The Driven Cavity

MATH1091: ODE methods for a reaction diffusion equation

http://people.sc.fsu.edu/~jburkardt/classes/math1091_2020/cavity/cavity.pdf

The driven cavity is a model of forced rotating flow, somewhat similar to what happens in a washing machine.

The Driven Cavity

We will use the stream function and vorticity formulation that we saw in the previous lecture, and try to put together a model of a fluid that is forced to rotate because of a force applied on one side.

1 A more realistic rotating flow

In the previous set of notes, we saw that we could use the function `uv_vortex()` to define the rotating velocity field of an incompressible fluid inside a square region. However, this example is of limited realism. We don't have any idea *why* the fluid is rotating; moreover, although the velocity pattern is "legal", it is not one that would be followed by a typical fluid under rotation.

For people who study fluid mechanics, one of the most common idealized flows is known as the *driven cavity*. In this case, we suppose that a fluid in a square box is initially at rest. Then, a constant sliding force is applied along one side of the box. You might imagine some sort of rubber belt is racing along that side. The effect of this force is to disturb the fluid, and to gradually make it rotate.

Our discussion will begin with the vortex flow from last time, to which we will add a blob of hot water. We will assume at first that the vortex velocities never change, and so we will concentrate on how the advection equation spreads the temperature around the region.

In order to look at a more realistic model, in which the velocities can change with time, we will try to set up a stream function and vorticity program for the driven cavity. Once we have that running, we will again insert a blob of hot water and watch, hoping to see it swirl araound as before.

2 Functions for rotating flow in a square

Last time, we used the function `uv_vortex.m` to define a rotating flow. However, this is not suitable for our driven cavity problem, for which the flow must initially be exactly zero along the walls. However, it is possible to come up with an incompressible flow field that will be zero on the walls, and will rotate inside. For that purpose, the web page provides the functions:

- `[ X, Y ] = xy_square(nr,nc);`
- `[ U, V ] = uv_square(X,Y);`
- `PSI = psi_square(X,Y);`

3 Exercise #1: Two ways to compute a velocity field

Copy the file *skeleton1.m*, which is an outline of a code that computes the velocity field `[U1,V1]` in the square directly, using `uv_square()` and then computes a second version `[U2,V2` indirectly, by computing the stream function, and then deriving the velocities using `uv_from_psi()`.

These two velocity fields will be similar, but not identical. That's because `uv_square()` is evaluating an exact formula for the continuous velocity field, whereas `uv_from_psi()` has to use finite differences to estimate the velocity from a table of stream function values.

Interestingly, this means that `[U1,V1]` will be samples of a flow that has zero divergence, assuming we can use derivatives. On the other hand, `[U2,V2]` will have zero divergence, assuming tha we are using finite differences. We must always keep in mind the fact that computationally, we are working with a discrete data, and that we are only approximating continuous quantities.

Fill in the missing portions of the code. In particular, the last portion of the code computes $||U1 - U2|| + ||V1 - V2||$, whose value would be tiny if the two approaches were equivalent. Because of discretization error, we will see that the norm of the difference is amall but noticeable.

4 Exercise #2: Examining the discrete divergence

In the previous discussion, we created a function `D=divergence(X,Y,U,V)`, which used finite differences to estimate the divergence of a velocity field. We know that for a mathematical, continuous velocity field of an incompressible fluid, the mathematical divergence must be exactly zero. We have seen that for the rotating flow, we have two versions of the velocity field. Now we want to use finite differences to approximate the divergence for each field.

We might guess that the finite difference estimate for the continuous velocity field `[U1,V1]` will have some error, but we might assume that the finite difference estimate for the finite difference velocity field `[U2,V2]` will have even more error, since it involves an approximation of an approximation. Alternatively, maybe since `[U2,V2]` was computed using finite differences, it has a better chance of zeroing out the finite difference version of the divergence.

In order to find out, copy *skeleton2.m* and fill in the details. This program computes and compares the divergences of these two fields for grids of order 11, 21, 41 and 81. Because we are comparing quantities at different spatial resolutions, it's important to use the RMS norm in order to see convergence behavior properly. If the divergence is not zero, that's a numerical error. Does that error go down as we double the grid number? Does doubling the grid number halve the error? Which version of the velocity field reports a smaller divergence?

5 Advection in a constant velocity field

The velocity field defined by `uv_square()` does not change over time. This means it's not the most realistic physical model, but we can still use it as a starting point for an experiment. Let's suppose that the water in the square is rotating according to the given velocity field, but that at the start of our test, there is a blob of hot water up in one corner. Physically, we would expect that hot water to gradually swirl around the region, and to get spread out.

Even with a fixed velocity field, we should be able to simulate this process by using a version of the advection equation in 2D. Assuming our variables are X, Y, U, V, T, the equation is:

$$\frac{\partial T}{\partial t} = -U\frac{\partial T}{\partial X} - V\frac{\partial T}{\partial Y}$$

For our computational model, we will update the "old" temperature field `TO` to the new values `TN` using the following discrete version of the equation:

$$TN = \frac{1}{4}(TO(N) + TO(S) + TO(E) + TO(W)) - dt * U(C)\frac{TO(E) - TO(W)}{2\,dx} - dt * V(C)\frac{TO(N) - TO(S)}{2\,dy}$$

The strange feature of using $\frac{1}{4}(TO(N) + TO(S) + TO(E) + TO(W))$ in place of $TO(C)$ is taken to improve the stability of the formula.

6 Exercise #3: Temperature advection in a constant velocity field

Copy the file *skeleton3.m*, which sets up the rotating flow field, and initializes the temperature to zero, except for a small blob of hot water. In the time loop, insert the necessary statements for the advection equation, as shown above. If you do this successfully, when you run the code you should see plots that indicate that the hot water blob is being carried around the region by the velocity field.

Professor Trenchea pointed out that we need to take lots of small time steps (1001 in this case), in order to avoid the stability problems that can arise when using an explicit time stepping method.

7 Time-dependent stream function and vorticity equations

From the previous notes, we found that, given the vorticity ω, we can compute the stream function by solving an equation that is similar to what we did for the 2D heat equation:

$$\nabla^2\psi(x,y) = -\omega(x,y)$$

For the driven cavity, we have zero boundary conditions for ψ on the three nonmoving walls, and along the wall that is moving with velocity *utop*, we set the condition

$$\psi = 2 * dx * utop$$

We can treat both variables as functions of time. In particular, the instantaneous time derivative of the vorticity can be written as:

$$\frac{\partial \omega}{\partial t} = -\frac{\partial \psi}{\partial x}\frac{\partial \omega}{\partial y} + \frac{\partial \psi}{\partial y}\frac{\partial \omega}{\partial x} + \nu\frac{\partial^2 \omega}{\partial x^2} + \nu\frac{\partial^2 \omega}{\partial y^2}$$

where ν is the fluid viscosity. For the driven cavity, we again assume zero boundary conditions on ω on the three nonmoving walls, while along the wall that is moving we set an appropriate condition.

In order to model the flow over time we

1. update ψ, an initial condition (step 1), or solving $\nabla^2\psi = -\omega$;
2. update ω, an initial condition (step 1), or a time step;
3. compute velocity using `uv_from_psi()`;

8 Exercise #4: Stream function/vorticity solver for driven cavity

Copy the file *skeleton4.m*, which almost solves the driven cavity problem, using the stream function and vorticity formulation. The part that is missing is the update of the vorticity. In the vorticity differential equation, replace each derivative by the corresponding finite difference as follows;

1. $\frac{\partial\omega}{\partial t}$: use the usual forward difference;
2. $\frac{\partial\psi}{\partial x}, \frac{\partial\omega}{\partial y}, \frac{\partial\psi}{\partial y}, \frac{\partial\omega}{\partial x}$: use a (first order) centered difference;
3. $\frac{\partial^2\omega}{\partial x^2}, \frac{\partial^2\omega}{\partial y^2}$: use a (second order) centered difference

The code assumes that the new vorticity is `ON`, the new stream function is `PN`, and the old values are in `OO`. Also, I have defined the indices `C, N, S, E, W` for you, so that you can write the update equations in a simple form. In particular, the equation might begin this way:

```
    ON(C) = OO(C) + dt * ( ...
      - ( 1.0 / dx / dy / 4.0 ) * ( SN(E) - SN(W) ) .* ( OO(N) - OO(S) ) ...
```

Complete this equation, and run the code. If things go well, you should see that the velocity plots indicate the beginning of rotational flow.

9 Homework #9: Temperature advection in the driven cavity

Copy the file *skeleton5.m*, which is an almost complete program for the driven cavity problem, with the addition of a temperature field that is to be transported using the same advection equation that you used in exercise #3 to compute the updated temperature `TN` from the previous values in `TO`.

Like exercise 3, this problem includes a "blob" of hot water that should be moved by the flow. Unlike exercise 3, the velocity field is computed at each time step, and varies. The flow in the driven cavity problem is slower, so we won't see the blob move quite as fast or as far. Nonetheless, this example should suggest to you how a complicated system can be modeled by repeated solving equations for each of the physical variables.

The code as written should display the solution at each time step, and will save the last plot in the file *hw9.png*.

Send me the plot *hw9.png* at **jvb25@pitt.edu**. I would like to see your work by Friday, 03 July 2020.