```python
"""
Aluno: Thales Carl Lavoratti (151000656)
Código do problema do cilindro girante com o cilindro parado
"""

import math as mt
import numpy as np
import matplotlib.pyplot as plt
import auxiliar as aux


##############################
#  Setting input paramethers
##############################
yNumberOfNodes = 6
xNumberOfNodes = 6
w = 1.0 #[m]
re = 0.1 #[m]
ri = 0.04 #[m]
L =  mt.sin(0.25*mt.pi)*(re-ri)#[m]
k = 15.0#[W/mºC]
Ti = 250.0 #[ºC]
Te = 30.0 #[ºC]


#############################
# Mesh generation in y
#############################

yNodesPositions = []
deltaY = L/(yNumberOfNodes - 1);
ySum = mt.sin(0.25*mt.pi)*ri;
for i in range(yNumberOfNodes):
    yNodesPositions.append(ySum)
    ySum += deltaY

#############################
# Mesh generation in x
#############################

xNodesPositions = []
deltaX = L/(float(xNumberOfNodes) - 1);
xSum = mt.cos(0.25*mt.pi)*ri;
for i in range(xNumberOfNodes):
    xNodesPositions.append(xSum)
    xSum += deltaX


###############################################################
# Generating the matrix of coefficients and independent vector
###############################################################
A = []
b = []
numberOfNodes = xNumberOfNodes * yNumberOfNodes

for i in range(numberOfNodes):
    A.append([])
    for j in range(numberOfNodes):
        A[i].append(0.0)
    b.append(0.0)
#bottom
for j in range(xNumberOfNodes):
    ap = 1.0
    A[j][j] = ap
    b[j] = aux.analyticSolution(xNodesPositions[j],yNodesPositions[0])

#center
for i in range(1,yNumberOfNodes-1):
```

```
    for j in range(xNumberOfNodes):
        if  j == 0 or j == (xNumberOfNodes - 1):
            ap = 1.0
            A[i*xNumberOfNodes+j][i*xNumberOfNodes+j] = ap
            b[i*xNumberOfNodes+j] = aux.analyticSolution(xNodesPositions[j],yNodesPositions[i])
        else:
            aEast  =  k*w*deltaY/deltaX
            aWest  =  k*w*deltaY/deltaX
            aSouth =  k*w*deltaX/deltaY
            aNorth =  k*w*deltaX/deltaY
            ap = aEast + aWest + aSouth + aNorth
            A[i*xNumberOfNodes+j][j+xNumberOfNodes*(i-1)] = -aSouth
            A[i*xNumberOfNodes+j][i*xNumberOfNodes+j-1] = -aWest
            A[i*xNumberOfNodes+j][i*xNumberOfNodes+j] = ap
            A[i*xNumberOfNodes+j][i*xNumberOfNodes+j+1] = -aEast
            A[i*xNumberOfNodes+j][j+xNumberOfNodes*(i+1)] = -aNorth
#top
for j in range(xNumberOfNodes):
    ap = 1.0
    A[(yNumberOfNodes-1)*xNumberOfNodes+j][(yNumberOfNodes-1)*xNumberOfNodes+j] = ap
    b[(yNumberOfNodes-1)*xNumberOfNodes+j] = aux.analyticSolution(xNodesPositions[j],yNodesPositi

#############################
# Solving the linear system
############################
solution = np.linalg.solve(np.array(A),np.array(b))
temperatureField = []
for i in range(yNumberOfNodes):
    temperatureField.append([])
    for j in range(xNumberOfNodes):
        temperatureField[i].append(solution[i*xNumberOfNodes+j])

##################################
#Plotting the solution
########################
xx, yy = np.meshgrid(xNodesPositions,yNodesPositions)
plt.contourf(xx,yy,np.array(temperatureField))
plt.colorbar(orientation="vertical")
plt.xlabel("x[m]")

#####################################
# Analytical solution
####################################
anSolution = []
for i in range(yNumberOfNodes):
    anSolution.append([])
    for j in range(xNumberOfNodes):
        anSolution[i].append(aux.analyticSolution(xNodesPositions[i],yNodesPositions[j]))

#########################################
# Error with respect to the 1D solution
#########################################
errors = []
for i in range(yNumberOfNodes):
    errors.append([])
    for j in range(xNumberOfNodes):
        exactTemperature = anSolution[i][j]
        aproxTemperature = temperatureField[i][j]
        diff = (exactTemperature - aproxTemperature)/(Ti - Te)
        errors[i].append(abs(diff))

errors = np.array(errors)
maximumError = errors.max()

import csv

with open("./results/anSolution.csv","w") as output:
    writer = csv.writer(output,lineterminator='\n')
```

```python
    for i in range(len(anSolution)):
        outputVector = ['{:.4f}'.format(x) for x in anSolution[i]]
        writer.writerow(outputVector)

with open("./results/temperatureField.csv","w") as output:
    writer = csv.writer(output,lineterminator='\n')
    for i in range(len(temperatureField)):
        outputVector = ['{:.4f}'.format(x) for x in temperatureField[i]]
        writer.writerow(outputVector)

with open("./results/errors.csv","w") as output:
    writer = csv.writer(output,lineterminator='\n')
    for i in range(len(errors)):
        outputVector = ['{:.3e}'.format(x) for x in errors[i]]
        writer.writerow(outputVector)

with open("./results/xNodesPositions.csv","w") as output:
    writer = csv.writer(output,lineterminator='\n')
    outputVector = ['{:.4f}'.format(x) for x in xNodesPositions]
    writer.writerow(outputVector)
```