 \right) d\Omega \\ & \text{Subject to} \quad \text{mass} \geq \text{mass}_0 \\ & \quad \text{sgn}_x \cdot p_x \leq \text{sgn}_x \cdot p_{x0} \\ & \quad \text{sgn}_y \cdot p_y \leq \text{sgn}_y \cdot p_{y0} \\ & \quad \gamma_{\min} \leq \gamma \leq 1 \end{aligned} \quad (32)$$

where sgn_x and sgn_y are the values to change the sign of inequality, i.e., -1 or 1 , p_x and p_y indicate the particle position, and p_{x0} , p_{y0} are constants, as shown in Fig. 9b. The reason for setting the fluid dissipation energy to the objective function is to allow a smooth fluidic channel. The fluid inlet is defined at the lower left part, and three fluid outlets are considered. After releasing a particle in front of the fluid inlet, it travels to each outlet. Constraints are imposed such that the particle moves away from the inlet through each outlet. In cases 1 and 2, the particle moves out through the outlet on the upper left side and upper right side. In cases 3 and 4, the topologically optimized layouts for the particle to pass through the lower right outlet with different constraints are designed. Fig. 10a–d show the optimized layouts for each case. As illustrated, the present optimization algorithm can successfully determine the optimized

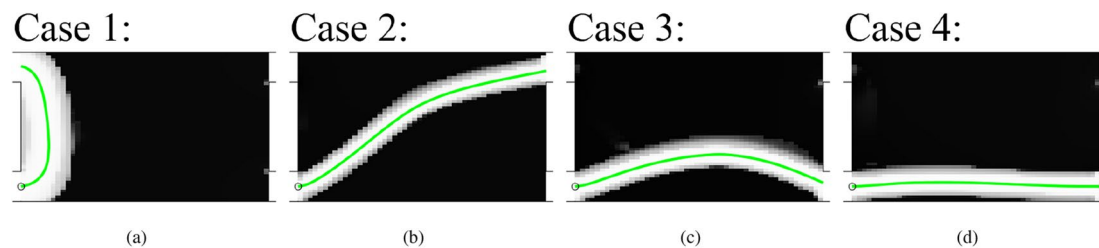


Fig. 10 The optimized layouts of the example 2 (The optimization results can be obtained by ParticleInFluid2D(50, 50, 0.05, 0.03, 0.01, 4, 1e-5, 0.8, 0.1, 1.5, 4, 2, opcase) where opcase = 1, 2, 3 and 4 for **a**, **b**, **c** and **d**, respectively)

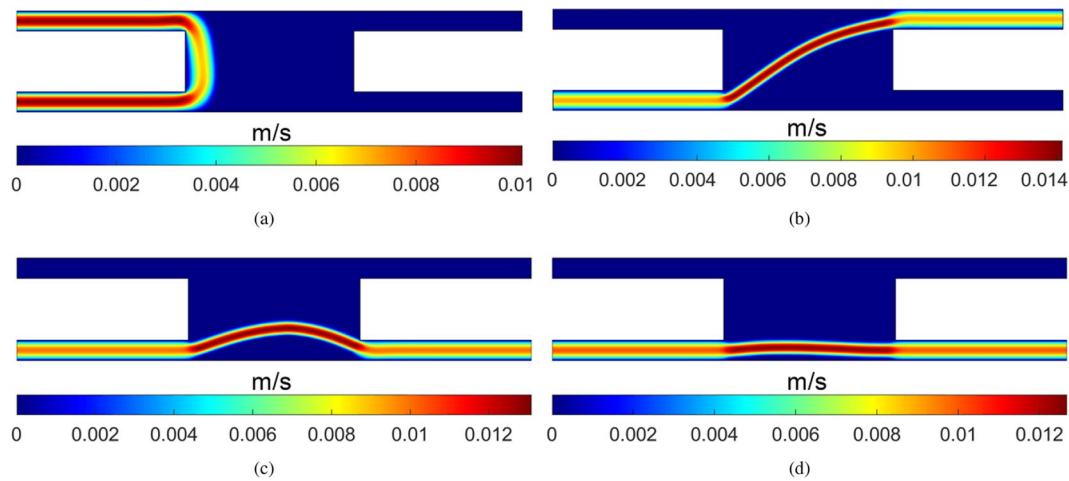


Fig. 11 The fluid velocities of the designs in Fig. 10

layouts. Some gray elements appear and to remove the gray element issue, some common approaches such as a higher penalization, additional constraints or filtering scheme can be employed. In addition, it is observed that the gray element tends to be disappear by increasing the number of particles (Yoon and So 2021). Figure 11a–d shows the fluid velocities for each case. Note that curved outlets are obtained in particular for (b) and (c) because of the inertia effects of the particle and fluid velocity. The convergence histories are shown in Fig. 12. To obtain the present results, the `ParticleInFluid2D.m` program also can be run, i.e., `ParticleInFluid2D(50, 50, 0.05, 0.03, 0.01, 4, 1e-5, 0.7, 0.1, 1.5, 3, 2, 1)` in Fig. 10a, `ParticleInFluid2D(50, 50, 0.05, 0.03, 0.01, 4, 1e-5, 0.7, 0.1, 1.5, 3, 2, 2)` for Fig. 10b, `ParticleInFluid2D(50, 50, 0.05, 0.03, 0.01, 4, 1e-5, 0.7, 0.1, 1.5, 3, 2, 3)` for Fig. 10c, and `ParticleInFluid2D(50, 50, 0.05, 0.03, 0.01, 4, 1e-5, 0.7, 0.1, 1.5, 3, 2, 4)` for Fig. 10d. The following optimization formulation is solved for the results shown in Fig. 10.

4.4 Example 3: Separation of particles

For the final example, the separation of three particles with different drag forces is considered in Fig. 13a. Starting with an empty channel, the three particles enter the center channel, as shown in Fig. 13b. Thus, the objective of this problem is to determine an optimized layout in the design domain to move the three particles to the three channels on the right-hand side. To achieve this objective, the present study adopts the following optimization formulation:

$$\begin{aligned} & \text{Minimize} \quad \int_{\Omega} \left(\frac{\mu}{2} \|\nabla \mathbf{u} + \nabla \mathbf{u}^T\|^2 + \alpha \|\mathbf{u}\|^2 \right) d\Omega \\ & \text{Subject to} \quad \text{mass}_1 \leq \text{mass} \leq \text{mass}_2 \\ & \quad \text{position constraints} \\ & \quad \gamma_{\min} \leq \gamma \leq 1 \end{aligned} \quad (33)$$

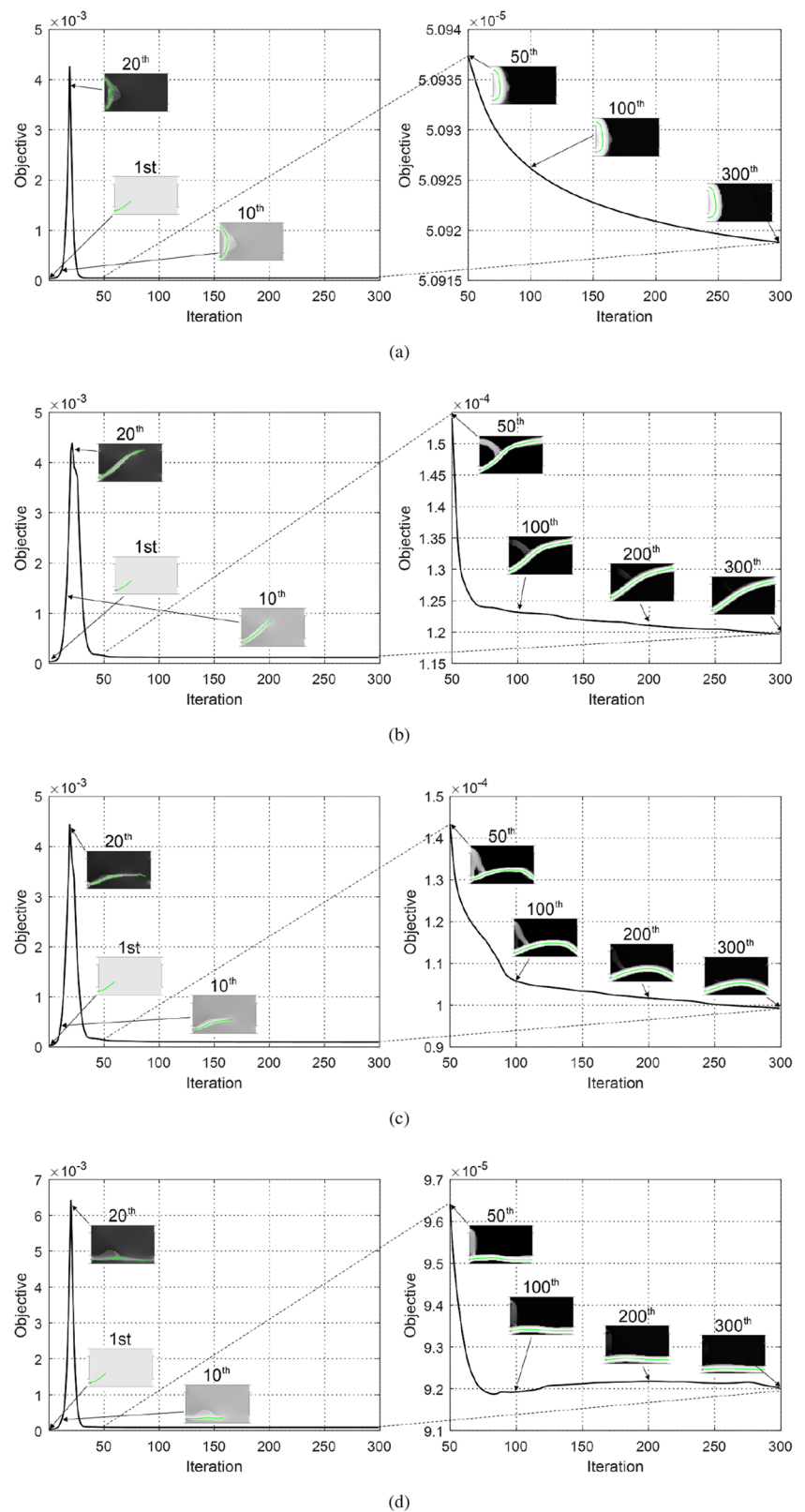
where the objective function is the fluid dissipation energy, and additional constraints are imposed to separate the particles depending on the material property. To converge the mass to mass_0 with many constraints and the objective, the mass is constrained by lower bound $\text{mass}_1 = 0.99 \times \text{mass}_0$, and the upper bound $\text{mass}_2 = 1.01 \times \text{mass}_0$. It is worth noticing that the mass constraint is employed to improve the optimization process. In other words, more mass does not guarantee a better design with a lower objective in the particle and fluid optimization problem unlike the compliance minimization problem (Fig. 14). In this example, the mass_0 is 50% of the design domain and the following constraints are imposed on position:

Position constraints:

$$\begin{aligned} 0.065 \text{ m} & \leq p_{x1} \\ 0.024 \text{ m} & \leq p_{y1} \\ 0.065 \text{ m} & \leq p_{x2} \\ 0.012 \text{ m} & \leq p_{y2} \leq 0.018 \text{ m} \\ 0.065 \text{ m} & \leq p_{x3} \\ p_{y3} & \leq 0.006 \text{ m} \end{aligned} \quad (34)$$

where (p_{xi}, p_{yi}) denotes the position of the i th particle. In the present example, the drag coefficients of the three particles are arbitrarily set to $F_D = 200, F_D = 40, F_D = 20$. In this example, the density filter with the p -norm (35) and S-shaped sigmoid function (36) are simultaneously applied.

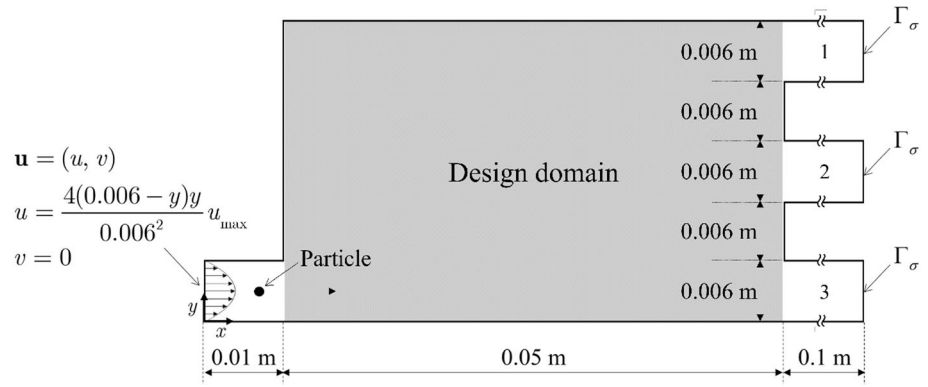
Fig. 12 The optimization histories of the example 2



The p -norm density filter is utilized to set the design variable to the maximum value of design variables among the neighbor elements. This density filter and S-shaped function

was used together in Yoon and Yi (2019); Yoon and Kim (2003). With the above formulations, the optimization result and its fluid velocity are shown in Fig. 15a and b. It should

Fig. 13 Example 3. Topology optimization to control the positions of three particles. Problem definition ($\alpha_{\max} = 10^7$, $n = 4$, mass_0 is 50% of the design domain, $t_f = 8$ s, Input velocity: $u_{\max} = 0.02$ m/s, Fluid: density = 1000 kg/m^3 , viscosity = $3 \times 10^{-3} \text{ Pa} \cdot \text{s}$, Particle 1: $F_D = 200 \text{ N s/kg m}$, Particle 2: $F_D = 40 \text{ N s/kg m}$, Particle 3: $F_D = 20 \text{ N s/kg m}$, Position constraints: $0.065 \text{ m} \leq p_{x1}$, $0.024 \text{ m} \leq p_{y1}$, $0.065 \text{ m} \leq p_{x2}$, $0.012 \text{ m} \leq p_{y2} \leq 0.018 \text{ m}$, $0.065 \text{ m} \leq p_{x3}$, $p_{y3} \leq 0.006 \text{ m}$)



be noted that the trajectories rendered by different colors depend on the drag force. The trajectories shown in Fig. 15a show that the three particles are successfully separated. The green ($F_D = 200$), red ($F_D = 40$), and blue ($F_D = 20$) lines show detailed particle trajectories. In front of the design domain, an expansion chamber moving the three particles together appears. In the middle of the design domain, a large hook-shaped structure moving the three particles down separates the particles, and the rectangular structure appearing in the rear of the design domain successfully separates the particle with $F_D = 200$, rendered in green, from the other two particles, after which the remaining two particles are separated. The convergence of the design variables is illustrated in Fig. 15c. Some gray elements appear in this example. To remove the gray elements, the no-gray element constraint (37), the density filter with the p -norm, and the S-shaped sigmoid function are applied. An S-shaped function is employed (Yoon and Kim 2003).

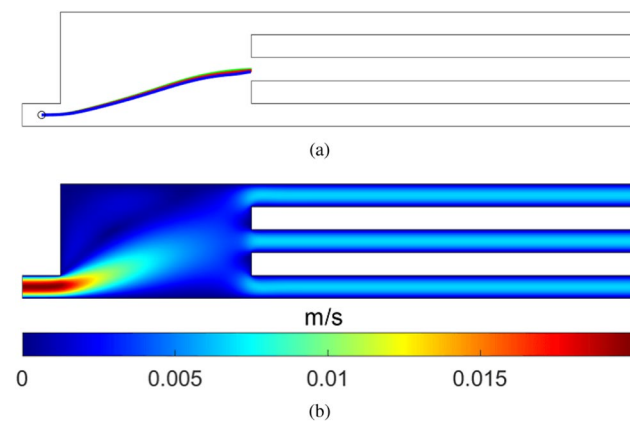


Fig. 14 The analysis result without the solid. **a** The trajectories of the particles and **b** the fluid velocity

$$\text{Density filter (p-norm filter): } \left(\sum_{e \in \text{Neighbor}} \gamma_e^p \right)^{\frac{1}{p}} \quad (35)$$

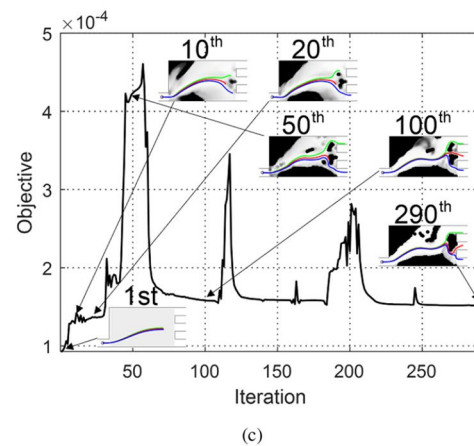
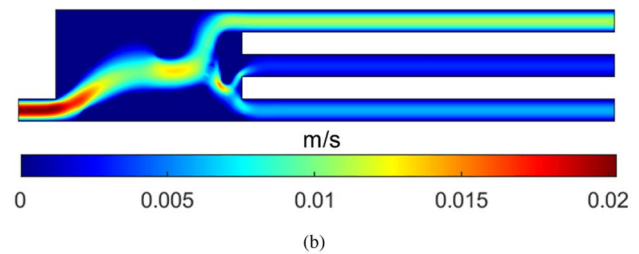
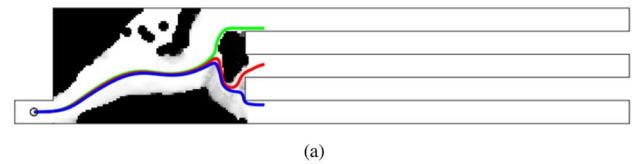


Fig. 15 An optimization result. **a** The trajectories of the particles, **b** the fluid velocity, and **c** the optimization history (The optimization results can be obtained by ParticleInFluid2D(100, 60, 0.05, 0.03, 0.02, 8, 1e-5, 0.5, 0.25, 2.5, 4, 3, 1)). (Color figure online)

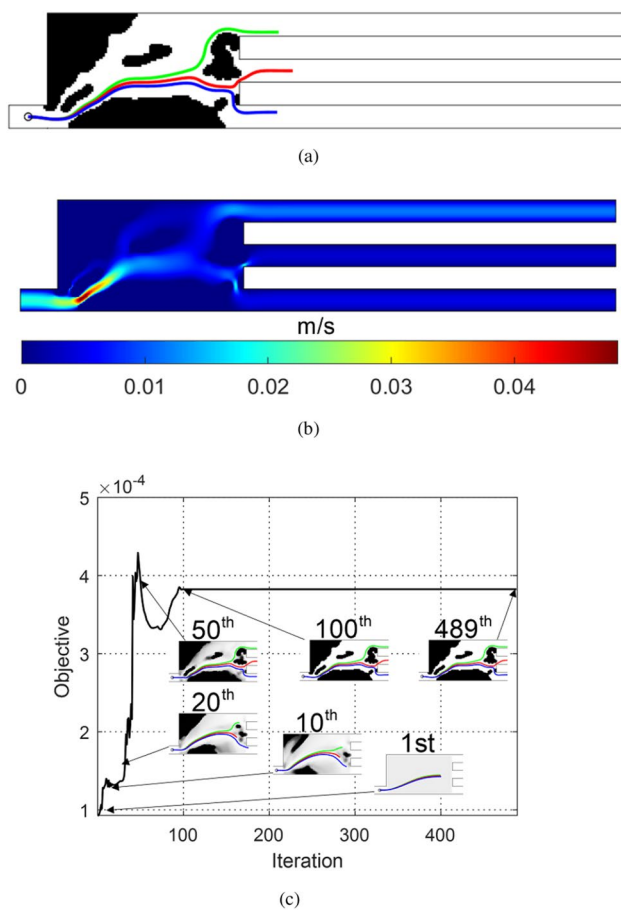


Fig. 16 An optimization result with the no-gray constraint. **a** The trajectories of the particles, **b** the fluid velocity, and **c** the optimization history (The optimization results can be obtained by ParticleInFluid2D(100, 60, 0.05, 0.03, 0.02, 8, 1e-5, 0.5, 0.25, 2.5, 4, 3, 2))

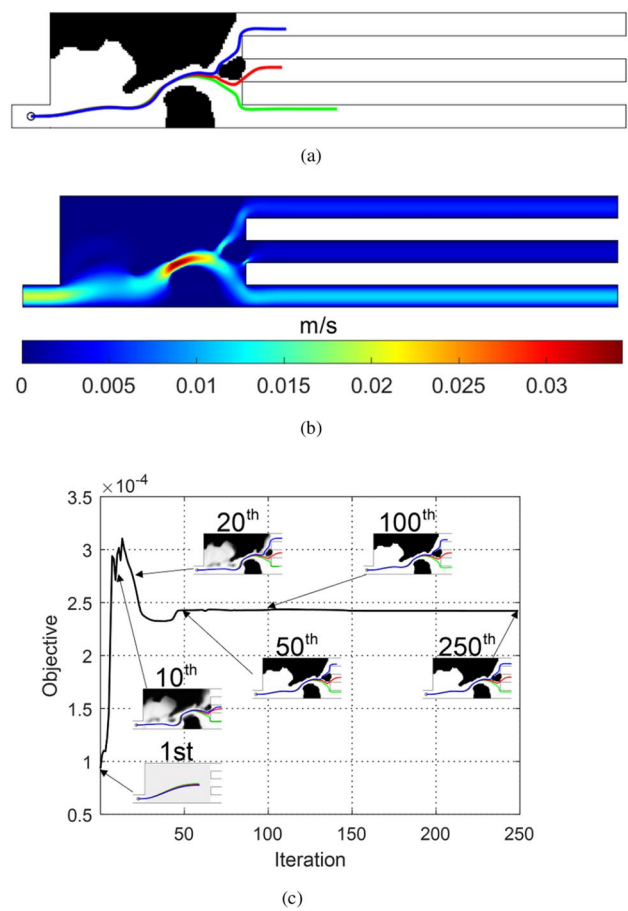


Fig. 18 An optimization result with the reversely applied position constraints. **a** The trajectories of the particles, **b** the fluid velocity, and **c** the optimization history (The optimization results can be obtained by ParticleInFluid2D(100, 60, 0.05, 0.03, 0.02, 8, 1e-5, 0.5, 0.25, 2.5, 4, 3, 3))

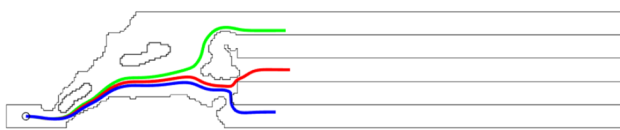


Fig. 17 The postprocessed design and the particle trajectories

$$\text{S-Shape function: } S(\gamma_e) = \frac{1}{1 + \exp(a(\gamma_e - b))} \quad (36)$$

$$\text{No-gray element constraint: } \sum_{e=1}^{NE} (1 - \gamma_e) \gamma_e \leq \varepsilon \quad (37)$$

where the p value for the density filter is set to six and the parameters of the Sigmoid function are set $a = -30, b = 0.5$. After applying the Sigmoid function $\tilde{\gamma} = S(\gamma)$, the inverse permeability is modelled, i.e., $\alpha = \alpha_{\max} \tilde{\gamma}^n$. With these modifications, the design shown in Fig. 16 can be obtained and the separation function can be achievable successfully. To verify the optimized design with the approximated solid region, the design is postprocessed to Fig. 17 and the trajectories in Fig. 17 are almost same. These examples demonstrate the program's validity. Figure 16c shows the design optimization histories. In addition, it is also possible to reshuffle the locations of the particles. For an example, Fig. 18 shows another result with the different locations of the particles. The optimization results can be obtained by ParticleInFluid2D(100, 60, 0.05, 0.03, 0.02, 8, 1e-5, 0.5, 0.25, 2.5, 4, 3, opcase) where opcase is one among 1, 2 and 3.

5 Conclusions

This paper presents a set of MATLAB codes for topology optimization to control the motion of a particle in a steady-state laminar fluid. It is important to control particles in a fluid, and the MATLAB code can help understand fluid topology optimization to achieve this. For simplicity, a steady-state incompressible laminar flow is assumed and solved using the Newton–Raphson method, and the particle size and the effect of the particle on the fluid are neglected. First, the nonlinear steady-state Navier–Stokes equation is solved using the finite element-based MATLAB code (`Steady_Navier_Stokes_2D`). Then, the positions, velocities, and accelerations of particles suspended in the fluid are computed using the code (`Particle_2D`). After obtaining the objective function and constraints, their sensitivity analyses are evaluated by solving the adjoint equations for the position and velocity of the particle (`Sensitivity_Position` and `SensitivityVelocitySquare`). In several optimization formulations, the velocity of the particle or fluid viscous dissipation is considered as the objective function, and the positions of the particles are constrained. Several optimized layouts to control and separate the particles can be obtained using the present MATLAB code. In future research, the size of the particles and the effect of contact between the particle and the wall or among particles can be considered.

Funding This work was supported by a National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (NRF-2019R1A2C2084974).

Declarations

Conflict of interest On behalf of all the authors, the corresponding author states that there are no conflicts of interest.

Replication of results This research provides an introductory code for all the examples.

References

- Alexandersen J, Andreassen CS (2020) A review of topology optimization for fluid-based problems. *Fluids* 5(1):29
- An IH, Lee CH, Lim JH, Lee HY, Yook SJ (2021) Development of a miniature cyclone separator operating at low reynolds numbers as a pre-separator for portable black carbon monitors. *Adv Powder Technol* 32(12):4779–4787. <https://doi.org/10.1016/j.appt.2021.10.027>
- Andreassen CS (2020) A framework for topology optimization of inertial microfluidic particle manipulators. *Struct Multidisc Optim* 61(6):2481–2499. <https://doi.org/10.1007/s00158-019-02483-5>
- Andreassen CS, Sigmund O (2013) Topology optimization of fluid-structure-interaction problems in poroelasticity. *Comput Methods Appl Mech Eng* 258:55–62. <https://doi.org/10.1016/j.cma.2013.02.007>
- Bendsøe MP, Kikuchi N (1988) Generating optimal topologies in structural design using a homogenization method. *Comput Methods Appl Mech Eng* 71(2):197–224. [https://doi.org/10.1016/0045-7825\(88\)90086-2](https://doi.org/10.1016/0045-7825(88)90086-2)
- Bendsøe MP, Sigmund O (2003) Topology optimization: theory, methods, and applications. Springer, Berlin
- Borrvall T, Petersson J (2003) Topology optimization of fluids in stokes flow. *Int J Numer Methods Fluids* 41(1):77–107. <https://doi.org/10.1002/flid.426>
- Deng Y, Liu Z, Zhang P, Liu Y, Wu Y (2011) Topology optimization of unsteady incompressible Navier–Stokes flows. *J Comput Phys* 230(17):6688–6708. <https://doi.org/10.1016/j.jcp.2011.05.004>
- Dilgen CB, Dilgen SB, Fuhrman DR, Sigmund O, Lazarov BS (2018) Topology optimization of turbulent flows. *Comput Methods Appl Mech Eng* 331:363–393. <https://doi.org/10.1016/j.cma.2017.11.029>
- Dinler A, Okumus I (2018) Inertial particle separation in curved networks: a numerical study. *Chem Eng Sci* 182:119–131. <https://doi.org/10.1016/j.ces.2018.02.029>
- Donea J, Huerta A (2003) Finite element methods for flow problems. Wiley, Chichester
- Gersborg-Hansen A, Sigmund O, Haber RB (2005) Topology optimization of channel flow problems. *Struct Multidisc Optim* 30(3):181–192. <https://doi.org/10.1007/s00158-004-0508-7>
- Hashemi Z, Abouali O, Ahmadi G (2016) Direct numerical simulation of particle–fluid interactions: A review. *Iran J Sci Technol Trans Mech Eng* 41(1):71–89. <https://doi.org/10.1007/s40997-016-0035-3>
- Hood K, Kahkeshani S, Di Carlo D, Roper M (2016) Direct measurement of particle inertial migration in rectangular microchannels. *Lab Chip* 16(15):2840–50. <https://doi.org/10.1039/c6lc00314a>
- Hu HH, Joseph DD, Crochet MJ (1992) Direct simulation of fluid particle motions. *Theor Comput Fluid Dyn* 3(5):285–306. <https://doi.org/10.1007/BF00717645>
- Lee H, Balachandar S (2010) Drag and lift forces on a spherical particle moving on a wall in a shear flow at finite re. *J Fluid Mech* 657:89–125. <https://doi.org/10.1017/s0022112010001382>
- Madadelahi M, Acosta-Soto LF, Hosseini S, Martinez-Chapa SO, Madou MJ (2020) Mathematical modeling and computational analysis of centrifugal microfluidic platforms: a review. *Lab Chip* 20(8):1318–1357. <https://doi.org/10.1039/c9lc00775j>
- Makhija D, Pingen G, Yang R, Maute K (2012) Topology optimization of multi-component flows using a multi-relaxation time lattice Boltzmann method. *Comput Fluids* 67:104–114. <https://doi.org/10.1016/j.compfluid.2012.06.018>
- Papoutsis-Kiachagias EM, Giannakoglou KC (2016) Continuous adjoint methods for turbulent flows, applied to shape and topology optimization: Industrial applications. *Arch Comput Methods Eng* 23(2):255–299. <https://doi.org/10.1007/s11831-014-9141-9>
- Papoutsis-Kiachagias E, Kontoleon E, Zymaris A, Papadimitriou D, Giannakoglou K (2011) Constrained topology optimization for laminar and turbulent flows, including heat transfer. In: CIRA (ed) EUROGEN, Evolutionary and Deterministic Methods for Design, Optimization and Control, Capua, Italy
- Picelli R, Vicente WM, Pavanetto R (2017) Evolutionary topology optimization for structural compliance minimization considering design-dependent FSI loads. *Finite Elem Anal Des* 135:44–55. <https://doi.org/10.1016/j.finel.2017.07.005>
- Prohm C, Troltzsch F, Stark H (2013) Optimal control of particle separation in inertial microfluidics. *Eur Phys J E Soft Matter* 36(10):118. <https://doi.org/10.1140/epje/i2013-13118-8>

- Sigmund O (2001) A 99 line topology optimization code written in Matlab. *Struct Multidisc Optim* 21(2):120–127. <https://doi.org/10.1007/s001580050176>
- Svanberg K (1987) The method of moving asymptotes—a new method for structural optimization. *Int J Numer Methods Eng* 24(2):359–373. <https://doi.org/10.1002/nme.1620240207>
- van Dijk NP, Maute K, Langelaar M, van Keulen F (2013) Level-set methods for structural topology optimization: a review. *Struct Multidisc Optim* 48(3):437–472. <https://doi.org/10.1007/s00158-013-0912-y>
- Walsh MJ (1976) Influence of particle drag coefficient on particle motion in high-speed flow with typical laser velocimeter applications. Report, NASA
- Wang C, Zhao Z, Zhou M, Sigmund O, Zhang XS (2021) A comprehensive review of educational articles on structural and multidisciplinary optimization. *Struct Multidisc Optim* 64(5):2827–2880. <https://doi.org/10.1007/s00158-021-03050-7>
- Wu YC, Yang B (2018) An overview of numerical methods for incompressible viscous flow with moving particles. *Arch Comput Methods Eng* 26(4):1255–1282. <https://doi.org/10.1007/s11831-018-9277-0>
- Yoon GH (2010) Topological design of heat dissipating structure with forced convective heat transfer. *J Mech Sci Technol* 24(6):1225–1233. <https://doi.org/10.1007/s12206-010-0328-1>
- Yoon GH (2012) Topological layout design of electro-fluid-thermal-compliant actuator. *Comput Methods Appl Mech Eng* 209–212:28–44. <https://doi.org/10.1016/j.cma.2011.11.005>
- Yoon GH (2014) Stress-based topology optimization method for steady-state fluid–structure interaction problems. *Comput Methods Appl Mech Eng* 278:499–523. <https://doi.org/10.1016/j.cma.2014.05.021>
- Yoon GH (2016) Topology optimization for turbulent flow with Spalart–Allmaras model. *Comput Methods Appl Mech Eng* 303:288–311. <https://doi.org/10.1016/j.cma.2016.01.014>
- Yoon GH (2020) Transient sensitivity analysis and topology optimization for particle motion in steady state laminar fluid. *Comput Methods Appl Mech Eng*. <https://doi.org/10.1016/j.cma.2020.113096>
- Yoon GH (2022) Transient sensitivity analysis and topology optimization of particle suspended in transient laminar fluid. *Comput Methods Appl Mech Eng* 393:114696. <https://doi.org/10.1016/j.cma.2022.114696>
- Yoon GH, Kim YY (2003) The role of s-shape mapping functions in the simp approach for topology optimization. *KSME Int J* 17(10):1496–1506. <https://doi.org/10.1007/BF02982329>
- Yoon GH, So H (2021) Development of topological optimization schemes controlling the trajectories of multiple particles in fluid. *Struct Multidiscip Optim*. <https://doi.org/10.1007/s00158-020-02817-8>
- Yoon GH, Yi B (2019) A new coating filter of coated structure for topology optimization. *Struct Multidisc Optim* 60(4):1527–1544. <https://doi.org/10.1007/s00158-019-02279-7>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.