Eduard Algar

Analysis of an airfoil

Index:

Introduction & aims:

- Laplace equation.
- Solve the behaviour of a fluid around a body.
- Compare the results with the analytical ones (cylinder)

Laplace’s equation:

- From where it comes from?

$$\nabla^2\psi = 0 \qquad \frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial y^2} = 0$$

- Navier Stokes.
- Doing the following assumptions:
  - Incompressible flow ($M<0.3$)
  - Inviscid flow (Kutta condition), $\xi = \nabla * \vec{V} = 0$
  - Steady flow.

Velocity and pressure fields:

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

- Bernoulli equation

Boundary conditions:

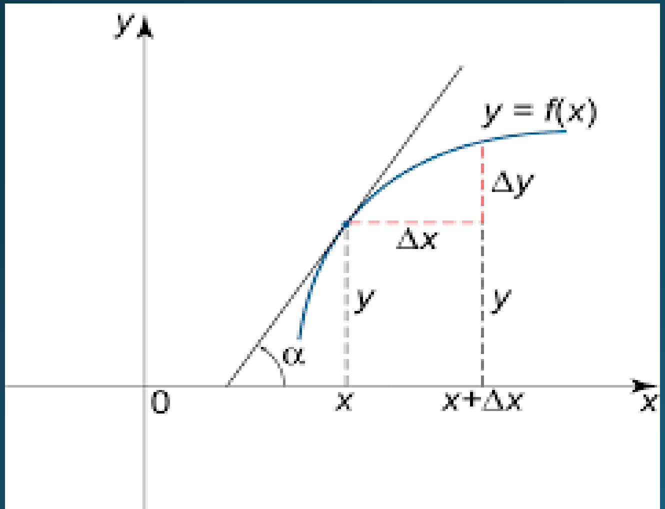
	$\vec{V}$	ψ	ϕ
Inlet	u=V_∞ v=0	$\frac{\partial\psi}{\partial x} = 0\,, \frac{\partial\psi}{\partial y} = V_\infty$	$\frac{\partial\phi}{\partial x} = V_\infty\,, \frac{\partial\phi}{\partial y} = 0$
Outlet			
Solid Wall	$V_n = 0, V_t \neq 0$	$\psi = \mathrm{C}$	$\frac{\partial\phi}{\partial n} = 0$

Mathematical solution:

- Derivatives.
- Taylor series.

$$\frac{\Delta y}{\Delta x} = \frac{f(x_i + \Delta x) - f(x_i)}{\Delta x}$$

$$\lim_{\Delta x \to 0} \frac{f(x_i + \Delta x) - f(x_i)}{\Delta x} = \frac{dy}{dx}$$



$$p(x) = f(a) + f'(a)(x-a) + \frac{f''(a)(x-a)^2}{2!} + \ldots + \frac{f^{(n)}(a)(x-a)^n}{n!}$$

Mathematical solution

Finite differences

Two types of solutions:

- implicit
- explicit

MF solution proposal method.

Discussion of the method.

$$f''(x_i) = \frac{f(x_i + h) - 2f(x_i) + f(x_i - h)}{2\Delta x^2}$$

$$f_i{}^{k+1} = f_i{}^k + \alpha\frac{\Delta t}{\Delta y^2}\left(f_{i+1}{}^k - 2f_i{}^k + f_{i-1}{}^k\right)$$

Coding:

- Definition of the grid (L = 100, T= 100, n = 100, m = 100)
- Initial conditions.
- OOP.
- Creation of the grid & initial guess.

File Edit View Source Project Addins About

Start Page | NACA.py | airfoil+NACA.py | (*) cilindro_simple.py | airfoil_naca.py

```python

densidad = 1.225
class VECTOR():

    def __init__(self,x=0,y=0):
        self.x=x
        self.y=y
        self.modulo = 0
        self.mod = False
    def Modulo(self):
        mod = M.sqrt(self.x**2+self.y**2)
        self.modulo= mod
        self.mod= True
    def __str__(self):
        return str(self.x) +' '+str(self.y)
    def normaliza(self):
        if not self.mod:
            self.Modulo()
        self.x/=self.modulo; self.y/=self.modulo

class cuerpo(object):
    X=0
    Y=0
    r=0
    vx=[1]
    vy=[1]

    def __init__(self,x,y,r=0):
        self.X = x
        self.Y = y
        self.r = r
        self.vx=[]
        self.vy=[]
        i=0
        while i  <=(2*M.pi):
            self.vx.append(self.r*M.cos(i) +self.X)
            self.vy.append(self.r*M.sin(i)+self.Y)
            i+=0.0001

    def contenido(self, x,y):
        x =x*dt
        y= y+dy
        cont = False
        if((((x-self.X)**2)+((y-self.Y)**2)))<=self.r**2:
            cont =True
        return cont


def stream_function(x,y,K):
    x = x*dt
    y = y*dy
    resultado = 5250   #700000 vortex 70000 vortex + doublet 7000  700 vortex + double (invertit) 5000 doublet -100000
    '''if x!=0 and y!=0:
        num= -K*M.sin(M.atan(x/y))
        cuo = 2*M.pi*M.sqrt(x**2+y**2)
        resultado = num/cuo #+ Vinf*M.cos(M.atan(i/m))*M.sqrt(x**2+y**2)*M.sin(M.atan(x/y))
    else:
        num= -K*M.sin(M.pi/2)
        cuo = 2*M.pi*M.sqrt(x**2+y**2)
        resultado = num/cuo#+Vinf*M.cos(M.atan(i/m))*M.sqrt(x**2+y**2)*M.sin(M.atan(x/y))'''
```

File Edit View Source Project Addins About

Start Page | NACA.py | airfoil+NACA.py | cilindro_simple.py | airfoil_naca.py

```python
L = 100
T= 100
n = 100
m = 100
dt = T/m
dy = L/n
u=100
v=0
K=200
Vinf =u
ro = 1.225
iteracion = 200
malla = []

cilindro = cuerpo(50,50,10)

filas = n + 1 ; columnas = m + 1;

for i in range(filas):
    malla.append([0]*columnas)

b=0
i=0
while b < len(malla[0]):
    malla[0][i] = L *Vinf
    malla[n][b] = 0
    i +=1
    b += 1

i = 0
b=0
while (i < len(malla)):
    malla[i][0] = Vinf * (n-i)*dy
    malla[i][m] = Vinf * (n-i)*dy
    b+=1
    i+=1

i = 0
while i < len(malla):
    b = 0
    while b < m:
        if (i>0) and (i<len(malla)-1) and (b>0) and (b<len(malla[i])-1):
            g1 = (malla[0][b])
            g2 = + (malla [n] [len(malla[i])-1])
            g3 = (g1+g2)/2
            malla[i] [b] = g3
        b += 1
    b = 0
    while b < m:
        if(cilindro.contenido(i,b)):
            malla[i][b] = stream_function(i,b,K)
        b+=1
    i+= 1


b=0
i=0
while b < len(malla[0]):
    malla[0][i] = L *Vinf
    malla[n][b] = 0
```

Start Page | NACA.py | airfoil+NACA.py | cilindro_simple.py | airfoil_naca.py

```python

###metode numeric:

i = 1
b = 0
valor = 999999999

i = n
j=0
@jit(parallel=True)
def laplace():
    for i in range(iteracion):
        print('empezando iteracion: '+ str(i))
        while i>=0:
            j=0
            while j< len(malla):
                if(cilindro.contenido(i,j)):
                    malla[i][j]= stream_function(i,j,K)
                else:
                    if((j!=0) and (j!=m)) and((i>0) and (i<n)):
                        valor = ((dy**2)*(malla[i+1][j] + malla[i-1][j]) + (dt**2) * (malla[i][j+1]+ malla [i][j-1])) /(2*(dt**2 +dy**2))
                        malla[i][j]=valor
                j +=1
            i-=1

laplace()
print('terminated')


F=[]

i = len(malla)-1
#print('\n','*'*150,'\n')
while i >=0:
    F.append(malla[i])
    i -=1
cp=pyplot.contourf(F)
pyplot.colorbar()
pyplot.xlabel('x')
pyplot.ylabel('y')
pyplot.title('iteration n° ' + str(iteracion))
circle = pyplot.Circle((50,50),10,facecolor="w",edgecolor="k")
pyplot.axes().add_artist(circle)
pyplot.axis('equal')
pyplot.show()

print(cilindro.X,cilindro.Y)

pyplot.plot(cilindro.vx, cilindro.vy)
#print(cilindro.vx, cilindro.vy)
pyplot.grid()
pyplot.axis('equal')
pyplot.show()



'''
Y,X = np.mgrid[-10:10:.01, -10:10:.01]
U, V = Y**2, X**2
print(V)
```

File Edit View Source Project Addins About

Start Page | NACA.py | airfoil+NACA.py | cilindro_simple.py | airfoil_naca.py

```python

i = n
j=0
u,v = np.zeros((100,100)),np.zeros((100, 100))
while i>=0:
    j=0
    while j< len(u):
        if((j!=0) and (j!=m)) and((i>0) and (i<n)):
            valor = ((malla[i+1][j]-malla[i-1][j])) /(2*dt)
            u[i][j]=-valor
        j +=1
    i-=1

i = n
j=0
v = np.zeros((100, 100))
while i>=0:
    j=0
    while j< len(v):
        if((j!=0) and (j!=m)) and((i>0) and (i<n)):
            valor = ((malla[i][j+1]-malla[i][j-1])) /(2*dy)
            v[i][j]=valor
        j +=1
    i-=1



nx, ny = 64, 64
x = np.linspace(0, n, n)
y = np.linspace(0, m, m)
#X, Y = np.meshgrid(x, y)
#Ex, Ey = np.zeros((ny, nx)), np.zeros((ny, nx))

pyplot.streamplot(x,y,u, v )
circle = pyplot.Circle((50,50),10,facecolor="r",edgecolor="k")
pyplot.axes().add_artist(circle)
pyplot.axis('equal')
pyplot.show()

VV=malla
i = n
j=0
i = 0

while i<len(VV)-1:
        j=0
        while j< len(u):
            VV[i][j]= M.sqrt(u[i][j]**2+u[i][j]**2)
            j +=1
        #print(VV)
        i+=1

P=malla
i = n
j=0
while i>=0:
    j=0
    while j< len(malla):
        if((j!=0) and (j!=m)) and((i>0) and (i<n)):
            valor = (((Vinf**2)-(VV[i][j]**2))/2)*ro
```

File Edit View Source Project Addins About

Start Page | NACA.py | airfoil+NACA.py | cilindro_simple.py | airfoil_naca.py

```python
nx, ny = 64, 64
x = np.linspace(0, n, n)
y = np.linspace(0, m, m)
#X, Y = np.meshgrid(x, y)
#Ex, Ey = np.zeros((ny, nx)), np.zeros((ny, nx))

pyplot.streamplot(x,y,u, v )
circle = pyplot.Circle((50,50),10,facecolor="r",edgecolor="k")
pyplot.axes().add_artist(circle)
pyplot.axis('equal')
pyplot.show()

VV=malla
i = n
j=0
i = 0

while i<len(VV)-1:
        j=0
        while j< len(u):
            VV[i][j]= M.sqrt(u[i][j]**2+u[i][j]**2)
            j +=1
        #print(VV)
        i+=1

P=malla
i = n
j=0
while i>=0:
    j=0
    while j< len(malla):
        if((j!=0) and (j!=m)) and((i>0) and (i<n)):
            valor = (((Vinf**2)-(VV[i][j]**2))/2)*ro
            P[i][j]=valor
        j +=1
    i-=1

pyplot.contourf(P)
pyplot.colorbar()
pyplot.xlabel('x')
pyplot.ylabel('y')
circle = pyplot.Circle((50,50),10,facecolor="w",edgecolor="k")
pyplot.axes().add_artist(circle)
pyplot.axis('equal')
pyplot.streamplot(x,y,u, v )
pyplot.show()

V = []
for i in range(filas):
    vector = VECTOR()
    V.append([vector]*columnas)
i = n
j=0
while i>=0:
    j=0
    while j< len(u):
        if((j!=0) and (j!=m)) and((i>0) and (i<n)):
            V[i][j].x,V[i][j].y=u[i][j],v[i][j]
        j +=1
    i-=1
```

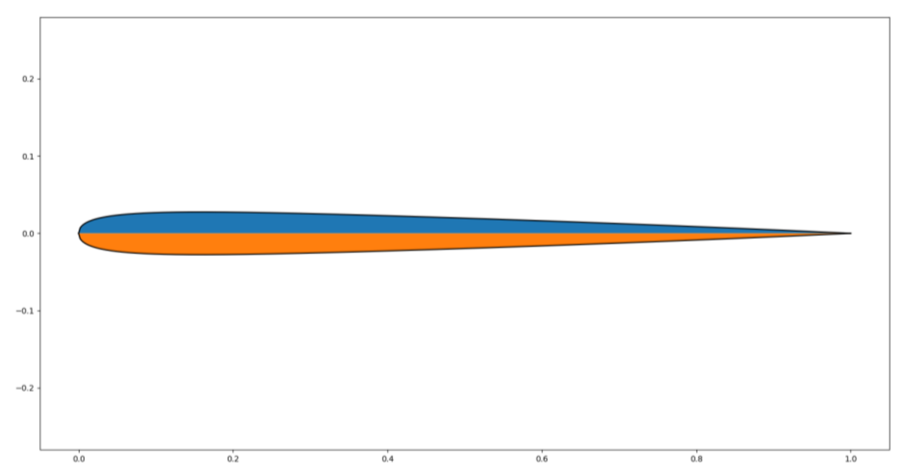
NACA CLASS & IMG RECONAISSANCE

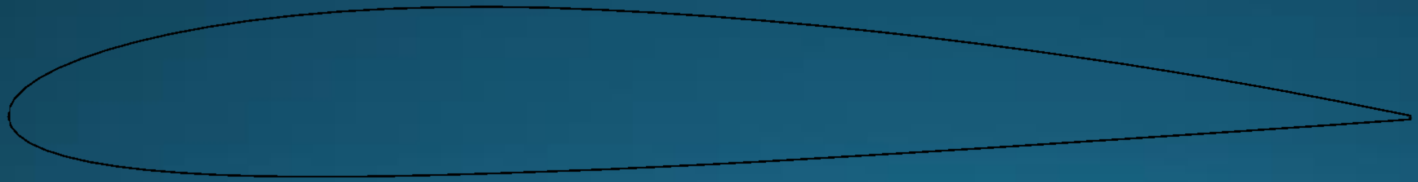
A BIT OF THE OBJECT:

- 118ish lines:
- Main methods:
  - Contenido.
  - Dame_vectores.
  - Escala.
  - Ajustar posicion.
  - Crea_poligono.

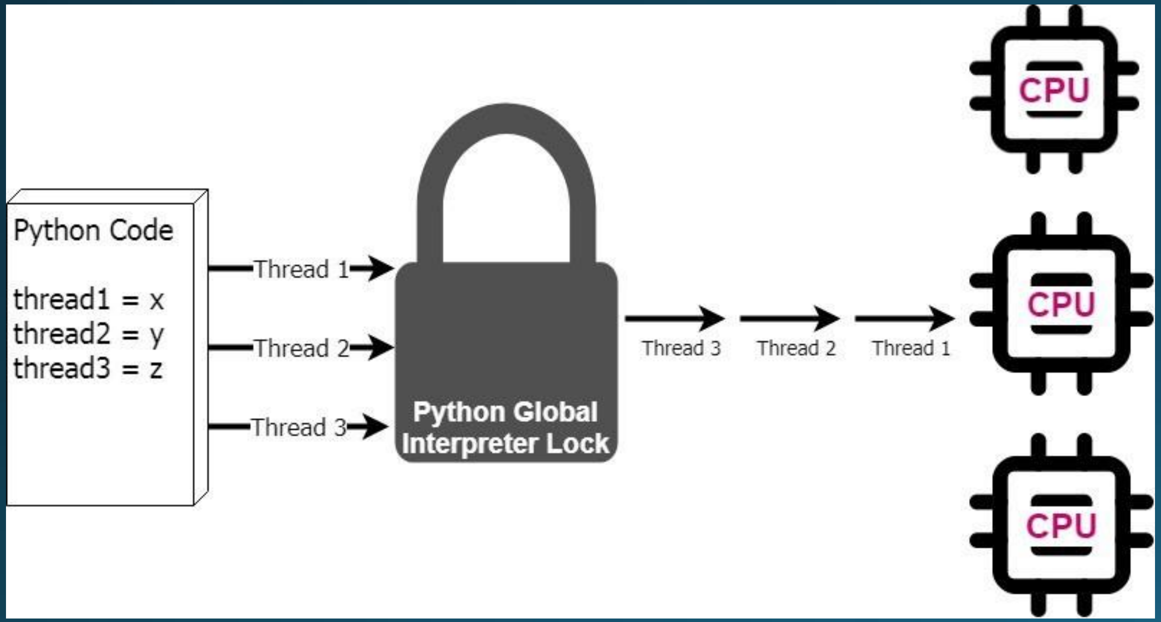
Image reconnaissance:

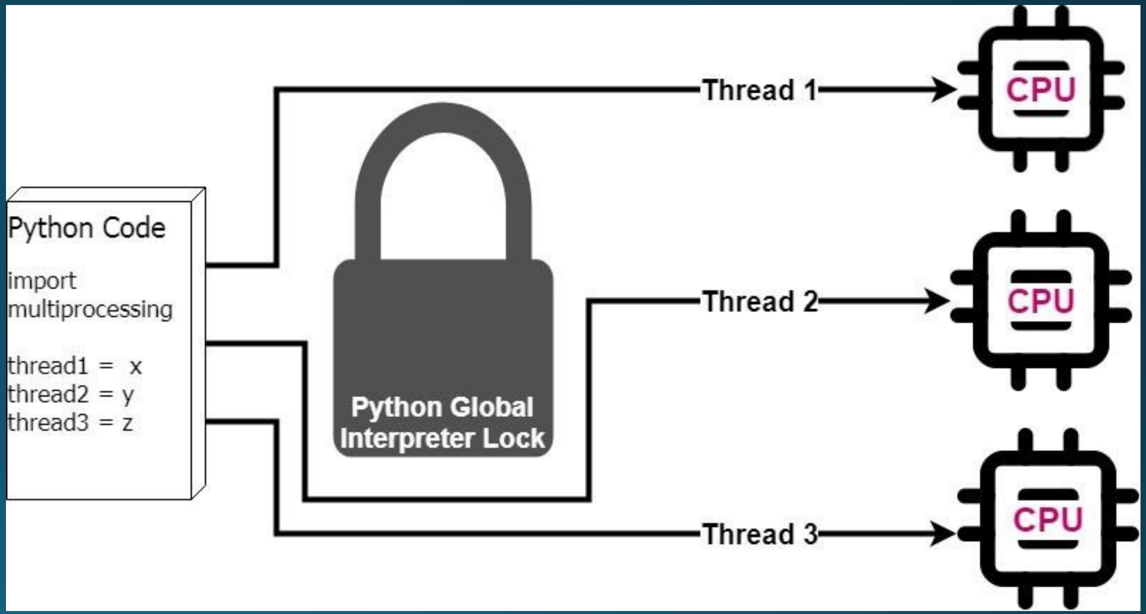
OpenCV



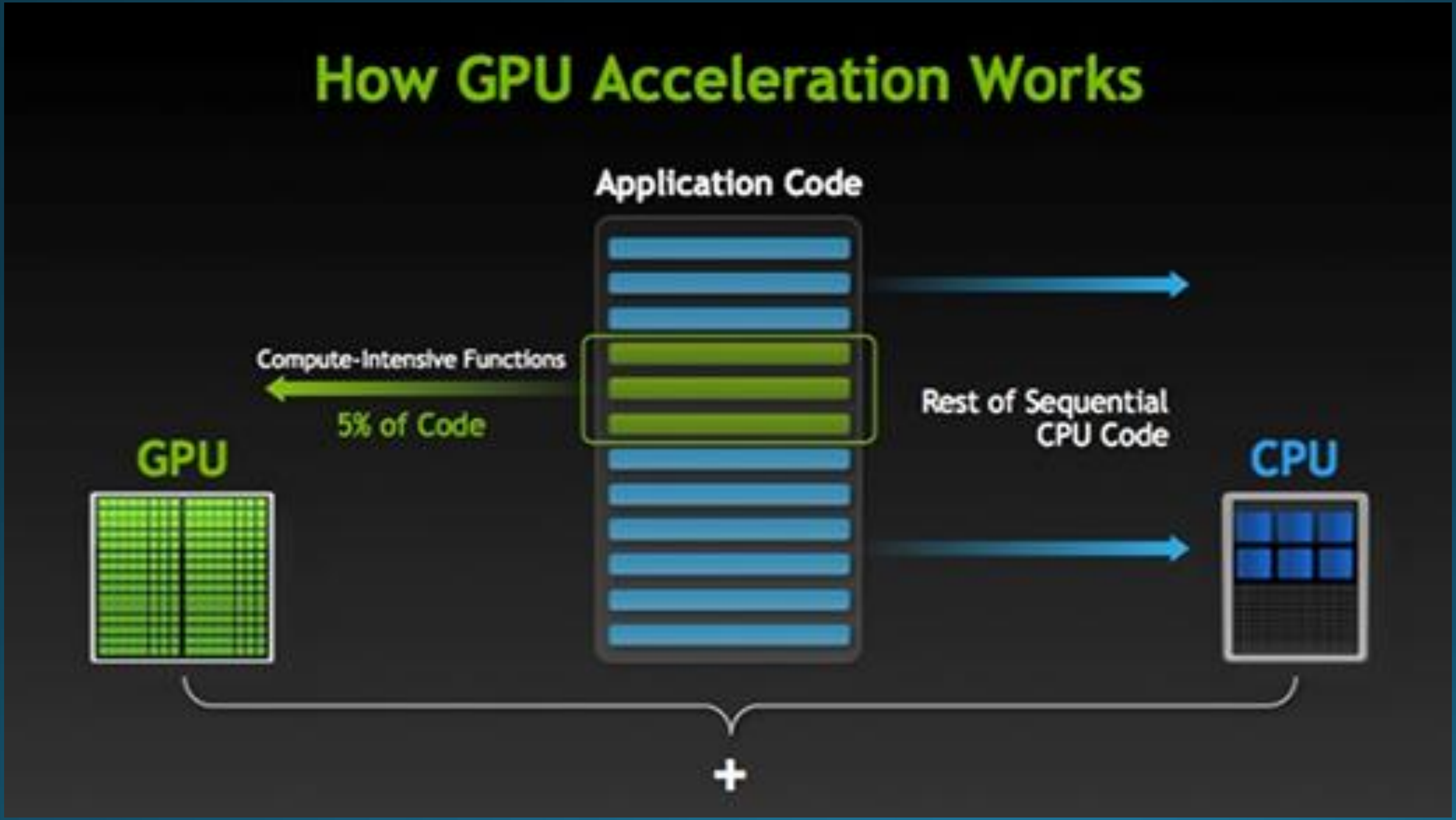


Python performance:



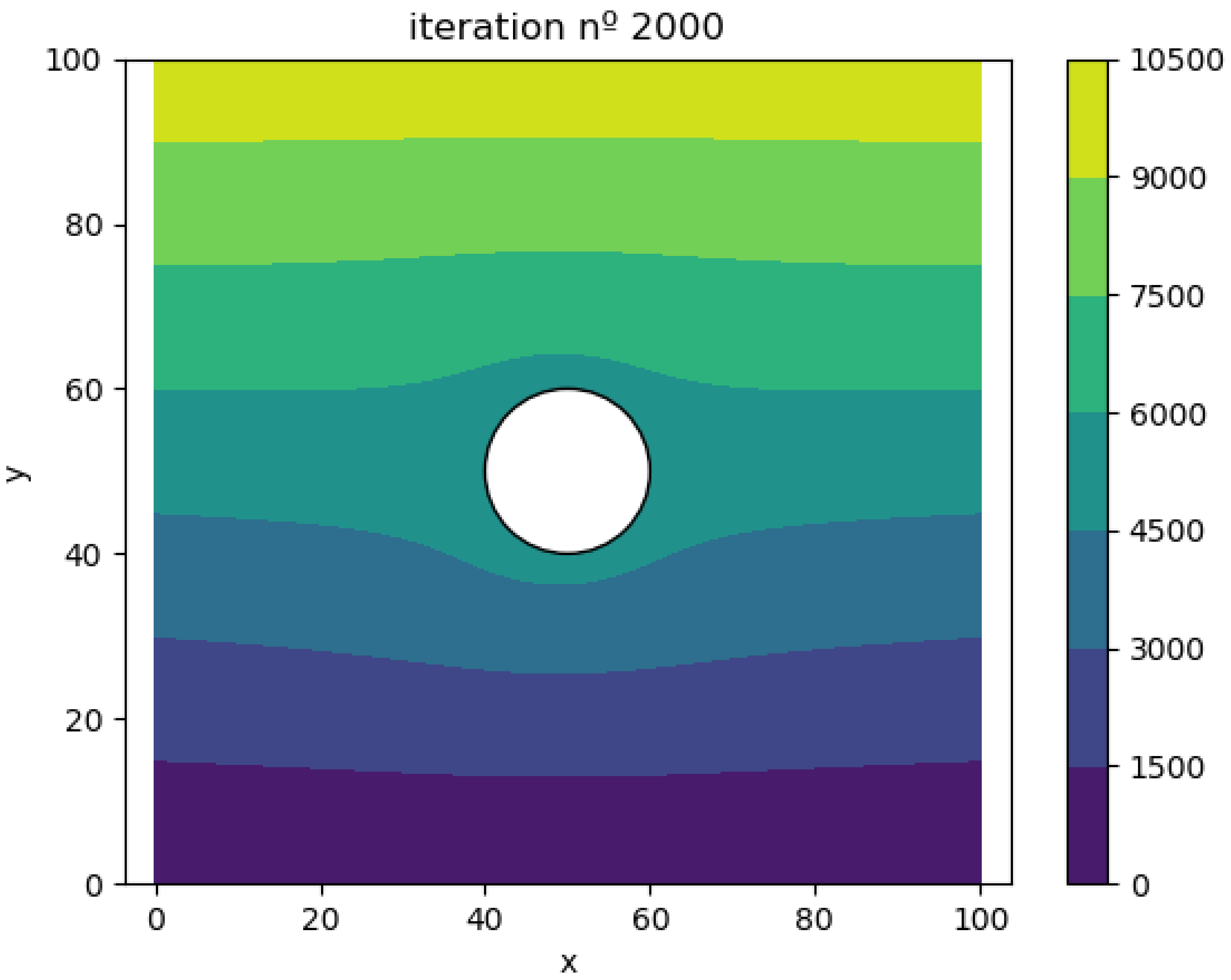


CPU vs GPU


How GPU Acceleration Works
Application Code
Compute-Intensive Functions
5% of Code
GPU
Rest of Sequential
CPU Code
CPU
+

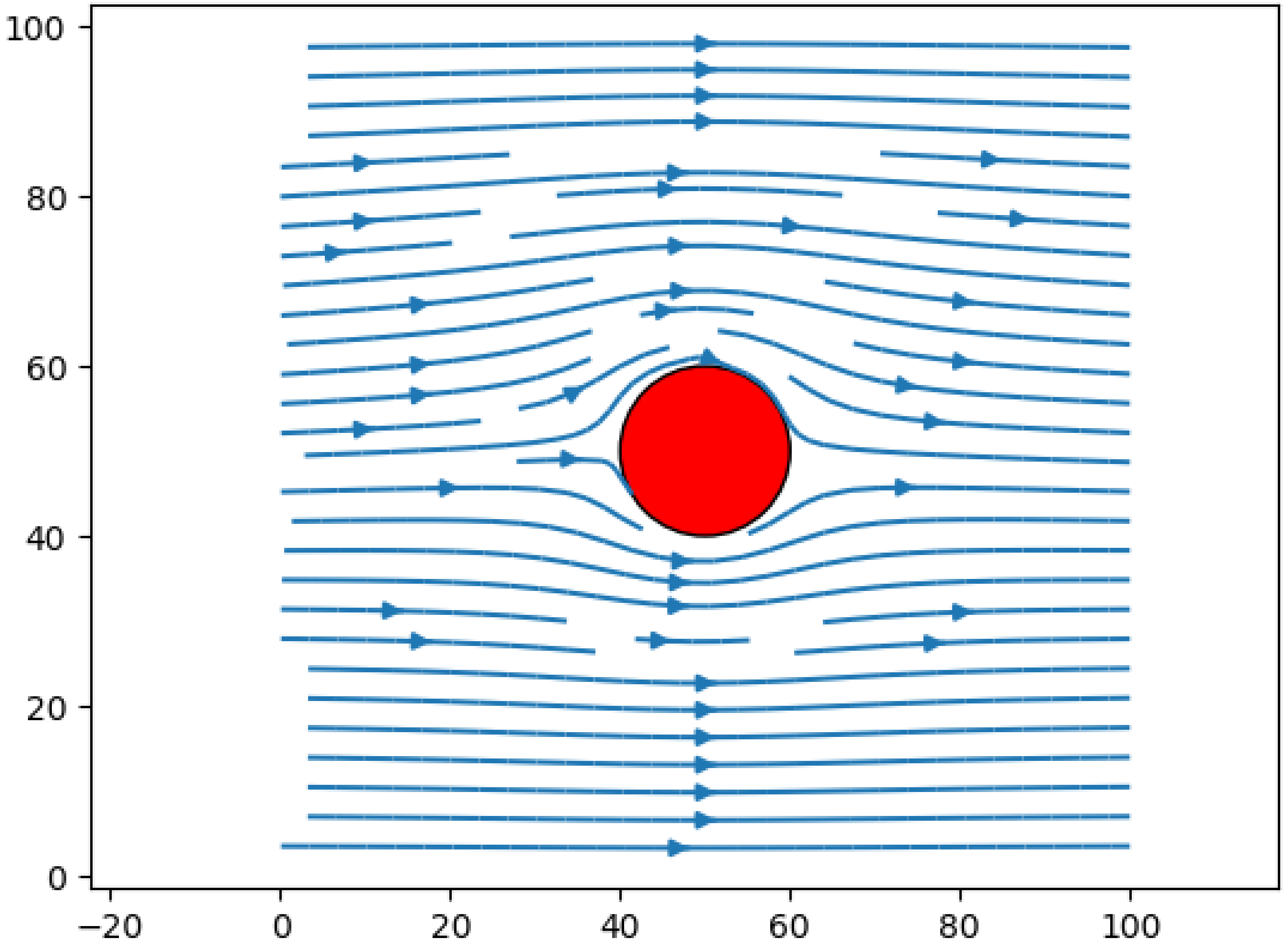
cylinder

Results

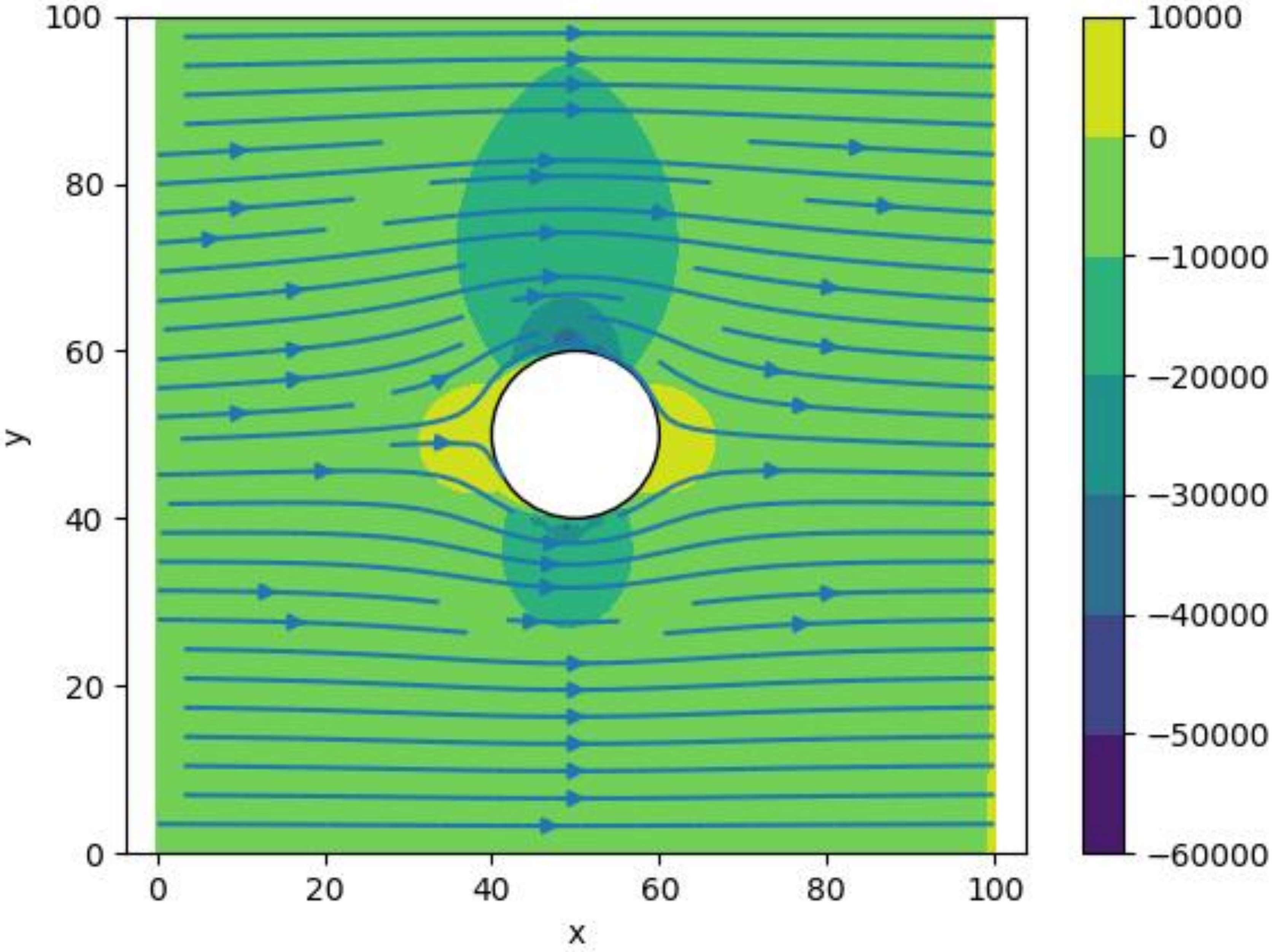


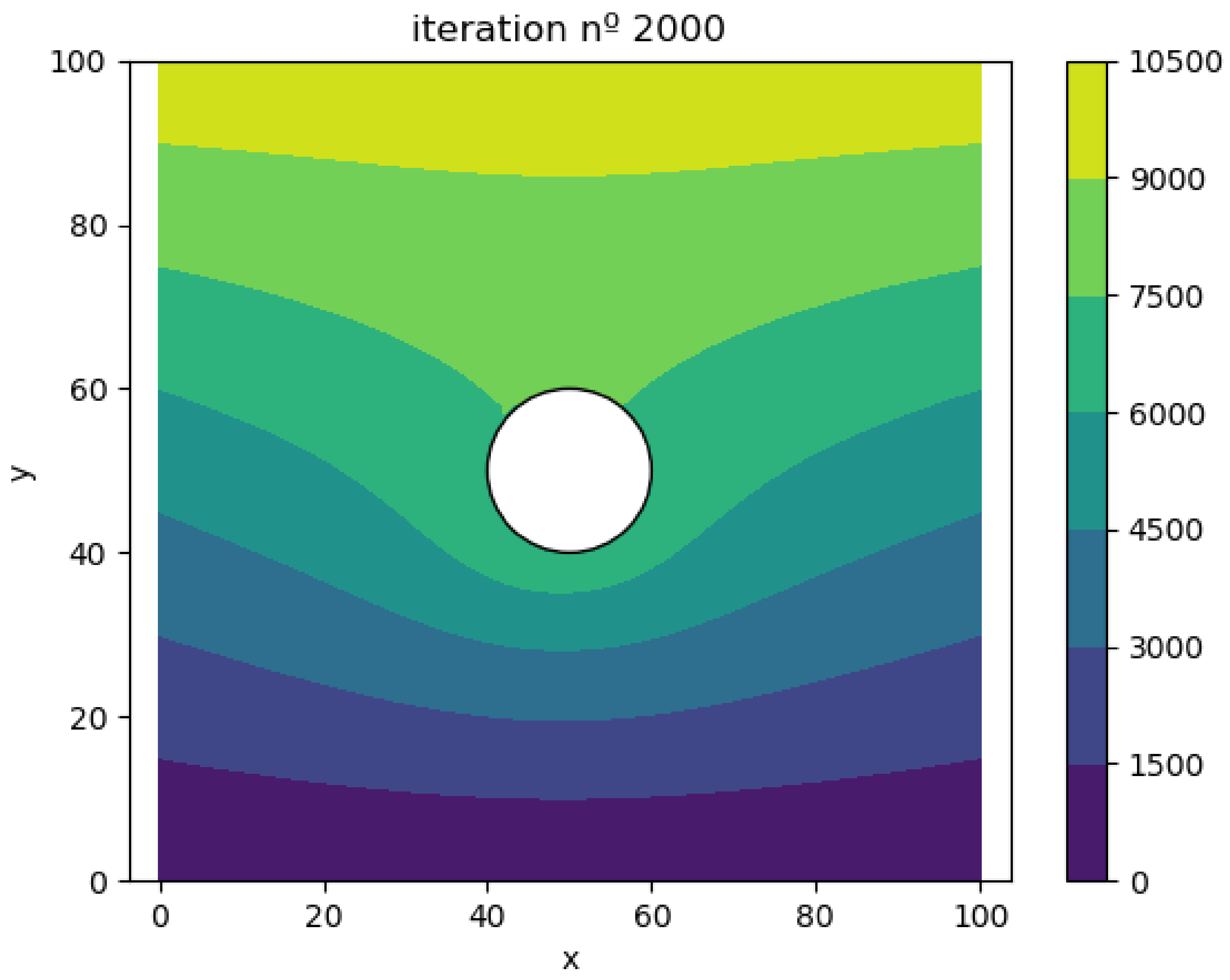
iteration nº 2000
100
80
60
40
20
0
y
0
20
40
60
80
100
x
10500
9000
7500
6000
4500
3000
1500
0

Velocity field:

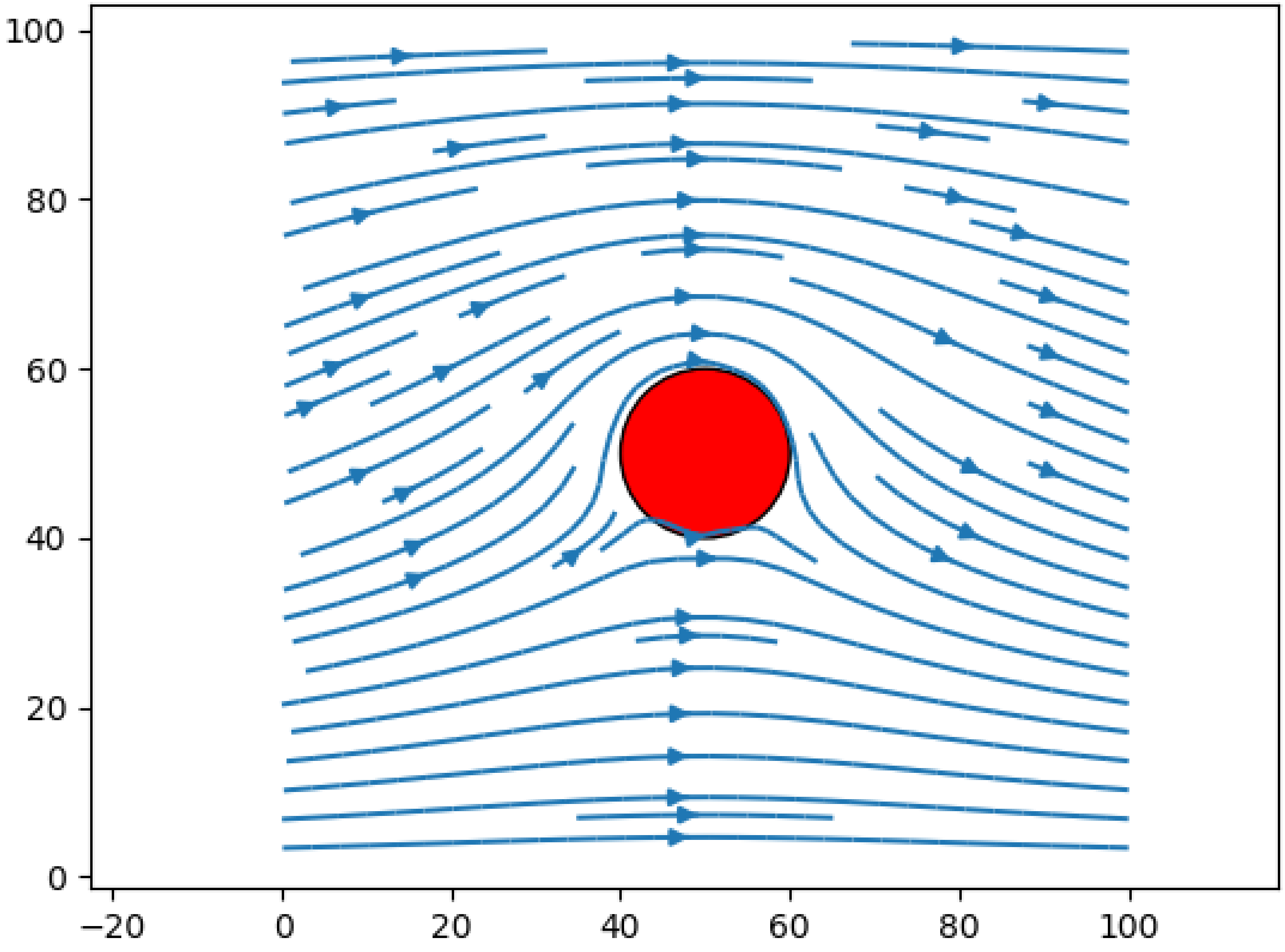


Pressure distribution:

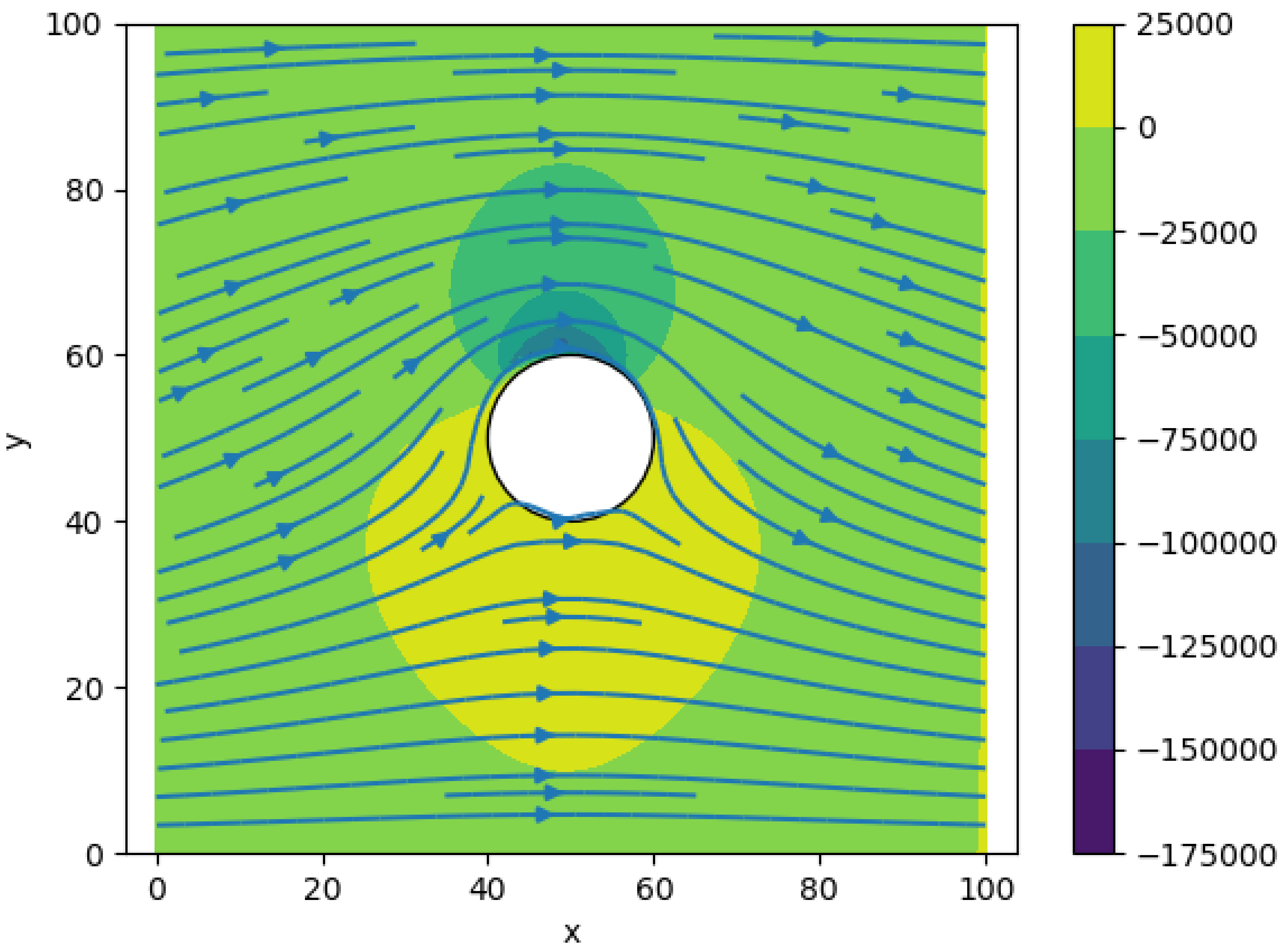


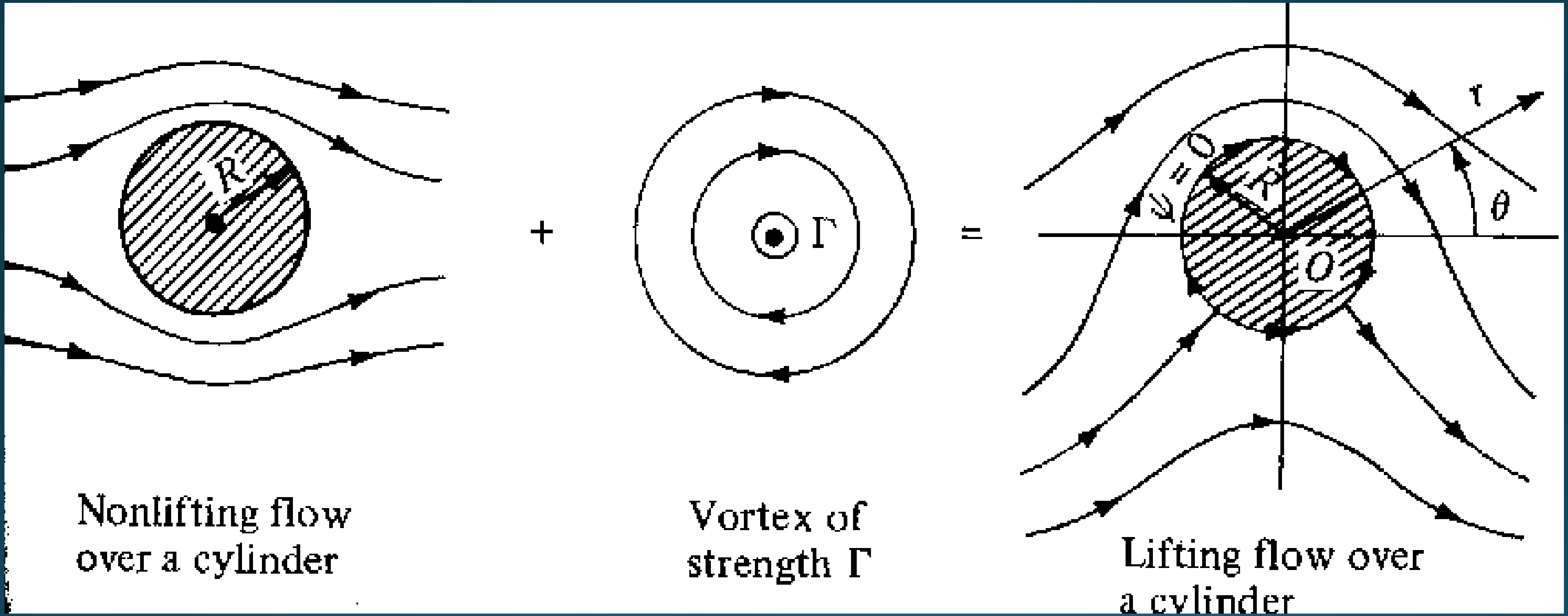

iteration nº 2000
y
x
0
20
40
60
80
100
0
1500
3000
4500
6000
7500
9000
10500

Velocity field:



Pressure distribution:

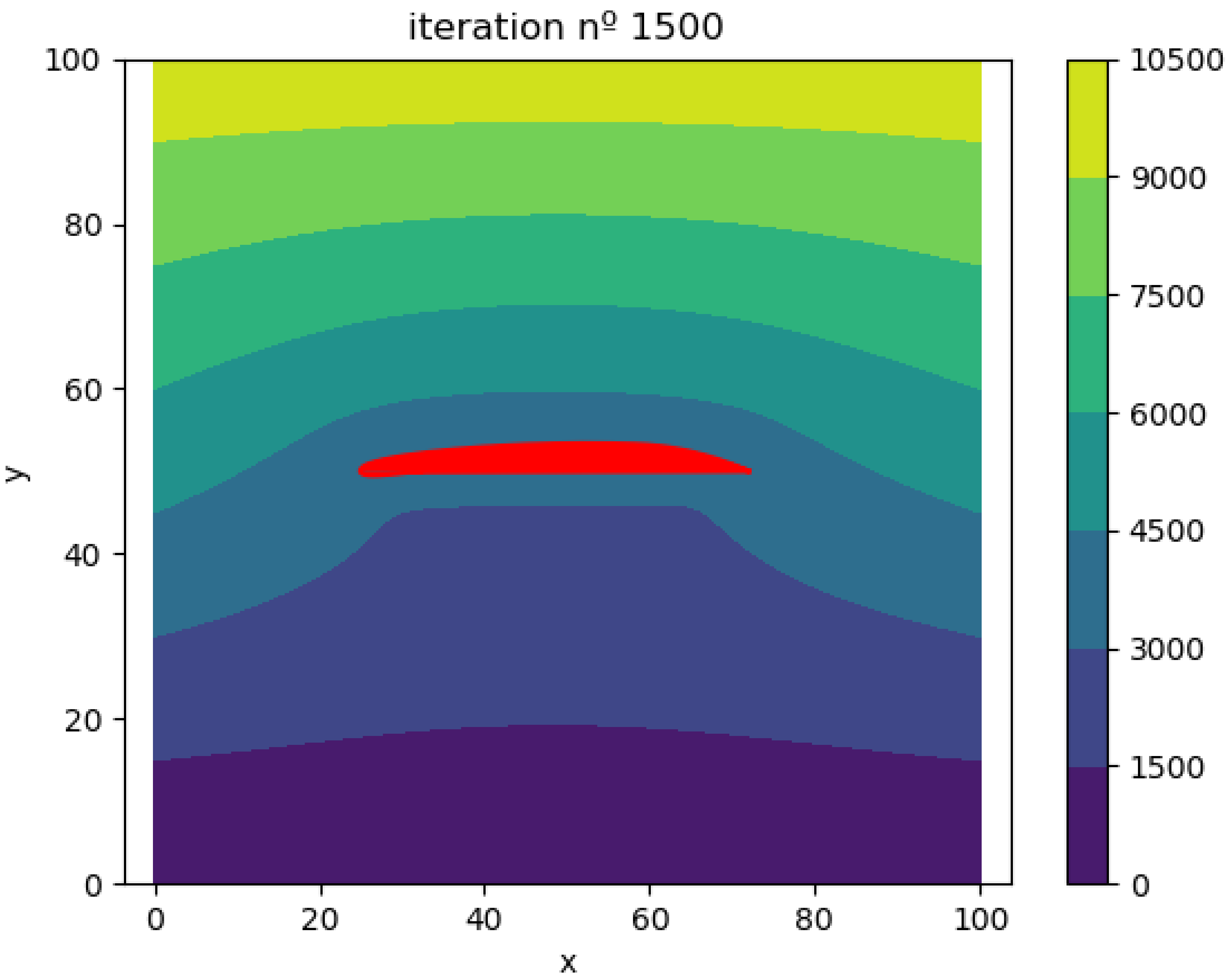


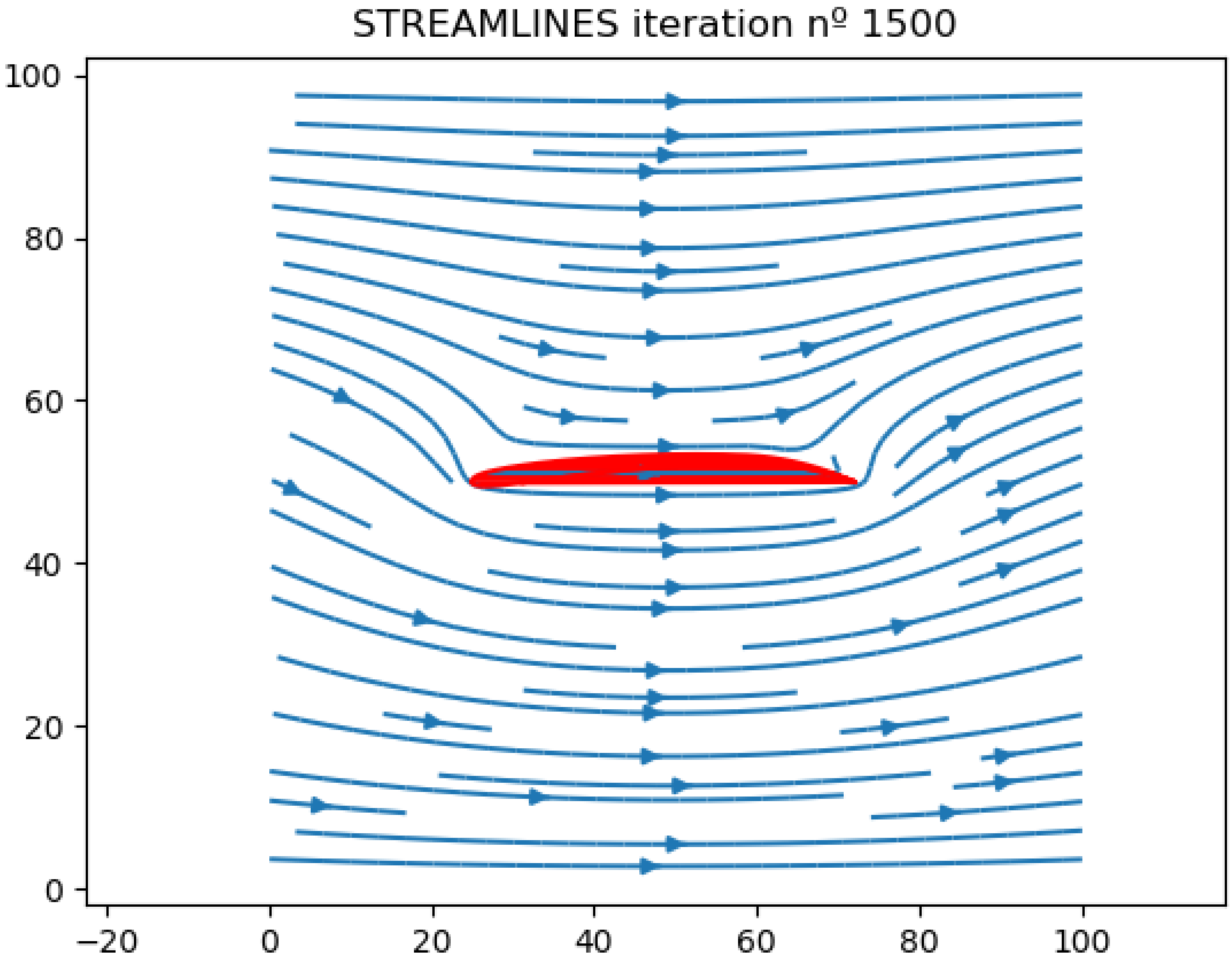


R
+
Γ
=
ψ = 0
R
r
θ
O
Nonlifting flow over a cylinder
Vortex of strength Γ
Lifting flow over a cylinder

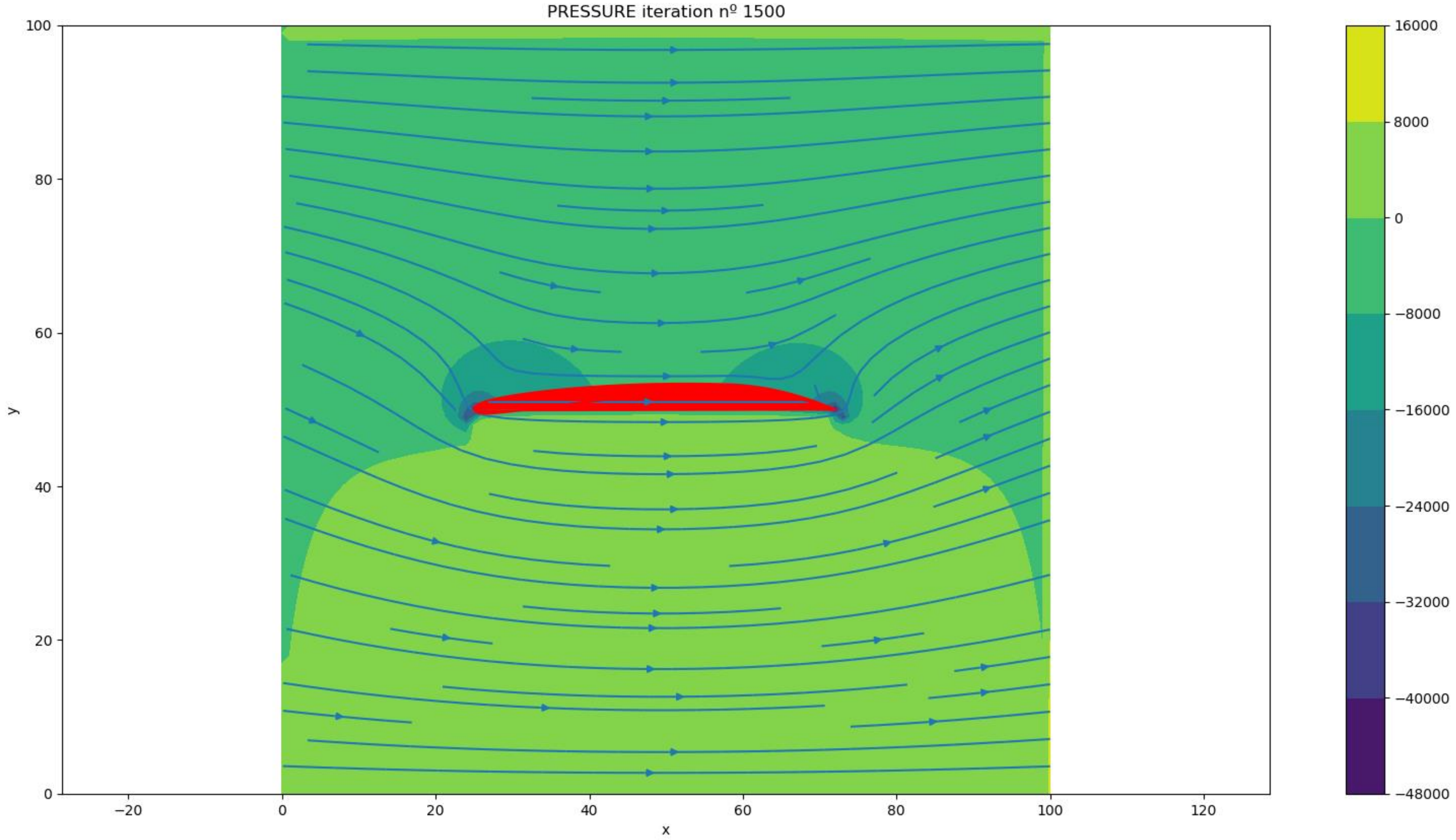
Airfoil

results

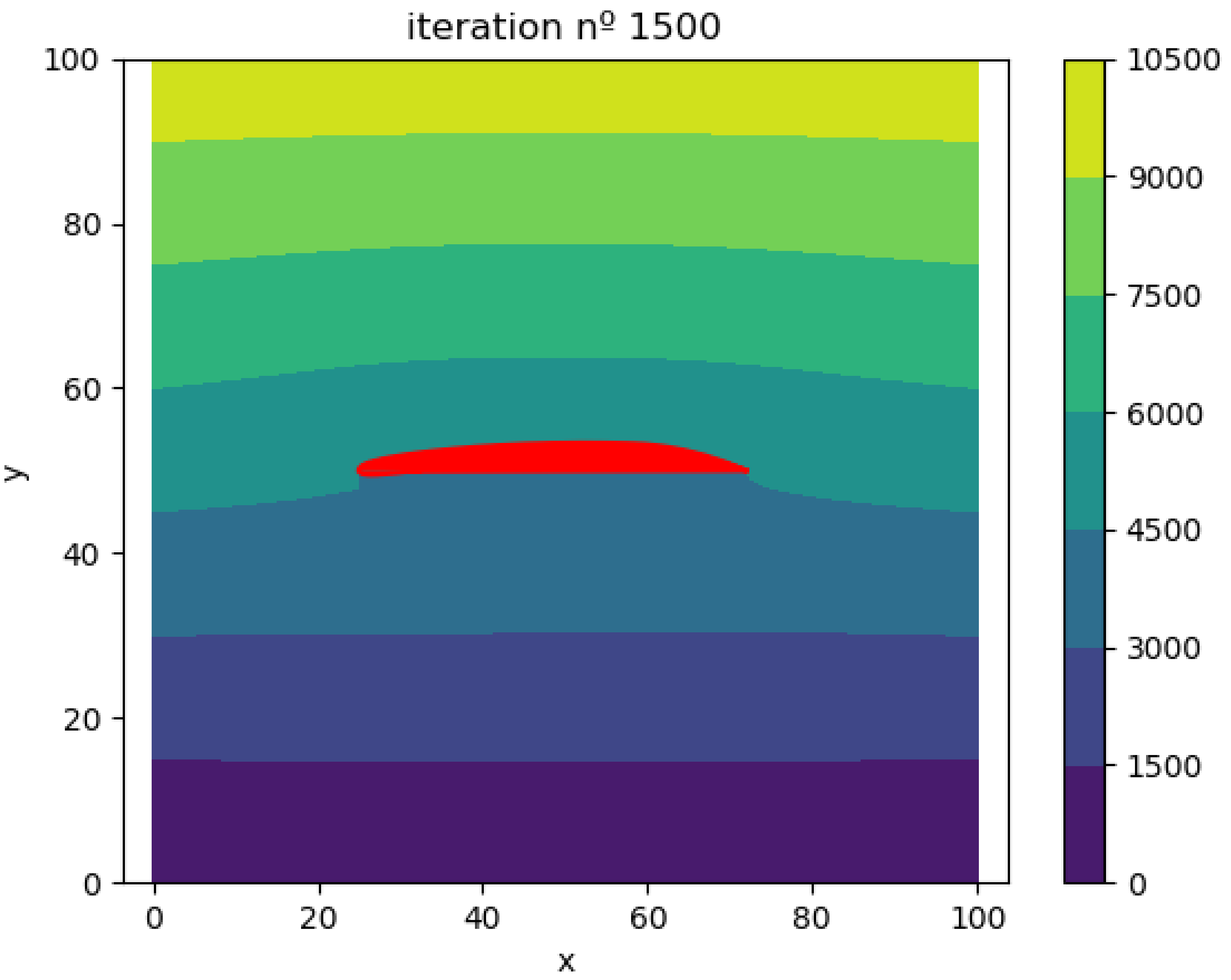

iteration nº 1500
100
80
60
40
20
0
y
0
20
40
60
80
100
x
10500
9000
7500
6000
4500
3000
1500
0

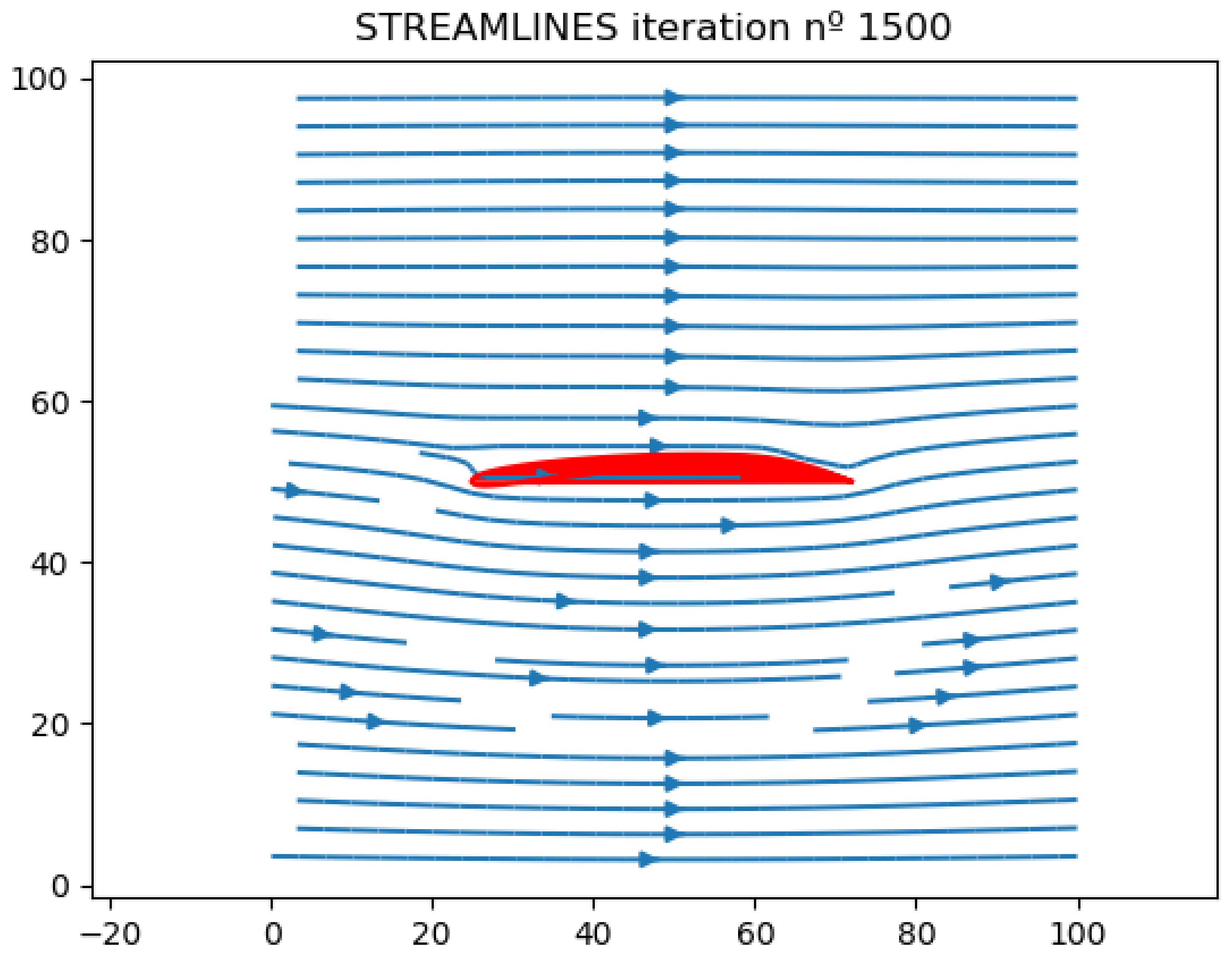


STREAMLINES iteration nº 1500
100
80
60
40
20
0
−20
0
20
40
60
80
100

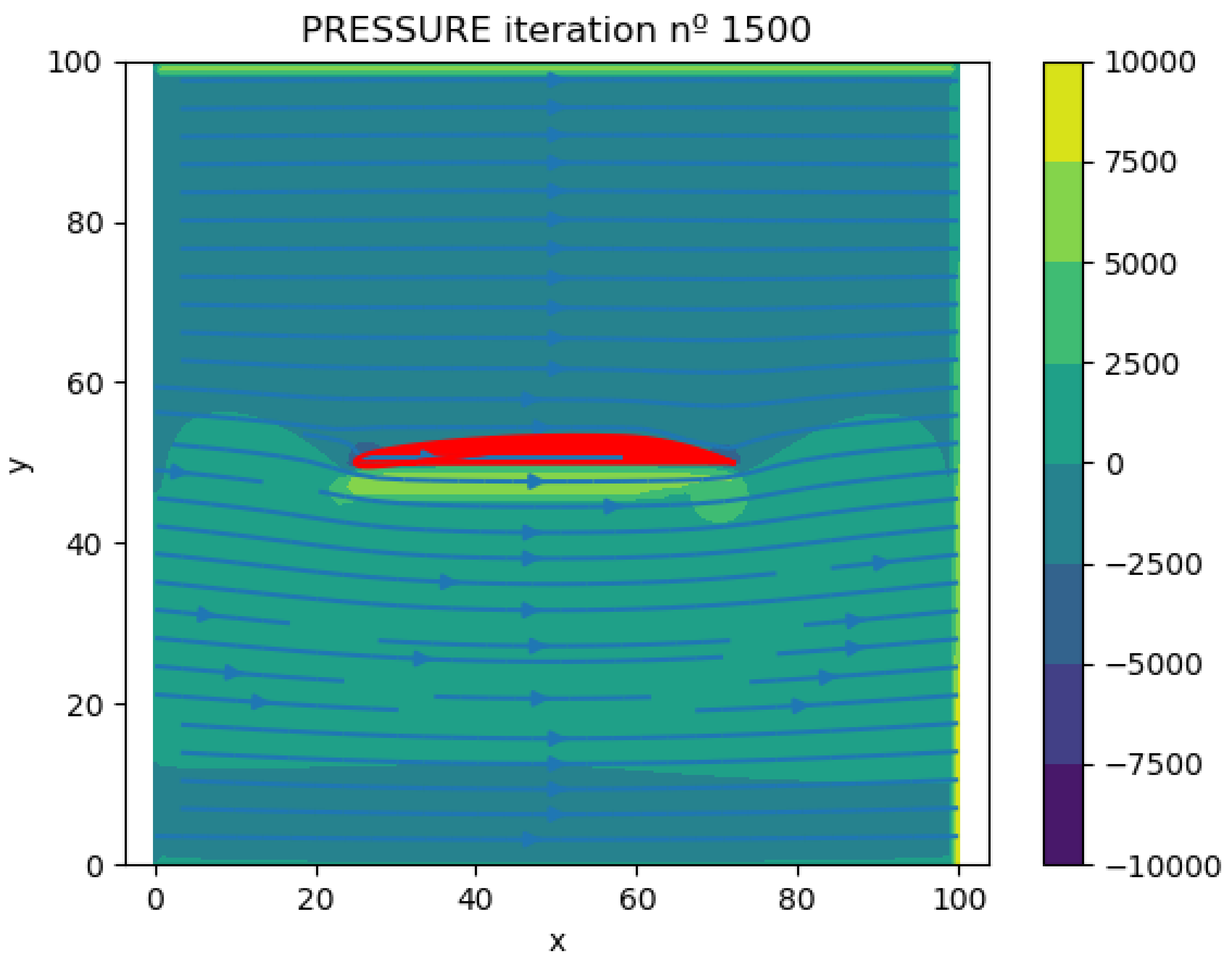


PRESSURE iteration nº 1500
100
80
60
40
20
0
y
−20
0
20
40
60
80
100
120
x
16000
8000
0
−8000
−16000
−24000
−32000
−40000
−48000


iteration nº 1500
100
80
60
40
20
0
y
0
20
40
60
80
100
x
10500
9000
7500
6000
4500
3000
1500
0



STREAMLINES iteration nº 1500
100
80
60
40
20
0
−20
0
20
40
60
80
100



PRESSURE iteration nº 1500
y
x
0
20
40
60
80
100
10000
7500
5000
2500
0
−2500
−5000
−7500
−10000

Conclusion & further improvements

Performance:

- Cylinder ~ 2.5min
- Airfoil 10.5h
- Other mathematical solution?
  - Implicit method
  - Thomas's algorithm
- More cuda technology?

Thank you for your attention!

Questions?