Driven-cavity benchmark problem in pressure-velocity formulation

Chemical Reaction Engineering
and Chemical Kinetics

Advanced Transport Phenomena
Prof. Alberto Cuoci

Objectives

- Write a numerical code in MATLAB(R) to solve the driven cavity problem based on the velocity/pressure formulation
- Implement finite volume discretization on staggered grids
- Implement the projection algorithm for managing the coupling between pressure and velocity
- Use the numerical code for performing sensitivity analysis with respect to several parameters
- Compare the numerical results with experimental data available in the literature

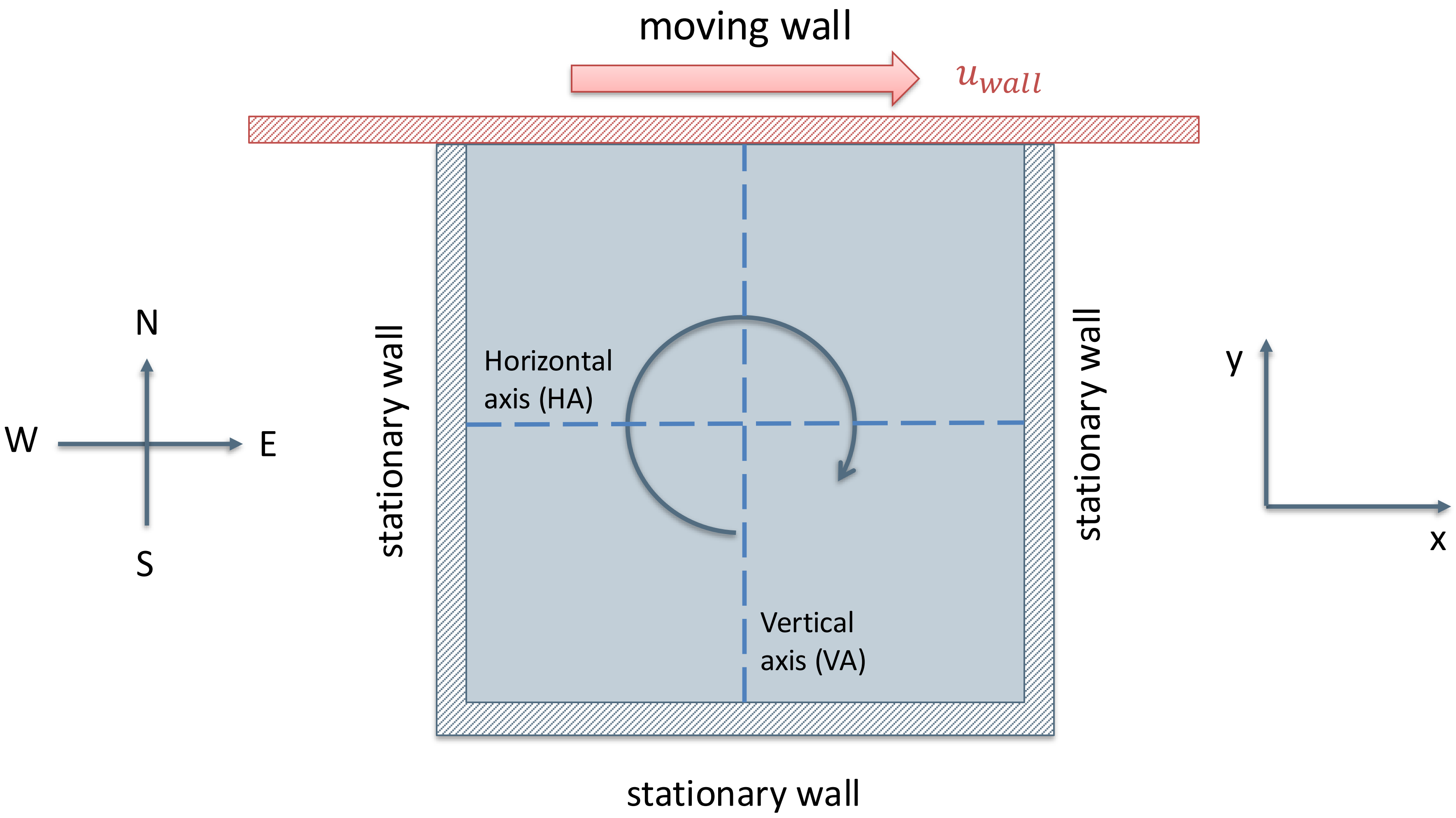
Outline

1. **Mathematical formulation**
2. **Numerical formulation**
   a. mesh
   b. finite volume formulation of momentum equations
   c. Poisson equation for pressure
   d. correction on velocity
   e. boundary conditions
3. **Experimental data**
4. **Results**
   a. comparison with experimental data
   b. grid sensitivity
5. **Final comments**

Outline

1. **Mathematical formulation**
2. **Numerical formulation**
   a. mesh
   b. finite volume formulation of momentum equations
   c. Poisson equation for pressure
   d. correction on velocity
   e. boundary conditions
3. **Experimental data**
4. **Results**
   a. comparison with experimental data
   b. grid sensitivity
5. **Final comments**

The driven-cavity problem



Navier-Stokes equations: differential formulation

Navier-Stokes equations in 2D

$$
\begin{cases}
\dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial y} = 0 \\[2ex]
\dfrac{\partial u}{\partial t} + \left(u \dfrac{\partial u}{\partial x} + v \dfrac{\partial u}{\partial y} \right) = -\dfrac{1}{\rho} \dfrac{\partial p}{\partial x} + \upsilon \left(\dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2} \right) \\[2ex]
\dfrac{\partial v}{\partial t} + \left(u \dfrac{\partial v}{\partial x} + v \dfrac{\partial v}{\partial y} \right) = -\dfrac{1}{\rho} \dfrac{\partial p}{\partial y} + \upsilon \left(\dfrac{\partial^2 v}{\partial x^2} + \dfrac{\partial^2 v}{\partial y^2} \right)
\end{cases}
$$

Conservation of mass

Conservation of momentum

Navier-Stokes equations: integral formulation

Navier-Stokes equations in 2D (integral formulation)

$$
\begin{cases}
\oint_S \vec{\boldsymbol{u}} \cdot \boldsymbol{n} dS = 0 \\
\dfrac{\partial}{\partial t} \int_V u dV = -\oint_S u\vec{\boldsymbol{u}} \cdot \boldsymbol{n} dS - \dfrac{1}{\rho} \oint_S p n_x dS + \upsilon \oint_S \nabla u \cdot \boldsymbol{n} dS \\
\dfrac{\partial}{\partial t} \int_V v dV = -\oint_S v\vec{\boldsymbol{u}} \cdot \boldsymbol{n} dS - \dfrac{1}{\rho} \oint_S p n_y dS + \upsilon \oint_S \nabla v \cdot \boldsymbol{n} dS
\end{cases}
$$

Conservation of mass

Conservation of momentum

Outline

1. Mathematical formulation
2. **Numerical formulation**
   a. mesh
   b. finite volume formulation of momentum equations
   c. Poisson equation for pressure
   d. correction on velocity
   e. boundary conditions
3. Experimental data
4. Results
   a. comparison with experimental data
   b. grid sensitivity
5. Final comments

Numerical algorithm

Initial fields given

Determine velocity field boundary conditions

Advect: $\vec{\boldsymbol{u}}'_{i,j} = \vec{\boldsymbol{u}}^n_{i,j} + \Delta t\left(-\boldsymbol{A}^n_{i,j} + \boldsymbol{D}^n_{i,j}\right)$

Solve Poisson equation (SOR):

$$\nabla_h^2 P_{i,j} = \frac{1}{\Delta t}\nabla_h \boldsymbol{u}'_{i,j}$$

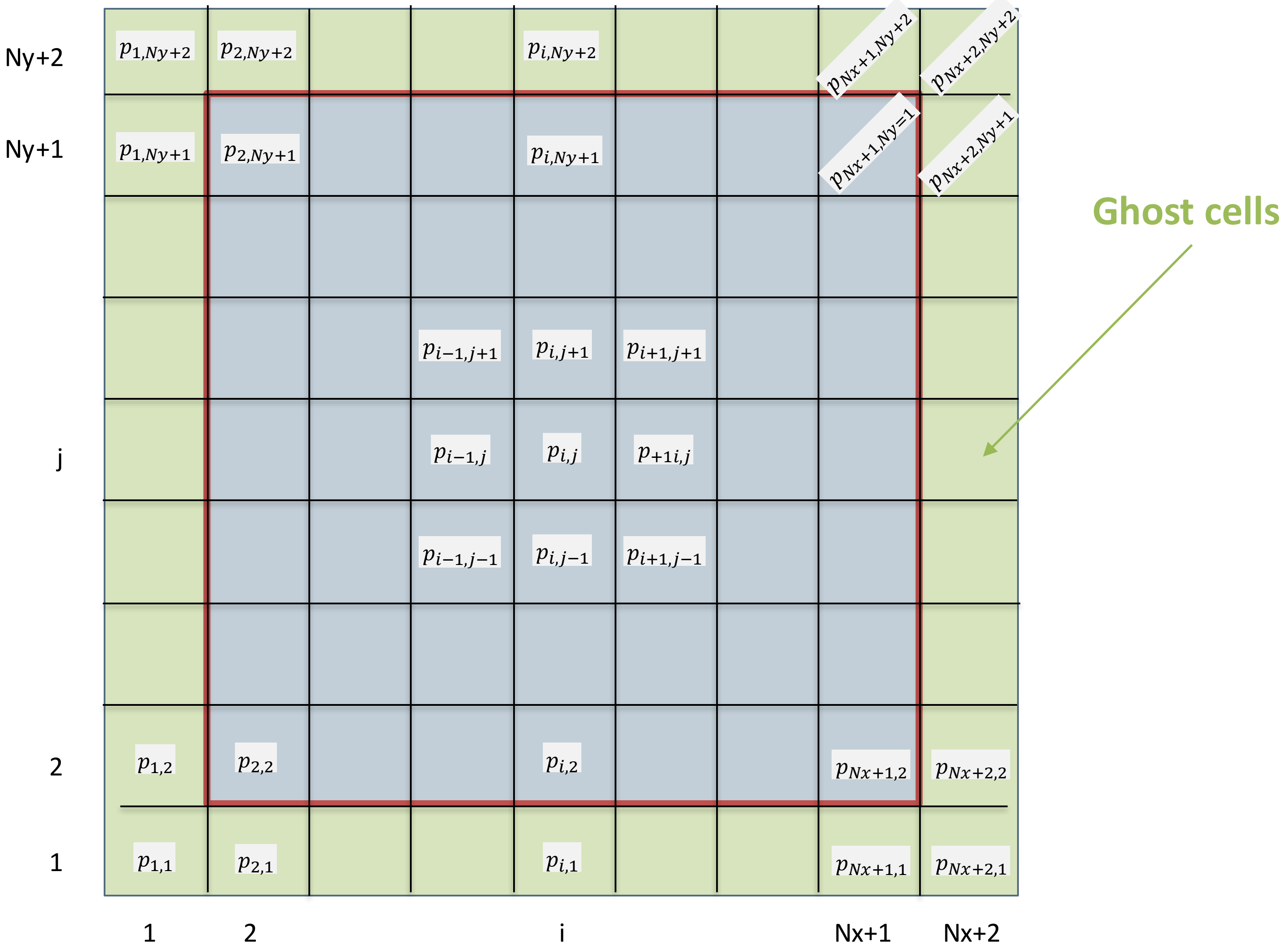
Projection:

$$\vec{\boldsymbol{u}}^{n+1}_{i,j} = \vec{\boldsymbol{u}}'_{i,j} - \Delta t \nabla_h P_{i,j}$$

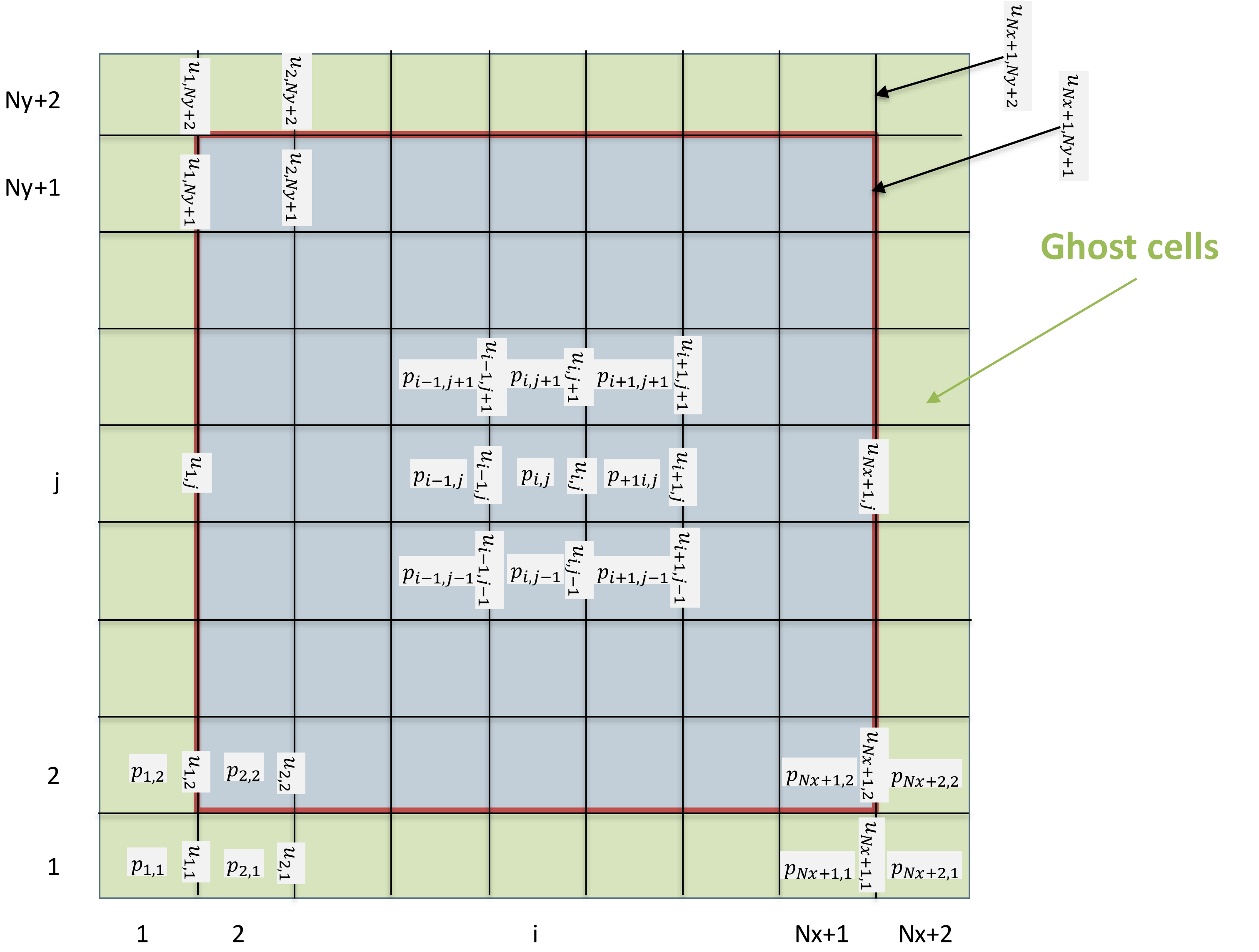
$t = t + \Delta t$

```
Initialize parameters and arrays
Set time step

for is=1:nstep


    Set BCs for tangential
    velocity (ghost points)

    Find predicted velocity

    Solve for pressure using SOR

    Find the projected velocity
    by adding the pressure
    gradient

    Post-processing (plot)

end
```

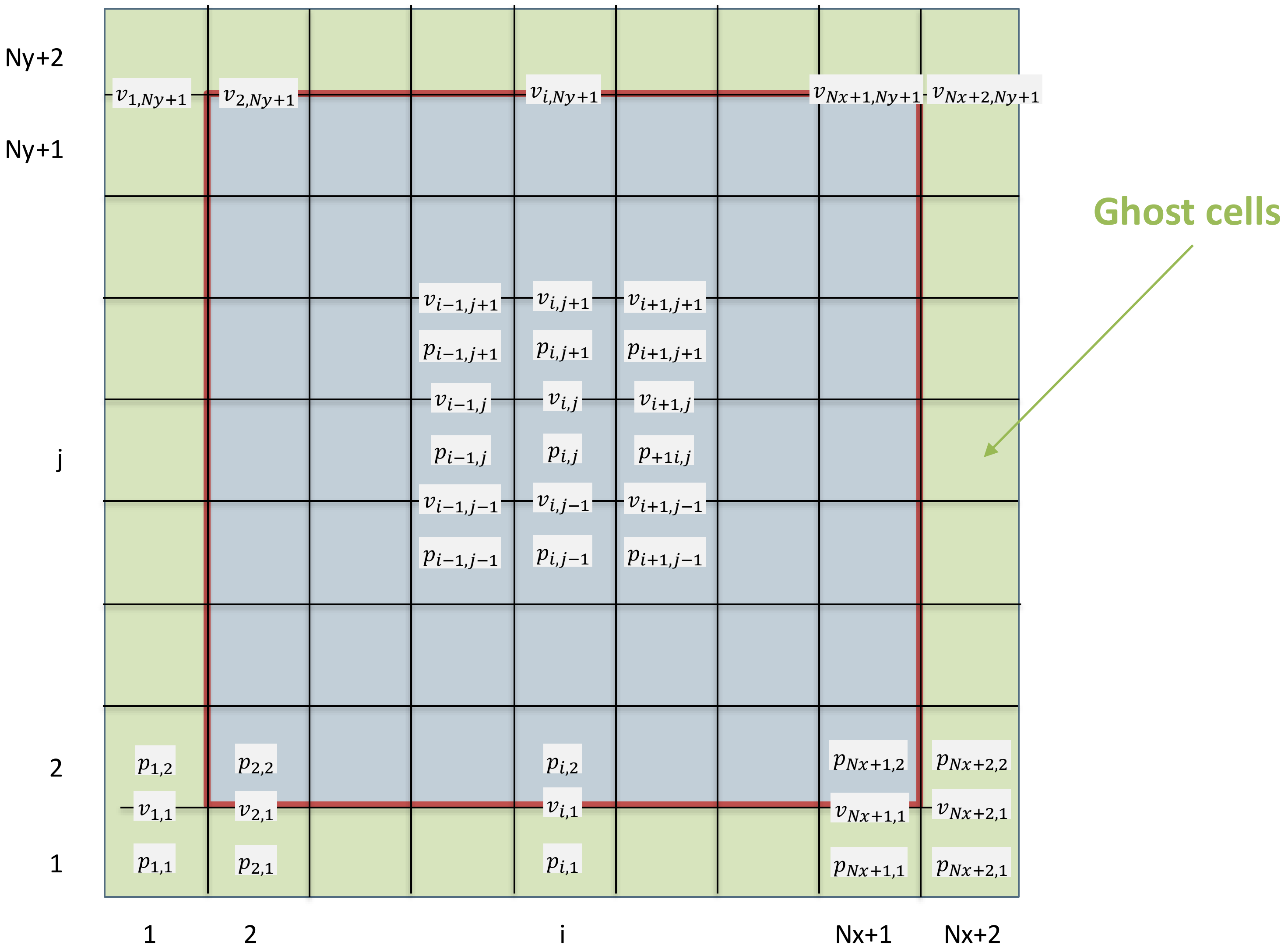
Computational grid: pressure $(N_X + 2) \times (N_Y + 2)$



Computational grid: u-velocity $(N_X + 1) \times (N_Y + 2)$



Computational grid: v-velocity $(N_X + 2) \times (N_Y + 1)$



Momentum equation along x

$$\frac{u_{i+1/2,j}^{n+1} - u_{i+1/2,j}^{n}}{\Delta t} = \cdots$$

$$\dots = -\left(\left[u_{i+1,j}^{n}\right]^2 - \left[u_{i,j}^{n}\right]^2 + u_{i+\frac{1}{2},j+\frac{1}{2}}^{n} v_{i+\frac{1}{2},j+\frac{1}{2}}^{n} - u_{i+\frac{1}{2},j-\frac{1}{2}}^{n} v_{i+\frac{1}{2},j-\frac{1}{2}}^{n}\right)\frac{1}{h} + \cdots$$

$$\dots + \frac{\upsilon}{h^2}\left(u_{i+\frac{3}{2},j}^{n} + u_{i-\frac{1}{2},j}^{n} + u_{i+\frac{1}{2},j+1}^{n} + u_{i+\frac{1}{2},j-1}^{n} - 4u_{i+\frac{1}{2},j}^{n}\right) + \cdots$$

$$\dots - \frac{P_{i+1,j} - P_{i,j}}{h}$$

$$\frac{u_{i+1/2,j}^{n+1} - u_{i+1/2,j}^{n}}{\Delta t} = -A_{i+\frac{1}{2},j}^{n} + D_{i+\frac{1}{2},j}^{n} - \nabla_{hi} P_{i,j}$$

Projection algorithm: temporary velocity (I)

$$\frac{U_{i+1/2,j}^{n+1} - u_{i+1/2,j}^{n}}{\Delta t} = \cdots$$

Temporary velocity (from the projection algorithm)

$$\dots = -\left(\left[u_{i+1,j}^{n}\right]^2 - \left[u_{i,j}^{n}\right]^2 + u_{i+\frac{1}{2},j+\frac{1}{2}}^{n} v_{i+\frac{1}{2},j+\frac{1}{2}}^{n} - u_{i+\frac{1}{2},j-\frac{1}{2}}^{n} v_{i+\frac{1}{2},j-\frac{1}{2}}^{n}\right)\frac{1}{h} + \cdots$$

$$\dots + \frac{\upsilon}{h^2}\left(u_{i+\frac{3}{2},j}^{n} + u_{i-\frac{1}{2},j}^{n} + u_{i+\frac{1}{2},j+1}^{n} + u_{i+\frac{1}{2},j-1}^{n} - 4u_{i+\frac{1}{2},j}^{n}\right)$$

$$\frac{U_{i+1/2,j}^{n+1} - u_{i+1/2,j}^{n}}{\Delta t} = -A_{i+\frac{1}{2},j}^{n} + D_{i+\frac{1}{2},j}^{n}$$

Projection algorithm: temporary velocity (II)

Temporary velocity (from the projection algorithm)

$$U^{n+1}_{i+1/2,j} = u^{n}_{i+1/2,j} + \Delta t \left(-A^{n}_{i+\frac{1}{2},j} + D^{n}_{i+\frac{1}{2},j}\right)$$

$$A^{n}_{i+1/2,j} = -\left(\left[u^{n}_{i+1,j}\right]^2 - \left[u^{n}_{i,j}\right]^2 + u^{n}_{i+\frac{1}{2},j+\frac{1}{2}} v^{n}_{i+\frac{1}{2},j+\frac{1}{2}} - u^{n}_{i+\frac{1}{2},j-\frac{1}{2}} v^{n}_{i+\frac{1}{2},j-\frac{1}{2}}\right)\frac{1}{h}$$

The four terms in the expression above are not directly available, but interpolation is needed

$$D^{n}_{i+\frac{1}{2},j} = \frac{\upsilon}{h^2}\left(u^{n}_{i+\frac{3}{2},j} + u^{n}_{i-\frac{1}{2},j} + u^{n}_{i+\frac{1}{2},j+1} + u^{n}_{i+\frac{1}{2},j-1} - 4u^{n}_{i+\frac{1}{2},j}\right)$$

The terms above are directly available

Implementation of diffusion term

$$D^n_{i,j+\frac{1}{2}} = \frac{\nu}{h^2}\left(v^n_{i,j+\frac{3}{2}} + v^n_{i,j-\frac{1}{2}} + v^n_{i+1,j+\frac{1}{2}} + v^n_{i-1,j+\frac{1}{2}} - 4v^n_{i,j+\frac{1}{2}}\right)$$

The terms above are directly available

Be careful! Since a fractional number is not allowed in computer program, redefine velocity node indices:

$$\begin{cases} u(i,j) = u_{i+1/2,j} \\ v(i,j) = v_{i,j+1/2} \end{cases}$$

$$D^n_{(i,j)} = \frac{\nu}{h^2}\left(v^n_{(i,j+1)} + v^n_{(i,j-1)} + v^n_{(i+1,j)} + v^n_{(i-1,j)} - 4v^n_{(i,j)}\right)$$

```
D = (nu/h^2)*(v(i+1,j)+v(i-1,j)+v(i,j+1)+v(i,j-1)-4*v(i,j));
```

Implementation of advection term (I)

$$A^n_{i+1/2,j} = -\left(\left[u^n_{i+1,j}\right]^2 - \left[u^n_{i,j}\right]^2 + u^n_{i+\frac{1}{2},j+\frac{1}{2}} v^n_{i+\frac{1}{2},j+\frac{1}{2}} - u^n_{i+\frac{1}{2},j-\frac{1}{2}} v^n_{i+\frac{1}{2},j-\frac{1}{2}}\right)\frac{1}{h}$$

The four terms in the expression above are not directly available, but interpolation is needed

$$u^n_{i+1,j} = \frac{u^n_{i+3/2,j} + u^n_{i+1/2,j}}{2}$$

$$u^n_{i+\frac{1}{2},j+\frac{1}{2}} v^n_{i+\frac{1}{2},j+\frac{1}{2}} = \frac{u^n_{i+\frac{1}{2},j} + u^n_{i+\frac{1}{2},j+1}}{2} \frac{v^n_{i,j+\frac{1}{2}} + v^n_{i+1,j+\frac{1}{2}}}{2}$$

$$u^n_{i,j} = \frac{u^n_{i+1/2,j} + u^n_{i-1/2,j}}{2}$$

$$u^n_{i+\frac{1}{2},j-\frac{1}{2}} v^n_{i+\frac{1}{2},j-\frac{1}{2}} = \frac{u^n_{i+\frac{1}{2},j} + u^n_{i+\frac{1}{2},j-1}}{2} \frac{v^n_{i,j-\frac{1}{2}} + v^n_{i+1,j-\frac{1}{2}}}{2}$$

Implementation of advection term (II)

$$u_{i+1,j}^{n} = \frac{u_{i+3/2,j}^{n} + u_{i+1/2,j}^{n}}{2}$$

$$\downarrow$$

$$\boxed{u_{e}^{n} = \frac{u_{(i+1,j)}^{n} + u_{(i,j)}^{n}}{2}}$$

$$u_{i,j}^{n} = \frac{u_{i+1/2,j}^{n} + u_{i-1/2,j}^{n}}{2}$$

$$\downarrow$$

$$\boxed{u_{w}^{n} = \frac{u_{(i,j)}^{n} + u_{(i-1,j)}^{n}}{2}}$$

Implementation of advection term (III)

$$u^n_{i+\frac{1}{2},j+\frac{1}{2}} v^n_{i+\frac{1}{2},j+\frac{1}{2}} = \frac{u^n_{i+\frac{1}{2},j} + u^n_{i+\frac{1}{2},j+1}}{2} \frac{v^n_{i,j+\frac{1}{2}} + v^n_{i+1,j+\frac{1}{2}}}{2}$$

$$\boxed{u^n_n v^n = \frac{u^n_{(i,j)} + u^n_{(i,j+1)}}{2} \frac{v^n_{(i,j)} + v^n_{(i+1,j)}}{2}}$$

$$u^n_{i+\frac{1}{2},j-\frac{1}{2}} v^n_{i+\frac{1}{2},j-\frac{1}{2}} = \frac{u^n_{i+\frac{1}{2},j} + u^n_{i+\frac{1}{2},j-1}}{2} \frac{v^n_{i,j-\frac{1}{2}} + v^n_{i+1,j-\frac{1}{2}}}{2}$$

$$\boxed{u^n_s v^n = \frac{u^n_{(i,j)} + u^n_{(i,j-1)}}{2} \frac{v^n_{(i,j-1)} + v^n_{(i+1,j-1)}}{2}}$$

Implementation of advection/diffusion along x

```
% Temporary u-velocity
for i=2:nx
    for j=2:ny+1

        ue2 = 0.25*( u(i+1,j)+u(i,j) )^2;
        uw2 = 0.25*( u(i,j)+u(i-1,j) )^2;
        unv = 0.25*( u(i,j+1)+u(i,j) )*( v(i+1,j)+v(i,j) );
        usv = 0.25*( u(i,j)+u(i,j-1) )*( v(i+1,j-1)+v(i,j-1) );

        A = (ue2-uw2+unv-usv)/h;
        D = (nu/h^2)*(u(i+1,j)+u(i-1,j)+u(i,j+1)+u(i,j-1)-4*u(i,j));

        ut(i,j)=u(i,j)+dt*(-A+D);

    end
end
```

Momentum equation along y

$$\frac{v^{n+1}_{i,j+1/2} - v^{n}_{i,j+1/2}}{\Delta t} = \cdots$$

$$\ldots = -\left(u^{n}_{i+\frac{1}{2},j+\frac{1}{2}} v^{n}_{i+\frac{1}{2},j+\frac{1}{2}} - u^{n}_{i-\frac{1}{2},j+\frac{1}{2}} v^{n}_{i-\frac{1}{2},j+\frac{1}{2}} + \left[v^{n}_{i,j+1}\right]^2 - \left[v^{n}_{i,j}\right]^2 \right) \frac{1}{h} + \cdots$$

$$\ldots + \frac{\nu}{h^2}\left(v^{n}_{i,j+\frac{3}{2}} + v^{n}_{i,j-\frac{1}{2}} + v^{n}_{i+1,j+\frac{1}{2}} + v^{n}_{i-1,j+\frac{1}{2}} - 4 v^{n}_{i,j+\frac{1}{2}} \right) + \cdots$$

$$\ldots - \frac{P_{i,j+1} - P_{i,j}}{h}$$

$$\boxed{\frac{v^{n+1}_{i,j+1/2} - v^{n}_{i,j+1/2}}{\Delta t} = -A^{n}_{i,j+1/2} + D^{n}_{i,j+1/2} - \nabla_{hj} P_{i,j}}$$

Projection algorithm: temporary velocity (I)

$$\frac{V_{i,j+1/2}^{n+1} - v_{i,j+1/2}^{n}}{\Delta t} = \cdots$$

Temporary velocity (from the projection algorithm)

$$\dots = -\left(u^n_{i+\frac{1}{2},j+\frac{1}{2}} v^n_{i+\frac{1}{2},j+\frac{1}{2}} - u^n_{i-\frac{1}{2},j+\frac{1}{2}} v^n_{i-\frac{1}{2},j+\frac{1}{2}} + \left[v^n_{i,j+1}\right]^2 - \left[v^n_{i,j}\right]^2 \right) \frac{1}{h} + \cdots$$

$$\dots + \frac{\nu}{h^2} \left(v^n_{i,j+\frac{3}{2}} + v^n_{i,j-\frac{1}{2}} + v^n_{i+1,j+\frac{1}{2}} + v^n_{i-1,j+\frac{1}{2}} - 4 v^n_{i,j+\frac{1}{2}} \right) + \cdots$$

$$\boxed{\frac{V_{i,j+1/2}^{n+1} - v_{i,j+1/2}^{n}}{\Delta t} = -A^n_{i,j+\frac{1}{2}} + D^n_{i,j+\frac{1}{2}}}$$

Projection algorithm: temporary velocity (II)

Temporary velocity (from the projection algorithm)

$$V_{i,j+1/2}^{n+1} = v_{i,j+1/2}^{n} + \Delta t \left(-A_{i,j+\frac{1}{2}}^{n} + D_{i,j+\frac{1}{2}}^{n} \right)$$

$$A_{i,j+\frac{1}{2}}^{n} = -\left(u_{i+\frac{1}{2},j+\frac{1}{2}}^{n} v_{i+\frac{1}{2},j+\frac{1}{2}}^{n} - u_{i-\frac{1}{2},j+\frac{1}{2}}^{n} v_{i-\frac{1}{2},j+\frac{1}{2}}^{n} + \left[v_{i,j+1}^{n} \right]^2 - \left[v_{i,j}^{n} \right]^2 \right) \frac{1}{h}$$

The four terms in the expression above are not directly available, but interpolation is needed

$$D_{i,j+\frac{1}{2}}^{n} = \frac{\upsilon}{h^2} \left(v_{i,j+\frac{3}{2}}^{n} + v_{i,j-\frac{1}{2}}^{n} + v_{i+1,j+\frac{1}{2}}^{n} + v_{i-1,j+\frac{1}{2}}^{n} - 4 v_{i,j+\frac{1}{2}}^{n} \right)$$

The terms above are directly available

Implementation of diffusion term

$$D^n_{i+\frac{1}{2},j} = \frac{\upsilon}{h^2}\left(u^n_{i+\frac{3}{2},j} + u^n_{i-\frac{1}{2},j} + u^n_{i+\frac{1}{2},j+1} + u^n_{i+\frac{1}{2},j-1} - 4u^n_{i+\frac{1}{2},j}\right)$$

The terms above are directly available

Be careful! Since a fractional number is not allowed in computer program, redefine velocity node indices:

$$\begin{cases} u(i,j) = u_{i+1/2,j} \\ v(i,j) = v_{i,j+1/2} \end{cases}$$

$$D^n_{(i,j)} = \frac{\upsilon}{h^2}\left(u^n_{(i+1,j)} + u^n_{(i-1,j)} + u^n_{(i,j+1)} + u^n_{(i,j-1)} - 4u^n_{(i,j)}\right)$$

```
D = (nu/h^2)*(u(i+1,j)+u(i-1,j)+u(i,j+1)+u(i,j-1)-4*u(i,j));
```

Implementation of advection term (I)

$$A^n_{i,j+\frac{1}{2}} = -\left(u^n_{i+\frac{1}{2},j+\frac{1}{2}} v^n_{i+\frac{1}{2},j+\frac{1}{2}} - u^n_{i-\frac{1}{2},j+\frac{1}{2}} v^n_{i-\frac{1}{2},j+\frac{1}{2}} + \left[v^n_{i,j+1}\right]^2 - \left[v^n_{i,j}\right]^2\right)\frac{1}{h}$$

The four terms in the expression above are not directly available, but interpolation is needed

$$v^n_{i,j+1} = \frac{v^n_{i,j+3/2} + v^n_{i,j+1/2}}{2}$$

$$u^n_{i+\frac{1}{2},j+\frac{1}{2}} v^n_{i+\frac{1}{2},j+\frac{1}{2}} = \frac{u^n_{i+\frac{1}{2},j} + u^n_{i+\frac{1}{2},j+1}}{2} \frac{v^n_{i,j+\frac{1}{2}} + v^n_{i+1,j+\frac{1}{2}}}{2}$$

$$v^n_{i,j} = \frac{v^n_{i,j+1/2} + v^n_{i,j-1/2}}{2}$$

$$u^n_{i-\frac{1}{2},j+\frac{1}{2}} v^n_{i-\frac{1}{2},j+\frac{1}{2}} = \frac{u^n_{i-\frac{1}{2},j} + u^n_{i-\frac{1}{2},j+1}}{2} \frac{v^n_{i,j+\frac{1}{2}} + v^n_{i-1,j+\frac{1}{2}}}{2}$$

Implementation of advection term (II)

$$v_{i,j+1}^{n} = \frac{v_{i,j+3/2}^{n} + v_{i,j+1/2}^{n}}{2}$$

$$\Downarrow$$

$$\boxed{v_{n}^{n} = \frac{v_{(i,j+1)}^{n} + v_{(i,j)}^{n}}{2}}$$

$$v_{i,j}^{n} = \frac{v_{i,j+1/2}^{n} + v_{i,j-1/2}^{n}}{2}$$

$$\Downarrow$$

$$\boxed{v_{s}^{n} = \frac{v_{(i,j)}^{n} + v_{(i,j-1)}^{n}}{2}}$$

Implementation of advection term (III)

$$u^n_{i+\frac{1}{2},j+\frac{1}{2}} v^n_{i+\frac{1}{2},j+\frac{1}{2}} = \frac{u^n_{i+\frac{1}{2},j} + u^n_{i+\frac{1}{2},j+1}}{2} \frac{v^n_{i,j+\frac{1}{2}} + v^n_{i+1,j+\frac{1}{2}}}{2}$$

$$\boxed{u^n\, v^n_e = \frac{u^n_{(i,j)} + u^n_{(i,j+1)}}{2} \frac{v^n_{(i,j)} + v^n_{(i+1,j)}}{2}}$$

$$u^n_{i-\frac{1}{2},j+\frac{1}{2}} v^n_{i-\frac{1}{2},j+\frac{1}{2}} = \frac{u^n_{i-\frac{1}{2},j} + u^n_{i-\frac{1}{2},j+1}}{2} \frac{v^n_{i,j+\frac{1}{2}} + v^n_{i-1,j+\frac{1}{2}}}{2}$$

$$\boxed{u^n\, v^n_w = \frac{u^n_{(i-1,j)} + u^n_{(i-1,j-1)}}{2} \frac{v^n_{(i,j)} + v^n_{(i-1,j)}}{2}}$$

Implementation of advection/diffusion along y

```
% Temporary v-velocity
for i=2:nx+1
    for j=2:ny

        vn2 = 0.25*( v(i,j+1)+v(i,j) )^2;
        vs2 = 0.25*( v(i,j)+v(i,j-1) )^2;
        veu = 0.25*( u(i,j+1)+u(i,j) )*( v(i+1,j)+v(i,j) );
        vwu = 0.25*( u(i-1,j+1)+u(i-1,j) )*( v(i,j)+v(i-1,j) );

        A = (vn2 - vs2 + veu - vwu)/h;
        D = (nu/h^2)*(v(i+1,j)+v(i-1,j)+v(i,j+1)+v(i,j-1)-4*v(i,j));

        vt(i,j)=v(i,j)+dt*(-A+D);

    end
end
```

Discretized Poisson equation (I)

$$P_{i,j}^{\alpha+1} = \beta\left\{\left(P_{i+1,j}^{\alpha} + P_{i-1,j}^{\alpha+1} + P_{i,j+1}^{\alpha} + P_{i,j-1}^{\alpha+1}\right) - \frac{h}{\Delta t}\left(u'_{i,j} - u'_{i-1,j} + v'_{i,j} - v'_{i,j-1}\right)\right\}\gamma + (1-\beta)P_{i,j}^{\alpha}$$

All the terms above can be calculated directly, no interpolation is needed

Interior nodes: $i = 3 \ldots N_x;\; j = 3 \ldots N_y$ $\quad$ $\boldsymbol{\gamma = 1/4}$

Edge nodes: $i = 2; i = N_x + 1; j = 2; j = N_y + 1$ $\quad$ $\boldsymbol{\gamma = 1/3}$

Corner nodes: $(i,j) = (2,2), (N_x + 1,2), \left(2, N_y + 1\right), \left(N_x + 1, N_y + 1\right)$ $\quad$ $\boldsymbol{\gamma = 1/2}$

Discretized Poisson equation (II)

$$P_{i,j}^{\alpha+1} = \beta\left\{\left(P_{i+1,j}^{\alpha} + P_{i-1,j}^{\alpha+1} + P_{i,j+1}^{\alpha} + P_{i,j-1}^{\alpha+1}\right) - \frac{h}{\Delta t}\left(u'_{i,j} - u'_{i-1,j} + v'_{i,j} - v'_{i,j-1}\right)\right\}\gamma + (1-\beta)P_{i,j}^{\alpha}$$

$$P_{i,j}^{\alpha+1} = \beta\{\delta_{i,j} - S_{i,j}\}\gamma + (1-\beta)P_{i,j}^{\alpha} \qquad \begin{cases} \delta_{i,j} = \left(P_{i+1,j}^{\alpha} + P_{i-1,j}^{\alpha+1} + P_{i,j+1}^{\alpha} + P_{i,j-1}^{\alpha+1}\right) \\ S_{i,j} = \dfrac{h}{\Delta t}\left(u'_{i,j} - u'_{i-1,j} + v'_{i,j} - v'_{i,j-1}\right) \end{cases}$$

```
for i=2:nx+1
    for j=2:ny+1

        delta = p(i+1,j)+p(i-1,j)+p(i,j+1)+p(i,j-1);
        S = (h/dt)*(ut(i,j)-ut(i-1,j)+vt(i,j)-vt(i,j-1));

        p(i,j)=beta*gamma(i,j)*(delta-S)+(1-beta)*p(i,j);
    end
end
```

Correction on velocity

$$\vec{\boldsymbol{u}}_{i,j}^{n+1} = \vec{\boldsymbol{u}}'_{i,j} - \Delta t \nabla_h P_{i,j}$$

$$\begin{cases} u_{i,j}^{n+1} = u'_{i,j} - \Delta t \nabla_{hi} P_{i,j} \\ v_{i,j}^{n+1} = v'_{i,j} - \Delta t \nabla_{hj} P_{i,j} \end{cases}$$

$$\begin{cases} u_{(i,j)}^{n+1} = u'_{(i,j)} - \dfrac{\Delta t}{h}\left(p_{(i+1,j)}^{n} - p_{(i,j)}^{n}\right) \\ v_{(i,j)}^{n+1} = v'_{(i,j)} - \dfrac{\Delta t}{h}\left(p_{(i,j+1)}^{n} - p_{(i,j)}^{n}\right) \end{cases}$$

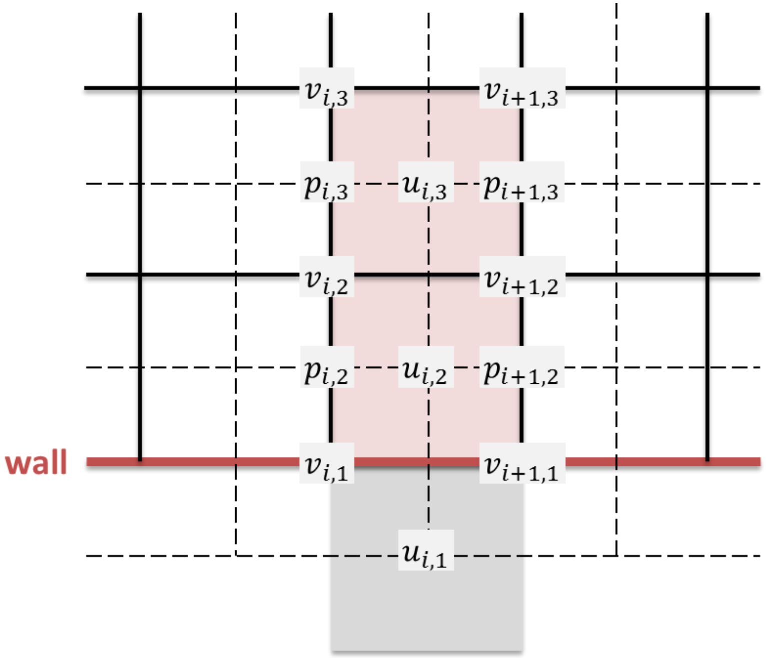
```
% Correct the velocity
u(2:nx,2:ny+1)=ut(2:nx,2:ny+1)-(dt/h)*(p(3:nx+1,2:ny+1)-p(2:nx,2:ny+1));
v(2:nx+1,2:ny)=vt(2:nx+1,2:ny)-(dt/h)*(p(2:nx+1,3:ny+1)-p(2:nx+1,2:ny));
```

Boundary conditions (parallel velocities)

Velocity of wall is given, U_{wall} (no-slip)

Solve for the "ghost" velocity

$$u_{i,1} = 2U_{wall} - u_{i,2}$$



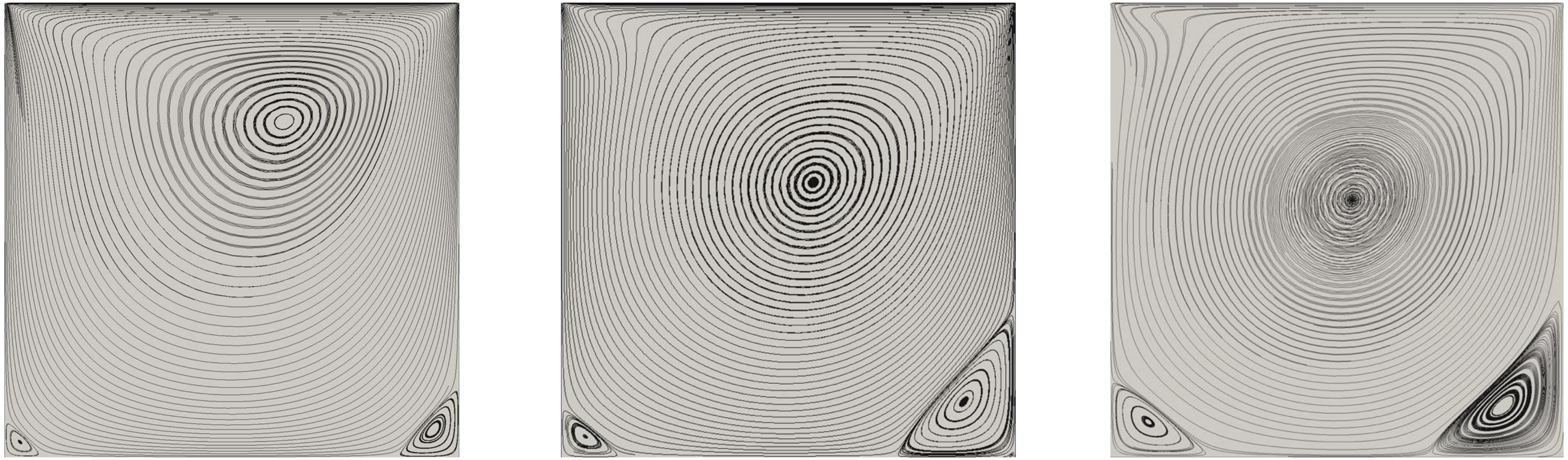

```
% Boundary conditions
u(1:nx+1,1)=2*us-u(1:nx+1,2);                % south wall
u(1:nx+1,ny+2)=2*un-u(1:nx+1,ny+1);          % north wall
v(1,1:ny+1)=2*vw-v(2,1:ny+1);                % west wall
v(nx+2,1:ny+1)=2*ve-v(nx+1,1:ny+1);          % east wall
```

Outline

1. **Mathematical formulation**
2. **Numerical formulation**
   a. mesh
   b. finite volume formulation of momentum equations
   c. Poisson equation for pressure
   d. correction on velocity
   e. boundary conditions
3. **Experimental data**
4. **Results**
   a. comparison with experimental data
   b. grid sensitivity
5. **Final comments**

Streamlines

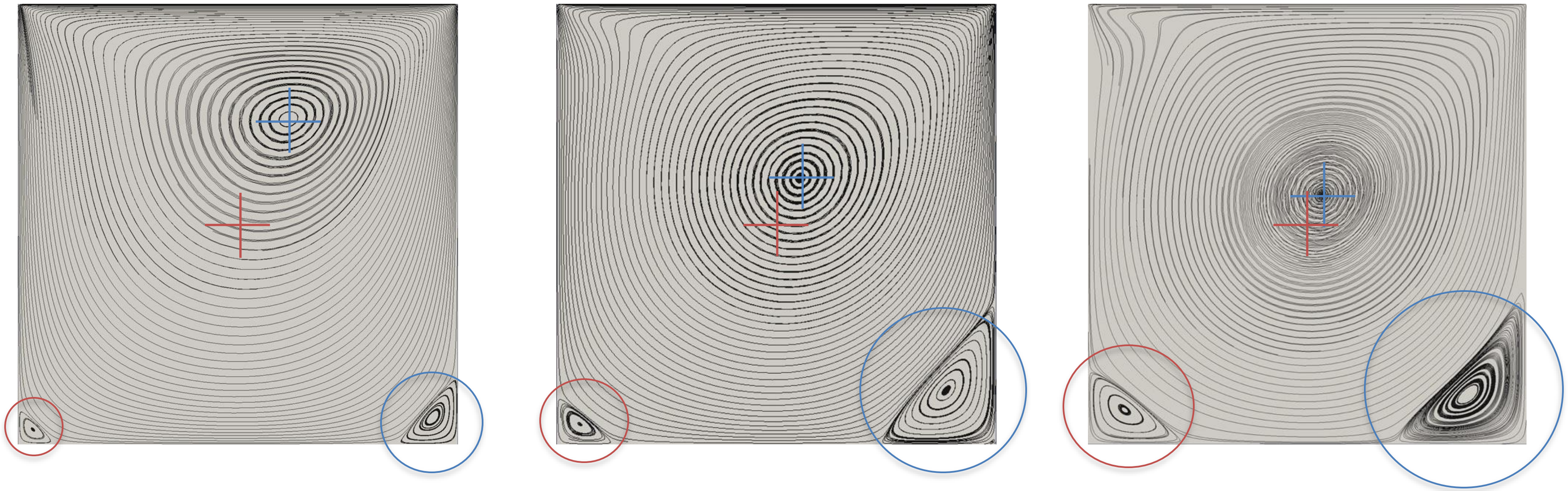
$Re = 100$ $Re = 400$ $Re = 1000$



Streamlines at steady state conditions
Numerical results with 1000 x 1000 grids (very fine)

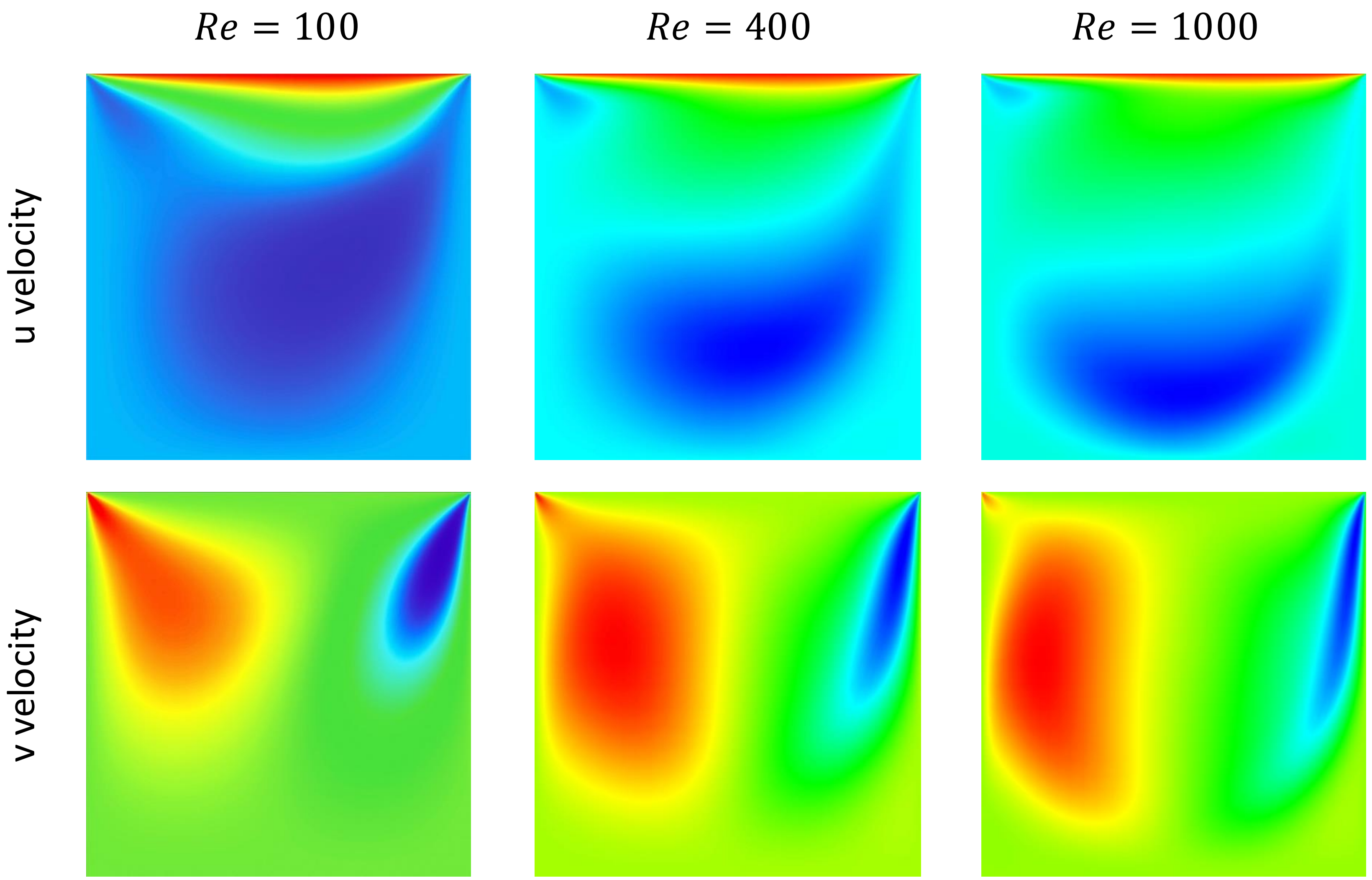
Streamlines

$Re = 100$ $\quad$ $Re = 400$ $\quad$ $Re = 1000$

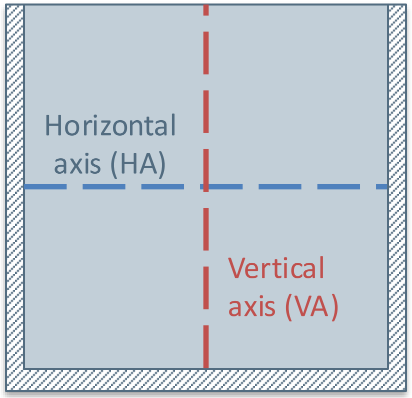


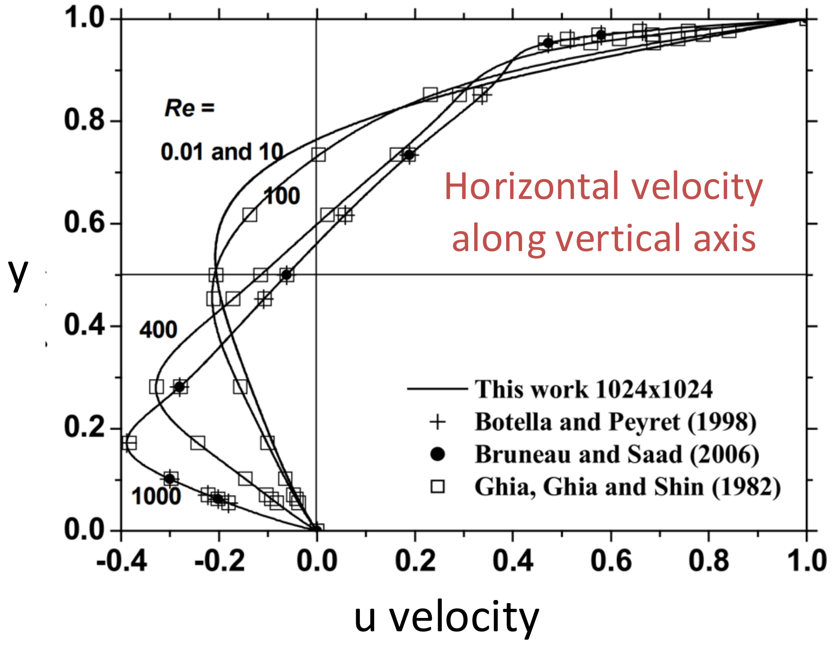
Streamlines at steady state conditions
Numerical results with 1000 x 1000 grids (very fine)

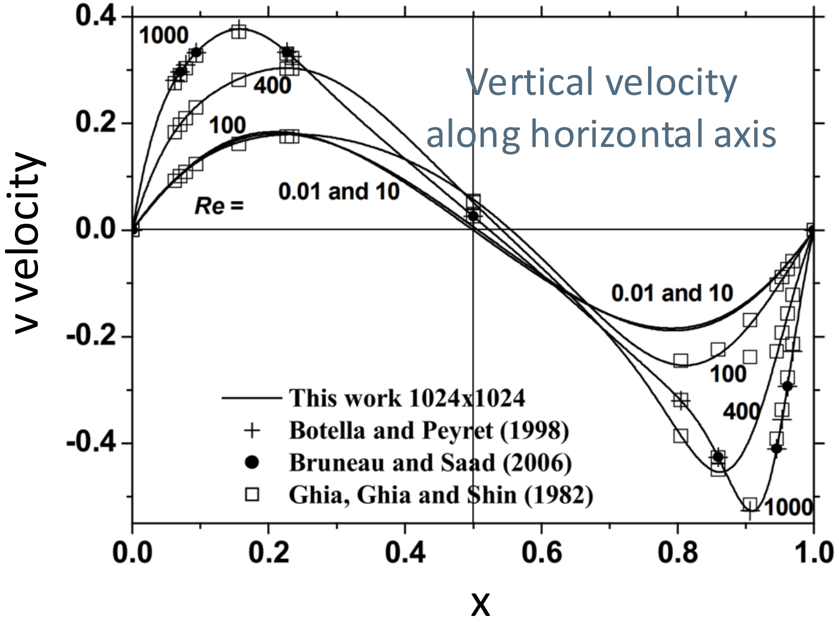
Velocity fields



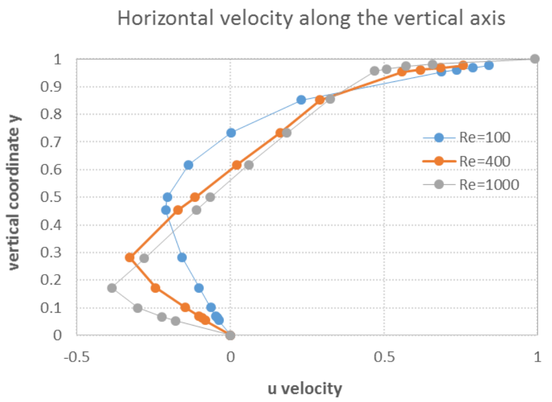
Experimental data (I)

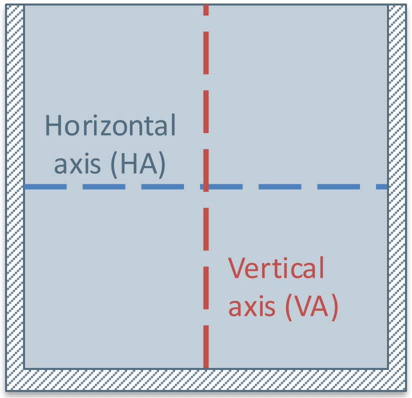


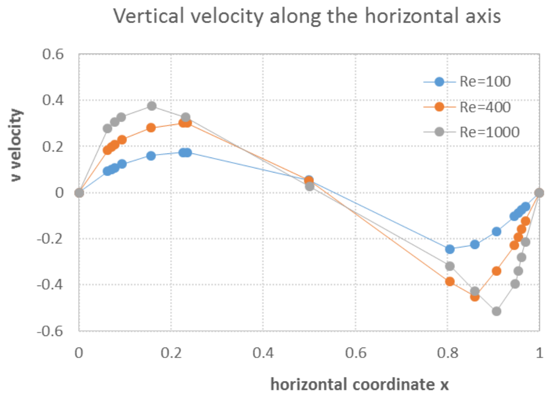




Experimental data (II)







Horizontal velocity along Vertical Axis					
y	Re=100	y	Re=400	y	Re=1000
0	0	0	0	0.00057	0.00088
0.0547	-0.0372	0.0547	-0.0819	0.0531	-0.179
0.0625	-0.0419	0.0625	-0.0927	0.06698	-0.22449
0.0703	-0.0477	0.0703	-0.1034	0.09974	-0.30102
0.1016	-0.0643	0.1016	-0.1461	0.17233	-0.38589
0.1719	-0.1015	0.1719	-0.243	0.27906	-0.27969
0.2812	-0.1566	0.2812	-0.3273	0.4526	-0.1106
0.4531	-0.2109	0.4531	-0.1712	0.49948	-0.06524
0.5	-0.2058	0.5	-0.1148	0.61818	0.05953
0.6172	-0.1364	0.6172	0.0214	0.7329	0.18432
0.7344	0.0033	0.7344	0.1626	0.85561	0.32561
0.8516	0.2315	0.8516	0.2909	0.95642	0.47005
0.9531	0.6872	0.9531	0.5589	0.96444	0.5093
0.9609	0.7372	0.9609	0.6176	0.9735	0.57231
0.9688	0.7887	0.9688	0.6844	0.98159	0.65908
0.9766	0.8412	0.9766	0.7582	0.99999	0.9907

Vertical velocity along the Horizontal axis					
x	Re=100	x	Re=400	x	Re=1000
0	0	0	0	0	0
0.0625	0.0923	0.0625	0.1836	0.06241	0.27821
0.0703	0.1009	0.0703	0.1971	0.07812	0.30477
0.0781	0.1089	0.0781	0.2092	0.09233	0.32848
0.0938	0.1232	0.0938	0.2297	0.15804	0.37485
0.1563	0.1608	0.1563	0.2812	0.23252	0.32618
0.2266	0.1751	0.2266	0.302	0.50117	0.02612
0.2344	0.1753	0.2344	0.3017	0.80552	-0.31774
0.5	0.0545	0.5	0.0519	0.85976	-0.42715
0.8047	-0.2453	0.8047	-0.386	0.90582	-0.51565
0.8594	-0.2245	0.8594	-0.4499	0.94706	-0.3951
0.9063	-0.1691	0.9063	-0.3383	0.95388	-0.33906
0.9453	-0.1031	0.9453	-0.2285	0.9607	-0.28017
0.9531	-0.0886	0.9531	-0.1925	0.96902	-0.21559
0.9609	-0.0739	0.9609	-0.1566	1	0.00E+00
0.9688	-0.0591	0.9688	-0.1215		
1	0	1	0		

Outline

1. **Mathematical formulation**
2. **Numerical formulation**
   a. mesh
   b. finite volume formulation of momentum equations
   c. Poisson equation for pressure
   d. correction on velocity
   e. boundary conditions
3. **Experimental data**
4. **Results**
   a. comparison with experimental data
   b. grid sensitivity
5. **Final comments**

MATLAB® code

MATLAB(R) Code

The complete MATLAB® code is available on GitHub:
https://github.com/acuoci/CFDofReactiveFlows/blob/master/codes/driven_cavity/driven_cavity_2d_staggered.m

C++ Code

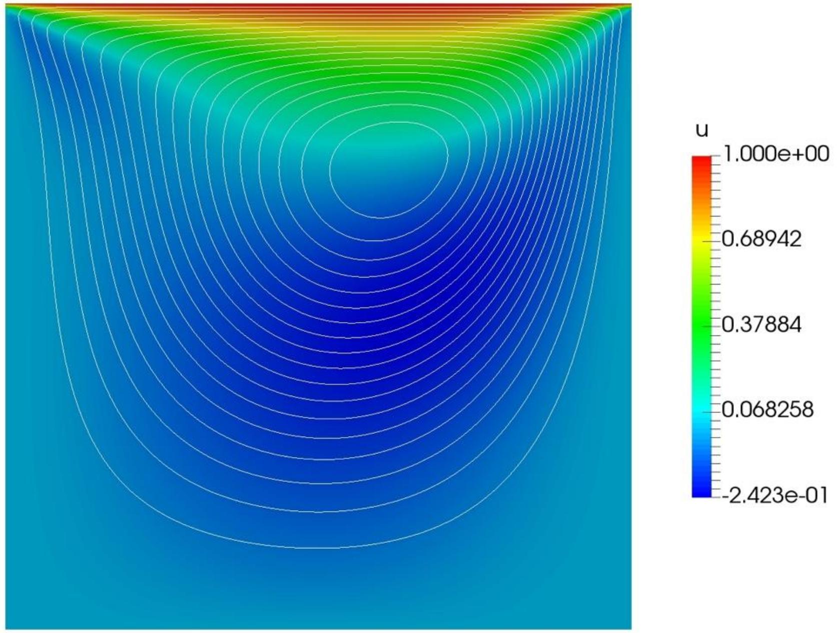
A C++ version is also available:
https://github.com/acuoci/CFDofReactiveFlows/blob/master/codes/driven_cavity/driven_cavity_2d_staggered.cpp

This version is based on Eigen C++ numerical libraries, for managing vectors, matrices, and linear algebra operations. The Eigen C++ libraries can be freely downloaded at:
http://eigen.tuxfamily.org/index.php?title=Main_Page

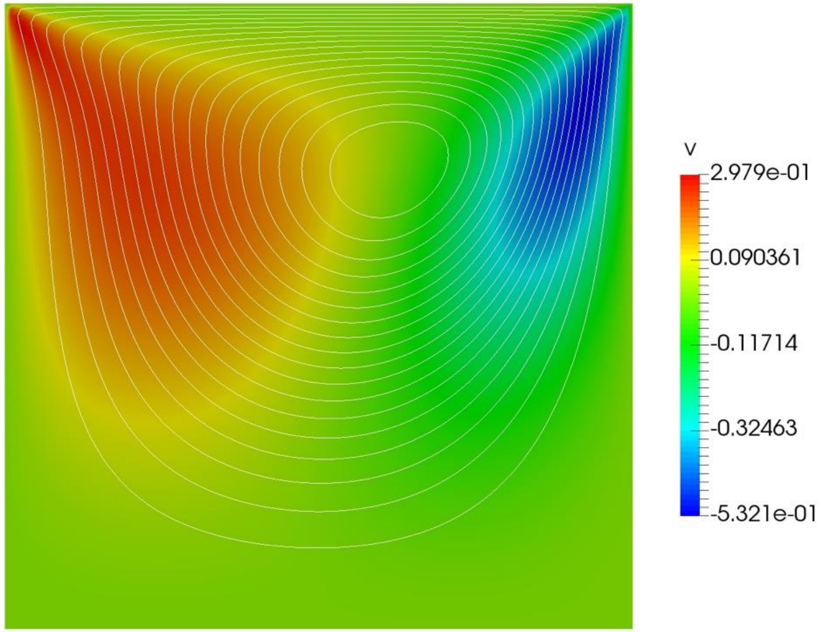
Graphical post-processing must be performed using external tools, like Tecplot, Paraview, etc. Paraview is strongly suggested:
https://www.paraview.org/download/

Steady state solution Re=100

u velocity + streamlines



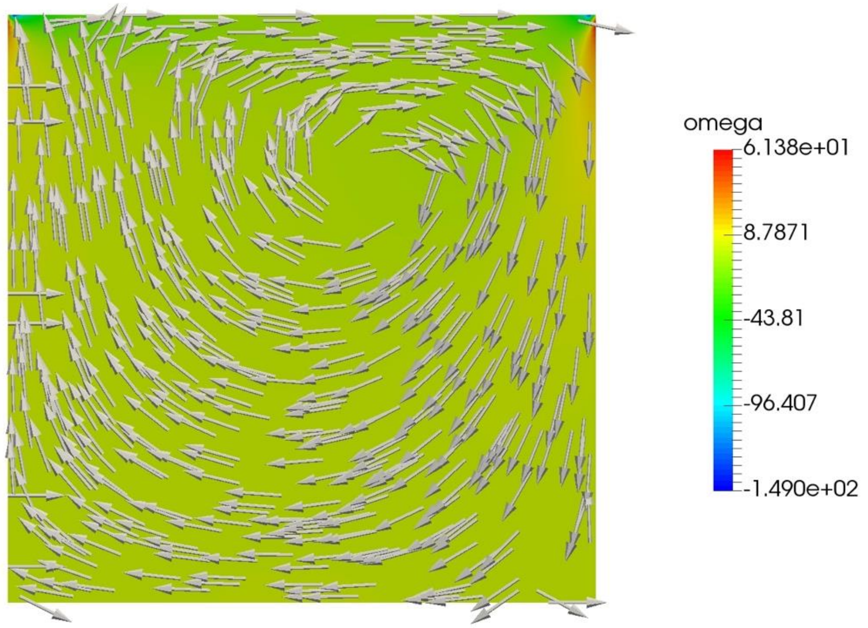
v velocity + streamlines



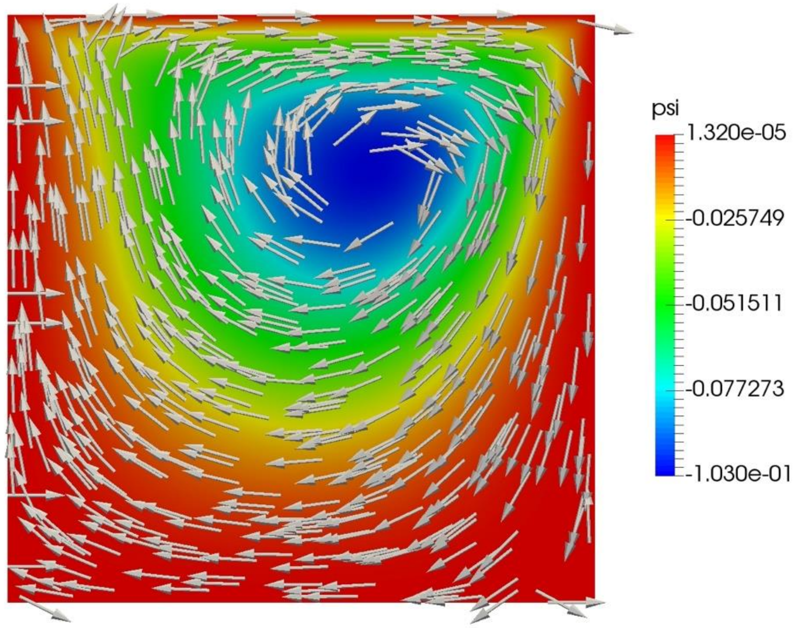
Grid: 100 x 100
SOR tolerance: 0.0001
SOR coefficient: 1.9

Steady state solution Re=100

Vorticity + velocity vectors



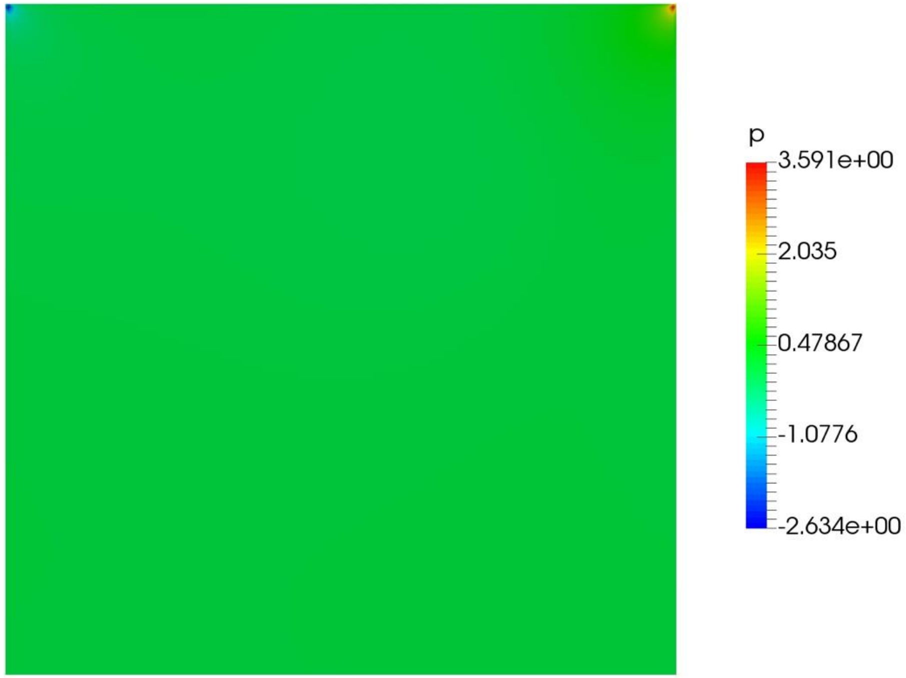
Streamfunction + velocity vectors



Grid: 100 x 100
SOR tolerance: 0.0001
SOR coefficient: 1.9

Steady state solution Re=100

Pressure [Pa]

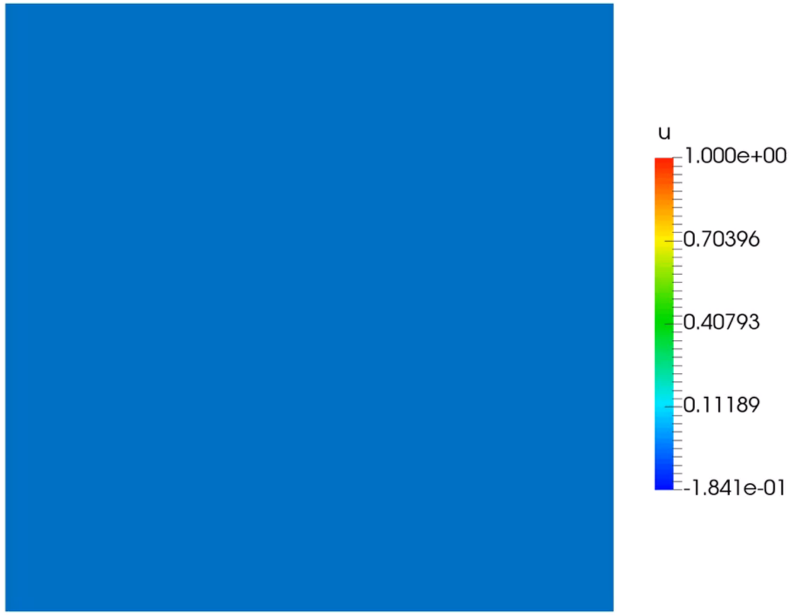


What is happening to the pressure?

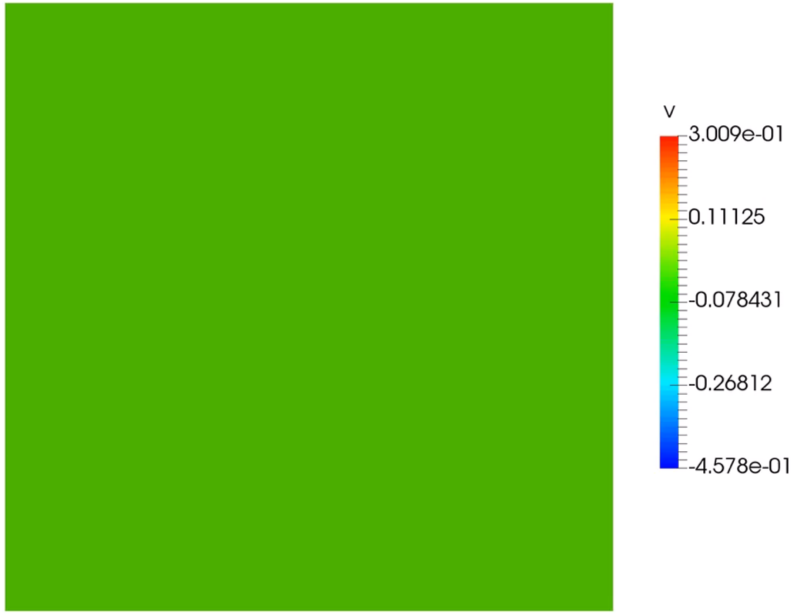
Grid: 100 x 100
SOR tolerance: 0.0001
SOR coefficient: 1.9

Time evolution for Re=100

u velocity + streamlines



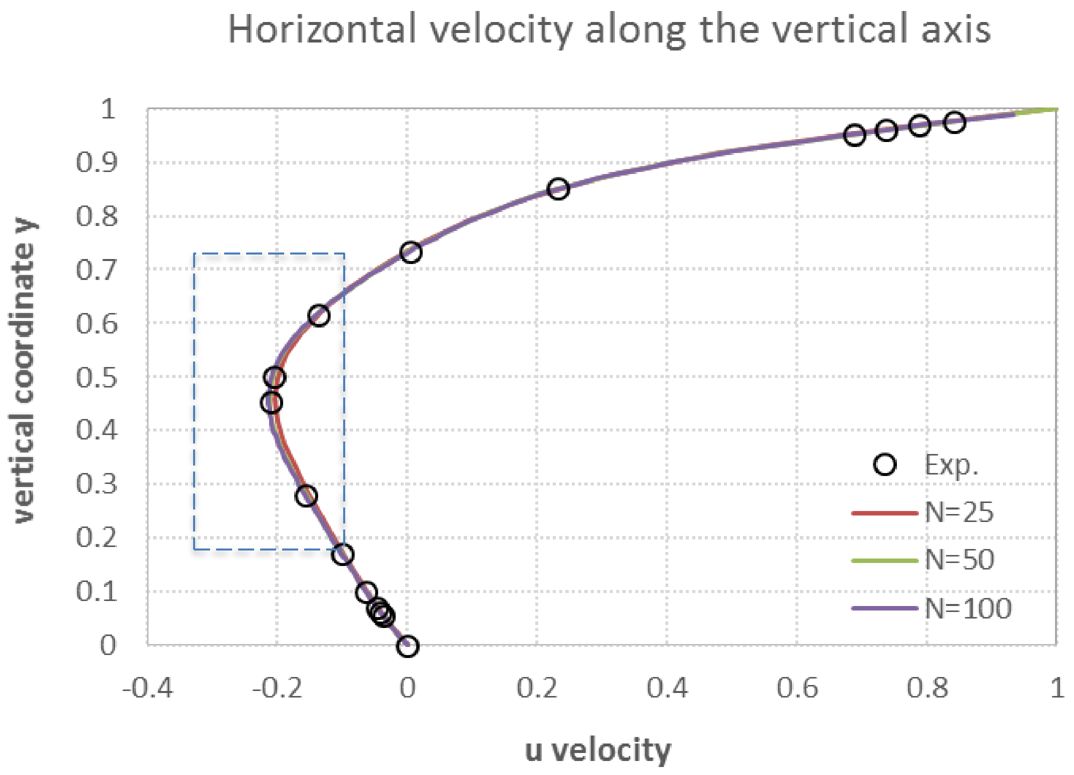
v velocity + streamlines

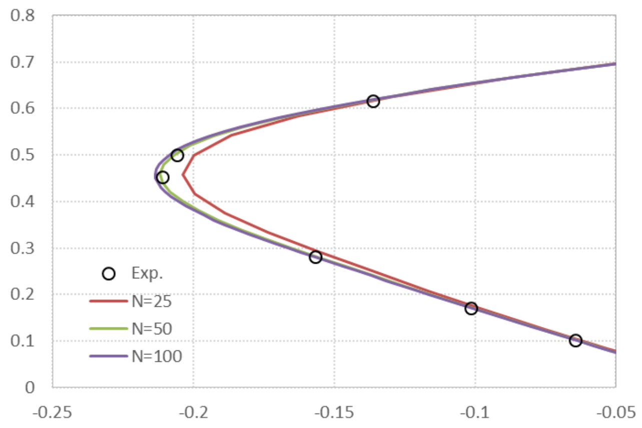


Grid: 100 x 100
SOR tolerance: 0.0001
SOR coefficient: 1.9

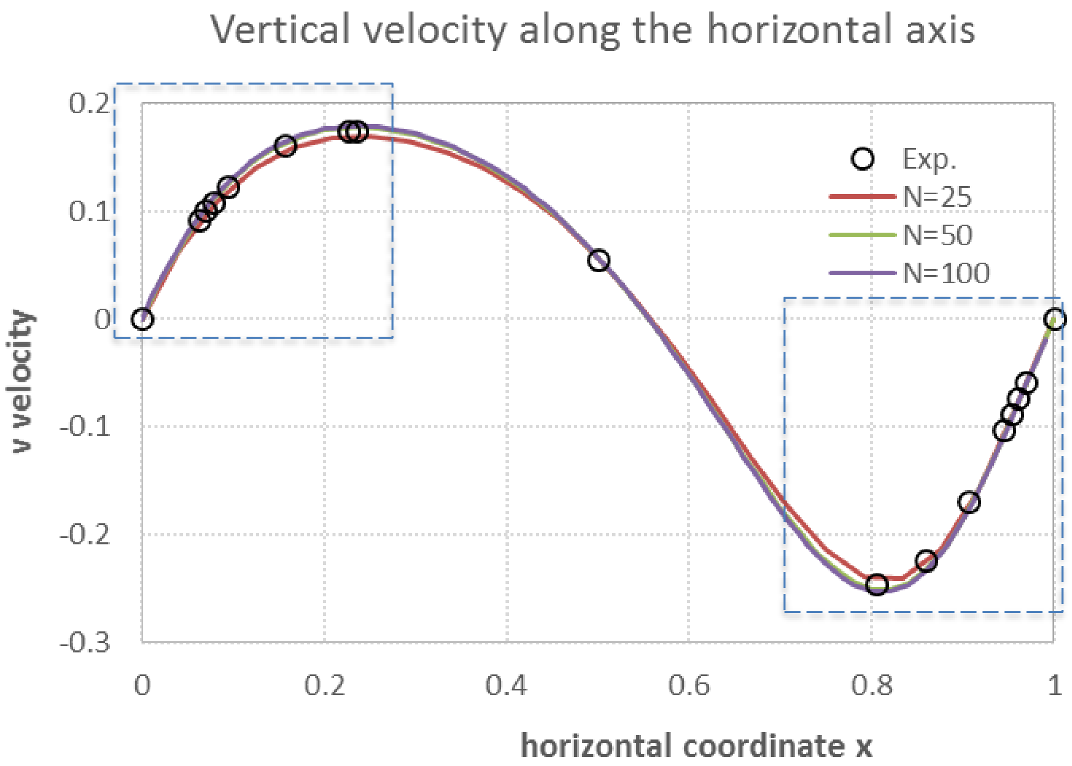
Time evolution from $t = 0$ to $t = 0.50$

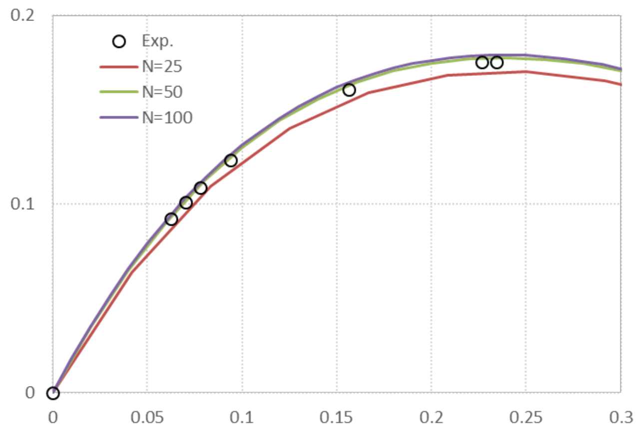
Comparison with exp. data at Re=100 (I)

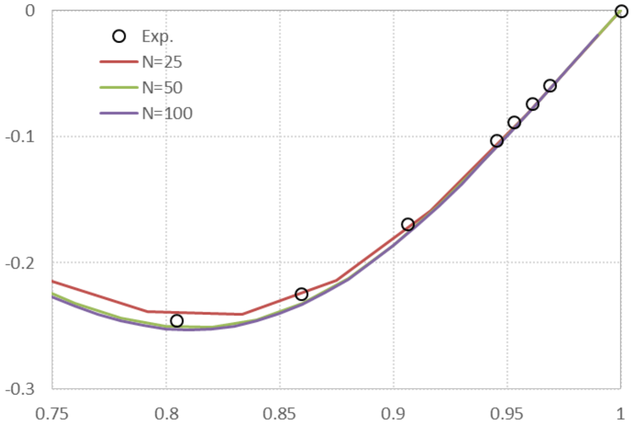




Comparison with exp. data at Re=100 (II)

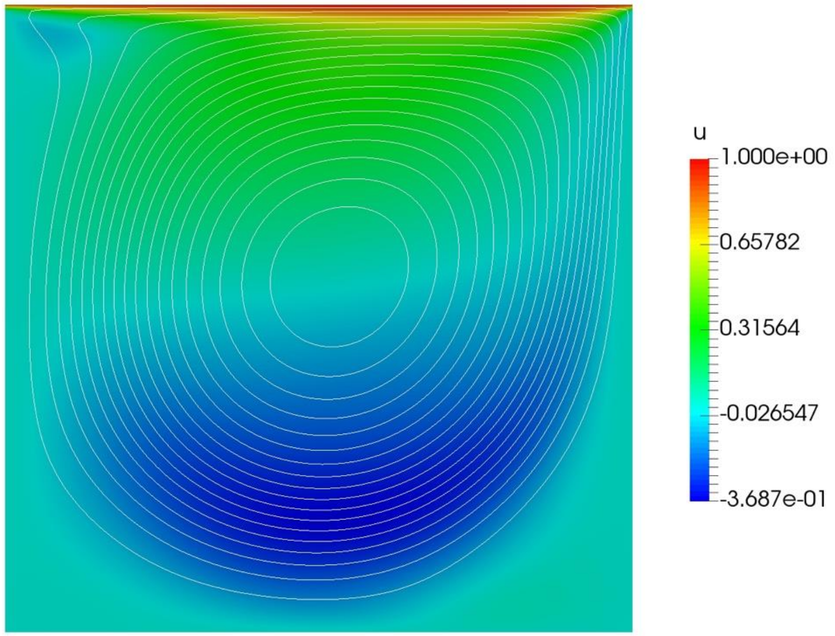




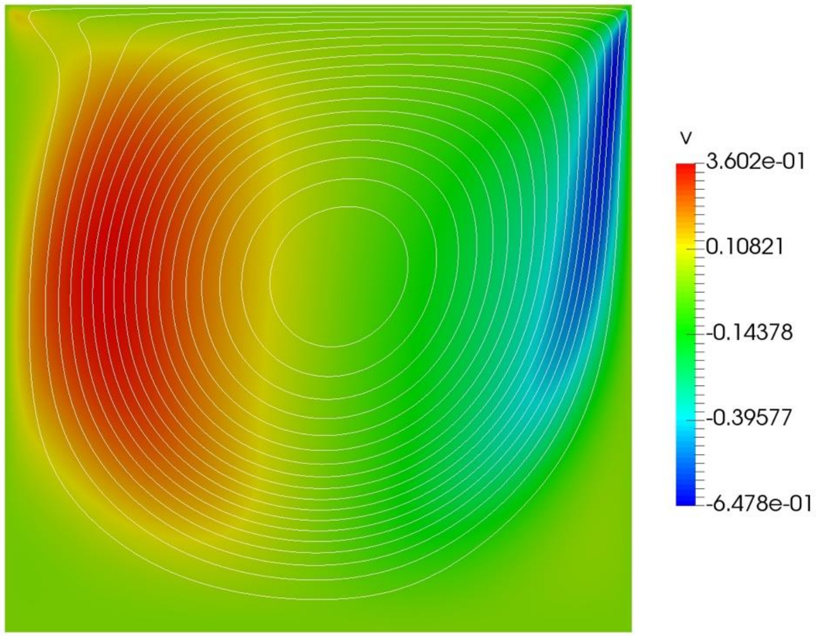


Steady state solution Re=1000

u velocity + streamlines

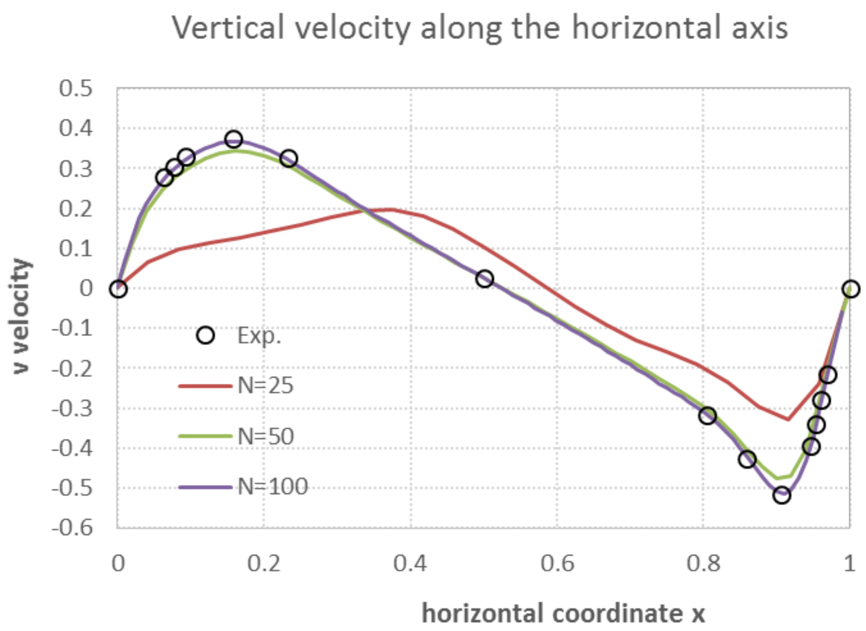


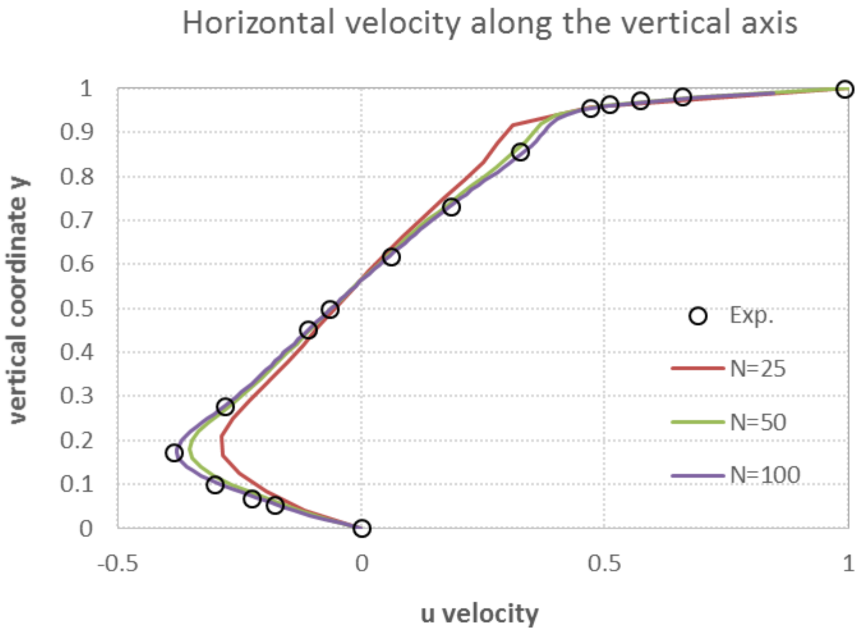
v velocity + streamlines



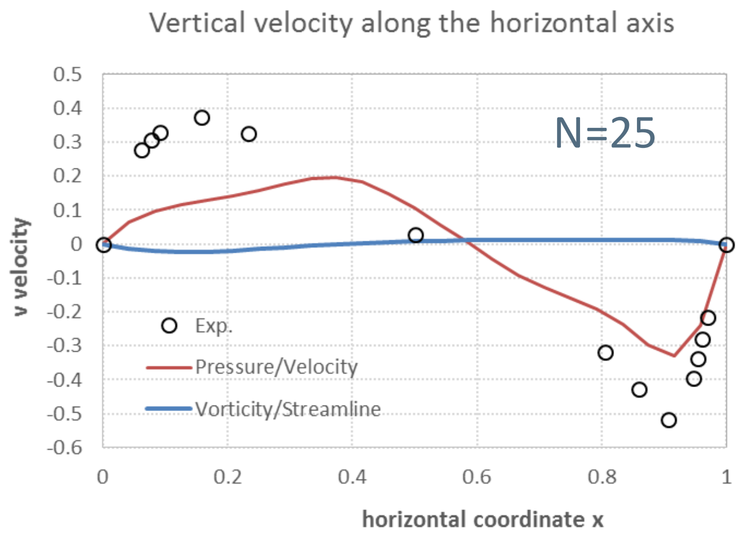
Grid: 100 x 100
SOR tolerance: 0.0001
SOR coefficient: 1.9

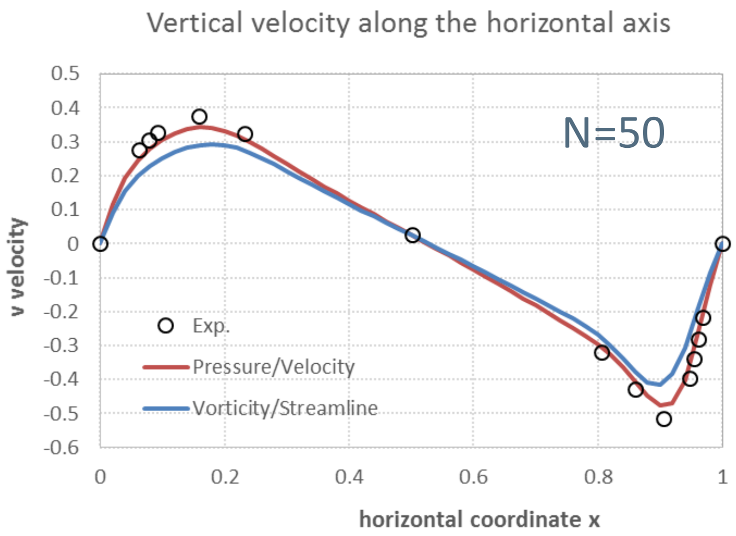
Comparison with exp. data at Re=1000

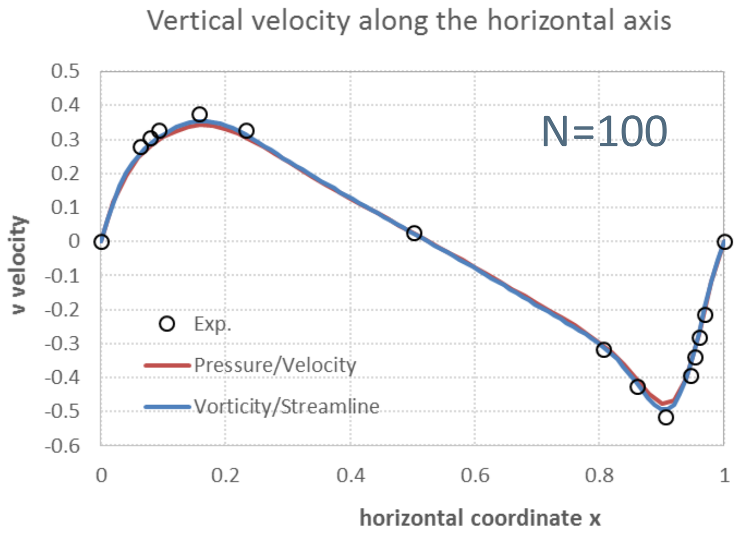




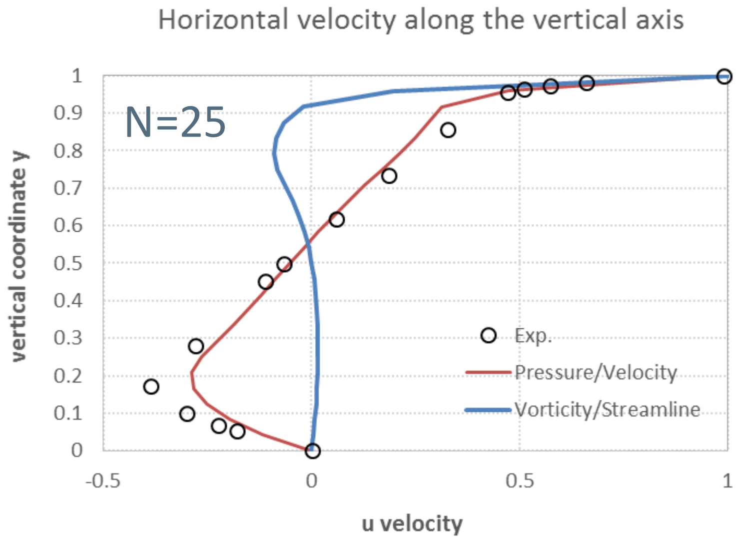
Comparison with vorticity/streamline, Re=1000 (I)

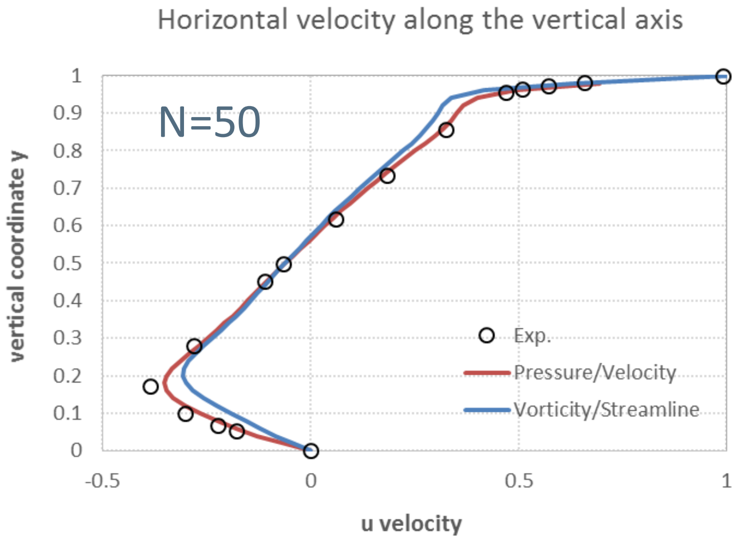


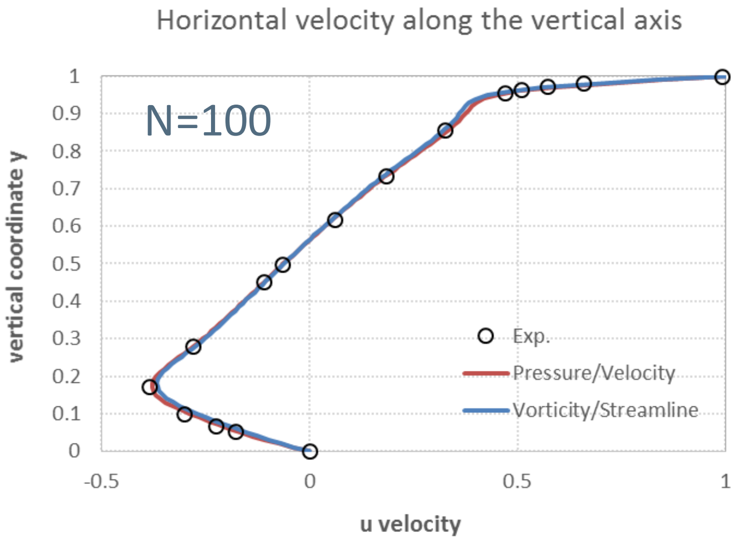




Comparison with vorticity/streamline, Re=1000 (II)







Outline

1. **Mathematical formulation**
2. **Numerical formulation**
   a. mesh
   b. finite volume formulation of momentum equations
   c. Poisson equation for pressure
   d. correction on velocity
   e. boundary conditions
3. **Experimental data**
4. **Results**
   a. comparison with experimental data
   b. grid sensitivity
5. **Final comments**

Final comments

- A numerical code for solving the Navier-Stokes equations in the pressure/velocity formulation has been implemented in MATLAB(R)
- The code was based on the finite volume discretization on a staggered grid
- The Projection Method was adopted for managing the coupling between pressure and velocity
- The driven-cavity problem has been solved at different Reynolds' numbers
- The numerical results have been compared with experimental measurements
- The sensitivity of solution with the mesh has been assessed

Suggested exercises/extensions (I)

- [EASY] Add the equation of a passive (dimensionless) scalar ϕ (for example dimensionless temperature)

$$\frac{\partial \phi}{\partial t} + \left(u \frac{\partial \phi}{\partial x} + v \frac{\partial \phi}{\partial y} \right) = \Gamma \left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} \right)$$

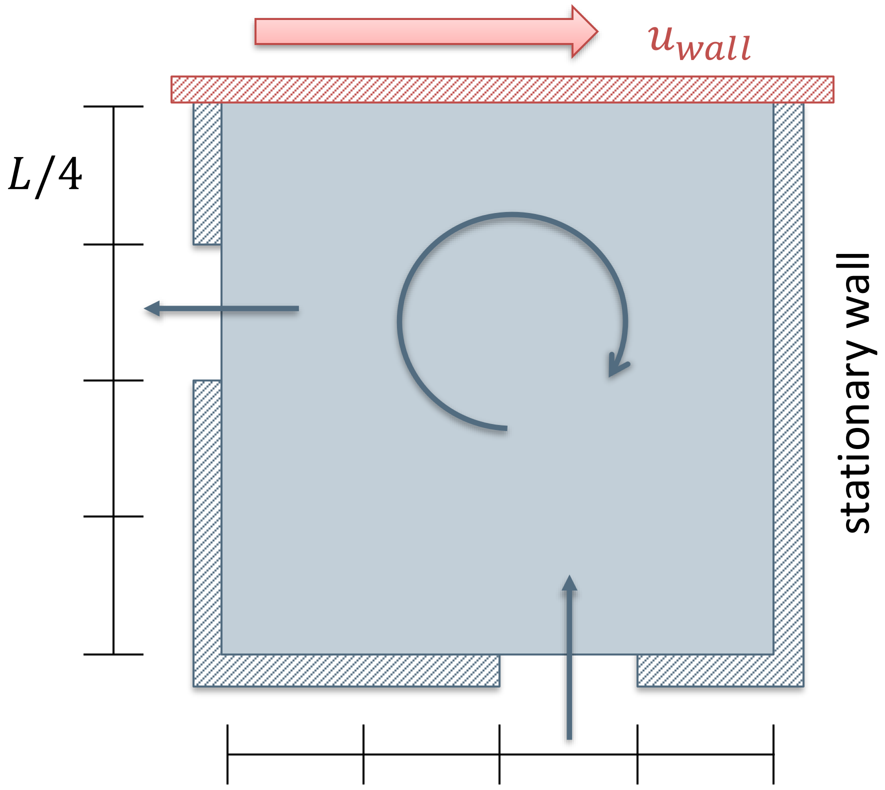
Boundary conditions
$$\begin{cases} \phi_{north} = 0 \\ \phi_{south} = 1 \\ \phi_{east} = 0 \\ \phi_{west} = 0 \end{cases}$$

Initial conditions
$$\phi(t = 0) = 0$$

Test different values of the diffusion coefficient, starting with $\Gamma = 0.1\ m^2/s$ as reference value

Suggested exercises/extensions (II)

- [HARD] Modify the boundary conditions in order to account for an inlet stream from the south boundary and an outlet stream from the west boundary, according to what reported in the picture



Outlet boundary conditions

$$\begin{cases} \dfrac{\partial u}{\partial x} = 0 \\ \dfrac{\partial v}{\partial x} = 0 \end{cases}$$

Inlet boundary conditions

$$\begin{cases} u = 0 \\ v = v_{inlet} \end{cases}$$

Questions?