# Flat plate

April 30, 2025

[49]:
```
import numpy                as np
import matplotlib.pyplot    as plt
from   scipy.integrate import quad, solve_ivp
```

## 1 Parameters

[2]:
```
alpha = 10 / 180 * np.pi # angle of attack
```

[3]:
```
y_bar  = lambda x: np.zeros_like(x)
dy_bar = lambda x: np.zeros_like(x)
```

## 2 Uniform far field solution

[4]:
```
u_inf = np.cos(alpha)    # incoming flow
v_inf = np.sin(alpha)
```

## 3 Vortex sheet solution

### 3.1 Vortex distribution

[5]:
```
gamma = lambda x: -2*np.sin(alpha)*np.sqrt((1-x)/x)
```

### 3.2 Horizontal velocity

[6]:
```
def uc(x,y):
    if y==0:
        return 0
    else:
        integrand = lambda xi: gamma(xi) * y / ( (x-xi)**2 + y**2 )
        result = quad(integrand, 0, 1)
        return -result[0] / (2*np.pi)
```

[7]:
```
u = lambda x,y: u_inf + uc(x,y)
```

### 3.3 Vertical velocity

[8]:
```
def vc(x,y):
    if y==0 and x>=0 and x<=1:
        return dy_bar(x) - np.sin(alpha)
    else:
        integrand = lambda xi: gamma(xi) * (x-xi) / ( (x-xi)**2 + y**2 )
        result = quad(integrand, 0, 1)
        return result[0] / (2*np.pi)
```

[9]:
```
v = lambda x,y: v_inf + vc(x,y)
```

## 4 Visualization

### 4.1 Plate shape

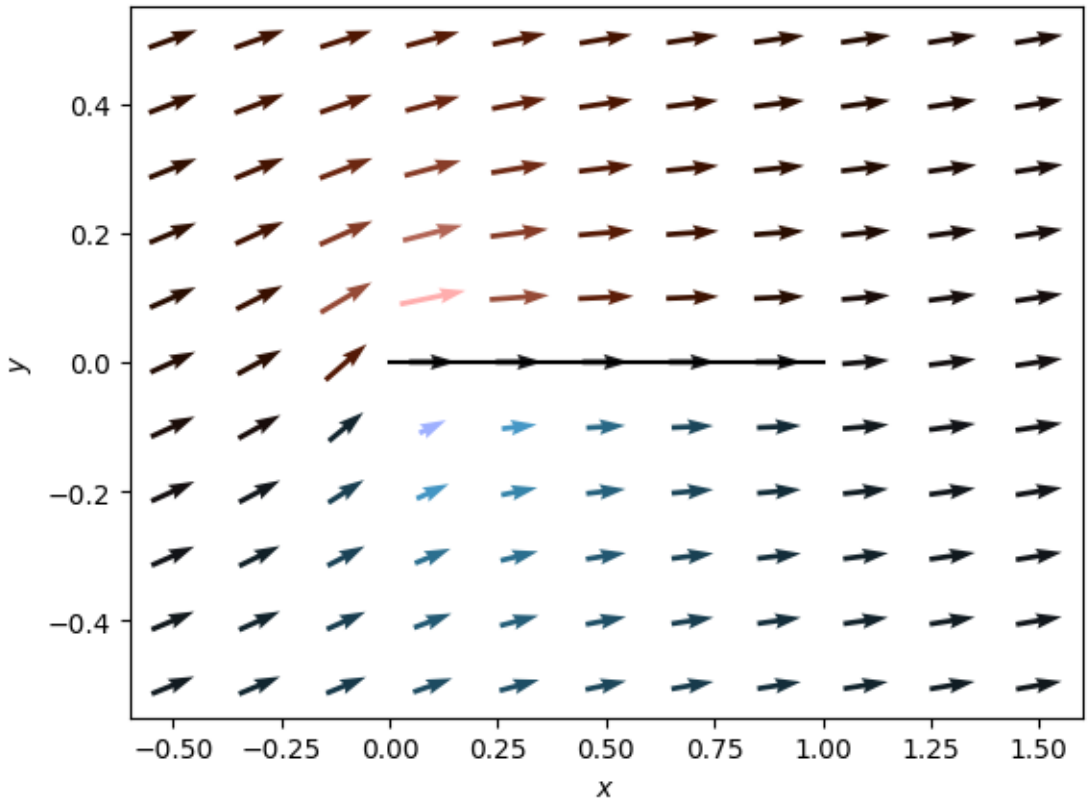
[10]:
```
N_plate = 11
x_plate = np.linspace(0, 1, N_plate)
y_plate = y_bar(x_plate)
```

### 4.2 Velocity vectors

[11]:
```
N_arrows   = 11
xmin, xmax = -0.5, 1.5
ymin, ymax = -0.5, 0.5

x_vec, y_vec = np.meshgrid( np.linspace(xmin, xmax, N_arrows),
                            np.linspace(ymin, ymax, N_arrows) )

u_vec = np.zeros((N_arrows,N_arrows))
v_vec = np.zeros((N_arrows,N_arrows))

for i in range(N_arrows**2):
    u_vec.flat[i] = u(x_vec.flat[i], y_vec.flat[i])
    v_vec.flat[i] = v(x_vec.flat[i], y_vec.flat[i])
```

[13]:
```
fig,ax = plt.subplots()
ax.plot(x_plate, y_plate, 'k-')
ax.quiver(x_vec, y_vec, u_vec, v_vec, np.sqrt(u_vec**2+v_vec**2),
          angles='xy',pivot='mid',cmap='berlin')
ax.set_xlabel(r'$x$')
ax.set_ylabel(r'$y$')
```

[13]:
```
Text(0, 0.5, '$y$')
```



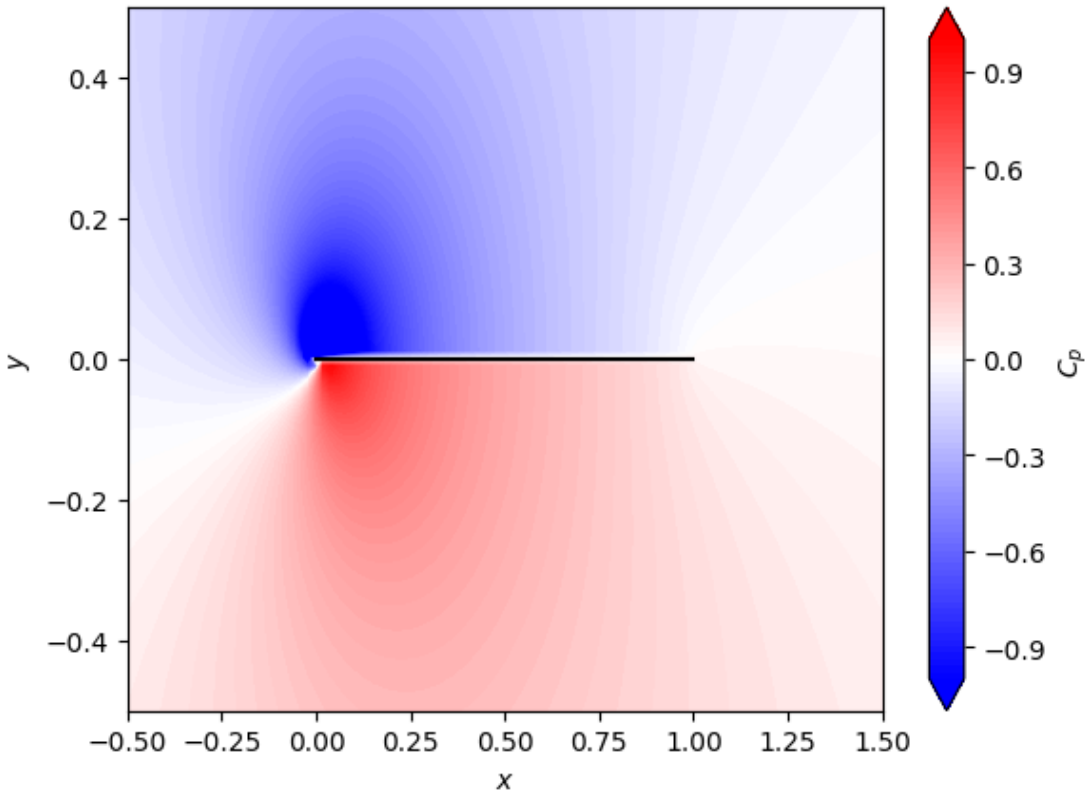

### 4.3 Pressure

[47]:
```
Np = 101

xp,yp = np.meshgrid( np.linspace(xmin,xmax,Np),
                     np.linspace(ymin,ymax,Np) )

up = np.zeros((Np,Np))
vp = np.zeros((Np,Np))

for i in range(Np**2):
    up.flat[i] = u(xp.flat[i], yp.flat[i])
    vp.flat[i] = v(xp.flat[i], yp.flat[i])

p = 1 - (up**2 + vp**2)
```

```
/tmp/ipykernel_2081/1271131423.py:6: IntegrationWarning: The integral is
probably divergent, or slowly convergent.
  result = quad(integrand, 0, 1)
```

[48]:
```
fig,ax = plt.subplots()

vmin = -1
vmax = 1
levels = np.linspace(vmin,vmax,Np)

pplot = ax.contourf(xp,yp,p,levels,cmap='bwr',vmin=vmin,vmax=vmax,extend='both')

cbar = fig.colorbar(pplot)
cbar.locator.nbins = 7
cbar.set_label(r'$C_p$')

ax.plot(x_plate, y_plate, 'k-')

ax.set_xlabel(r'$x$')
ax.set_ylabel(r'$y$')
```

[48]:
```
Text(0, 0.5, '$y$')
```



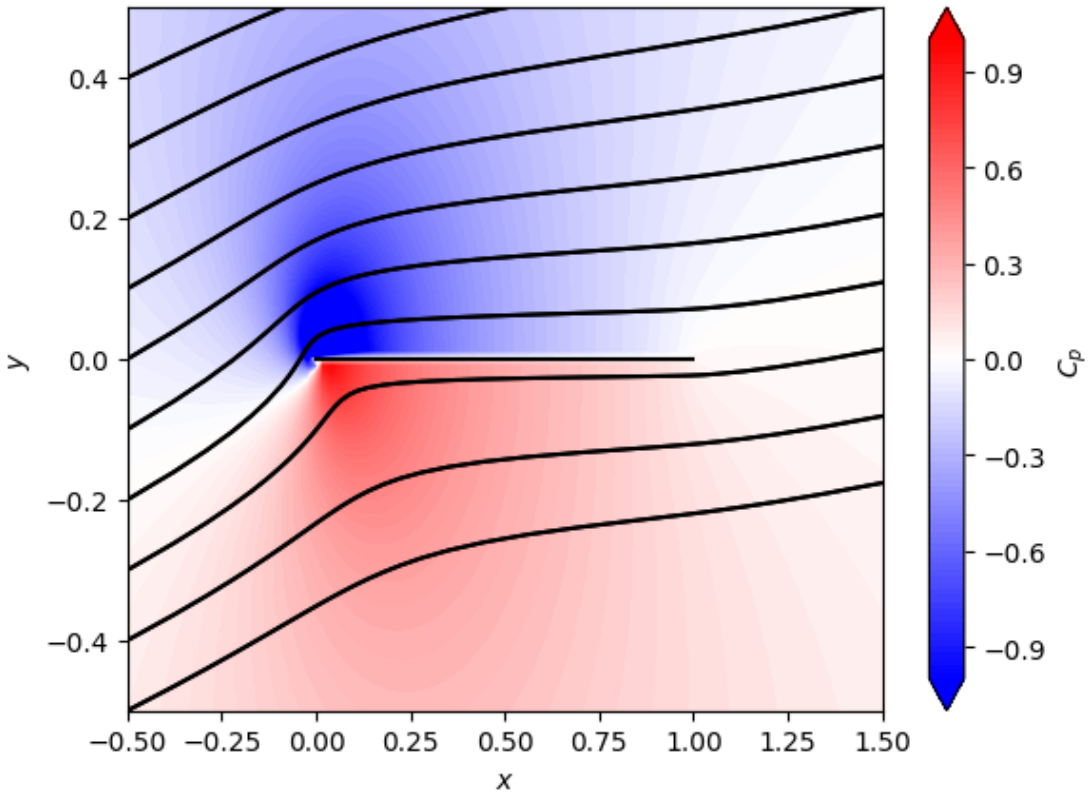
### 4.4 Streamlines

[55]:
```
dy_dx = lambda x,Y: [ v(x,y) / u(x,y) for y in Y ]

Ns = 11
y0 = np.linspace(ymin, ymax, Ns)

sol = solve_ivp(dy_dx, (xmin,xmax), y0, vectorized=False, atol=1e-8, rtol=1e-8)
```

```
/tmp/ipykernel_2081/1271131423.py:6: IntegrationWarning: The integral is
probably divergent, or slowly convergent.
  result = quad(integrand, 0, 1)
```

[57]:
```
ax.plot( sol.t, sol.y.T, 'k-' )
ax.set_ylim(ymin,ymax)
fig
```

[57]:

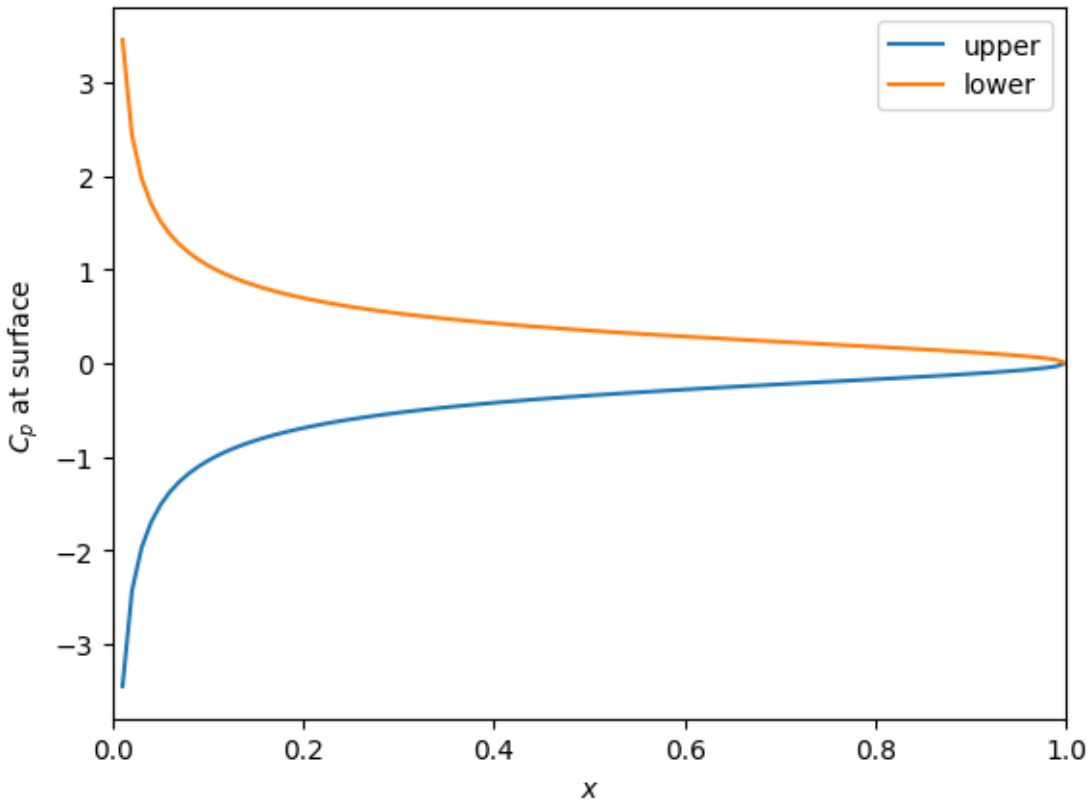


### 4.5 Surface pressure

[67]:
```
uc_0p = lambda x: -gamma(x) / 2
uc_0m = lambda x:  gamma(x) / 2
cp_0p = lambda x:  gamma(x)
cp_0m = lambda x: -gamma(x)

x_surf = np.linspace(0,1,Np)

fig,ax = plt.subplots()
ax.plot(x_surf, cp_0p(x_surf), label='upper')
ax.plot(x_surf, cp_0m(x_surf), label='lower')
ax.set_xlim(0,1)
ax.set_xlabel(r'$x$')
ax.set_ylabel(r'$C_p$ at surface')
ax.legend()
```

```
/tmp/ipykernel_2081/2581333322.py:1: RuntimeWarning: divide by zero encountered
in divide
  gamma = lambda x: -2*np.sin(alpha)*np.sqrt((1-x)/x)
```

[67]:
```
<matplotlib.legend.Legend at 0x7ff33c28ee90>
```



# 5 Forces

## 5.1 Lift coefficient

[63]:
```
Gamma = quad(gamma,0,1)
cL = -2*Gamma[0]
print('C_L        = ', cL)
print('2 pi alpha = ', 2*np.pi*alpha)
```

```
C_L        =  1.0910636785353718
2 pi alpha =  1.096622711232151
```