PHYS 447 Spring 2016

Computational fluid dynamics

Prof. Moshe Rozali *submitted by: C. Leung*

1 Introduction

This intent of this project was to learn about the physics of fluids and observe the behaviour in computer simulations. The process was carried out by starting with simplifying assumptions on the model, choosing a discretization method, then writing code in Python. At each following step, an assumption would be generalized so that the change could be observed. The equations were examined in 1-dimension first so an example will be shown before the transition to 2-dimensions.

Fluid flow equations addressed:

- linear convection
- nonlinear convection
- diffusion
- Burgers equation (convection with diffusion)
- Navier-Stokes (cavity flow and channel flow conditions)

The work here closely follows that presented in the course material for Professor L.A. Barba's Computational Fluid Dynamics course, specifically the section leading up to the Navier-Stokes equations.

2 Physics

The equations governing models of flow are described in this section.

2.1 Introduction to models in 1-dimensional form

Beginning with 1-dimensional models, we can describe the differences between the behaviours of the systems dictated by these equations. They have their analogues in 2-dimensions.

Starting off, the simplest model looked at was the 1-dimensional equation describing convective flow without diffusion. It describes the propagation of a wave with constant speed c. There is no deformation of shape of the initial condition's wave as it propagates. 1

In the nonlinear case, the constant speed is replaced by u which is the solution $u(x,t) = u_0(x - ct)$ given an initial condition $u(x,0) = u_0(x)$ The speed of propagation is different depending upon the

location. Because of this, shocks may be observed (see corresponding treatment in the numerical considerations section). It is also known as the inviscid Burgers equation. 2

Then diffusion is considered without any convection. If u is temperature, then this is the heat equation. It is governed by a second-order partial derivative. 3

For the last of the equations with time evolution, nonlinear convection and diffusion are treated simultaneously. Since viscosity is now accounted for with the diffusion term, this is the general Burgers equation. It has applications outside of fluid dynamics, for example in acoustics. 4

linear convection	$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0$	(1)
(inviscid Burgers) nonlinear convection	$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0$	(2)
linear diffusion	$\frac{\partial u}{\partial t} = \nu\frac{\partial^2 u}{\partial x^2}$	(3)
Burgers' equation	$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}$	(4)

2.2 Extension to 2-dimensional models

The expressions introduced in their 1-dimensional form are now shown extended to two dimensions.

2.2.1 Linear convection (2d)

The conversion from one- to two-dimensions is straightforward for linear convection. As two spatial components are changing, it is sufficient to duplicate the original spatial variation in the new dimension.

$$\frac{\partial u}{\partial t} + c\left(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial x}\right) = 0 \tag{5}$$

2.2.2 Nonlinear convection (2d)

Nonlinear in an extra dimension requires a different treatment. The constant velocity is now a vector-valued velocity with two dimensions (u, v), with partial derivatives, we now have coupled equations.

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

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = 0 \tag{6}$$

2.2.3 Diffusion (2d)

Diffusion affects flow equally in both components in two dimensions, so again, a duplication in the y-component works.

$$\frac{\partial u}{\partial t} = \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{7}$$

2.2.4 Burgers (2d)

As in the case with nonlinear convection, coupled PDEs are required to describe the system. The two independent variables u, and v are coupled in this system to form a vector field.

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

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \tag{8}$$

2.2.5 Navier-Stokes for incompressible fluid

Velocity and pressure need to be included and each affects the other.

We need to involve the continuity equation for fluid with constant density ρ (mass conservation) and the vector equation for conservation of momentum, which depends upon ρ. Equation (9) acts as a constraint to the equation of motion (10).

For compressible flows, we are able to relate pressure and density, but since we are dealing with the incompressible version, we have no such equation of state. A different way of coupling the equations is required.

continuity equation (conservation of mass) $$\nabla \cdot \vec{u} = 0 \tag{9}$$

conservation of momentum $$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\vec{u} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\vec{u} \tag{10}$$

Componentwise, the equations become:

conservation of mass
$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{11}$$

conservation of momentum
$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{12}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \tag{13}$$

The equations retain the same form: in conservation of momentum, the first term is the steady partial, the second and third are associated with convection, the final term is the viscous term multiplying the Laplacian of that component velocity and is associated with diffusion.

By taking derivatives of expressions12 and 13, we obtain the divergence of the momentum. That is, adding $\frac{\partial}{\partial x}$ (12) and $\frac{\partial}{\partial y}$ (13), we have, term by term:

$$\frac{\partial}{\partial t}\left(\frac{\partial u}{\partial x}\right)$$

$$+\frac{\partial u}{\partial x}\frac{\partial u}{\partial x} + u\frac{\partial^2 u}{\partial x^2}$$

$$+\frac{\partial v}{\partial x}\frac{\partial u}{\partial y} + v\frac{\partial^2 u}{\partial x \partial y}$$

$$= -\frac{1}{\rho}\frac{\partial^2 p}{\partial x^2}$$

$$+ \nu\left(\frac{\partial^3 u}{\partial x^3} + \frac{\partial}{\partial x}\frac{\partial^2 u}{\partial y^2}\right)$$

add to similar treatment of partial with respect to y on (13):

$$\frac{\partial}{\partial t}\frac{\partial v}{\partial y} + \frac{\partial u}{\partial y}\frac{\partial u}{\partial x} + u\frac{\partial}{\partial y}\frac{\partial v}{\partial x} + \frac{\partial v}{\partial y}\frac{\partial v}{\partial y} + v\frac{\partial^2 v}{\partial y^2} = -\frac{1}{\rho}\frac{\partial^2 p}{\partial y^2} + \nu\left(\frac{\partial}{\partial y}\frac{\partial^2 v}{\partial x^2} + \frac{\partial^3 v}{\partial y^3}\right)$$

the left-hand side becomes

$$\begin{aligned}&\frac{\partial}{\partial t}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) + \frac{\partial u}{\partial x}\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y}\frac{\partial u}{\partial x} + u\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial}{\partial y}\frac{\partial v}{\partial x}\right) + \frac{\partial v}{\partial x}\frac{\partial u}{\partial y} + \frac{\partial v}{\partial y}\frac{\partial v}{\partial y} + v\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)\\ =&\frac{\partial}{\partial t}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) + \frac{\partial u}{\partial x}\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y}\frac{\partial u}{\partial x} + u\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) + \frac{\partial v}{\partial x}\frac{\partial u}{\partial y} + \frac{\partial v}{\partial y}\frac{\partial v}{\partial y} + v\frac{\partial}{\partial y}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)\end{aligned}$$

looking at the common terms in parentheses (divergence of velocity), we know these = 0 from the conservation of mass equation, which we must obey (11), so we are left with the remaining terms

$$= \frac{\partial u}{\partial x}\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y}\frac{\partial u}{\partial x} + \frac{\partial v}{\partial x}\frac{\partial u}{\partial y} + \frac{\partial v}{\partial y}\frac{\partial v}{\partial y} = \left(\frac{\partial u}{\partial x}\right)^2 + 2\frac{\partial v}{\partial x}\frac{\partial u}{\partial y} + \left(\frac{\partial v}{\partial y}\right)^2 \tag{14}$$

the right-hand side becomes

$$\begin{aligned} &-\frac{1}{\rho}\left(\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2}\right) + \nu\left(\frac{\partial^3 u}{\partial x^3} + \frac{\partial}{\partial x}\frac{\partial^2 u}{\partial y^2} + \frac{\partial}{\partial y}\frac{\partial^2 v}{\partial x^2} + \frac{\partial^3 v}{\partial y^3}\right) \\ &= -\frac{1}{\rho}\left(\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2}\right) + \nu\left[\frac{\partial^2}{\partial x^2}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) + \frac{\partial^2}{\partial y^2}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)\right] \end{aligned}$$

again, apply equation (11) to yield

$$= -\frac{1}{\rho}\left(\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2}\right) \tag{15}$$

Putting LHS (14) and RHS (15) together, we have

$$\left(\frac{\partial u}{\partial x}\right)^2 + 2\frac{\partial v}{\partial x}\frac{\partial u}{\partial y} + \left(\frac{\partial v}{\partial y}\right)^2 = -\frac{1}{\rho}\left(\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2}\right)$$

(re-arrange terms)

$$\Rightarrow \quad \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = -\rho\left[\left(\frac{\partial u}{\partial x}\right)^2 + 2\frac{\partial v}{\partial x}\frac{\partial u}{\partial y} + \left(\frac{\partial v}{\partial y}\right)^2\right] \tag{16}$$

which has form

$$\Rightarrow \quad \nabla^2 p = function$$

This is Poisson's equation for pressure in two dimensions. Equations (9) and 10) are satisfied simultaneously. This relation between pressure and velocity can be used to couple the equations for Navier-Stokes. (When RHS is 0, it is the Laplace equation.) f is the function such that pressure satisfies the Poisson equation.

3 Numerical schemes and considerations

In some simpler cases, the discretization of the above continuous equations can be a straightforward translation. In general, the following differencing schemes are used for this project:

- time stepping: forward difference
- convection: backward difference
- diffusion: central difference, second order

The discretized equations are solved for the values of interest. The value at the next time step, is dependent upon various values of the current time step. We start with given initial conditions.

The spatial component is discretized into a grid according to the number of dimensions and solutions are found for each point over a set number of time steps, or after a specified tolerance is satisfied indicate steady state having been reached.

Superscripts n are used for time steps, and subscripts i and j for spatial grid points.

Solving these equations has some issues that do not arise in the continuous form and these will be addressed along the way in the examples that follow.

3.1 Discretization examples (with transposition to solve for next iterative step)

Simpler examples of discretization will be shown to have the idea before moving on to more generalized models. However, all of the differencing schemes here are based on using a Taylor expansion about a point to obtain the definition of the derivative and in turn we use an approximation to that derivative by disregarding higher order terms.

3.1.1 Linear Convection: 1-d to 2-d

With forward difference on the time dependent term and backward difference on the second, Equation (1) becomes:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + c\frac{u_i^n - u_{i-1}^n}{\Delta x} = 0 \tag{17}$$

Transpose this to get values at time t_{n+1} at position x_i.

$$u_i^{n+1} = u_i^n - \frac{c\Delta t}{\Delta x}(u_i^n - u_{i-1}^n) \tag{18}$$

In two dimensions, we have x- and y-components. From (5) or above (17), we get:

$$\frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t} + c\frac{u_{i,j}^n - u_{i-1,j}^n}{\Delta x} + c\frac{u_{i,j}^n - u_{i,j-1}^n}{\Delta y} = 0 \tag{19}$$

$$\Rightarrow \quad u_{i,j}^{n+1} = u_{i,j}^n - \frac{c\Delta t}{\Delta x}(u_{i,j}^n - u_{i-1,j}^n) - \frac{c\Delta t}{\Delta y}(u_{i,j}^n - u_{i,j-1}^n) \tag{20}$$

The subsequent models will be discretized in their 2-dimensional forms. But first, it would be helpful to address some constraints in using differencing methods in numerical schemes.

3.1.2 Accuracy and the CFL condition

When using these methods to compute solutions, there are sources of error that will alter the results. Awareness of these effects will help to interpret the results correctly. Firstly, there is the

error introduced from the discretizations being derived from approximations. If the grid spacing is wide, these errors will dominate over the imprecision inherent to the computer architecture and the selected precision level in the programming language.

If we feed an initial condition of a particular waveform to the linear convection model, the expected solution would be the same shape waveform in another location after being convected over time, with speed u without any other effect. However, we do see a shape change in Figure 1. The initial condition is a hat function and as it is convected, it also accumulates round-off error at every iteration. This numerical diffusion is from the method of computation and not expected in the physics of the system. The effect can be somewhat mitigated by taking a finer spatial mesh, as in Figure 2.

It would be tempting to take smaller and smaller grid sizes, but a problem arises when the grid size is smaller than the distance moved by a point on the waveform in one timestep, i.e. if $u\Delta t > \Delta x$. Non-physical artefacts appear in the computations plotted in Figure 4.

To prevent this from occurring. The size of the timestep and grid size must be constrained against each other using

$$\sigma = \frac{u\Delta t}{\Delta x} \leq \sigma_{max} \tag{21}$$

where σ is the Courant number and the u is the speed (non-negative). The constraint is called the Courant-Friedrichs-Lewy or CFL condition. For explicit methods (as we have here), σ should be less than 1. This will be a consideration for any part of this project where values may change faster than the timestep is able to show so it will be in effect at any point onward from here. Adding the condition to decide timesteps based on chosen gridspacing (or vice versa) will prevent these undesired artefacts from appearing.

3.1.3 Nonlinear convection

As with equations in (6), we need to handle non-constant velocity fields u and v.

$$\begin{aligned}
&\frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t} + u_{i,j}^n \frac{u_{i,j}^n - u_{i-1,j}^n}{\Delta x} + v_{i,j}^n \frac{u_{i,j}^n - u_{i,j-1}^n}{\Delta y} = 0 \\
&\frac{v_{i,j}^{n+1} - v_{i,j}^n}{\Delta t} + u_{i,j}^n \frac{v_{i,j}^n - v_{i-1,j}^n}{\Delta x} + v_{i,j}^n \frac{v_{i,j}^n - v_{i,j-1}^n}{\Delta y} = 0
\end{aligned} \tag{22}$$

$\Rightarrow$

$$\begin{aligned}
u_{i,j}^{n+1} &= u_{i,j}^n - \frac{u_{i,j}^n \Delta t}{\Delta x}(u_{i,j}^n - u_{i-1,j}^n) - \frac{v_{i,j}^n \Delta t}{\Delta y}(u_{i,j}^n - u_{i,j-1}^n) \\
v_{i,j}^{n+1} &= v_{i,j}^n - \frac{u_{i,j}^n \Delta t}{\Delta x}(v_{i,j}^n - v_{i-1,j}^n) - \frac{v_{i,j}^n \Delta t}{\Delta y}(v_{i,j}^n - v_{i,j-1}^n)
\end{aligned} \tag{23}$$

3.1.4 Diffusion

This is diffusion without convection from equation (7). This has second-order partial derivatives in the spatial components so a second-order scheme will be used. A central difference is appropriate to the isotropic nature of diffusion; the value at a particular point is subjected to the numerical influence of its neighbours in a manner that is independent of which direct the neighbour is.

So the discretization becomes:

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} = \nu \left(\frac{u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}}{\Delta x^2} + \frac{u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}}{\Delta y^2} \right) \tag{24}$$

$$\Rightarrow \quad u_{i,j}^{n+1} = u_{i,j}^{n} + \frac{\nu \Delta t}{\Delta x}(u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}) + \frac{\nu \Delta t}{\Delta y}(u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}) \tag{25}$$

3.1.5 Burgers' Equation

This handles nonlinear convective flow with diffusion simultaneously. Discretizing from equations (8) goes as expected from the the previous steps.

$$\begin{aligned}
&\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} + u_{i,j}^{n}\frac{u_{i,j}^{n} - u_{i-1,j}^{n}}{\Delta x} + v_{i,j}^{n}\frac{u_{i,j}^{n} - u_{i,j-1}^{n}}{\Delta y} = \nu \left(\frac{u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}}{\Delta x^2} + \frac{u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}}{\Delta y^2} \right) \\
&\frac{v_{i,j}^{n+1} - v_{i,j}^{n}}{\Delta t} + u_{i,j}^{n}\frac{v_{i,j}^{n} - v_{i-1,j}^{n}}{\Delta x} + v_{i,j}^{n}\frac{v_{i,j}^{n} - v_{i,j-1}^{n}}{\Delta y} = \nu \left(\frac{v_{i+1,j}^{n} - 2v_{i,j}^{n} + v_{i-1,j}^{n}}{\Delta x^2} + \frac{v_{i,j+1}^{n} - 2v_{i,j}^{n} + v_{i,j-1}^{n}}{\Delta y^2} \right)
\end{aligned} \tag{26}$$

$\Rightarrow$

$$\begin{aligned}
u_{i,j}^{n+1} &= u_{i,j}^{n} - \frac{u_{i,j}^{n}\Delta t}{\Delta x}(u_{i,j}^{n} - u_{i-1,j}^{n}) - \frac{v_{i,j}^{n}\Delta t}{\Delta y}(u_{i,j}^{n} - u_{i,j-1}^{n}) + \nu \Delta t \left(\frac{u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}}{\Delta x^2} + \frac{u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}}{\Delta y^2} \right) \\
v_{i,j}^{n+1} &= v_{i,j}^{n} - \frac{u_{i,j}^{n}\Delta t}{\Delta x}(v_{i,j}^{n} - v_{i-1,j}^{n}) - \frac{v_{i,j}^{n}\Delta t}{\Delta y}(v_{i,j}^{n} - v_{i,j-1}^{n}) + \nu \Delta t \left(\frac{v_{i+1,j}^{n} - 2v_{i,j}^{n} + v_{i-1,j}^{n}}{\Delta x^2} + \frac{v_{i,j+1}^{n} - 2v_{i,j}^{n} + v_{i,j-1}^{n}}{\Delta y^2} \right)
\end{aligned} \tag{27}$$

3.2 Navier-Stokes

At this point, some of the expressions that we have in the continuous case do not work so well when directly translated to discretized form for our numerical computations. We will see that in trying to iteratively solve a discretized Poisson equation, while we would like to have a divergence-free velocity, due to numerical effects, it may not always equal zero at each iteration. The strategy used here will be to calculate what the conservation of momentum equation should be at the next step but then before considering the step complete, do an intermediate calculation for the divergence being zero, thereby preserving continuity (conservation of mass), and only then complete that iteration.

3.2.1 Discretizing the momentum equation

First, we examine how the problem of non-zero divergence of velocity may arise when we discretize the equation of motion (10) keeping it in vector form.

$$\begin{aligned} & \frac{\vec{u}^{n+1} - \vec{u^n}}{\Delta t} + \vec{u}^n \cdot \nabla \vec{u}^n = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \vec{u}^n \\ \Rightarrow \quad & \vec{u}^{n+1} = \vec{u}^n + \Delta t \left[-\vec{u}^n \cdot \nabla \vec{u}^n - \frac{1}{\rho}\nabla p - \nu \nabla^2 \vec{u}^n \right] \end{aligned} \tag{28}$$

Following similar steps as in the continuous case, we take the divergence, here $\nabla\cdot(28)$.

$$\nabla \cdot \vec{u}^{n+1} = \nabla \cdot \vec{u}^n + \Delta t \left[-\nabla \cdot \vec{u}^n \cdot \nabla \vec{u}^n - \frac{1}{\rho}\nabla^2 p - \nu \nabla (\nabla^2 \cdot \vec{u}^n) \right] \tag{29}$$

To satisfy continuity (9), the divergence of the velocity (vector) should be zero, that is, $\nabla \cdot \vec{u}^{n+1} = 0$, however it is possible that $\nabla \cdot \vec{u}^n \neq 0$ due to numerical errors during computation of the velocity field for Navier-Stokes during the previous step. Because of this, the discretized version cannot simply have $\nabla \cdot \vec{u}^n$ cancelled out at every occurrence above as we did for the non-discretized case. So compute pressure such that (9) is satisfied. Then use that intermediate step's velocity field solution in the Poisson equation. The momentum equation and the Poisson equation are both solved within one iterative step. When solving Poisson, force divergence to be zero in order to have a corrected value for velocity, then use that vakue for the momentum. Alternating between the two approximates simultaneous satisfaction of both equations.

3.2.2 Solving discrete Poisson

Recall that Poisson has the form (35), so rearrange (29) accordingly (with its LHS=0 since we must have that $\vec{u}^{n+1} = 0$ for continuity to be satisfied).

$$\frac{-\nabla \cdot \vec{u}^n}{\Delta t} = -\nabla \cdot (\vec{u}^n \cdot \nabla \vec{u}^n) - \frac{1}{\rho}\nabla^2 p - \nu \nabla^2 (\nabla \vec{u}^n)$$

$$\Rightarrow \quad \nabla^2 p = -\rho \left[\frac{-\nabla \cdot \vec{u}^n}{\Delta t} + \nabla \cdot (\vec{u}^n \cdot \nabla \vec{u}^n) + \nu \nabla^2 (\nabla \cdot \vec{u}^n) \right] \tag{30}$$

This is the equation that needs to be solved next.

To aid in developing a numerical setup for complete Navier-Stokes, we start with some simpler cases to build up smaller pieces of code. We begin with Laplace's equation and progress to Poisson.

3.2.3 Laplace

Start with Laplace for the purpose of computational stepping. This will be a simplified form that will help to build up to the full forms required for computation.

The Laplace equation shows a steady state of a system given boundary conditions. There is no time parameter. It is the Poisson equation without a source term (RHS = 0).

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y} = 0 \tag{31}$$

This is discretized using second-order central differencing. As the Laplace operator here has an effect like that of diffusion, in that it is isotropic. A central differencing scheme is appropriate since it matches the underlying physics. Over the entire grid of solutions, each point is solved using contributions from the nearest neighbours in the x and y directions. This is a five-point stencil used to obtain the solution for that central point. Calculations are repeated for every point on the grid.

These pressure values p (for the entire grid) all happen at "false time-step" n. There is no time-step, but we do solve by iteratively solving at n every time. The "time" superscript seen in the discretizations for the previous models will be omitted for these terms since they would all be n.

$$\frac{p_{i+1,j} - 2p_{i,j} + p_{i-1,j}}{\Delta x^2} + \frac{p_{i,j+1} - 2p_{i,j} + p_{i,j-1}}{\Delta y^2} = 0 \tag{32}$$

The next step is to isolate $p_{i,j}$, the value solved on each iteration.

$$\Delta y^2(p_{i+1,j} - 2p_{i,j} + p_{i-1,j}) + \Delta x^2(p_{i,j+1} - 2p_{i,j} + p_{i,j-1}) = 0$$

$$\Rightarrow \quad \Delta y^2(p_{i+1,j} + p_{i-1,j}) + \Delta x^2(p_{i,j+1} + p_{i,j-1}) = 2p_{i,j}(\Delta x^2 + \Delta y^2)$$

$$\Rightarrow \quad p_{i,j} = \frac{\Delta y^2(p_{i+1,j} + p_{i-1,j}) + \Delta x^2(p_{i,j+1} + p_{i,j-1})}{2(\Delta x^2 + \Delta y^2)} \tag{33}$$

To simplify the formula, we can choose the same grid spacing in both directions, i.e. let $\Delta x = \Delta y$

$$\Rightarrow \quad p_{i,j} = \frac{\Delta x^2(p_{i+1,j} + p_{i-1,j}) + \Delta x^2(p_{i,j+1} + p_{i,j-1})}{2(\Delta x^2 + \Delta x^2)}$$

$$\Rightarrow \quad p_{i,j} = \frac{p_{i+1,j} + p_{i-1,j} + p_{i,j+1} + p_{i,j-1}}{4} \tag{34}$$

Equation (34) can be written as a system of linear equations which in matrix form becomes a band matrix, i.e. a matrix that is zeros except for the diagonal two entries above, and two entries below the diagonal. From this perspective, the iterative solver can be seen as the Jacobi method of solving linear systems.

Aside: the Jacobi method

Any system of equations with vector of unknowns $\vec{x} = [x_1, x_2, ..., x_n]$ can be written in the form $\vec{x} = T\vec{x} + c$ where T is the band matrix. It can be formed by isolating each x_i in turn, for each

given equation in the system. The initial input $\vec{x}$ is typically a vector of all zeros. Used on the right-hand-side, this produces a first step to approximating the solution for true x. At the next iteration, those solutions are input. The process repeats until subsequent solutions stop changing by some chosen tolerance. This process can be accelerated by using most-recently calculated x_i values from within the same iteration. This is the Gauss-Seidel method. The values converge faster since the more accurate values are being used as early as possible. A further improvement would be to use successive over-relaxation.

3.2.4 Poisson

The Laplace equation from above could be seen as a simplified version of what we do now. The Poisson equation has a source term b on the right-hand-side. (Compare to the Laplace equation (31).) The Poisson equation is required to fully represent the effect of pressure in the Navier-Stokes equations, and it has the effect of relaxing initial sources in the field.

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y} = b \tag{35}$$

We can discretize in the same manner as for Laplace above.

$$\frac{p_{i+1,j} - 2p_{i,j} + p_{i-1,j}}{\Delta x^2} + \frac{p_{i,j+1} - 2p_{i,j} + p_{i,j-1}}{\Delta y^2} = b_{i,j} \tag{36}$$

Again, the next step is transposition to isolate $p_{i,j}$.

$$\Delta y^2(p_{i+1,j} - 2p_{i,j} + p_{i-1,j}) + \Delta x^2(p_{i,j+1} - 2p_{i,j} + p_{i,j-1}) = b_{i,j}$$

$$\Rightarrow \quad p_{i,j} = \frac{1}{2(\Delta x^2 + \Delta y^2)} \left[\Delta y^2(p_{i+1,j} + p_{i-1,j}) + \Delta x^2(p_{i,j+1} + p_{i,j-1}) - b_{i,j}\Delta x^2 \Delta y^2\right] \tag{37}$$

With $\Delta x = \Delta y$, this becomes

$$p_{i,j} = \frac{1}{4} \left[(p_{i+1,j} + p_{i-1,j}) + (p_{i,j+1} + p_{i,j-1}) - b_{i,j}\Delta x^2\right] \tag{38}$$

where all terms are from the n-th iteration, including the source term $b_{i,j}$. So to interpret this system: The pressure at a grid point depends on contributions from its four nearest neighbours, but decreased by the initial source amount weighted by grid-spacing squared. For a fine grid, the pressure decrease is less for a given point (but there are more points nearby to be affected by this smaller influence also). The magnitude of relaxation decreases with each iteration as the pressure values become closer to the source values, smoothed over differences in neighbouring points. (See this relaxation effect graphically in the results section(12).)

3.2.5 Modelling cavity flow

With different boundary conditions, we can model different types of flow using the Navier-Stokes equations.

We have equations for motion (12) and (13) and one for pressure (16) from before. Now we use their discretized forms. Start with (12), for u-momentum. Use forward difference in time, backward difference for convective terms, and second-order central difference for diffusive terms.

$$\begin{aligned}&\frac{u_{i,j}^{n+1}-u_{i,j}^{n}}{\Delta t}+u_{i,j}^{n}\frac{u_{i,j}^{n}-u_{i-1,j}^{n}}{\Delta x}+v_{i,j}^{n}\frac{u_{i,j}^{n}-u_{i-1,j}^{n}}{\Delta y}\\&\qquad=-\frac{1}{\rho}\frac{p_{i+1,j}-p_{i-1,j}}{2\Delta x}+\nu\left(\frac{u_{i+1,j}^{n}-2u_{i,j}^{n}+u_{i-1,j}^{n}}{\Delta x^{2}}+\frac{u_{i,j+1}^{n}-2u_{i,j}^{n}+u_{i,j-1}^{n}}{\Delta y^{2}}\right)\end{aligned}$$

Isolate $u_{i,j}^{n+1}$ to the LHS.

$$\begin{aligned}u_{i,j}^{n+1}=u_{i,j}^{n}+\Delta t\Bigg[&-\frac{1}{\rho}\frac{p_{i+1,j}-p_{i-1,j}}{2\Delta x}-u_{i,j}^{n}\frac{u_{i,j}^{n}-u_{i-1,j}^{n}}{\Delta x}-v_{i,j}^{n}\frac{u_{i,j}^{n}-u_{i-1,j}^{n}}{\Delta y}\\&+\nu\left(\frac{u_{i+1,j}^{n}-2u_{i,j}^{n}+u_{i-1,j}^{n}}{\Delta x^{2}}+\frac{u_{i,j+1}^{n}-2u_{i,j}^{n}+u_{i,j-1}^{n}}{\Delta y^{2}}\right)\Bigg]\end{aligned}\tag{39}$$

Likewise, for v.

$$\begin{aligned}&\frac{v_{i,j}^{n+1}-v_{i,j}^{n}}{\Delta t}+u_{i,j}^{n}\frac{v_{i,j}^{n}-v_{i-1,j}^{n}}{\Delta x}+v_{i,j}^{n}\frac{v_{i,j}^{n}-v_{i-1,j}^{n}}{\Delta y}\\&\qquad=-\frac{1}{\rho}\frac{p_{i+1,j}-p_{i-1,j}}{2\Delta y}+\nu\left(\frac{v_{i+1,j}^{n}-2v_{i,j}^{n}+v_{i-1,j}^{n}}{\Delta x^{2}}+\frac{v_{i,j+1}^{n}-2v_{i,j}^{n}+v_{i,j-1}^{n}}{\Delta y^{2}}\right)\end{aligned}$$

$\Rightarrow$

$$\begin{aligned}v_{i,j}^{n+1}=v_{i,j}^{n}+\Delta t\Bigg[&-\frac{1}{\rho}\frac{p_{i+1,j}-p_{i-1,j}}{2\Delta y}-u_{i,j}^{n}\frac{v_{i,j}^{n}-v_{i-1,j}^{n}}{\Delta x}-v_{i,j}^{n}\frac{v_{i,j}^{n}-v_{i-1,j}^{n}}{\Delta y}\\&+\nu\left(\frac{v_{i+1,j}^{n}-2v_{i,j}^{n}+v_{i-1,j}^{n}}{\Delta x^{2}}+\frac{v_{i,j+1}^{n}-2v_{i,j}^{n}+v_{i,j-1}^{n}}{\Delta y^{2}}\right)\Bigg]\end{aligned}\tag{40}$$

The Poisson equation (16), discretizes directly as:

$$\frac{p^n_{i+1,j} - 2p^n_{i,j} + p^n_{i-1,j}}{\Delta x^2} + \frac{p^n_{i,j+1} - 2p^n_{i,j} + p^n_{i,j-1}}{\Delta y^2}$$
$$= -\rho \left[\left(\frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} \right)^2 + 2\frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y} \frac{v_{i+1,j} - v_{i-1,j}}{2\Delta x} + \left(\frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y} \right)^2 \right]$$

$$\Rightarrow \quad p_{i,j} = -\frac{1}{2(\Delta x^2 + \Delta y^2)} \cdot$$
$$\{\rho \left[\frac{\Delta y^2}{4}(u_{i-1,j} - u_{i-1,j}) + \frac{\Delta x \Delta y}{2}(v_{i+1,j} - v_{i-1,j})(u_{i,j+1} - u_{i,j-1}) \right.$$
$$\left. + \frac{\Delta x^2}{4}(v_{i-1,j} - v_{i-1,j}) \right]$$
$$+\Delta y^2(p_{i+1,j} + p_{i-1,j})$$
$$+\Delta x^2(p_{i,j+1} + p_{i,j-1})\}$$

However, when we discretize to solve for the problem of cavity flow, we do so iteratively using a pseudo time-step that eventually brings us to a steady state solution.

$$\frac{p^n_{i+1,j} - 2p^n_{i,j} + p^n_{i-1,j}}{\Delta x^2} + \frac{p^n_{i,j+1} - 2p^n_{i,j} + p^n_{i,j-1}}{\Delta y^2}$$
$$= \frac{\rho}{\Delta t} \left(\frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} + \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y} \right)$$
$$+ \rho \left(-\frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} - 2\frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y} \frac{v_{i+1,j} - v_{i-1,j}}{2\Delta x} - \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y} \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y} \right) \tag{41}$$

Solve for pressure at point (i, j). All values are at "time step" n. The LHS is equivalent to

$$\frac{\Delta y^2 p_{i+1,j} - 2\Delta y^2 p_{i,j} + \Delta y^2 p_{i-1,j} + \Delta x^2 p_{i,j+1} - 2\Delta x^2 p_{i,j} + \Delta x^2 p_{i,j-1}}{\Delta x^2 \Delta y^2}$$
$$\Leftrightarrow \frac{-2\Delta y^2 p_{i,j} - 2\Delta x^2 p_{i,j}}{\Delta x^2 \Delta y^2} + \frac{\Delta y^2 p_{i+1,j} + \Delta y^2 p_{i-1,j} + \Delta x^2 p_{i,j+1} + \Delta x^2 p_{i,j-1}}{\Delta x^2 \Delta y^2}$$
$$\Leftrightarrow p_{i,j} \frac{2(-\Delta y^2 - \Delta x^2)}{\Delta x^2 \Delta y^2} + \frac{\Delta y^2 (p_{i+1,j} + p_{i-1,j}) + \Delta x^2 (p_{i,j+1} + p_{i,j-1})}{\Delta x^2 \Delta y^2}$$

Now the whole expression with $p_{i,j}$ isolated on the LHS is the following:

$$p_{i,j} = \frac{-\Delta x^2 \Delta y^2}{2(\Delta y^2 + \Delta x^2)} \left[\frac{\rho}{\Delta t} \left(\frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} + \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y} \right) \right.$$
$$+ \rho \left(- \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} - 2 \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y} \frac{v_{i+1,j} - v_{i-1,j}}{2\Delta x} - \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y} \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y} \right)$$
$$\left. - \frac{\Delta y^2 (p_{i+1,j} + p_{i-1,j}) + \Delta x^2 (p_{i,j+1} + p_{i,j-1})}{\Delta x^2 \Delta y^2} \right]$$

$$= \frac{-\Delta x^2 \Delta y^2 \rho}{2(\Delta y^2 + \Delta x^2)} \left[\frac{1}{\Delta t} \left(\frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} + \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y} \right) \right.$$
$$\left. - \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} - 2 \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y} \frac{v_{i+1,j} - v_{i-1,j}}{2\Delta x} - \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y} \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y} \right]$$
$$+ \frac{\Delta y^2 (p_{i+1,j} + p_{i-1,j}) + \Delta x^2 (p_{i,j+1} + p_{i,j-1})}{2(\Delta y^2 + \Delta x^2)} \tag{42}$$

3.2.6 Modelling channel flow

To model channel flow, there is an added source term for one of the components. It acts as a pressure gradient in one direction to simulate flow in a channel along that axis. Selecting the direction of the flow to be along the x-component, the u-momentum expression gets the extra source term F. So (12) becomes

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} = \frac{1}{\rho} \frac{\partial p}{\partial x} + \nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^u}{\partial y^2} \right) + F \tag{43}$$

while the v-momentum expression is unchanged. (We could have chosen to swap the axes.) The Poisson expression is the same as in the cavity flow case. Discretizations for the latter two expressions will be the same as the previous simulation. The discretization for the u-momentum becomes

$$u_{i,j}^{n+1} = u_{i,j}^n + \Delta t \left[- \frac{1}{\rho} \frac{p_{i+1,j} - p_{i-1,j}}{2\Delta x} - u_{i,j}^n \frac{u_{i,j}^n - u_{i-1,j}^n}{\Delta x} - v_{i,j}^n \frac{u_{i,j}^n - u_{i-1,j}^n}{\Delta y} \right.$$
$$+ \nu \left(\frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{\Delta x^2} + \frac{u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n}{\Delta y^2} \right)$$
$$\left. + F \right] \tag{44}$$

so there is an addition amount $F\Delta t$ for the u-velocity every time step.

4 Coding strategy

The coding followed very closely the approach used by the online CFD course material from Professor Barba. A few alterations were made to the style and output, but the computations shadow the numerical methods above so the logic remains largely the same. What remains in common is that the language used is Python, using common modules `numpy` and `matplotlib`. While exploring the computations, the code began in nested loop format, the inner loops calculating the values for each spatial position within that timestep, the outer loops completing once per timestep. Final code vectorized the computations so that one of the nestings was eliminated and it runs faster (due to optimizations made by `numpy` which allow the array operations).

4.1 Alterations

Beyond the above, some changes and additions were made for this project.

4.1.1 Keeping solutions time-array

The code was altered to retain solutions at each timestep. The original code overwrote the previous steps and produced only the final result. Holding the intermediate solutions in an array (2-d for the 1-d spatial problems, 3-d for the 2-d spatial ones) allows revisiting interim steps and also animation of the results as they evolved over time.

4.1.2 Parametrization in functions

Each model had operations grouped by functions so that the code could be reused and would easier for reading and future modification in their modularized form. The functions were parametrized so that it is easier to change values (such as timesteps, or initial conditions) in one place (from the caller of the function) rather than having to modify it in the body of the code, possibly in several places.

4.1.3 Visualization

One of the separated functions is for plotting. Having the entire set of solutions, any timestep can be plotted when provided as an input to the plot function. Alternatively, another function was made for animating the time evolution of the system with the option of visualization as a surface plot or a flat colourmap of values.

4.2 Results

Running code is available at at the repository for this project, the same code used for generating the results below. For each case here, particular initial conditions were specified and a selection of notable results are shown. For more detail, animation and full solutions using different parameters can be tested at https://github.com/cphl/cfd the code that is frozen for the purpose of reference

of the generated plots is stored in the folder `report_code` but the files outside of that folder have functionality beyond plotting stills for this report.

4.2.1 Linear Convection

This is the simplest 1-d case. Plots at multiple timesteps have been overlaid. The initial condition is the hat function starting on the left. Note the false diffusion caused by roundoff in computations such that it ends up being a shorter, wider waveform without hat shape on the right. Figure 1 from `01-1_linconv.py`.

Figure 1: 1-d linear convection only. Numerical diffusion. Number of grid points nx=41

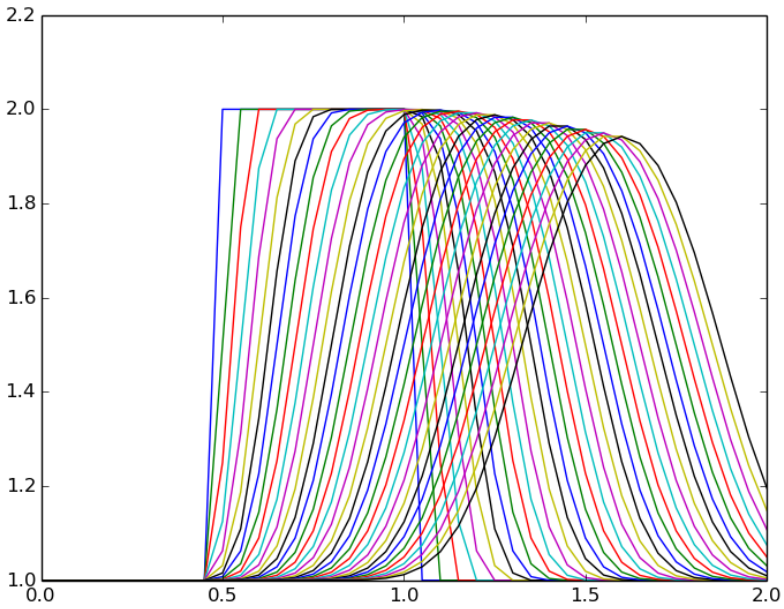


Figure 2 shows that making a finer spatial grid reduces the numerical diffusion and it retains more of the initial condition's hat shape.

Figure 2: 1-d linear convection only. Less false diffusion. Number of grid points nx=71

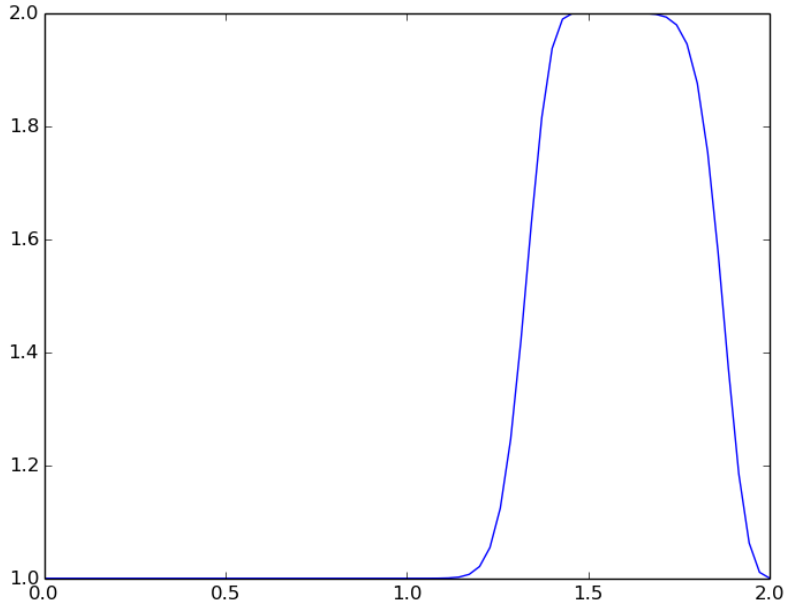
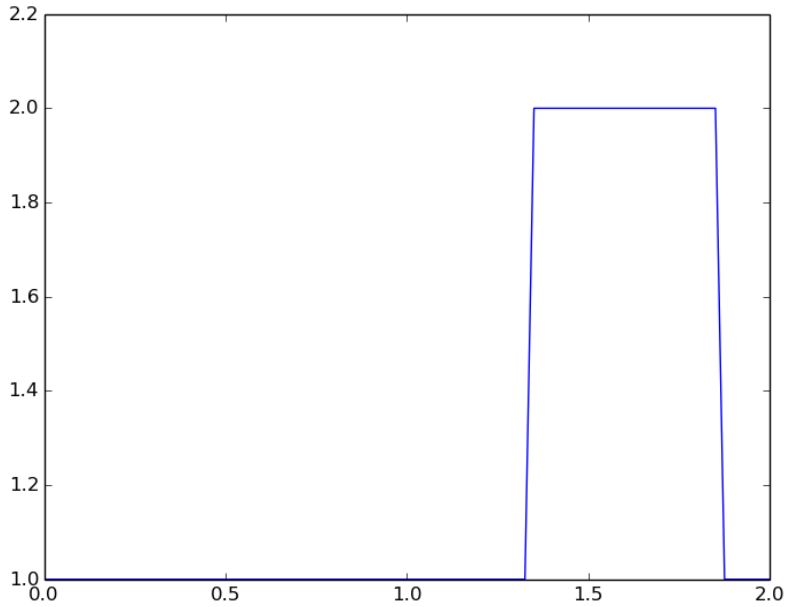


Figure 3: 1-d linear convection only. Hat mostly retained. Number of grid points nx=81



In figure 3 it appears that the hat has been retained by selecting a fine mesh consisting of 81 points along the x-axis, but as soon as this is increased to 83 points, we see the artefacts that arise

from not being constrained by the CFL condition (figure 4) and these effect grow quickly with the addition of only a few more points to the spatial grid as seen in figure 5.

Figure 4: 1-d linear convection only. CFL condition broken. Number of grid points nx=83

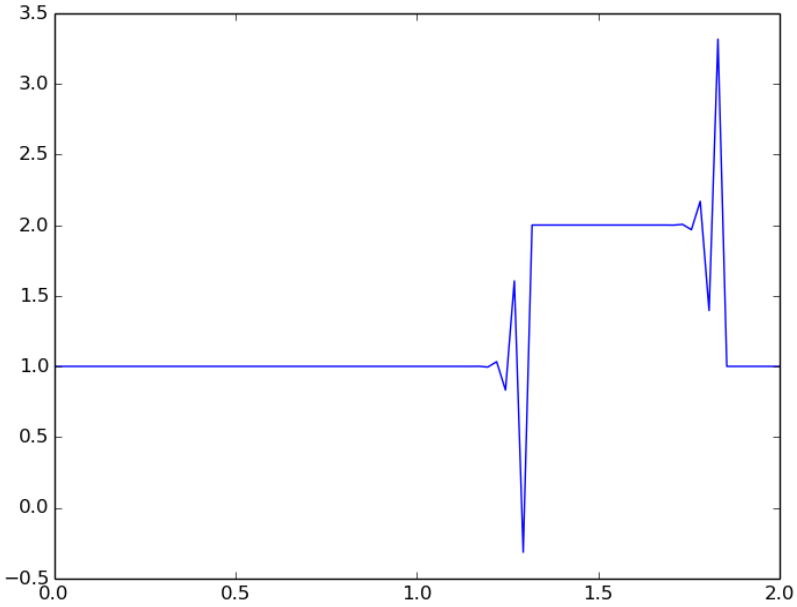
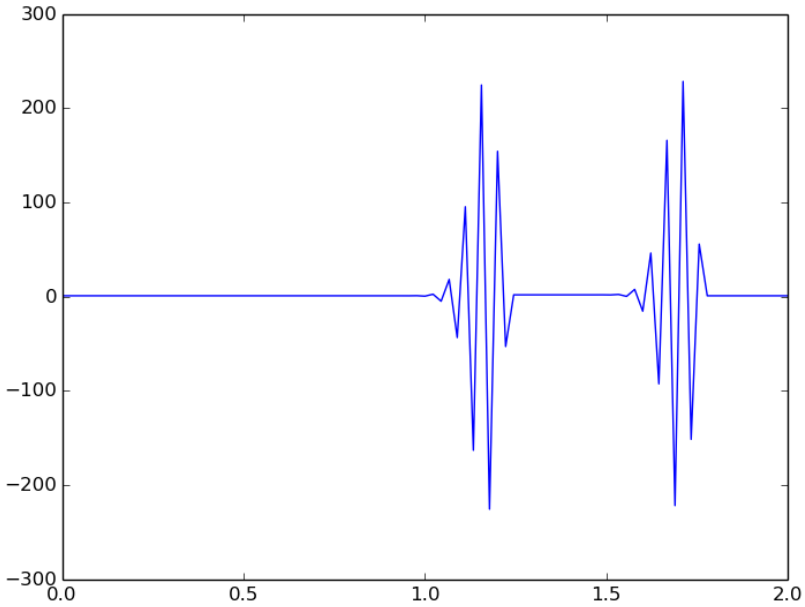


Figure 5: 1-d linear convection only. Numerical artefacts dominate. nx=91



4.2.2 Diffusion

Moving into two dimensions. Here are some plots showing true physical diffusion. The initial condition is a 2-d hat function shown in the first figure (6), an intermediate step (figure 7) is shown before it ends up at figure 8 after 50 timesteps. Large spatial grid (31 by 31) was chosen for visualization purposes. Script used: `diffusion_2d.py`

Figure 6: 2-d diffusion: Initial condition

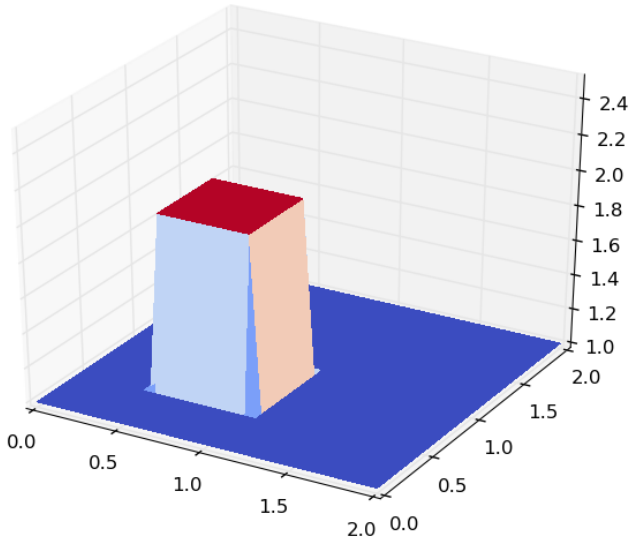


Figure 7: 2-d diffusion: Intermediate step (10 steps later)

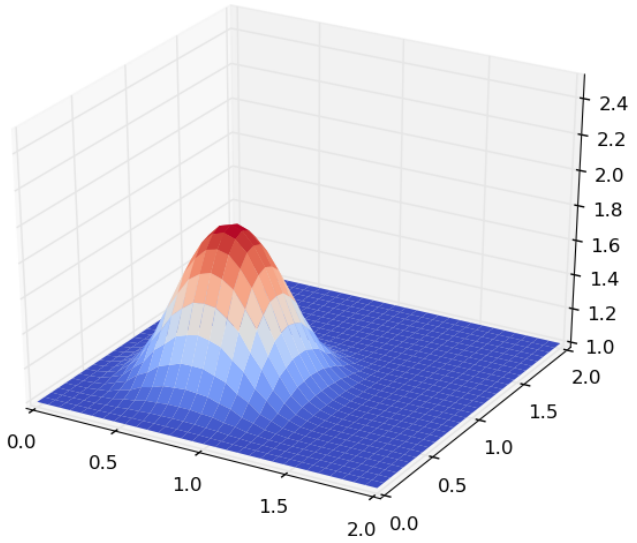
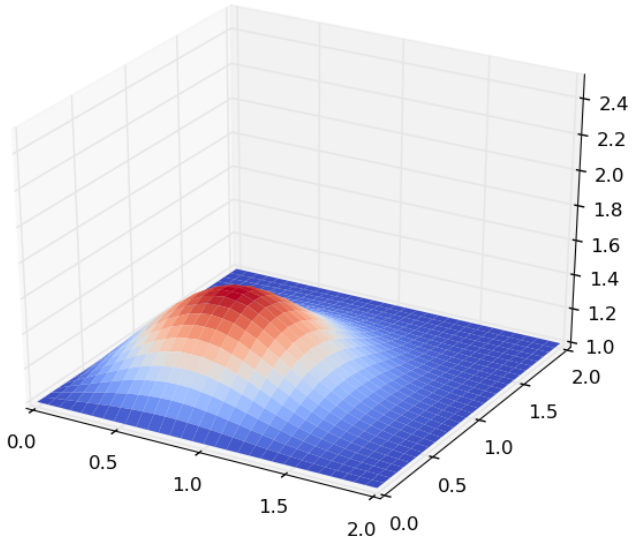


Figure 8: 2-d diffusion: Final step (50 steps after initial)



4.2.3 Burgers

Combining nonlinear convection and diffusion in 2-dimensional space, this is an example of a flat colourmap plot of Burgers equation acting on a initial condition of a hat function in figure 9 to become figure 10 after 155 timesteps. See code `burger_driver`for velocity field inital values and other parameters.

Figure 9: Initial condition: 2-d hat on flat colourmap

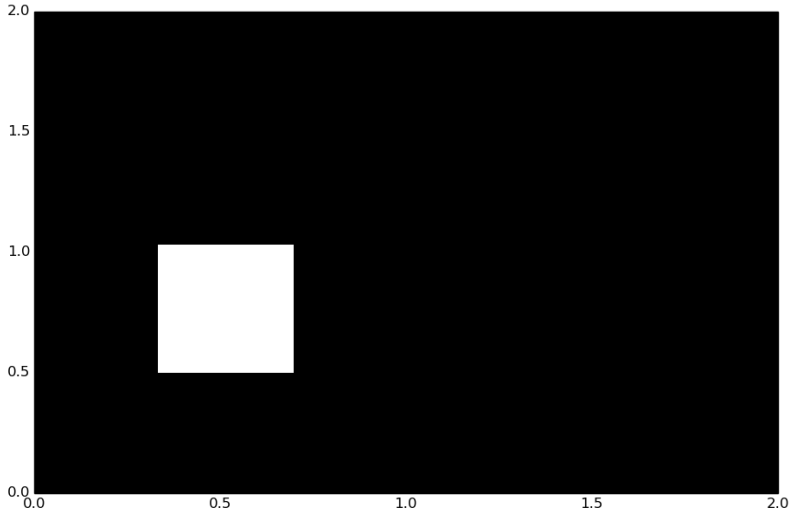
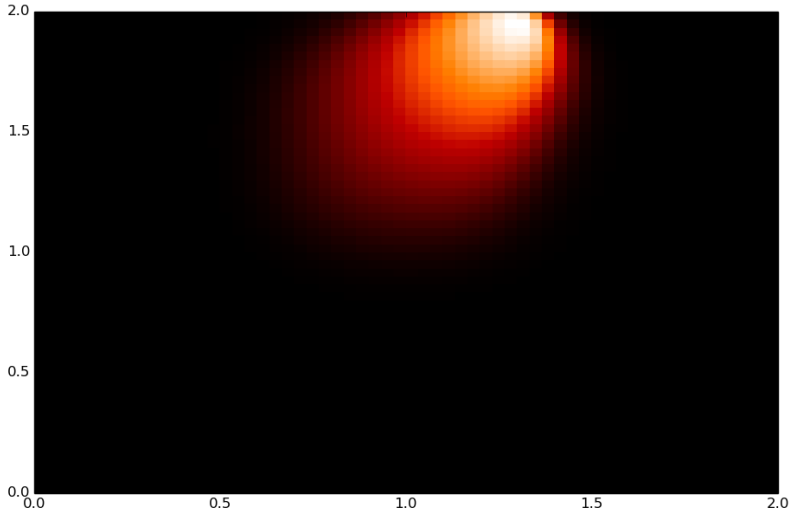


Figure 10: Burgers equation: after 155 timesteps



4.2.4 Poisson

Figures 11 and 12 show the effect of Poisson's equation in relaxing the initial conditions of a source and a sink. The code is directly adapted from the "12 Steps" in `13_poisson.py`. See the change between 10 steps for relaxation and 5000 steps.

Figure 11: Poisson equation on a spike and a sink: 10 iterations

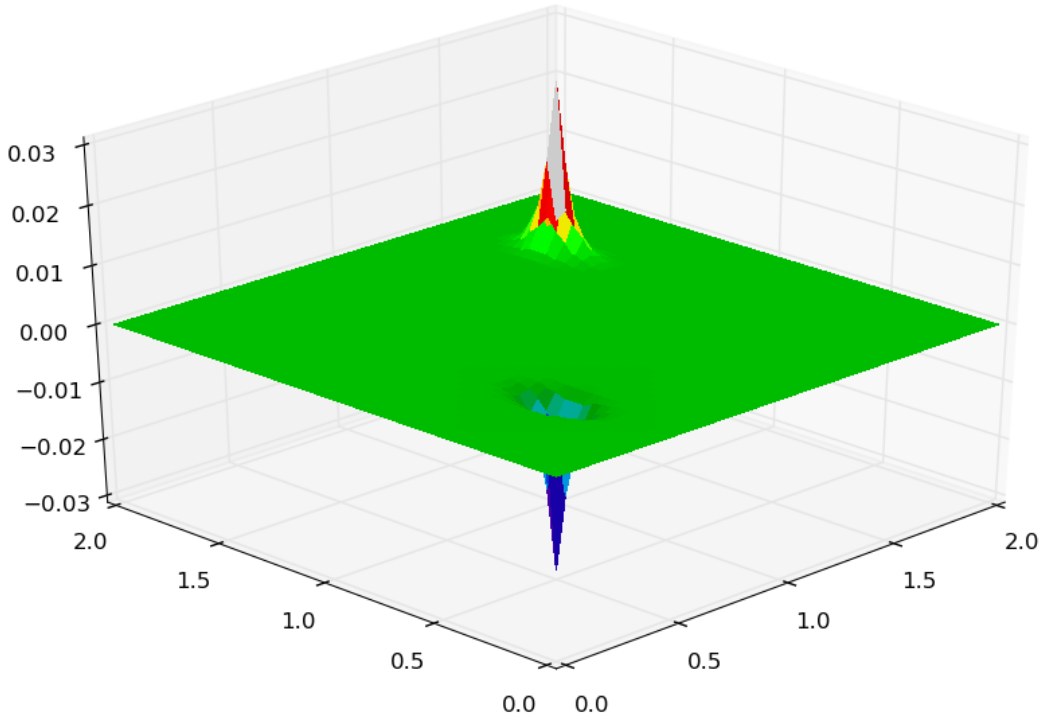
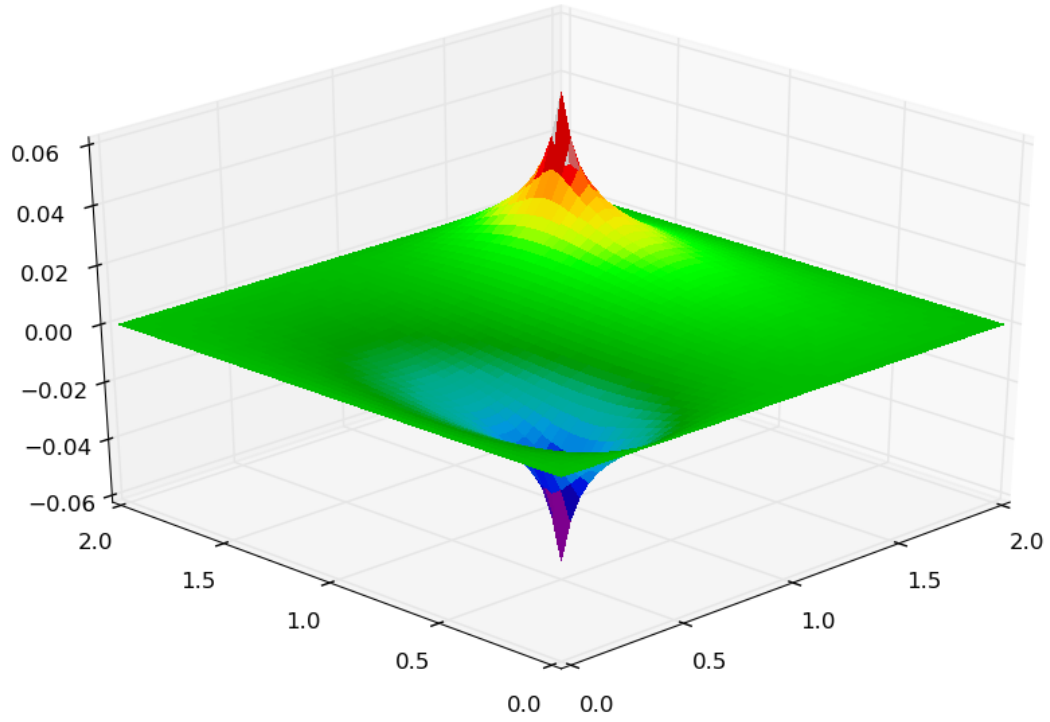


Figure 12: Poisson equation on a spike and a sink: 5000 iterations



4.2.5 Navier-Stokes cavity flow

The next two scripts `ns_cavity.py` and `ns_channel.py` are directly from the "12 Steps" notes with minor modifications to work with current libraries and some adjustments made to parameters for visualization. Otherwise it follows directly from the numerical schemes but with auxiliary variables to hold temporary values instead of having to calculate it in long expressions that are prone to typing transcription errors.

Cavity flow has a the external velocity on the top boundary (figure 13). This boundary condition is reset at every iteration while the inside of the field changes with each iteration (until steady state). Figure 13 shows the system near initial condition, far from steady state. Then 14 shows in interim point, and 15 and 16 show that we are near steady state.

The arrows indicate velocity. Colour represents the pressure.

Figure 13: Navier-Stokes cavity flow: shows top boundary condition velocity, 20 iterations

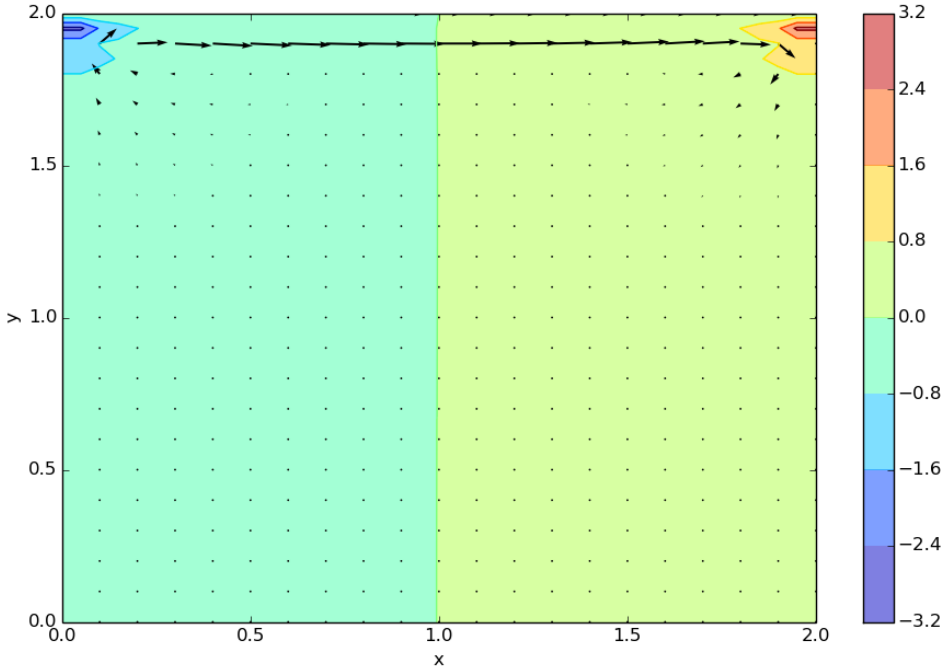


Figure 14: Navier-Stokes cavity flow: far from steady state, after 80 iterations

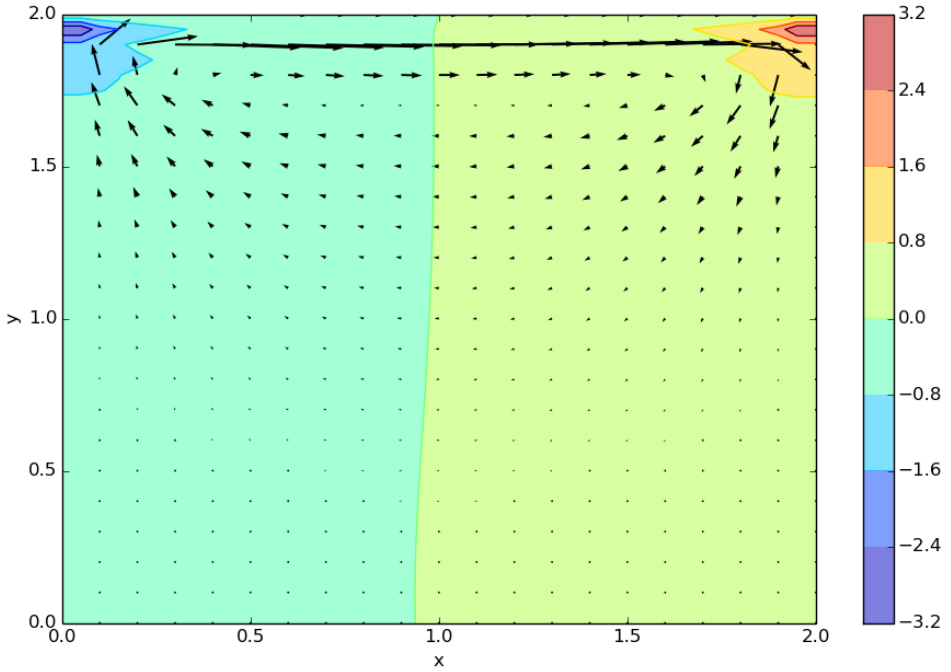


Figure 15: Navier-Stokes cavity flow: near steady state, 500 iterations

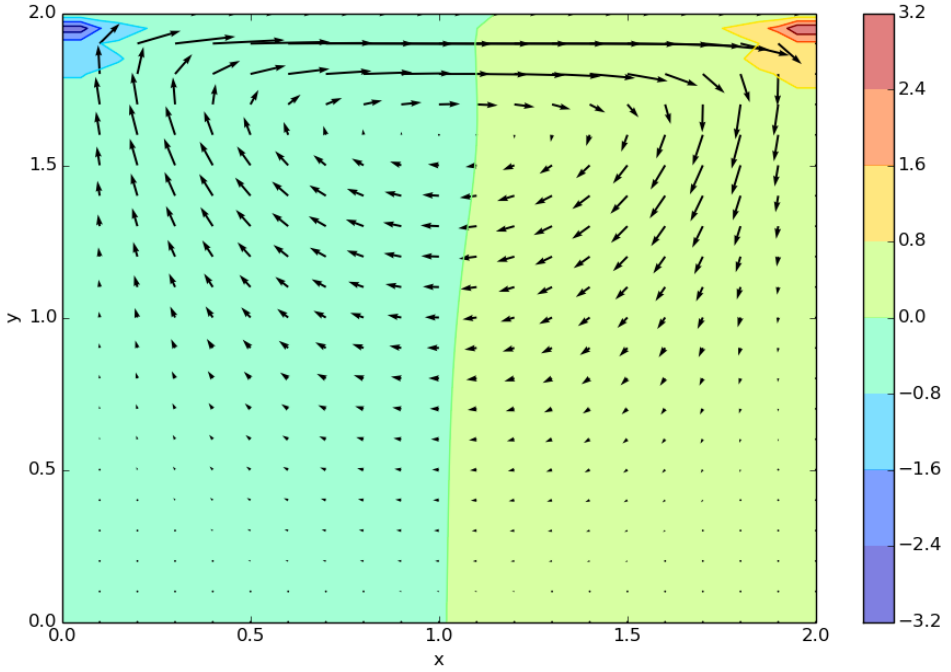
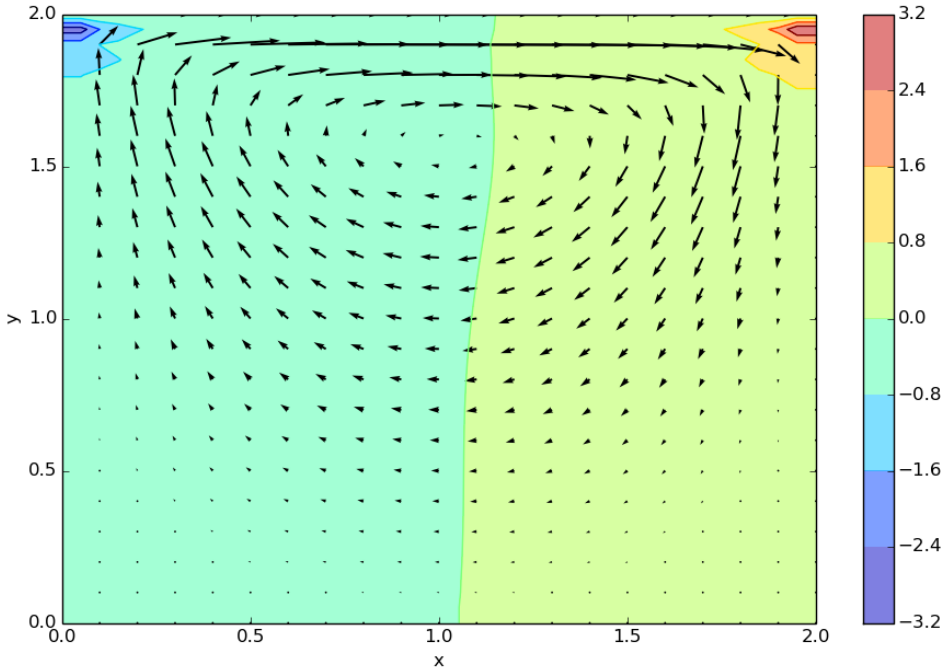


Figure 16: Navier-Stokes cavity flow: near steady state, 700 interations, compare to previous



4.2.6 Navier-Stokes: channel flow

Channel flow as the extra source term for persistent pressure gradient in one direction ($F = 1$). Walls are reset to zero at every iteration. Periodic boundary conditions were used such that the left and right walls wrap and this plot can represent flow seen in an infinitely long channel. (Another option would be to extend the x-axis very long and only inspect the middle section, but computationally, periodic conditions work well here.) The gradient of the velocity is such that it matches the zero velocity at the top and bottom boundaries while the middle of the channel could accelerate until it reached the steady state value. The code is set up to terminate once it reaches a tolerance of less than 0.001 difference in results between subsequent iterations. This final state is shown in Figure 17 but if we take a look before it has reached a reasonable steady state, with larger differences between iterations, we would have something like Figure 18 where the middle of the channel has not had the chance to come up to speed yet.

Figure 17: Navier-Stokes channel flow: near steady state, tolerance = 0.001

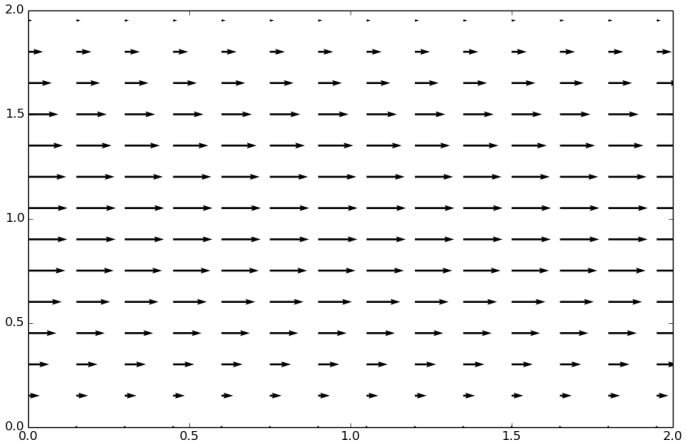
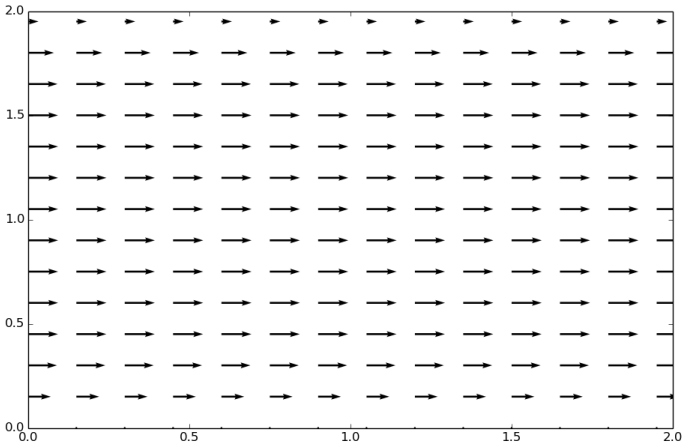


Figure 18: Navier-Stokes channel flow: far from steady state, tolerance = 0.05



Further additions desired

Extensions to the code would make future computations and visualizations run smoother: writing solutions to disk (small change) and functionality to read solutions from disk at a later time, better usability of animation functions –aim for ease of usage when changing platforms, ability to sample every k-th timestep of the solutions to quickly change granularity of animation (partially complete), saving the animations once complete (at debug stage), debugging the just-in-time compilation library so that the code could run faster for larger data sets, and potentially other changes that could optimize the runtime at the computational level.

At the numerical level, some improvements might be gained by changing the discretization scheme. For example, it might be acceptable to use implicit methods in some cases, and thereby improve the stability of the problem. I suspect there may be some steps where programming language optimizations can be used with regards to sparse matrix computations, for example in using Poisson's equation. If it were valid to apply these methods for the physics of the system, the numerical scheme combined with language features could increase the accuracy and the speed of the computations.

References

Barba, L.A. "CFD Python: 12 steps to Navier-Stokes." Accessed Jan-May 2016.

- http://lorenabarba.com/blog/cfd-python-12-steps-to-navier-stokes/
- https://github.com/barbagroup/CFDPython
- https://piazza.com/bu/spring2012/me702/home

Course notes, Phys 410. Fall 2015, UBC.

DeVries, P.L. *A first course in Computational Physics*, 2nd ed. 2011.

Coding references cited within the scripts at the code repository at https://github.com/cphl/cfd