Fluid Simulations for Undergraduates

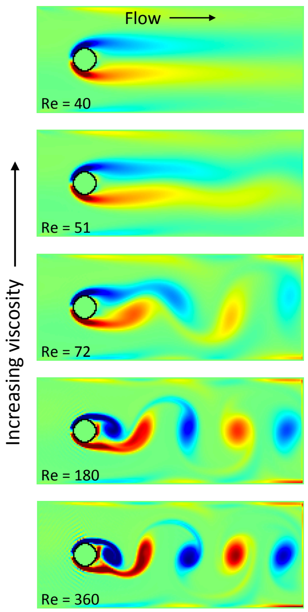
Dan Schroeder, Weber State University, physics.weber.edu/schroeder/fluids

Interactive exploration

and numerical experiments

Vortex shedding by a circular barrier

(color indicates curl of the velocity)



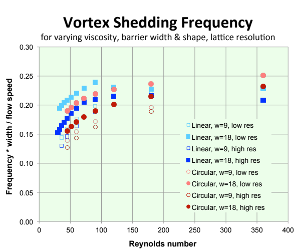
Laminar flow

Becoming unstable

Vortex shedding

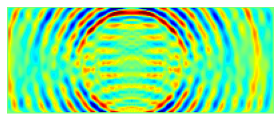
Stronger vortices

Stronger still



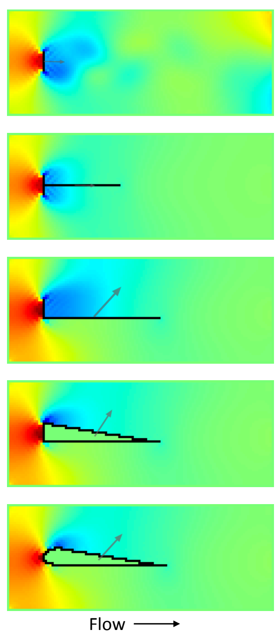
Sound waves

(color indicates density)



Force on a barrier

(color indicates density)



Linear barrier sheds vortices, so force oscillates.

Spoiler stops vortex shedding.

Right angle produces low-pressure zone, hence upward force.

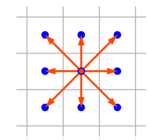
Wedge isn't too different from right angle.

Airfoil isn't too different from wedge.

How it works

(Lattice-Boltzmann algorithm)

- Discretize two-dimensional space with a square lattice.
- Allow only 9 fundamental displacements and velocities.
- Simulation variables n_i are the 9 *densities*, at each lattice site, of molecules with the 9 allowed velocities.



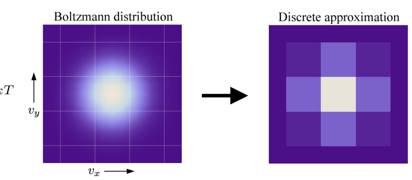
$\vec{e}_0 = 0$
$\vec{e}_1 = (1,0)$ $\quad \vec{e}_5 = (1,1)$
$\vec{e}_2 = (0,1)$ $\quad \vec{e}_6 = (-1,1)$
$\vec{e}_3 = (-1,0)$ $\quad \vec{e}_7 = (-1,-1)$
$\vec{e}_4 = (0,-1)$ $\quad \vec{e}_8 = (1,-1)$

- From these we can compute total density ρ and macroscopic flow velocity $\vec{u}$:

$$\rho = \sum n_i \qquad u_x = \frac{(n_1+n_5+n_8)-(n_3+n_6+n_7)}{\rho} \qquad u_y = \frac{(n_2+n_5+n_6)-(n_4+n_7+n_8)}{\rho}$$

- To model *thermal* velocities ($\vec{v}$), discretize the Boltzmann distribution. Weights are determined by equating moments, up to 4th order, of the continuous and discrete distributions.

$$D(\vec{v}) = \frac{m}{2\pi kT} e^{-m|\vec{v}|^2/2kT}$$



$$w_0 = \frac{4}{9} \qquad w_1 = w_2 = w_3 = w_4 = \frac{1}{9} \qquad w_5 = w_6 = w_7 = w_8 = \frac{1}{36}$$

- Total (discretized) velocity is flow velocity plus thermal velocity: $\vec{e}_i = \vec{u} + \vec{v}$ $\quad (|\vec{u}| \ll 1)$
Plug into Boltzmann distribution and expand to second order in $\vec{u}$ to obtain equilibrium densities:

$$D(\vec{v}) \longrightarrow \frac{m}{2\pi kT} \exp\Big(-\frac{m}{2kT}|\vec{e}_i - \vec{u}|^2\Big) \quad \ldots \quad n_i^{\text{eq}} = \rho\, w_i \Big[1 + 3\,\vec{e}_i\cdot\vec{u} + \tfrac{9}{2}(\vec{e}_i\cdot\vec{u})^2 - \tfrac{3}{2}|\vec{u}|^2\Big]$$

- During each time step, molecules within each lattice cell collide and relax toward these equilibrium values, by an amount that depends on the relaxation time τ (which increases with increasing viscosity):

$$n_i \longrightarrow n_i + \frac{1}{\tau}(n_i^{\text{eq}} - n_i)$$

- The algorithm is simply to alternate these collisions with "streaming" in which the molecules move into adjacent cells according to their velocities. (When molecules hit a barrier, they bounce back instead.)

- The pros code this in Fortran or C, but for 10^4 to 10^5 lattice sites, on today's personal computers, you can get by with an interpreted language. My Python/NumPy code is only 125 lines; Java or JavaScript requires about twice that, not including GUI controls.

- See the web site for more details on the theory, code examples, and references. Enjoy!