Applied Mathematics 225

Unit 5: Special topics in scientific computing

Lecturer: Chris H. Rycroft

Derivation of the Navier–Stokes equations

Consider a fluid in d dimensions with spatially varying density $\rho(\mathbf{x}, t)$ and velocity $\mathbf{u}(\mathbf{x}, t)$. From conservation of mass we obtain

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0. \tag{1}$$

Let Ω be a small control volume of fluid. Integrating Eq. 1 and applying the divergence theorem yields

$$\frac{\partial}{\partial t} \int_{\Omega} \rho d\mathbf{x} = - \int_{\partial \Omega} \rho \mathbf{n} \cdot \mathbf{u} \, dS$$

where $\mathbf{n}$ is an outward-pointing normal vector. Hence the change in mass in Ω is equal to the mass flux across the boundary $\partial \Omega$.

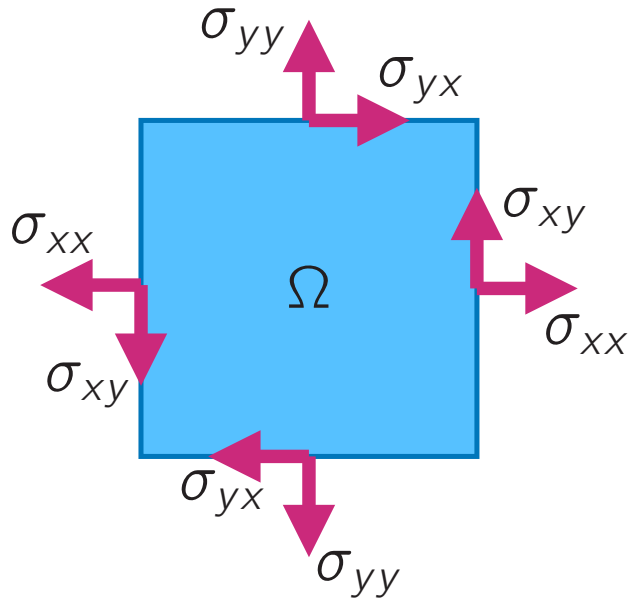
Derivation of the Navier–Stokes equations

To derive an equation for the velocity, we first introduce the Cauchy stress tensor, which represents the forces created by small control volume.

In two dimensions

$$\boldsymbol{\sigma} = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{yx} & \sigma_{yy} \end{pmatrix}$$

where the components are shown in the diagram. Since the control volume cannot generate a torque, the stress tensor is symmetric, so that $\sigma_{xy} = \sigma_{yx}$.



Derivation of the Navier–Stokes equations

By considering Newton's second law applied to the small control volume we obtain

$$\rho \frac{d\mathbf{u}}{dt} = \nabla \cdot \boldsymbol{\sigma} + \mathbf{f}.$$

Here, $d/dt = \partial/\partial t + (\mathbf{u} \cdot \nabla)$ is the convective derivative.

The term $\nabla \cdot \boldsymbol{\sigma}$ represents how local stress imbalances generate a force on the control volume.

$\mathbf{f}$ represents any external applied forces (*e.g.* gravity).

Fluid stress tensor

For a Newtonian fluid the stress tensor is given by

$$\boldsymbol{\sigma} = -p\mathrm{I} + \mu(\nabla\mathbf{u} + (\nabla\mathbf{u})^{\mathsf{T}})$$

where p is the pressure and μ is a constant called the dynamic viscosity. The second term is the viscous stress, and represents the fluid resistance to applied deformation.

In the Newtonian fluid model, viscous stresses are linearly proportional to the deformation rate. Examples where this is a good model include air, water, and glycerol.

Pressure

To obtain a closed set of equations, we need an equation for the pressure. This follows similar considerations to the 1D gas model considered in the finite volume section. In the most general case, this could involve density and temperature of the fluid.

A simpler approach, applicable to liquids such as water, is to model the fluid as weakly compressible, so that

$$\rho = \rho_0(1 - \alpha(p - p_0))$$

for some constants ρ_0, p_0, and α, where α is small.

Compressive waves move with speed $c = \sqrt{\rho_0/\alpha}$.

Difficulties with compressibility

Typical fluids of interest (*e.g.* water) have very low compressibility.

Pressure waves travel at very fast speeds c on the order of km/s. The CFL condition requires that $\Delta t \leq \Delta x/c$, which is a very strong restriction. This makes it difficult to simulate fluids on many timescales of practical interest (`e.g.` minutes, hours).

One approach is to make the fluid artifically soft by increasing α, thereby decreasing c.

An alternative approach is to model the fluid as incompressible, so that $\nabla \cdot \mathbf{u} = 0$.

Incompressible Navier–Stokes equations

In that case we must solve the incompressible Navier–Stokes (NS) equations

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{\nabla p}{\rho} + \nu \nabla^2 \mathbf{u}$$

subject to the divergence-free criterion $\nabla \cdot \mathbf{u} = 0$. Here $\nu = \mu/\rho$ is the kinematic viscosity.

We no longer have an explicit update equation for ρ (or equivalently p). Instead, it is determined implicitly in order to maintain the divergence-free criterion. It is not immediately obvious how to efficiently solve this system (one equation plus one constraint) numerically.

Chorin's projection method

In 1968 Alexandre Chorin introduced the projection method[1] for the incompressible Navier–Stokes equations.

This work was the successor to a previous paper[2] that used a simpler, more direct approach based on artificial compressibility.

Since its introduction, Chorin's projection method has been extensively studied and developed, and now forms the basis of a wide variety of computational fluid dynamics software.

---

[1] A. J. Chorin, *Numerical Solution of the Navier–Stokes Equations*, Mathematics of Computation **22**, 745–762 (1968).

[2] A. J. Chorin, *A Numerical Method for Solving Incompressible Viscous Flow Problems*, Mathematics of Computation **2**, 12–26 (1967).

Chorin's projection method

Let $\mathbf{u}^n$ be the discretized velocity field at timestep n, and consider taking a timestep of size Δt to $\mathbf{u}^{n+1}$. To begin, compute the intermediate velocity $\mathbf{u}^*$ by considering convection and viscosity:

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = -(\mathbf{u}^n \cdot \nabla)\mathbf{u}^n + \nu \nabla^2 \mathbf{u}^n. \tag{2}$$

Then, to obtain $\mathbf{u}^{n+1}$, we would need to compute

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} = -\frac{\nabla p^{n+1}}{\rho} \tag{3}$$

but we do not know p^{n+1}.

(We have made a choice here to treat p as being at step $n+1$. This is consistent with the method being first-order accurate in time, like a backward Euler step.)

Chorin's projection method

Taking the divergence of Eq. 3 yields

$$\nabla \cdot \mathbf{u}^{n+1} - \nabla \cdot \mathbf{u}^{*} = -\frac{\Delta t}{\rho} \nabla^2 p^{n+1}. \tag{4}$$

We require that the velocity field at the end of the timestep is divergence-free, and hence $\nabla \cdot \mathbf{u}^{n+1} = 0$, so

$$\nabla \cdot \mathbf{u}^{*} = \frac{\Delta t}{\rho} \nabla^2 p^{n+1}, \tag{5}$$

which is an elliptic equation for the pressure p^{n+1}. Once p^{n+1} is known, then Eq. 3 can be used to find $\mathbf{u}^{n+1}$.

Chorin's projection method

The complete method is therefore

1. Compute the intermediate velocity

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = -(\mathbf{u}^n \cdot \nabla)\mathbf{u}^n + \nu \nabla^2 \mathbf{u}^n.$$

2. Solve the elliptic problem for the pressure

$$\nabla \cdot \mathbf{u}^* = \frac{\Delta t}{\rho} \nabla^2 p^{n+1}.$$

3. Use the pressure to correct the velocity to be divergence-free

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} = -\frac{\nabla p^{n+1}}{\rho}.$$

Information transfer for the incompressible NS equations

Note that it is natural that the solution method for the incompressible Navier–Stokes equations should involve an elliptic problem. Elliptic problems usually have non-local solutions, so that a localized source term leads to a non-local solution.

For example in $\mathbb{R}^3$, the Poisson equation for a function $u(\mathbf{x}, t)$ with a localized delta function source term is

$$\nabla^2 u = -4\pi\delta(\mathbf{x}).$$

This has solution

$$u(\mathbf{x}) = \frac{1}{|\mathbf{x}|},$$

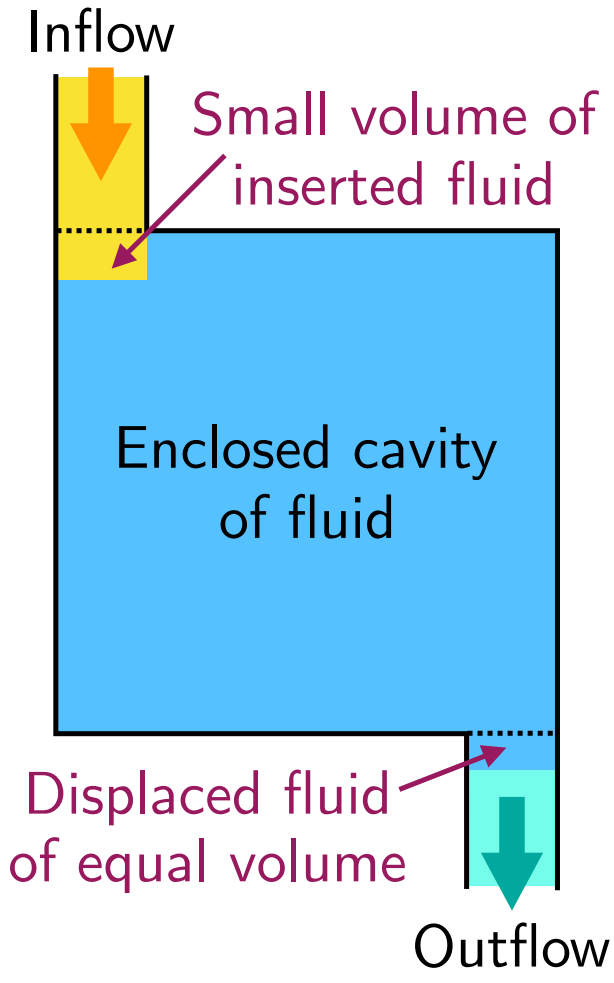
which is non-zero everywhere.

Information transfer for the incompressible NS equations

Consider the hypothetical case shown, of an incompressible fluid in a cavity, with inflow and outflow pipes. If a small volume of fluid is inserted at the inflow, then the same volume must be instantaneously removed at the outflow.

In the simulation, the non-locality of the elliptic problem allows this information to propagate instantaneously across the entire cavity.

This is different from the compressive case, where the insertion of fluid would cause a pressure wave to propagate across the cavity at a finite speed.



Pressure projection: boundary conditions

We must apply boundary conditions on the elliptic problem. If we have a pressure boundary condition, then it is straightforward and we obtain a Dirichlet condition for p.

For a velocity boundary condition such as $\mathbf{u} = \mathbf{0}$, we can take the scalar product of Eq. 3 with an outward-pointing normal vector $\mathbf{n}$ to obtain

$$\mathbf{n} \cdot \mathbf{u}^* = \frac{\Delta t}{\rho} \mathbf{n} \cdot \nabla p,$$

which is a Neumann condition for p.[3]

[3] Note that there are subtleties here when trying to achieve higher accuracy. This will be briefly discussed later.

Pressure projection: approaches

There is freedom in how to implement the pressure projection. Two families of methods are

1. "Projection" methods: these construct the elliptic problem so that the velocity field $\mathbf{u}^{n+1}$ satisfies a discrete divergence constraint exactly.
2. "Pressure-Poisson" methods: these solve a Poisson problem for the pressure so that a discrete divergence constraint is satisfied only up to the truncation error of the method.

While option 1 may seem more desirable, it has been shown to lead to weak instabilities in some cases. Option 2 also provides more flexibility in implementation.

An example of option 2 is the finite-element based "approximate" projection method of Almgren *et al.*[4]

[4] A. S. Almgren, J. B. Bell, and W. G. Szymczak, *A numerical method for the incompressible Navier–Stokes equations based on an approximate projection*, SIAM J. Sci. Comput. **17**, (1996).

Numerical example

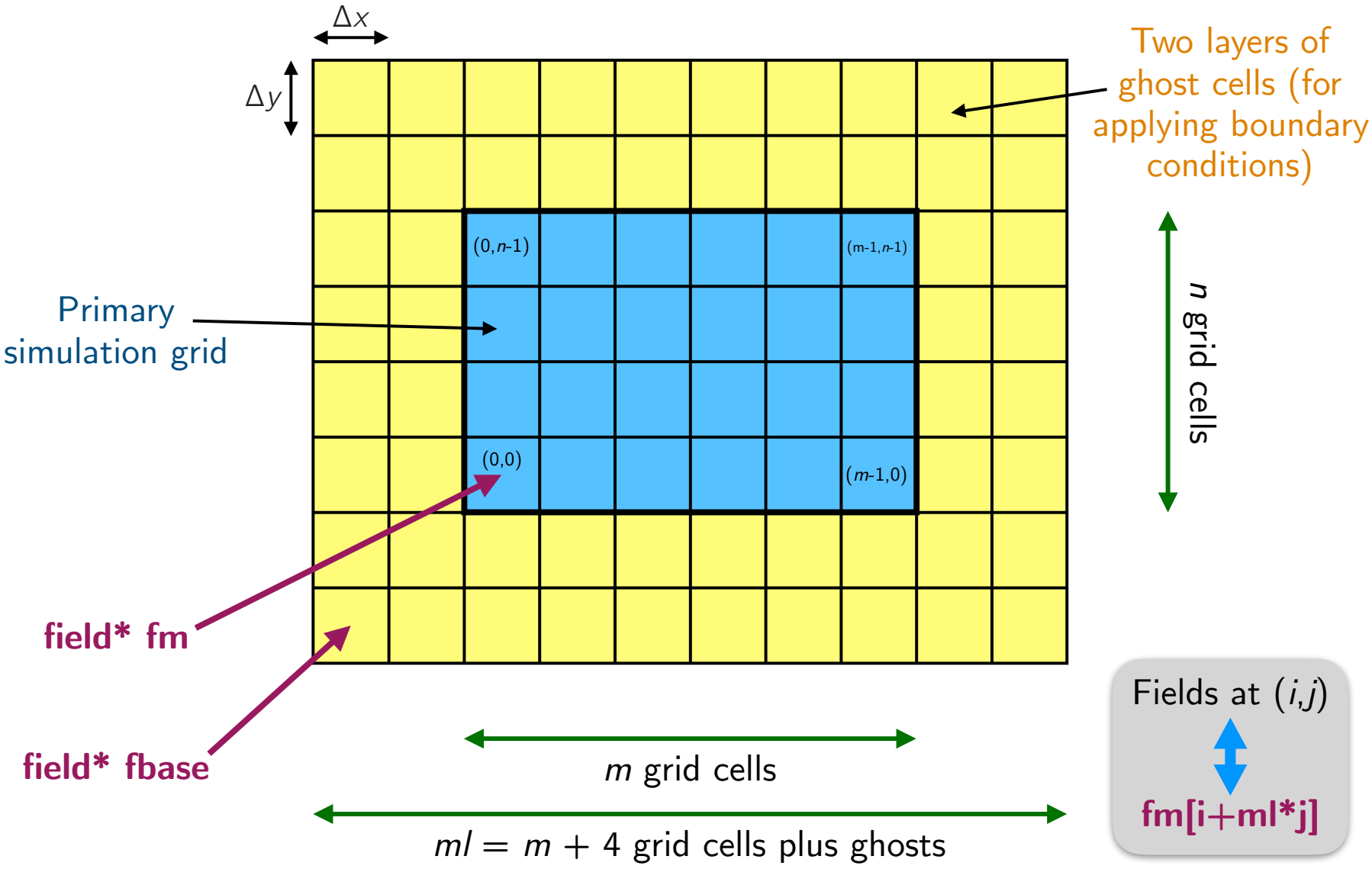
The directory `5a_fluid_sim` in the `am225_examples` repository contains a two-dimensional C++ implementation of Chorin's projection method.

The simulation is primarily in the `fluid_2d` class. The simulation runs on a rectangular grid and can use both periodic and non-periodic boundary conditions.

The `fluid_2d` class uses a 2D array of `field` data structures. The `field` data structure contains all of the required simulation fields: velocity $\mathbf{u} = (u, v)$, intermediate velocity $\mathbf{u}^* = (u^*, v^*)$, and pressure p.

The program `fluid_test.cc` is a front-end that runs a simple test case where several fluid vortices move around and interact. It saves snapshots of the fields at regular intervals.

Overall grid structure

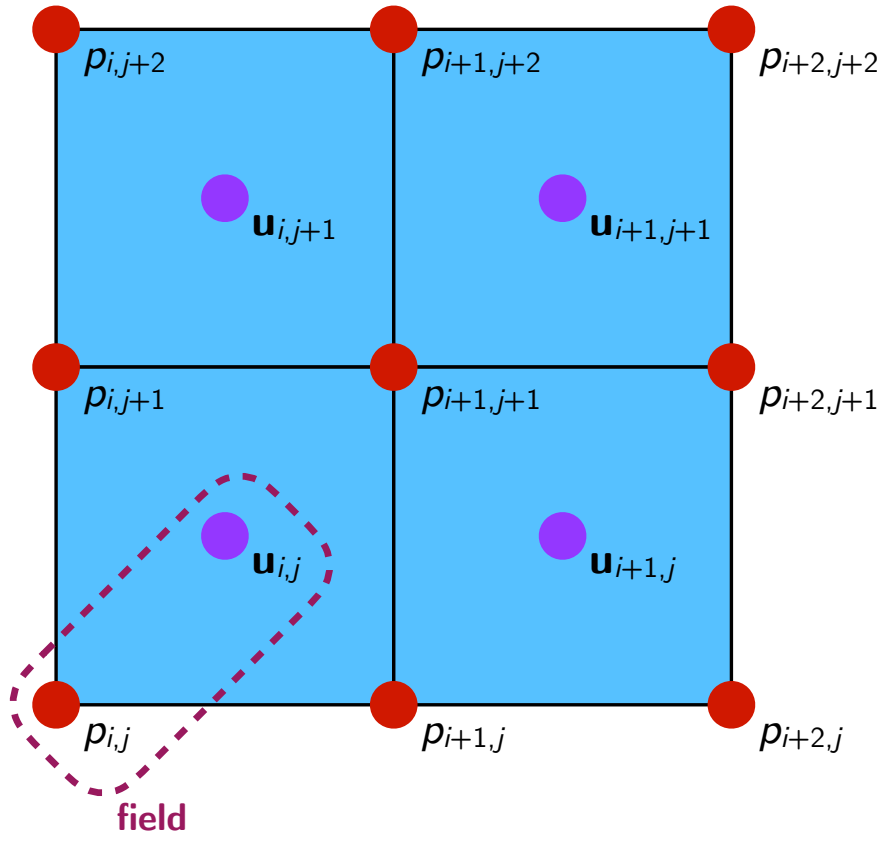


Field discretization

The velocity field is discretized at grid cell centers and the pressure field is discretized at grid cell corners. This is a typical approach and has some stability advantages over co-locating all fields in one position.

The `field` structure includes the (i, j) entries of $\mathbf{u}$, p, and $\mathbf{u}^*$.

As shown on the previous slide, the main grid is padded with two layers of ghost cells. These are continually updated to be consistent with the required boundary conditions.



Advection and viscous terms

The advection term $(\mathbf{u}^n \cdot \nabla)\mathbf{u}^n$ is calculated using the ENO method introduced in AM225 Unit 4.

The Laplacian term for the viscous stress is calculated using the centered difference formula

$$[\nabla^2 \mathbf{u}]^n_{i,j} = \frac{\mathbf{u}^n_{i+1,j} - 2\mathbf{u}^n_{i,j} + \mathbf{u}^n_{i-1,j}}{\Delta x^2} + \frac{\mathbf{u}^n_{i,j+1} - 2\mathbf{u}^n_{i,j} + \mathbf{u}^n_{i,j-1}}{\Delta y^2}.$$

Pressure projection

The projection step uses the finite-element based approximate projection technique of Almgren *et al.* The pressure is treated as piecewise bilinear on grid cells, so that $p(\mathbf{x}) = \sum_l p_l \psi_l(\mathbf{x})$ for basis functions $\psi_l(\mathbf{x})$.[5] The field $\mathbf{u}^*$ is treated as piecewise constant on grid cells.

In weak form the projection is

$$\int_\Omega \psi_k \nabla^2 p d\mathbf{x} = \int_\Omega \psi_k \nabla \cdot \mathbf{u}^* d\mathbf{x}.$$

On a periodic domain, integrating by parts yields

$$-\sum_l p_l \int_\Omega \nabla \psi_k \cdot \nabla \psi_l d\mathbf{x} = -\int_\Omega \mathbf{u}^* \cdot \nabla \psi_k \, d\mathbf{x},$$

which leads to a sparse linear system to solve for the p_l.

[5] These are the same basis functions as in question 2 on homework 3.

Multigrid library

The linear system is solved using the multigrid method.[6] This technique is highly efficient for Poisson problems. Algorithmically, multigrid is even more efficient than conjugate gradient or FFT, requiring $O(1)$ work per gridpoint.[7]

The code uses the TGMG (Templated Geometric MultiGrid) library developed by the Rycroft group. A copy of the library is provided in the `am225_examples/tgmg` directory.

The class `mgs_fem` describes the multigrid linear system to be solved in the `fluid_2d` class, and is used by TGMG.

---

[6] See the notes from AM205 Unit 5.

[7] See the method comparison in chapter 6 of J. Demmel's *Applied Numerical Linear Algebra* book.

Pressure correction

Once the pressure is computed, the velocity $\mathbf{u}^{n+1}$ is computed using a centered finite-difference stencil. For the x velocity component

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

and for the y velocity component

$$v_{i,j}^{n+1} = v_{i,j}^{*} - \frac{\Delta t}{2\rho\Delta x}(p_{i+1,j+1}^{n+1} - p_{i+1,j}^{n+1} + p_{i,j+1}^{n+1} - p_{i,j}^{n+1}).$$

Timestep choice

The CFL condition for the advection term enforces a maximum timestep

$$\Delta t_{\text{adv}} = \min\left(\frac{\Delta x}{u_{\max}}, \frac{\Delta y}{v_{\max}}\right)$$

where $u_{\max}$ and $v_{\max}$ are the maximum horizontal and vertical fluid velocity components taken over the entire grid, respectively. The viscous term requires that the timestep be smaller than

$$\Delta t_{\text{vis}} = \frac{1}{2\nu(\Delta x^{-2} + \Delta y^{-2})}.$$

In the code, the timestep is chosen as

$$\Delta t = s \min(\Delta_{\text{adv}}, \Delta_{\text{vis}})$$

where $s \leq 1$ is a padding factor. The code makes a further small adjustment to ensure that an integer multiple of timesteps precisely matches the snapshot output interval.

Tracers

To visualize the fluid flow, the code simulates a number of passive tracer particles with positions $\mathbf{X}_k(t)$. The tracers are initially located at random positions throughout the simulation domain.

The tracers follow the ODE

$$\frac{d\mathbf{X}_k}{dt} = \mathbf{u}_{\text{bilin}}(\mathbf{X}_k(t), t)$$

where the time derivative is simulated using the explicit Euler method, and $\mathbf{u}_{\text{bilin}}$ is the bilinear interpolation of the velocity field.

Simulation output

The code creates a directory `ftest.out` for storing the simulation output. The fields are stored in the Gnuplot binary matrix format, which has been used in previous examples.

A Perl script `gnuplot_movie.pl` is provided that can process all of the output files into PNG images:

```
./gnuplot_movie.pl -t ftest.out p
```

The Perl script supports several other options, which can be seen by typing:

```
./gnuplot-movie.pl -h
```

Movie

Once the PNG images are created, the script will try to make a movie of the output.

- On the Mac and Linux it tries to use FFmpeg to make a QuickTime-compatible movie using the modern HEVC codec, which provides excellent quality and compression.
- On Windows, the script does not automatically make a movie. However, the program VirtualDub is one option for making movies. FFmpeg can also be installed.

On the Mac it's also possible to use qt_tools, a set of command-line utilities to access the built-in QuickTime libraries in macOS. They work well but the tools/libraries have not been updated in a long time and there is no HEVC support.

Improvements

The paper by Brown *et al.*[8] describes many improvements and extensions to the method.

A general framework for achieving second-order accuracy involves solving

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} + \nabla p^{n+1/2} = -[(\mathbf{u} \cdot \nabla)\mathbf{u}]^{n+1/2} + \frac{\nu}{2}\nabla^2(\mathbf{u}^{n+1} + \mathbf{u}^n)$$

subject to $\nabla \cdot \mathbf{u}^{n+1} = 0$ with boundary conditions $\mathbf{u}^{n+1}|_{\partial\Omega} = \mathbf{u}_b^{n+1}$.

Here $[(\mathbf{u} \cdot \nabla)\mathbf{u}]^{n+1/2}$ represents a second-order approximation to the convective derivative at the half-timestep $t^{n+1/2}$.

---

[8] D. L. Brown, R. Cortez, and M. L. Minion, *Accurate Projection Methods for the Incompressible Navier–Stokes Equations*, J. Comput. Phys. **168**, 464–499 (2000).

General outline for a second-order method

1. Solve for the intermediate field $\mathbf{u}^*$

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} + \nabla q = -[(\mathbf{u} \cdot \nabla)\mathbf{u}]^{n+1/2} + \frac{\nu}{2}\nabla^2(\mathbf{u} + \mathbf{u}^*).$$

   with BCs $B(\mathbf{u}^*) = 0$, where q is a pressure approximation to $p^{n+1/2}$.

2. Perform the projection

$$\mathbf{u}^* = \mathbf{u}^{n+1} + \Delta t \nabla \phi^{n+1}$$

   with BCs $B(\mathbf{u}^*) = 0$ and $\mathbf{u}^{n+1}|_{\partial\Omega} = \mathbf{u}_b^{n+1}$.

3. Update the pressure

$$p^{n+1/2} = q + L(\phi^{n+1}).$$

Helmholtz–Hodge decomposition

Any vector field[9] $\mathbf{u}$ can be decomposed into a divergence-free part $\mathbf{u}_{\text{sol}}$ and an irrotational (curl-free) part $\mathbf{u}_{\text{irrot}}$, so that

$$\mathbf{u} = \mathbf{u}_{\text{sol}} + \mathbf{u}_{\text{irrot}} = \mathbf{u}_{\text{sol}} + \nabla\phi.$$

This is one of the mathematical underpinnings of the projection method, and is also used in an alternative approach called the gauge method.

[9] In a simply connected domain.