

Do the Local Motion, Part I: Shape Up! (Atmo 336, Fall 2007)

Getting Started. In class we talked about how the local structure of the velocity field can lead to changes in both size and shape for a small fluid particle. But of course one can only do so much by drawing sketches on a whiteboard. So for the next two labs we go Hollywood (or maybe Disney?) and actually put the particles into motion. We'll consider three particle types in particular: a square (as in class), a circle and the great state of Texas.

To animate the particles we'll need a couple of ingredients. First we'll need an initial shape, which we define by drawing an imaginary curve—called the *bounding curve*—around the particle's border. The second ingredient is of course a velocity field. Once we have the initial shape and the velocity field, we then ask: how does the particle's bounding curve change in time as the velocity field moves it around?

For the moment you'll only need to worry about the first two ingredients—namely, the initial particle shape and the velocity field. The code for moving the bounding curve around will then be provided as a script.

Setting Things Up. Ok, first to set up the particle shape. As mentioned above, ideally we'd define the shape by drawing a continuous curve around the particle border. But unfortunately there's really no way to draw a continuous curve on a computer. So instead what we'll do is approximate the curve's shape by sampling a finite number of points along its length (which we can then store on the computer). We'll then track the motion of these points and use their time-dependent positions to approximate the motion of the curve.

Copy the scripts. To get started, first copy the script files *particle.m*, *velocity.m* and *flowmovie.m* from the shared ATMO336 directory (under the subdirectory *lab_3_scripts*). The first two scripts contain the particle definitions and velocity definition—these are the things you'll actually have to modify. The other script has the code for moving the bounding curve around and need not be changed.

Take a look at the *particle.m* script. The script contains definitions for three particle shapes: a square, a circle and the great state of Texas. To pick a shape for your movies you then just have to set the *shapeflag* variable at the top of the script. But unfortunately the definitions for the square and the circle are incomplete. (Bummer!) Our first job is thus to finish these two definitions.

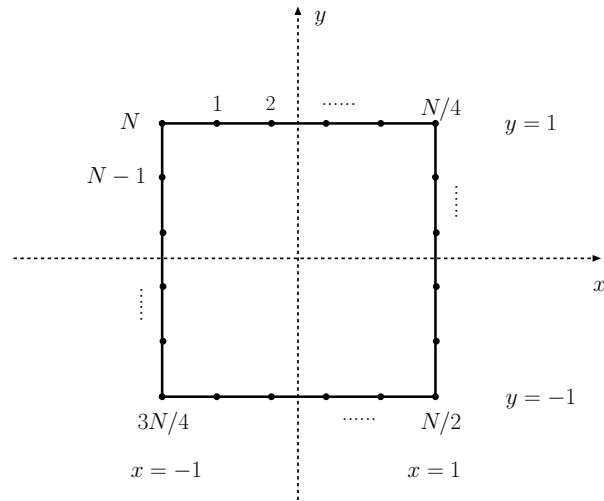
The square particle. Consider a square of side length 2 with the center at the origin [so that the corners are at (-1,1), (1,1), (1,-1), and (-1,-1)]. To put the square into the code we'll sample a set of N points along its border (as illustrated below). The position of the k^{th} point is then stored in the variables $x(k)$ and $y(k)$ in the script. So that our set of N points includes the corners, we'll need to choose N so that it is divisible by 4. The first $N/4$ points then go along the top edge of the square, the next $N/4$ points along the right side, and so on.

In the *particle.m* script you'll find the following incomplete definition for the square. Fill in the missing pieces:

```

N = 100;
ds = 8./N;
for k=1:N/4
    x(k) = -1. + k*ds;
    y(k) = 1.;
end
for k=(N/4)+1:N/2
    x(k) = 1.;
    y(k) = 1. - (k-N/4)*ds;
end
for k=(N/2)+1:3*N/4
    x(k) = ;
    y(k) = ;
end
for k=(3*N/4)+1:N
    x(k) = ;
    y(k) = ;
end

```



The circular particle. Ok, now we'll do the same thing for the circle. First to define the circle in words: we'll consider a circle centered at the origin with a radius of 1. But to put the circle into the computer we'll again need to sample a set of N points along its length (as illustrated below). And to make sure that we initially have points on the x and y axis we'll again want to make sure that N is divisible by 4.

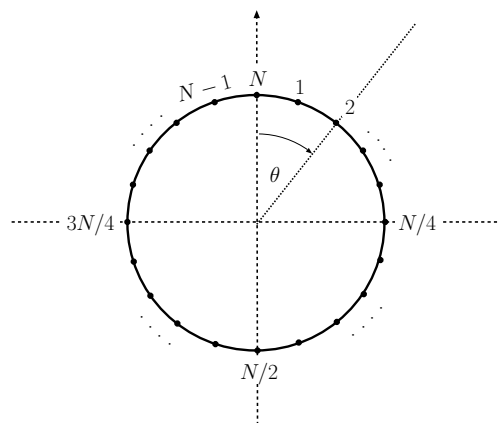
When dealing with circles it makes sense to define the positions using polar coordinates. Suppose we let r be the radius of the circle (here fixed at 1) and we let θ be the angle along the circle as measured from the y axis (as illustrated below). How do we convert from r and θ to the Cartesian coordinates x and y ?

Once we have the conversion between polar and Cartesian coordinates we can then add our circle to our script as follows:

```

N = 100;
dthet = 2*pi/N;
for k=1:N
    thet = k*dthet;
    r = 1;
    x(k) = ;
    y(k) = ;
end

```



The great state of Texas. Our third shape option for the particle is the outline of the great state of Texas. This one is a bit hard to define mathematically, so we went ahead and hard-coded the positions for you. Nothing to modify here.

The velocity field. The final needed ingredient is a velocity field. The velocity for the lab is defined in the separate function file *velocity.m*. The function takes two arguments— x and y as defined above—and returns the corresponding velocity components u and v . For the moment you can just set these to

$$\begin{aligned}u &= x; \\v &= 0.*y;\end{aligned}$$

since this is what you'll use for your test calculations. But of course you'll need to change these definitions as you progress through the lab. (*Note:* To set u or v to zero use something like $u = 0.*x$ or $v = 0.*y$ rather than $u = 0.$ or $v = 0.$ This makes sure the function works with matrices in addition to scalars.)

Test Your Particle Shapes. After you set up the initial shape and velocity field, you can see the fruits of your labor by running *flowmovie* on the MATLAB command line. Let's take it for a test drive, starting with the square.

Testing the square. Select the square by setting the *shapeflag* variable at the top of *particle.m* to zero. Then issue the *flowmovie* command. Assuming everything is ok, *flowmovie* will start by showing you the initial state. You should then ask yourself (no need to write down the answers):

- Is the initial particle shape in fact a square?
- Are there N points shown along the particle border ($N/4$ along each side)?
- Does my set of points include the corners of the square?
- Does my velocity field look like $(u, v) = (x, 0)$?

If the answer to any of these questions is no, then you'll need to revisit the shape and/or velocity definitions in *particle.m* and *velocity.m*.

After showing the initial state, the *flowmovie* command will pause. Just hit the space bar to continue bravely on and make a movie. You should see that the square is gradually stretched out along the x -axis as expected. To save the movie for later viewing, type

```
movie2avi(M,'test_square');
```

which creates the movie file *test_square.avi* in the current directory. Double click on the file and make sure that the movie is ok.

Testing the circle. Change the *shapeflag* variable in *particle.m* to 1 and repeat the steps described above for the circle. When reviewing the initial state ask yourself

- Is the initial particle shape in fact a circle?
- Are there N points shown along the particle border ($N/4$ along each quarter circle)?

If the answer to either of these questions is no, go back and correct the shape definition in *particle.m*.

Create a movie as described above and save the movie to a file called *test_circle.avi*. Double click on the movie file and make sure that the movie runs ok.

Testing the great state of Texas. Change the *shapeflag* variable in *particle.m* to 2 and repeat the steps described above for the great state of Texas. Save your movie to a file called *test_texas.avi*.

Turn 'em in. Copy your three movies to the figure directory to turn 'em in.

Fun With Simple Stretching and Shearing. Pick your favorite particle shape—doesn't matter which—and make a movie for each of the following flow fields:

- $(u, v) = (0, y)$ (save movie to *stretch_y.avi*)
- $(u, v) = (y, 0)$ (save movie to *shear_y.avi*)
- $(u, v) = (0, x)$ (save movie to *shear_x.avi*)
- $(u, v) = (x + y, x + y)$ (save movie to *what_the.avi*)

Copy your movies to the figure directory to turn in. And guess what....you're done! At least for now.....