PMR 3401/2020
Mecânica Computacional para Mecatrônica

2º Exercício Programa

ESCOLA POLITÉCNICA

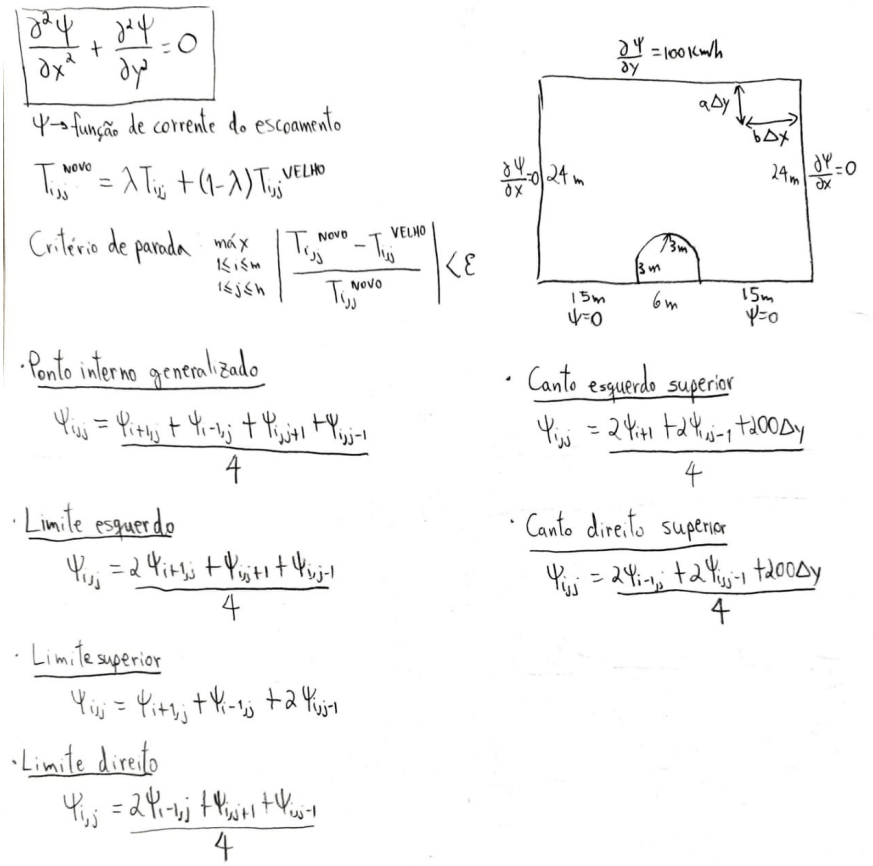
Autor:
Fabiano Wang Tzuu NºUSP: 10333571
Gustavo Marangoni Rubo NºUSP: 4584080

PARTE I

a) Para plotar a função de corrente do escoamento ψ, a equação (6) foi analisada com base na Figura 3 que contém as condições de contorno.

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

A partir da Figura 3 do enunciado, foram analisados os contornos do limite esquerdo, limite direito, canto esquerdo superior, canto direito superior e limite superior para as bordas da figura. Como a borda inferior possui ψ nula, ela não precisava ser analisada. Além disso, as bordas do galpão também foram examinadas, de forma que a análise foi dividida em borda do galpão esquerdo, borda do galpão direito e borda do teto do galpão. No caso do teto do galpão, a sua análise foi dividida em cinco partes. As duas primeiras referem-se à parte esquerda do galpão, onde o ponto pode possuir borda irregular tanto na direção x como na direção y ou apenas na direção x. As duas outras referem-se aos mesmos casos, mas da parte direita do teto. O último caso refere-se ao teto do galpão, onde existe borda irregular na direção y apenas. Para os outros pontos (pontos generalizados), foi feita a análise padrão. Por fim, a imagem a seguir mostra a conclusão das análises feitas:



Galpão

$$a = \frac{21 - i\Delta y - \sqrt{3^2 - (j\Delta x - 18)^2}}{\Delta y}$$

$$b = \frac{|18 - j\Delta x| - 3 \cdot \cos\left(\text{arcsen}\left(\frac{21 - i\Delta y}{3}\right)\right)}{\Delta x}$$

- Borda esquerda

$$\psi_{i,j} = \frac{b}{b+1}\left(\frac{\psi_{i-1,j}}{b+1} + \frac{\psi_{i,j+1}}{2} + \frac{\psi_{i,j-1}}{2}\right)$$

- Borda direita

$$\psi_{i,j} = \frac{b}{b+1}\left(\frac{\psi_{i+1,j}}{b+1} + \frac{\psi_{i,j+1}}{2} + \frac{\psi_{i,j-1}}{2}\right)$$

- Teto esquerdo inferior

$$\psi_{i,j} = \frac{b\,\psi_{i,j+1}}{2(b+1)} + \frac{b\,\psi_{i,j-1}}{2(b+1)} + \frac{b\,\psi_{i-1,j}}{(b+1)^2}$$

- Teto direito inferior

$$\psi_{i,j} = \frac{b\,\psi_{i,j+1}}{2(b+1)} + \frac{b\,\psi_{i,j-1}}{2(b+1)} + \frac{b\,\psi_{i+1,j}}{(b+1)^2}$$

- Teto esquerdo intermediário

$$\psi_{i,j} = \frac{ab}{a+b}\left(\frac{\psi_{i-1,j}}{b+1} + \frac{\psi_{i,j+1}}{a+1}\right)$$

- Teto direito intermediário

$$\psi_{i,j} = \frac{ab}{a+b}\left(\frac{\psi_{i+1,j}}{b+1} + \frac{\psi_{i,j+1}}{a+1}\right)$$

- Teto superior

$$\psi_{i,j} = \frac{a\,\psi_{i+1,j}}{2(a+1)} + \frac{a\,\psi_{i-1,j}}{2(a+1)} + \frac{a\,\psi_{i,j+1}}{(a+1)^2}$$

Todas as análises anteriores foram passadas em forma de código em Python. A seguir, está o código. É importante observar que os índices i e j da análise anterior se diferem dos índices i e j do código devido ao ponto de referência utilizado. Aqui, também foi utilizado um $\Delta y = \Delta x = 0.5$, escolhido arbitrariamente.

CÓDIGO:

```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d
from matplotlib import cm


V = 100 / 3.6 #Velocidade do vento em m/s
h = 3 #metros
d = 5 * h
L = 2 * h
H = 8 * h


dx = dy = 0.5


linhas = int(H//dy) + 1
colunas = int((2*d+L) //dx) + 1


matriz = np.zeros(shape = (linhas, colunas))


def main():
  epsilon = 0.01
  erro_max = 1
  contador = 0


  while erro_max > epsilon:
    erro_max = 0
    contador += 1


    for i in range(len(matriz) - 1):
      for j in range(len(matriz[0])):


        altura = 24 - i * dy
        comprimento = j * dx
        modulo = np.sqrt(np.power(18 - j * dx, 2) + np.power(altura - 3, 2))


        #considera corrente dentro do galpão igual a zero
        if comprimento >= 15 and comprimento <= 21 and altura <= 3:
          psi = 0


        elif comprimento >= 15 and comprimento <= 21 and altura >= 3  and altura <= 6 and modulo
<= 3:
          psi = 0


        elif altura <= 3 and comprimento + dx > 15 and comprimento - dx < 21:


          #condicao da borda esquerda do galpão
          if 15 - comprimento < dx and 15 - comprimento > 0:
            b = round(15 % dx, 1)


              psi = (b / (b + 1)) * ((matriz[i][j-1] / (b + 1)) + ((matriz[i-1][j] + matriz[i+1]
[j]) / 2))


          #condicao da borda direita do galpao
          elif comprimento - 21 < dx and comprimento - 21 > 0:
            b = round(21 % dx, 1)


              psi = (b / (b + 1)) * ((matriz[i][j+1] / (b + 1)) + ((matriz[i-1][j] + matriz[i+1]
[j]) / 2))


        #condicao para borda do telhado
        elif comprimento >= 15 and comprimento <= 21 and altura < 6 + dy and altura >= 3:
           if comprimento < 18 and np.sqrt(np.power(18 - (j + 1) * dx, 2) + np.power(21 - i * dy,
2)) < 3 and np.sqrt(np.power(18 - j * dx, 2) + np.power(21 - (i + 1) * dy, 2)) < 3:
            #condicao para borda do teto esquerdo do galpão com uso de a e b
            a = (21 - i * dy - np.sqrt(9 - np.power(j * dx - 18, 2))) / dy
            b = (abs(18 - j * dx) - 3 * np.cos(np.arcsin((21 - i * dy) / 3))) / dx


            psi = ((a * b) / (a + b)) * ((matriz[i][j-1] / (b + 1)) + (matriz[i-1][j] / (a + 1)))


          elif comprimento > 18 and np.sqrt(np.power(18 - (j - 1) * dx, 2) + np.power(21 - i *
dy, 2)) < 3 and np.sqrt(np.power(18 - j * dx, 2) + np.power(21 - (i + 1) * dy, 2)) < 3:
```

```
            #condicao para a borda do teto direito do galpao com uso de a e b
            a = (21 - i * dy - np.sqrt(9 - np.power(j * dx - 18, 2))) / dy
            b = (abs(18 - j * dx) - 3 * np.cos(np.arcsin((21 - i * dy) / 3))) / dx

            psi = ((a * b) / (a + b)) * ((matriz[i][j+1] / (b + 1)) + (matriz[i-1][j] / (a + 1)))

          elif comprimento < 18 and np.sqrt(np.power(18 - (j + 1) * dx, 2) + np.power(21 - i *
dy, 2)) < 3:
            #condicao para a borda do teto esquerdo do galpão com uso de b
            b = abs((abs(18 - j * dx) - 3 * np.cos(np.arcsin((21 - i * dy) / 3))) / dx)

            psi = b * ((matriz[i-1][j] / (2 * (b + 1))) + (matriz[i+1][j] / (2 * (b + 1))) +
              (matriz[i][j-1] / ((b + 1) * (b + 1))))

          elif comprimento > 18 and np.sqrt(np.power(18 - (j - 1) * dx, 2) + np.power(21 - i *
dy, 2)) < 3:
            #condicao para a borda do teto direito do galpão com uso de b
            b = (abs(18 - j * dx) - 3 * np.cos(np.arcsin((21 - i * dy) / 3))) / dx

              psi = b * ((matriz[i-1][j] / (2 * (b + 1))) + (matriz[i+1][j] / (2 * (b + 1))) +
(matriz[i][j+1] / ((b + 1) * (b + 1))))

        elif np.sqrt(np.power(18 - j * dx, 2) + np.power(21 - (i + 1) * dy, 2)) < 3:
          #condicao para a borda do teto na parte superior com uso de a
          a = abs((21 - i * dy - np.sqrt(9 - np.power(j * dx - 18, 2))) / dy)

            psi = a * ((matriz[i][j+1] / (2 * a + 2)) + (matriz[i][j-1] / (2 * a + 2)) +
(matriz[i-1][j] / ((a + 1) * (a + 1))))

         #atualiza os pontos dentro da condicao inicial, mas que não se encaixam em nenhum dos
casos anteriores
        else:
          psi = (matriz[i][j+1] + matriz[i][j-1] + matriz[i-1][j] + matriz[i+1][j]) / 4

      elif (i == 0 or j == 0) and j != (len(matriz[0]) - 1):
        #canto superior esquerdo
        if i == 0 and j == 0:
          psi = (2 * matriz[i][j+1] + 2 * matriz[i+1][j] + 2 * V * dy) / 4

        #limite esquerdo
        elif j == 0:
          psi = (2 * matriz[i][j+1] + matriz[i-1][j] + matriz[i+1][j]) / 4

        #limite superior sem contar canto superior esquerdo
        elif i == 0 and j != (len(matriz[0]) - 1):
          psi = (matriz[i][j+1] + matriz[i][j-1] + 2 * matriz[i+1][j] + 2 * V * dy) / 4



      elif i ==0 or j == (len(matriz[0]) - 1):
        #canto superior direito
        if i == 0 and j == (len(matriz[0]) - 1):
          psi = (2 * matriz[i][j-1] + 2 * matriz[i+1][j] + 2 * V * dy) / 4

        #limite direito
        elif j == (len(matriz[0]) - 1):
          psi = (2 * matriz[i][j-1] + matriz[i-1][j] + matriz[i+1][j]) / 4

      #para pontos i, j generalizados internos
      else:
        psi = (matriz[i][j+1] + matriz[i][j-1] + matriz[i-1][j] + matriz[i+1][j]) / 4

      # Sobrerrelaxação:
      fator = 1.85
      psi_novo = fator*psi + (1-fator)*matriz[i][j]
      erro_atual = abs((psi_novo - matriz[i][j])/psi_novo) if psi_novo != 0 else 0
      erro_max = max(erro_atual, erro_max) #atualiza com maior erro
      matriz[i][j] = psi_novo

  print("Passo:", dx)
  print("Epsilon:", epsilon)
  print("Iterações:", contador)
  print("Média de psi da borda superior:", np.mean(matriz[0]))
```

```
  parte_1a(matriz)

def parte_1a(matriz):
  fig = plt.figure()
  ax = fig.add_subplot(111)
  ax.set_title("Função de corrente do escoamento 2D")
  cax = ax.matshow(matriz, cmap='inferno')
  fig.colorbar(cax)
  plt.show()

  nx = int((2 * d + L)/dx) + 1
  ny = int((H)/dy) + 1
  x = np.linspace(0, 2*d+L, nx)
  y = np.linspace(0, H, ny)
  X, Y = np.meshgrid(x, y)
  fig1 = plt.figure()
  ax1 = fig1.gca(projection='3d')
  ax1.set_title("Função de corrente do escoamento 3D")
  s = ax1.plot_surface(X, Y, matriz, cmap=cm.inferno, linewidth=5, antialiased=True)
  ax1.set_xlabel("x (m)")
  ax1.set_ylabel("y (m)")
  ax1.set_zlabel("Psi (m²/s)")
  plt.show()

main()
```

Ao final, os valores de corrente de escoamento variaram de 0 até cerca de 600. Nota-se que a velocidade V de 100km/h foi convertida para 27.78m/s. A seguir, estão imagens das figuras resultantes:

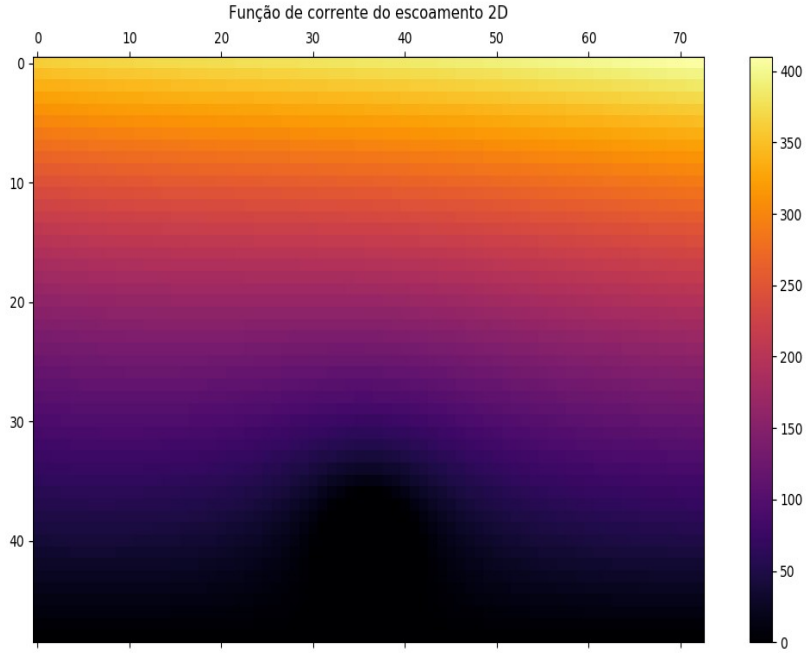


Figura 1: Gráfico 2D da função ψ de corrente do escoamento para passo de 0.5 m. ε = 0.01; Iterações: 101.

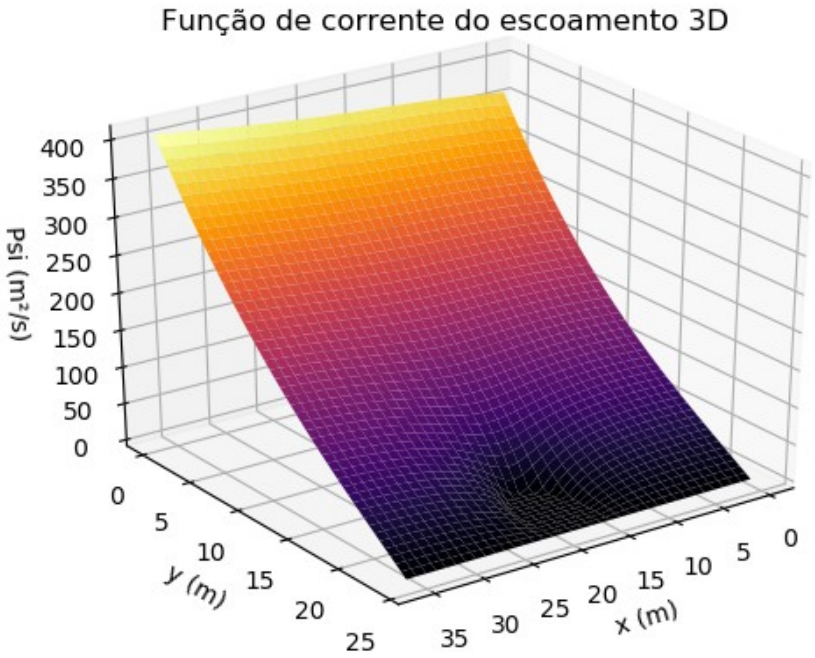


Figura 2: Gráfico 3D da função ψ de corrente do escoamento para passo de 0.5 m ε = 0.01; Iterações: 101.

Para as análises da 2ª derivada de cada ponto, foi feita a 2ª diferença central quando possível. Isso porque o erro seria da ordem do passo ao quadrado, diminuindo consideravelmente o erro. Entretanto, para pontos de bordas, não é possível utilizar o tal aproximação tão exata. Por isso, foram utilizadas técnicas de manipulação ensinadas durante as aulas. Em alguns casos, como em alguns pontos ao redor do teto do galpão, foram utilizados métodos que consideram o tamanho de seus passos menores, o que faz com que o erro ficasse maior(erro da ordem do passo). Entretanto, isso é consertado ao utilizar o método de sobrerrelaxação, que impõe um erro (no caso de 0.01) e a função é iterada até o erro máximo ser menor que o erro desejado.

O código também foi testado para diferentes passos. Para passos maiores, a imagem resultante fica com uma resolução menor. Para passos menores, os gráficos ficam um pouco mais nítidos, mas essa diferença é mínima, indicando que o uso de um passo de 0.5m está suficientemente bom para esse sistema. Uma outra diferença é que para passos menores, o código demora tempos muito mais longos para gerar os resultados, enquanto para passos maiores, os gráficos são gerados em um tempo bem mais curto. A seguir, está um exemplo de um gráfico gerado com um passo de 0.3m e 0.8m:

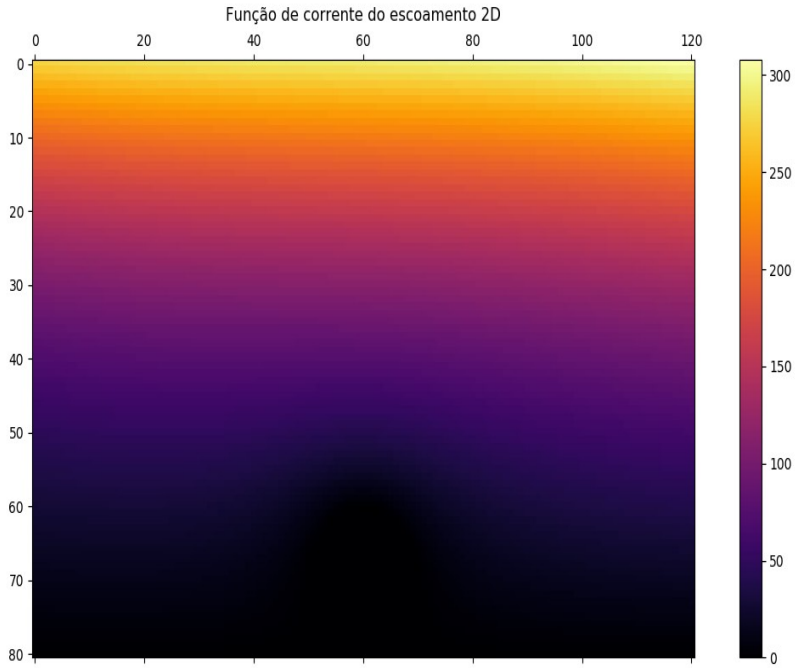


Figura 3: Gráfico 2D da função ψ de corrente do escoamento para passo de 0.3 m ε = 0.01; Iterações: 157.

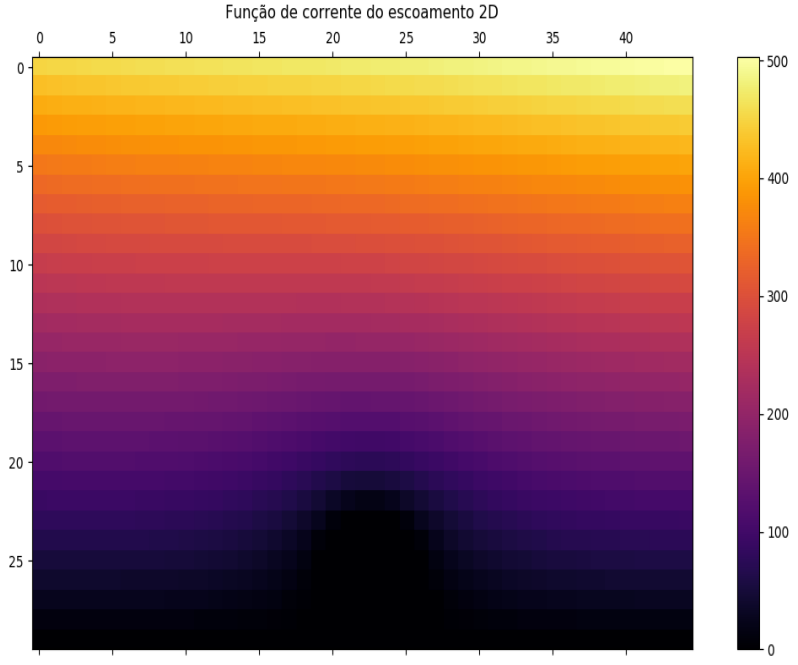


Figura 4: Gráfico 2D da função ψ de corrente do escoamento para passo de 0.8 m ε = 0.01; Iterações: 66.

O processo foi analisado para erros diferentes também. Utilizando um erro de 0.01, como recomendado pelo enunciado, resulta em um gráfico do sistema não tão próximo da realidade, pois ainda não convergiu o suficiente. Percebe-se que o uso de um erro de 0.001 resulta em gráficos mais consistentes e ideais para análise. Dessa forma, os gráficos utilizados na Parte I foram todos considerando um erro mínimo de 0.0001. Observamos também que quanto menor o passo, o valor final convergia menos pro resultado real.

Em suma, percebe-se que, para o sistema descrito, os valores de corrente do escoamento ψ é aproximadamente nulo perto do chão e dentro do galpão. A medida que a altitude aumenta, a corrente aumenta gradualmente até alcançar valores próximos de 600.

```
Passo: 0.5
Epsilon: 0.0001
Iterações: 570
Média de psi da borda superior: 595.49108586189
```

Figura 5: Parâmetros usados em uma simulação e seus resultados

b) Para plotar os vetores de velocidade absoluta do escoamento, estaremos analisando a primeira derivada em relação ao espaço da função de corrente do escoamento. Para isso, serão analisadas as derivadas em relação a x (velocidade -v) e em relação a y (velocidade u), como mostrado a seguir. Tais imposições foram feitas no enunciado (equação (5))para se chegar na equação de corrente do escoamento.

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

Para esse caso em que estaremos analisando a 1ª derivada da função de corrente do escoamento, utilizaremos a 1ª diferença aproximada central, já que possui o menor erro, da ordem do passo ao quadrado. No caso de v, utilizaremos pontos vizinhos acima e abaixo do ponto em questão e, no caso de u, utilizaremos pontos vizinhos da direita e da esquerda. Manipulações deverão ser feitas para os pontos perto de bordas, como derivadas progressivas e regressivas de três pontos (que mantém o erro na mesma ordem) e a expansão de Taylor utilizando passos menores que os definidos (utilizando a e b). É interessante notar que alguns casos particulares para uma direção não se aplicam para a outra direção. Por exemplo, para o limite superior da figura, a 1ª diferença aproximada central funciona na direção x, mas para a direção y, é preciso fazer a 1ª diferença aproximada regressiva de três pontos.

Aqui, a partir da matriz de corrente do escoamento, serão geradas uma matriz com os valores da velocidade na direção em x e outra matriz com os valores da velocidade na direção em y. Em seguida, ambas serão somadas vetorialmente, resultando na matriz com as velocidades absolutas, mostradas com o uso da função *quiver*. A seguir, estão listadas as equações finais para u e v utilizadas, depois de serem manipuladas.

$u = \frac{\partial \Psi}{\partial y}$ (velocidade na direção x)

$v = -\frac{\partial \Psi}{\partial x}$ (velocidade na direção y)

Diferenciação numérica em função de Ψ

Direção x (velocidade x)

- Ponto interno generalizado

$$u = \frac{\Psi_{i,j+1} - \Psi_{i,j-1}}{2\Delta y}$$

- Limite superior

$u = 27,78$ m/s

- Limite inferior

$u = 0$

- Interior do galpão

$u = 0$

- Borda do teto do galpão

$$a = \frac{21 - i\Delta y - \sqrt{13^2 - (j\Delta x - 18)^2}}{\Delta y}$$

$$u = \frac{1}{a(a+1)\Delta y}\left[(1-a^2)\Psi_{i,j} + a^2 \Psi_{i,j+1}\right]$$

Direção y (velocidade y)

- Ponto interno generalizado

$$v = \frac{\Psi_{i-1,j} - \Psi_{i+1,j}}{2\Delta x}$$

- Limite esquerdo

$v = 0$

- Limite direito

$v = 0$

- Borda do galpão esquerda

$$v = \frac{-3\Psi_{i,j} + 4\Psi_{i-1,j} - \Psi_{i-2,j}}{2\Delta x}$$

- Borda do galpão direita

$$v = \frac{\Psi_{i+2,j} - 4\Psi_{i+1,j} + 3\Psi_{i,j}}{2\Delta x}$$

- Borda do teto do galpão esquerda

$$b = \frac{118 - j\Delta x| - 3\cos\left(\text{arcsen}\left(\frac{21 - i\Delta y}{3}\right)\right)}{\Delta x}$$

$$v = \frac{-1}{(b^2+b)\Delta x}\left[-b^2 \Psi_{i-1,j} + (b^2-1)\Psi_{i,j}\right]$$

- Borda do teto do galpão direita

$$v = \frac{-1}{(b^2+b)\Delta x}\left[b^2 \Psi_{i+1,j} + (1-b^2)\Psi_{i,j}\right]$$

Para implementar nas equações finais acima, foram feitas as mesmas adaptações nos índices do item anterior. Seguem a seguir imagens da implementação em Python para a modelagem da velocidade absoluta. É importante notar que este item necessita do item anterior, então o código foi implementado logo abaixo dos códigos apresentados anteriormente.

CÓDIGO:

```
def parte_1b(matriz):
 velocidadex = np.zeros(shape = (linhas, colunas)) #matriz de velocidades na direção x
 velocidadey = np.zeros(shape = (linhas, colunas)) #matriz de velocidades na direção y

 #geração da matriz de velocidades em y, vetor v
 for i in range(len(matriz)): #percorre todas as linhas da matriz
  for j in range(len(matriz[0])): #percorre todas as colunas da matriz
   altura = 24 - i * dy
   comprimento = j * dx
   modulo = np.sqrt(np.power(18 - j * dx, 2) + np.power(altura - 3, 2))


   if comprimento >= 15 and comprimento <= 21 and altura <= 3:#considera corrente dentro do galpão igual a
zero
    velocidadey[i][j] = 0

   elif comprimento >= 15 and comprimento <= 21 and altura >= 3  and altura <= 6 and modulo <= 3:
    velocidadey[i][j] = 0

   elif j == 0: #condição do limite esquerdo
    velocidadey[i][j] = 0

   elif j == len(matriz[0]) - 1: #condição do limite direito
    velocidadey[i][j] = 0

    elif 15 - comprimento < dx and 15 - comprimento > 0 and altura <= 3: #condição da borda do galpão
esquerdo e suas proximidades
    velocidadey[i][j] = (-3 * matriz[i][j] + 4 * matriz[i][j-1] - matriz[i][j-2]) / (2 * dx)

    elif comprimento - 21 < dx and comprimento - 21 > 0 and altura <= 3: #condição da borda do galpão
direito e suas proximidades
    velocidadey[i][j] = (matriz[i][j+2] - 4 * matriz[i][j+1] + 3 * matriz[i][j]) / (2 * dx)

   elif comprimento < 18 and comprimento >= 15 and np.sqrt(np.power(18 - (j + 1) * dx, 2) + np.power(21 - i
* dy, 2)) < 3 and altura >= 3:
   #condição telhado esquerdo
    b = (abs(18 - j * dx) - 3 * np.cos(np.arcsin((21 - i * dy) / 3))) / dx

    velocidadey[i][j] = (-1 / (b * (b + 1) * dx)) * (-b * b * matriz[i][j-1] + (b * b - 1) * matriz[i][j])

   elif comprimento > 18 and comprimento <= 21 and np.sqrt(np.power(18 - (j - 1) * dx, 2) + np.power(21 - i
* dy, 2)) < 3 and altura >= 3:
   #condição do telhado direito
    b = (abs(18 - j * dx) - 3 * np.cos(np.arcsin((21 - i * dy) / 3))) / dx

    velocidadey[i][j] = (-1 / (b * (b + 1)* dx)) * (b * b * matriz[i][j+1] + (1 - b * b) * matriz[i][j])

   else: #pontos internos generalizados
    velocidadey[i][j] = (matriz[i][j-1] - matriz[i][j+1]) / (2 * dx)

 #geração da matriz de velocidade em x, vetor u
 for i in range(len(matriz)):
  for j in range(len(matriz[0])):
   altura = 24 - i * dy
   comprimento = j * dx
   modulo = np.sqrt(np.power(18 - j * dx, 2) + np.power(altura - 3, 2))

   if comprimento >= 15 and comprimento <= 21 and altura < 3:#considera corrente dentro do galpão igual a
zero
    velocidadex[i][j] = 0

   elif comprimento >= 15 and comprimento <= 21 and altura >= 3  and altura <= 6 and modulo <= 3:
    velocidadex[i][j] = 0

   elif i == 0: #condição do limite superior
    velocidadex[i][j] = V

   elif i == len(matriz) - 1: #condição do limite inferior
    velocidadex[i][j] = 0

   elif comprimento >= 15 and comprimento <= 21 and np.sqrt(np.power(21 - (i + 1) * dy, 2) + np.power(18 - j
* dx, 2)) < 3:
    #condição da borda do galpão superior e suas proximidades
    a = (21 - i * dy - np.sqrt(9 - np.power(j * dx - 18, 2))) / dy

    velocidadex[i][j] = (1/(a * (a + 1) * dy)) * ((1 - a * a) * matriz[i][j] + a * a * matriz[i-1][j])

   else: #pontos internos generalizados
```

```
    velocidadex[i][j] = (matriz[i-1][j] - matriz[i+1][j]) / (2 * dy)


max_vel = np.amax(np.sqrt(np.power(velocidadex, 2) + np.power(velocidadey, 2)))
print("Velocidade máxima:", max_vel)
x = np.linspace(0, 36, (36/dx) + 1)
y = np.linspace(24, 0, (24/dx) + 1)

fig3, ax3 = plt.subplots()
ax3.set_title("Vetores de velocidade absoluta do escoamento")
M = np.hypot(velocidadex, velocidadey)
Q = ax3.quiver(x, y, velocidadex, velocidadey, M, units='x', width=0.08, scale=20, cmap=cm.inferno)
qk = ax3.quiverkey(Q, 0.9, 0.9, 20, r'$20 \frac{m}{s}$', labelpos='E', coordinates='figure')

plt.show()
```

Dessa forma, a função *quiver* uniu as duas funções com vetores da direção x e vetores na direção y, resultando em um gráfico de flechas apresentada a seguir.

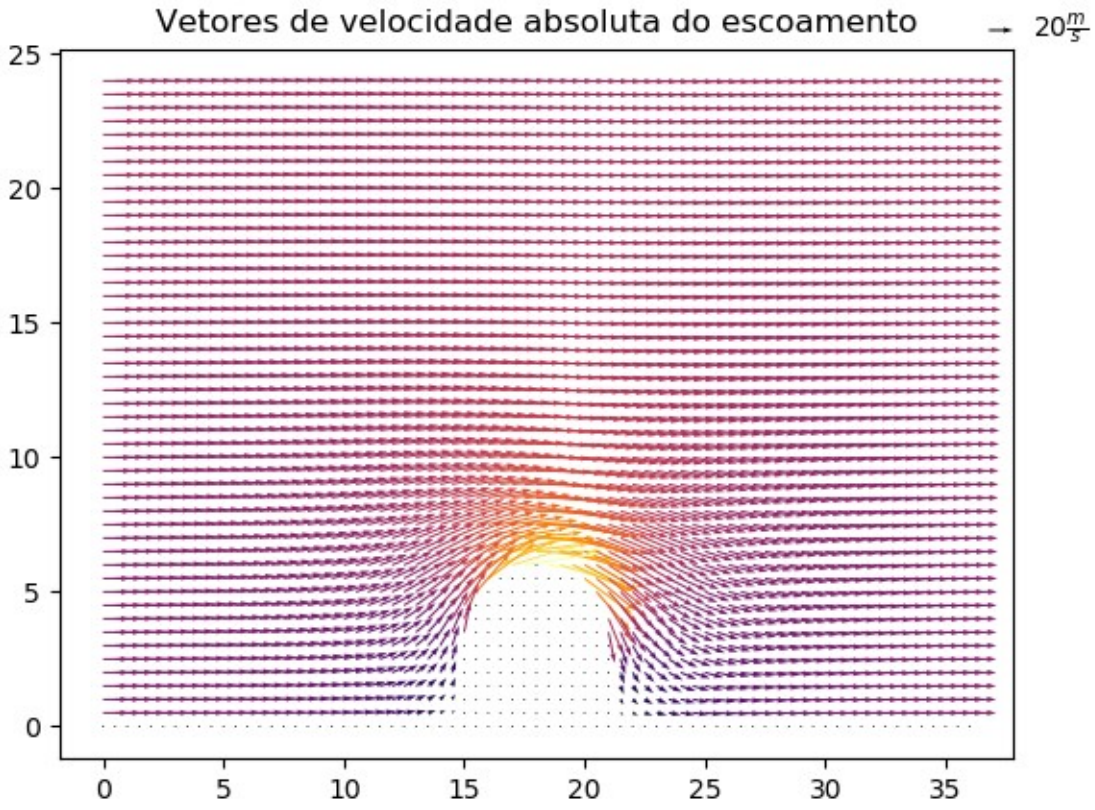


Figura 6: Gráfico com vetores de velocidade absoluta com uso de passo de 0.5m
ε = 0.0001; Iterações: 570.

Em suma, pode-se observar que aos vetores de velocidade absoluta são majoritariamente na direção de x no sentido positivo, com valores próximos a zero no chão e dentro do galpão (como imposto no enunciado da atividade). Ao redor do galpão, observa-se que o vento rodeia o teto em direções de tangentes, onde encontram-se as maiores velocidades. Passando o galpão, a velocidade volta a ser predominantemente para a direita. A velocidade máxima atingida é de 62 m/s, ou 223 km/h, no topo do galpão.

c) Para a análise da pressão (p(x, y) - p_{atm}) no domínio, utilizaremos a equação (8) do

enunciado:

$$p(x,y) - p_{atm} = \rho \frac{\gamma_{ar} - 1}{\gamma_{ar}} \left(\frac{(0)^2}{2} - \frac{(\sqrt{u(x,y)^2 + v(x,y)^2})^2}{2} \right)$$

A partir dela, é possível reduzí-la, de forma que obtêm-se o seguinte:

$$p(x-y) - p_{atm} = \frac{\rho(\gamma_{ar} - 1)}{2\gamma_{ar}} \left[-u^2(x,y) - v^2(x,y) \right]$$

Dessa forma, para encontrar a pressão no domínio, será preciso as matrizes encontradas no item anterior u(x, y) de velocidade na direção x e v(x, y) de velocidade na direção y. Logo, os códigos escritos para essa análise se encontram abaixo:

CÓDIGO:

```
def parte_1c(velocidadex, velocidadey):
 x = np.linspace(0, 36, (36/dx) + 1.0)
 y = np.linspace(24, 0, (24/dx) + 1.0)

 dpressaodominio = np.zeros(shape = (linhas, colunas)) #matriz da vriação de pressão no domínio

 #propriedades do ar
 ro = 1.25 #kg/m^3
 gama = 1.4

 for i in range(len(matriz)):
  for j in range(len(matriz[0])):
   dpressaodominio[i][j] = ((ro * (gama - 1)) / (2 * gama)) * (-np.power(velocidadex[i][j], 2) -
np.power(velocidadey[i][j], 2))

 min_pres = np.amin(dpressaodominio)
 print("Pressão mínima:", min_pres)

 fig = plt.figure()
 ax = fig.add_subplot(111)
 ax.set_title("Variação de pressão no domínio")
 cax = ax.matshow(dpressaodominio, interpolation='nearest', cmap=cm.inferno)
 fig.colorbar(cax)
 plt.show()

 nx = int((2 * d + L)/dx) + 1
 ny = int((H)/dy) + 1
 x = np.linspace(2*d+L, 0, nx)
 y = np.linspace(H, 0, ny)
 X, Y = np.meshgrid(x, y)
 ax1 = fig.gca(projection="3d")
 ax1.set_title("Variação de pressão p(x, y) - p_atm no domínio")
 s = ax1.plot_surface(X, Y, dpressaodominio, cmap=cm.inferno, linewidth=5, antialiased=True)
 ax1.set_xlabel("x (m)")
 ax1.set_ylabel("y (m)")
 ax1.set_zlabel("Pressão (pascal)")
 plt.show()
```

Após implementadas, foram gerados os seguintes gráficos:

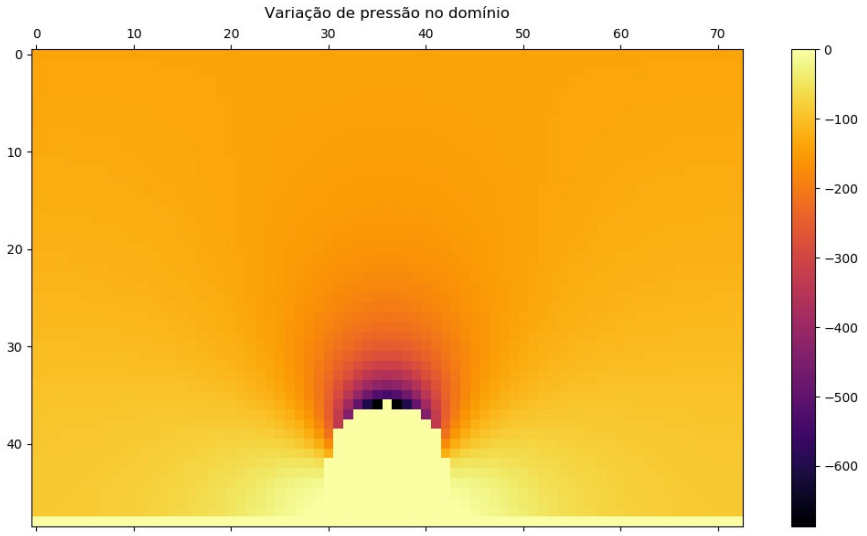


Figura 7: Gráfico 2D da variação de pressão (p(x, y) - p_{atm}) no domínio com passo 0.5m
ε = 0.0001; Iterações: 570.

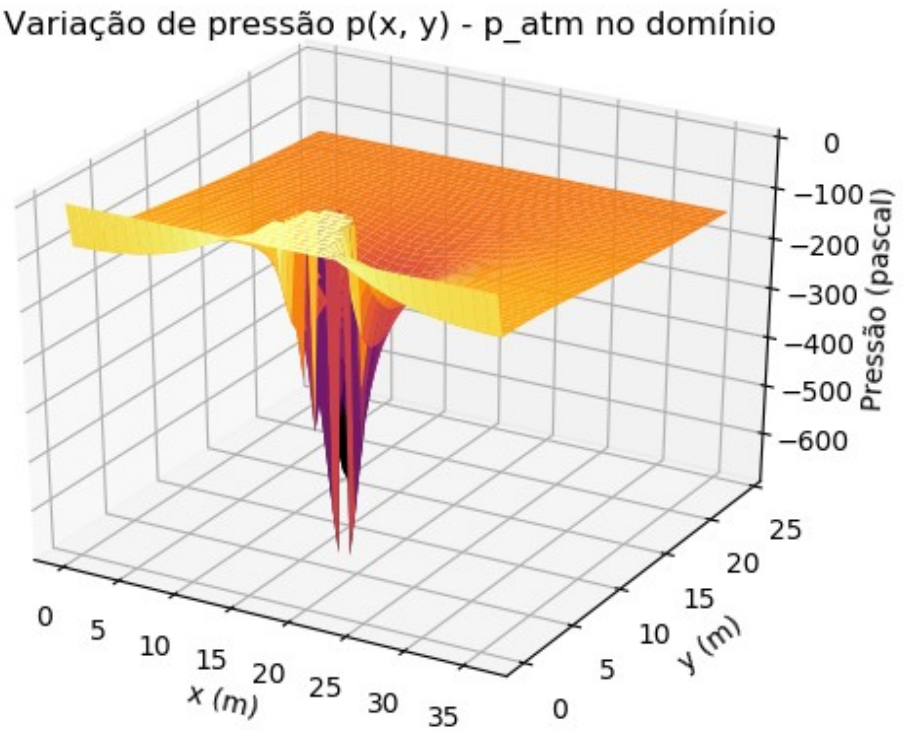
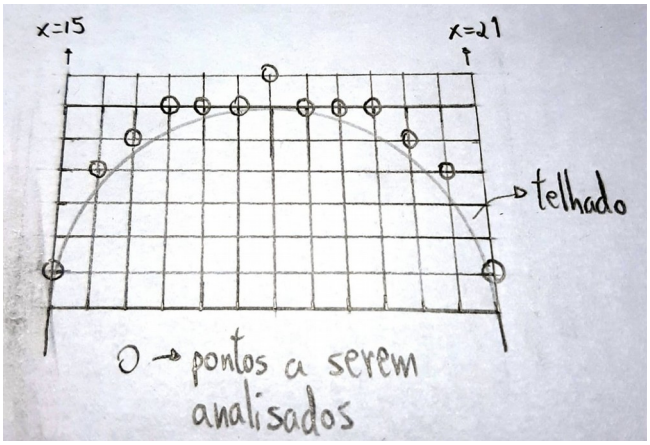


Figura 8: Gráfico 3D da variação de pressão (p(x, y) - p_{atm}) no domínio com passo 0.5m
ε = 0.0001; Iterações: 570.

Em suma, a análise de pressões do sistema nos mostrou que a pressão varia entre 0 e 150 ao longo do espaço. Entretanto, é possível ver valores muito menores de pressão exatamente acima do telhado do galpão, que chegam a aproximadamente -

687Pa. Observa-se que dentro do galpão e em seus limites, as velocidades foram definidas no enunciado sendo nulas.

d) Para analisarmos a pressão (p(x, y) - p_{atm}) ao longo do telhado, usaremos como base o resultado do item anterior. Como o sistema está sendo resolvido numericamente, serão observados pontos discretos ao longo do telhado, de acordo com a divisão do passo adotado. Os valores serão analisados para cada x distinto ao longo do telhado. Ou seja, para cada x que está dentro do telhado, será escolhido um ponto da malha que está mais próximo ao telhado. Como na Parte I estamos utilizando o passo de 0.5 metros, os pontos a serem analisados estão esboçados a seguir:



Nota-se que os pontos da malha que ficam exatamente sobre o telhado não estão sendo analisados. Isso porque, como indica o enunciado, a corrente de escoamento no galpão é nula. Dessa forma, pontos que coincidem com o galpão foram considerados como pontos que fazem parte do interior do galpão e, portanto, possuindo pressão próxima a zero. Portanto, para que a análise de pressões ao longo do telhado seja mais verossímil, escolheremos pontos que não fazem parte do interior do galpão.

A implementação do algoritmo descrito anteriormente é feito a seguir para diferentes dx e dy escolhidos.

CÓDIGO:

```
def parte_1d(dpressaodominio):

 l = []
 c = []
 for j in range(round(15//dx), round(21//dx) + 1):
  for i in range(round(18//dy - 1), round(21//dy) + 1):
   altura = 24 - i * dy

   if altura - np.sqrt(9 - np.power(j * dx - 18, 2)) - 3 <= dy and altura - np.sqrt(9 - np.power(j * dx - 18,
2)) - 3 > 0:
    l.append(i)
    c.append(j)

 pressaoteto = []
 for i in range(len(l)):
  pressaoteto.append(dpressaodominio[l[i]][c[i]])

 fig = plt.figure()
 ax = fig.add_subplot(111)
 ax.set_title("Variação da pressão na borda do telhado do galpão")
 ax.plot(pressaoteto)
 ax.set_xlabel("x (m)")
```

```
ax.set_ylabel("Pressão (N/m²)")
plt.show()

return pressaoteto
```

É importante notar que esse código também é continuação dos códigos apresentados anteriormente, pois faz uso do resultado obtido no item anterior. O código escrito resultou no seguinte gráfico para a variação da pressão ao longo do telhado.

Figura 9: Gráfico que mostra a variação da pressão na borda do telhado com passo de 0.5m

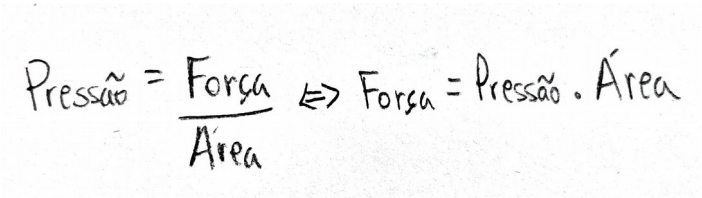
É importante notar o formato em “W” que aparece no meio do gráfico. Como mencionado anteriormente, isso ocorre devido à discretização definida de passo de 0.5 metro. Como um dos pontos da malha era o topo do telhado, o ponto possui corrente de escoamento nulo, por definição do enunciado. Dessa forma, foi analisada a pressão um ponto acima do topo do telhado.É essencial notar que a pressão no topo do telhado na realidade é o ponto de máxima pressão manométrica em módulo, mesmo que a análise mostre o contrário. Também é preciso notar que no eixo x apresenta-se o número de pontos analisados (um total de 13 pontos da borda do telhado).

Dos pontos analisados, quanto maior a distância entre o ponto da malha e o telhado real, mais diferente da realidade a pressão é representada. Nesse caso, as maiores distâncias foram do topo do telhado e nas extremidades do telhado, com a distância igual ao passo. Por isso que o uso de passos menores são melhores para modelagens desse tipo. Entretanto, com o uso de passo de 0.5 m, é possível ter uma boa noção do que está ocorrendo sobre o telhado.

Analisando o gráfico resultante, é possível observar que a pressão no telhado vai diminuindo conforme se aproxima do topo do telhado, local de pressão mínima. A

pressão manométrica mínima do gráfico é de cerca de -700,53 N/m². A pressão do topo do telhado é maior, porém é da mesma magnitude, sendo suficientemente próxima a esse número.

e) Para calcularmos a força vertical resultante que atua no telhado, devemos calcular a pressão vertical que está sendo aplicada no telhado primeiro. Após encontrarmos a pressão vertical, podemos aplicar a seguinte fórmula:



Como temos todas as dimensões do galpão, a área é calculada facilmente. Para calcularmos a pressão vertical, será preciso fazer a soma das pressões encontradas no item anterior e multiplicadas pelas respectivas áreas de atuação. Como a pressão é uma grandeza escalar, para encontrar as forças na vertical, as áreas em questão devem ser as áreas horizontais. Logo, a área em cada ponto da malha discretizada será calculada como o comprimento do galpão multiplicado pelo passo.

É importante notar que os códigos utilizados foram os mesmos mostrados nos itens anteriores. A partir daqui, somam-se todas as pressões verticais aplicadas no telhado. Utilizando a fórmula apresentada e multiplicando a área de atuação horizontal pela pressão total, obtém-se a força vertical resultante que atua no telhado. O código para esse último cálculo está abaixo:

CÓDIGO:

```
def parte_1e(pressaoteto):
 comprimento = 60 #comprimento do telhado
 forca_total = 0 #armazena a somatória das pressões sobre o telhado

 for i in range(len(pressaoteto)):
  forca_total = forca_total + pressaoteto[i] *dx * comprimento

 print("Força total:", forca_total)
```

Ao final, obtém-se que a força vertical aplicada no telhado equivale a aproximadamente 176.047,41 Newtons. Essa força muda um pouco dependendo do passo, mas converge para aproximadamente 1,8 10^5 N à medida que diminuimos o passo.Dessa forma, conclui-se que a força vertical aplicada no telhado é grande o suficiente para arrancar o telhado no caso das configurações iniciais apresentadas no sistema. É por isso que é tão comum ver em ventanias e tempestades fortes telhados inteiros serem levados embora.

PARTE II

a) Para a análise da distribuição de temperatura no ar em ºC, utilizaremos como base a

equação (9) do enunciado:

$$k\nabla^2 T - \rho c_p \boldsymbol{u}.\nabla \mathrm{T} = 0$$

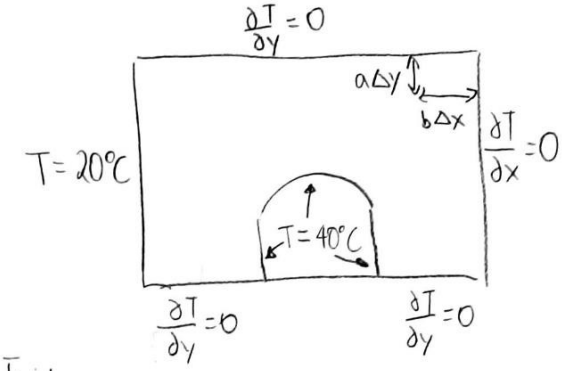
Para facilitar, essa equação foi reescrita de forma que temos o seguinte:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} - \frac{\rho C_{par}}{K_{ar}}\left(u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y}\right) = 0$$

Fazemos com que $K = \frac{\rho * c_{par}}{K_{ar}}$, obtendo:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} - K\left(u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y}\right) = 0$$

Analisamos, como no item 1a, as três bordas do domínio excluindo a esquerda, os dois cantos direitos do domínio, as duas bordas laterais do galpão e o seu telhado, dividido em esquerdo, direito e superior. Assim como foi dito no enunciado, existe dominância do termo convectivo das primeiras derivadas de T. Para evitar que isso interfira nos resultados, as derivadas primeiras de T serão equacionadas de formas diferentes, dependendo do sentido da velocidade do vento na direção x e na direção y (sinal de u(x, y) e v(x, y)). Analisando a matriz resultante u(x, y) para diferentes passos, percebe-se que todos os seus valores são maiores ou iguais a zero. Dessa forma, é preciso apenas modelar o sistema para casos de v(x, y) maior ou menor que zero. A seguir, estão o resultado da modelagem do sistema e seus casos particulares de borda.



$\varepsilon = 0{,}01$

$\Delta x = \Delta y = 0{,}375 \; ; \lambda = 1{,}85$

$u > 0 \Rightarrow \frac{\partial T}{\partial x} = \frac{T_{ij} - T_{i-1,j}}{\Delta x}$ | $v > 0 \Rightarrow \frac{\partial T}{\partial y} = \frac{T_{ij} - T_{i,j-1}}{\Delta y}$

$v < 0 \Rightarrow \frac{\partial T}{\partial y} = \frac{T_{i,j+1} - T_{ij}}{\Delta y}$

Ponto interno

$v > 0$

$$T_{ij} = \frac{1K \cdot \Delta x (u T_{i-1,j} + v T_{i,j-1}) + T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}}{1K \Delta x (u+v) + 4}$$

$v < 0$

$$T_{ij} = \frac{1K \cdot \Delta x (u T_{i-1,j} - v T_{i,j+1}) + T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}}{1K \Delta x (u-v) + 4}$$

Limite esquerdo

$T = T_{fora} \Leftrightarrow T_{ij} = 20°C$

Limite superior

$$T_{ij} = \frac{T_{i+1,j} + T_{i-1,j} + 2T_{i,j-1} + 1K u \Delta x T_{i-1,j}}{1K \cdot \Delta x \cdot u + 4}$$

Limite direito

$v > 0$

$$T_{ij} = \frac{2T_{i-1,j} + T_{i,j+1} + T_{i,j-1} + 1K \Delta x v T_{i,j-1}}{1K \Delta x v + 4}$$

$v < 0$

$$T_{ij} = \frac{2T_{i-1,j} + T_{i,j+1} + T_{i,j-1} - 1K \Delta x v T_{i,j+1}}{-1K \Delta x v + 4}$$

Limite inferior

$$T_{i,j} = \frac{T_{i+1,j} + T_{i-1,j} + 2T_{i,j+1} + 1K\,u\,\Delta x\,T_{i-1,j}}{1K\,\Delta x\,u + 4}$$

Canto esquerdo superior

$$T_{i,j} = T_{fora} = 20°C$$

Canto esquerdo inferior

$$T_{i,j} = T_{fora} = 20°C$$

Canto direito superior

$$T_{i,j} = \frac{T_{i-1,j} + T_{i,j-1}}{2}$$

Canto direito inferior

$$T_{i,j} = \frac{T_{i-1,j} + T_{i,j+1}}{2}$$

Galpão

$$a = \frac{21 - i\Delta y - \sqrt{3^2 - (j\Delta x - 18)^2}}{\Delta y}$$

$$b = \frac{118 - j\Delta x| - 3\cos\left(\text{arc sen}\left(\frac{21 - i\Delta y}{3}\right)\right)}{\Delta x}$$

Borda esquerda

$v \geq 0$

$$T_{i,j} = \frac{\frac{80}{b(b+1)} + \frac{2\,T_{i-1,j}}{b+1} + T_{i,j+1} + T_{i,j-1} + 1K\,\Delta x\,(u\,T_{i-1,j} + v\,T_{i,j-1})}{\frac{2}{b} + 2 + 1K\,\Delta x\,(u+v)}$$

$v \leq 0$

$$T_{i,j} = \frac{\frac{80}{b(b+1)} + \frac{2\,T_{i-1,j}}{b+1} + T_{i,j+1} + T_{i,j-1} + 1K\,\Delta x\,(u\,T_{i-1,j} - v\,T_{i,j+1})}{\frac{2}{b} + 2 + 1K\,\Delta x\,(u-v)}$$

Borda direita

$v > 0$

$$T_{i,j} = \frac{\frac{80}{b(b+1)} + \frac{2\bar{T}_{i+1,j}}{b+1} + \bar{T}_{i,j+1} + \bar{T}_{i,j-1} + |K|\Delta x(u\bar{T}_{i-1,j} + v\bar{T}_{i,j-1})}{\frac{2}{b} + 2 + |K|\Delta x(u+v)}$$

$v < 0$

$$T_{i,j} = \frac{\frac{80}{b(b+1)} + \frac{2\bar{T}_{i+1,j}}{b+1} + \bar{T}_{i,j+1} + \bar{T}_{i,j-1} + |K|\Delta x(u\bar{T}_{i-1,j} - v\bar{T}_{i,j+1})}{\frac{2}{b} + 2 + |K|\Delta x(u-v)}$$

Telhado esquerdo

$v > 0$

$$T_{i,j} = \frac{\frac{80}{b(b+1)} + \frac{80}{a(a+1)} + \frac{2\bar{T}_{i-1,j}}{b+1} + \frac{2\bar{T}_{i,j+1}}{a+1} + |K|\Delta x(u\bar{T}_{i-1,j} + v\bar{T}_{i,j-1})}{\frac{2}{b} + \frac{2}{a} + |K|\Delta x(u+v)}$$

$v < 0$

$$T_{i,j} = \frac{\frac{80}{b(b+1)} + \frac{80}{a(a+1)} + \frac{2\bar{T}_{i-1,j}}{b+1} + \frac{2\bar{T}_{i,j+1}}{a+1} + |K|\Delta x(u\bar{T}_{i-1,j} - v\bar{T}_{i,j+1})}{\frac{2}{b} + \frac{2}{a} + |K|\Delta x(u-v)}$$

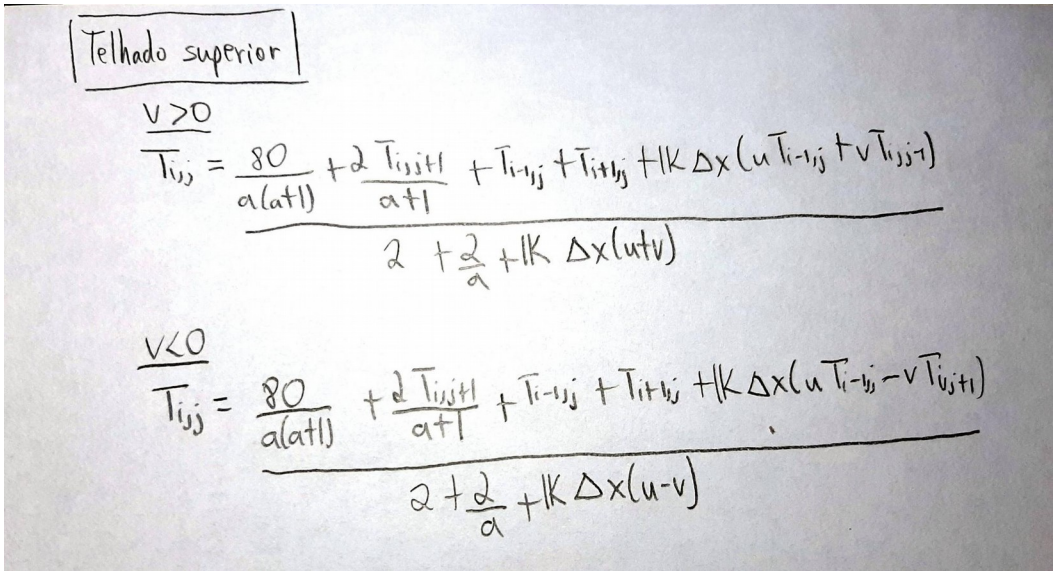
Telhado direito

$v > 0$

$$T_{i,j} = \frac{\frac{80}{b(b+1)} + \frac{80}{a(a+1)} + \frac{2\bar{T}_{i+1,j}}{b+1} + \frac{2\bar{T}_{i,j+1}}{a+1} + |K|\Delta x(u\bar{T}_{i-1,j} + v\bar{T}_{i,j-1})}{\frac{2}{b} + \frac{2}{a} + |K|\Delta x(u+v)}$$

$v < 0$

$$T_{i,j} = \frac{\frac{80}{b(b+1)} + \frac{80}{a(a+1)} + \frac{2\bar{T}_{i+1,j}}{b+1} + \frac{2\bar{T}_{i,j+1}}{a+1} + |K|\Delta x(u\bar{T}_{i-1,j} - v\bar{T}_{i,j+1})}{\frac{2}{b} + \frac{2}{a} + |K|\Delta x(u-v)}$$



Como pedido, para a Parte II, utilizaremos um passo de h/8 (0.375m) e erro de 0.01 com o fator λ = 1.15 para o cálculo da malha correspondente à variação de temperaturas. A seguir, está apresentado o código implementado para e=tal modelagem do sistema. É importante notar que para o cálculo dos valores de ψ, foi utilizado um erro de 0.0001 para melhor convergência dos resultados, com passo h/8 e λ = 1.15.

CÓDIGO:

```
def parte_2a(velocidadex, velocidadey):
 malha = np.zeros(shape=(linhas, colunas))
 contador = 0
 erro_max = 1

 while erro_max > epsilon:
  contador += 1
  erro_max = 0

  for i in range(len(malha)): #percorre as linhas da matriz
   for j in range(len(malha[0])): #percorre as colunas da matriz
    altura = 24 - i * dy
    comprimento = j * dx
    modulo = np.sqrt(np.power(18 - j * dx, 2) + np.power(altura - 3, 2))

    #configura temperatura dentro do galpão
    if comprimento >= 15 and comprimento <= 21 and altura <= 3:#considera corrente dentro do galpão igual
a zero
     temperatura = 40

    elif comprimento >= 15 and comprimento <= 21 and altura >= 3  and altura <= 6 and modulo <= 3:
     temperatura = 40

    #condicao do contorno do sistema
    elif i == 0 or j == 0 or i == len(malha) - 1 or j == len(malha[0]) - 1:

     #condicao do canto esquerdo superior
     if i==0 and j==0:
      temperatura = 20

     #condicao do canto direito superior
     elif i == 0 and j == len(malha[0]) - 1:
      temperatura = (malha[i][j-1] + malha[i+1][j]) / 2

     #condicao do canto esquerdo inferior
     elif i == len(malha) - 1 and j == 0:
      temperatura = 20

     #condicao do canto direito inferior
     elif i == len(malha) - 1 and j == len(malha[0]) - 1:
```

```
      temperatura = (malha[i][j-1] + malha[i-1][j]) / 2

     #condicao do limite esquerdo
     elif j == 0:
      temperatura = 20

     #condicao do limite superior
     elif i == 0:
      temperatura = (malha[i][j+1] + malha[i][j-1] + 2 * malha[i+1][j] + K * velocidadex[i][j] * dx *
malha[i][j-1]) / (K * dx * velocidadex[i][j] + 4)

     #condicao do limite direito
     elif j == len(malha[0]) - 1:
      if velocidadey[i][j] >= 0:
       temperatura = (2 * malha[i][j-1] + malha[i-1][j] + malha[i+1][j] + K * dx * velocidadey[i][j] *
malha[i+1][j]) / (K * dx * velocidadey[i][j] + 4)

      elif velocidadey[i][j] < 0:
       temperatura = (2 * malha[i][j-1] + malha[i-1][j] + malha[i+1][j] - K * dx * velocidadey[i][j] *
malha[i+1][j]) / (-K * dx * velocidadey[i][j] + 4)

     #condicao do limite inferior
     elif i == len(malha) - 1:
      temperatura = (malha[i][j+1] + malha[i][j-1] + 2 * malha[i-1][j] + K * velocidadex[i][j] * dx *
malha[i][j-1]) / (K * dx * velocidadex[i][j] + 4)

    elif altura <= 3 and comprimento + dx > 15 and comprimento - dx < 21:
     #condicao da borda esquerda do galpao
     if 15 - comprimento < dx and 15 - comprimento > 0:
      b = round(15 % dx, 1)

      if velocidadey[i][j] >= 0:
       temperatura = ((80/(b*(b+1))) + (2*malha[i][j-1]/(b+1)) + malha[i-1][j] + malha[i+1][j] +
K*dx*(velocidadex[i][j]*malha[i][j-1] + velocidadey[i][j]*malha[i+1][j])) / ((2/b) + 2 +
K*dx*(velocidadex[i][j] + velocidadey[i][j]))

      elif velocidadey[i][j] < 0:
       temperatura = ((80/(b*(b+1))) + (2*malha[i][j-1]/(b+1)) + malha[i-1][j] + malha[i+1][j] +
K*dx*(velocidadex[i][j]*malha[i][j-1] - velocidadey[i][j]*malha[i-1][j])) / ((2/b) + 2 +
K*dx*(velocidadex[i][j] - velocidadey[i][j]))

     #condição da borda direita do galpao
     elif comprimento - 21 < dx and comprimento - 21 > 0:
      b = round(21 % dx, 1)

      if velocidadey[i][j] >= 0:
       temperatura = ((80/(b*(b+1))) + (2*malha[i][j+1]/(b+1)) + malha[i-1][j] + malha[i+1][j] +
K*dx*(velocidadex[i][j]*malha[i][j-1] + velocidadey[i][j]*malha[i+1][j])) / ((2/b) + 2 +
K*dx*(velocidadex[i][j] + velocidadey[i][j]))

      elif velocidadey[i][j] < 0:
       temperatura = ((80/(b*(b+1))) + (2*malha[i][j+1]/(b+1)) + malha[i-1][j] + malha[i+1][j] +
K*dx*(velocidadex[i][j]*malha[i][j-1] - vrlocidadey[i][j]*malha[i-1][j])) / ((2/b) + 2 +
K*dx*(velocidadex[i][j] - velocidadey[i][j]))

    #condicao da borda do telhado
    elif comprimento >= 15 and comprimento <= 21 and altura < 6 + dy and altura >= 3:

     #condicao para borda do teto esquerda com uso de a e b
     if comprimento < 18 and np.sqrt(np.power(18 - (j + 1) * dx, 2) + np.power(21 - i * dy, 2)) < 3 and
np.sqrt(np.power(18 - j * dx, 2) + np.power(21 - (i + 1) * dy, 2)) < 3:
      a = (21 - i * dy - np.sqrt(9 - np.power(j * dx - 18, 2))) / dy
      b = (abs(18 - j * dx) - 3 * np.cos(np.arcsin((21 - i * dy) / 3))) / dx

      if velocidadey[i][j] >= 0:
       temperatura = ((80/(b*(b+1))) + (80/(a*(a+1))) + (2*malha[i][j-1]/(b+1)) + (2*malha[i-1][j]/
(a+1)) + K*dx*(velocidadex[i][j]*malha[i][j-1] + velocidadey[i][j]*malha[i+1][j])) / ((2/b) + (2/a) +
K*dx*(velocidadex[i][j] + velocidadey[i][j]))

      elif velocidadey[i][j] < 0:
       temperatura = ((80/(b*(b+1))) + (80/(a*(a+1))) + (2*malha[i][j-1]/(b+1)) + (2*malha[i-1][j]/
(a+1)) + K*dx*(velocidadex[i][j]*malha[i][j-1] - velocidadey[i][j]*malha[i-1][j])) / ((2/b) + (2/a) +
K*dx*(velocidadex[i][j] - velocidadey[i][j]))

     #condicao para borda do teto direita com uso de a e b
     elif comprimento > 18 and np.sqrt(np.power(18 - (j - 1) * dx, 2) + np.power(21 - i * dy, 2)) < 3 and
np.sqrt(np.power(18 - j * dx, 2) + np.power(21 - (i + 1) * dy, 2)) < 3:
      a = (21 - i * dy - np.sqrt(9 - np.power(j * dx - 18, 2))) / dy
      b = (abs(18 - j * dx) - 3 * np.cos(np.arcsin((21 - i * dy) / 3))) / dx

      if velocidadey[i][j] >= 0:
```

```
       temperatura = ((80/(b*(b+1))) + (80/(a*(a+1))) + (2*malha[i][j+1]/(b+1)) + (2*malha[i-1][j]/
(a+1)) + K*dx*(velocidadex[i][j]*malha[i][j-1] + velocidadey[i][j]*malha[i+1][j])) / ((2/b) + (2/a) +
K*dx*(velocidadex[i][j] + velocidadey[i][j]))

      elif velocidadey[i][j] < 0:
       temperatura = ((80/(b*(b+1))) + (80/(a*(a+1))) + (2*malha[i][j+1]/(b+1)) + (2*malha[i+1][j]/
(a+1)) + K*dx*(velocidadex[i][j]*malha[i][j-1] - velocidadey[i][j]*malha[i-1][j])) / ((2/b) + (2/a) +
K*dx*(velocidadex[i][j] - velocidadey[i][j]))


     #condicao para borda inferior do teto esquerda com uso de b
     elif comprimento < 18 and np.sqrt(np.power(18 - (j + 1) * dx, 2) + np.power(21 - i * dy, 2)) < 3:
      b = abs((abs(18 - j * dx) - 3 * np.cos(np.arcsin((21 - i * dy) / 3))) / dx)

      if velocidadey[i][j] >= 0:
       temperatura = ((80/(b*(b+1))) + (2*malha[i][j-1]/(b+1)) + malha[i-1][j] + malha[i+1][j] +
K*dx*(velocidadex[i][j]*malha[i][j-1] + velocidadey[i][j]*malha[i+1][j])) / ((2/b) + 2 +
K*dx*(velocidadex[i][j] + velocidadey[i][j]))

      elif velocidadey[i][j] < 0:
       temperatura = ((80/(b*(b+1))) + (2*malha[i][j-1]/(b+1)) + malha[i-1][j] + malha[i+1][j] +
K*dx*(velocidadex[i][j]*malha[i][j-1] - velocidadey[i][j]*malha[i-1][j])) / ((2/b) + 2 +
K*dx*(velocidadex[i][j] - velocidadey[i][j]))

     #condicao para borda inferior do teto direita com uso de b
     elif comprimento > 18 and np.sqrt(np.power(18 - (j - 1) * dx, 2) + np.power(21 - i * dy, 2)) < 3:
      b = abs((abs(18 - j * dx) - 3 * np.cos(np.arcsin((21 - i * dy) / 3))) / dx)

      if velocidadey[i][j] >= 0:
       temperatura = ((80/(b*(b+1))) + (2*malha[i][j+1]/(b+1)) + malha[i-1][j] + malha[i+1][j] +
K*dx*(velocidadex[i][j]*malha[i][j-1] + velocidadey[i][j]*malha[i+1][j])) / ((2/b) + 2 +
K*dx*(velocidadex[i][j] + velocidadey[i][j]))

      elif velocidadey[i][j] < 0:
       temperatura = ((80/(b*(b+1))) + (2*malha[i][j+1]/(b+1)) + malha[i-1][j] + malha[i+1][j] +
K*dx*(velocidadex[i][j]*malha[i][j-1] - velocidadey[i][j]*malha[i-1][j])) / ((2/b) + 2 +
K*dx*(velocidadex[i][j] - velocidadey[i][j]))

     #condicao para borda superior do teto com uso de a
     elif np.sqrt(np.power(18 - j * dx, 2) + np.power(21 - (i + 1) * dy, 2)) < 3:
      a = (21 - i * dy - np.sqrt(9 - np.power(j * dx - 18, 2))) / dy

      if velocidadey[i][j] >= 0:
       temperatura = ((80/(a*(a+1))) + (2*malha[i-1][j]/(a+1)) + malha[i][j-1] + malha[i][j+1] +
K*dx*(velocidadex[i][j]*malha[i][j-1] + velocidadey[i][j]*malha[i+1][j])) / ((2/a) + 2 +
K*dx*(velocidadex[i][j] + velocidadey[i][j]))

      elif velocidadey[i][j] < 0:
       temperatura = ((80/(a*(a+1))) + (2*malha[i-1][j]/(a+1)) + malha[i][j-1] + malha[i][j+1] +
K*dx*(velocidadex[i][j]*malha[i][j-1] - velocidadey[i][j]*malha[i-1][j])) / ((2/a) + 2 +
K*dx*(velocidadex[i][j] - velocidadey[i][j]))

     #atualiza os pontos dentro da condicao inicial, mas que não se encaixam em nenhum dos casos
anteriores
     else:
      if velocidadey[i][j] >= 0:
       temperatura = (K*dx*(velocidadex[i][j]*malha[i][j-1] + velocidadey[i][j]*malha[i+1][j]) +
malha[i][j+1] + malha[i][j-1] + malha[i-1][j] + malha[i+1][j]) / (K*dx*(velocidadex[i][j] +
velocidadey[i][j]) + 4)

      elif velocidadey[i][j] < 0:
       temperatura = (K*dx*(velocidadex[i][j]*malha[i][j-1] - velocidadey[i][j]*malha[i-1][j]) +
malha[i][j+1] + malha[i][j-1] + malha[i-1][j] + malha[i+1][j]) / (K*dx*(velocidadex[i][j] -
velocidadey[i][j]) + 4)

    #atualiza pontos internos genéricos
    else:
     if velocidadey[i][j] >= 0:
      temperatura = (K*dx*(velocidadex[i][j]*malha[i][j-1] + velocidadey[i][j]*malha[i+1][j]) +
malha[i][j+1] + malha[i][j-1] + malha[i-1][j] + malha[i+1][j]) / (K*dx*(velocidadex[i][j] +
velocidadey[i][j]) + 4)

     elif velocidadey[i][j] < 0:
      temperatura = (K*dx*(velocidadex[i][j]*malha[i][j-1] - velocidadey[i][j]*malha[i-1][j]) +
malha[i][j+1] + malha[i][j-1] + malha[i-1][j] + malha[i+1][j]) / (K*dx*(velocidadex[i][j] -
velocidadey[i][j]) + 4)

    # Sobrerrelaxação:
    fator = 1
    temp_nova = fator*temperatura + (1-fator)*malha[i][j]
    erro_atual = abs((temp_nova - malha[i][j])/temp_nova) if temp_nova != 0 else 0
```

```
    erro_max = max(erro_atual, erro_max) #atualiza com maior erro
    malha[i][j] = temp_nova

fig = plt.figure()
ax = fig.add_subplot(111)
ax.set_title("Temperatura do domínio (°C)")
cax = ax.matshow(malha, interpolation='nearest', cmap=cm.inferno)
fig.colorbar(cax)

plt.show()
```

O código utilizado faz uso de de códigos anteriores que resultam nas matrizes de velocidade nas direções x e y. Para os requisitos do sistema, a distribuição de temperatura no ar em ºC ficou da seguinte forma:

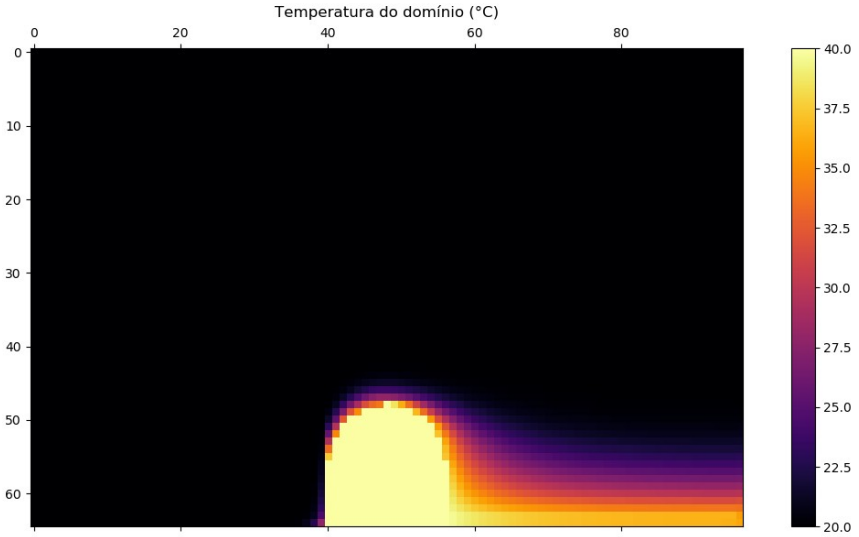


Figura 10: Gráfico 2D da temperatura para passo de 0.375 m.
ε = 0.01; Iterações: 20.

A temperatura no interior do galpão é fixada em 40°C, a temperatura na borda esquerda é fixada em 20°C. Podemos ver o efeito do vento soprando da esquerda para a direita no galpão, removendo calor e causando uma trilha atrás do galpão onde a temperatura está entre 20 e 40 °C. O resto do domínio permanece quase completamente em 20°C, não podendo trocar calor com o galpão.

b) Para a análise da taxa de calor retirada no prédio, usaremos como base a equação (11) do enunciado:

$$q = \int_A \vec{\mathbf{Q}}.\vec{\mathbf{n}}dA = -\int_A k\frac{\partial T}{\partial n}dA = -\int_A k\nabla T.\vec{n}dA$$

Para isso, dos pontos da malha, serão selecionados os pontos mais próximos das paredes e teto do galpão. A partir desses pontos, serão calculadas as derivadas primeiras das temperaturas em x e em y de cada um. É importante observar que os pontos que fazem parte da parede possuem a primeira derivada em y nulas, já que todos os pontos estão na mesma temperatura. Para isso, utilizaremos diferenças

aproximadas regressivas e progressivas de três pontos, que possuem um erro menor da ordem do passo ao quadrado.

Com as derivadas primeiras calculadas, multiplicaremos cada um pela respectiva área de atuação correspondente, considerando o comprimento do galpão. Nota-se que a área não é a área real, e sim a área perpendicular à região de atuação. Isso equivale à uma decomposição do vetor n, que é perpendicular à superfície e apontando para fora do galpão. A soma de todos os fluxos de calor multiplicados pelas suas áreas é equivalente à integral, já que o passo considerado é suficientemente pequeno. A seguir, o algoritmo descrito anteriormente está implementado no código a seguir:

CÓDIGO:

```
def parte_2b(malha):
 l = []
 c = []

 #armazena pontos da borda do prédio

 for j in range(round(15//dx) - 1, round(21//dx) + 2):
  for i in range(round(18//dy - 1), round(24//dy)):
   altura = 24 - i * dy
   comprimento = j * dx

   if altura < 3 and (comprimento == 14.625 or comprimento == 21.375):
    l.append(i)
    c.append(j)

   elif altura - np.sqrt(9 - np.power(j * dx - 18, 2)) - 3 <= dy and altura - np.sqrt(9 - np.power(j * dx -
18, 2)) - 3 > 0:
    l.append(i)
    c.append(j)

 q = 0 #armazena calor total trocado no prédio

 for i in range(len(l)):
 #calcula calor total trocado em cada ponto
  derivadax = 0
  derivaday = 0
  calor = 0
  altura = 24 - l[i] * dy
  comprimento = c[i] * dx

  if altura < 3 and comprimento < 18:
   derivadax = -(3 * malha[l[i]][c[i]] - 4 * malha[l[i]][c[i]-1] + malha[l[i]][c[i]-2]) / (2 * dx)

  elif altura < 3 and comprimento > 18:
   derivadax = (-malha[l[i]][c[i]+2] + 4 * malha[l[i]][c[i]+1] - 3 * malha[l[i]][c[i]]) / (2 * dx)

  elif altura >= 3 and comprimento <18:
   derivadax = -(3 * malha[l[i]][c[i]] - 4 * malha[l[i]][c[i]-1] + malha[l[i]][c[i]-2]) / (2 * dx)
   derivaday = (-malha[l[i]-2][c[i]] + 4 * malha[l[i]-1][c[i]] - 3 * malha[l[i]][c[i]]) / (2 * dy)

  elif altura >= 3 and comprimento > 18:
   derivadax = (-malha[l[i]][c[i]+2] + 4 * malha[l[i]][c[i]+1] - 3 * malha[l[i]][c[i]]) / (2 * dx)
   derivaday = (-malha[l[i]-2][c[i]] + 4 * malha[l[i]-1][c[i]] - 3 * malha[l[i]][c[i]]) / (2 * dy)

  else:
```

```
    derivadax = (malha[l[i]][c[i]+1] - malha[l[i]][c[i]-1]) / (2 * dx)
    derivaday = (-malha[l[i]-2][c[i]] + 4 * malha[l[i]-1][c[i]] - 3 * malha[l[i]][c[i]]) / (2 * dy)

  calor = k * derivadax * dx * comprimento_estrutura + k * derivaday * dy * comprimento_estrutura

  q = q + calor

print(q)
```

Ao final, obtém-se uma taxa de calor retirado do prédio de -419,31 Watts. Nota-se que a constante k_{ar} utiliza Kelvin como unidade para a temperatura. Entretanto, como a conversão de Celsius para Kelvin é somar 273,15 e ∇T é calculado pela diferença de temperaturas entre pontos, o uso de unidades diferentes para temperatura se anulam. O resultado varia bastante dependendo do passo utilizado.

CONCLUSÃO

Em suma, ao analisar todos os resultados encontrados, é possível ter uma boa noção do que acontece em sistemas desse tipo. Ventos de alta velocidade, como ocorre em grandes tempestades, pode chegar a causar enormes estragos. Para construções semelhantes ao galpão considerado, a baixa pressão causada pelos ventos em torno do telhado podem gerar forças próximas a 1,8 10^5N para cima, podendo facilmente arrancar telhados inteiros, dependendo do caso. Analisando a temperatura, observa-se também que a ação do vento consegue retirar calor de dentro do prédio, a taxas próximas de 420W, aumentando as temperaturas nas regiões mais próximas ao chão ao redor do galpão.