CFD Course Project

CFD Study of Liquid-Cooled Heat Sinks with Microchannel Flow Field Configurations

Qifan Gu

Project Statement and Literature Review

Heat dissipation is a critical problem in design of electronics, fuel cells, and concentrated solar cells [1]. Air-cooled heat sinks are widely used for effective cooling on these applications [2], while they cannot provide sufficient cooling capability for high-heat-releasing devices. Therefore, in recent years liquid-cooled heat sinks have increased applications in high-heat-releasing applications [3]. When the cooling system utilizes the phase change of the medium, its cooling capability will have further increase.

To understand and optimize the heat dissipation process of the cooling system, a large amount of CFD studies has been conducted to simulate the heat transfer phenomenon [4]–[6]. It has been demonstrated that classical fluid dynamics and heat transfer models and equations can obtain good agreement with experiment results for the microchannel size between 100-1000 μm [1], [7]. The channel size in this project is 900-1200 μm, which is mostly within the mentioned range, therefore classical fluid dynamics and heat transfer models and equations will be used in this project. One of the most popular applications of CFD in heat sink design is the optimization of heat sink configuration. This is because different heat sink configuration could have significantly different cooling performance. Fig. 1 shows some common flow field configurations in published literature.

The purpose of this project is to simulate the steady state flow in the channel and the corresponding heat transfer process with Serpentine-A configuration (see **Fig. 2**). The cooling fluid is water, which is injected from the upper side of the heat sink and is ejected from the lower side. The flow happens in a single continuous channel which has parallel configuration. To avoid the complexity of 3D computation, the problem is simplified as a 2D flow on the x-y plane (see **Fig. 2**). The heat source is located at the bottom of the heat sink, which is taken as a constant body heat flux with homogeneous distribution in this project. The boundaries of the heat sink are assumed to be adiabatic.

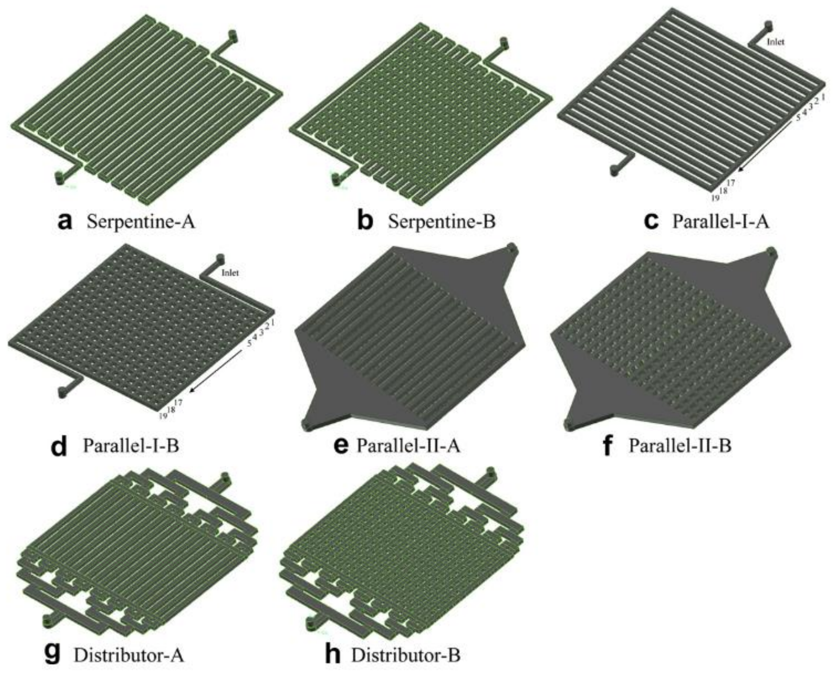

Fig. 1 – Some common flow field configurations for heat sinks.

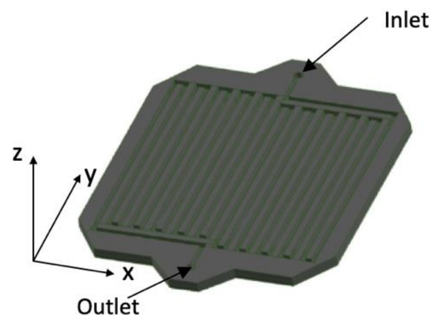

Fig. 2 – 3D image of the serpentine-A configuration

Verification Sources

Two aspects can be expected to be verified using the available numerical/analytical results:

Entrance Flow. Entrance flow is a very common phenomenon happening in pipe flows. Boundary layer starts from the entrance and its width grows along the flow direction, which results in a varying velocity profile in the entrance flow section. When the boundary layers developed from walls at different direction meets together, the entrance section ends and the flow is considered to be fully developed. The entrance section length calculated from the CFD simulation in this project can be compared with typical equation for Laminar entrance flow:

$$\frac{l_e}{D} = 0.06\, Re$$

Temperature Distribution. The steady state temperature distribution of Serpentine-A configuration is provide in [1] (see **Fig. 3**). The temperature result calculated from the CFD simulation in this project can be compared with **Fig. 3**. The trend of the temperature distribution should be similar, while some difference is expected due to: (1) the problem is simplified from 3D to 2D; and (2) the temperature result would be on the internal plane instead of the heating surface.

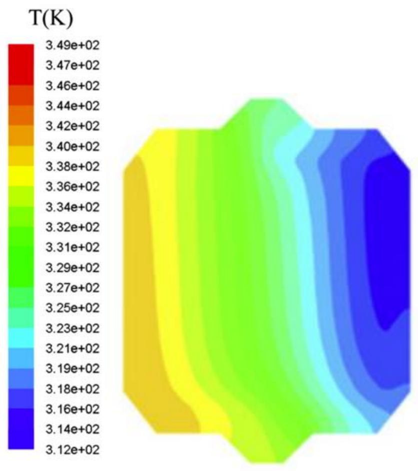

Fig. 3 – Temperature contour on heating surface for heat sinks with Serpentine-A configuration (Re=500, q''=41.5kW/m^2).

Physical Model and Numerical Method

1. Governing Equations

The flow/fluid in this project is assumed to be:

- Incompressible
- Constant viscosity
- Laminar

$$div\vec{V} = 0 \quad (1)$$

$$\frac{\partial \vec{V}}{\partial t} + div\left(u_j \vec{V}\right) = -\frac{\partial \pi}{\partial x_j} + \nu_f div(grad(u_j)) \quad (2)$$

$$\frac{\partial T}{\partial t} + div\left(\vec{V} \cdot T\right) = \frac{k}{\rho C_p} div(grad(T)) \quad (3)$$

Eqs. (1)–(3) can be explicitly written as:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \quad (4)$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial \pi}{\partial x} + \nu_f\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \quad (5)$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial \pi}{\partial y} + \nu_f\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \quad (6)$$

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y} = \frac{k}{\rho C_p}\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) \quad (7)$$

2. Numerical Scheme

In this project, the numerical method used is finite volume method (FVM). Pressure correction method is applied to satisfy Eq. (4) in each numerical time step. Staggered mesh is applied, as shown in Fig. 4. Details can be found in Lecture 10-Methods for Incompressible N-S Equation.

In each step, the velocity will be first calculated according to the convective term (F_c), viscous term (F_d) and the pressure term (Q_p), according to Eq. 8:

$$\frac{\tilde{U}^{n+1} - U^n}{\Delta t} = F_c + F_d + Q_p \quad (8)$$

Then the pressure correction term is calculated as:

$$\frac{p'_{i+1,j} - 2p'_{i,j} + p'_{i-1,j}}{{h_x}^2} + \frac{p'_{i1,j+1} - 2p'_{i,j} + p'_{i,j-1}}{{h_y}^2} = \frac{\rho}{h_x h_y \Delta t}\Delta\dot{\tilde{m}}_{i,j} \quad (9)$$

where $\Delta\dot{\tilde{m}}_{i,j}$ is the numerical mass rate difference on a cell.

The velocity correction will be calculated as:

$$u' = -\frac{\Delta t}{\rho}\nabla p' \quad (10)$$

Finally the velocity will be corrected by using this correction term.

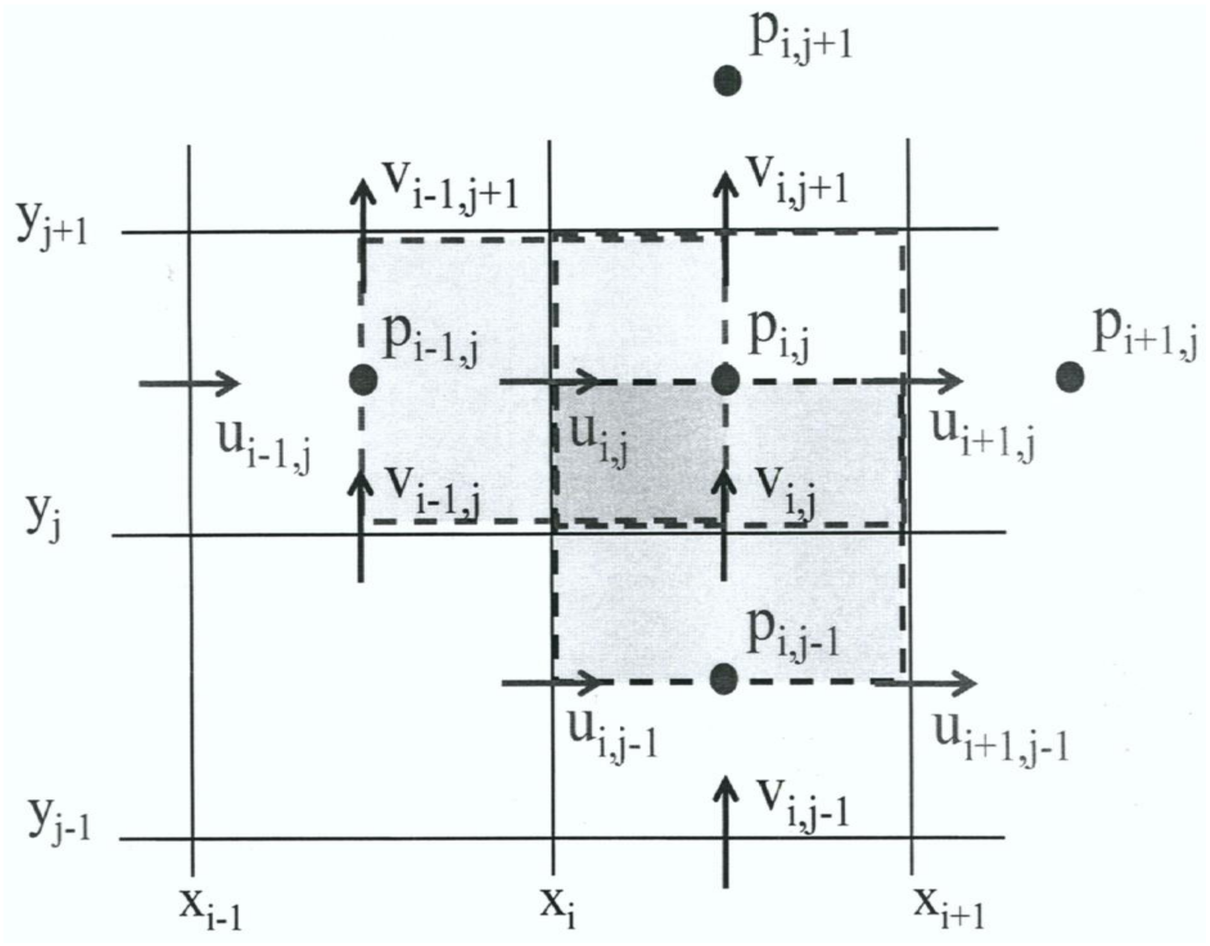

Fig. 4 – Staggered mesh.

Table 1. Parameters used in the simulation

Parameter	Value	Unit
Liquid density	1000	kg/m^3
Kinematic viscosity	8e-7	m^2/s
Atmospheric pressure	0	Pa

Simulation Results

1. Straight Channel

Since the geometry considered is a quite complex problem, I started using a simple straight channel. The result for the flow development in the straight channel is quite mature, and the development length equation of channel flow will be used for validation.

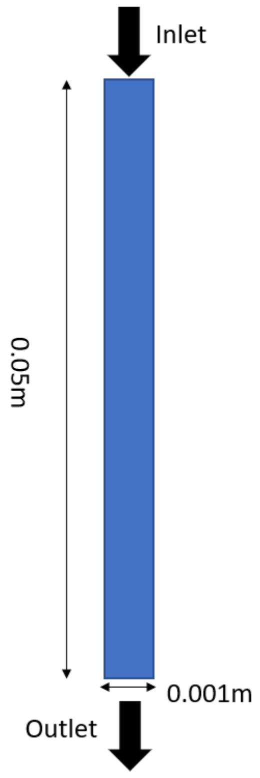

Fig. 5 – Straight channel.

In Fig. 6, the development of the boundary layer can easily be seen. The vertical velocity profile at different locations of the channel is also shown in Fig. 7. The development length of the channel flow is calculated from the simulation, and the flow is considered to be fully developed when the change of maximum v between two neighbor locations is less than 0.01%. The result is summarized in Table 2, and the simulation results are quite consistent with the results calculated using the equation for laminar channel flow.

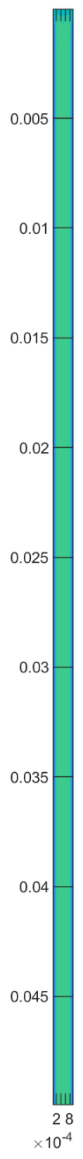	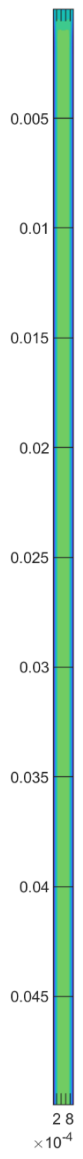	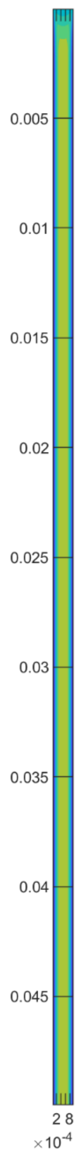	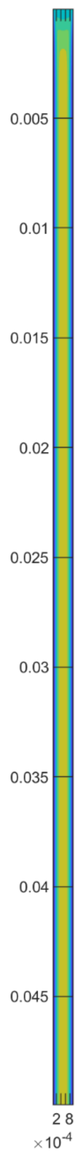	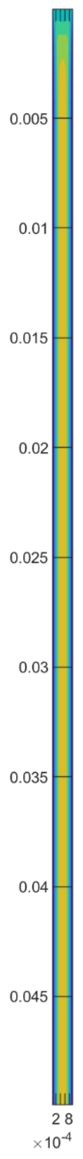	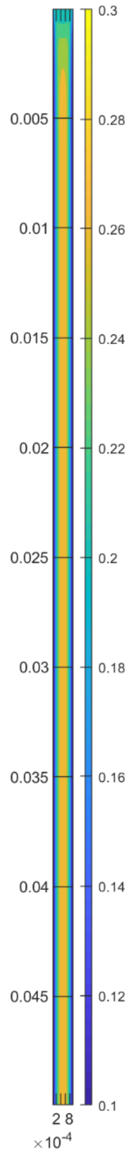
0.005s	0.01s	0.015s	0.02s	0.025s	0.03s

Fig. 6 – Color plot for vertical velocity at different times.

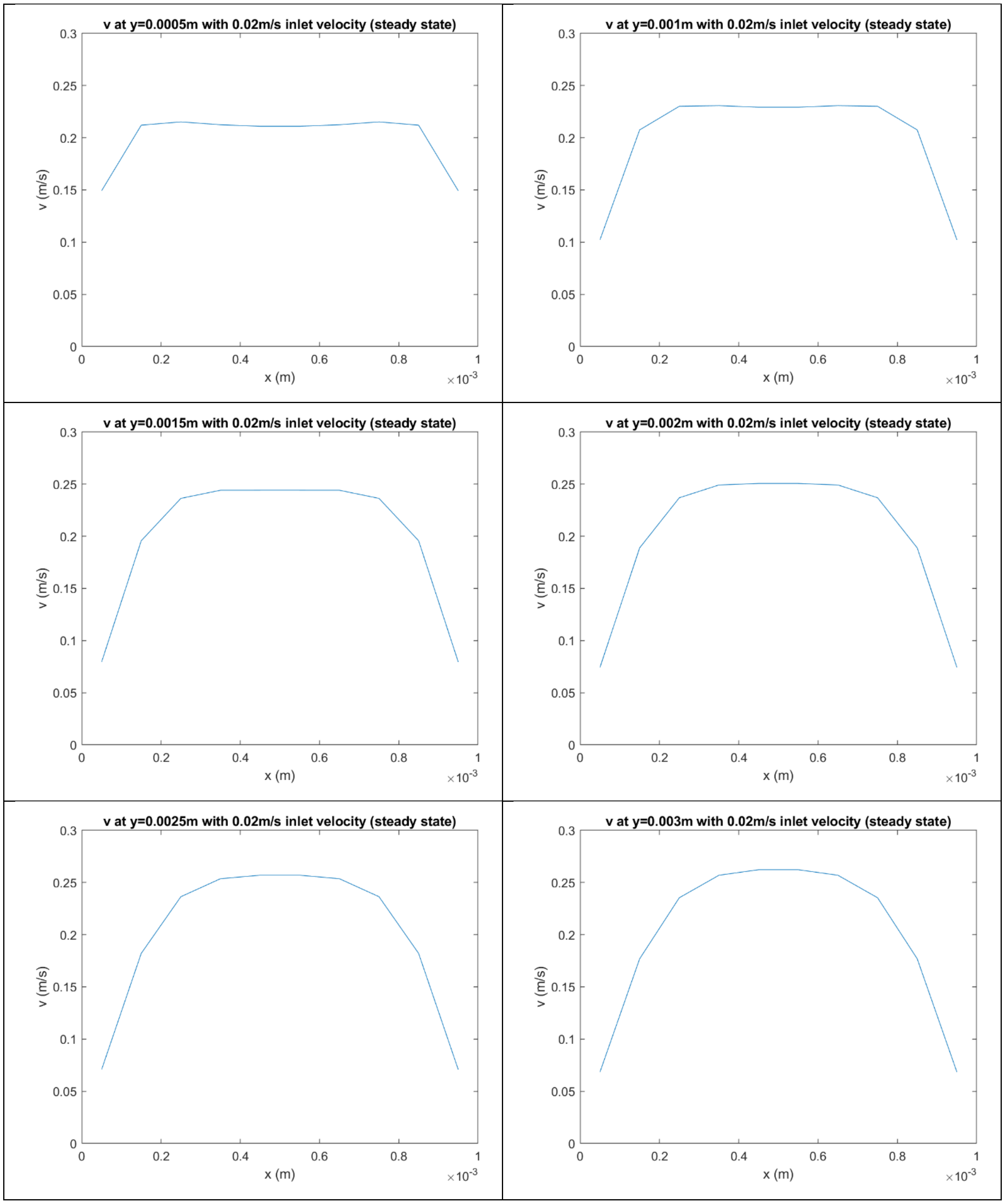

Fig. 7 – Vertical velocity vs. x at different y values

Table 2. Summary of development length at different inlet velocities.

Inlet velocity	Development Length Calculated Using Equation $\frac{l_e}{D} = 0.06\,Re$	Development Length from Simulation
m/s	m	m
0.02	0.0015	0.003
0.05	0.00375	0.0056
0.1	0.0075	0.0094
0.2	0.015	0.0148

2. Turning channels

Due to the high computational cost with the complex geometry shown in Fig. 2, I decided to use the geometry in Fig. 8 as the simulation case.

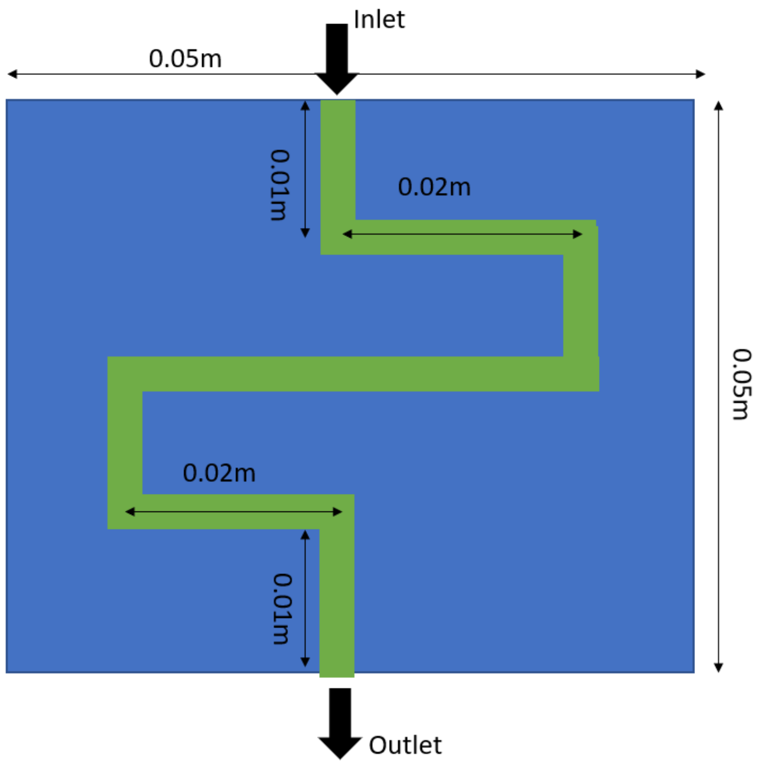

Fig. 8 – Turning channel.

Fig. 9 shows the velocity (root square of u^2+v^2) at 0.01s of the channel flow. It can be seen that the result captures the flow behavior at the turning points.

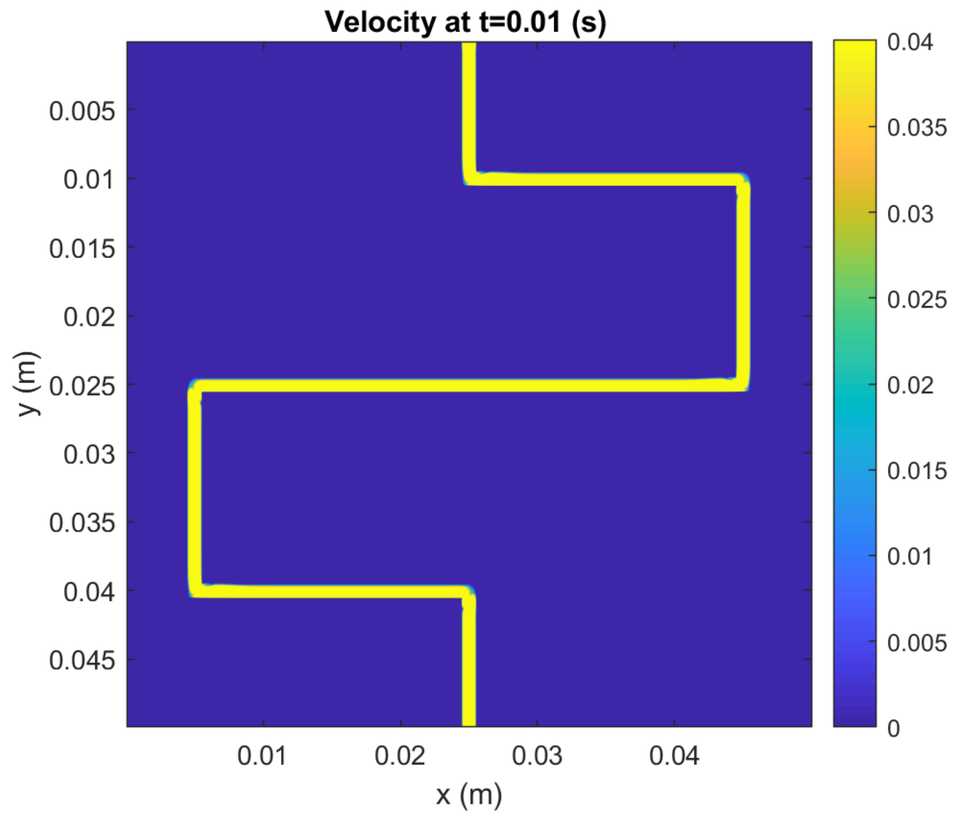

Fig. 9. Velocity color plot at steady state.

Fig. 10 and Fig. 11 show the velocity vector plot of the steady state results. It can be seen that the flow behavior at the turning points is captured in Fig. 11.

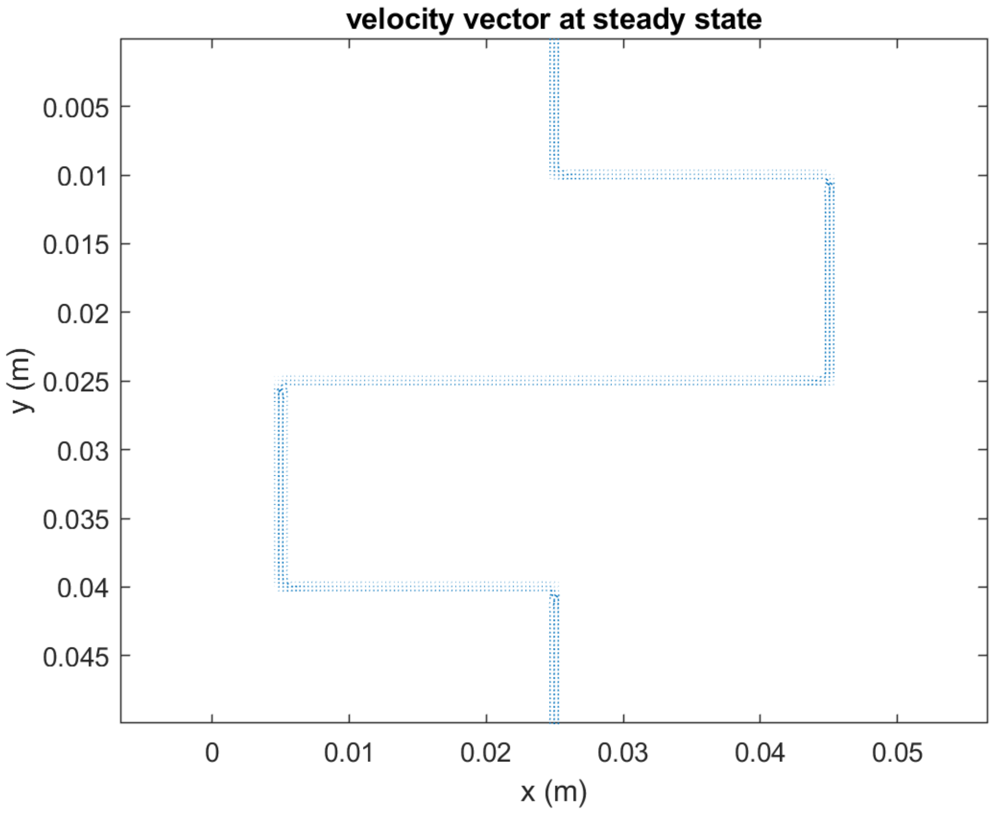

Fig. 10. Velocity vector at steady state.

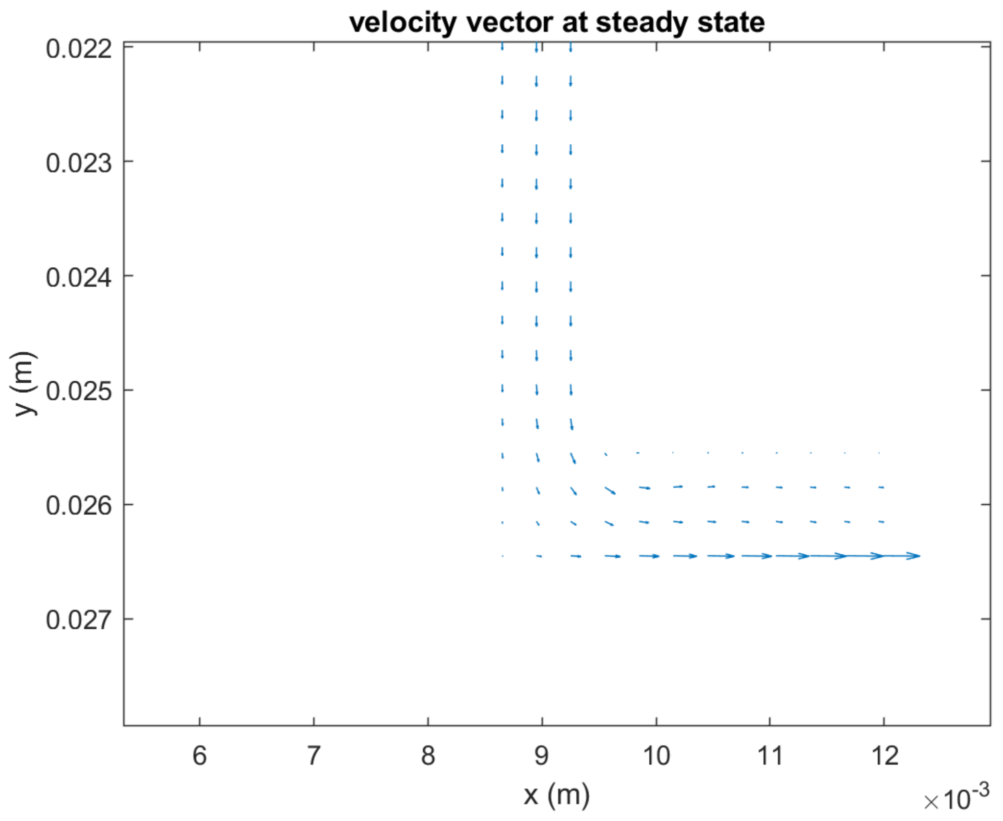

Fig. 11. Zoom in of Fig. 10.

3. Temperature Effect

Due to time limitation, the energy equation was not implemented into the simulation. However, since I have got the results and model with complex geometry, and the energy equation is quite decoupled from the mass and momentum equations (fluid properties independent of temperature), it should be expected that the energy equation can be easily added into the model to calculate the temperature distribution of the chip.

Summary

This project calculates the flow in a channel with complex geometry and the result shows good consistence with well-known equations and physical expectations.

Timeline

Period	Task
10.25 – 11.08	Finalize the literature; Complete the equations and boundary conditions of the 2D problem
11.09 – 11.22	Discretize the equations; Build the mesh; Implement the equations to the mesh
11.23 – 12.07	Debug; Generate results; Write the report
12.08 – 12.18	Finalize the results and the report

Reference

[1] B. Ramos-Alvarado, P. Li, H. Liu, and A. Hernandez-Guerrero, “CFD study of liquid-cooled heat sinks with microchannel flow field configurations for electronics, fuel cells, and concentrated solar cells,” *Appl. Therm. Eng.*, vol. 31, no. 14–15, pp. 2494–2507, 2011.

[2] M. B. Dogruoz, M. Urdaneta, and A. Ortega, “Experiments and modeling of the hydraulic resistance and heat transfer of in-line square pin fin heat sinks with top by-pass flow,” *Int. J. Heat Mass Transf.*, vol. 48, no. 23–24, pp. 5058–5071, Nov. 2005.

[3] R. Akhilesh, A. Narasimhan, and C. Balaji, “Method to improve geometry for heat transfer enhancement in PCM composite heat sinks,” *Int. J. Heat Mass Transf.*, vol. 48, no. 13, pp. 2759–2770, Jun. 2005.

[4] M. Bahiraei, S. Heshmatian, M. Goodarzi, and H. Moayedi, “CFD analysis of employing a novel ecofriendly nanofluid in a miniature pin fin heat sink for cooling of electronic components: Effect of different configurations,” *Adv. Powder Technol.*, vol. 30, no. 11, pp. 2503–2516, 2019.

[5] J. Wang, “Pressure drop and flow distribution in parallel-channel configurations of fuel cells: Z-type arrangement,” *Int. J. Hydrogen Energy*, vol. 35, no. 11, pp. 5498–5509, 2010.

[6] C. J. Kroeker, H. M. Soliman, and S. J. Ormiston, “Three-dimensional thermal analysis of heat sinks with circular cooling micro-channels,” *Int. J. Heat Mass Transf.*, vol. 47, no. 22, pp. 4733–4744, 2004.

[7] K. . Toh, X. . Chen, and J. . Chai, “Numerical computation of fluid flow and heat transfer in microchannels,” *Int. J. Heat Mass Transf.*, vol. 45, no. 26, pp. 5133–5141, Dec. 2002.

Source Code

```
% FVM, staggered mesh, simple channel without turning point
%P: time-invariant parameter
%Pt: time-variant parameter
clear all

%% inputs
%lengths
P.w=.05; %width,0.0009m
P.h=0.05; % length
P.depth_global = 0.0012; %depth of the plate
P.w_channel=0.001;
P.length1=0.01;
P.length2=0.02;
```

```matlab
v_options = [0.02 0.05 0.1 0.2 0.4 0.6];

%mass inlets
P.u_in=0; %no x component
for i = 1:length(v_options)
P.v_in=v_options(i);%0.03m/s

P.m_in=P.v_in*P.depth_global*P.w_channel;%total mass in


%fluid properties
P.rho=1000; %density
P.nu=0.8e-6; %kinematic viscosity
P.p_0=0; %atmospheric pressure

%% discretization
P.m_channel=10; %number of cells in the channel wedth
P.dx = P.w_channel/P.m_channel;
P.m = P.w/P.dx;
P.n=round(P.m*P.h/P.w); %number of cells in a column
P.dy=P.h/P.n;
P.dt=0.01 * min((min(P.dx,P.dy))^2/P.nu , min(P.dx,P.dy)/P.u_in);
P.dt=round(P.dt,4);
P.t_end=1;

%sigma
P.sigma_nu=P.dt/((min(P.dx,P.dy))^2/P.nu);
%check stability
if max(P.sigma_nu)>0.1
    disp('dt too high (sigma)');
end
%to check steady state
P.epsilon_u=1e-6;
P.epsilon_v=1e-6;
P.epsilon_p=1e-6;



[Pt,P] = Initialize_parameter(P);

saving.u_save=zeros(size(Pt.u));
saving.v_save=zeros(size(Pt.v));
saving.p_save=zeros(size(Pt.pi));
saving.t_save=0;
Pt = Update(P,Pt,saving);

end
%% Initialize the parameters
function [Pt,P] = Initialize_parameter(P)
    %unknowns
    Pt.pi=P.p_0/P.rho*ones(P.n,P.m); %pressure
    Pt.u=zeros(P.n,P.m+1); %u-velocity
    Pt.du=zeros(size(Pt.u));
    Pt.v=zeros(P.n+1,P.m); %v-velocity
    Pt.dv=zeros(size(Pt.v));
    Pt.p_prime=zeros(size(Pt.pi));
    Pt.u_prime=zeros(size(Pt.u));
    Pt.v_prime=zeros(size(Pt.v));

    %calculate other lengths
    P.length3=P.h/2-P.length1;

    %determine the location of the six points
    P.Point1_loc = [P.w/2,P.length1];
    P.Point2_loc = [P.w/2 + P.length2,P.length1];
    P.Point3_loc = [P.w/2 + P.length2,P.h/2];
    P.Point4_loc = [P.w/2 - P.length2,P.h/2];
    P.Point5_loc = [P.w/2 - P.length2,P.h-P.length1];
    P.Point6_loc = [P.w/2,P.h-P.length1];

    idx_x1 = round((P.w/2-P.length2-P.w_channel/2)/P.dx);
    idx_x2 = round((P.w/2-P.length2+P.w_channel/2)/P.dx);
    idx_x3 = round((P.w/2-P.w_channel/2)/P.dx);
    idx_x4 = round((P.w/2+P.w_channel/2)/P.dx);
    idx_x5 = round((P.w/2+P.length2-P.w_channel/2)/P.dx);
    idx_x6 = round((P.w/2+P.length2+P.w_channel/2)/P.dx);

    idx_y1 = round((P.length1-P.w_channel/2)/P.dy);
    idx_y2 = round((P.length1+P.w_channel/2)/P.dy);
    idx_y3 = round((P.h/2-P.w_channel/2)/P.dy);
    idx_y4 = round((P.h/2+P.w_channel/2)/P.dy);
    idx_y5 = round((P.h-P.length1-P.w_channel/2)/P.dy);
    idx_y6 = round((P.h-P.length1+P.w_channel/2)/P.dy);

    %% determine type of cell (mask)
    %solid
    P.cell_p = zeros(size(Pt.pi));
    P.cell_p(1:idx_y2,1:idx_y3) = NaN;
    P.cell_p(idx_y2+1:idx_y3,1:idx_x5) = NaN;
    P.cell_p(idx_y3+1:idx_y6,1:idx_x1) = NaN;
    P.cell_p(idx_y6+1:end,1:idx_x3) = NaN;
    P.cell_p(1:idx_y1,idx_x4+1:end) = NaN;
    P.cell_p(idx_y1+1:idx_y4,idx_x6+1:end) = NaN;
    P.cell_p(idx_y4+1:idx_y5,idx_x2+1:end) = NaN;
    P.cell_p(idx_y5+1:end,idx_x4+1:end) = NaN;

    P.cell_p(idx_y1+1,idx_x4+1:idx_x6) = 1; %North wall
    P.cell_p(idx_y3+1,idx_x1+1:idx_x5) = 1; %North wall
    P.cell_p(idx_y5+1,idx_x2+1:idx_x4) = 1; %North wall

    P.cell_p(1:idx_y2,idx_x3+1) = 2; %West wall
    P.cell_p(idx_y2+1:idx_y3,idx_x5+1) = 2; %West wall
    P.cell_p(idx_y3+1:idx_y6,idx_x1+1) = 2; %West wall
    P.cell_p(idx_y6+1:end,idx_x3+1) = 2; %West wall

    P.cell_p(idx_y2,idx_x3+1:idx_x5) = 3; %South wall
    P.cell_p(idx_y4,idx_x2+1:idx_x6) = 3; %South wall
    P.cell_p(idx_y6,idx_x1+1:idx_x3) = 3; %South wall

    P.cell_p(1:idx_y1,idx_x4) = 4; %East wall
    P.cell_p(idx_y1+1:idx_y4,idx_x6) = 4; %East wall
```

```
P.cell_p(idx_y4+1:idx_y5,idx_x2) = 4; %East wall
P.cell_p(idx_y5+1:end,idx_x4) = 4; %East wall

P.cell_p(1,idx_x3+1:idx_x4) = 9; %Inlet face
P.cell_p(end,idx_x3+1:idx_x4) = 10; %Outlet face
P.cell_p(idx_y3+1,idx_x1+1) = 5;%North-west corner
P.cell_p(idx_y2,idx_x3+1) = 6;%South-west corner
P.cell_p(idx_y6,idx_x1+1) = 6;%South-west corner
P.cell_p(idx_y4,idx_x6) = 7;%South-east corner
P.cell_p(idx_y1+1,idx_x6) = 8;%North-east corner
P.cell_p(idx_y5+1,idx_x4) = 8;%North-east corner
P.cell_p(1,idx_x3+1) = 11;%North-west corner at inlet
P.cell_p(end,idx_x3+1) = 12;%South-west corner at outlet
P.cell_p(end,idx_x4) = 13;%South-east corner at outlet
P.cell_p(1,idx_x4) = 14;%North-east corner at inlet

%solid
P.cell_u = zeros(size(Pt.u));
P.cell_u(1:idx_y2,1:idx_y3) = NaN;
P.cell_u(idx_y2+1:idx_y3,1:idx_x5) = NaN;
P.cell_u(idx_y3+1:idx_y6,1:idx_x1) = NaN;
P.cell_u(idx_y6+1:end,1:idx_x3) = NaN;
P.cell_u(1:idx_y1,idx_x4+2:end) = NaN;
P.cell_u(idx_y1+1:idx_y4,idx_x6+2:end) = NaN;
P.cell_u(idx_y4+1:idx_y5,idx_x2+2:end) = NaN;
P.cell_u(idx_y5+1:end,idx_x4+2:end) = NaN;

P.cell_u(idx_y1+1,idx_x4+2:idx_x6+1) = 1; %North wall
P.cell_u(idx_y3+1,idx_x1+2:idx_x5) = 1; %North wall
P.cell_u(idx_y5+1,idx_x2+2:idx_x4+1) = 1; %North wall

P.cell_u(1:idx_y2,idx_x3+1) = 2; %West wall
P.cell_u(idx_y2+1:idx_y3,idx_x5+1) = 2; %West wall
P.cell_u(idx_y3+1:idx_y6,idx_x1+1) = 2; %West wall
P.cell_u(idx_y6+1:end,idx_x3+1) = 2; %West wall

P.cell_u(idx_y2,idx_x3+2:idx_x5) = 3; %South wall
P.cell_u(idx_y4,idx_x2+2:idx_x6) = 3; %South wall
P.cell_u(idx_y6,idx_x1+2:idx_x3) = 3; %South wall

P.cell_u(1:idx_y1,idx_x4+1) = 4; %East wall
P.cell_u(idx_y1+1:idx_y4,idx_x6+1) = 4; %East wall
P.cell_u(idx_y4+1:idx_y5,idx_x2+1) = 4; %East wall
P.cell_u(idx_y5+1:end,idx_x4+1) = 4; %East wall

P.cell_u(1,idx_x3+1:idx_x4) = 9; %Inlet face
P.cell_u(end,idx_x3+1:idx_x4) = 10; %Outlet face
P.cell_u(idx_y3+1,idx_x1+1) = 5;%North-west corner
P.cell_u(idx_y2,idx_x3+1) = 6;%South-west corner
P.cell_u(idx_y6,idx_x1+1) = 6;%South-west corner
P.cell_u(idx_y4,idx_x6+1) = 7;%South-east corner
P.cell_u(idx_y1+1,idx_x6+1) = 8;%North-east corner
P.cell_u(idx_y5+1,idx_x4+1) = 8;%North-east corner
P.cell_u(1,idx_x3+1) = 11;%North-west corner at inlet
P.cell_u(end,idx_x3+1) = 12;%South-west corner at outlet
```

```
P.cell_u(end,idx_x4+1) = 13;%South-east corner at outlet
P.cell_u(1,idx_x4+1) = 14;%North-east corner at inlet

%solid
P.cell_v = zeros(size(Pt.v));
P.cell_v(1:idx_y2+1,1:idx_y3) = NaN;
P.cell_v(idx_y2+2:idx_y3+1,1:idx_x5) = NaN;
P.cell_v(idx_y3+2:idx_y6+1,1:idx_x1) = NaN;
P.cell_v(idx_y6+2:end,1:idx_x3) = NaN;
P.cell_v(1:idx_y1,idx_x4+1:end) = NaN;
P.cell_v(idx_y1+1:idx_y4+1,idx_x6+1:end) = NaN;
P.cell_v(idx_y4+2:idx_y5,idx_x2+1:end) = NaN;
P.cell_v(idx_y5+1:end,idx_x4+1:end) = NaN;

P.cell_v(idx_y1+1,idx_x4+1:idx_x6) = 1; %North wall
P.cell_v(idx_y3+1,idx_x1+1:idx_x5) = 1; %North wall
P.cell_v(idx_y5+1,idx_x2+1:idx_x4) = 1; %North wall

P.cell_v(1:idx_y2,idx_x3+1) = 2; %West wall
P.cell_v(idx_y2+2:idx_y3,idx_x5+1) = 2; %West wall
P.cell_v(idx_y3+1:idx_y6,idx_x1+1) = 2; %West wall
P.cell_v(idx_y6+2:end,idx_x3+1) = 2; %West wall

P.cell_v(idx_y2+1,idx_x3+1:idx_x5) = 3; %South wall
P.cell_v(idx_y4+1,idx_x2+1:idx_x6) = 3; %South wall
P.cell_v(idx_y6+1,idx_x1+1:idx_x3) = 3; %South wall

P.cell_v(1:idx_y1,idx_x4) = 4; %East wall
P.cell_v(idx_y1+1:idx_y4,idx_x6) = 4; %East wall
P.cell_v(idx_y4+2:idx_y5,idx_x2) = 4; %East wall
P.cell_v(idx_y5+1:end,idx_x4) = 4; %East wall

P.cell_v(1,idx_x3+1:idx_x4) = 9; %Inlet face
P.cell_v(end,idx_x3+1:idx_x4) = 10; %Outlet face
P.cell_v(idx_y3+1,idx_x1+1) = 5;%North-west corner
P.cell_v(idx_y2+1,idx_x3+1) = 6;%South-west corner
P.cell_v(idx_y6+1,idx_x1+1) = 6;%South-west corner
P.cell_v(idx_y4+1,idx_x6) = 7;%South-east corner
P.cell_v(idx_y1+1,idx_x6) = 8;%North-east corner
P.cell_v(idx_y5+1,idx_x4) = 8;%North-east corner
P.cell_v(1,idx_x3+1) = 11;%North-west corner at inlet
P.cell_v(end,idx_x3+1) = 12;%South-west corner at outlet
P.cell_v(end,idx_x4) = 13;%South-east corner at outlet
P.cell_v(1,idx_x4) = 14;%North-east corner at inlet


%%For Pressure Correction
%index cells with flow
P.cell_number_p=zeros(size(Pt.pi));

%numbering p_cells
k=0;
for j=1:P.n
    for i=1:P.m
        if ~isnan(P.cell_p(j,i))
```

```
                P.cell_number_p(j,i)=k+1;
                k=k+1;
            end
        end
    end

    P.A=zeros(k);
    P.b=zeros(k,1);

    %setting A matrix for P_prime solving (constant)
    for j=1:P.n
        for i=1:P.m
            switch P.cell_p(j,i)
                case 0 %general
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i) ) = -
(2/P.dx^2+2/P.dy^2);
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i+1) ) =
1/P.dx^2;
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i-1) ) =
1/P.dx^2;
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j+1,i) ) =
1/P.dy^2;
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j-1,i) ) =
1/P.dy^2;
                case 1 %North wall
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i) ) = -
(2/P.dx^2+1/P.dy^2);
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i+1) ) =
1/P.dx^2;
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i-1) ) =
1/P.dx^2;
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j+1,i) ) =
1/P.dy^2;
                case 2 %West wall
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i) ) = -
(1/P.dx^2+2/P.dy^2);
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i+1) ) =
1/P.dx^2;
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j+1,i) ) =
1/P.dy^2;
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j-1,i) ) =
1/P.dy^2;
                case 3 %South wall
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i) ) = -
(2/P.dx^2+1/P.dy^2);
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i+1) ) =
1/P.dx^2;
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i-1) ) =
1/P.dx^2;
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j-1,i) ) =
1/P.dy^2;
                case 4 %East wall
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i) ) = -
(1/P.dx^2+2/P.dy^2);
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i-1) ) =
1/P.dx^2;
```

```
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j+1,i) ) =
1/P.dy^2;
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j-1,i) ) =
1/P.dy^2;
                case 9 %Inlet face (north)
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i) ) = -
(2/P.dx^2+1/P.dy^2);
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i+1) ) =
1/P.dx^2;
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i-1) ) =
1/P.dx^2;
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j+1,i) ) =
1/P.dy^2;
                case 10 %Outlet face (south)
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i) ) = -
(2/P.dx^2+1/P.dy^2);
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i+1) ) =
1/P.dx^2;
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i-1) ) =
1/P.dx^2;
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j-1,i) ) =
1/P.dy^2;
                case num2cell([5,11]) %North-west corner at inlet
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i) ) = -
(1/P.dx^2+1/P.dy^2);
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i+1) ) =
1/P.dx^2;
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j+1,i) ) =
1/P.dy^2;
                case num2cell([6,12]) %South-west corner at outlet
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i) ) = -
(1/P.dx^2+1/P.dy^2);
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i+1) ) =
1/P.dx^2;
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j-1,i) ) =
1/P.dy^2;
                case num2cell([7,13]) %South-east corner at outlet
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i) ) = -
(1/P.dx^2+1/P.dy^2);
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i-1) ) =
1/P.dx^2;
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j-1,i) ) =
1/P.dy^2;
                case num2cell([8,14]) %North-east corner at inlet
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i) ) = -
(1/P.dx^2+1/P.dy^2);
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j,i-1) ) =
1/P.dx^2;
                    P.A( P.cell_number_p(j,i) , P.cell_number_p(j+1,i) ) =
1/P.dy^2;
            end
        end
    end
    P.A = P.A(1:end-1,1:end-1); %since last P doesn't need correction

    %% IC
    Pt.v(P.cell_v==9)=P.v_in;
    Pt.v(P.cell_v==11)=P.v_in;
    Pt.v(P.cell_v==14)=P.v_in;
end

function Pt = Update(P,Pt,saving)
    for t=P.dt:P.dt:P.t_end
        %% u momentum
        Pt.du=zeros(size(Pt.u));
        for i=1:P.m+1 %ith column
            for j=1:P.n %jth row
                if ~isnan(P.cell_u(j,i))
                    Fc_u=0; Fd_u=0; Qp_u=0;
                    switch P.cell_u(j,i)
                        case 0 %general flow
                            Fc_u = P.dy/4*( (Pt.u(j,i+1)+Pt.u(j,i)) *
(Pt.u(j,i+1)+Pt.u(j,i)) - (Pt.u(j,i-1)+Pt.u(j,i)) * (Pt.u(j,i-
1)+Pt.u(j,i)) ) ...
                                +P.dx/4*( (Pt.v(j+1,i-1)+Pt.v(j+1,i)) *
(Pt.u(j+1,i)+Pt.u(j,i)) - (Pt.v(j,i-1)+Pt.v(j,i)) * (Pt.u(j-
1,i)+Pt.u(j,i)) );
                            Fd_u = P.nu*( P.dy/P.dx*( (Pt.u(j,i+1)-Pt.u(j,i))
- (Pt.u(j,i)-Pt.u(j,i-1)) ) + P.dx/P.dy*( (Pt.u(j+1,i)-Pt.u(j,i)) -
(Pt.u(j,i)-Pt.u(j-1,i)) ) );
                            Qp_u = P.dy*(Pt.pi(j,i)-Pt.pi(j,i-1));
                        case 1 %North wall
                            Fc_u = P.dy/4*( (Pt.u(j,i+1)+Pt.u(j,i)) *
(Pt.u(j,i+1)+Pt.u(j,i)) - (Pt.u(j,i-1)+Pt.u(j,i)) * (Pt.u(j,i-
1)+Pt.u(j,i)) ) ...
                                +P.dx/4*( (Pt.v(j+1,i-1)+Pt.v(j+1,i)) *
(Pt.u(j+1,i)+Pt.u(j,i)) - 0 );
                            Fd_u = P.nu*( P.dy/P.dx*( (Pt.u(j,i+1)-Pt.u(j,i))
- (Pt.u(j,i)-Pt.u(j,i-1)) ) + P.dx/P.dy*( (Pt.u(j+1,i)-Pt.u(j,i)) -
1/3*(9*Pt.u(j,i)-Pt.u(j+1,i)) ));
                            Qp_u = P.dy*(Pt.pi(j,i)-Pt.pi(j,i-1));
                        case 3 %South wall
                            Fc_u = P.dy/4*( (Pt.u(j,i+1)+Pt.u(j,i)) *
(Pt.u(j,i+1)+Pt.u(j,i)) - (Pt.u(j,i-1)+Pt.u(j,i)) * (Pt.u(j,i-
1)+Pt.u(j,i)) ) ...
                                +P.dx/4*( 0 - (Pt.v(j,i-1)+Pt.v(j,i)) *
(Pt.u(j-1,i)+Pt.u(j,i)) );
                            Fd_u = P.nu*( P.dy/P.dx*( (Pt.u(j,i+1)-Pt.u(j,i))
- (Pt.u(j,i)-Pt.u(j,i-1)) ) + P.dx/P.dy*( ( 1/3*(9*Pt.u(j,i)-Pt.u(j-1,i)) -
(Pt.u(j,i)-Pt.u(j-1,i)))));
                            Qp_u = P.dy*(Pt.pi(j,i)-Pt.pi(j,i-1));
                        case 9 %Inlet face
                            Fc_u = P.dy/4*( (Pt.u(j,i+1)+Pt.u(j,i)) *
(Pt.u(j,i+1)+Pt.u(j,i)) - (Pt.u(j,i-1)+Pt.u(j,i)) * (Pt.u(j,i-
1)+Pt.u(j,i)) ) ...
                                +P.dx/4*( (Pt.v(j+1,i-1)+Pt.v(j+1,i)) *
(Pt.u(j+1,i)+Pt.u(j,i))-(4*P.v_in*P.u_in) );
                            Fd_u = P.nu*( P.dy/P.dx*( (Pt.u(j,i+1)-Pt.u(j,i))
- (Pt.u(j,i)-Pt.u(j,i-1)) ) + P.dx/P.dy*( 1/3*(8*P.u_in-
9*Pt.u(j,i)+Pt.u(j+1,i)) - (Pt.u(j,i)-Pt.u(j+1,i)) ) );
                            Qp_u = P.dy*(Pt.pi(j,i)-Pt.pi(j,i-1));
                        case 10 %Outlet face
```

```
                            Fc_u = P.dy/4*( (Pt.u(j,i+1)+Pt.u(j,i)) *
(Pt.u(j,i+1)+Pt.u(j,i)) - (Pt.u(j,i-1)+Pt.u(j,i)) * (Pt.u(j,i-
1)+Pt.u(j,i)) ) ...
                                +P.dx/4*( (Pt.v(j+1,i-1)+Pt.v(j+1,i)) *
(2*Pt.u(j,i)) - (Pt.v(j,i-1)+Pt.v(j,i)) * (Pt.u(j-1,i)+Pt.u(j,i)) );
                            Fd_u = P.nu*( P.dy/P.dx*( (Pt.u(j,i+1)-Pt.u(j,i))
- (Pt.u(j,i)-Pt.u(j,i-1)) ) + P.dx/P.dy*( 0 - (Pt.u(j,i)-Pt.u(j-1,i)) ) );
                            Qp_u = P.dy*(Pt.pi(j,i)-Pt.pi(j,i-1));
                    end
                    Pt.du(j,i)=-Fc_u+Fd_u-Qp_u;
                end
            end
        end

        %% v momentum
        Pt.dv=zeros(size(Pt.v));
        for i=1:P.m
            for j=1:P.n+1
                if ~isnan(P.cell_v(j,i))
                    Fc_v=0; Fd_v=0; Qp_v=0;
                    switch P.cell_v(j,i)
                        case 0 %general
                            Fc_v = P.dy/4*( (Pt.u(j,i+1)+Pt.u(j-1,i+1)) *
(Pt.v(j,i+1)+Pt.v(j,i)) - (Pt.u(j,i)+Pt.u(j-1,i)) * (Pt.v(j,i-
1)+Pt.v(j,i)) )...
                                +P.dx/4*( (Pt.v(j+1,i)+Pt.v(j,i)) *
(Pt.v(j+1,i)+Pt.v(j,i)) - (Pt.v(j-1,i)+Pt.v(j,i)) * (Pt.v(j-
1,i)+Pt.v(j,i)) );
                            Fd_v = P.nu*( P.dy/P.dx*( (Pt.v(j,i+1)-Pt.v(j,i))
- (Pt.v(j,i)-Pt.v(j,i-1)) ) + P.dx/P.dy*( (Pt.v(j+1,i)-Pt.v(j,i)) -
(Pt.v(j,i)-Pt.v(j-1,i)) ) );
                            Qp_v=P.dx*(Pt.pi(j,i)-Pt.pi(j-1,i));
                        case 2 %West wall
                            Fc_v = P.dy/4*( (Pt.u(j,i+1)+Pt.u(j-1,i+1)) *
(Pt.v(j,i+1)+Pt.v(j,i)) - 0 )...
                                +P.dx/4*( (Pt.v(j+1,i)+Pt.v(j,i)) *
(Pt.v(j+1,i)+Pt.v(j,i)) - (Pt.v(j-1,i)+Pt.v(j,i)) * (Pt.v(j-
1,i)+Pt.v(j,i)) );
                            Fd_v = P.nu*( P.dy/P.dx*( (Pt.v(j,i+1)-Pt.v(j,i))
- 1/3*(9*Pt.v(j,i)-Pt.v(j,i+1)) ) + P.dx/P.dy*( (Pt.v(j+1,i)-Pt.v(j,i)) -
(Pt.v(j,i)-Pt.v(j-1,i)) ) );
                            Qp_v=P.dx*(Pt.pi(j,i)-Pt.pi(j-1,i));
                        case 4 %East wall
                            Fc_v = P.dy/4*( 0 - (Pt.u(j,i)+Pt.u(j-1,i)) *
(Pt.v(j,i-1)+Pt.v(j,i)) )...
                                +P.dx/4*( (Pt.v(j+1,i)+Pt.v(j,i)) *
(Pt.v(j+1,i)+Pt.v(j,i)) - (Pt.v(j-1,i)+Pt.v(j,i)) * (Pt.v(j-
1,i)+Pt.v(j,i)) );
                            Fd_v = P.nu*( P.dy/P.dx*( 1/3*(-
9*Pt.v(j,i)+Pt.v(j,i-1)) - (Pt.v(j,i)-Pt.v(j,i-1)) ) +
P.dx/P.dy*( (Pt.v(j+1,i)-Pt.v(j,i)) - (Pt.v(j,i)-Pt.v(j-1,i)) ) );
                            Qp_v=P.dx*(Pt.pi(j,i)-Pt.pi(j-1,i));
                     end
                    Pt.dv(j,i)=-Fc_v+Fd_v-Qp_v;
                end
            end
        end
        %% update uncorrected u and v
        Pt.u=Pt.u+Pt.du*P.dt/P.dx/P.dy;
        Pt.v=Pt.v+Pt.dv*P.dt/P.dx/P.dy;

        %set BCs
        %u
        Pt.u(ismember(P.cell_u,[2,4,5:8,11:14])) = 0; %left/right walls and
conrners

        %v
        Pt.v(ismember(P.cell_v,[9,11,14])) = P.v_in; %inlet and corners
        Pt.v(ismember(P.cell_v,[1,3,5:8])) = 0; %lower/upper walls and
corners
        Pt.v(end,:) = Pt.v(end-1,:);%Outlet

        %check outlet

m_out=sum(Pt.v(ismember(P.cell_v,[10,12,13]))*P.depth_global*P.dx); %mass
outlet
        if m_out>1e-8

Pt.v(ismember(P.cell_v,[10,12,13]))=Pt.v(ismember(P.cell_v,[10,12,13]))*P.m_i
n/m_out; %set m_out to m_in (no mass source)
        else

Pt.v(ismember(P.cell_v,[10,12,13]))=Pt.v(ismember(P.cell_v,[10,12,13]))+P.m_i
n/(P.depth_global*P.w_channel); %if m_out=0

Pt.v(ismember(P.cell_v,[10,12,13]))=P.m_in/(P.depth_global*P.w_channel); %if
m_out=0
        end


        %% pressure correction
        %set up vector of b (based on delta_m_dot(j,i))
        for i=1:P.m
            for j=1:P.n
                if ~isnan(P.cell_p(j,i))
                    Pt.b(P.cell_number_p(j,i)) = ( (Pt.u(j,i+1)-
Pt.u(j,i))/P.dx + (Pt.v(j+1,i)-Pt.v(j,i))/P.dy ) *P.rho/P.dt;
                end
            end
        end

        %set p_prime(South East Corner) to P_0 (doesn't need correction)
        %doesn't need anything to do except reducing matrix size

        %size of A already reduced
        p_prime_vector=P.A\Pt.b(1:end-1)';
        p_prime_vector=[p_prime_vector;P.p_0];
        for i=1:P.m
            for j=1:P.n
                if P.cell_number_p(j,i)~=0
```

```
                    Pt.p_prime(j,i) = p_prime_vector(P.cell_number_p(j,i));
                end
            end
        end

        Pt.pi=Pt.pi+Pt.p_prime/P.rho;

        %% velocity correction
        for i=1:P.m
            for j=1:P.n
                %for middle region, low/upper walls, inlet, outlet
                if ismember(P.cell_u(j,i),[0,1,3,9,10])
                    Pt.u_prime(j,i) = -P.dt/P.rho/P.dx*( Pt.p_prime(j,i)-
Pt.p_prime(j,i-1) );
                end
                %for middle region, left/right walls
                if ismember(P.cell_v(j,i),[0,2,4])
                    Pt.v_prime(j,i) = -P.dt/P.rho/P.dy*( Pt.p_prime(j,i)-
Pt.p_prime(j-1,i) );
                end
            end
        end

        Pt.u=Pt.u+Pt.u_prime;
        Pt.v=Pt.v+Pt.v_prime;

        %% save parameters
        if mod(round(t,5),0.0001)==0
            t

%
%             [x,y] = ndgrid(0.5*P.dx:P.dx:P.w-0.5*P.dx,0:P.dy:P.h);
%             velocity = Pt.v;
%
% %             [x,y] = ndgrid(0:P.dx:P.w,0.5*P.dy:P.dy:P.h-0.5*P.dy);
% %             velocity = Pt.u;
%
%             pointsize = 10;
%             x=x';
%             y=y';
%
%             scatter(x(:),y(:), pointsize, velocity(:));
%             ax=gca;
%             ax.YDir='reverse';
%             colorbar
%             set(gcf, 'Units', 'normalized');
%             set(gcf, 'Position', [0 0 1 1]);
%             caxis([0 2*P.v_in])
%             hold on
%             drawnow()

            if mod(round(t,5),0.001)==0
                saving.save_time=t;
                saving.t_save(end+1)=t;
                saving.u_save(:,:,end+1)=Pt.u;
                saving.v_save(:,:,end+1)=Pt.v;
                saving.p_save(:,:,end+1)=Pt.pi*P.rho;
                save(['Data', ' v=',num2str(P.v_in*100),'e-
2','ComplexGeometry']);

                %% check steady state
                if max(max(abs(saving.u_save(:,:,end)-saving.u_save(:,:,end-
1))))<P.epsilon_u
                    if max(max(abs(saving.v_save(:,:,end)-
saving.v_save(:,:,end-1))))<P.epsilon_v
                        if max(max(abs(saving.p_save(:,:,end)-
saving.p_save(:,:,end-1))))<P.epsilon_p
                                t_end=t
                                break
                        end
                    end
                end
            end
        end

        %% check Re and CFL #
        %Re
%         Re_u=max(max(abs(Pt.u)))*P.w/P.nu;
%         Re_v=max(max(abs(Pt.v)))*P.w/P.nu;
%         %check laminar
%         if max(Re_u,Re_v)>2000
%             'Re # too high'
%             break
%         end
%         %CFL
%         CFL_u=P.dt/min(min(P.dx./abs(Pt.u)));
%         CFL_v=P.dt/min(min(P.dy./abs(Pt.v)));
%         %check stability
%         if max(CFL_u,CFL_v)>1
%             'dt too high (CFL)'
%             break
%         end

    end

end
```