Justine John "JJ" A. Serdoncillo
February 10, 2023

AEM 8202: Fluid Mechanics 2

Homework 3

Number 2

After applying the Boundary Element Method on the circular cylinder, the following plots are made for different values of n for the ψ_1 and $\frac{\partial \psi_1}{\partial n}$. It can be seen that even when changing the value of n, the trend of the lines does not change and looks sinusoidal in nature.

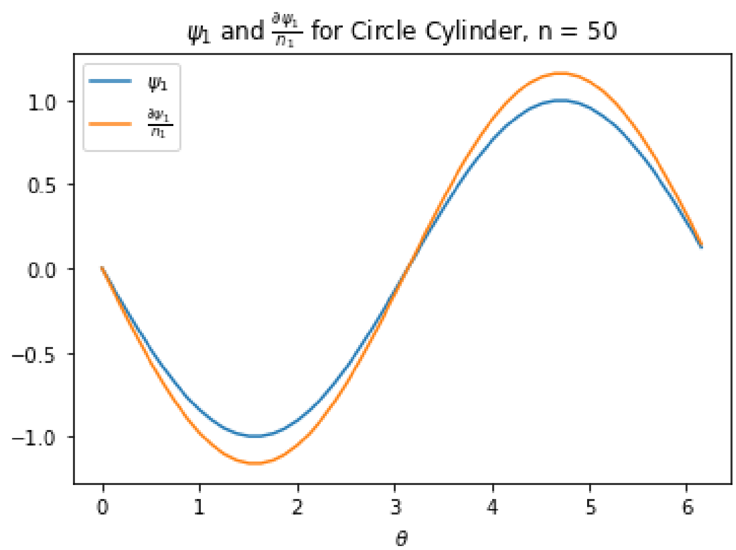

Figure 1: Plots of ψ_1 and $\frac{\partial \psi_1}{\partial n}$ for n=50

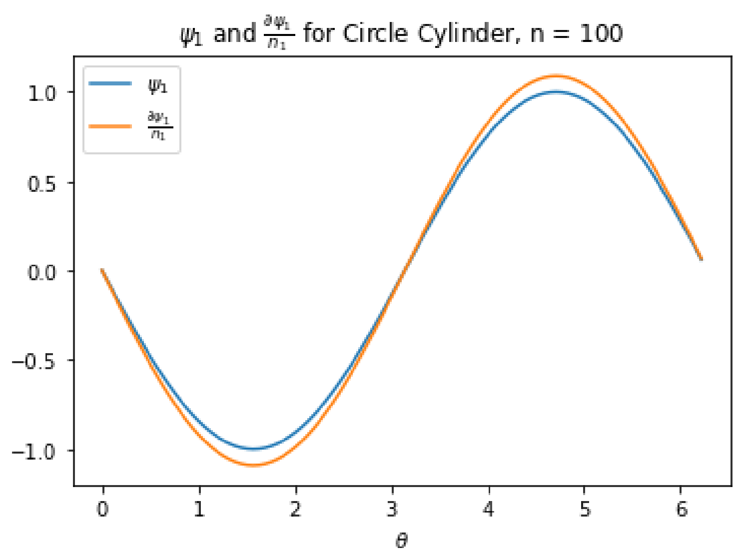

Figure 2: Plots of ψ_1 and $\frac{\partial \psi_1}{\partial n}$ for n=100

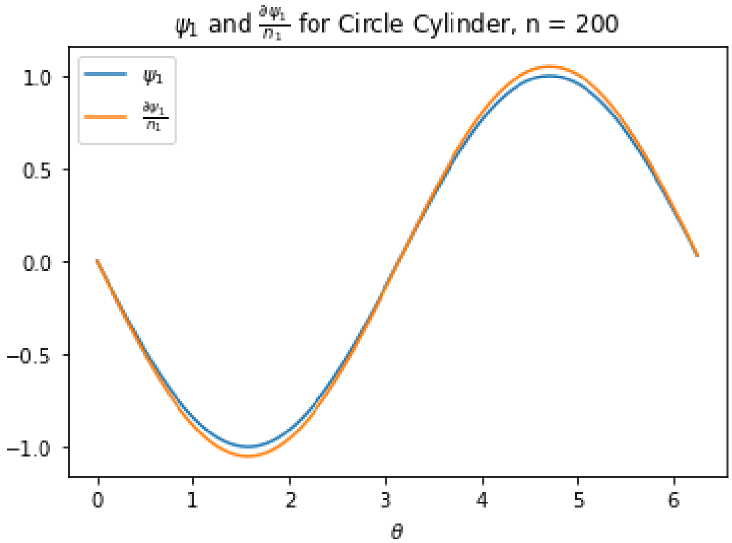

Figure 3: Plots of ψ_1 and $\frac{\partial \psi_1}{\partial n}$ for n=200

It can be seen that as n increase, the partial derivative has its peak getting closer to the 1 value.

Number 3

The following figures below shows the contours of the streamfunction. This corresponds to the streamlines of the flow around the cylinder.

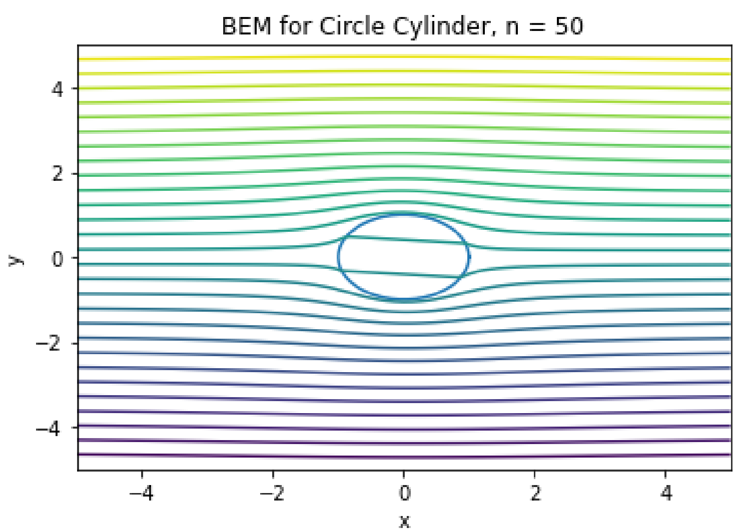

Figure 4: Streamlines for n=50

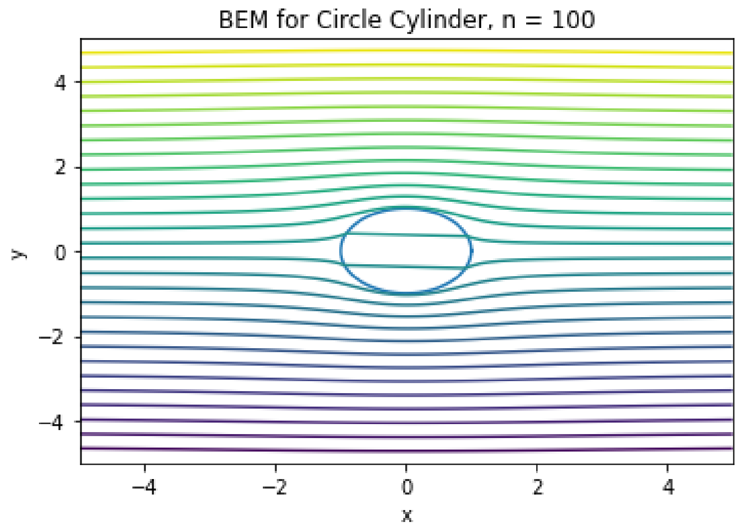

Figure 5: Streamlines for n=100

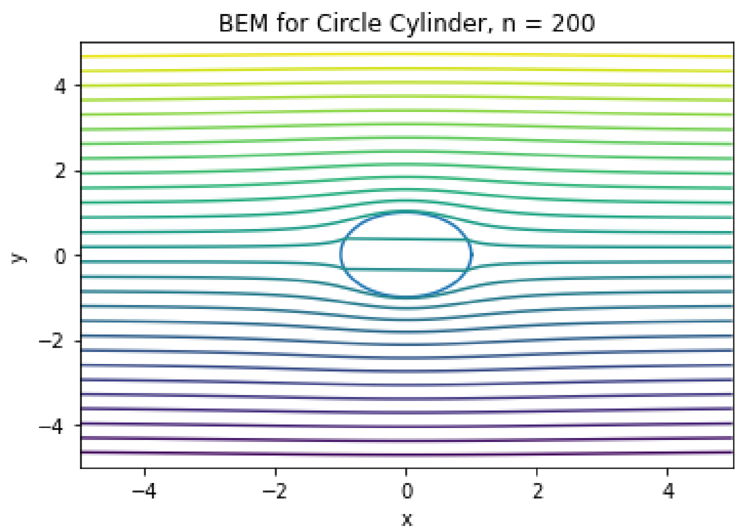

Figure 6: Streamlines for n=200

It can be seen that as N increases, the flow near the edge of the cylinder gets more refined. There is not that much of a difference as N increases though quantitatively.

Number 5

After modifying the code to apple to a uniform flow past a 4:1 ellipse parallel and also perpendicular to the flow. The following plots were made below. It can be seen that as n increases, the partial derivative gets smoother. For the horizontal ellipse, the flow doesn't seem to be affected that much because of its streamlines shape. The vertical ellipse changes the flow a lot due to the massive area blocking the flow. There is supposed to be no lines inside so it can be made better next time.

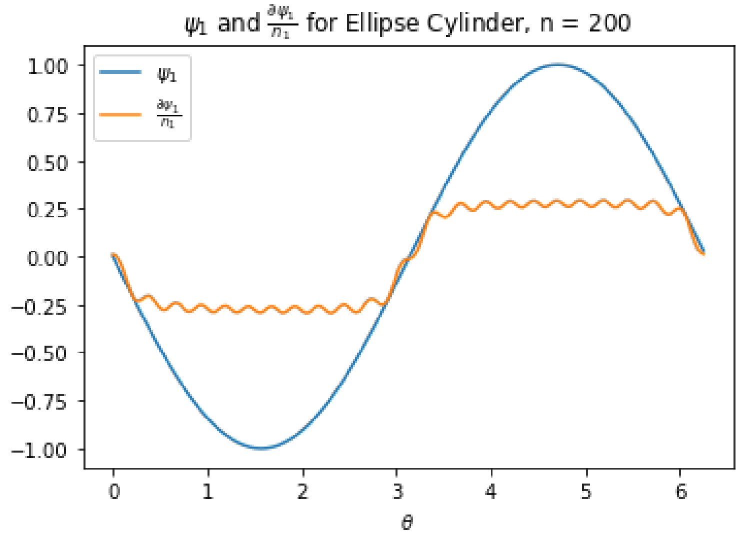

Figure 7: Plots of ψ_1 and $\frac{\partial \psi_1}{\partial n}$ for n=200 Horizontal

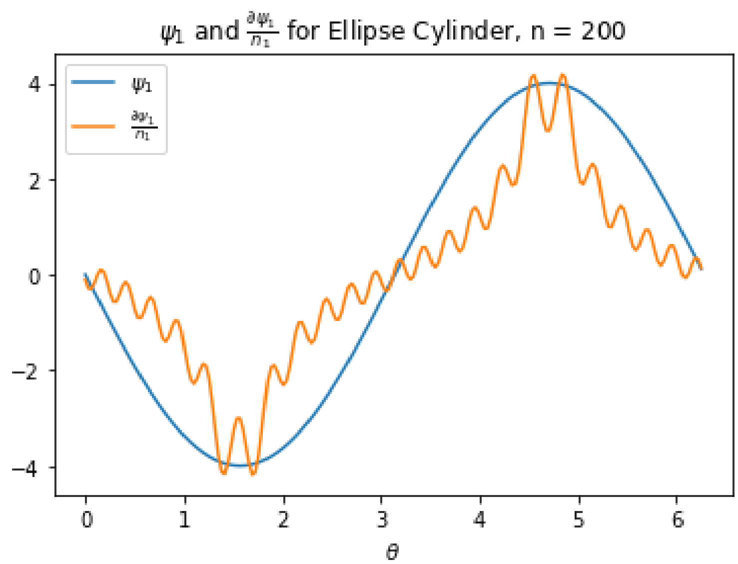

Figure 8: Plots of ψ_1 and $\frac{\partial \psi_1}{\partial n}$ for n=200 Vertical

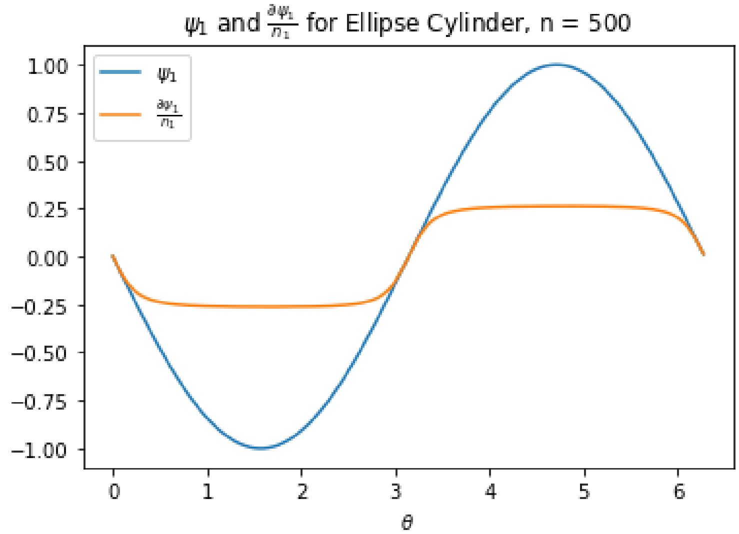

Figure 9: Plots of ψ_1 and $\frac{\partial \psi_1}{\partial n}$ for n=500 Horizontal

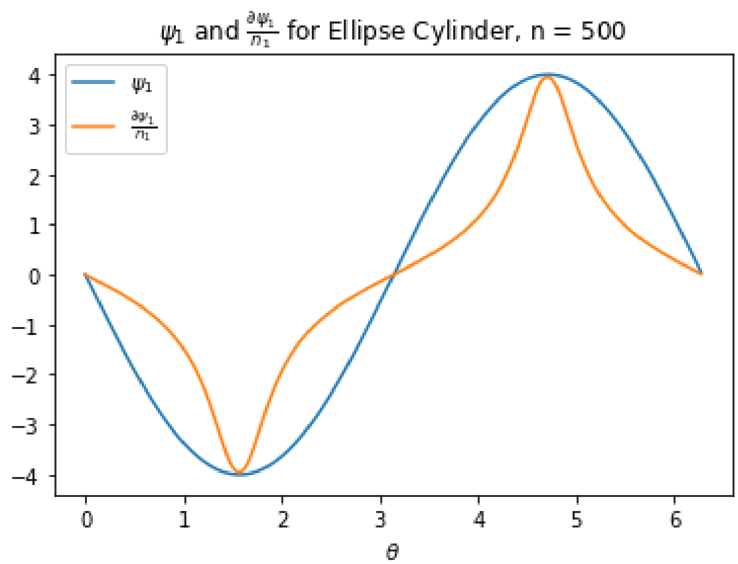

Figure 10: Plots of ψ_1 and $\frac{\partial \psi_1}{\partial n}$ for n=500 Vertical

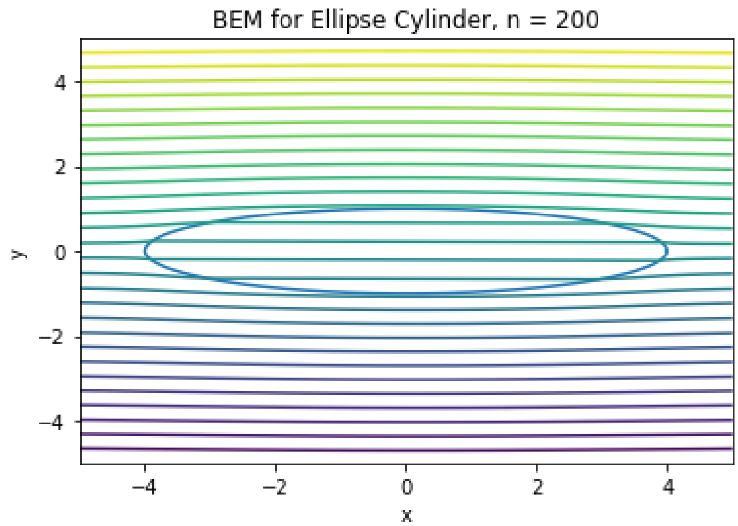

Figure 11: Streamlines for n=500 Horizontal

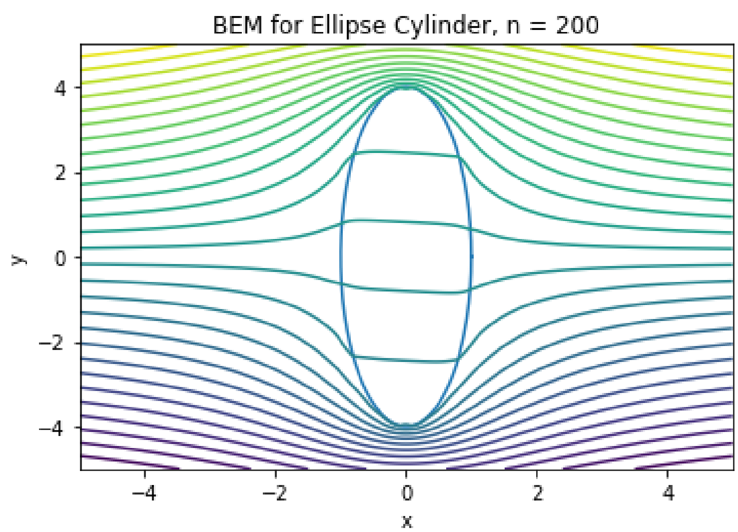

Figure 12: Streamlines for n=500 Vertical

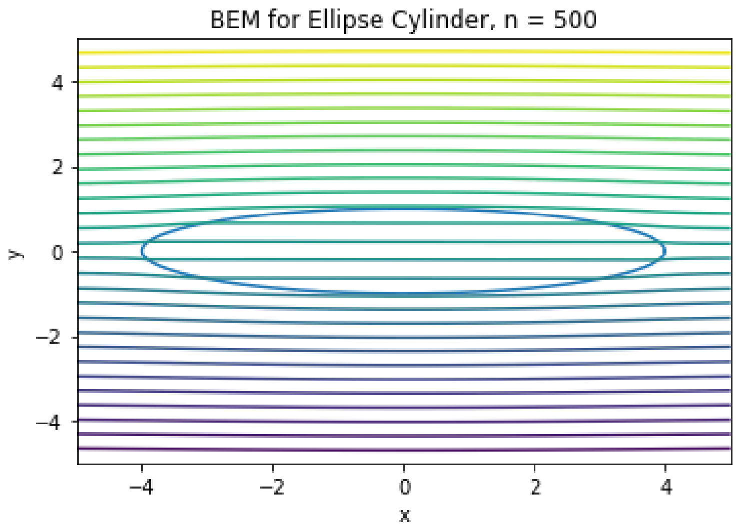

Figure 13: Streamlines for n=500 Horizontal

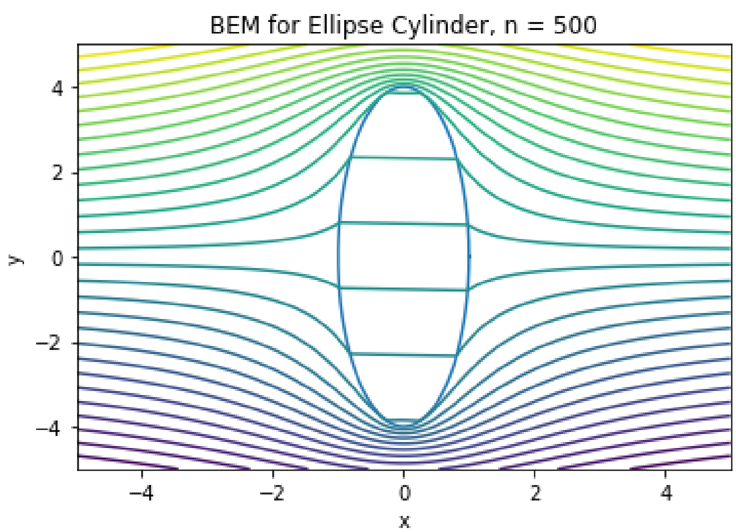

Figure 14: Streamlines for n=500 Vertical

Appendix)

Python Code for Problems 2,3,5

```
# -*- coding: utf-8 -*-
"""
Created on Fri Feb 10 10:06:00 2023

@author: jjser
"""
# Imported from CFD Project
import numpy as np
import matplotlib.pyplot as plt
import math
from numpy import linalg as la
from numpy import asarray
import pandas
import time
```

```python

#################
def G(r):
    return np.log(r)/(2*np.pi)
def dGdn(R,N,r):
    return R @ N / (2*np.pi*r)

def prob(a,b,n):
    if a == b:
        shape = 'Circle Cylinder'
    else:
        shape = 'Ellipse Cylinder'
    dtheta = 2*np.pi / n
    x0 = np.zeros(n+1)
    y0 = np.zeros(n+1)
    theta = np.zeros(n+1)

    # Creating the actual circle first
    for i in range(n+1):
        theta[i] = i * dtheta
        #r = math.sqrt((a*b)/(a**2*np.sin(theta[i])**2+b**2*np.cos(theta[i])**2))
        x0[i] = a*np.cos(theta[i])
        y0[i] = b*np.sin(theta[i])
    # Applying BEM
    A = np.zeros((n,n))
    B = np.zeros((n,n))
    area = np.zeros(n)
    cx = np.zeros(n)
    cy = np.zeros(n)
    N = np.zeros((n,2))

    #s = 2*max(a,b)
    s=5
    fig0, ax0 = plt.subplots()
    title = 'BEM for ' + shape + ', n = ' + str(n)
    ax0.set_title(title)
    ax0.set_xlabel('x')
    ax0.set_ylabel('y')
    ax0.set_xlim([-s,s])
    ax0.set_ylim([-s,s])

    for i in range(n):
        area[i] = math.sqrt((x0[i+1]-x0[i])**2+(y0[i+1]-y0[i])**2)
        cx[i] = 0.5 *  (x0[i+1]+x0[i])
        cy[i] = 0.5 *  (y0[i+1]+y0[i])
        N[i][0] = -(y0[i+1]-y0[i])
        N[i][1] =  (x0[i+1]-x0[i])
        N[i] = N[i] / la.norm(N[i])
    for i in range(n):
        for j in range(n):
            if i==j:
                B[i,j] = 0
                A[i,j] = -0.5
            else:
                R = [cx[j]-cx[i],cy[j]-cy[i]]
                r = la.norm(R)
                R = R/r
                A[i,j] = dGdn(R,N[j],r) * area[j]
                B[i,j] = G(r) * area[j]
                #ax0.quiver(cx[i],cy[i],R[0],R[1])

    # plotting
    ax0.plot(x0,y0)
    #ax0.plot(cx,cy,'r*')
    #ax0.quiver(cx,cy,N[:,0],N[:,1])

    # psi1 and dpsi1dn
    psi1 = np.zeros(n)
    for i in range(n):
        psi1[i] = -y0[i]

    dpsi1dn = la.inv(B) @ A @ psi1.T
```

```


    # Plot psi1 dpsi1dn
    fig, ax = plt.subplots()
    title = '$\\psi_1$ and $\\frac{\\partial \\psi_1}{n_1}$ for ' + shape + ', n = ' + str(n
    )
    ax.set_title(title)
    ax.set_xlabel('$\\theta$')

    ax.plot(theta[:-1],psi1,label='$\\psi_1$')
    ax.plot(theta[:-1],dpsi1dn,label='$\\frac{\\partial \\psi_1}{n_1}$')
    ax.legend()

    # Plotting the Cylinder
    g = 500
    x = np.linspace(-s,s,g)
    y = np.linspace(-s,s,g)
    X,Y = np.meshgrid(x,y)
    psi = np.zeros((g,g))
    # Streamfunction Contours
    for k in range(n):
        rx = cx[k] - X
        ry = cy[k] - Y
        rmag = np.sqrt(rx**2+ry**2)
        psi += ((rx*N[k][0]+ry*N[k][1])/(2*np.pi*r)-G(rmag)*dpsi1dn[k])*area[k]
    psi += Y
    levels = np.linspace(np.min(psi),np.max(psi),30)
    '''
    for i in range(g):
        for j in range(g):
            if X[i,j]:
                psi[i,j] = NaN
    '''

    ax0.contour(x, y, psi, levels=levels)
##### BEM for a 4:1 ellipse
prob(1,1,50)
prob(1,1,100)
prob(1,1,200)

prob(4,1,200)
prob(1,4,200)
prob(4,1,500)
prob(1,4,500)
start_time = time.time()
print("--- %10s seconds ---" % np.round((time.time() - start_time),4))
```