Plot the Rankine Oval by using python code

Ranjithkumar B

I. Problem Definition

This work presents the solving potential flow equation (Rankine Oval) Numerically by using finite diffrence method, and compare those results with the Exact solution for understanding the FDM.

II. Governing Equations

Rankine Oval will be formed by combining the three two types of elementary flows, those are
1) Uniform Flow (First elementary flow)
2) Source and sink (Second elementary flow)

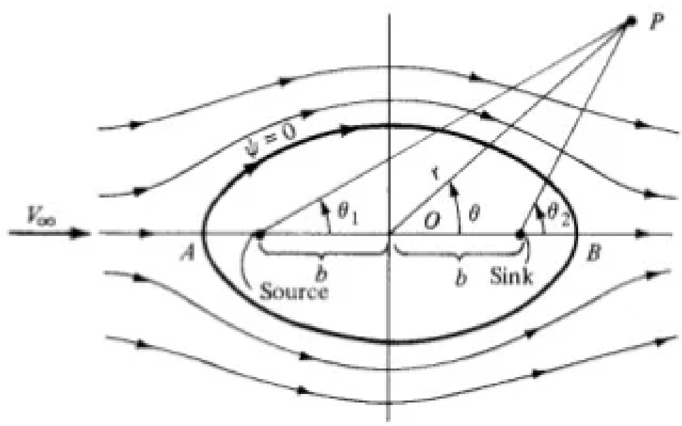

Fig. 1 Flow over a Rankine oval

Here the stream function equation is generally used for computing velocity components.

$$\psi = V_\infty r \sin\theta + \frac{\Lambda}{2\pi}\theta_1 - \frac{\Lambda}{2\pi}\theta_2$$

By using the stream function, the Radial and Tangential velocities are

$$u = \frac{\partial\psi}{\partial y}$$
$$v = \frac{\partial\psi}{\partial x}$$

The distance between the souce to center and sink to center should be same for Rankine oval.

Here the free stream velocity and source and sink strength are $V_\infty = 10\ m/s$ and $\Lambda = 100\ m^2/s$

III. Numerical calculation

By using finite diffrence method for solving the governing equation for 200 nodes for each direction.

$$
\begin{aligned}
u &= \frac{\partial \psi}{\partial y} \\
&= \frac{\psi_2 - \psi_1}{y_2 - y_1} \\
v &= \frac{\partial \psi}{\partial x} \\
&= \frac{\psi_2 - \psi_1}{x_2 - x_1}
\end{aligned}
$$

The above diffrence method only can solve or use $(N-1)$ grid only.For the last grid we use extrapolation method to get the velocity components.

$$
\begin{aligned}
u_N &= 2u_{N-1} - u_{N-2} \\
v_N &= 2v_{N-1} - v_{N-2}
\end{aligned}
$$

IV. Analytical calculation

Based on the coordinates and flow condition itself the exact solution obtained by below Equations

$$
\begin{aligned}
u &= V_\infty + \frac{\Lambda}{2\pi}\left(\frac{X+b}{(X+b)^2+Y^2} - \frac{X-b}{(X-b)^2+Y^2}\right) \\
v &= \frac{\Lambda}{2\pi}\left(\frac{Y}{(X+b)^2+Y^2} - \frac{Y}{(X-b)^2+Y^2}\right)
\end{aligned}
$$

The X and Y denotes the co-ordinates from the center.Here the origin offset to (0.5,0.5)

V. Solution and Observation

The numerical and Eaxct solution was ploted in a single plot for the comparision purpose.

1) For increaing the gid point, the accuracy of the solution also increase for certain limit.
2) The orange stream lines are taken form the Analytical equation, and the Blue color stream lines are computed by numerical method.
3) Those two streamlines are more over allined one on one in the figure 2.

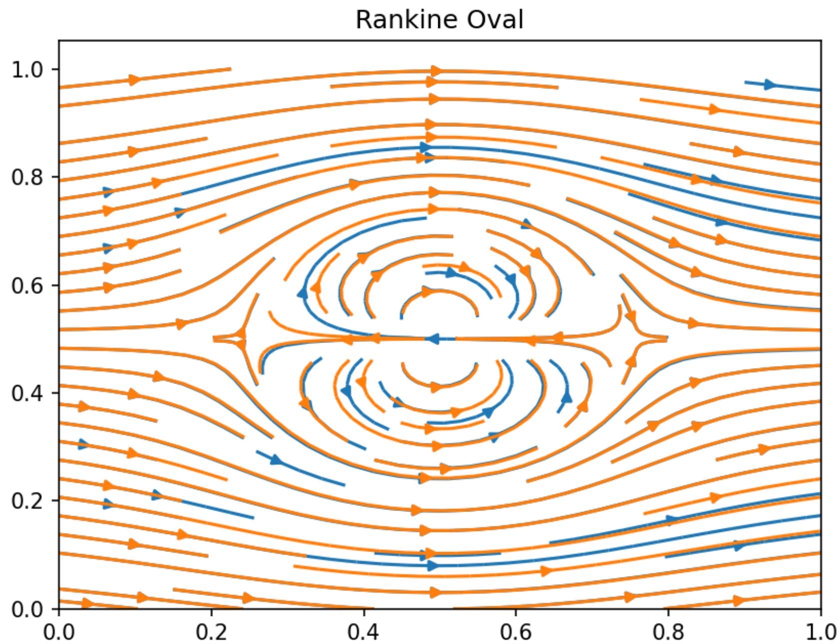

Fig. 2 Numerical and exact solution of Rankine oval

References

[1] Jhon D Anderson Jr., "Fundamentals of Aerosynamics," Fifth Edision 2010.

[2] Jhon D Anderson Jr., "Computational Fluid Dynamics The basics with applications,"Indian edition

A. Appendix - Python code

This section contains the *Python* code of Rankine Oval.

```python
#!/bin/python3

# Rankine -fULL Oval

import numpy as np
import matplotlib.pyplot as plt

# Quadrilateral mesh grid
x,y=np.meshgrid(np.linspace(0,1,200),np.linspace(0,1,200))

# Number of grid ponits in x axis
Nx=len(x)

# Number of grid ponits in y axis
Ny=len(y)

# x directional velocity commponenit array
u=np.zeros([Nx,Ny])

# y directional velocity component array
v=np.zeros([Nx,Ny])

# Free stream velocity
V_inf=10

# Source and sink strength
Lamda=100

# x location of the source
xc1=0.48

# x location of the sink
xc2=0.52

# The distace between the source/sink to the center
b=(xc2-xc1)/2

# location of the center
xc=0.5
yc=0.5

# computing section
# polar coordinate values from the center

# radius
r=np.sqrt((x-xc)**2+(y-yc)**2)
# Angle of the each postion w.r.t center
theta=np.arctan2(y-yc,x-xc)
# Angle of the each postion w.r.t source center
t1=np.arctan2(y-yc,x-xc1)
# Angle of the each postion w.r.t sink center
t2=np.arctan2(y-yc,x-xc2)

# Calculating Stream function
psi=V_inf*r*np.sin(theta)+(Lamda*(t1-t2)/2/np.pi)


# Numerical computing section
# Calculating velocity components
for i in range(0,Nx-1):
    for j in range(0,Ny-1):
        u[j,i]=(psi[j+1,i]-psi[j,i])/(y[j+1,i]-y[j,i])
        v[j,i]=-(psi[j,i+1]-psi[j,i])/(x[j,i+1]-x[j,i])

# calculating last grid ponit velocity components
```

```python
for i in range(0,Nx):
    u[i,Nx-1]=2*u[i,Nx-2]-u[i,Nx-3]
    u[Nx-1,i]=2*u[Nx-2,i]-u[Nx-3,i]

    v[i,Nx-1]=2*v[i,Nx-2]-v[i,Nx-3]
    v[Nx-1,i]=2*v[Nx-2,i]-v[Nx-3,i]

# Analytical solution

# Offceting the Coordination
X=x-xc
Y=y-yc

# x component velocity
a1=(X+b)/((X+b)**2+Y**2)
a2=(X-b)/((X-b)**2+Y**2)
ana_u=V_inf+(Lamda*(a1-a2)/2/np.pi)

# y component velocity
a1=Y/((X+b)**2+Y**2)
a2=Y/((X-b)**2+Y**2)
ana_v=Lamda*(a1-a2)/2/np.pi

# plotting section

plt.figure()
# plotting the numerical solution
plt.streamplot(x,y,u,v)
# plotting the analytical solution
plt.streamplot(x,y,ana_u,ana_v)
plt.title("Rankine Oval")
plt.savefig("Result.png",dpi=150)
plt.show()
```