Unsteady Flow Around an Infinite Cylinder

Aero 302: Fluid Dynamics

Zach Selzman

Over time, I have grown to enjoy working with `waterlily.jl`. This is an incompressible viscous flow solver developed by some people over at TU Delft[1]. The code runs in pure julia, which is nice for speed and simplicity. Instead of a mesh, I can use parametrically defined bodies, which dramatically reduces computation time. Ansys Fluent is likely more accurate, but my results seemed pretty good for a lightweight local solver, which is built off of the implicit LES technique (which should be more than strong enough for a simple unsteady cylinder case). The really nice thing is that `waterlily.jl` makes it really easy to iterate Re values to look at some shedding behavior.

The aim of this project is to observe how unsteady flow over an infinite aspect ratio cylinder varies with changes in Reynolds number. I analyzed the following range:

$$Re \in \{1.2 \times 10^2, 5 \times 10^2, 1 \times 10^3, 1 \times 10^4, 2 \times 10^5\}$$

The values considered were intentionally left relatively laminar to supplement a lab spent in a low-reynolds water tunnel. The simulations do have some drawbacks compared to the experiments conducted, which are discussed in depth below. The results are compared using the Strouhal number and with literature at the end of this short report.

$Re = 1.2 \times 10^2$:

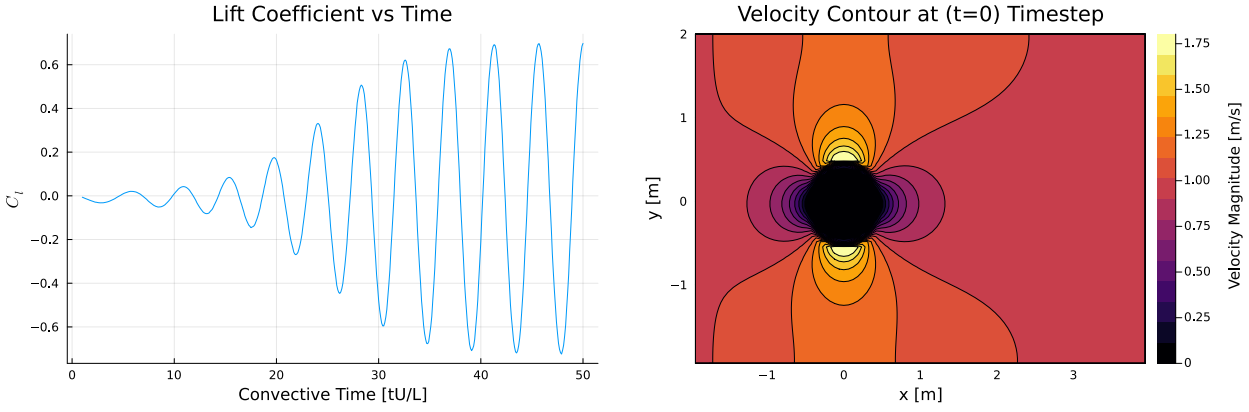
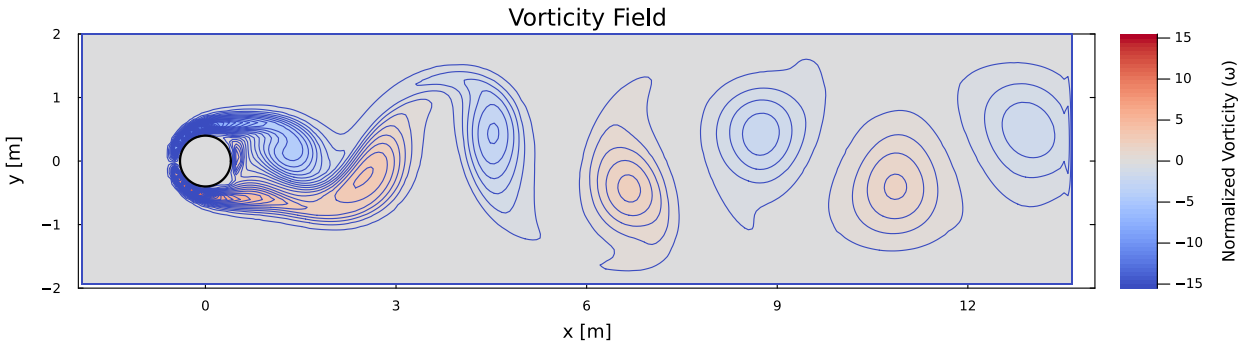


Figure 1: $Re = 120$

[1]The repository link is here: https://github.com/WaterLily-jl/WaterLily.jl



$Re = 5 \times 10^2$:

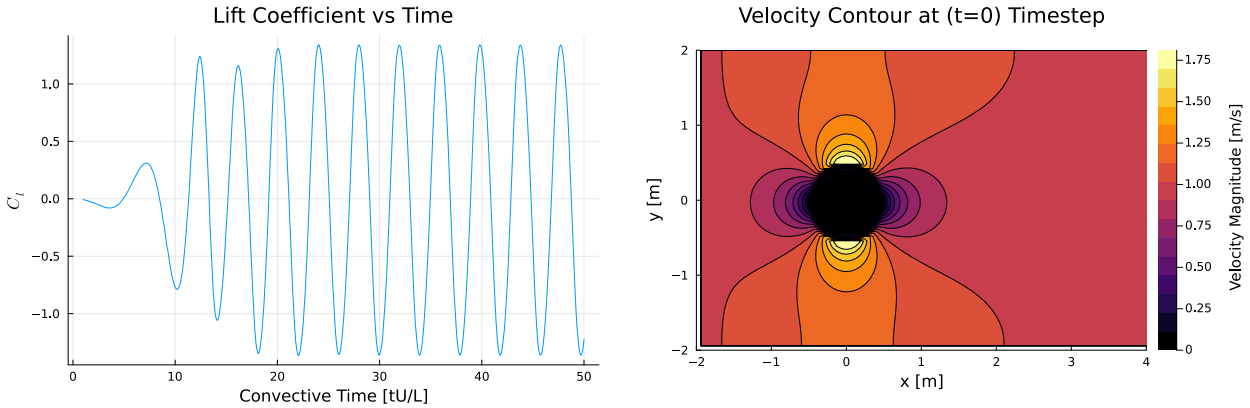
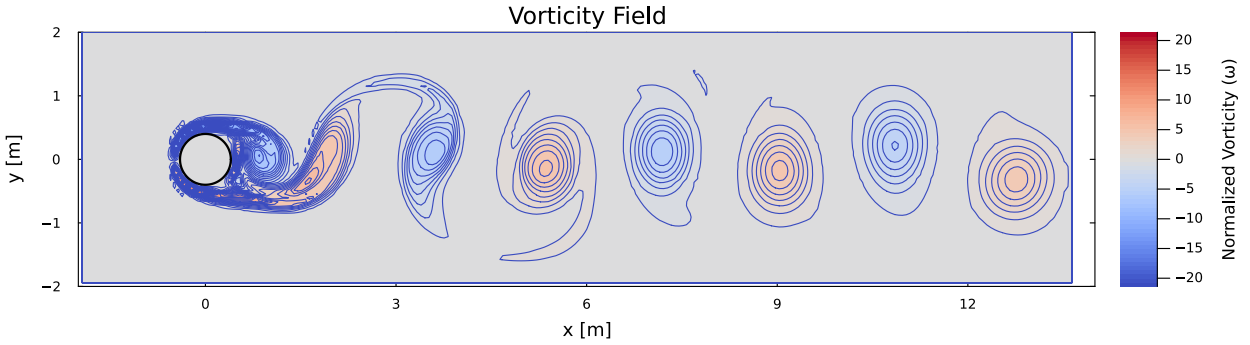


Figure 3: $Re = 500$



$Re = 1 \times 10^3$:

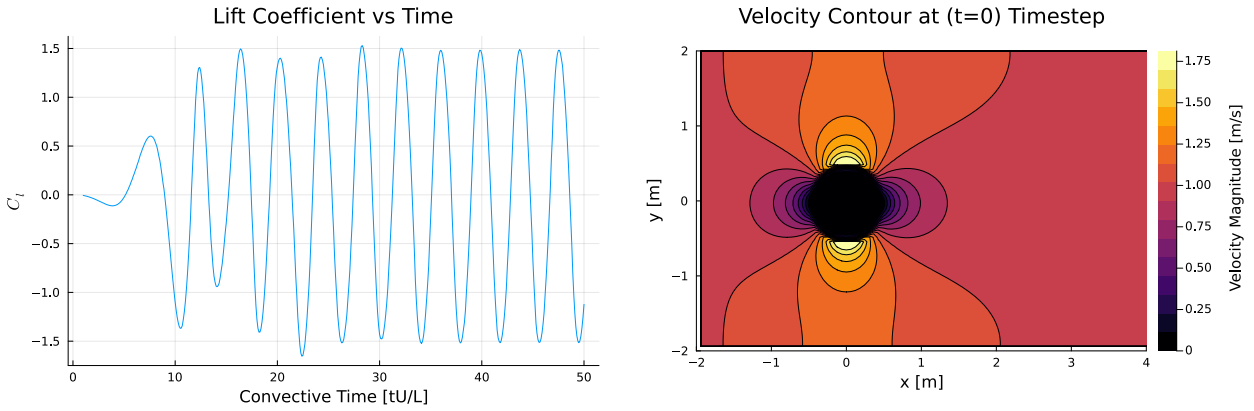
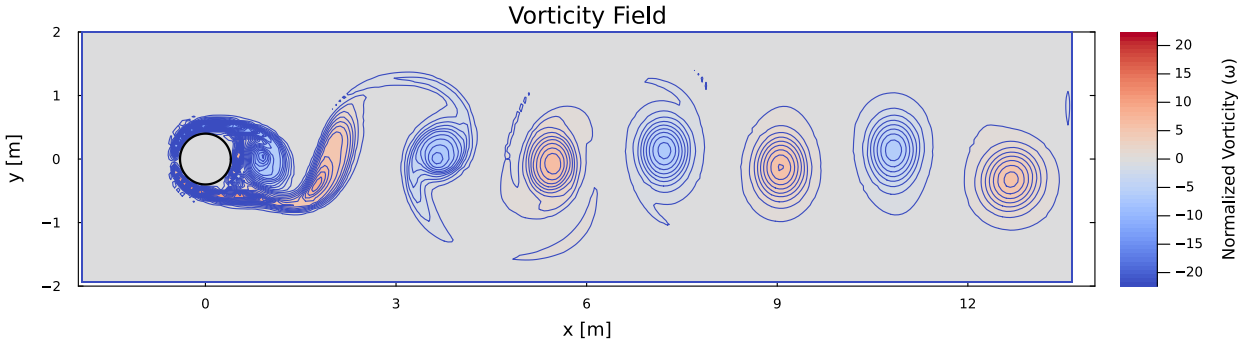


Figure 5: $Re = 1 \times 10^3$



$\boldsymbol{Re = 1 \times 10^4}$

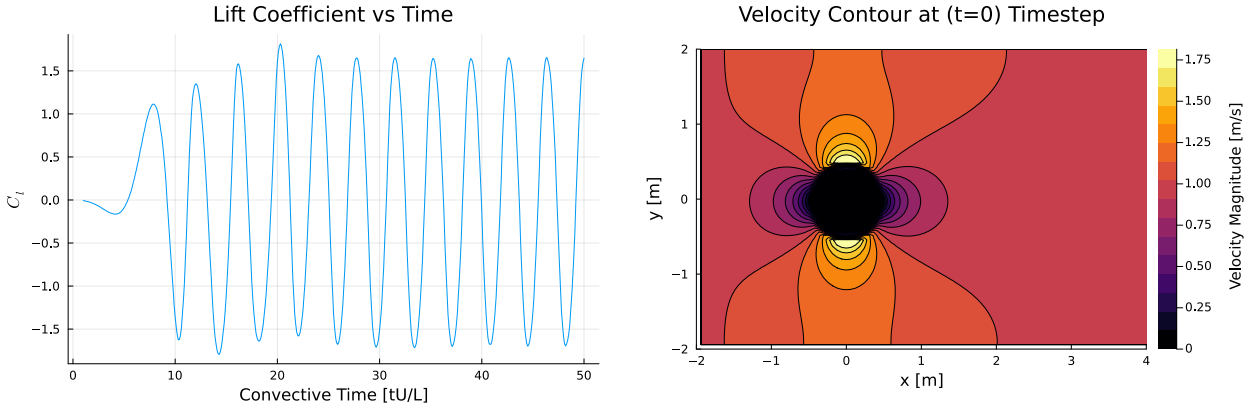
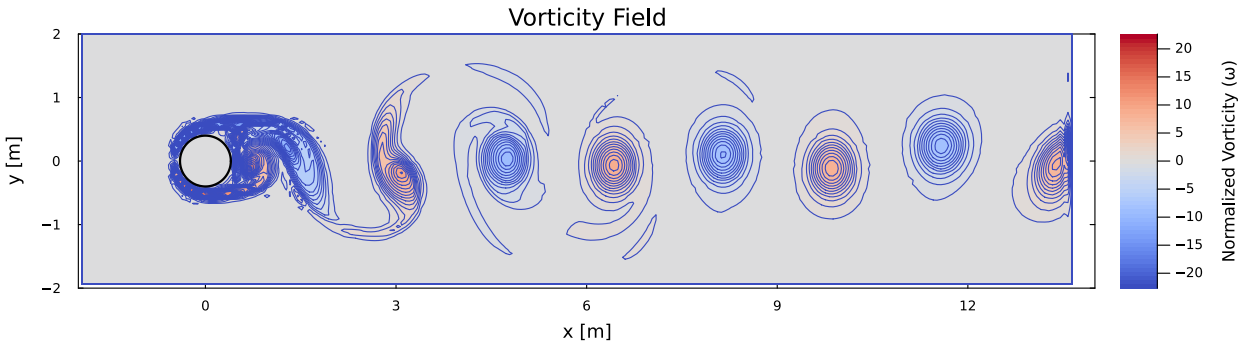


Figure 7: $Re = 1 \times 10^4$



$\boldsymbol{Re = 2 \times 10^5}$

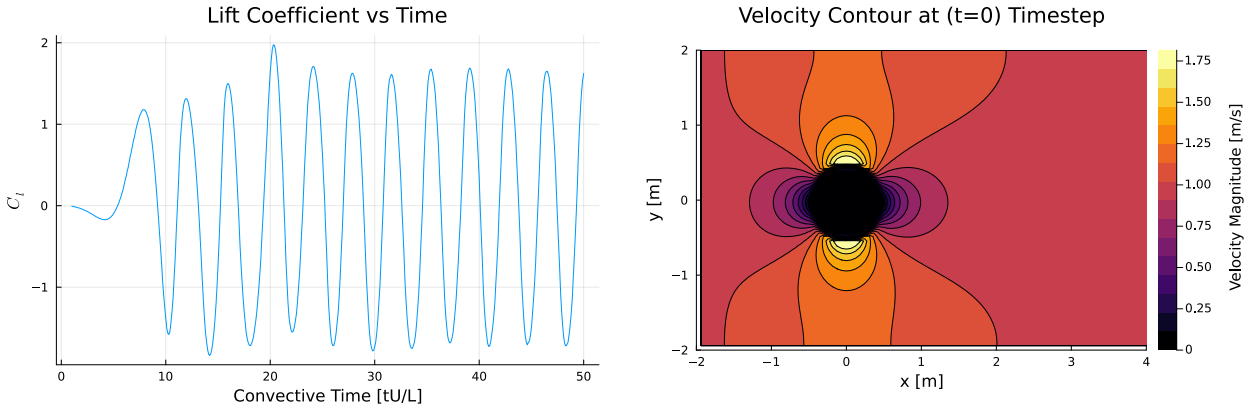
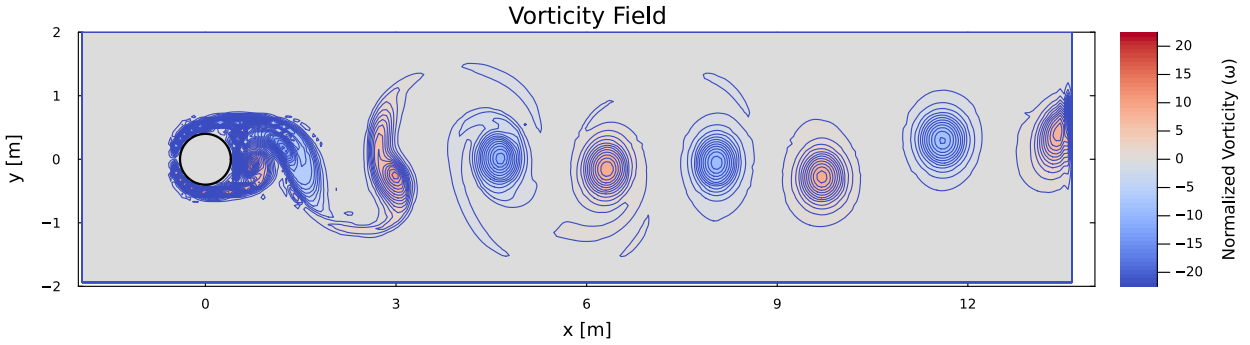


Figure 9: $Re = 2 \times 10^5$



It is worth noting that here, I chose to plot in convective time, which is nondimensionalized. To find Strouhal's number is quite simple:

$$St = \frac{fL}{U} = \frac{1}{t}\left(\frac{L}{U}\right)$$

where Convective time is given by:

$$t^* = \frac{tU}{L} \Longrightarrow t = \frac{t^* L}{U}$$

Substituting yields:

$$St = \frac{1}{\left(\frac{t^* \cancel{L}}{\cancel{U}}\right)} \left(\frac{\cancel{L}}{\cancel{U}}\right) \implies St = \frac{1}{t^*}$$

so I have to simply take the inverse of the distance on the x axis between peaks of each sinusoid to get Strouhal's number. The behavior of the flow in the higher Re flow region appears to be extremely similar to each other. This may be different in an alternative solver such as Ansys Fluent, however in `waterlily.jl`, the solver struggles to capture the turbulent effects that occur span-wise on the cylinder. This was heavily discussed in the water tunnel lab, in which we noticed there was vorticity about the $\hat{\imath}$ and $\hat{\jmath}$ directions. Because the simulation was in two dimensions, it is difficult for it to capture these adverse effects at high Re numbers. Perhaps Fluent has a 3D flow correction for this? Anyways, Figure 11 shows the progression of Strouhal numbers as Reynolds numbers increases.

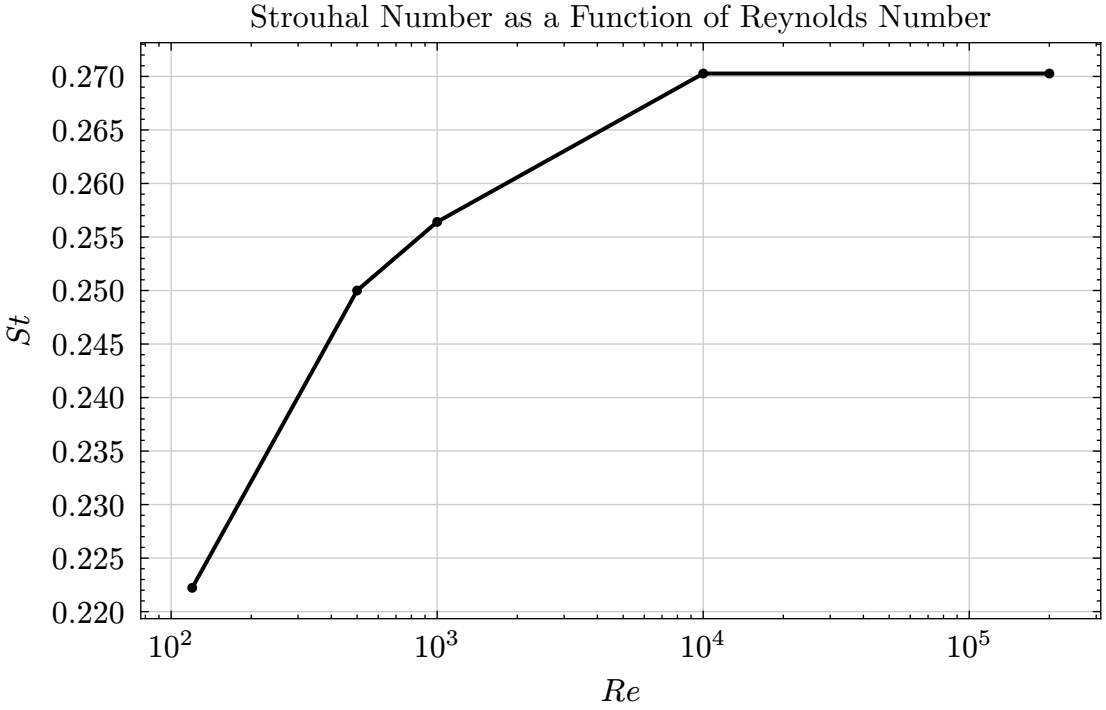
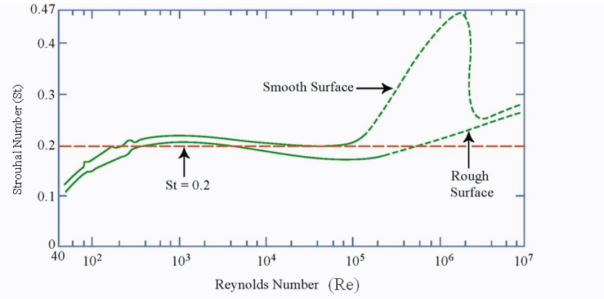


Figure 11: Strouhal and Reynolds number relation

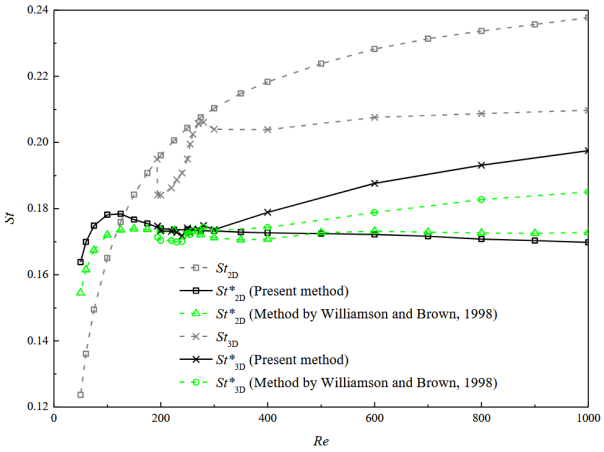
Generally, we can see that there is an increase in shedding as Re increases, however, this occurs up until a point, which makes sense, given my previous comments on 2D solvers. From literature[2], the results seem pretty good, although most studies either stopped analysis at $Re = 1 \times 10^3$ or went quite large[3]:

[2]Second Plot: https://www.researchgate.net/publication/265496181_Development_and_Validation_of_an_Aeroelastic_Ground_Wind_Loads_Analysis_Tool_for_Launch_Vehicles

[3]https://api.research-repository.uwa.edu.au/ws/files/88865981/Accepted_manuscript.pdf



(a) 2d experimental results



(b) 2d and 3d experimental results

Figure 12: Experiemental Data

Shown in Figure 12b, it is interesting to see the difference in 2D and 3D simulations, which capture turbulent effects outside of the $\hat{k}$ reference direction. The key thing is that St generally increases rapidly for increases in low Re. However, at high Re values, there appears to be more of a stagnation, as the flow looses energy to directions outside of the reference. From lab 4, the general trend was observed, with a marked increase in St as a function of a higher Re value. Moreover, we could easily witness the vorticity curling in the vertical direction. Thus, the general trend holds both for the labs and the simulations completed. We did not run high Re numbers in the water tunnel lab however, thus, the trend is only validated for the low speed, laminar flows.

1. Appendix A: `Waterlily.jl` Code

Julia

```julia
# Unsteady Cylinder code yo

using WaterLily
using Plots
function circle(n,m;Re=120,U=1) #note, for each iteration, I just had to change the Re value accordingly.
    # signed distance function to circle
    radius, center = m/8, m/2-1
    sdf(x,t) = √sum(abs2, x .- center) - radius

    Simulation((n,m), # domain, similar to a mesh in Ansys fluent
               (U,0),   #inlet condition
               2radius; #length (diameter)
               ν=U*2radius/Re,  #viscosity
               body=AutoBody(sdf)) # geometry
end
circ=circle(4*2^6, 2^6) # I define my n x m grid here, had to play with this one a bit
sim_step!(circ)

u = circ.flow.u[:,:,1] # first component is x

# the grid is in cell indices (n x m), I want physical coordinates
nx, ny =size(u)
D=1
center = ny/2 - 1
m=ny
dx=4*D/m
xcoord = range(0, step=dx, length=nx)
ycoord = range(0, step=dx, length=ny)
#shift to center
xcoord_shifted = xcoord .- center*dx
ycoord_shifted = ycoord .- center*dx # basically doing this to get the circle at 0,0

# plot velocity field
contourf(xcoord_shifted, ycoord_shifted, u',
aspect_ratio=1,
title="Velocity Contour at (t=0) Timestep",
xlabel="x [m]",
ylabel="y [m]",
xlim=(-2,4),
ylim=(-2,2),
 colorbar_title = "\n Velocity Magnitude [m/s]", right_margin=5Plots.mm)

```

```
#need to get cl oscillations to find Strouhal's number
 function get_forces!(sim,t)
    sim_step!(sim,t,remeasure=false)
    force = WaterLily.pressure_force(sim)
    force./(0.5*sim.L*sim.U^2) # normalize for coefficients
end

# Simulate through the time range and get forces
time = 1:0.1:50 # time scale is sim.L/sim.U
forces = [get_forces!(circ,t) for t in time];

#Plot it
plot(time,[last.(forces)], #spits out both lift and drag, we only care about
   lift here, drag should be a constant value, although I guess it would have
   small oscillations
    legend=false,
    title="Lift Coefficient vs Time",
    xlabel="Convective Time [tU/L]",
    ylabel="\$C_l\$")

#vorticity
ω = zeros(nx, ny)
@inside ω[I] = WaterLily.curl(3, I, circ.flow.u*circ.L/circ.U)

using Plots

p_vort = contourf(
    xcoord_shifted,
    ycoord_shifted,
    ω',
    xlabel = "x [m]",
    ylabel = "y [m]",
    title = "Vorticity Field",
    aspect_ratio = 1,
    colorbar_title = "Normalized Vorticity (ω)",
    right_margin = 1Plots.mm,
    bottom_margin = 5Plots.mm,
    left_margin= 5Plots.mm,
    size=(1100, 300),
    colorbar=true,
    legend=false,
    framestyle = :box,   # Ensures a frame boundary exists for the grid
    grid = false,
    c=:coolwarm,
    clims = (-maximum(abs.(ω)), maximum(abs.(ω))),
    levels=60,
```

```
87     xlim = (-2, 14),
88     ylim = (-2, 2),)
89 #making the plot look nicer, overlaying the cylinder for the amazing Aero
   302 graders:
90 θ     = range(0, 2π; length = 400)
91 Rphys = D/2.5
92 xcirc = Rphys * cos.(θ)
93 ycirc = Rphys * sin.(θ)
94 plot!(p_vort, xcirc, ycirc, lw = 2, color = :black)
```