Trabalho Computacional - Caso 7
Transporte de Calor e Massa
Resolução da Equação do Calor Transiente 2D utilizando Método das Diferenças Finitas

Marcos Eduardo Monteiro Junqueira

`memj401@hotmail.com`

`18/0023691`

16 de Outubro, 2020

1 Introdução

No estudo dos processo de troca de calor, o primeiro modelo a ser estudo é o da condução. No entanto, ainda que mais simples que os outros, este modelo ainda apresenta um alto grau de complexidade matemática, tornando quase que obrigatório a utilização de hipóteses simplificativas para o seu estudo.

Para o estudo da condução, a principal equação que rege todo o processo é dada na forma da Equação do Calor, explicitada abaixo:

$$\frac{\partial T}{\partial t}\rho C_p = k\nabla^2 T + \dot{q} \tag{1}$$

Em que k,C_p e $\dot{q}$ representam, respectivamente, a Condutividade Térmica, o Calor Específico e a Geração Interna do corpo estudo. A partir de uma observação cuidadosa da equação acima, é possível perceber que o processo da condução de calor apresenta um dependência espacial e temporaral, fato que torna a solução dessa equação em sua totalidade extremamente complexa e altamente custosa.

Em função disso, são utilizadas restrições e simplificações a fim de dimnuir sua comlexidade. A simplificação mais comum é a análise da equação em regime permanente e 1D, conhecida como Lei de Fourier, e possível resolvê-la por métodos com o das resistências térmicas. Outro simplificação bastante comum é o regime transiente sem variações de temperatura espacial, em que é utilizado o método da Capacitância Concentrada.

No entanto, o uso dessas simplificações restinge as suas aplicações e para muitos casos é insuficiente. Em função disso, foram desenvolvidos métodos numéricos aplicados juntamente com a computação a fim de modelar, em razoável precisão e tempo, condições de calor mais sofisticadas e próximas da realidade. No caso deste trabalho, foi realizado um estudo valendo-se do Método das Diferenças Finitas.

1.1 Método das Diferenças Finitas

O Método das Diferenças Finitas é um atifício numérico com o objetivo de facilitar o cálculo da Equação do Calor em regime transiente, considerando variações espaciais.

Neste método, realiza-se a discretização da Equação do Calor por meio de Séries de Taylor. Utilizando algumas simplificações e realizando o truncamento da série obtida, chega-se a seguinte expressão, que define a temperatura no instante seguinte $p+1$ de um ponto arbitrário $T_{m,n}$:

$$T^{p+1}_{m,n} = F_o(T^p_{m+1,n} + T^p_{m-1,n} + T^p_{m,n+1} + T^p_{m,n-1}) + (1-4F_o)T^p_{m,n} + \frac{\dot{q}\Delta^2}{k} \tag{2}$$

Em que F_o é o número Fourier, definido por $F_o = \frac{\alpha\Delta t}{\Delta x^2}$, onde α é a difusividade térmica do material. É de suma importância destacar que F_o está intimamente ligado á estabilidade da simulação e por isso, devem ser escolhidas as variações espaciais e temporais (Δx e Δt) com cautela.

2 Procedimento Computacional

2.1 Metodologia Aplicada ao Código

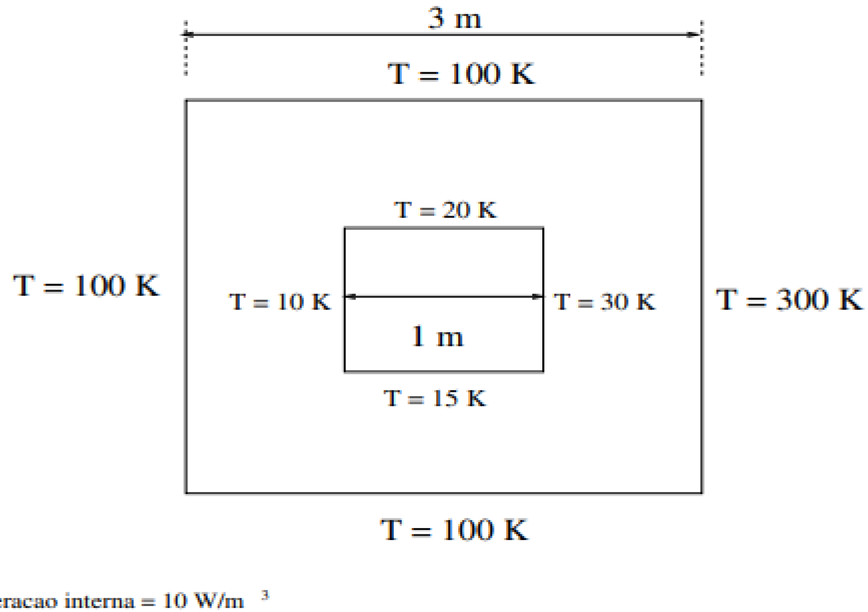


Figura 1 – Caso Teste Estudado

A implementação do trabalho em sua integridade foi realizada por meio da linguagem de programação *Python*[1] em função da sua legibililidade, facilidade de implementação e ubiquidade dentro da computação, permitindo que o código possa ser expandido e trabalho em cenários futuros, caso necessário. Para implementação foram criados dois arquivos: *tcm_trab.py*, responsável por definir as funcionalidades e métodos necessários para o trabalho, e *main.py*, ambiente em que o código fosse efetivamente utilizado. Essa divisão tem como objetivo separar as áreas de definição de funcionalidades e teste a fim de garantir uma melhor modularização do código.

No primeiro arquivo citado, tcm_trab.py, foram definidas 5 funções: *diferencas_finitas*; *animar*; *gerar_midias*;*custo_operacional* e *obter_perfis*, que serão explicadas a seguir. A função *diferencas_finitas* é onde ocorre a implementação propriamente dita do Método das Diferenças Finitas. Para o cálculo da malha foi-se estipulado $F_o = 0,25$, pois este valou demonstrou ser o melhor no quesito de estabilidade da simulação. Outro valor estipulado foi o $k = 81$ W/mK, valor este representado o material aço, já o restante dos valores foram definidos de acordo com as condições dadas. A função então desenha uma malha N X N em forma de matriz, para um dado valor N, com todas as temperaturas de acordo com a condição de contorno explicitada na figura 1, ressaltando que as temperaturas no interior da figura são $T = 0$ K. A partir disso, a matriz é atualizada iterativamente segundo a Eq.2 até atingir a condição de parada estipulada, em que a diferança entre o instante atual e o anterior seja inferior à 0.00001%., e retornando o número de iterações realizadas; a malha no seu estado final; o tempo consumido e um vetor com todos os instantes da malha.

Já a função *gerar_midias* tem o papel de gerar o vídeo da simulação conjutamente com uma imagem da malha em seu estado final. Nela é utilizado o número de iterações e o vetor de todos os instantes da

[1] *Observação: O código em sua integridade estará localizado na seçao 4.*

malaha realizadas pela *diferencas_finitas* a fim de montar o video. Primeiramente é montada a figura do último instante calculado na malha, que servirá de molde para todo o vídeo, em seguida a função *animar* será chamada para desenhar o respectivo frame do video, baseado no vetor dos instantes. Esse processo será chamado recursivamente um número de vezes igual ao número de iterações, definido acima. Por fim, o video criado é salvo[2] e a imagem utilizada é mostrada na tela.

Para a função *custo_ operacional*, um conjunto de valores nós N é dado e então a função *diferencas_finitas* e chamada para cada um deles. Em seguida os tempos de excução são armazenados e plotados em um gráfico, o qual é salvo ao final da excução.

Por fim, a função *obter_perfis* tem como objetivo criar os cinco perfis de temperatura para valores de x e y estipulados, a fim de observar a maior variedade de comportamentos dentro da malha. Para cada um dos valores de x ou y estipulados, a função cria extrai a respectiva coluna ou linha da malha em estado permanente obtida do método *diferencas_finitas*. A partir disso, os respectivos gráficos são então plotados e salvos.

2.2 Análise dos Resultados Simulados

2.2.1 Custo Operacional por Número de Nós

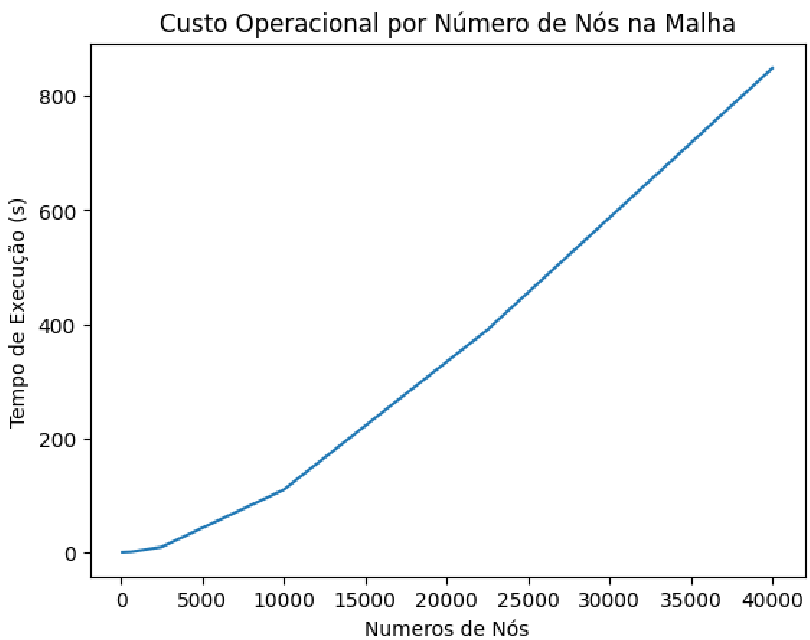
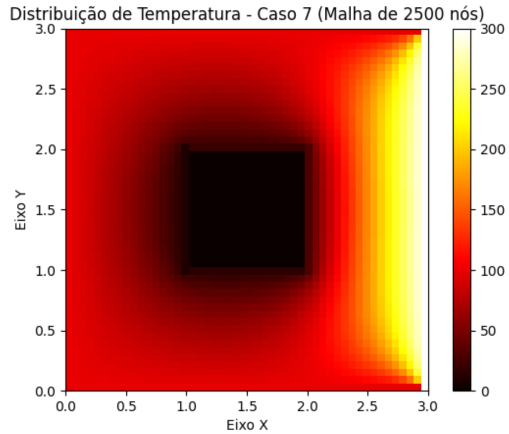


Figura 2 – Custo Operacional (s) por Número de Nós

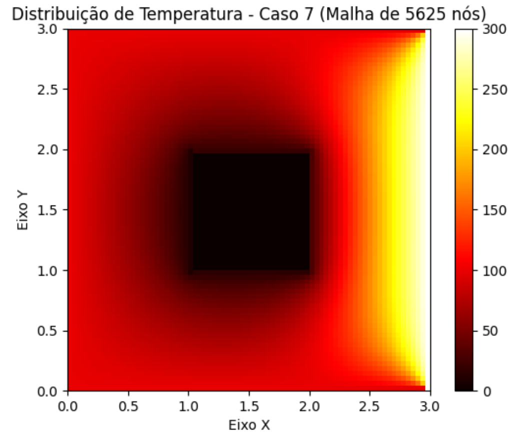
O gráfico acima foi gerado a partir da análise do utilizado nas simulações para malhas de 100, 625, 2500, 10000 e 40000 nós. A partir deste gráfico é possível observar a eficiência do método utilizado, que cresce linearmente com o número nós da malha. Disso pode-se observar que para pequenos valores de nós, o método é extremamente eficiente porém para grandes malhas o gasto em tempo de processamento será elevadoe poderá ser necessária a utilização de um método mais eficiente.

[2] *Observação: O vídeo gerado foi acelerado com a utilização de softwares externos a fim de se encaixar nos padrões estipulados pelo trabalho.*

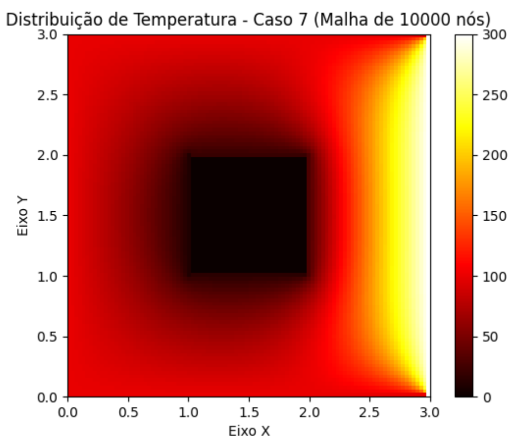
2.2.2 Simulação para Diferentes Malhas



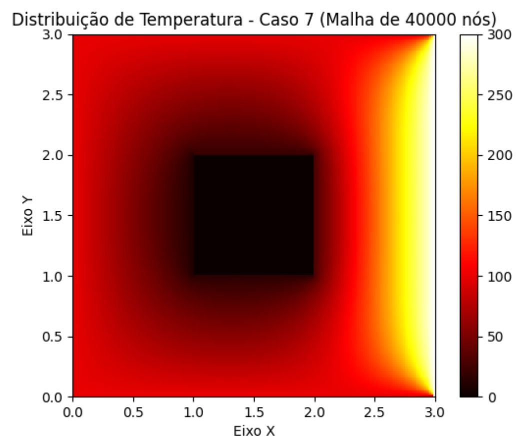
(a) Malha de 50X50 Nós



(b) Malha de 75X75 Nós



(c) Malha de 100X100 Nós



(d) Malha de 200X200 Nós

Figura 3 – Simulação do Caso para Diferentes Nós na Malha

A figura acima apresenta a simulação do caso estudado para diferentes tamanhos de malha. Como esperado, malhas maiores apresentam uma qualidade superior e um gradiente de temperatura muito mais suave em relação a malhas menores. No entanto, como visto na subseção acima o custo operacional cresce de acordo com o tamanho da malha, então dependendo da aplicação ou contexto inserido, é necessário se ponderar acerca do balanço tempo-qualidade.

2.2.3 Perfis de Temperatura

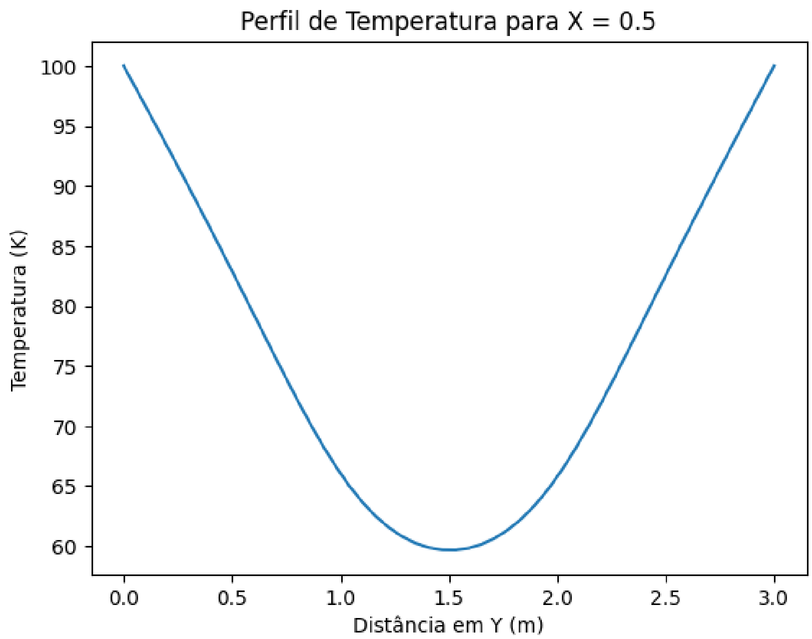


Figura 4 – Perfil de Temperatura para X = 0,5 m

O primeiro perfil de temperatura, mostrado acima, representa a variação da temperatura ao longo do eixo y para $x = 0,5$ m e começando na temperatura do fundo $T = 100$ K. Nele é possível perceber um decaimento da temperatura ao se aproximar do buraco, entre $y = 1$ m e $y = 2$ m. Esse decaimento se dá em função da condição de contorno das extremidades do buraco, que se mantém em temperatura constante. Esse fato força que as temperaturas a nessa região sejam menores que do resto do corpo, demonstrado pelo subsequente aumento da temperatura ao sair da área de influência da extremidade do buraco, retornando a temperatura do topo de $T = 100$ K. O segundo perfil,explicitado acima, representa a variação da temperatura ao longo

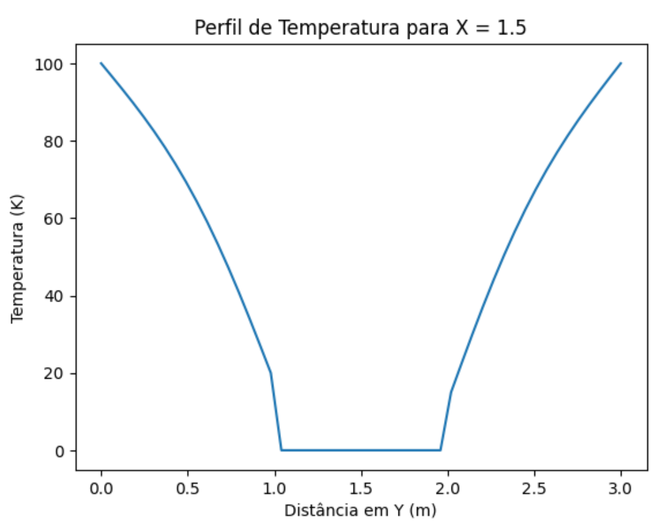


Figura 5 – Perfil de Temperatura para X = 1,5 m do eixo y para $x = 1,5$ m e começando na temperatura do fundo $T = 100$ K. Nele, é possível perceber um comportamento similiar ao perfil anterior, porém com uma abrupta mudança de temperatura entre $y = 1$ m e $y = 2$ m. Isso se dá pelo fato de que em $x = 1,5$ m a análise intercepta o interior do buraco, de tempertaura constante e $T = 0$ K. Pelo fato dessa temperatura ser arbitrariamente definda, ela não pertence adistribuição padrão da temperatura no corpo e causa essa queda abrupta no gráfico. O terceiro perfil, mostrado acima,

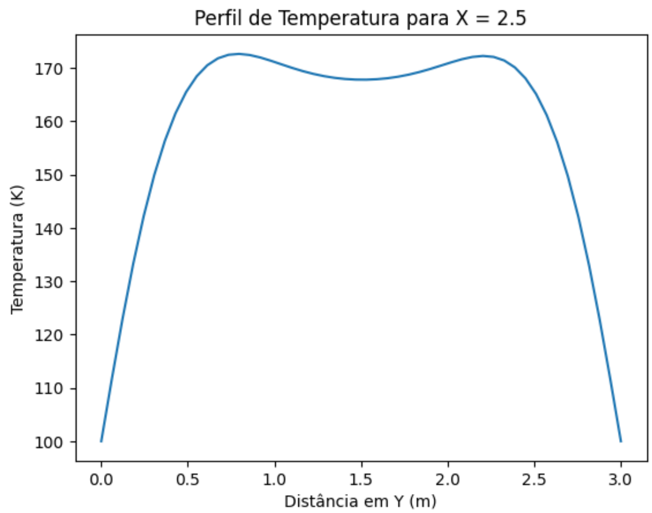


Figura 6 – Perfil de Temperatura para X = 2,5 m

representa a variação da temperatura ao longo do eixo y para $x = 2,5$ m e começando na temperatura do fundo $T = 100$ K. Nele, há uma influência muito maior da temperatura da borda do corpo, de valor $T = 300$ K. Em função disso é possível perceber um comportamento oposto ao do primeiro perfil, com um aumento da temperatura até um ponto de máximo e depois seu decréscimo até ao valor do topo, de $T = 100$K Além disso é possível perceber um pequeno decaimento da temperatura em $y = 1,5$ em função da influência da borda do burcao que tem $T = 30$ K e força que temperatua nesses pontos se tornem menores. Inclusive, esse comportamento é perceptível no esquema de cores da simulações da Fig.3.

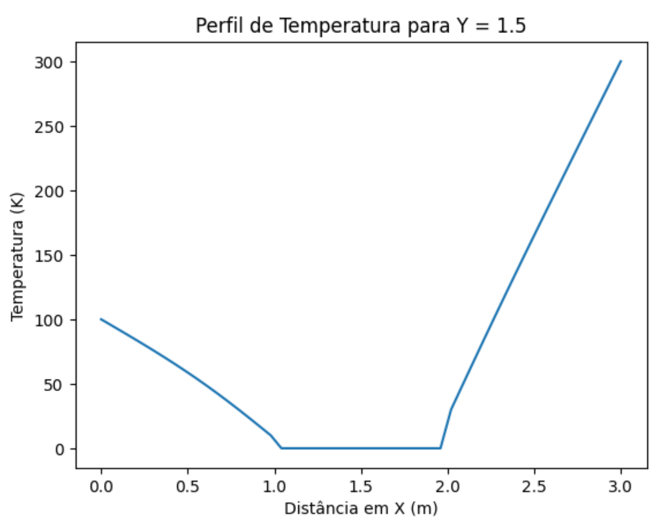


Figura 7 – Perfil de Temperatura para Y = 1,5 m

O quarto perfil, mostrado acima, representa a variação da temperatura ao longo do eixo x para $y = 1,5$ m e começando na temperatura da esquerda $T = 100$ K. Nele é possível notar a diminuição da temperatura, em função do cotorno do buraco que está em temperatura constante e menor que a do resto do corpo. Após isso há uma abruta queda para $T = 0$ K, que se mantém até $x = 2,0$ m. Isso ocorre em função do interior do buraco se manter a temperatura constante e não trocar calor com o corpo. Em seguida, há um aumento da temperatura qu se mantém até chegar a temperatura da outra extremidade, de valor $T = 300$ K.

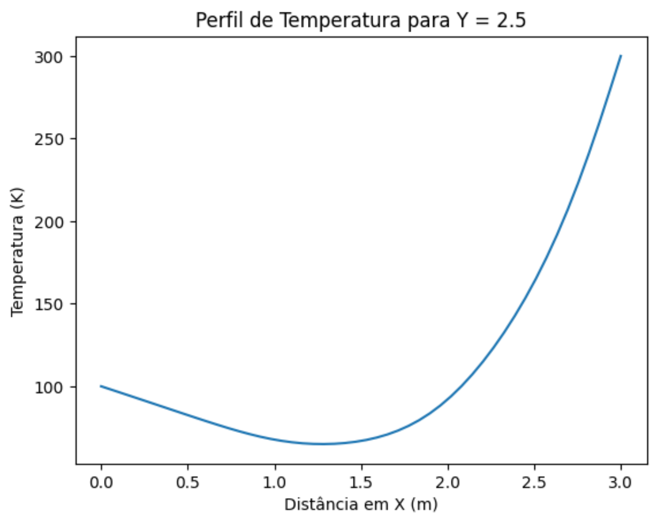


Figura 8 – Perfil de Temperatura para Y = 2,5 m

Por último, o quinto perfil, explicitado acima, representa a variação da temperatura ao longo do eixo x para $y = 2,5$ m e começando na temperatura da esquerda $T = 100$ K. Para este valor de y, pode-se perceber sua continuidade por toda a extensão do intervalo, sem aumentou ou quedas abruptas de temperatura. Isso se dá pelo fato de não haver contato direto com o buraco ou suas extremidades, reponsável por causar as variações abruptas da temperatura. No entanto, ainda é possível perceber sua influência na distribuição de temperatura de todo corpo, caracterizado pelo decrescimento da temperatura até $x = 1,5$ m. Após este intervalo, a influência da borda direita de $T = 300$ K torna-se predominante na distribuição de temperaturas, evidenciado pelo grande aumento de temperatura até a temperatura da própria borda direita.

3 Conclusão

A partir da aplicação do Método das Diferenças Finitas, foi possível realizar um estudo da condução do calor dentro de um corpo ao longo do espaço e do tempo. Neste estudo foi possíve analisar o impacto das condições de contorno na distribuição geral das temperaturas ao longo do espaço, por meio dos perfis de temperatua. Já o vídeo gerado serviu como ferramenta para se obter compreensão do processo de condução ao longo do tempo.

Outro aspecto analisado foi o próprio Método das Diferenças Finitas, por meio de análise de qualidade para o número de nós nas malhas e seu custo computacional. Disso, foi possível concluir que a qualidade da simulação está diretamente ligada ao número de nós utilizados, porém seu custo computacional de tempo cresce linearmente com o tamanho da malha. Por isso, é sempre necessário ponderar e selecionar o melhor valor de nós a fim de obter resultados em qualidade de simulação e temppo de execução dentro do razoável do conttexto empregado.

4 Código-Fonte

4.1 tcm_trab.py

```
import numpy as np
import time
import math
import copy
import matplotlib.pyplot as plt
import matplotlib.animation as animation


n_Fourier = 0.25 #Número de Fourier definido a fim de manter a estabilidade da simulação

geracao_interna = 10 #Valor estipulado no trabalho

k = 81 #condutividade termica do aço

tamanho = 3 #comprimento do objeto simulado, em metros


#Funcão utilizada para desenhar o i-ésimo frame da animação
def animar(i,animacao):
  quadro = plt.imshow(animacao[i], cmap='hot', interpolation='nearest',
      extent=[0,tamanho,0,tamanho])
  return quadro

#Função que realiza o método das diferenças finitas
def diferencas_finitas(numero_nos):
   inicio = time.process_time()

   delta_x = tamanho/(numero_nos)
   x_gap = math.floor(numero_nos/3) #Coordenada X da matriz da malha obde se encontra o gap no
       caso 7
   y_gap = math.floor(numero_nos/3) #Coordenada Y da matriz da malha obde se encontra o gap no
       caso 7
   intervalo_gap_x = round(numero_nos/3) #Intervalo de Coordenadas em X no qual se encontra o gap
       do caso 7
   intervalo_gap_y = round(numero_nos/3) #Intervalo de Coordenadas em X no qual se encontra o gap
       do caso 7

   matriz = np.zeros([numero_nos,numero_nos], dtype = float)

   matriz[0, 1:numero_nos] = 100 #Condição de Contorno do Topo
   matriz[numero_nos - 1, 1:numero_nos] = 100 #Condição de Contorno do Fundo
   matriz[:numero_nos, 0] = 100 #Condição de Contorno da Esquerda
   matriz[:numero_nos, numero_nos - 1] = 300 #Condição de Contorno da Direita
```

```python
    matriz[y_gap, x_gap :(x_gap + intervalo_gap_x)] = 20 #Condição de Contorno do Topo do Gap
    matriz[y_gap + intervalo_gap_y , x_gap: (x_gap + intervalo_gap_x)] = 15 #Condição de Contorno do
        Fundo do Gap
    matriz[y_gap : (y_gap + intervalo_gap_y + 1), x_gap] = 10 #Condição de Contorno da Esquerda do
        Gap
    matriz[y_gap : (y_gap + intervalo_gap_y + 1), (x_gap + intervalo_gap_x)] = 30 #Condição de
        Contorno da Direita do Gap
    matriz[y_gap + 1 : (y_gap + intervalo_gap_y), x_gap + 1 : (x_gap + intervalo_gap_x)] = 0
        #Condição de Contorno do interior do Gap


    matriz_atual = copy.deepcopy(matriz)
    matriz_anterior = copy.deepcopy(matriz)
    parada = False
    quadros = 0
    animacao = []

    while parada != True:
        quadros += 1
        animacao.append(copy.deepcopy(matriz_atual))
        if(quadros%100 == 0):
            print(quadros)
        for x in range(numero_nos):
            for y in range(numero_nos):
                if (x == 0 or y == 0) or (x == (numero_nos - 1) or y == (numero_nos - 1)):
                    continue
                elif (x >= x_gap and x <= (x_gap + intervalo_gap_x)) and (y >= y_gap and y <= (y_gap +
                    intervalo_gap_y)):
                    continue
                matriz_anterior[y,x] = copy.deepcopy(matriz_atual[y,x])
                matriz_atual[y,x] = n_Fourier*(matriz_atual[y,x+1] + matriz_atual[y,x-1] +
                    matriz_atual[y+1,x] + matriz_atual[y-1,x])
                + (1 - 4*n_Fourier)*matriz_atual[y,x]
                + (geracao_interna*(delta_x**2)/k)

                if ((abs(matriz_atual[y,x] - matriz_anterior[y,x])/matriz_atual[y,x]) <= 0.0000001):
                #Condição de Parada (Diferença entre dois pontos do estado atual e do anterior menor que
                    0.00001%)
                    parada = True
    tempo_consumido = time.process_time() - inicio
    print('Total de Frames: ' + (str)(quadros))
    return animacao,quadros,tempo_consumido,matriz_atual


#Função que gera o vídeo da simulação e mostra a imagem do estado permanente
def gerar_midias(numero_nos):
    resultado = diferencas_finitas(numero_nos)
    animacao = resultado[0]
```

```
    quadros = resultado[1]
    figura, eixos = plt.subplots()
    eixos.set(title = 'Distribuição de Temperatura - Caso 7 (Malha de ' + (str)(numero_nos**2) + '
        nós)', xlabel = 'Eixo X', ylabel = 'Eixo Y' )
    imagem = eixos.imshow(animacao[quadros], cmap='hot', interpolation='nearest',
        extent=[0,tamanho,0,tamanho])
    figura.colorbar(imagem)
    video = animation.FuncAnimation(figura,animar,frames=quadros,fargs=(animacao,))
    writermp4 = animation.FFMpegFileWriter(fps=30, bitrate=1800)
    video.save('sim.mp4', writer=writermp4)
    plt.show()

#Função que define o custo operacional, em s, para um série de valores de nós
def custo_operacional(valores_nos):
    tempos = []
    malhas = []

    for i in valores_nos:
        tempos.append(diferencas_finitas(i)[2])
    for j in valores_nos:
        malhas.append(j**2)

    plt.plot(malhas,tempos)
    plt.title('Custo Operacional por Número de Nós na Malha')
    plt.xlabel('Numeros de Nós')
    plt.ylabel('Tempo de Execução (s)')
    plt.savefig('CustoOperacional.png', bbox_inches = 'tight')

#Função que obtém os perfis de temperatura para X = 0.5,1.5 e 2.5 e Y = 1.5 e 2.5
def obter_perfis(numero_nos):
    y_observado = [math.floor(numero_nos/(tamanho/1.5)), math.floor(numero_nos/(tamanho/2.5))]
        #Valores de x defindos para análise
    x_observado = [(math.floor(numero_nos/(tamanho/0.5))),
        (math.floor(numero_nos/(tamanho/1.5))),math.floor(numero_nos/(tamanho/2.5))] #Valores de y
        definidos para análise
    valores_y = (1.5,2.5)
    valores_x = (0.5,1.5,2.5)
    intervalo = np.linspace(0,tamanho, num = numero_nos)
    estado_permanente = diferencas_finitas(numero_nos)[3]

    for i in range(len(x_observado)):
        temperaturas = np.array(estado_permanente[:,x_observado[i]])
        plt.plot(intervalo,temperaturas)
        plt.title('Perfil de Temperatura para X = ' + (str)(valores_x[i]))
        plt.xlabel('Distância em Y (m)')
        plt.ylabel('Temperatura (K)')
        plt.savefig('PerfilTempX'+ (str)(valores_x[i]) + '.png', bbox_inches = 'tight')
        plt.close('all')
```

```python
  for i in range(len(y_observado)):
    temperaturas = np.array(estado_permanente[y_observado[i],:])
    plt.plot(intervalo,temperaturas)
    plt.title('Perfil de Temperatura para Y = ' + (str)(valores_y[i]))
    plt.xlabel('Distância em X (m)')
    plt.ylabel('Temperatura (K)')
    plt.savefig('PerfilTempY'+ (str)(valores_y[i]) + '.png', bbox_inches = 'tight')
    plt.close('all')
```

4.2 main.py

```python
import tcm_trab as tcm
numero_nos = 200 #Número N de nos para criação de uma malha [N,N]

valores = (10,25,50,100,150,200) #Valores de nos utilizados para analise do custo operacional

tcm.custo_operacional(valores)
tcm.gerar_midias(numero_nos)
tcm.obter_perfis(numero_nos)
```