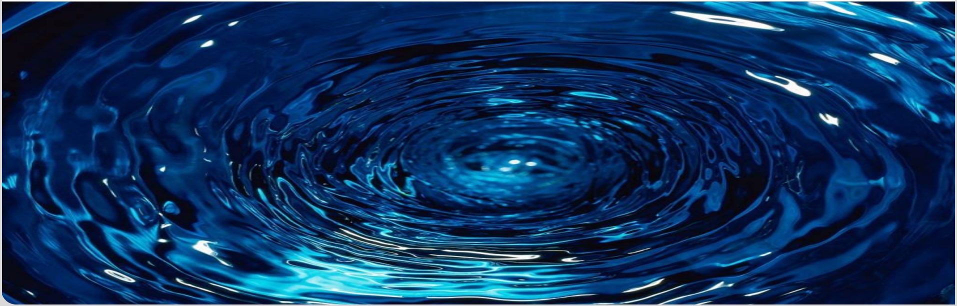
Karlsruhe Institute of Technology

Fractional step method for the incompressible Navier-Stokes equations

as part of the course Computational Fluid Dynamics II

Michael Stumpf | June 24, 2014

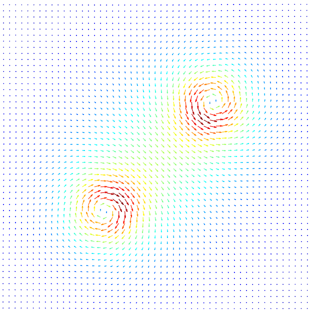
INSTITUTE FOR HYDROMECHANICS



Outline

Aim of this project

- simulation of the interaction between two counterrotating vortices
- tracking of vortex centers
- influence of Reynolds number and distance between centers



quiver plot of velocity

Subtasks

Numerical elaboration

- spatial and temporal discretization
- stability analysis

Implementation

- implementation of discretized equations in C++
- verification of the code with help of analytic solution (Taylor-Green)

Simulation & Post-Processing

- simulation of the vortices and parameter studies
- graphical post-processing with Matlab

Numerical elaboration

starting point: dimensionless, incompressible Navier-Stokes equations

$$\nabla \cdot \mathbf{u} = 0$$
$$\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p = \frac{1}{\mathrm{Re}} \nabla^2 \mathbf{u}$$

Problems

- incompressible $\rightarrow$ no equation for pressure
- non-linear convective term

Non-linear term

- hard to discretize (most higher-order schemes need unknown velocity field at future time step)
- second order discretization can be archieved with **Adams-Bashforth** temporal discretization

$$\partial_t \mathbf{H} = \frac{1}{2}\left(3\mathbf{H}^n - \mathbf{H}^{n-1}\right) + \mathcal{O}(\Delta t^2) \qquad \text{where} \quad \mathbf{H} = -\mathbf{u} \cdot \nabla \mathbf{u}$$

$\Rightarrow$ needs velocity fields from current and previous time step, which are known

Incompressibility

role of pressure: force velocity field to be divergence-free

Fractional step method (Kim and Moin, 1985)

- calculate a velocity field which is not divergence-free

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = \underbrace{\frac{1}{2}\left(3\mathbf{H}^n - \mathbf{H}^{n-1}\right)}_{\text{Adams-Bashforth } \mathcal{O}(\Delta t^2)} + \frac{1}{\mathrm{Re}} \underbrace{\frac{1}{2}\nabla^2\left(\mathbf{u}^* + \mathbf{u}^n\right)}_{\text{Crank-Nicholson } \mathcal{O}(\Delta t^2)}$$

- correct the preliminary field with help of a correctional term ϕ (derived from the continuity equation)

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} = -\nabla\phi, \qquad \nabla^2\phi = \frac{1}{\Delta t}\nabla \cdot \mathbf{u}^*$$

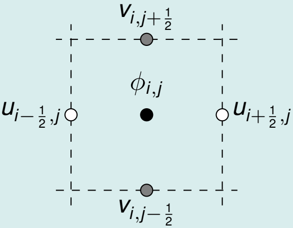
Discretization

Temporal discretization

- convective term: Adams-Bashforth $\mathcal{O}(\Delta t^2)$
- diffusive term: Crank-Nicholson $\mathcal{O}(\Delta t^2)$

Spatial discretization

- central difference on staggered grid $\mathcal{O}(\Delta x^2)$



Stability analysis

- von Neumann analysis for prototype equation (1D advection-diffusion)

$$u_{,t} = -cu_{,x} + \nu u_{,xx}, \qquad \nu \geq 0$$

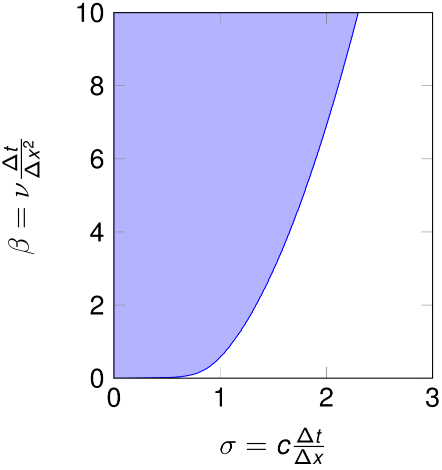
- analysis results in quadratic equation

$$G - 1 = -\frac{\sigma}{4}\left(3 - \frac{1}{G}\right)\left(e^{I\theta} - e^{-I\theta}\right) + \frac{\beta}{2}(G+1)\left(e^{I\theta} + e^{-I\theta} - 2\right)$$

with the stability parameters

$$\sigma = c\frac{\Delta t}{\Delta x} \qquad \beta = \nu\frac{\Delta t}{\Delta x^2}$$

Stability analysis



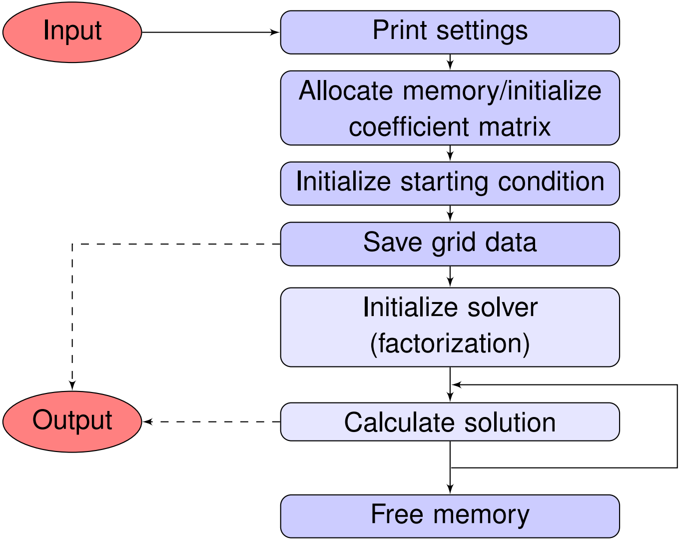
Implementation

- square uniform grid on 2D domain
- boundary conditions are periodic
- calculations are done in C++ (efficient, fast code)
- post-processing is done in Matlab (convenient plotting utilities)

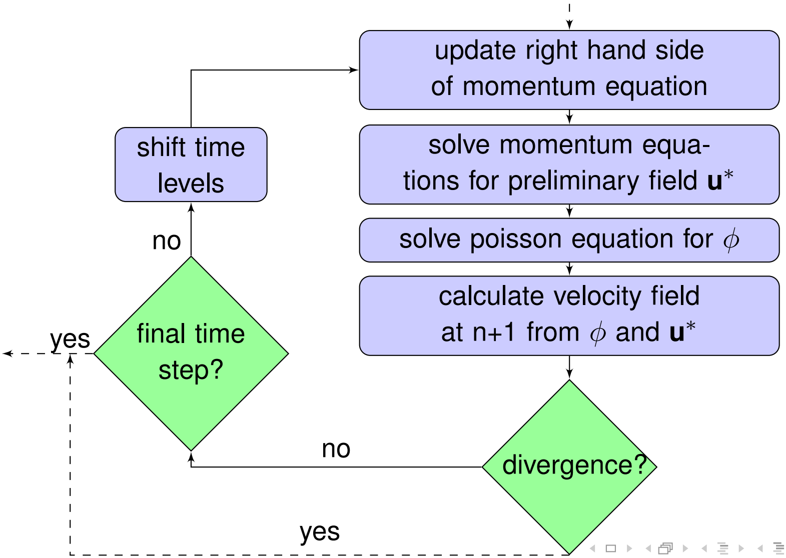
Eigen3 library

- library for linear algebra, matrix and vector operations
- includes various algorithms for solving
- open source (MPL2 license)
  (`http://eigen.tuxfamily.org/`)

Program flow chart



Solving algorithm



Verification

- Verification of the code using the analytic solution of incompressible Navier-Stokes by Taylor and Green
- root mean square derivation (RMSD)

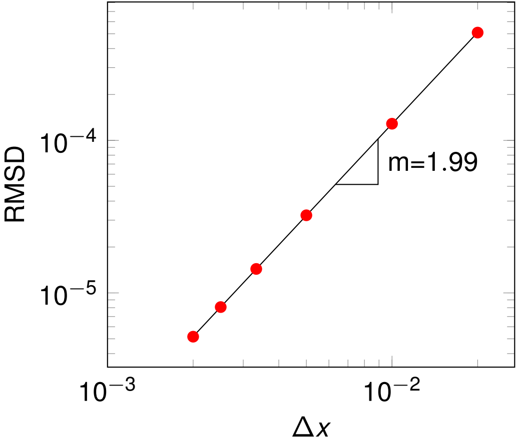
$$\mathrm{RMSD} = \frac{1}{u_{TG,max}} \sqrt{\frac{\sum^{N} (u - u_{TG})^2}{N}}$$

N: number of gridpoints

- double-logarithmic plot RMSD vs. $\Delta x / \Delta t$

$\Rightarrow$ error should decrease with slope 2 (second order)

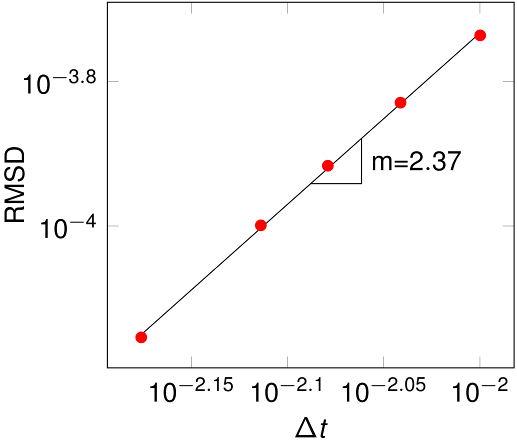
Verification - grid width



Parameters

Δt	10^{-4}
time steps	1000
Re_{TG}	10

Verification - time step



Parameters

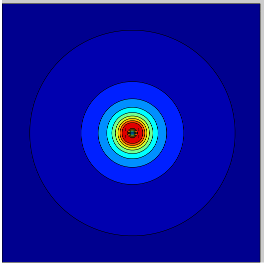
Δx	0.004
N(1D)	250
Re_{TG}	10

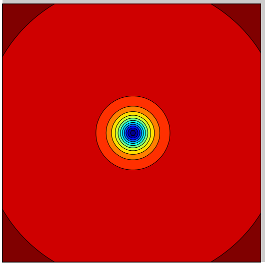
Vortex simulation

- initial velocity field with two counterrotating vortices
- two parameters:

a	characterstic radius (determines size)
Re_Γ	Reynoldsnumber of vortex (determines circumferential velocity)

$$v_\theta = \frac{\nu Re_\Gamma}{2\pi r}\left(1 - exp\left(-\frac{r^2}{a^2}\right)\right)$$


velocity magnitude


pressure

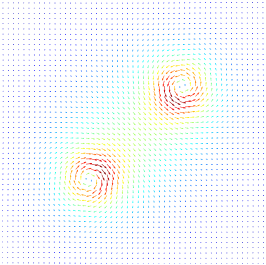
Vortex simulation

- length ratio L (normalized distance)

$$L = \frac{b}{a} \qquad \text{where } b\text{: distance}$$

- Reynoldsnumber Re_Γ (intertia vs. viscosity)

$$Re_\Gamma = \frac{\Gamma}{\nu} \qquad \text{where } \Gamma\text{: circulation}$$

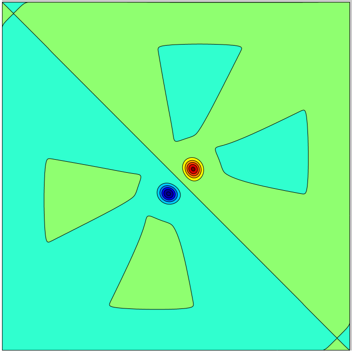


quiver plot of velocity

Vortex tracking

Karlsruhe Institute of Technology

- location of vortex centers is recorded every time step
- criteria: global maximum and minimum of vorticity



vorticity

Post-processing

- C++ code exports binary data on
  1) velocity
  2) pressure
  3) vorticity
- export interval for field data is adjustable
- vortex position is saved separately every time step
- various Matlab-scripts for graphical post-processing

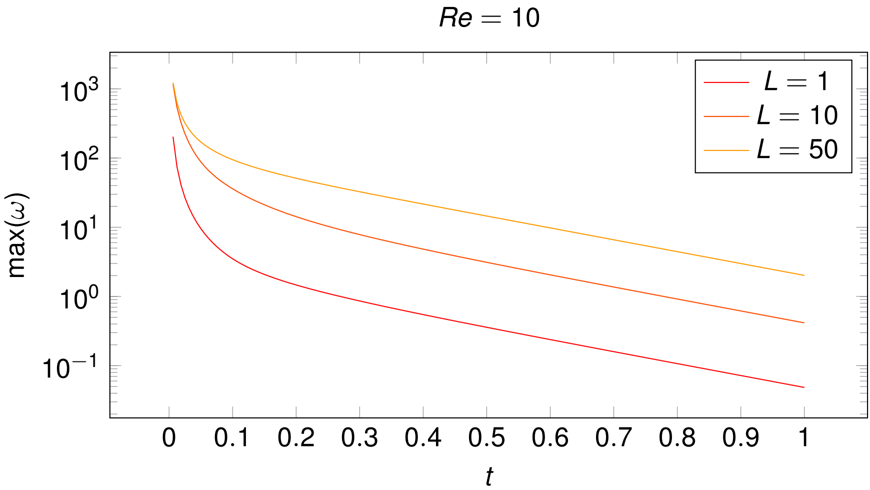
Results

Simulations have been performed for

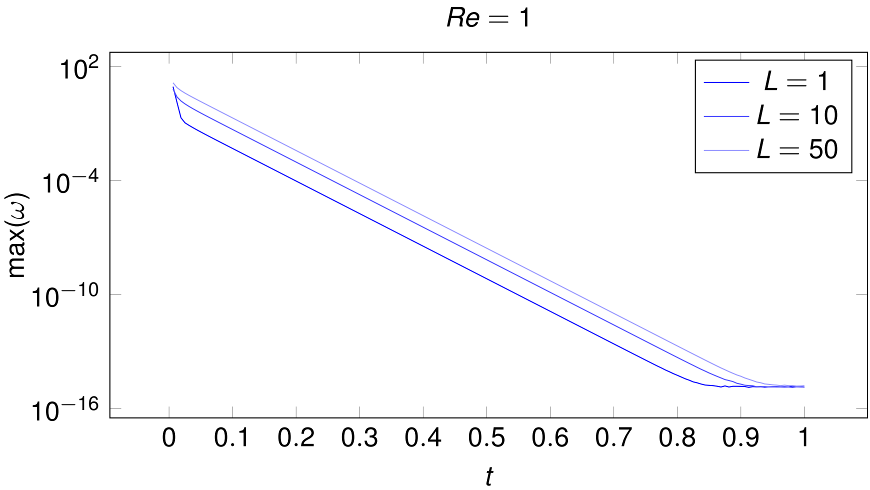
1) $Re = 1$ and $Re = 10$
2) $L = 1$, $L = 10$ and $L = 50$

Maximum of vorticity vs. time

Karlsruhe Institute of Technology



Maximum of vorticity vs. time



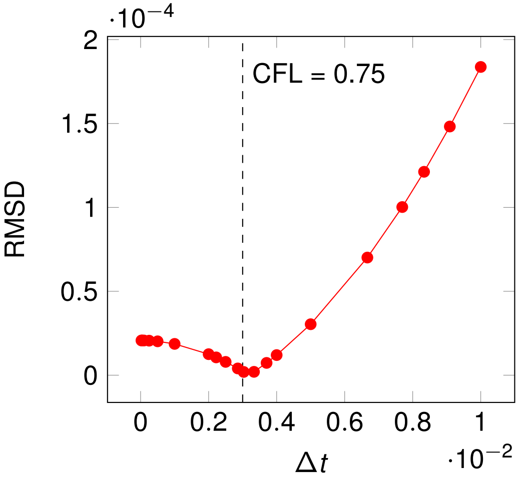
Source code

- source code is available at github
- `https://github.com/donlimon/CFD.git` (branch: final)

Git command

```
git clone -b final https://github.com/donlimon/CFD.git
```

Verification - time step (full data)



Parameters

Δx	0.004
N(1D)	250
Re_{TG}	10

Discretization - momentum (x)

x-component:

$$
\begin{aligned}
&\left(\frac{1}{\Delta t}+\frac{2}{\mathrm{Re}\,\Delta x^2}\right)u^*_{i+\frac{1}{2},j}-\frac{1}{2\,\mathrm{Re}\,\Delta x^2}\left(u^*_{i+\frac{3}{2},j}+u^*_{i-\frac{1}{2},j}+u^*_{i+\frac{1}{2},j+1}+u^*_{i+\frac{1}{2},j-1}\right)\\
&=-\frac{1}{4\Delta x}\left[3\left(u^n_{i+\frac{1}{2},j}\cdot\left(u^n_{i+\frac{3}{2},j}-u^n_{i-\frac{1}{2},j}\right)+v^n_{i+\frac{1}{2},j}\cdot\left(u^n_{i+\frac{1}{2},j+1}-u^n_{i+\frac{1}{2},j-1}\right)\right)\right.\\
&\qquad\left.-\left(u^{n-1}_{i+\frac{1}{2},j}\cdot\left(u^{n-1}_{i+\frac{3}{2},j}-u^{n-1}_{i-\frac{1}{2},j}\right)+v^{n-1}_{i+\frac{1}{2},j}\cdot\left(u^{n-1}_{i+\frac{1}{2},j+1}-u^{n-1}_{i+\frac{1}{2},j-1}\right)\right)\right]\\
&+\frac{1}{2\,\mathrm{Re}\,\Delta x^2}\left(u^n_{i+\frac{3}{2},j}+u^n_{i-\frac{1}{2},j}+u^n_{i+\frac{1}{2},j+1}+u^n_{i+\frac{1}{2},j-1}-4u^n_{i+\frac{1}{2},j}\right)+\frac{1}{\Delta t}u^n_{i+\frac{1}{2},j}
\end{aligned}
$$

Discretization - momentum (y)

y-component:

$$\left(\frac{1}{\Delta t}+\frac{2}{\mathrm{Re}\,\Delta x^2}\right)v^*_{i,j+\frac{1}{2}}-\frac{1}{2\,\mathrm{Re}\,\Delta x^2}\left(v^*_{i,j+\frac{3}{2}}+v^*_{i,j-\frac{1}{2}}+u^*_{i+1,j+\frac{1}{2}}+v^*_{i-1,j+\frac{1}{2}}\right)$$

$$=-\frac{1}{4\Delta x}\left[3\left(u^n_{i,j+\frac{1}{2}}\cdot\left(v^n_{i+1,j+\frac{1}{2}}-v^n_{i-1,j+\frac{1}{2}}\right)+v^n_{i,j+\frac{1}{2}}\cdot\left(v^n_{i,j+\frac{3}{2}}-v^n_{i,j-\frac{1}{2}}\right)\right)\right.$$

$$\left.-\left(u^{n-1}_{i,j+\frac{1}{2}}\cdot\left(v^{n-1}_{i+1,j+\frac{1}{2}}-v^{n-1}_{i-1,j+\frac{1}{2}}\right)+v^{n-1}_{i,j+\frac{1}{2}}\cdot\left(v^{n-1}_{i,j+\frac{3}{2}}-v^{n-1}_{i,j-\frac{1}{2}}\right)\right)\right]$$

$$+\frac{1}{2\,\mathrm{Re}\,\Delta x^2}\left(v^n_{i+1,j+\frac{1}{2}}+v^n_{i-1,j+\frac{1}{2}}+v^n_{i,j+\frac{3}{2}}+v^n_{i,j-\frac{1}{2}}-4v^n_{i,j+\frac{1}{2}}\right)+\frac{1}{\Delta t}v^n_{i,j+\frac{1}{2}}$$

Discretization - poisson

$$\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1} - 4\phi_{i,j} = \frac{\Delta x}{\Delta t}\left(u^*_{i+\frac{1}{2},j} - u^*_{i-\frac{1}{2},j} + v^*_{i,j+\frac{1}{2}} - v^*_{i,j-\frac{1}{2}}\right)$$

Discretization - corrector

$$u^{n+1} - u^* = -\Delta t \frac{\partial \phi}{\partial x}$$
$$v^{n+1} - v^* = -\Delta t \frac{\partial \phi}{\partial y}$$