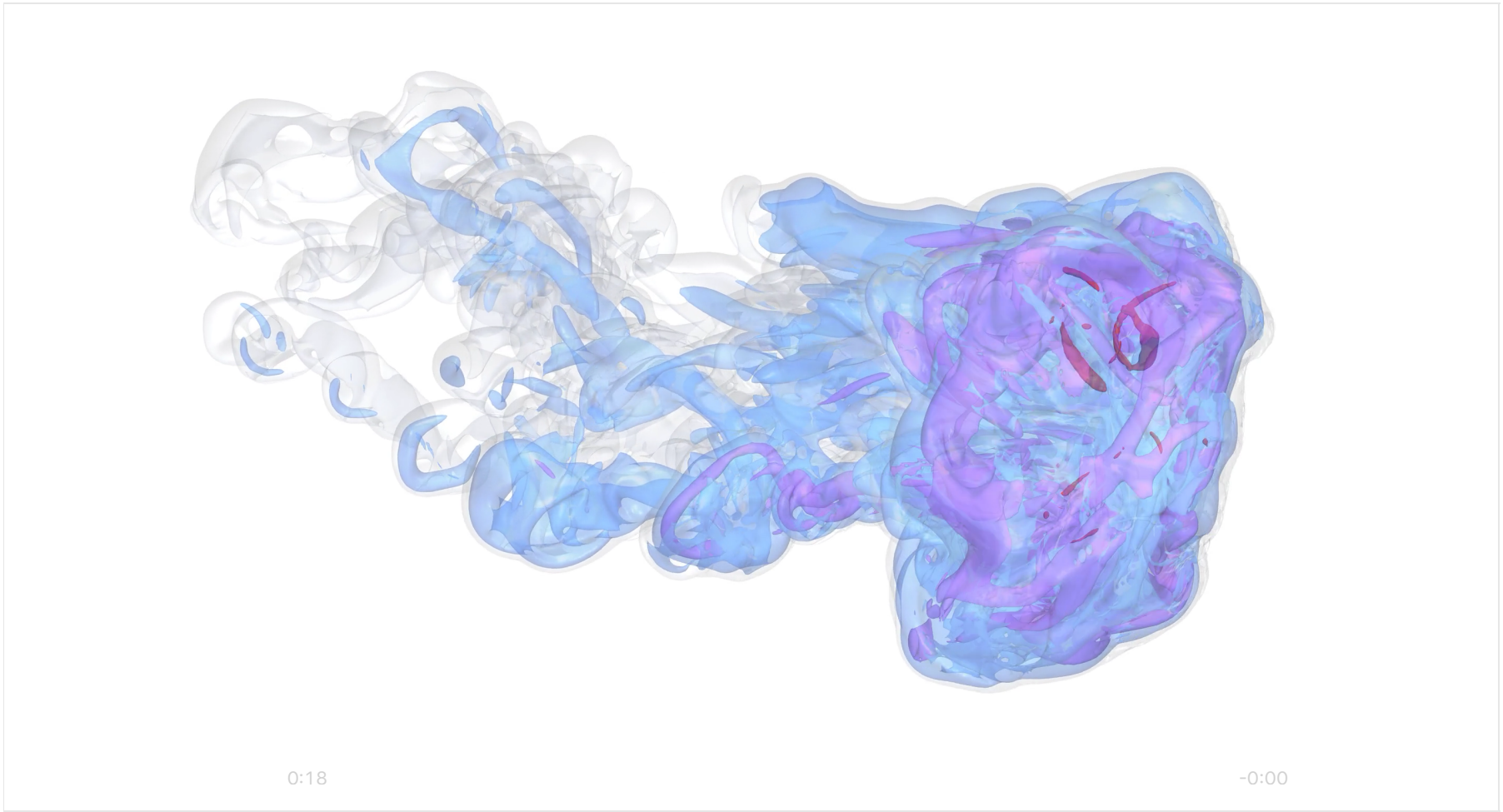
Where do linear systems come from?

For example from smoke rings

Movie by Sebastian Liska (Colonius group at Caltech)



Agenda

- Today:
  - Brief description of incompressible Navier Stokes equations.
  - Even briefer description of the discretization.
  - Poisson with Dirichlet.
  - Poisson with Neumann.
  - Description of your homework / mini-project.
  - Split into groups.
- Monday: Using FFT to solve Poisson's eq.
- Wednesday: Block cyclic reduction / demo of how to use SuperLU.
- Friday: Presentations.

Incompressible Navier Stokes

Conservation of momentum and mass followed by boundary and initial conditions:

$$
\begin{aligned}
&\partial \mathbf{u}/\partial t + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \nu\Delta\mathbf{u} + \mathbf{F}, && x \in \Omega,\ t > 0,\\
&\nabla\cdot\mathbf{u} = 0, && x \in \bar{\Omega},\ t > 0\\
&B(\mathbf{u}, p) = \mathbf{g}, && x \in \partial\Omega,\ t > 0,\\
&\mathbf{u}(\mathbf{x}, 0) = \mathbf{f}(\mathbf{x}), && x \in \Omega,\ t = 0.
\end{aligned}
$$

- Here $\mathbf{u} = (u, v)$ is the velocity, p is the kinematic pressure (scaled with the reciprocal of the constant density), ν is the kinematic viscosity and $\mathbf{F}$ is a force per unit volume (which we take to be zero).
- Bounded domain in two (or 3) dimensions.
- Assume incompressibility is satisfied for initial data.
- Great! But where is $A\mathbf{x} = \mathbf{b}$???
- And where is Poisson?

Incompressible Navier Stokes

Take the divergence of the momentum equation ("velocity-pressure" formulation).

$$
\begin{aligned}
&\partial\mathbf{u}/\partial t + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \nu\Delta\mathbf{u} + F, && x \in \Omega,\ t > 0,\\
&\Delta p + J(\nabla\mathbf{u}) - \alpha\nabla\cdot\mathbf{u} = \nabla\cdot\mathbf{F}, && x \in \Omega,\ t > 0,\\
&B(\mathbf{u}, p) = \mathbf{g}, && x \in \partial\Omega,\ t > 0,\\
&\nabla\cdot\mathbf{u} = 0, && x \in \partial\Omega,\ t > 0\\
&\mathbf{u}(\mathbf{x}, 0) = \mathbf{f}(\mathbf{x}), && x \in \Omega,\ t = 0.
\end{aligned}
$$

- Note that we added "zero" $\alpha\nabla\cdot\mathbf{u}$, which will clean out any numerically created divergence.
- In two dimensions the term $J(\nabla\mathbf{u})$ is $J(\nabla\mathbf{u}) = u_x^2 + v_y^2 + 2u_y v_x$.
- Great! Now we have 3 Poisson, one for p and two to deal with the velocities in the viscous term.

How to discretize the equations?

We use the finite difference discretization by Henshaw and Petersson:
http://www.overtureframework.org/publications/henshawPetersson2001.pdf

In the equations:

$$\partial \mathbf{u}/\partial t + (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p = \nu \Delta \mathbf{u}, \quad x \in \Omega,\ t > 0,$$
$$\Delta p + J(\nabla \mathbf{u}) - \alpha \nabla \cdot \mathbf{u} = 0, \quad x \in \Omega,\ t > 0.$$

- Replace derivatives with centered finite differences.
- Discretize BOTH velocity and pressure on the same rectilinear grid.
- What are the boundary conditions? In particular for pressure.
- Better to solve three smaller $A\mathbf{x} = \mathbf{b}$ than one big.
- What about checkerboard instability?

CFD urban myths (from a 2017 book...)

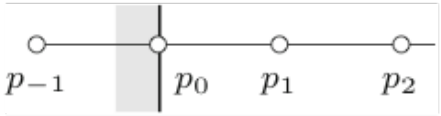


Fig. 3.7 Stencil for pressure near the boundary for a regular grid

consider a pressure boundary condition p_0 as shown in Fig. 3.7, the odd-number pressure values become independent of the boundary condition. If the pressure gradient ($\partial p/\partial x = [-p_{-1}+p_1]/2\Delta$) is specified, then the even-number pressure points become decoupled from the boundary conditions. As a result for either case of the boundary condition, the odd and even-number pressure values are not coupled and only enforce smoothness at every other point. This allows for spatial oscillations in the pressure solution to develop between adjacent values. Because of the pattern that this oscillation generates, this numerical instability is referred to as the checkerboard instability.

If we try to suppress the pressure oscillation by replacing the left-hand side of Eq. (3.84) with the left-hand side of Eq. (3.74) on a regular grid, that equation would create incompatibility between the differencing of the continuity equation and the pressure gradient. The use of a solution from such discrete pressure Poisson equation does not satisfy mass balance, which leads to error accumulation and eventual deterioration of the computation.

CFD myths

The incorrect belief that you “must” discretize on a staggered FD grid has led to a cottage industry for methods on staggered grids.

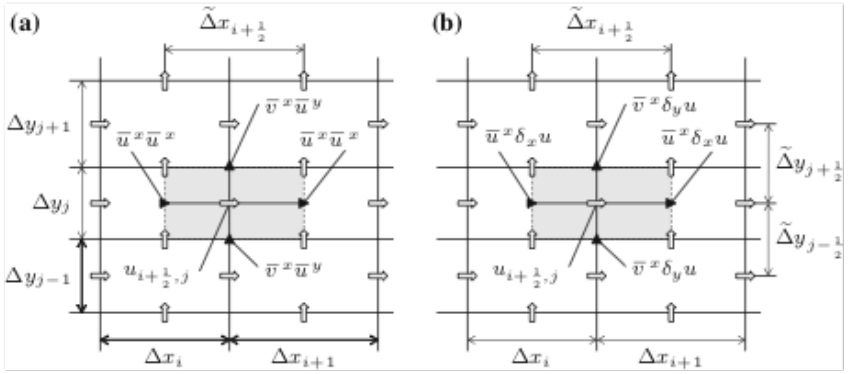
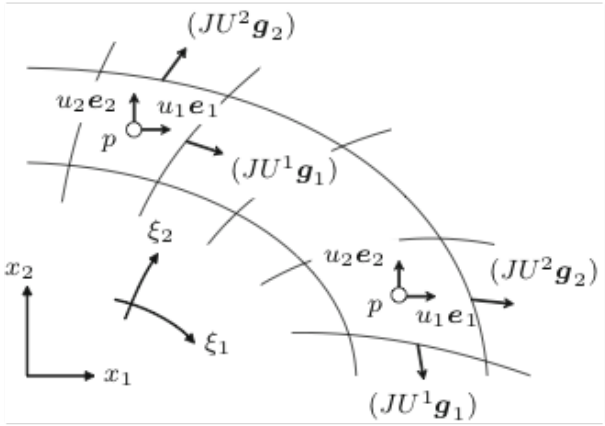


Fig. 3.10 The second-order accurate central-difference stencil for $(i+\frac{1}{2}, j)$ and its control volume for the advective term in the x-direction momentum equation for nonuniform grid. **a** Divergence form $[\partial(u^2)/\partial x + \partial(uv)/\partial y]_{i+\frac{1}{2},j}$. **b** Gradient form $[u\partial u/\partial x + v\partial u/\partial y]_{i+\frac{1}{2},j}$



There are some drawbacks with this:

- Difficult to get high order accurate solvers.
- Memory inefficient on curvilinear grids.
- Leads to saddle point systems.

Boundary conditions for the pressure

Suppose $\mathbf{u} = \mathbf{g}$ on the boundary. Then we have that the momentum equation

$$\partial \mathbf{u}/\partial t + (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p = \nu \Delta \mathbf{u}, \quad x \in \partial\Omega,\ t > 0,$$

becomes (time is tangential to the boundary so we know that $\mathbf{u}_t = \mathbf{g}_t$ there)

$$\partial \mathbf{g}/\partial t + (\mathbf{g} \cdot \nabla)\mathbf{u} + \nabla p = \nu \Delta \mathbf{u}, \quad x \in \partial\Omega,\ t > 0,$$

Extracting the normal component of the pressure we find

$$\partial p/\partial n = \mathbf{n} \cdot (-\partial \mathbf{g}/\partial t + (\mathbf{g} \cdot \nabla)\mathbf{u} + \nu \Delta \mathbf{u}), \quad x \in \partial\Omega,\ t > 0.$$

- Note two things!
- This is not a “new” boundary condition, no new information was added, we did not exploit $\nabla \cdot \mathbf{u} = 0$.
- This boundary condition is hard to discretize in a stable way. Sometimes $\partial p/\partial n = 0$ is used for no-slip b.c. but that is not consistent with the equations.

A better boundary condition

$$\partial p/\partial n = \mathbf{n} \cdot (-\partial \mathbf{g}/\partial t + (\mathbf{g} \cdot \nabla)\mathbf{u} + \nu \Delta \mathbf{u}), \quad x \in \partial\Omega,\ t > 0.$$

Now recall from vector calculus that

$$\Delta \mathbf{u} = \nabla(\nabla \cdot \mathbf{u}) - \nabla \times \nabla \times \mathbf{u}.$$

If we add the information that $\nabla \cdot \mathbf{u} = 0$ we get the so called curl-curl boundary condition

$$\partial p/\partial n = \mathbf{n} \cdot (-\partial \mathbf{g}/\partial t + (\mathbf{g} \cdot \nabla)\mathbf{u} - \nu \nabla \times \nabla \times \mathbf{u}), \quad x \in \partial\Omega,\ t > 0.$$

This boundary condition uses new information and enforces incompressibility on the boundary and is stable and does not cause a time step restriction.

This is what is implemented in the code you will use.

Time stepping method

We treat the viscous term, L_I, with the trapezoidal rule and the (nonlinear) explicit term, L_E, using Adams-Bashforth. We set the forcing to zero.

$$\frac{\mathbf{U}^{n+1} - \mathbf{U}^n}{\Delta t} = \frac{3}{2} L_E \mathbf{U}^n - \frac{1}{2} L_E \mathbf{U}^{n-1} + \frac{1}{2}(L_I \mathbf{U}^{n+1} + L_I \mathbf{U}^n)$$

Time stepping method

We treat the viscous term, L_I, with the trapezoidal rule and the (nonlinear) explicit term, L_E, using Adams-Bashforth. We set the forcing to zero and get

$$\frac{\mathbf{U}^{n+1} - \mathbf{U}^n}{\Delta t} = \frac{3}{2} L_E \mathbf{U}^n - \frac{1}{2} L_E \mathbf{U}^{n-1} + \frac{1}{2}(L_I \mathbf{U}^{n+1} + L_I \mathbf{U}^n).$$

Rearranging we find

$$(\frac{\mathbf{I}}{\Delta t} - \frac{1}{2} L_I)\mathbf{U}^{n+1} = \frac{\mathbf{U}^n}{\Delta t} + \frac{3}{2} L_E \mathbf{U}^n - \frac{1}{2} L_E \mathbf{U}^{n-1} + \frac{1}{2} L_I \mathbf{U}^n.$$

Poisson with Dirichlet boundary conditions

We have:

$$(\frac{\mathbf{I}}{\Delta t} - \frac{1}{2}L_I)\mathbf{U}^{n+1} = \frac{\mathbf{U}^n}{\Delta t} + \frac{3}{2}L_E\mathbf{U}^n - \frac{1}{2}L_E\mathbf{U}^{n-1} + \frac{1}{2}L_I\mathbf{U}^n.$$

Denote by $\mathcal{L}_D$ the matrix corresponding to a centered finite difference discretization of $u_{xx} + u_{yy}$ with Dirichlet boundary conditions and using uniform grid spacing h_x and h_y and with ordering of the degrees of freedom being fast in x and arranged with u and v being the slowest variable. Then we will have to solve the systems

$$(\frac{\mathbf{I}}{\Delta t} - \frac{1}{2}\mathcal{L}_D)\mathbf{W}^{n+1} = \mathbf{h}.$$

Where $\mathbf{W}$ is the grid function for either u or v and $\mathbf{h}$ incorporates the right hand side from above and boundary conditions.

This is an easy problem! It is S.P.D. so you can use many different methods to solve it.

Poisson with Neumann boundary conditions

Once $\mathbf{U}^{n+1}$ is known we solve for the pressure. The equation for the pressure takes the form

$$\mathcal{L}_N \mathbf{P}^{n+1} = \mathbf{h},$$

where $\mathcal{L}_N$ is like $\mathcal{L}_D$ but with Neumann boundary conditions. Again $\mathbf{h}$ incorporates the right hand side and boundary conditions.

Why is this problem more difficult to solve?

- ANY constant satisfies $p_{xx} = 0$ and $p_x = 0$ on the boundary.
- What is a null vector to $\mathcal{L}_N$?
- A column vector with all ones, call it $\mathbf{r}$.

Poisson with Neumann boundary conditions

We can make the problem unique by solving (the extra condition is that the mean pressure is zero)

$$\begin{pmatrix} \mathcal{L}_N & \mathbf{r} \\ \mathbf{r}^T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{P}^{n+1} \\ \beta \end{pmatrix} = \begin{pmatrix} \mathbf{h} \\ 0 \end{pmatrix}.$$

The extended matrix above is non-singular and can be factored with dense linear algebra. This is what is currently done in the code:

```
CALL DGETRF(sys_size_pbig,sys_size_pbig,LapPbig,sys_size_pbig,ipiv_pbig,INFO)

! !!! YOUR CODE REPLACES THIS !!!!
! Solve for p
CALL DGETRS('N',sys_size_pbig,1,LapPbig,sys_size_pbig,IPIV_pbig,&
 pbvecbig,sys_size_pbig,INFO)
! !!! END YOUR CODE REPLACES THIS !!!!
```

Homework / mini-project

- Clone the git repository at https://github.com/appelo/ins4nla
- Run the code and plot the solution in matlab. The current setup is for a lid-driven cavity flow with Reynolds number $Re = 1/\nu = 100$.
- Experiment to see how many gridpoints you need in order to resolve the above flow, you will have to adjust the time-step as well to make sure that the explicit term does not cause a time instability.
- From classical theory (see http://rdcu.be/l9gt) we know that the smalles scale in an unforced incompressible flow in two dimensions scales on the square root of the viscosity divided by the maximum norm of the vorticity at the initial time.
- Take the grid spacing to be proportional to the smallest scale and estimate how many gridpoints you will need in each direction to run a computation at $Re = 8000$, which roughly is when the lid-driven cavity problem transitions. Can you solve this problem with dense linear algebra?
- Select a method for the Dirichlet - Poisson problem and implement it instead of the dense LAPACK routines.

Homework / mini-project

- Select a better/faster method for the Dirichlet - Poisson problem and implement it instead of the dense LAPACK routines. Possible options are:
  - Stationary methods (Jacobi etc.)
  - PCG, GMRES, etc.
- Note that the routine apply_velocity_laplacian(lu,u,nx,ny,hx,hy) returns the action of the Laplacian (the viscosity is NOT included) but remember that you need to invert $(\frac{\mathbf{I}}{\Delta t} - \frac{1}{2}\mathcal{L}_D)$.
- If your method can be preconditioned try to find a good preconditioner.
- For the pressure solve the small (original) problem is semi-definite while the extended system is indefinite so your options are more limited. If you solve the extended system what are your options?
- It is in-fact possible to use CG on a semi-definite matrix. In exact arithmetic it is sufficient as to make sure that the right hand side and the initial guess is in the range of the matrix. In finite precision, unless there is only a few iterations needed, you may benefit from projecting the solution and the residual onto the range of the matrix every now and then (note that this is easy and cheap as you know the null vector).

Homework / mini-project competitions

- What team can get the largest speedup compared to the original version of the code? WONDERFUL PRIZES WILL BE AWARDED!!!
- What team can make the best fluid flow movie? WONDERFUL PRIZES WILL BE AWARDED!!!
- Questions?
- Time to split into teams.