Navier-Stokes Equations

Computer Session B7

Computational Fluid Dynamics (CFD)

The field of computational fluid dynamics (CFD) deals primarily with the numerical solution of the Navier-Stokes equations. This is a vast field involving both continuum mechanics, thermodynamics, mathematics, and computer science. The applications are many and ranges from optimizing the mixing of air and fuel in a turbine engine to predicting the stresses in the walls of human blood vessels. The grand theoretical challenge for CFD is however the understanding of *turbulence*. Turbulence is the highly chaotic flow pattern exhibited by fast moving fluids with low viscosity. Think of the irregular plume of smoke rising from a cigarette, for example. Physically, turbulence is caused by a combination of dissipation of energy into heat at the microscopic level, with a large transport of momentum at the macroscopic level. The basic measure of the tendency for a fluid to develop a turbulent flow is the dimensionless Reynolds number, defined by $\mathrm{Re} = UL/\nu$, where ν is the viscosity and U and L is a representative velocity and length scale, respectively. A high Reynolds number implies a turbulent flow, while a low implies a steady state laminar flow. Because turbulence occurs on all length scales, down to the smallest ($\mathrm{Re}^{-3/4}$) it is very difficult to simulate using finite elements on a perhaps coarse mesh with large spacing between the nodes. This is further complicated by the fact that turbulent flows are highly convective, which requires stabilization of the corresponding finite element methods. Moreover, turbulence is a peculiar phenomenon which can amplify itself over a period of time. Thus the Navier-Stokes equations are both nonlinear and time dependent. In this computer session we implement a simple, but somewhat heuristic, numerical method for the solution of the Navier-Stokes equations introduced by Chorin and Temam in 1967.

The Navier-Stokes Equations

The Navier-Stokes equations for a viscous incompressible fluid occupying a domain Ω reads: find the velocity $\boldsymbol{u}$ and the pressure p such that

$$\dot{\boldsymbol{u}} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nabla p - \nu \Delta \boldsymbol{u} = \boldsymbol{f}, \quad \text{in } \Omega \times I \tag{1a}$$

$$\nabla \cdot \boldsymbol{u} = 0, \quad \text{in } \Omega \times I \tag{1b}$$

$$\boldsymbol{u} = \boldsymbol{g}, \quad \text{in } \Gamma_D \times I \tag{1c}$$

$$\nu \boldsymbol{n} \cdot \nabla \boldsymbol{u} - p\boldsymbol{n} = \boldsymbol{0}, \quad \text{in } \Gamma_N \times I \tag{1d}$$

$$\boldsymbol{u}(\cdot, 0) = \boldsymbol{0}, \quad \text{in } \Omega \tag{1e}$$

where ν is the viscosity, and $\boldsymbol{f}$ is a given force. The boundary $\partial\Omega$ of Ω is divided into two parts Γ_D and Γ_N associated with the "no-slip" and the "do-nothing" boundary conditions (1c) and (1d). $\boldsymbol{g}$ is a given function describing the velocity on Γ_D. In the following, Ω will be a channel with a bluff body submerged into it. Γ_D denotes either the rigid walls of the channel, with $\boldsymbol{g} = \boldsymbol{0}$, or the inflow region, with $\boldsymbol{g}$ the inflow velocity profile. Γ_N denotes the outlet on which we assume the boundary condition $\nu \boldsymbol{n} \cdot \nabla \boldsymbol{u} - p\boldsymbol{n} = \boldsymbol{0}$, where $\boldsymbol{n}$ is the outward unit normal. As usual $\dot{v} = \partial_t v$ denotes time derivative and $I = (0, T]$ is the time interval with final time T.

In component form the momentum equation (1a) and the incompressibility constraint (1b) are given by

$$\dot{u}_1 + u_1 \frac{\partial u_1}{\partial x_1} + u_2 \frac{\partial u_1}{\partial x_2} + \frac{\partial p}{\partial x_1} - \nu \Delta u_1 = f_1 \tag{2}$$

$$\dot{u}_2 + u_1 \frac{\partial u_2}{\partial x_1} + u_2 \frac{\partial u_2}{\partial x_2} + \frac{\partial p}{\partial x_2} - \nu \Delta u_2 = f_2 \tag{3}$$

$$\frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} = 0 \tag{4}$$

Chorin's Projection Method

In strong form Chorin's projection method can be derived as follows. Discretizing the momentum equation (1a) in time using the Euler forward method we end up with the time stepping scheme

$$\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\delta t} + (\boldsymbol{u}^n \cdot \nabla)\boldsymbol{u}^n + \nabla \tilde{p} - \nu \Delta \boldsymbol{u}^n = \boldsymbol{f}^n, \quad n = 1, 2, 3, \ldots \tag{5}$$

where δt is the timestep and the superscript n indicates the iterate number. We shall return to explain the tilde on the pressure shortly. Now, adding and subtracting a tentative, or intermediate, velocity $\boldsymbol{u}^*$ in the discrete time derivative $\delta t^{-1}(\boldsymbol{u}^{n+1} - \boldsymbol{u}^n)$ we get

$$\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^* + \boldsymbol{u}^* - \boldsymbol{u}^n}{\delta t} + (\boldsymbol{u}^n \cdot \nabla)\boldsymbol{u}^n + \nabla \tilde{p} - \nu \Delta \boldsymbol{u}^n = \boldsymbol{f}^n \tag{6}$$

This equation holds if both

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\delta t} = -(\boldsymbol{u}^n \cdot \nabla)\boldsymbol{u}^n + \nu \Delta \boldsymbol{u}^n + \boldsymbol{f}^n \tag{7}$$

and

$$\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^*}{\delta t} = -\nabla \tilde{p} \tag{8}$$

hold simultaneously. The decomposition of (6) into (7) and (8) is called operator splitting. The advantage of doing this is that we get a decoupling of the diffusion and convection of the velocity, and the pressure acting to enforce the incompressibility constraint. Thus, if we know $\boldsymbol{u}^n$, then we can compute $\boldsymbol{u}^*$ from (7) without having to worry about (8). Further, taking the divergence of the last equation we have

$$\nabla \cdot \frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^*}{\delta t} = -\nabla \cdot \nabla \tilde{p} \tag{9}$$

but since we desire $\nabla \cdot \boldsymbol{u}^{n+1} = 0$ this reduces to

$$-\nabla \cdot \frac{\boldsymbol{u}^*}{\delta t} = -\Delta \tilde{p} \tag{10}$$

This can be viewed as a Poisson type equation for the pressure $\tilde{p}$ and is frequently referred to as the *pressure Poisson equation*. As a consequence, given $\boldsymbol{u}^*$ we can solve (10) to get a pressure $\tilde{p}$ which makes the next velocity $\boldsymbol{u}^{n+1}$ divergence free. Since $\tilde{p}$ is manufactured from the tentative velocity $\boldsymbol{u}^*$, it is not the real physical pressure p. Hence, we use a tilde to distinguish it from p. The actual computation of $\boldsymbol{u}^{n+1}$ is done by reusing (8) in the form

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^* - \delta t \nabla \tilde{p} \tag{11}$$

This line of reasoning leads us to the following algorithm:

Algorithm 1 Chorin's Projection Method

1: Given the initial condition $\boldsymbol{u}^0 = 0$.
2: **for** $n = 1, 2, 3, \ldots$ **do**
3: Obtain the tentative velocity $\boldsymbol{u}^*$ from

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\delta t} = -(\boldsymbol{u}^n \cdot \nabla)\boldsymbol{u}^n + \nu \Delta \boldsymbol{u}^n + \boldsymbol{f}^n \tag{12}$$

4: Solve the pressure Poisson equation

$$-\nabla \cdot \boldsymbol{u}^* = -\delta t \Delta \tilde{p} \tag{13}$$

5: Update the velocity

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^* - \delta t \nabla \tilde{p} \tag{14}$$

6: **end for**

Problem 1. ☆ How would (5) look if we used the Euler backward method?

Boundary Conditions. The boundary conditions for $\boldsymbol{u}^*$ and $\tilde{p}$ are not clear and has caused a lot of confusion and controversy. The simplest way of enforcing these is to put the Dirichlet velocity boundary conditions (1c) on $\boldsymbol{u}^*$, and a Neumann boundary condition $\boldsymbol{n} \cdot \nabla \tilde{p} = 0$ on the pressure. The exception is the outflow boundary condition (1d) which is imposed by assuming $\boldsymbol{n} \cdot \nabla \boldsymbol{u}^n = \boldsymbol{0}$ and $\tilde{p} = 0$. This generally means that $\boldsymbol{u}^{n+1}$ will not satisfy the velocity boundary conditions other than in a very vague sense. The cause of controversy is the zero Neumann boundary conditions for the pressure, which are not physical and leads to a poor quality of both $\tilde{p}$ and $\boldsymbol{u}^{n+1}$ near the boundary. This has raised questions of the value of the projection method. Numerous methods have been suggested to remedy this with, at least, partial success. However, the simplicity of the projection method has made it very popular and hard to replace.

Finite Element Approximation

In order to derive a numerical method we next apply finite elements to the equations of Algorithm 1. To this end, let $\mathcal{K} = \{K\}$ be a mesh of Ω into shape regular triangles K of size h_K, and let $h = \max_{K \in \mathcal{K}} h_K$ denote a global mesh size. Further,

let V_h be the space of all continuous piecewise linear functions on $\mathcal{K}$ with the usual basis $\{\varphi_i\}_{i=1}^N$ of hat functions. We shall use the hat functions to represent both the velocity and the pressure, that is,

$$u_1^n \approx \sum_{j=1}^N \xi_j^n \varphi_j, \qquad u_2^n \approx \sum_{j=1}^N \eta_j^n \varphi_j \tag{15}$$

$$\tilde{p} \approx \sum_{j=1}^N \theta_j \varphi_j \tag{16}$$

with a similar representation for $\boldsymbol{u}^*$.

Next we note that (7) can be considered as two decoupled equations for u_1^* and u_2^*, and we can thus write down a finite element formulation of (7) on the form

$$\boldsymbol{M}\boldsymbol{\xi}^* = \boldsymbol{M}\boldsymbol{\xi}^n - \delta t(\boldsymbol{C} + \nu \boldsymbol{A})\boldsymbol{\xi}^n + \boldsymbol{b}_1 \tag{17}$$

$$\boldsymbol{M}\boldsymbol{\eta}^* = \boldsymbol{M}\boldsymbol{\eta}^n - \delta t(\boldsymbol{C} + \nu \boldsymbol{A})\boldsymbol{\eta}^n + \boldsymbol{b}_2 \tag{18}$$

where the entries of the mass matrix $\boldsymbol{M}$ are given by the usual expression $\boldsymbol{M}_{ij} = (\varphi_j, \varphi_i)$, $i, j = 1, \ldots, N$. In practice $\boldsymbol{M}$ is often lumped for greater computational efficiency. The matrix $\boldsymbol{A}$ arises from the Laplace operator and is just the usual stiffness matrix with entries $\boldsymbol{A}_{ij} = (\nabla\varphi_j, \nabla\varphi_j)$, $i, j = 1, \ldots, N$. The matrix $\boldsymbol{C}$ stemming from the nonlinear term is a bit more complicated. With a slight abuse of notation its entries are given by

$$\boldsymbol{C}_{ij} = (\boldsymbol{u}^k \cdot \nabla\varphi_j, \varphi_i), \quad i, j = 1, \ldots, N \tag{19}$$

where $\boldsymbol{u}^k$ of course must be replaced by its space discrete counterpart. Note that $\boldsymbol{C}$ depends on the current velocity and must thus be reassembled at each timestep n. The vectors $\boldsymbol{b}_s$, $s = 1, 2$, contain contributions from the volume force $\boldsymbol{f}$ and $\boldsymbol{b}_{s,i} = (f_s, \varphi_i)$, $i = 1, \ldots, N$. In addition, when assembling care must be taken so that the boundary condition $\boldsymbol{u}^* = \boldsymbol{g}$ on Γ_D is properly enforced. This is a minor detail and we wait for a bit with showing how it is done.

Equation (10) is a standard Poisson equation, yielding the discrete form

$$\boldsymbol{A}\boldsymbol{\theta} = -(\boldsymbol{B}_1\boldsymbol{\xi}^* + \boldsymbol{B}_2\boldsymbol{\eta}^*)/\delta t \tag{20}$$

where $\boldsymbol{A}$ has the same form as above, and the entries of $\boldsymbol{B}_s$ are given by

$$\boldsymbol{B}_{s,ij} = (\partial_{x_s}\varphi_j, \varphi_i), \quad i, j = 1, \ldots, N, \ s = 1, 2 \tag{21}$$

To be able to solve the linear system (20) suitable adjustments for the boundary condition $\tilde{p} = 0$ on Γ_N has to be made to $\boldsymbol{A}$. Perhaps the simplest way of doing so is to add large values to the diagonal entries of $\boldsymbol{A}$ corresponding to nodes on Γ_N.

Finally, the discrete form of the update (11) is given by

$$\boldsymbol{M}\boldsymbol{\xi}^{n+1} = \boldsymbol{M}\boldsymbol{\xi}^{*} - \delta t \boldsymbol{B}_1 \boldsymbol{\theta} \tag{22}$$

$$\boldsymbol{M}\boldsymbol{\eta}^{n+1} = \boldsymbol{M}\boldsymbol{\eta}^{*} - \delta t \boldsymbol{B}_2 \boldsymbol{\theta} \tag{23}$$

The time step of the suggested scheme is limited by the use of the Euler forward method. For numerical stability it is necessary that the time step δt is of magnitude h/U for convection dominated flow $\nu < Uh$, and h_K^2/ν for diffusion dominated flow $\nu \geq Uh$.

Computer Implementation

We now turn to the practical implementation of our numerical method. The assembly of the matrices $\boldsymbol{A}$, $\boldsymbol{M}$, and $\boldsymbol{B}_s$ should by now be easy. We list the code.

```
function [A,M,Bx,By] = assemble(p,e,t)
nt=size(t,2);  np=size(p,2);
A=sparse(np,np);
M=zeros(np,1);
Bx=sparse(np,np);
By=sparse(np,np);
for i=1:nt
  % element nodes, coordinates, and area
  nodes = t(1:3,i);
  x=p(1,nodes);  y=p(2,nodes);
  area=polyarea(x,y);
  % hat function gradients
  b_=[y(2)-y(3); y(3)-y(1); y(1)-y(2)]/2/area;
  c_=[x(3)-x(2); x(1)-x(3); x(2)-x(1)]/2/area;
  % stiffness matrix
  A(nodes,nodes)=A(nodes,nodes)+(b_*b_'+c_*c_')*area;
  % lumped mass matrix (stored as a vector)
  M(nodes)=M(nodes)+[1; 1; 1]*area/3;
  % gradient matrices
```

```
  Bx(nodes,nodes)=Bx(nodes,nodes)+ones(3,1)*b_'*area/3;
  By(nodes,nodes)=By(nodes,nodes)+ones(3,1)*c_'*area/3;
end
```

Input is the usual point, edge and triangle matrix `p`, `e`, and `t`. Output is the assembled stiffness matrix `A`, the lumped mass matrix `M` stored as a vector, and the gradient matrices `Bx` and `By`. Further, given two vectors `xi` and `eta` containing the nodal values of the current velocity approximation, the assembly of the nonlinear matrix $\boldsymbol{C}$ can be done with a slight modification of the previous subroutine.

```
function C = reassemble(p,e,t,xi,eta)
nt=size(t,2);  np=size(p,2);
C=sparse(np,np);
for i=1:nt
  nodes=t(1:3,i);
  x=p(1,nodes);  y=p(2,nodes);
  % compute the mean velocity on the element
  u=mean( xi(nodes));
  v=mean(eta(nodes));
  area=polyarea(x,y);
  b_=[y(2)-y(3); y(3)-y(1); y(1)-y(2)]/2/area;
  c_=[x(3)-x(2); x(1)-x(3); x(2)-x(1)]/2/area;
  C(nodes,nodes)=C(nodes,nodes)+ones(3,1)*(u*b_'+v*c_')*area/3;
end
```

Putting the pieces together, we get the main routine.

```
function MyNSSolver()
clear all, close all
geom=channel();
[p,e,t]=initmesh(geom,'hmax',0.4);
[A,M,Bx,By]=assemble(p,e,t);
np=size(p,2);
xi=zeros(np,1); eta=zeros(np,1);
dt=0.01;  % time step
nu=0.005; % viscosity
[R,mask,g]=bcs(p,e);
for k=1:1000
  C=reassemble(p,e,t,xi,eta);
```

```
  xi =xi -dt*(nu*A+C)*xi ./M; xi =xi.*mask+g;
  eta=eta-dt*(nu*A+C)*eta./M; eta=eta.*mask;
  theta=(A+R)\-(Bx*xi+By*eta)/dt;
  xi =xi -dt*(Bx*theta)./M;
  eta=eta-dt*(By*theta)./M;
  pdeplot(p,e,t,'flowdata',[xi eta]),axis equal,pause(.1)
end
```

Here, the velocity boundary conditions are enforced with the use of two vectors `mask` and `g`, The entries of `mask` is either 0 or 1 signifying which nodes should be modified for the boundary conditions. The entries of `g` contains the corresponding nodal boundary values. The pressure boundary conditions are enforced by penalization as described before. Adding the diagonal matrix `R` to the pressure stiffness matrix `A` effectively zeros out selected nodal pressures on the boundary. The boundary data `mask`, `g`, and `R` is set with the next subroutine.

```
function [R,mask,g] = bcs(p,e)
np=size(p,2);
R=sparse(np,np);
mask=ones(np,1);
g=zeros(np,1);
bdrynodes=unique(union(e(1,:),e(2,:)));
for i=1:length(bdrynodes)
  node=bdrynodes(i);
  x=p(1,node);  y=p(2,node);
  if x<0.001
    g(node)=2.5*0.25*y*(4-y); % inflow velocity
  end
  if (x>19.999 & y>0.001 & y<3.999)
    R(node,node)=1.e+6; % outflow, p=0
  else
    mask(node)=0; % u=0
  end
end
```

Simulating the Flow Past a Sphere

Let us finally consider a specific application, namely, two-dimensional flow past a sphere submerged into a channel. The bounding box of the channel is rectangular and of dimension $[0,\ 20] \times [0,\ 4]$ with the lower left corner at origo. The submerged sphere has a radius of 0.75 and is centered at $(4, 2)$, cf. Figure 1.

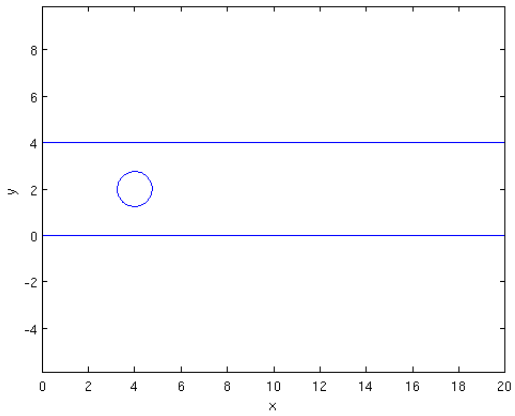

Figure 1: Geometry of channel and sphere.

The geometry matrix for the computational domain is given by

```
function geom = channel()
geom=[ 2  2  2  2    1    1    1    1
      20 20  0  0 3.25    4 4.75    4
      20  0  0 20    4 4.75    4 3.25
       0  4  0  0    2 1.25    2 2.75
       4  4  4  0 1.25    2 2.75    2
       1  1  0  1    0    0    0    0
       0  0  1  0    1    1    1    1
       0  0  0  0    4    4    4    4
       0  0  0  0    2    2    2    2
       0  0  0  0 0.75 0.75 0.75 0.75];
```

At the inlet $x = 0$ we assume a parabolic velocity profile of the form

$$\boldsymbol{g} = \begin{bmatrix} 0.625y(4-y) \\ 0 \end{bmatrix} \tag{24}$$

giving a maximum inflow velocity of 2.5. On the outflow $x = 20$ we employ the do-nothing boundary condition (1d). The remaining walls of the channel, including the surface of the sphere, are assumed to have no-slip boundary conditions (i.e., zero velocity).

This is a classical benchmark problem for fluid dynamics simulations and the main feature of the problem is that a vortex street forms behind the sphere if the viscosity of the fluid is small enough.

Problem 2. ✔ Plot the computational domain `geom=channel()` using either `pdegplot` or `pdemesh`.

Problem 3. ✔ Cut and paste the above code into an m-file `MyNSSolver` and run a simulation of the aforementioned problem with the default parameter settings.

Problem 4. ✔ In each time step plot the pressure. Note the poor quality of the pressure, particularly around the boundary. This is a consequence of the artificial Neumann boundary condition.

Problem 5. ✔ Run a sequence of simulations with varying viscosity ν from 0.1 to 0.005. In each run make 1000 timesteps using $\delta t = 0.01$. Study the transition from laminar to almost turbulent flow when you decrease ν. Make plots of the velocity. *Tip:* Save your final plots - these simulations are quite time consuming.