Supersonic streamlines

March 31, 2025

[1]:
```
import numpy                as np
import matplotlib.pyplot    as plt
from   scipy.integrate import solve_ivp
```

```
/home/lukas/.local/lib/python3.10/site-
packages/matplotlib/projections/__init__.py:63: UserWarning: Unable to import
Axes3D. This may be due to multiple versions of Matplotlib being installed (e.g.
as a system package and as a pip package). As a result, the 3D projection is not
available.
  warnings.warn("Unable to import Axes3D. This may be due to multiple versions
of "
```

1 Parameters

[2]:
```
M       = 2                         # Mach number
epsilon = 0.1                       # maximum thickness of the profile
H       = lambda x: 0.5 - 2*x**2    # normalized thickness function
dHdx    = lambda x: -4*x            # H'
```

2 Coordinate transform

[3]:
```
xi  = lambda x,y: x - np.sqrt(M**2-1)*y
eta = lambda x,y: x + np.sqrt(M**2-1)*y
```

3 Division into sub-regions

[4]:
```
upper = lambda x,y: (y>0) & (xi(x,y) >-1/2) & (xi(x,y) <1/2)
lower = lambda x,y: (y<0) & (eta(x,y)>-1/2) & (eta(x,y)<1/2)
```

4 Analytical velocity perturbation

The velocity components are given by

$$u = 1 + \epsilon \frac{\partial \phi_1}{\partial x},$$

$$v = \epsilon \frac{\partial \phi_1}{\partial y}.$$

We will denote

$$u_1 = \frac{\partial \phi_1}{\partial x} \quad \text{and} \quad v_1 = \frac{\partial \phi_1}{\partial y}.$$

We can express u_1 and v_1 in terms of the analytical solution $\tilde{\phi}$ as follows:

$$\frac{\partial \phi_1}{\partial x} = \begin{cases} (\partial \xi / \partial x)\, \mathrm{d}\tilde{\phi}/\mathrm{d}\xi = \mathrm{d}\tilde{\phi}/\mathrm{d}\xi = -(M_\infty^2 - 1)^{-1/2} H'(\xi) & \text{for } y > 0 \wedge -1/2 < \xi < 1/2 \\ (\partial \eta / \partial x)\, \mathrm{d}\tilde{\phi}/\mathrm{d}\eta = \mathrm{d}\tilde{\phi}/\mathrm{d}\eta = -(M_\infty^2 - 1)^{-1/2} H'(\eta) & \text{for } y < 0 \wedge -1/2 < \eta < 1/2 \\ 0 & \text{otherwise} \end{cases}$$

and

$$\frac{\partial \phi_1}{\partial y} = \begin{cases} (\partial \xi / \partial y)\, \mathrm{d}\tilde{\phi}/\mathrm{d}\xi = -(M_\infty^2 - 1)^{1/2} \mathrm{d}\tilde{\phi}/\mathrm{d}\xi = H'(\xi) & \text{for } y > 0 \wedge -1/2 < \xi < 1/2 \\ (\partial \eta / \partial y)\, \mathrm{d}\tilde{\phi}/\mathrm{d}\eta = (M_\infty^2 - 1)^{1/2} \mathrm{d}\tilde{\phi}/\mathrm{d}\eta = -H'(\eta) & \text{for } y < 0 \wedge -1/2 < \eta < 1/2 \\ 0 & \text{otherwise} \end{cases}.$$

[5]:
```
u1 = lambda x,y: -dHdx( xi(x,y)*upper(x,y)
                      + eta(x,y)*lower(x,y) ) / np.sqrt(M**2-1)

v1 = lambda x,y: dHdx(xi(x,y))*upper(x,y) - dHdx(eta(x,y))*lower(x,y)

u  = lambda x,y: 1 + epsilon * u1(x,y)
v  = lambda x,y:     epsilon * v1(x,y)
```

5 Pressure

The pressure perturbation $(p - p_\infty)$ is typically expressed in dimensionless form as the *pressure coefficient*

$$C_p = \frac{p - p_\infty}{\rho_\infty u_\infty^2 / 2}.$$

The perssure is related to velocity through the Bernoulli equation. In dimensionless form, the Bernoulli equation can be expressed as follows:

$$C_p = 1 - |\vec{U}|^2.$$

[6]:
```
p  = lambda x,y: 1.0 - ( u(x,y)**2 + v(x,y)**2 )
```

6 Distribution of points for plotting

[7]:
```
xmin, xmax = -0.75, 1.5  # boundaries of the visualized region
ymin, ymax = -0.5 , 0.5
N          = 100         # number of points for pressure in each direction
Nu         = 10          # number of velocity vectors in each direction
```

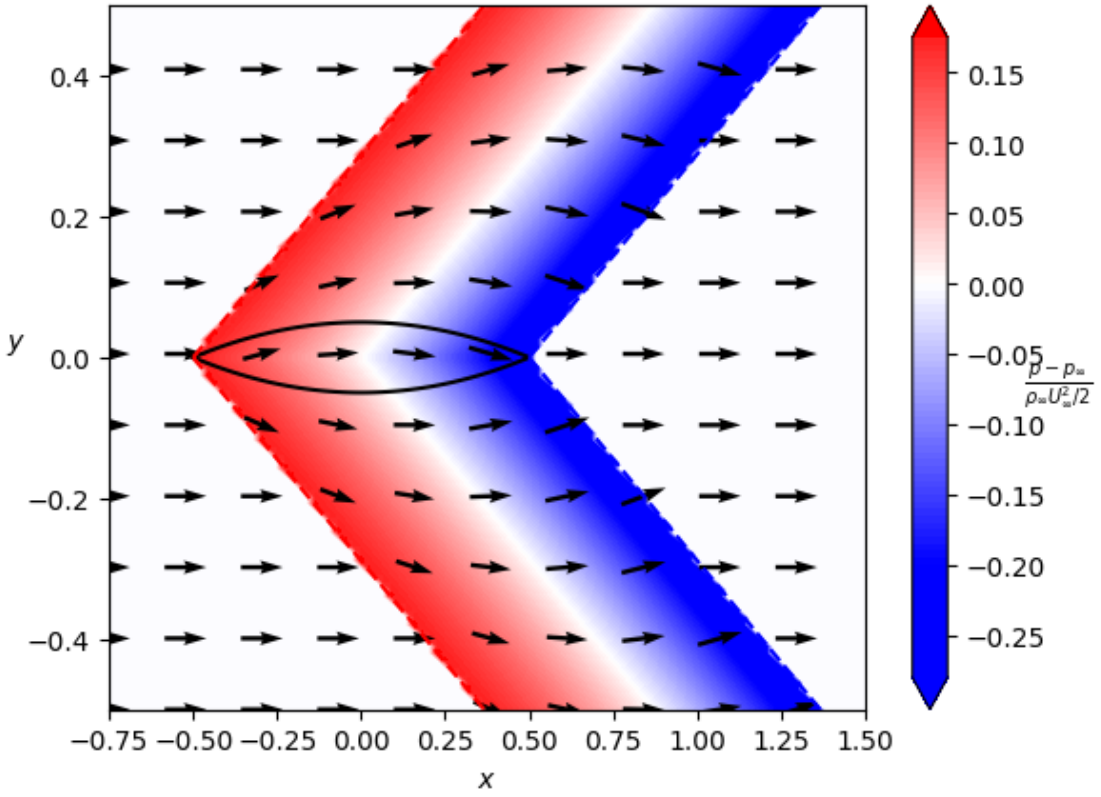
Create uniform grids of points

[8]:
```
x,y   = np.meshgrid( np.linspace(xmin,xmax,N) ,
                     np.linspace(ymin,ymax,N) )
iu    = slice(0,N,int(N/Nu))
xu,yu = x[iu,iu], y[iu,iu]
```

7 Plot

[9]:
```
fig,ax = plt.subplots()

# Pressure
p_field = ax.contourf(x,y,p(x,y),N
            ,cmap='bwr',vmin=-2*epsilon,vmax=2*epsilon,extend='both')

cbar = plt.colorbar(p_field)

cbar.set_label(r'$\frac{p-p_\infty}{\rho_\infty U_\infty^2/2}$'
              , rotation='horizontal')

# Velocity vectors
ax.quiver(xu,yu,u(xu,yu),v(xu,yu),angles='xy',pivot='mid')

# Shape of the profile
x_shape = np.linspace(-0.5, 0.5, N)
y_shape = epsilon * H(x_shape)
ax.plot(x_shape, y_shape, 'k-', x_shape, -y_shape, 'k-')

# Characteristics (shock-wave fronts)
y_char = np.array([0, ymax])
x_char = np.sqrt(M**2-1)*y_char
ax.plot(x_char-0.5, y_char, 'r--')
ax.plot(x_char-0.5,-y_char, 'r--')
ax.plot(x_char+0.5, y_char, 'b--')
ax.plot(x_char+0.5,-y_char, 'b--')

# Labels
ax.set_xlabel(r'$x$')
ax.set_ylabel(r'$y$',rotation='horizontal')
```

[9]: Text(0, 0.5, 'y')

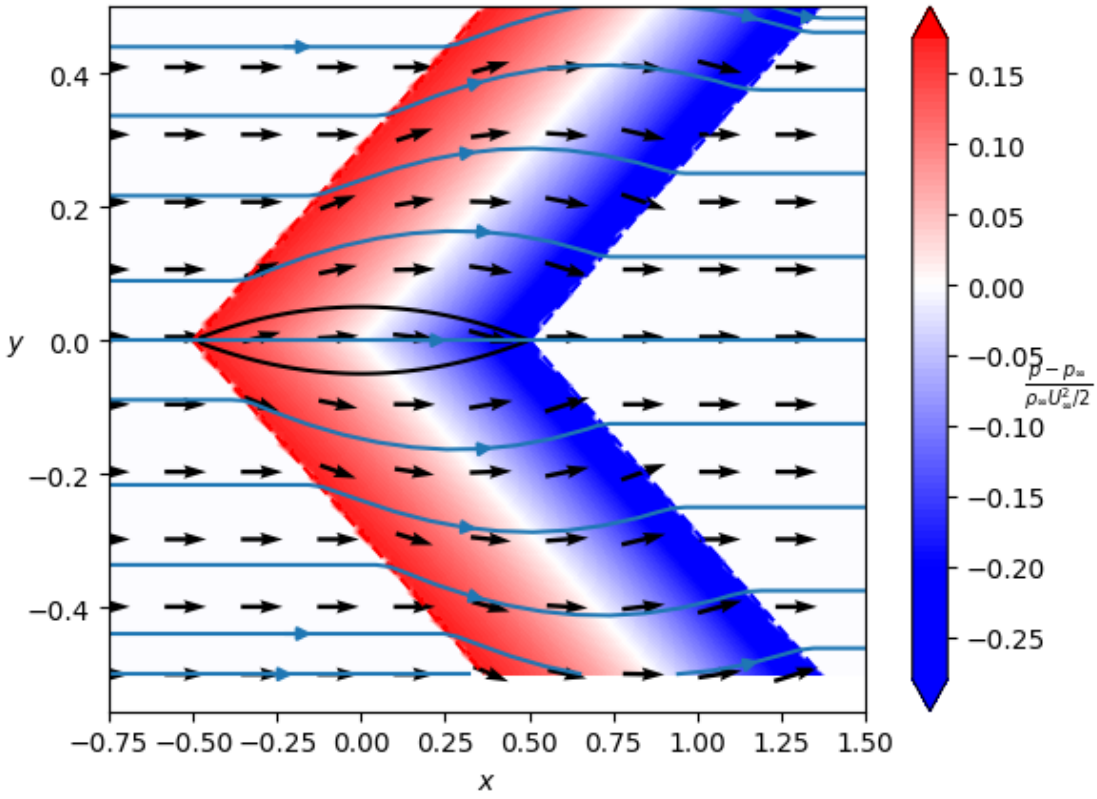


8 Streamlines

One possibility is to use matplotlib.pyplot.streamplot, but it uses interpolation between the grid points, which introduces additional numerical error. Thus, one must use a sufficient number of grid points.

[10]:
```
ax.streamplot(x,y,u(x,y),v(x,y),density=0.3,broken_streamlines=False)
fig
```

[10]:

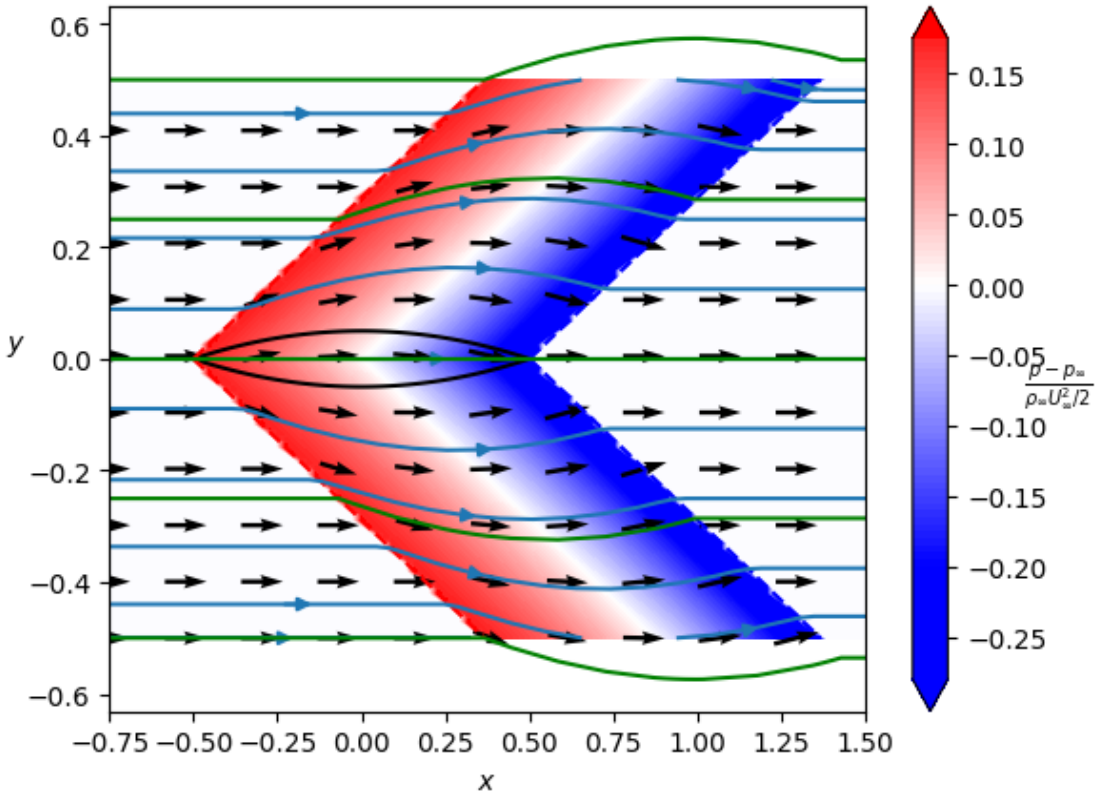


The streamlines can also be computed with an initial value solver, scipy.integrate.solve_ivp, as follows:

[11]:
```
dydx   = lambda x,y: v(x,y)/u(x,y)
y0     = np.linspace(ymin, ymax, int(Nu/2))
x_span = (xmin, xmax)

sol    = solve_ivp(dydx, x_span, y0, vectorized=True, atol=1e-8, rtol=1e-8)
```

[12]:
```
ax.plot(sol.t, sol.y.T, 'g-')
fig
```

[12]:


0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
y
−0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00 1.25 1.50
x
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
$\frac{p-p_\infty}{\rho_\infty U_\infty^2/2}$

[ ]: