Computational Fluid Dynamics
http://www.nd.edu/~gtryggva/CFD-Course/

Solving the Navier-Stokes Equations in Primitive Variables

Grétar Tryggvason
Spring 2011

Computational Fluid Dynamics
Outline

The projection method—review
Methods for the Navier-Stokes Equations
- Moin and Kim
- Bell, et al

Colocated grids
Boundary conditions

Computational Fluid Dynamics
Discretization in time

Summary of discrete vector equations

$$\frac{\mathbf{u}^{n+1}_{i,j}-\mathbf{u}^{n}_{i,j}}{\Delta t}=-\mathbf{A}^{n}_{i,j}-\nabla_h P_{i,j}+\mathbf{D}^{n}_{i,j}$$ Evolution of the velocity

$$\nabla_h \cdot \mathbf{u}^{n+1}_{i,j}=0$$ Constraint on velocity

No explicit equation for the pressure!

Computational Fluid Dynamics
Discretization in time

Split

$$\frac{\mathbf{u}^{n+1}_{i,j}-\mathbf{u}^{n}_{i,j}}{\Delta t}=-\mathbf{A}^{n}_{i,j}-\nabla_h P_{i,j}+\mathbf{D}^{n}_{i,j}$$

into

$$\frac{\mathbf{u}^{t}_{i,j}-\mathbf{u}^{n}_{i,j}}{\Delta t}=-\mathbf{A}^{n}_{i,j}+\mathbf{D}^{n}_{i,j} \quad\Rightarrow\quad \mathbf{u}^{t}_{i,j}=\mathbf{u}^{n}_{i,j}+\Delta t\left(-\mathbf{A}^{n}_{i,j}+\mathbf{D}^{n}_{i,j}\right)$$

and

$$\frac{\mathbf{u}^{n+1}_{i,j}-\mathbf{u}^{t}_{i,j}}{\Delta t}=-\nabla_h P_{i,j} \quad\Rightarrow\quad \mathbf{u}^{n+1}_{i,j}=\mathbf{u}^{t}_{i,j}-\Delta t\,\nabla_h P_{i,j}$$

by introducing the temporary velocity $\mathbf{u}^t$

Projection Method

Computational Fluid Dynamics
Discretization in time

To derive an equation for the pressure we take the divergence of

$$\mathbf{u}^{n+1}_{i,j}=\mathbf{u}^{t}_{i,j}-\Delta t\,\nabla_h P_{i,j}$$

and use the mass conservation equation

$$\nabla_h \cdot \mathbf{u}^{n+1}_{i,j}=0$$

The result is

$$\nabla_h \cdot \cancelto{0}{\mathbf{u}^{n+1}_{i,j}}=\nabla_h \cdot \mathbf{u}^{t}_{i,j}-\Delta t\,\nabla_h \cdot \nabla_h P_{i,j}$$

$$\longrightarrow \quad \nabla^2_h P_{i,j}=\frac{1}{\Delta t}\nabla_h \cdot \mathbf{u}^{t}_{i,j}$$

Computational Fluid Dynamics
Discretization in time

1. Find a temporary velocity using the advection and the diffusion terms only:

$$\mathbf{u}^{t}_{i,j}=\mathbf{u}^{n}_{i,j}+\Delta t\left(-\mathbf{A}^{n}_{i,j}+\mathbf{D}^{n}_{i,j}\right)$$

2. Find the pressure needed to make the velocity field incompressible

$$\nabla^2_h P_{i,j}=\frac{1}{\Delta t}\nabla_h \cdot \mathbf{u}^{t}_{i,j}$$

3. Correct the velocity by adding the pressure gradient:

$$\mathbf{u}^{n+1}_{i,j}=\mathbf{u}^{t}_{i,j}-\Delta t\,\nabla_h P_{i,j}$$

Computational Fluid Dynamics
Algorithm

Initial field given

Determine u, v boundary conditions

Advect $\mathbf{u}^t_{i,j} = \mathbf{u}^n_{i,j} + \Delta t\left(-\mathbf{A}^n_{i,j} + \mathbf{D}^n_{i,j}\right)$

Poisson equation for $P_{i,j}$ (iteration)

Projection $\mathbf{u}^{n+1}_{i,j} = \mathbf{u}^t_{i,j} - \Delta t\, \nabla_h P_{i,j}$

t=t+Δt

Computational Fluid Dynamics

Forward in time, centered in space (summary):

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\Delta t} = -\mathbf{A}\left(\boldsymbol{u}^n\right) + \nu\nabla^2\boldsymbol{u}^n$$

$$\nabla^2 p = \frac{1}{\Delta t}\nabla\cdot\boldsymbol{u}^*$$

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^* - \Delta t\nabla p$$

Time step limitations

$$\Delta t \le \frac{2\nu}{U^2} \quad \& \quad \Delta t \le \frac{1}{4}\frac{h^2}{\nu}$$

where

$$U^2 = \max(u^2 + v^2)$$

Computational Fluid Dynamics

Forward in time, centered in space:

$$\Delta t_{adv} \le \frac{2\nu}{U^2} \quad \& \quad \Delta t_{diff} \le \frac{1}{4}\frac{h^2}{\nu}$$

Define

$$\tau = \frac{L}{U}$$

Therefore the nondimensional time step is:

$$\Delta t' = \frac{\Delta t}{\tau} \le \frac{2\nu}{U^2}\frac{U}{L} = \frac{2}{\mathrm{Re}} \quad \text{and} \quad \Delta t' = \frac{\Delta t}{\tau} \le \frac{1}{4}\frac{h^2}{\nu}\frac{U}{L} = \frac{1}{4}\left(\frac{h}{L}\right)^2 \mathrm{Re}$$

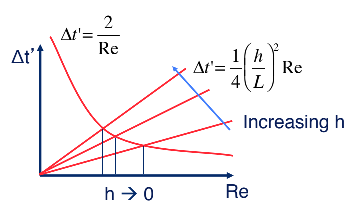
And Δt → 0 for Re → 0 and Re → ∞

Computational Fluid Dynamics

Forward in time, centered in space:

Δt → 0 for Re → 0 and Re → ∞

What is the maximum timestep?



$$\mathrm{Re}_{\Delta t'\max} = \sqrt{8}\frac{L}{h}$$

and

$$\Delta t'_{\max} = \frac{1}{\sqrt{2}}\frac{h}{L}$$

Computational Fluid Dynamics

Advanced Solvers

For low Re, use implicit methods for diffusion term
For high Re, use stable advection schemes

Combine both for schemes intended for all Re

Computational Fluid Dynamics

Fully Implicit

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = \frac{1}{2}\left[-\left(\mathbf{A}(\mathbf{u}^n) + \mathbf{A}(\mathbf{u}^{n+1})\right) + \nu\left(\nabla_h^2\mathbf{u}^n + \nabla_h^2\mathbf{u}^{n+1}\right)\right] - \nabla p$$

$$\nabla\cdot\mathbf{u}^{n+1} = 0$$

Solve by iteration

Rarely used due to the complications of the nonlinear system that must be solved for the advection terms

Predictor-Corrector

A second order method can be developed by first taking a forward step, then a backward step and average the results:

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\Delta t} = -\nabla \boldsymbol{u}^n \boldsymbol{u}^n + \nu \nabla^2 \boldsymbol{u}^n$$

$$\nabla^2 p^* = \frac{1}{\Delta t} \nabla \cdot \boldsymbol{u}^*$$

$$\boldsymbol{u}^1 = \boldsymbol{u}^* - \Delta t \nabla p^*$$

Backward step using the predicted velocity:

$$\frac{\boldsymbol{u}^{**} - \boldsymbol{u}^1}{\Delta t} = -\nabla \boldsymbol{u}^1 \boldsymbol{u}^1 + \nu \nabla^2 \boldsymbol{u}^1$$

$$\nabla^2 p^{**} = \frac{1}{\Delta t} \nabla \cdot \boldsymbol{u}^{**}$$

$$\boldsymbol{u}^2 = \boldsymbol{u}^{**} - \Delta t \nabla p^{**}$$

Then average the results

$$\boldsymbol{u}^{n+1} = \frac{1}{2}\left(\boldsymbol{u}^n + \boldsymbol{u}^2\right)$$

Adam-Bashford/Crank-Nicolson

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = -\left(\frac{3}{2}\mathbf{A}(\mathbf{u}^n) - \frac{1}{2}\mathbf{A}(\mathbf{u}^{n-1})\right) + \frac{\nu}{2}\left(\nabla_h^2 \mathbf{u}^n + \nabla_h^2 \mathbf{u}^{n+1}\right) - \nabla p$$

$$\nabla \cdot \mathbf{u}^{n+1} = 0$$

Split:

$$\frac{\tilde{\mathbf{u}} - \mathbf{u}^n}{\Delta t} = -\left(\frac{3}{2}\mathbf{A}(\mathbf{u}^n) - \frac{1}{2}\mathbf{A}(\mathbf{u}^{n-1})\right) + \frac{\nu}{2}\nabla_h^2 \mathbf{u}^n$$

$$\left.\begin{array}{l} \dfrac{\mathbf{u}^{n+1} - \tilde{\mathbf{u}}}{\Delta t} = -\nabla p + \dfrac{\nu}{2}\nabla_h^2 \mathbf{u}^{n+1} \\ \nabla \cdot \mathbf{u}^{n+1} = 0 \end{array}\right\} \quad \nabla_h^2 p = \frac{1}{\Delta t}\nabla \cdot \tilde{\mathbf{u}}$$

The correction equation is implicit and must be solved by an iteration in the same way as the pressure equation

Method of Kim and Moin (JCP 59 (1985), 8-23)

$$\frac{\tilde{\mathbf{u}} - \mathbf{u}^n}{\Delta t} = -\left(\frac{3}{2}\mathbf{A}(\mathbf{u}^n) - \frac{1}{2}\mathbf{A}(\mathbf{u}^{n-1})\right) + \frac{\nu}{2}\left(\nabla_h^2 \mathbf{u}^n + \nabla_h^2 \tilde{\mathbf{u}}\right)$$

$$\left.\begin{array}{l} \dfrac{\mathbf{u}^{n+1} - \tilde{\mathbf{u}}}{\Delta t} = -\nabla \phi \\ \nabla \cdot \mathbf{u}^{n+1} = 0 \end{array}\right\} \quad \nabla_h^2 \phi = \frac{1}{\Delta t}\nabla \cdot \tilde{\mathbf{u}}$$

The first equation is implicit and must be solved by an iteration in the same way as the pressure equation

Notice that Φ is not exactly p. Adding the first two equations gives

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = -\left(\frac{3}{2}\mathbf{A}(\mathbf{u}^n) - \frac{1}{2}\mathbf{A}(\mathbf{u}^{n-1})\right) + \frac{\nu}{2}\left(\nabla_h^2 \mathbf{u}^n + \nabla_h^2 \mathbf{u}^{n+1}\right) + \underbrace{\frac{\nu}{2}\left(\nabla_h^2 \tilde{\mathbf{u}} - \nabla_h^2 \mathbf{u}^{n+1}\right) - \nabla \phi}_{\nabla p}$$

Where we have added and subtracted an implicit diffusion term.

Using $\dfrac{\mathbf{u}^{n+1} - \tilde{\mathbf{u}}}{\Delta t} = -\nabla \phi$

we can rewrite the last terms as:

$$\frac{\nu}{2}\left(\nabla_h^2 \tilde{\mathbf{u}} - \nabla_h^2 \mathbf{u}^{n+1}\right) - \nabla \phi = \frac{\nu}{2}\nabla_h^2 \phi - \nabla \phi = \nabla p$$

Method of Bell, Colella and Glaz (JCP 85 (1989), 7-83)

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = -\mathbf{A}(\mathbf{u}^{n+1/2}) + \frac{\nu}{2}\left(\nabla_h^2 \mathbf{u}^n + \nabla_h^2 \mathbf{u}^{n+1}\right) - \nabla p$$

$$\nabla \cdot \mathbf{u}^{n+1} = 0$$

A Godunov method is used for the advection terms

A complete Runge-Kutta time integration (Weinan E.)

First a half step:

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\frac{1}{2}\Delta t} = -\nabla \boldsymbol{u}^n \boldsymbol{u}^n + \nu \nabla^2 \boldsymbol{u}^n$$

$$\nabla^2 p^* = \frac{2}{\Delta t} \nabla \cdot \boldsymbol{u}^*$$

$$\boldsymbol{u}^1 = \boldsymbol{u}^* - \frac{\Delta t}{2} \nabla p^*$$

continue for the second step:

$$\frac{\boldsymbol{u}^{**} - \boldsymbol{u}^n}{\frac{1}{2}\Delta t} = -\nabla \boldsymbol{u}^1 \boldsymbol{u}^1 + \nu \nabla^2 \boldsymbol{u}^1$$

$$\nabla^2 p^{**} = \frac{2}{\Delta t} \nabla \cdot \boldsymbol{u}^{**}$$

$$\boldsymbol{u}^2 = \boldsymbol{u}^{**} - \frac{\Delta t}{2} \nabla p^{**}$$

Computational Fluid Dynamics

A complete Runge-Kutta time integration (continued)

Take a full step using the predicted velocity

$$\frac{\boldsymbol{u}^{***}-\boldsymbol{u}^{n}}{\Delta t}=-\nabla\boldsymbol{u}^{2}\boldsymbol{u}^{2}+\nu\nabla^{2}\boldsymbol{u}^{2}$$

$$\nabla^{2}p^{***}=\frac{1}{\Delta t}\nabla\cdot\boldsymbol{u}^{***}$$

$$\boldsymbol{u}^{3}=\boldsymbol{u}^{***}-\Delta t\nabla p^{***}$$

Then compute $k=\Delta t\left(-\nabla\boldsymbol{u}^{3}\boldsymbol{u}^{3}+\nu\nabla^{2}\boldsymbol{u}^{3}\right)$

And finally

$$\boldsymbol{u}^{+}=\tfrac{1}{3}\left(-\boldsymbol{u}^{n}+\boldsymbol{u}^{1}+2\boldsymbol{u}^{2}+\boldsymbol{u}^{3}\right)+\tfrac{1}{6}k$$

$$\nabla^{2}p^{+}=\frac{1}{\Delta t}\nabla\cdot\boldsymbol{u}^{+}$$

$$\boldsymbol{u}^{n+1}=\boldsymbol{u}^{+}-\Delta t\nabla p^{+}$$

Computational Fluid Dynamics

Simplified Fourth order method

$$\frac{\boldsymbol{u}^{*}-\boldsymbol{u}^{n}}{\frac{1}{2}\Delta t}=-\nabla\boldsymbol{u}^{n}\boldsymbol{u}^{n}+\nu\nabla^{2}\boldsymbol{u}^{n}$$

$$\frac{\boldsymbol{u}^{**}-\boldsymbol{u}^{*}}{\frac{1}{2}\Delta t}=-\nabla\boldsymbol{u}^{*}\boldsymbol{u}^{*}+\nu\nabla^{2}\boldsymbol{u}^{*}$$

$$\frac{\boldsymbol{u}^{***}-\boldsymbol{u}^{n}}{\Delta t}=-\nabla\boldsymbol{u}^{*}\boldsymbol{u}^{*}+\nu\nabla^{2}\boldsymbol{u}^{*}$$

$$\boldsymbol{u}^{+}=\tfrac{1}{3}\left(-\boldsymbol{u}^{n}+\boldsymbol{u}^{*}+2\boldsymbol{u}^{**}+\boldsymbol{u}^{***}\right)+\frac{\Delta t}{6}\left(-\nabla\boldsymbol{u}^{***}\boldsymbol{u}^{***}+\nu\nabla^{2}\boldsymbol{u}^{***}\right)$$

$$\nabla^{2}p^{+}=\frac{1}{\Delta t}\nabla\cdot\boldsymbol{u}^{+}$$

$$\boldsymbol{u}^{n+1}=\boldsymbol{u}^{+}-\Delta t\nabla p^{+}$$

Computational Fluid Dynamics

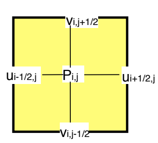
Colocated grids

Computational Fluid Dynamics

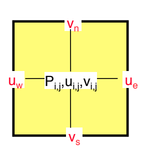
Colocated grids
Although staggered grids have been very successful, in some cases it is desirable to use co-located (or colocated) grids where all variables are located at the same physical point.

Computational Fluid Dynamics

Staggered grids



Colocated grids



All variables are stored at the same location

Computational Fluid Dynamics

$$\left.\begin{aligned}&u_{i,j}^{*}=u_{i,j}-\Delta tA(u_{i,j}^{n})\\&u_{i,j}^{n+1}=u_{i,j}^{*}-\frac{\Delta t}{h}(p_e-p_w)\end{aligned}\right\}\text{ Split equations for the } u \text{ velocity}$$

$$u_e^{n+1}-u_w^{n+1}+v_n^{n+1}-v_s^{n+1}=0$$

First idea: use averaging for the variables on the edges:

$$p_e=\frac{1}{2}\left(p_{i+1,j}+p_{i,j}\right)\qquad u_e=\frac{1}{2}\left(u_{i+1,j}+u_{i,j}\right)$$

$$u_{i,j}^{n+1}=u_{i,j}^{*}-\frac{\Delta t}{h}\left(\frac{1}{2}\left(p_{i+1,j}+\cancel{p_{i,j}}\right)-\frac{1}{2}\left(\cancel{p_{i,j}}+p_{i-1,j}\right)\right)$$

$$u_{i,j}^{n+1}=u_{i,j}^{*}-\frac{\Delta t}{2h}\left(p_{i+1,j}-p_{i-1,j}\right)$$

Substituting

$$u_e = \frac{1}{2}\left(u_{i+1,j} + u_{i,j}\right) \quad \text{and} \quad u_{i,j}^{n+1} = u_{i,j}^{*} - \frac{\Delta t}{2h}\left(p_{i+1,j} - p_{i,j-1}\right)$$

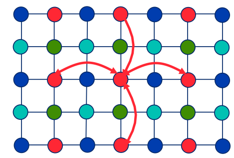
into

$$u_e^{n+1} - u_w^{n+1} + v_n^{n+1} - v_s^{n+1} = 0$$

yields

$$p_{i+2,j} + p_{i-2,j} + p_{i,j+2} + p_{i,j-2} - 4p_{i,j} = \frac{2h}{\Delta t}\left(u_{i+1,j}^{*} - u_{i-1,j}^{*} + v_{i,j+1}^{*} - v_{i,j-1}^{*}\right)$$

A straight forward application discretization on colocated grids results in a very wide stencil for the pressure



The pressure points are also uncoupled and the pressure field can develop oscillations

The remedy is to find the pressures that make the edge velocities incompressible

Rhie and Chow. AIAA Journal. 21 (1983), 1525-1532.

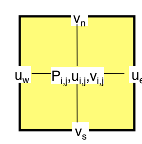
The Rhie and Chow method

Instead of interpolating (the final velocity)

$$u_e^{n+1} = \frac{1}{2}\left(u_{i+1,j}^{n+1} + u_{i,j}^{n+1}\right)$$

interpolate (the intermediate velocity)

$$u_e^{*} = \frac{1}{2}\left(u_{i+1,j}^{*} + u_{i,j}^{*}\right)$$



and then find

$$u_e^{n+1} = u_e^{*} - \frac{\Delta t}{h}\left(p_{i+1,j} - p_{i,j}\right)$$

In effect, “pretend” we are using a staggered grid!

Substitute

$$u_e^{n+1} = u_e^{*} - \frac{\Delta t}{h}\left(p_{i+1,j} - p_{i,j}\right) \quad \text{where} \quad u_e^{*} = \frac{1}{2}\left(u_{i+1,j}^{*} + u_{i,j}^{*}\right)$$

into

$$u_e^{n+1} - u_w^{n+1} + v_n^{n+1} - v_s^{n+1} = 0$$

giving

$$\left(u_e^{*} - \frac{\Delta t}{h}\left(p_{i+1,j} - p_{i,j}\right)\right) - \left(u_w^{*} - \frac{\Delta t}{h}\left(p_{i,j} - p_{i-1,j}\right)\right)$$
$$+\left(v_n^{*} - \frac{\Delta t}{h}\left(p_{i,j+1} - p_{i,j}\right)\right) - \left(v_s^{*} - \frac{\Delta t}{h}\left(p_{i,j} - p_{i,j-1}\right)\right) = 0$$

Rearrange:

$$p_{i+1,j} + p_{i-1,j} + p_{i,j+1} + p_{i,j-1} - 4p_{i,j} = \frac{h}{\Delta t}\left(u_e^{*} - u_w^{*} + v_n^{*} - v_s^{*}\right)$$

Substituting for the velocities:

$$p_{i+1,j} + p_{i-1,j} + p_{i,j+1} + p_{i,j-1} - 4p_{i,j} = \frac{h}{2\Delta t}\left(u_{i+1,j}^{*} - u_{i-1,j}^{*} + v_{i,j+1}^{*} - v_{i,j-1}^{*}\right)$$

For the correction of the momentum equation we still use the average of the pressures

$$p_e = \frac{1}{2}\left(p_{i+1,j} + p_{i,j}\right)$$

giving

$$u_{i,j}^{n+1} = u_{i,j}^{*} - \frac{\Delta t}{2h}\left(p_{i+1,j} - p_{i-1,j}\right)$$

Computational Fluid Dynamics

The boundary conditions for the velocity are now very simple: The velocity at nodes on the wall is simply the wall velocity.

The pressure boundary is more complex:

Computational Fluid Dynamics

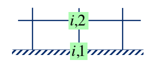
Find the pressure gradient by applying the Navier-Stokes equations at a point at the boundary

$$\rho\left(\cancel{\frac{\partial v}{\partial t}}+\cancel{u\frac{\partial v}{\partial x}}+\cancel{v\frac{\partial v}{\partial y}}\right)=-\frac{\partial p}{\partial y}+\mu\left(\cancel{\frac{\partial^2 v}{\partial x^2}}+\frac{\partial^2 v}{\partial y^2}\right)$$

At the wall, most of the terms are zero

$$\frac{\partial p}{\partial y}=\mu\frac{\partial^2 v}{\partial y^2} \rightarrow \frac{p_{i,2}-p_{i,1}}{h}=\mu\left(\frac{\partial^2 v}{\partial y^2}\right)_{i,1}$$

Evaluated by one-sided differences



$$\rightarrow \frac{p_{i,2}-p_{i,1}}{h}=\mu\frac{v_{i,1}^n-2v_{i,2}^n+v_{i,3}^n}{h^2}$$

Computational Fluid Dynamics

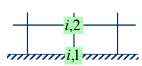
$$\frac{p_{i,2}-p_{i,1}}{h}=\mu\frac{v_{i,1}^n-2v_{i,2}^n+v_{i,3}^n}{h^2}$$

Define:

$$p_{i,2}-p_{i,1}=\frac{h}{\Delta t}v_{i,1}^*$$

We can write:

$$v_{i,1}^*=\frac{\Delta t}{h}\left(p_{i,2}-p_{i,1}\right)=\frac{\mu\Delta t}{h^2}\left(v_{i,1}^n-2v_{i,2}^n+v_{i,3}^n\right)$$



Computational Fluid Dynamics

Write the pressure equation for j=2

$$p_{i+1,2}+p_{i-1,2}+p_{i,3}+p_{i,1}-4p_{i,2}=\frac{h}{2\Delta t}\left(u_{i+1,2}^*-u_{i-1,2}^*+v_{i,3}^*-v_{i,1}^*\right)$$

And use

$$p_{i,2}-p_{i,1}=\frac{h}{\Delta t}v_{i,1}^*$$

For the pressure at j=1

Computational Fluid Dynamics

The algorithm is therefore:

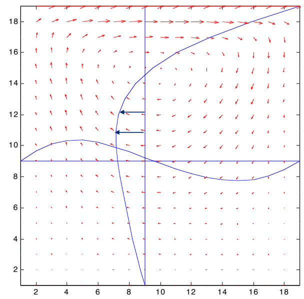
1. First find predicted velocities:
   $u_{i,j}^*=u_{i,j}-\Delta t A(\mathbf{u}^n)$ and $v_{i,j}^*=v_{i,j}-\Delta t A(\mathbf{v}^n)$
2. Find pressure by solving:
   $$p_{i+1,j}+p_{i-1,j}+p_{i,j+1}+p_{i,j-1}-4p_{i,j}=\frac{h}{2\Delta t}\left(u_{i+1,j}^*-u_{i-1,j}^*+v_{i,j+1}^*-v_{i,j-1}^*\right)$$
   suitably modified at the boundaries
3. Correct the velocities:
   $$u_{i,j}^{n+1}=u_{i,j}^*-\frac{\Delta t}{2h}\left(p_{i+1,j}-p_{i-1,j}\right) \text{ and } v_{i,j}^{n+1}=v_{i,j}^*-\frac{\Delta t}{2h}\left(p_{i,j+1}-p_{i,j-1}\right)$$

Computational Fluid Dynamics

```
% projection method in primitive variables on a colocated mesh
nx=19;ny=19;dt=0.0025;nstep=300;mu=0.1;maxit=100;beta=1.2;h=1/nx; time=0;
u=zeros(nx,ny);v=zeros(nx,ny);p=zeros(nx,ny); ut=zeros(nx,ny);vt=zeros(nx,ny); u(2:nx-1,ny)=1.0;

for is=1:nstep
  for i=2:nx-1,for j=2:ny-1
    ut(i,j)=u(i,j)+dt*(-(0.5/h)*(u(i,j)*(u(i+1,j)-u(i-1,j))+v(i,j)*(u(i,j+1)-u(i,j-1)))+(mu/h^2)*(u(i+1,j)+u(i-1,j)+u(i,j+1)+u(i,j-1)-4*u(i,j)));
    vt(i,j)=v(i,j)+dt*(-(0.5/h)*(u(i,j)*(v(i+1,j)-v(i-1,j))+v(i,j)*(v(i,j+1)-v(i,j-1)))+ (mu/h^2)*(v(i+1,j)+v(i-1,j)+v(i,j+1)+v(i,j-1)-4*v(i,j)));
  end,end

  vt(2:nx-1,1)=(mu*dt/h^2)*(v(2:nx-1,3)-2*v(2:nx-1,2));   vt(2:nx-1,ny)=(mu*dt/h^2)*(v(2:nx-1,ny-2)-2*v(2:nx-1,ny-1));
  ut(1,2:ny-1)=(mu*dt/h^2)*(u(3,2:ny-1)-2*u(2,2:ny-1));  ut(nx,2:ny-1)=(mu*dt/h^2)*(u(nx-2,2:ny-1)-2*u(nx-1,2:ny-1));

  for it=1:maxit                          % solving for pressure
     p(2:nx-1,1)=p(2:nx-1,2)+(h/dt)*vt(2:nx-1,1);   p(2:nx-1,ny)=p(2:nx-1,ny-1)-(h/dt)*vt(2:nx-1,ny);
     p(1,2:ny-1)=p(2,2:ny-1)+(h/dt)*ut(1,2:ny-1);   p(nx,2:ny-1)=p(nx-1,2:ny-1)-(h/dt)*ut(nx,2:ny-1);
     for i=2:nx-1,for j=2:ny-1
       p(i,j)=beta*0.25*(p(i+1,j)+p(i-1,j)+p(i,j+1)+p(i,j-1)-  (0.5*h/dt)*(ut(i+1,j)-ut(i-1,j)+vt(i,j+1)-vt(i,j-1)))+(1-beta)*p(i,j);
    end,end
              p(floor(nx/2),floor(ny/2))=0.0; % set the pressure in the center. Needed since bc is not incorporated into eq
  end

  u(2:nx-1,2:ny-1)=ut(2:nx-1,2:ny-1)-(0.5*dt/h)*(p(3:nx,2:ny-1)-p(1:nx-2,2:ny-1));
  v(2:nx-1,2:ny-1)=vt(2:nx-1,2:ny-1)-(0.5*dt/h)*(p(2:nx-1,3:ny)-p(2:nx-1,1:ny-2));

  time=time+dt
  hold off,quiver(flipud(rot90(u)),flipud(rot90(v)),'r');  hold on;axis([1 nx 1 ny]);axis square,pause(0.01)
end

plot(10*u(floor(ny/2),1:nx)+floor(nx/2),1:nx),hold on;  plot([1,nx],[floor(ny/2),floor(ny/2)]),plot([floor(nx/2),floor(nx/2)],[1,ny])
plot(10*v(1:nx,floor(ny/2))+floor(ny/2));axis square,axis([1,nx, 1,ny])
```

Computational Fluid Dynamics

Colocated grids



Computational Fluid Dynamics

Why colocated grids:

- Sometimes simpler for body fitted grids
- Easy to use methods for hyperbolic equations
- Easier to implement AMR
- Some people just don't like staggered grids!

Computational Fluid Dynamics

Boundary conditions
Inflow and outflow

Computational Fluid Dynamics

Boundary conditions

- Fully enclosed flow
  - Driven cavity
- Internal flow (inflow & outflow)
- External flow
  - Flow over bodies

Computational Fluid Dynamics

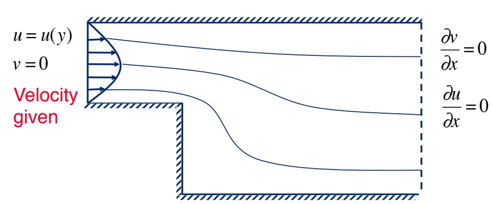
For fully enclosed domains, such as in the driven cavity problem, the application of the boundary conditions is relatively straight forward.

For domains with inflows it is usually reasonable to specify the velocity profile at the inlet

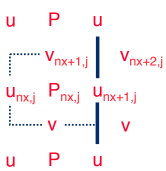
The major problem is how to handle outflow boundaries in such a way that the fluid leaves in a "physically plausible" way

Computational Fluid Dynamics

The Backward Facing Step Problem



CFD — Computational Fluid Dynamics



$$\frac{\partial u}{\partial x}=0$$ u(nx+1,j)=u(nx,j);

$$\frac{\partial v}{\partial x}=0$$ v(nx+2,1:ny+1)=v(nx+1,1:ny+1);

For the pressure, obtain an equation by applying conservation of mass to the control volume next to the boundary

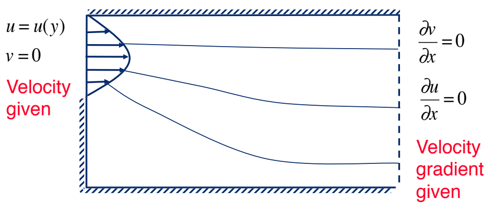
p(nx,j)=(1/2)*(p(nx,j+1)+p(nx,j-1)-(h/dt)*(v(nx+1,j+1)-v(nx+1,j)));

CFD — Computational Fluid Dynamics

Example

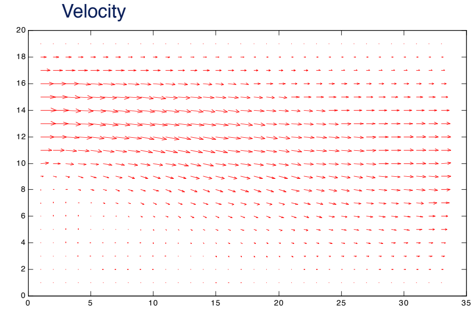
CFD — Computational Fluid Dynamics

Simplified Backward Facing Step Problem



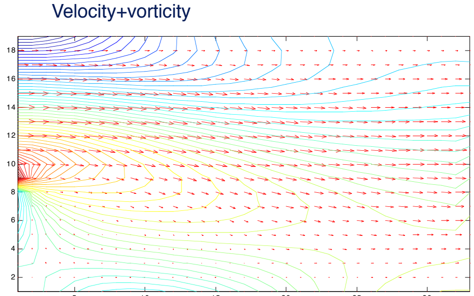
CFD — Computational Fluid Dynamics

Velocity



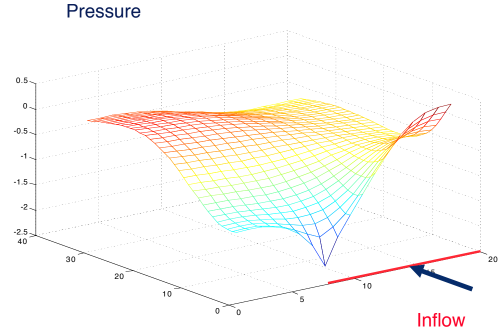
CFD — Computational Fluid Dynamics

Velocity+vorticity



CFD — Computational Fluid Dynamics

Pressure

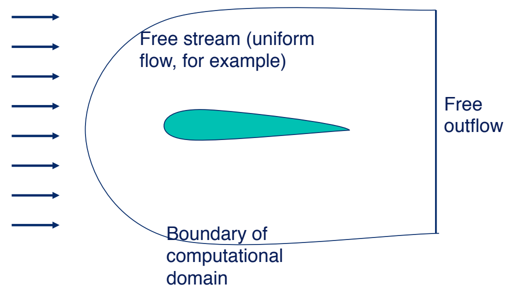


Computational Fluid Dynamics

Periodic Boundary conditions
In many cases it is possible to use periodic boundary conditions, where what flows out through one boundary reappears flowing in through the opposite boundary. Such conditions are particularly suitable for theoretical studies of idealized flows. For such boundaries it is easiest to specify the pressure drop, but by adjusting the pressure gradient it is possible to specify the volume flux

Computational Fluid Dynamics

External flow



Computational Fluid Dynamics

Other ways to deal with free-stream boundaries
- Include potential flow perturbation
- Compute flow from vorticity distribution
- Map the boundary at infinity to a finite distance

Fundamentally, the specification of the boundary conditions does not have a unique solution and is also faced by experimentalists. However, by taking the boundaries far away and checking the solution for the effect of moving the boundaries, good results can be obtained

Computational Fluid Dynamics

The two-dimensional programs developed in the project and shown here can be extended to fully three-dimensional flows in a relatively straight forward way, replacing u(i,j) by u(i,j,k), etc. The time required to run the code increases significantly and visualizing the output becomes more challenging.