Karlsruhe Institute of Technology

Project Report

Topology Optimization of Bend for Different Flow Configurations

Computational Fluid Dynamics and Simulation Lab SS 2025

Jiaqi Lan

August 23, 2025

Contents

1 Introduction

In the process of fluid transportation, the optimized design of bent pipes can effectively reduce energy loss and pressure drop of the fluid. Among them, the optimization process of bent pipes can be addressed using numerical simulation methods. The model used in this report is based on the lattice Boltzmann method (LBM) and topology optimization (TO) [1][2]. The LBM is a numerical approach for simulating fluid flow by modeling particle collisions and propagation on a lattice grid. TO is a mathematical method for designing optimal structures by optimizing the distribution of material within a given design domain.

In addition, to determine the optimal topology of the bent pipe, this model employs a homogenized lattice Boltzmann method with a porosity parameter P [3]. Meanwhile, to find the minimum value of the objective function, this report uses the adjoint method to construct the adjoint homogenized lattice Boltzmann equations [4]. Finally, a gradient-based line search algorithm is applied to iteratively solve for the minimum of the objective function [5].

This report will explore the numerical stability and the results of topology optimization simulations for different resolutions, Reynolds numbers, objective functions, shapes of the design domain, and positions of inlet and outlet in order to analyze the characteristics of this topology optimization model and its optimization results.

2 Problem Statement

In this report, a 2D simplified model of a bent pipe is investigated, as shown in Fig. 1. The design domain is defined as a square with a side length of L, from which two right-angled isosceles triangles with leg lengths of L/2 are removed. The purpose of removing these two triangles is to reduce the computational domain, thereby decreasing the simulation time. Here, L = 0.5 m. The inlet boundary condition Γ_{inlet} is defined as a fixed inflow rate $u = 0.06$ m/s. The outlet boundary condition Γ_{outlet} is set as an outlet with a pressure value of 0.

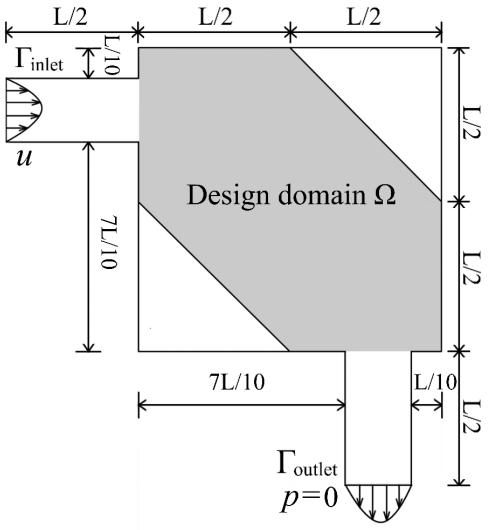

Figure 1. Design domain and boundary conditions

In the initial parameter settings, the fluid density $\rho = 1$ kg/m^3, and the kinematic viscosity $\nu = 0.001$ m^2/s. According to Eq. (2.1), the Reynolds number Re = 30. The model resolution N = 20.

$$\mathrm{Re} = \frac{u\mathrm{L}}{\nu} \tag{2.1}$$

2.1 Governing Equations

In this model, the topology optimization problem can be formulated as follows:

$$\min_{\boldsymbol{\alpha}} \; J(f, \boldsymbol{\alpha}) \quad \text{subject to } G(f, \boldsymbol{\alpha}) = 0 \tag{2.2}$$

where f is the state, $\vec{\alpha}$ is the vector of design variables, J is an objective function, and G is the equality constraint.

2.1.1 Constraint Equation

Based on the LBM method, the constraint $G(f, \boldsymbol{\alpha}) = 0$ can be expressed as the homogenized lattice BGK-Boltzmann equation [3]:

$$f_i(\mathbf{x} + \mathbf{e}_i \Delta t, t + \Delta t) - f_i(\mathbf{x}, t) + \frac{\Delta t}{\tau} \left[f_i(\mathbf{x}, t) - f_{i,P}^{eq}(\mathbf{x}, t) \right] = 0 \quad \text{in } \Omega_{\Delta x} \times I_{\Delta t} \tag{2.3}$$

where $\tau \in \mathbb{R}^+$ is the relaxation time, $e_i \in \mathbb{R}^2$ for $i \in 0, 1, ..., 8$ in the velocity model $D2Q9$, as shown in Fig. 2, $f_i : \Omega_{\Delta x} \times I_{\Delta t} \longrightarrow \mathbb{R}$ is the particle distribution function, and $f_{i,P}^{eq} : \Omega_{\Delta x} \times I_{\Delta t} \longrightarrow \mathbb{R}$ is the discrete equilibrium distribution function for the homogenized BGK-Boltzmann equation, which is expressed as:

$$f_{i.P}^{eq} = w_i \rho \left[1 + \frac{\mathbf{e}_i \cdot P\mathbf{u}}{c_s^2} + \frac{(\mathbf{e}_i \cdot P\mathbf{u})^2}{2c_s^4} - \frac{P\mathbf{u} \cdot P\mathbf{u}}{2c_s^2} \right] \tag{2.4}$$

where w_i are weights corresponding to the D2Q9 model as shown in Fig. 2, $\rho : \Omega \times I \longrightarrow \mathbb{R}^+$ is the macroscopic density, $\mathbf{u} : \Omega \times I \longrightarrow \mathbb{R}^2$ is the macroscopic velocity, the lattice speed of sound is $c_s = \Delta t / \sqrt{3} \Delta x$, and the porosity distribution is $P : \Omega \longrightarrow [0, 1]$, which represents the fluid permeability at point $\mathbf{x}$; when $P(\mathbf{x})$ is 0, it means the point is impermeable, and when $P(\mathbf{x})$ is 1, it means the fluid can pass through completely. Its definition is as follows [4]:

$$P(\alpha) = \frac{e^{\alpha}}{e^{\alpha} + G_h} \quad \text{with } G_h = \Delta x^2 \nu \tau \tag{2.5}$$

In addition, the macroscopic density ρ and velocity $\mathbf{u}$ can be represented as follows:

$$\rho(\mathbf{x}, t) = \sum_{i=0}^{8} f_i \tag{2.6}$$

$$\mathbf{u}(\mathbf{x}, t) = \frac{1}{\rho} \sum_{i=0}^{8} e_i f_i \tag{2.7}$$

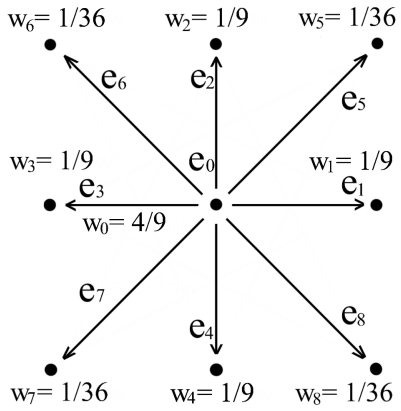

Figure 2. The distribution of e_i and w_i in D2Q9 model

2.1.2 Objective Function

The objective function $J(f, \boldsymbol{\alpha})$ of this model can be chosen as either the total dissipation, which includes viscous dissipation energy and porous dissipation, or the pressure drop. When the objective function J is defined as the total dissipation, its expression is as follows:

$$J(f, \boldsymbol{\alpha}) = \sum_{\mathbf{x}\in\Omega} \left[\varepsilon_{\text{visc}}(\mathbf{x}) + \varepsilon_{\text{porous}}(\mathbf{x})\right] + \boldsymbol{\alpha} \cdot \frac{V}{V_{\text{ref}}} \quad \text{in } \Omega_{\Delta x} \tag{2.8}$$

where $\varepsilon_{\text{visc}}$ is viscous dissipation, $\varepsilon_{\text{porous}}$ is porous dissipation, V is the total material volume, and $V_{\text{ref}} = 0.25$ is the reference volume. The expression for viscous dissipation $\varepsilon_{\text{visc}}$ is given as follows:

$$\varepsilon_{\text{visc}}(\mathbf{x}) = \frac{1}{2}\nu \left(\frac{1}{\tau\rho(\mathbf{x})c_s^2\Delta t}\right)^2 \boldsymbol{\Pi}^{\text{neq}}(\mathbf{x})^2 \tag{2.9}$$

where $\boldsymbol{\Pi}^{\text{neq}}(\mathbf{x})$ is the non-equilibrium stress tensor, and its expression is as follows:

$$\boldsymbol{\Pi}^{\text{neq}}_{xy} = \sum_{i=0}^{8} e_{ix}e_{iy}(f_i - f^{eq}_{i,P}) \tag{2.10}$$

where e_{ix} is the component in the x direction of the i-th discrete velocity. e_{iy} is defined similarly. The expression for viscous dissipation $\varepsilon_{\text{visc}}$ is given as follows:

$$\varepsilon_{\text{porous}}(\mathbf{x}) = \nu \cdot \frac{\mathbf{u}(\mathbf{x})^2}{K(\mathbf{x})} \tag{2.11}$$

where $K : \mathbb{R} \to [0, \infty)$ is the local permeability. It is defined as follows:

$$K(\mathbf{x}) = e^{\alpha} + G_h \tag{2.12}$$

The total material volume V is given as follows:

$$V = \sum_{\mathbf{x}\in\Omega} P(\mathbf{x}) \cdot \Delta x^2 \tag{2.13}$$

When the objective function J is defined as the pressure drop, its expression is as follows:

$$J(f, \boldsymbol{\alpha}) = \sum_{\mathbf{x}\in\Omega_{\text{MN=7}}} p(\mathbf{x}) - \sum_{\mathbf{x}\in\Omega_{\text{MN=8}}} p(\mathbf{x}) + \boldsymbol{\alpha} \cdot \frac{V}{V_{\text{ref}}} \quad \text{in } \Omega_{\Delta x} \tag{2.14}$$

where $\Omega_{\text{MN=7}}$ is the region with MN (material number) = 7. $\Omega_{\text{MN=8}}$ is defined similarly.

2.1.3 Adjoint Method

In order to get the minimum J, that is, to achieve $\frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{\alpha}} = 0$ using gradient-based optimization, the adjoint method is used here. By introducing an adjoint variable λ, the adjoint equation can be constructed as follows:

$$\frac{\partial J(f, \boldsymbol{\alpha})}{\partial f} + \lambda \cdot \frac{\partial G(f, \boldsymbol{\alpha})}{\partial f} = 0 \tag{2.15}$$

here, $\lambda \cdot \frac{\partial G}{\partial f}$ is defined as a new function ϕ. By coupling the adjoint equations with the homogenized lattice BGK-Boltzmann equation with the method proposed by Krause et al., the following expression for the discrete adjoint homogenized lattice BGK-Boltzmann equation can be obtained [3]:

$$\phi_i(\mathbf{x} + \mathbf{e}_i\Delta t, t + \Delta t) - \phi_i(\mathbf{x}, t) + \frac{\Delta t}{\tau}(\phi_i - \phi_{i,P}^{eq}) + \frac{\partial J}{\partial f} = 0 \quad \text{in } \Omega_{\Delta x} \times I_{\Delta t} \tag{2.16}$$

The definition of discrete adjoint equilibrium distribution $\phi_{i,P}^{eq}$ is given as follows:

$$\phi_{P,i}^{eq}(\mathbf{x}, t) = \frac{1}{\rho}\sum_{j=0}^{8} \phi_j\big(1 + 3P(\mathbf{e}_j - Pu)(\mathbf{e}_i - u)\big) f_P^{eq}(\mathbf{x}, t) \tag{2.17}$$

After solving for $\frac{\partial G}{\partial \boldsymbol{\alpha}}$ in (2.3), $\frac{\partial J}{\partial \boldsymbol{\alpha}}$ in (2.8 or 2.14) and λ in (2.16), we substitute them into the following equation to obtain the value of $\frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{\alpha}}$:

$$\frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{\alpha}} = \frac{\partial J}{\partial \boldsymbol{\alpha}} + \lambda \cdot \frac{\partial G}{\partial \boldsymbol{\alpha}} \tag{2.18}$$

2.1.4 Gradient-based Line Search Algorithm

To solve for $\frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{\alpha}} = 0$, the gradient-based line search algorithm is used here. After specifying the initial values of $\boldsymbol{\alpha}_0$, maximal number of iterations, the iterative process to solve for $\frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{\alpha}} = 0$ begins. At iteration step $k \in 1, 2, ..., k_{max}$, using L-BFGS (Quasi-Newton Method), an approximate Hessian matrix H_k is constructed so that [5]:

$$\mathbf{d}_k = -H_k^{-1}(\frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{\alpha}})_k \tag{2.19}$$

A one-dimensional search is performed along direction $\mathbf{d}_k$ with a step size of $\beta_{k,m}$ ($m \in 1, 2, ..., m_{max}$), through which the optimal step size β_k^* is determined:

$$\min_{\beta} \; J(f_k, \boldsymbol{\alpha}_k + \beta_{k,m} \cdot \mathbf{d}_k) \tag{2.20}$$

The variables are then updated accordingly:

$$\boldsymbol{\alpha}_{k+1} = \beta_k^* \cdot \mathbf{d}_k \tag{2.21}$$

The computed solution is then substituted back into equations (2.3), (2.8 or 2.14) and (2.16), then $(\frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{\alpha}})_{k+1}$ and H_k is updated once more. These steps are repeated until $J(f_k, \boldsymbol{\alpha}_k) < \varepsilon$ or $(\frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{\alpha}})_k < \varepsilon$ or $k = k_{max}$, where ε is a user-specified threshold.

2.2 Geometry and Boundary Conditions

The number of materials in the model is shown in Fig. 3.

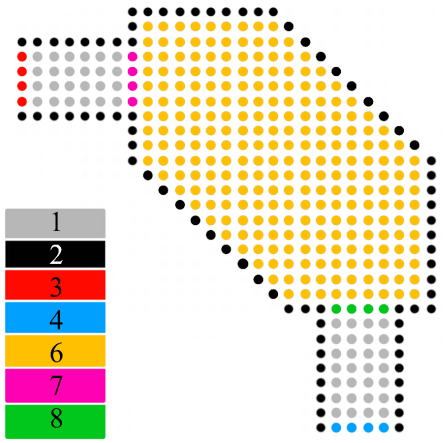

Figure 3. Distribution of material numbers (MN)

In primal mode, the fluid is simulated using the homogenized lattice BGK-Boltzmann method.

MN = 1: Flow
MN = 2: Boundary (**`BounceBack`** boundary)
MN = 3: Poiseuille inflow (**`InterpolatedVelocity`** boundary)
MN = 4: Outflow (**`InterpolatedPressure`** boundary)
MN = 6: Flow (porosity adjustment zone)
MN = 7: Flow (inlet pressure measurement zone)
MN = 8: Flow (outlet pressure measurement zone)

In dual mode, the fluid is simulated using the adjoint homogenized lattice BGK-Boltzmann method.

MN = 1, 6, 7, 8: Flow
MN = 2, 3, 4: Boundary (**`BounceBack`** boundary)

3 Results and Discussion

Parameters can be modified by editing the file:
`release-1.8.1/Examples/optimization/PipeBendSolver2d/parameter.xml`
Functions and geometry can be modified by editing the file:
`release-1.8.1/Examples/optimization/PipeBendSolver2d/pipeBend2d.cpp`

3.1 Numerical Stability Analysis

The analysis of numerical stability is based on whether stable solutions can be obtained in the model under different Reynolds numbers and resolutions. In this case, the model uses the pressure drop as the objective function.

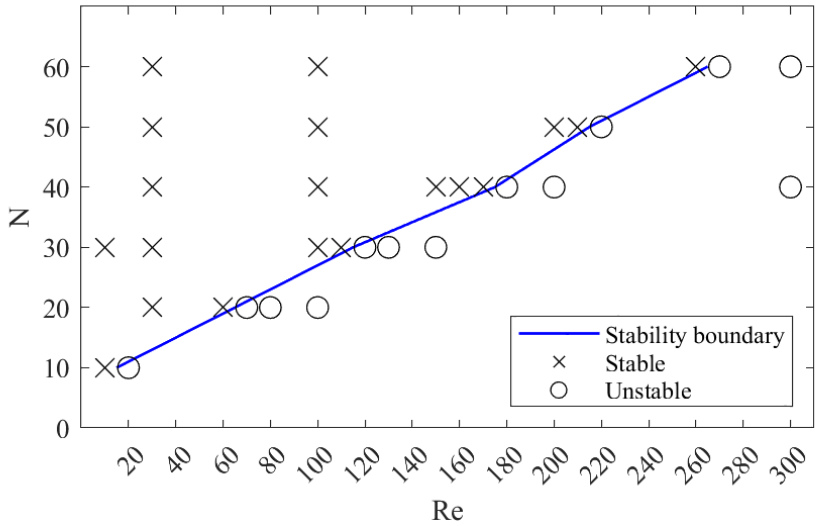

Figure 4. Numerical stability for different Reynolds numbers (Re) and resolutions (N)

As shown in Fig. 4, with the increase of the Reynolds number, a higher resolution is required for the model to reach a steady state. This is because, according to Equations (2.1) and (3.1), as the Reynolds number Re increases (which corresponds to a smaller kinematic viscosity ν), the physical time step Δt of the model becomes larger when $\Delta x = \mathrm{L}/\mathrm{N}$ remains constant. To

reduce the physical time step Δt (i.e., to make $u_{lb} = u_{phys} \cdot \Delta t/\Delta x$ smaller), the resolution N must be increased to decrease Δx, thereby achieving the steady state.

$$\tau - 0.5 = \frac{v.\Delta t}{c_s^2.\Delta x^2} \quad \text{which } \tau = 0.525 \tag{3.1}$$

In addition, as seen in Figure 4, the boundary line that determines the steady state appears to be approximately linear as shown in (3.2).

$$\mathrm{N}^* = 0.1985 \cdot \mathrm{Re}^* + 6.88 \tag{3.2}$$

3.2 Influence of Resolution

To investigate the effect of different resolution N on the optimization model, I will fix the Reynolds number Re at 30 and then test the model by selecting resolutionN $\in \{20, 30, 40, 50, 60\}$. In this case, the model uses the pressure drop as the objective function. The results are shown in Table 3.1 and Fig. 5.

Resolution N	Pressure drop Δp/Pa	Total dissipation ε/(W/m)
20	0.0103152	0.0143624
30	0.0089949	0.0120357
40	0.0086677	0.0113242
50	0.0084641	0.0108933
60	0.0085612	0.0101771

Table 3.1: Pressure drop and total dissipation for different resolution N

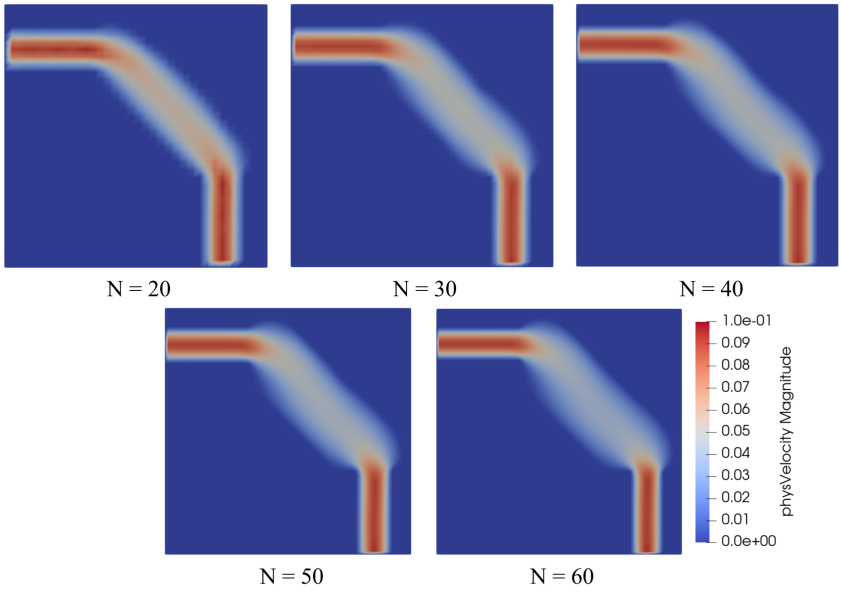

Figure 6. Distribution of velocity $\|\mathbf{u}\|$ for different resolution N

As shown in Table 3.1 and Fig. 5, the pressure drop and total dissipation obtained at different resolutions are not the same. This is because the porosity P is related to the lattice size Δx, as described by (2.5). In addition, as the resolution N increases, both the pressure drop Δp and the total dissipation ε decrease.

3.3 Influence of Reynolds Number

To investigate the effect of the Reynolds number on the results, the resolution N = 60 was chosen, which is capable of simulating a wide range of Reynolds numbers. The Reynolds number was set to Re $\in \{30, 100, 260\}$. In this case, the model uses the pressure drop as the objective function. The results are shown in Table 3.2 and Figure 6.

Reynolds number Re	Pressure drop Δp/Pa	Total dissipation ε/(W/m)
30	0.0085612	0.0101771
100	0.0030890	0.0045732
260	0.0014959	0.0028059

Table 3.2: Pressure drop and total dissipation for different Reynolds number Re

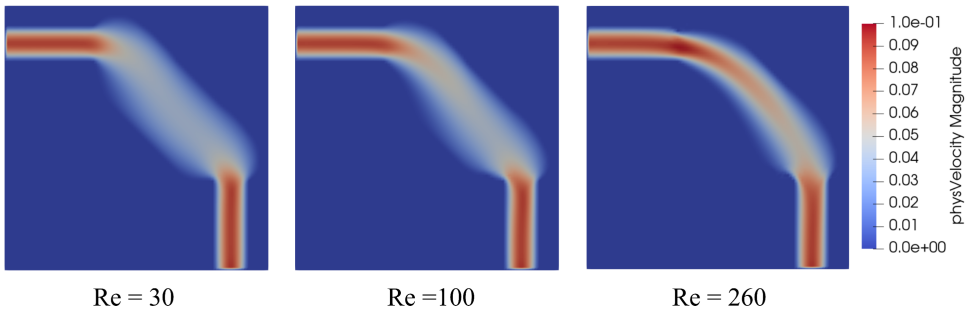

Figure 6. Distribution of velocity $\|\mathbf{u}\|$ for different Reynolds number Re

As shown in Table 3.2 and Figure 6, with the increase of the Reynolds number, both the pressure drop and the total dissipation decrease continuously. This is because as the Reynolds number increases, the kinematic viscosity of the fluid decreases, which leads to smaller viscous dissipation caused by viscous forces. According to (2.8), this results in a lower total dissipation. For the pressure drop, this can be explained as follows: according to the conservation of momentum of the incompressible homogenized Navier–Stokes equations shown below, a smaller kinematic viscosity leads to a smaller stress tensor. When the velocity remains unchanged, this results in a smaller pressure gradient, which means the pressure drop decreases [4].

$$(\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \nu \Delta \mathbf{u} + \nu K(\alpha)^{-1}\mathbf{u} \tag{3.3}$$

In addition, it can be observed that as the Reynolds number increases, the optimized channel becomes progressively narrower.

3.4 Investigation of Different Objective Function

To investigate the effect of different objective functions on the model, the resolution N = 30 and the Reynolds number Re $\in \{10, 30, 100\}$ were selected for comparison. The results are shown in Table 3.3 and Fig. 7.

Objective	Pressure drop		Total dissipation	
Re	Δp/Pa	ε/(W/m)	Δp/Pa	ε/(W/m)
10	0.0231601	0.0287334	0.0328625	0.0016036
30	0.0089949	0.0120357	0.0138446	0.0013146
100	0.0035485	0.0055066	0.0065513	0.0010814

Table 3.3: Pressure drop and total dissipation for different objective function

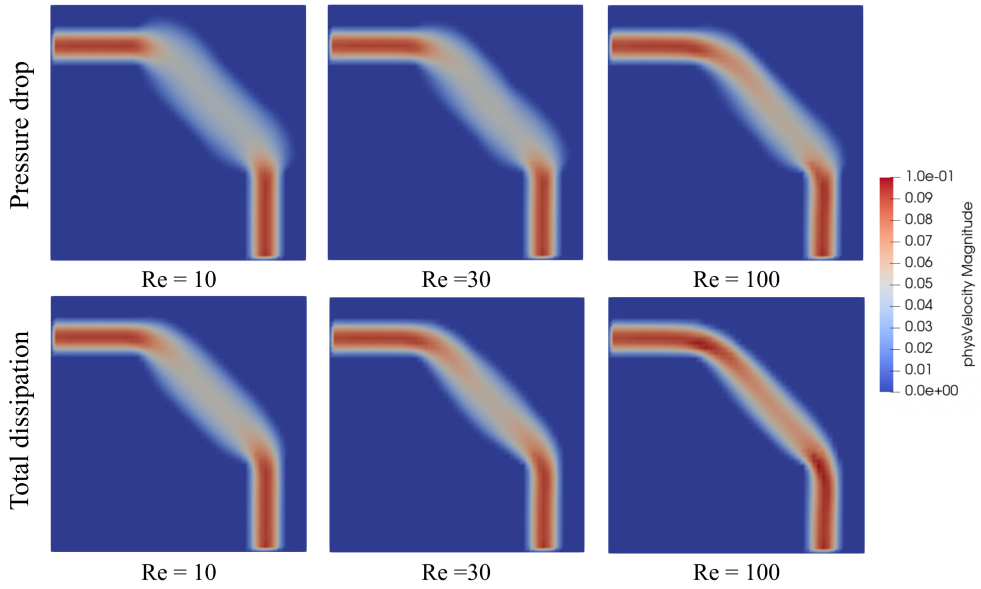

Figure 7. Distribution of velocity $\|\mathbf{u}\|$ for different objective function

From the results, it can be seen that the minimum values obtained vary depending on the choice of objective function. For example, when the objective function is the pressure drop, a lower pressure drop can be achieved. In addition, as shown in the figure, when the total dissipation is used as the objective function, the final geometry of the bend becomes smoother, and the area of the simulated bend is smaller.

3.5 Investigation of Different Design Domain

As shown in Figure 1, to reduce computational cost, the model's design domain has two triangular regions removed on either side. To investigate the effect of different design domains

on the simulation, the design domain is restored to a full square for comparison. Among them, the resolution N = 30 and the Reynolds number Re $\in \{10, 30, 100\}$ were selected for comparison, and the model uses the pressure drop as the objective function. The results are shown in Table 3.4 and Fig. 8.

Domain	Hexagon domain		Square domain	
Re	Δp/Pa	ε/(W/m)	Δp/Pa	ε/(W/m)
10	0.0231601	0.0287334	0.0214632	0.0269669
30	0.0089949	0.0120357	0.0079775	0.0110939
100	0.0035485	0.0055066	0.0031187	0.0051985

Table 3.4: Pressure drop and total dissipation for different design domain

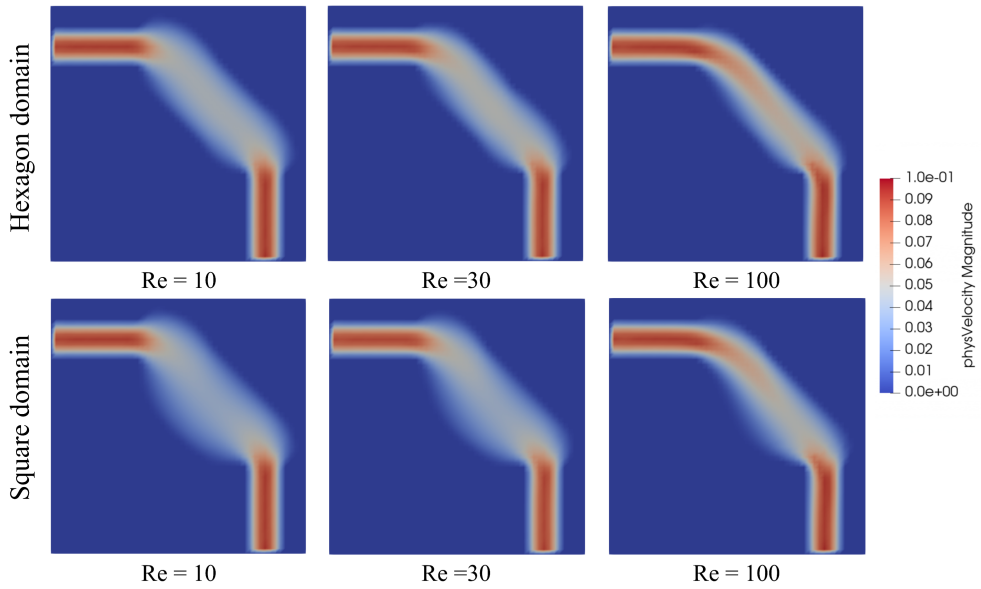

Figure 8. Distribution of velocity $\|\mathbf{u}\|$ for different design domain

From the results, it can be seen that in the uncut square design domain, smaller pressure drop and total dissipation are achieved. This indicates that under these experimental conditions, the cut domain affects the minimum value of the objective function. The geometry shown in the figure also reveals that, in the square domain, the fluid occupies a larger area, extending beyond the original cut boundary.

3.6 Investigation of Different Geometry

To explore the performance of this topology optimization approach with other geometric configurations, three different shapes were selected:

Geometry 1: the outlet pipe located on the right side.
Geometry 2: two outlet pipes located on the right side.
Geometry 3: two outlet pipes located on the top and bottom sides.

Here, the resolution N = 30 and Reynolds number Re = 100 are used, with the pressure drop as the objective function, while keeping other parameters unchanged.

Modify the **`release-1.8.1/Examples/optimization/PipeBendSolver2d`**.

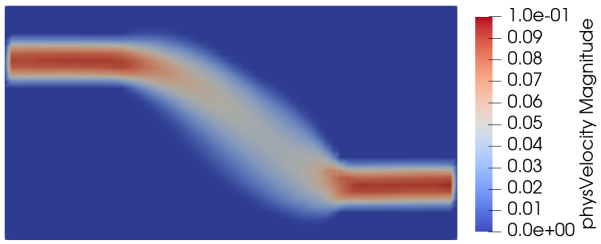

Figure 9. Distribution of velocity ||**u**|| for Geometry 1

For geometry 1, modified code on lines 145-146 and 177-185:

```
145: Vector<T,2> origin(-params.inflowLength, -params.outflowLength);
146: Vector<T,2> extend(params.lengthX + params.inflowLength, params.lengthY
+ params.outflowLength);
177: origin= Vector<T,2>( params.lengthX, params.lengthY - params.inflowY
- params.outflowRadius );
178: extend = Vector<T,2>( params.L, 2.  * params.outflowRadius );
179: IndicatorCuboid2D<T> outflow( extend, origin );
180: origin[1] = -.5*params.L;
181: std::shared_ptr<IndicatorF2D<T> > objectiveDomainOutflow
182: = std::make_shared<IndicatorCuboid2D<T> >( extend, origin );
183: origin = Vector<T,2>( params.lengthX, params.lengthY - params.inflowY
- params.outflowRadius - params.L );
184: extend = Vector<T,2>( params.outflowLength + params.L,
2.*params.outflowRadius + 2*params.L );
185: IndicatorCuboid2D<T> extendedOutflow( extend, origin );
```

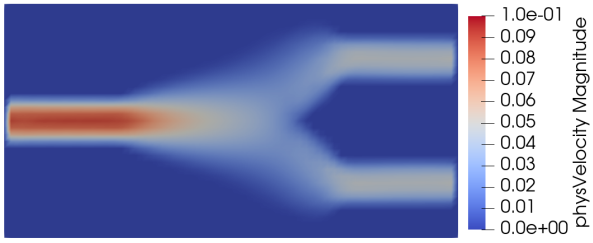

Figure 10. Distribution of velocity ||**u**|| for Geometry 2

For geometry 2, modified code on lines 145-146 and 177-194:

```
145: Vector<T,2> origin(-params.inflowLength, -params.outflowLength);
146: Vector<T,2> extend(params.lengthX + params.inflowLength, params.lengthY
+ params.outflowLength);
177: origin= Vector<T,2>( ( params.lengthX + params.outflowLength -.5*params.L,
params.lengthY - params.inflowY - params.outflowRadius );
178: extend = Vector<T,2>( params.L, 2.  * params.outflowRadius );
179: IndicatorCuboid2D<T> outflow( extend, origin );
180: origin[0] = params.lengthX -.5*params.L;
181: std::shared_ptr<IndicatorF2D<T> > objectiveDomainOutflow
182: = std::make_shared<IndicatorCuboid2D<T> >( extend, origin );
183: origin = Vector<T,2>( params.lengthX -.5*params.L, params.lengthY
- params.inflowY - params.outflowRadius - params.L );
184: extend = Vector<T,2>( params.outflowLength + params.L,
2.*params.outflowRadius + 2*params.L );
185: IndicatorCuboid2D<T> extendedOutflow( extend, origin );
187: origin = Vector<T,2>( params.lengthX + params.outflowLength
-.5*params.L, params.inflowY - params.outflowRadius );
188: extend = Vector<T,2>( params.L, 2.  * params.outflowRadius );
189: IndicatorCuboid2D<T> outflow( extend, origin );
190: origin[0] = params.lengthX -.5*params.L;
191: std::shared_ptr<IndicatorF2D<T> > objectiveDomainOutflow
192: = std::make_shared<IndicatorCuboid2D<T> >( extend, origin );
193: origin = Vector<T,2>( params.lengthX -.5*params.L, params.inflowY
- params.outflowRadius - params.L);
194: extend = Vector<T,2>( params.outflowLength + params.L,
2.*params.outflowRadius + 2*params.L );
195: IndicatorCuboid2D<T> extendedOutflow( extend, origin );
```

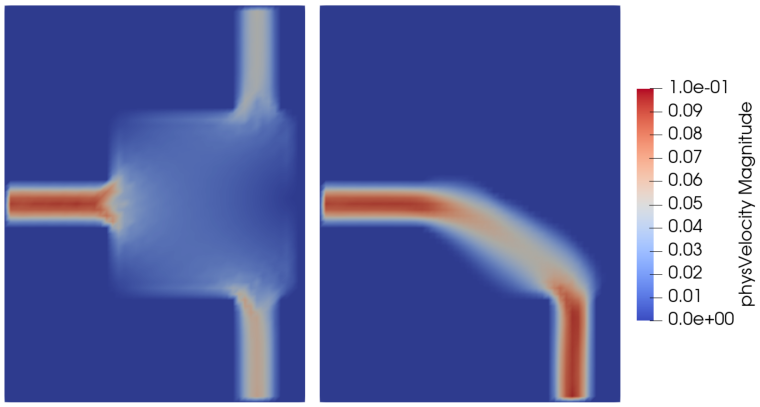

Figure 11. Distribution of velocity $\|\mathbf{u}\|$ for Geometry 3
(The left figure shows the velocity distribution at the first iteration step, while the right figure shows the velocity distribution at the final iteration step)

For geometry 3, modified code on lines 145-146 and 177-194:

```
145: Vector<T,2> origin(-params.inflowLength, -params.outflowLength);
146: Vector<T,2> extend(params.lengthX + params.inflowLength, params.lengthY
+ 2 * params.outflowLength);
177: origin = Vector<T,2>( params.outflowX - params.outflowRadius,
-params.outflowLength -.5*params.L );
178: extend = Vector<T,2>( 2.  * params.outflowRadius, params.L );
179: IndicatorCuboid2D<T> outflow( extend, origin );
180: origin[1] = -.5*params.L;
181: std::shared_ptr<IndicatorF2D<T> > objectiveDomainOutflow
182: = std::make_shared<IndicatorCuboid2D<T> >( extend, origin );
183: origin = Vector<T,2>( params.outflowX - params.outflowRadius - params.L,
-params.outflowLength -.5*params.L );
184: 2.*params.outflowRadius + 2*params.L, params.outflowLength
+ params.L );
185: IndicatorCuboid2D<T> extendedOutflow( extend, origin );
187: origin = Vector<T,2>( params.outflowX - params.outflowRadius,
params.lengthY + params.outflowLength -.5*params.L );
188: extend = Vector<T,2>( 2.  * params.outflowRadius, params.L );
189: IndicatorCuboid2D<T> outflow( extend, origin );
190: origin[1] = -.5*params.L;
191: std::shared_ptr<IndicatorF2D<T> > objectiveDomainOutflow
192: = std::make_shared<IndicatorCuboid2D<T> >( extend, origin );
193: origin = Vector<T,2>( params.outflowX - params.outflowRadius - params.L,
params.lengthY -.5*params.L );
194: 2.*params.outflowRadius + 2*params.L, params.outflowLength
+ params.L );
195: IndicatorCuboid2D<T> extendedOutflow( extend, origin );
```

As shown in Fig. 11, it is worth noting that under the conditions of Geometry 3, the optimized channel only extends toward one of the outlets.

4 Conclusion

The conclusions of this report can be summarized in the following 5 points:

- As the Reynolds number increases, a higher resolution is required to stabilize the model, and this relationship is approximately linear.
- At the same Reynolds number, when using pressure drop as the objective function, both the pressure drop and the total dissipation decrease as the resolution increases.
- At the same resolution, when using pressure drop as the objective function, both the pressure drop and the total dissipation decrease as the Reynolds number increases.
- When the total dissipation is used as the objective function, the resulting total dissipation is lower, and the optimized channel geometry becomes smoother with a smaller area.
- A trimmed design domain can limit the final optimization result; in an untrimmed square domain, lower pressure drop and total dissipation can be achieved.

References

[1] T Krüger et al. *The Lattice Boltzmann method*. Springer International Publishing Switzerland, 2017. DOI: 10.1007/978-3-319-44649-3.

[2] MP Bendsøe and O Sigmund. *Topology optimization: the ory, methods, and applications*. Springer Berlin, Heidelberg, 2003. DOI: 10.1007/978-3-662-05086-6.

[3] M J Krause, G Thäter, and V Heuveline. "Adjoint-based fluid flow control and optimisation with lattice boltzmann methods". In: *Computers & Mathematics with Applications* 65 (6) (2013), pp. 945–960. DOI: 10.1016/j.camwa.2012.08.007.

[4] M J Krause et al. "Particle flow simulations with homogenised lattice boltzmann methods". In: *Particuology* 34 (2017), pp. 1–13. DOI: 10.1016/j.partic.2016.11.001.

[5] R H Byrd et al. "A limited memory algorithm for bound constrained optimization". In: *SIAM Journal on Scientific Computing* 16 (5) (1995), pp. 1190–1208. DOI: 10.1137/0916069.