Use of numerical simulation to solve the Couette flow problem

Mike Schuresko
Department of Applied Math and Statistics
University of California, Santa Cruz

June 20, 2006

Abstract

The Couette flow is a simplified 2d fluid dynamics problem. In this project I describe a numerical simulation I developed in order to look at turbulence in this flow.

1 Introduction

The Couette flow is a 2-dimensional fluid flow with toroidal boundaries along one axis, and fixed boundaries along the other. See Figure 1 and Figure 2.

u and v are the velocity components of the flow, while x and y are the coordinates. Along the y boundaries the fluid is allowed to slip, but not interpenetrate.

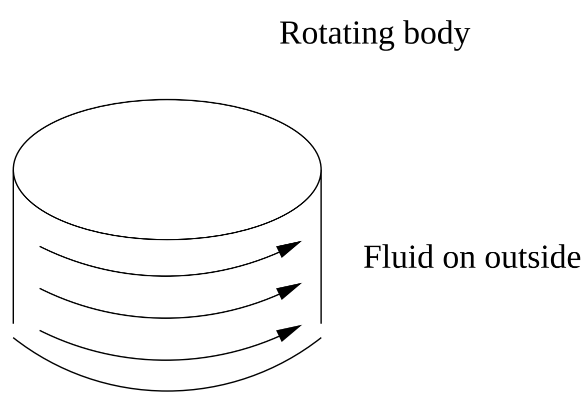

Figure 1: 3d setup of Couette flow

We will let $\vec{u} = (u, v)$.

The general equations governing this system are

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial \rho}{\partial x} + \nu(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) + F(y)$$

and

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} v\frac{\partial v}{\partial y} = -\frac{\partial \rho}{\partial y} + \nu(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2})$$

with the constraint that

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0.$$

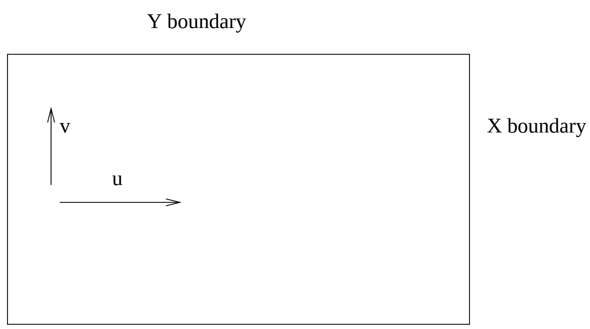

Figure 2: 2d setup of Couette flow and axis labels

2 Laminar flow

Under the assumption that flow is independent of x, this becomes

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial y} = -\frac{\partial \rho}{\partial x} + \nu\frac{\partial^2 u}{\partial y^2} + F(y)$$

and

$$\frac{\partial v}{\partial t} + v\frac{\partial v}{\partial y} = -\frac{\partial \rho}{\partial y} + \nu\frac{\partial^2 v}{\partial y^2}$$

with the constraint that

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$$

with the constraint that

$$\frac{\partial v}{\partial y} = 0$$

Combining these gives

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial y} = -\frac{\partial \rho}{\partial x} + \nu\frac{\partial^2 u}{\partial y^2} + F(y)$$

and $v = 0$ and $\frac{\partial \rho}{\partial x} = 0$ which implies

$$\frac{\partial u}{\partial t} = \nu\frac{\partial^2 u}{\partial y^2} + F(y).$$

The resulting equation is linear, and can be solved analytically quite easily.

If we let $F(y) = \sum_{k=0}^{\infty} a_k e^{2\pi k y}$ and $u(t) = \sum_{k=0}^{\infty} u_k(t) e^{2\pi k y}$ we have $\frac{d}{dt}u_k = -\nu 4k^2\pi^2 u_k + a_k$, which gives the dynamics of $u_k(t) = \frac{a_k}{\nu 4k^2\pi^2} + (u_k(0) - \frac{a_k}{\nu 4k^2\pi^2})e^{-rt}$ where $r = \nu(4k^2\pi^2)$.

There are three things of note here (assuming the forcing is only in x, which is true for this problem)

- If the system starts with no v flow, then it remains in a state of no v flow.
- If the system starts with no high-spatial-frequency flow components, it will remain in such a state
- One would be hard pressed to describe the exponential decay of superimposed modes described by this equation as "turbulent"

So achieving turbulence requires one of two things

- A forcing function with a v component
- or an initial condition with a v component.

Since our forcing function is only in x, we have to pick initial conditions with a v flow component to get any hope of turbulence.

However, the predictability of the uniform flow case makes it easy to test the numerical solver. See Section 4.

3 Numerical method

3.1 Stream function transformation

Recall that the full original equations are

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=-\frac{\partial \rho}{\partial x}+\nu(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2})+F(y)$$

and

$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}v\frac{\partial v}{\partial y}=-\frac{\partial \rho}{\partial y}+\nu(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2})$$

with the constraint that

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0.$$

As mentioned in the notes, the constraint $\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0$ is difficult to maintain numerically.

One way to ensure this constraint is to maintain the equivalent constraint of $\nabla\times\Psi e_z=ue_x+ve_y$ for some scalar function Ψ and e_x,e_y,e_z as the canonical basis of $\mathbb{R}^3$. Recall that for any function, f, in $\mathbb{R}^3\mapsto\mathbb{R}^3$, $\nabla\dot{\nabla}\times f=0$. By maintaining the constraint that Ψ exist such that $\nabla\times\Psi e_z=ue_x+ve_y$, we are maintaining that $\nabla\dot{(}ue_x+ve_y)=0$, or equivalently $\frac{\partial}{\partial x}u+\frac{\partial}{\partial y}v=0$.

Such a function, Ψ, is called a "stream function" and is a standard device in fluid dynamics (see (1)). It has the following properties

- Along a curve c with differential tangent dl and differential normal dn, the flux through the line is $\int_c<u,v>\dot{dn}=\int_c<-\frac{\partial}{\partial y}\Psi,\frac{\partial}{\partial x}\Psi>\dot{dn}=\int_c<\frac{\partial}{\partial x}\Psi,\frac{\partial}{\partial y}\Psi>\dot{dl}=\Psi_{final}-\Psi_{initial}$, making the difference between the stream function at two points the volume of flux through a curve connecting the points.
- Since streamlines are tangent to the flow, the value of Ψ must be the same along a streamline. This is very helpful for visualization.

With the addition of the stream function, the original equations become

$$\Psi_{yt}+\Psi_y\Psi_{xy}-\Psi_x\Psi_{yy}=-\frac{\partial\rho}{\partial x}+\nu(\frac{\partial^2\Psi_y}{\partial x^2}+\frac{\partial^2\Psi_y}{\partial y^2})+F(y)$$

and

$$-\Psi_{xt}+\Psi_y\Psi_{xx}+\Psi_x\Psi_{xy}=-\frac{\partial\rho}{\partial y}-\nu(\frac{\partial^2\Psi_x}{\partial x^2}+\frac{\partial^2\Psi_x}{\partial y^2})$$

Cancelling out the pressure is done by subtracting $\frac{\partial}{\partial y}$eq1 $-\frac{\partial}{\partial x}$eq2. When the terms are cancelled out, we are left with $-\Psi_{xxt}-\Psi_y\Psi_{xxx}+\Psi_x\Psi_{xxy}-\Psi_{yyt}-\Psi_y\Psi_{xyy}+\Psi_x\Psi_{yyy}=-\nu(\Psi_{xxxx}+\Psi_{xxyy}+\Psi_{yyxx}+\Psi_{yyyy})+\frac{\partial}{\partial y}F(y)$

This equation can be simplified by introducing a term called the "vorticity", ω, which is defined such that $\omega e_z=\nabla\times u$. Sincd $u=\nabla\times(\Psi e_z)$, $\omega e_z=e_z(-\frac{\partial^2}{\partial y^2}\Psi-\frac{\partial^2}{\partial x^2}\Psi)$.

The final equation is now

$$\omega_t+\Psi_y\omega_x-\Psi_x\omega_y=\nu(\Psi_x x+\Psi_y y)+F(y)_y.$$

The numerical method used is described in Table 1.

3.2 Transformed Boundary Conditions

In order to make calculations easier, we use the boundary conditions that $v=0$ and $\frac{\partial u}{\partial y}$ at the top and bottom boundaries.

This is convenient, as it entails $\omega=\nabla\times(u,v,0)$ is zero at the top and bottom boundaries. It also entails that $\frac{\partial}{\partial x}\Psi=0$ at the boundaries. The only remaining question is which constants to pick for Ψ. It turns out that there

Name: Numerical method for Couette flow
Goal: Create a stable numerical method for simulating incompressible 2d Navier-Stokes with simple boundary conditions
Assumes: Solutions exist to the Navier Stokes equations for most initial conditions

---

Set $\vec{u} = (u, v)$ according to the initial conditions.
$\{$Generate ω from numerically evaluating $\nabla \times \vec{u}\}$
for all ω_{ij} **do**
  $\omega_{ij} := \frac{u_{i,j+1} - u_{i,j-1}}{\delta Y} - \frac{u_{i+1,j} - u_{i-1,j}}{\delta X}$
end for
for all iterations i **do**
  Calculate Ψ from ω by performing conjugate gradient to invert $\nabla \dot{\nabla} \Psi = \omega$ using the last Ψ as a guess.
  if $i <$ degree(Adams Bashforth) **then**
    Calculate linear and non-linear components of ω_t
    Explicitly step ω
    Store non-linear component of ω_t
  else
    Plug past non-linear terms into Adams Bashforth
    Call the result times δT δ_{NL}
    Letting Δ be the linear operator for the Laplacian, use conjugate gradient to apply inverse of $(I + \Delta)$ in the Crank Nicholson equation $\frac{1}{2}(I + \Delta)\omega^{next}_{linear} = \frac{1}{2}(I + \Delta)\omega^{curr} + F$.
    $\omega^{next} := \omega^{next}_{linear} + \delta_{NL}$.
  end if
  if $i \mod n_{\text{print freq}} = 0$ **then**
    Calculate $(u, v, 0) = \nabla \times \Psi e_z$.
    Save out Ψ, u and v.
  end if
end for
End algorithm.

Table 1: Overall Algorithm.

are many choices for Ψ which work. I picked $\Psi = 0$ It can be proven that for any choice of interior values of ω such that $\sum_{i,j} \omega_{ij} = 0$ (this corresponds to a lack of net inflow or outflow of material due to Stokes' theorem), there exists a setting of the interior values of Ψ with 0 exterior boundaries such that $\Delta\Psi = \omega$. One simple proof is noting that Δ is a graph Laplacian operator over a connected graph – never mind, I'll finish this thought later.

3.3 Conversions

3.4 Term splitting

3.5 Numerical method

For an overview of the algorithm, see Table 1.

What follows is a listing of the various files and what they do The files described here can be found at http://www.soe.ucsc.edu/ mds/couette along with the current copy of this report.

- main.f has the main program, which consists of one procedure call. This call can be swapped out to run various test functions
- runSim.f contains three procedures
  - runSim is the main procedure and contains the initial conditions for the print frequency, the timestep (fixed for Adams Bashforth), the aspect ratio and the grid spacing
  - DoRkStep does the initial non-implicit step to kickstart semi-implicit Adams Bashforth
  - DoSIABStep does the semi-implicit adams bashforth
- iniVals.f contains functions for all the other initial conditions
- initializeFlow.f sets the initial flow. If you want a random valid flow, replace it with the file "alt init rename to initializeFlow dot f"
- testChain.f and TestPsiFromW.f just test the conversion chain
- derivs.f calulates the linear and nonlinear components of the derivative of ω
- multLat2d.f applies the linear operator Δ to a given Ψ
- conjgrad.f contains two conjugate gradient routines, one which operates over linear functions applied to 1d vectors (not currently used) and one which operates over lienar functions applied to 2d arrays
- psiFromW.f, uFromPsi.f and wFromU.f are conversion routines
- printresult.f contains the print functions I need
- arrayManip.f contains array manipulation primitives to allow me to write code that looks more like matlab code (or Fortran90 from what I have heard)

4 Testing

For the test, I picked the forcing function corresponding to $u_{initial}(x, y) = -\nu \frac{\partial^2}{\partial y^2} F(y)$ and the forcing function as described in class, and checked that I had started on a fixed point.

Then I repeated the experiment with a different initial condition that also depended solely on x and checked that the system converged to $-\nu \frac{\partial^2}{\partial y^2}$.

4.1 Numerical Diffusion

One way to test the numerical diffusion of the non-linear component would be to set viscosity to zero, change the boundary conditions to fully toroidal, initialize the flow to be 0 outside of a given non-axis-aligned channel and uniform in the channel direction inside the channel, run for one time step, and compare the Fourrier transform of the initial condition and the result.

5 Turbulence

5.1 Discussion of $v = 0$ case

5.2 Choice of initial conditions

Luckily we can pick a consistent initial flow if we pick some arbitrary function Ψ that satisfies Ψ's boundary conditions, and set the initial condition for u to be its curl.

I picked $\Psi e_z = \sin(2\pi x)\sin(2\pi y)e_z$, which is zero at the boundaries and has a curl of $< -2\pi\sin(2\pi x)\sin(2\pi y), 2\pi\cos(2\pi x)$ co

6 Visualization

In general, picking seed points from which to draw streamlines is a hard problem in visualization. The nice thing about incompressible flow in general, and the method we are using in particular, is that surfaces of constant Ψ correspond to streamlines.

In order to visualize streamlines (see Figure 3) I found the min and max values of Ψ, and used the square of sin function with a frequency of 20 times this difference to map the Ψ values onto the red channel of color.

On top of this I drew blew glyphs corresponding to (scaled) flow magnitude and direction. The ellipses on the glyphs are actually the tails of the vectors rather then the heads.

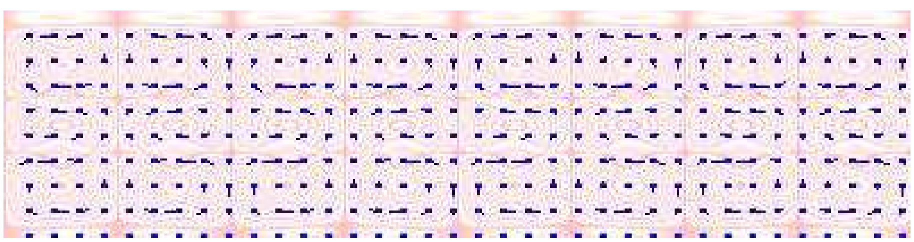

Figure 3: Single frame of Couette flow

7 Conclusions and future work

References

[1] V. Authors, "Stream function," `http://en.wikipedia.org/wiki/Stream_function`, 2003.