INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

Nícolas Silva Fernandes Lopes - GU3011925

Pedro Henrique Braga Cavalcante - GU3011496

Mateus Lucio de Souza Belletti - GU3011585

Computação para Automação

Trabalho Final

Professor: Delfim Pinto Carneiro Junior

Guarulhos

2022

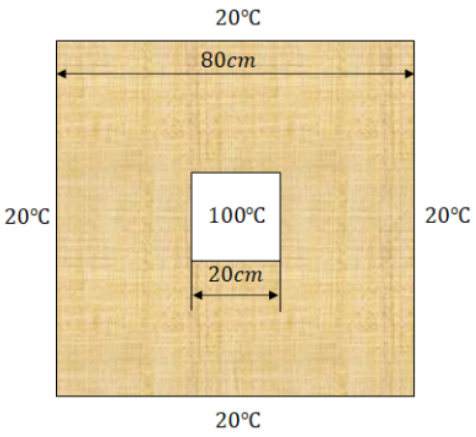
Sumário

Repositório do Trabalho

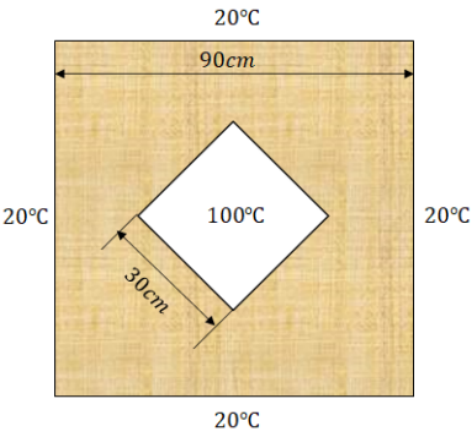
Todos os códigos e imagens podem ser encontrados em um repositório no github
link do repositório: github.com/thenickz/computacao_trabalho_final

Problema Proposto

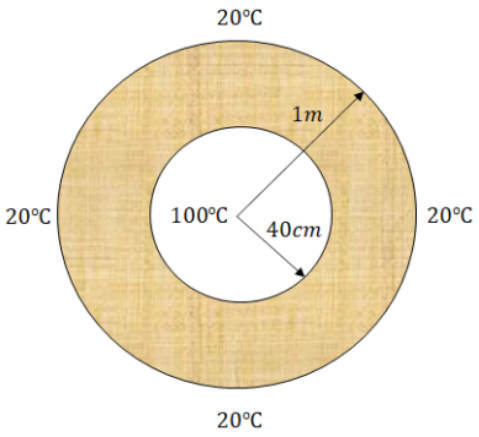
Considerar três chapas planas conforme Figura a seguir com coeficiente de difusibilidade $\alpha = 0{,}016\ m^2/s$ e temperatura inicial de $30°C$. Suas bordas externas são mantidas à temperatura de $20°C$ e suas bordas internas à temperatura de $100°C$. Determinar a distribuição de temperatura nas chapas *10, 20, 30 segundos* utilizando-se o Método de Euler Explícito considerando uma condição de estabilidade $s = 0{,}1$ no cálculo de Δt. Escolher uma grade de discretização adequada e apresentar os resultados graficamente.



chapa 1



chapa 2



chapa 3

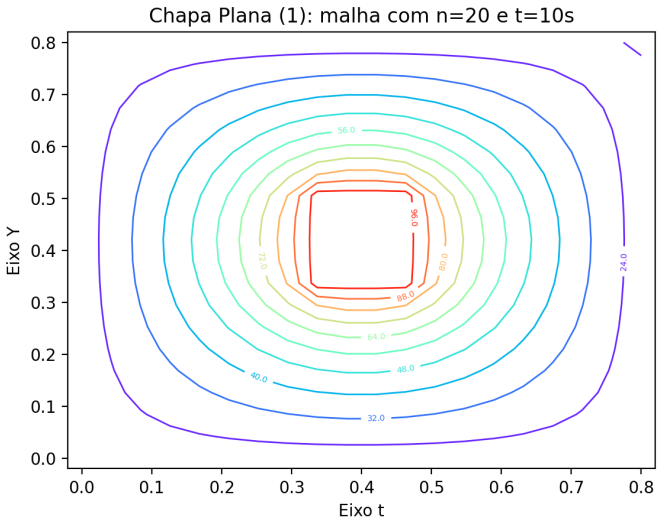
1. Chapa

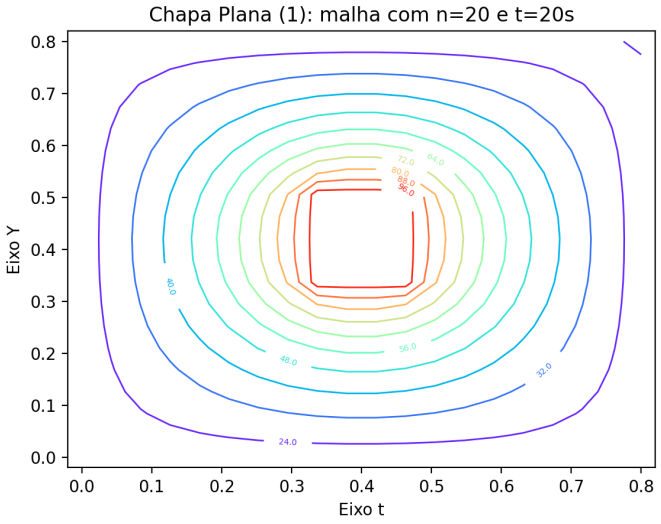
1.1 Código

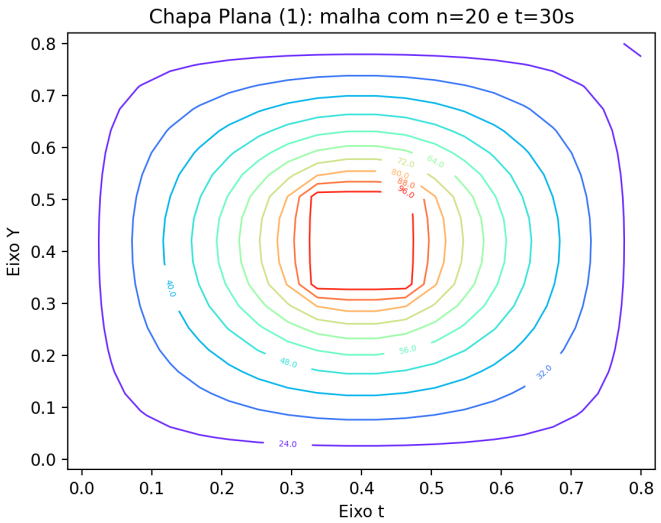
```
# código da chapa 1
import matplotlib.pyplot as plt
import numpy as np
# configurando variáveis
tempo_total = 30
Te = 20 # temperatura na extremidade
Ti = 30 # temperatura inicial da chapa
Tb = 100 # temperatura do buraco central
L = 0.8 # tamanho do lado da chapa
n = 20 # valor de n da malha
a = 0.016
s = 0.1
dx = L/n
dy = dx
dt = s/(a*(1/dx**2 + 1/dy**2))
# configurando vetores e matriz
x = np.linspace(0, L, n)
y = np.linspace(0, L, n)
T = Ti * np.ones([n, n]) # Ti deixa o valor inicial em toda a matriz
# aplicando as temperaturas nas extremidades
T[0:n-1, 0] = Te
T[0:n-1, n-1] = Te
T[0, 0:n-1] = Te
T[n-1, 0:n-1] = Te
# aplicando as temperaturas no buraco da chapa
meio = int(n/2) # metade do tamanho da chapa
tamanho = int(meio/4) # tamanho do buraco
for k in range(int(n/4)):
    T[meio-tamanho+k, meio-tamanho:meio+tamanho] = 100

# iniciar método de euler explícito
imag_10s = True
imag_20s = True
t = 0
while True:
    To = T.copy() # salva os valores da matriz T
    for j in range(1, n-1):
        for i in range(1, n-1):
```

```
            T[i, j] = To[i, j] + dt*a*(((To[i-1, j] - 2*To[i, j] +
                      To[i+1, j])/dx**2) + ((To[i, j-1] - 2*To[i,j] +
                      To[i, j+1])/dy**2))
    # aplicando as temperaturas no buraco da chapa, pois são constantes
    for k in range(int(n/4)):
        T[meio-tamanho+k, meio-tamanho:meio+tamanho] = Tb
    t += dt
    # verifica se chegou nos tempos para salvar o estado atual e depois
    # plotar
    if t >= 10 and imag_10s:
        imag_10s = False
        T10 = T.copy()
    elif t >= 20 and imag_20s:
        imag_20s = False
        T20 = T.copy()
    elif t >= 30:
        T30 = T.copy()
        # saindo do loop pois t já chegou em 30s
        break
# por ser 3 gráficos decidi salvar como imagem para visualizar todos no
# final
# preparando plot
X, Y = np.meshgrid(x, y)
# função para facilitar a plotagem
def salvar_grafico(arquivo_nome, matrix, tempo):
    ax = plt.subplot()
    plt.xlabel('Eixo t')
    plt.ylabel('Eixo Y')
    plt.xlim([-0.02, 0.82])
    plt.ylim([-0.02, 0.82])
    plt.title(f'Chapa Plana (1): malha com n={n} e t={tempo}s')
    grafico = ax.contour(X, Y, matrix, 12, vmin=Te, vmax=Tb,
                         linewidths=1, cmap='rainbow')
    ax.clabel(grafico, fontsize=5, fmt ='%1.1f')
    plt.savefig(f'{arquivo_nome}{tempo}.png', dpi=200)
    ax.clear()
# salvando as imagens
salvar_grafico('chapa_01_t', T10, 10)
salvar_grafico('chapa_01_t', T20, 20)
salvar_grafico('chapa_01_t', T30, 30)
```

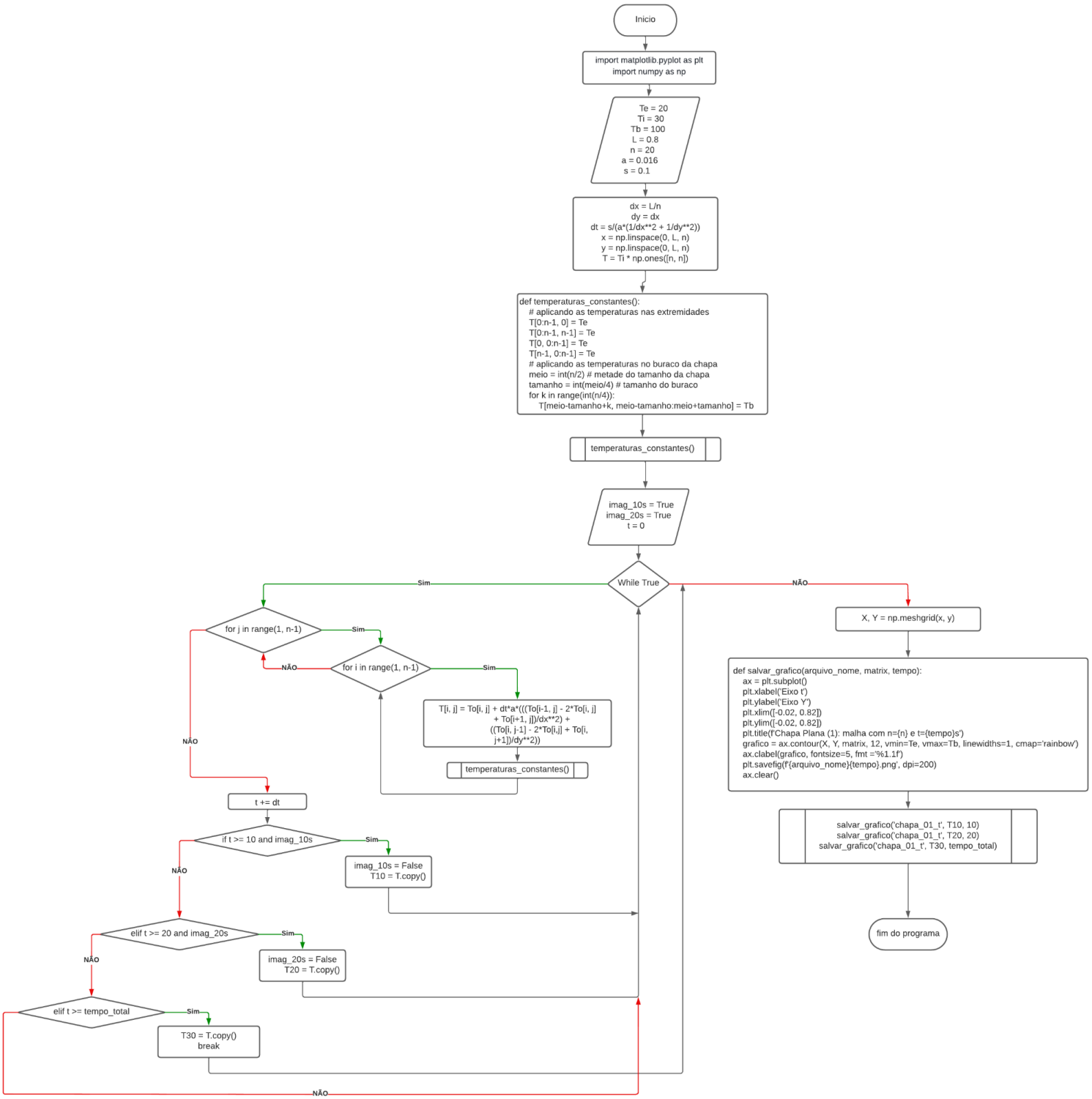
1.2 Imagens


Chapa Plana (1): malha com n=20 e t=10s
Eixo Y
Eixo t
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
96.0
88.0
80.0
72.0
64.0
56.0
48.0
40.0
32.0
24.0


Chapa Plana (1): malha com n=20 e t=20s
Eixo Y
Eixo t
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
96.0
88.0
80.0
72.0
64.0
56.0
48.0
40.0
32.0
24.0


Chapa Plana (1): malha com n=20 e t=30s
Eixo Y
Eixo t
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
96.0
88.0
80.0
72.0
64.0
56.0
48.0
40.0
32.0
24.0

1.3 Fluxograma



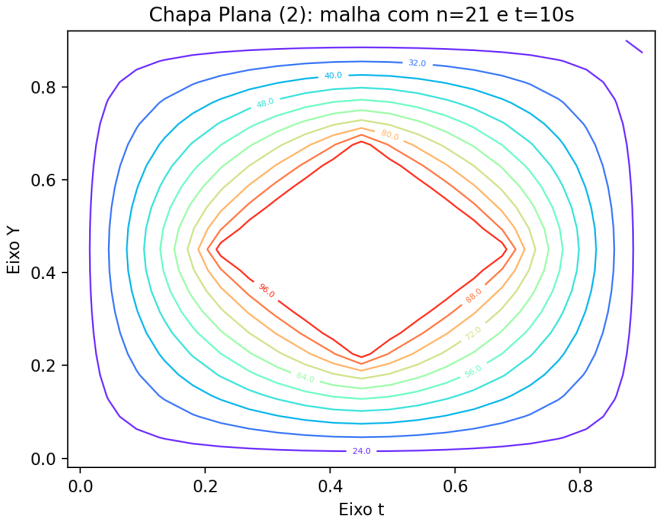
2. Chapa

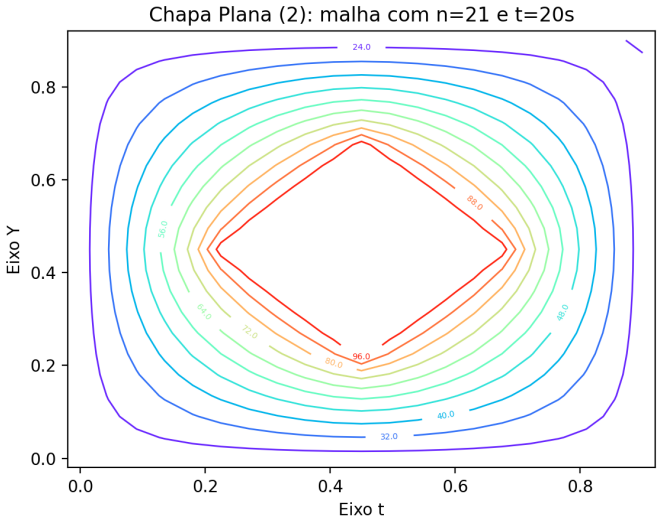
2.1 Código

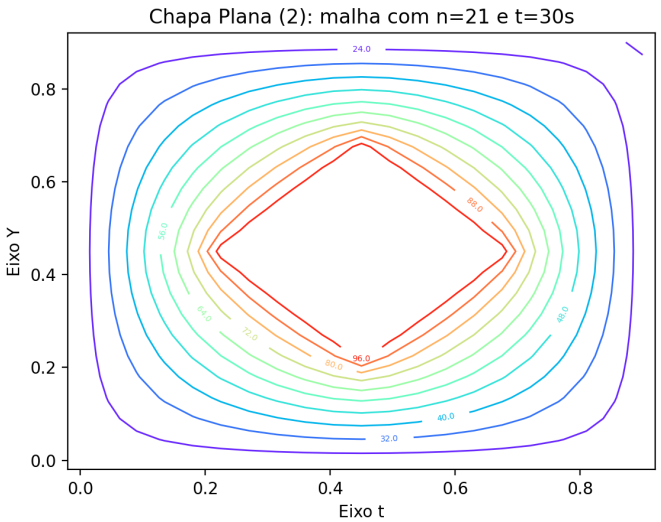
```
# código da chapa 2
import matplotlib.pyplot as plt
import numpy as np
# configurando variáveis
tempo_total = 30
Te = 20 # temperatura na extremidade
Ti = 30 # temperatura inicial da chapa
Tb = 100 # temperatura do buraco central
L = 0.9 # tamanho do lado da chapa
n = 21 # valor de n da malha
a = 0.016
s = 0.1
dx = L/n
dy = dx
dt = s/(a*(1/dx**2 + 1/dy**2))
# configurando vetores e matriz
x = np.linspace(0, L, n)
y = np.linspace(0, L, n)
T = Ti * np.ones([n, n])#Ti deixa o valor inicial em toda a matriz
# aplicando as temperaturas nas extremidades
T[0:n-1, 0] = Te
T[0:n-1, n-1] = Te
T[0, 0:n-1] = Te
T[n-1, 0:n-1] = Te
# aplicando as temperaturas no buraco da chapa
def meio_da_chapa():
    meio = int(n/2) # metade do tamanho da chapa
    tamanho = int(meio*2/3) # tamanho do buraco
    T[meio, meio:meio+tamanho] = Tb
    for k in range(tamanho):
        T[meio-tamanho+k, meio:meio+k] = Tb
        T[meio-tamanho+k, meio:meio-k:-1] = Tb
        T[meio+tamanho-k, meio:meio+k] = Tb
        T[meio:meio+k, meio-tamanho+k:meio] = Tb
meio_da_chapa()
# iniciar método de euler explícito
imag_10s = True
imag_20s = True
t = 0
while True:
    To = T.copy() # salva os valores da matriz T
    for j in range(1, n-1):
```

```
        for i in range(1, n-1):
            T[i, j] = To[i, j] + dt*a*(((To[i-1, j] - 2*To[i, j] +
To[i+1, j])/dx**2) + ((To[i, j-1] - 2*To[i,j] + To[i, j+1])/dy**2))
            # aplicando as temperaturas no buraco da chapa, pois
            # são constantes
            meio_da_chapa()
    t += dt
    # verifica se chegou nos tempos para salvar o estado atual e
    # depois plotar
    if t >= 10 and imag_10s:
        imag_10s = False
        T10 = T.copy()
    elif t >= 20 and imag_20s:
        imag_20s = False
        T20 = T.copy()
    elif t >= tempo_total:
        T30 = T.copy()
        # saindo do loop pois t já chegou em 30s
        break
# por ser 3 gráficos decidi salvar como imagem para visualizar todos no
final
# preparando plot
X, Y = np.meshgrid(x, y)
# função para facilitar a plotagem
def salvar_grafico(arquivo_nome, matrix, tempo):
    ax = plt.subplot()
    plt.xlabel('Eixo t')
    plt.ylabel('Eixo Y')
    plt.xlim([-0.02, 0.92])
    plt.ylim([-0.02, 0.92])
    plt.title(f'Chapa Plana (2): malha com n={n} e t={tempo}s')
    grafico = ax.contour(X, Y, matrix, 12, vmin=Te, vmax=Tb,
                         linewidths=1, cmap='rainbow')
    ax.clabel(grafico, fontsize=5, fmt ='%1.1f')
    plt.savefig(f'{arquivo_nome}{tempo}.png', dpi=200)
    ax.clear()


# salvando as imagens
salvar_grafico('chapa_02_t', T10, 10)
salvar_grafico('chapa_02_t', T20, 20)
salvar_grafico('chapa_02_t', T30, 30)
```

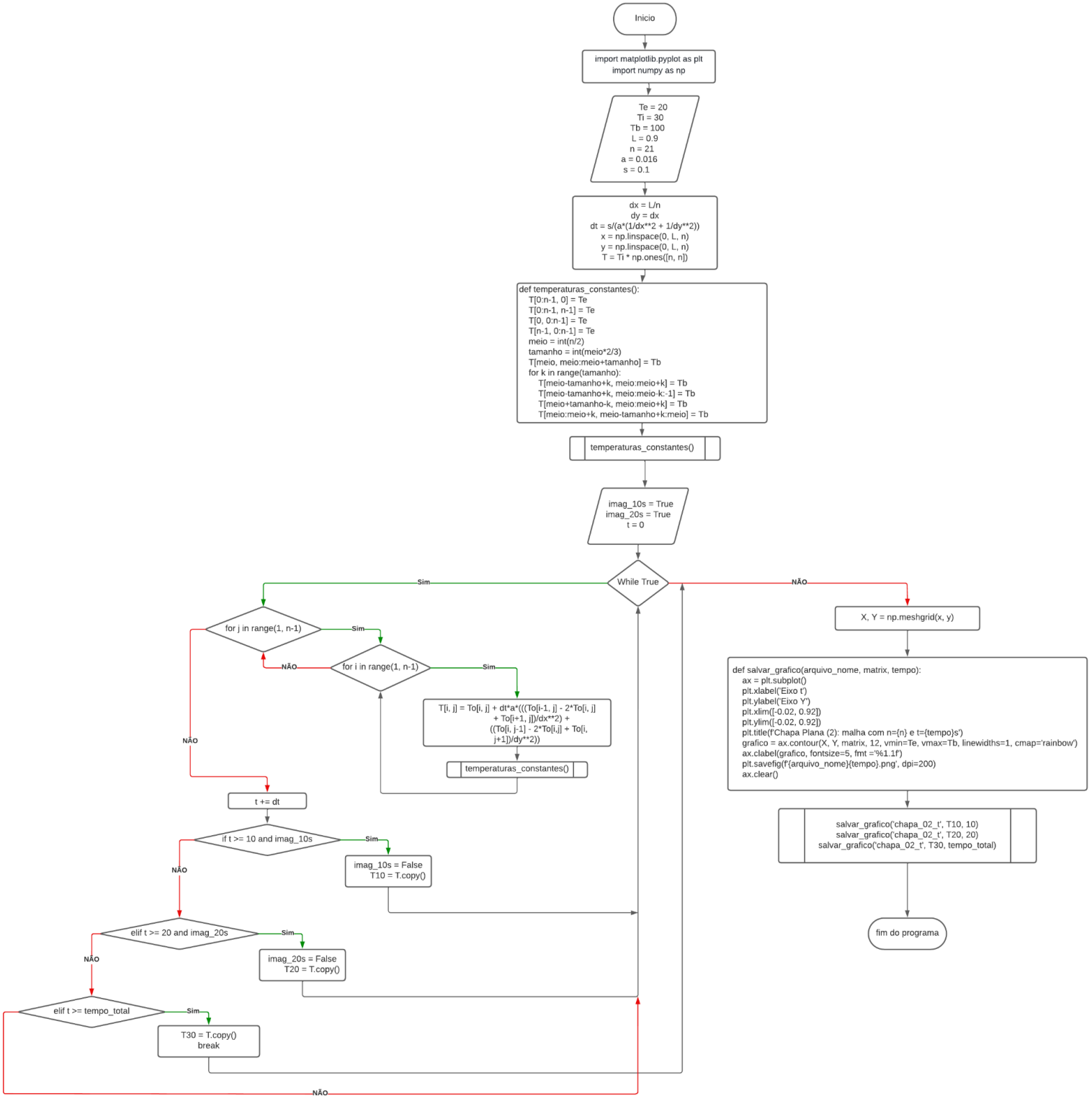
2.2 Imagens


Chapa Plana (2): malha com n=21 e t=10s
Eixo Y
Eixo t
0.0
0.2
0.4
0.6
0.8
32.0
40.0
48.0
80.0
96.0
88.0
72.0
56.0
64.0
24.0


Chapa Plana (2): malha com n=21 e t=20s
Eixo Y
Eixo t
0.0
0.2
0.4
0.6
0.8
24.0
88.0
96.0
48.0
40.0
32.0


Chapa Plana (2): malha com n=21 e t=30s
Eixo Y
Eixo t
0.0
0.2
0.4
0.6
0.8
24.0
88.0
96.0
48.0
40.0
32.0

2.3 Fluxograma



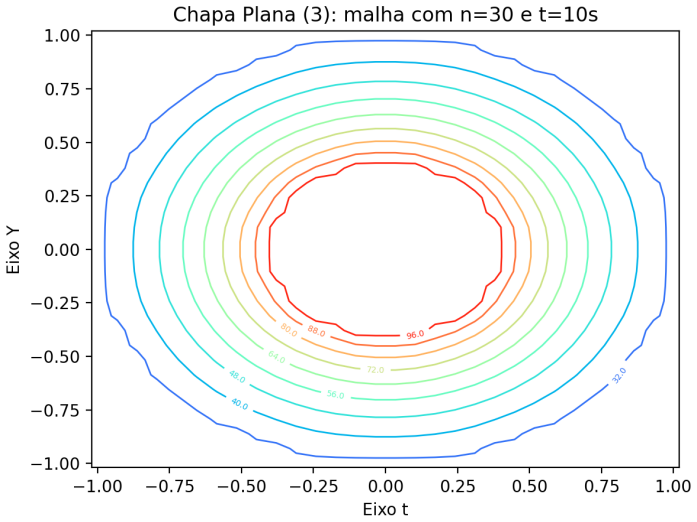
3. Chapa

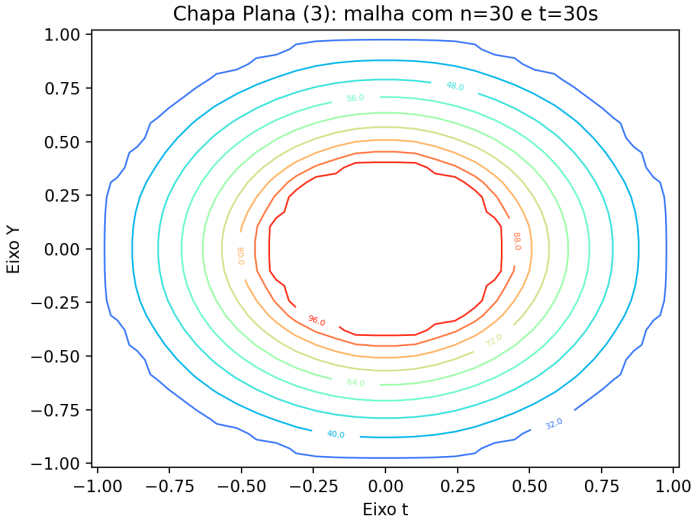
3.1 Código

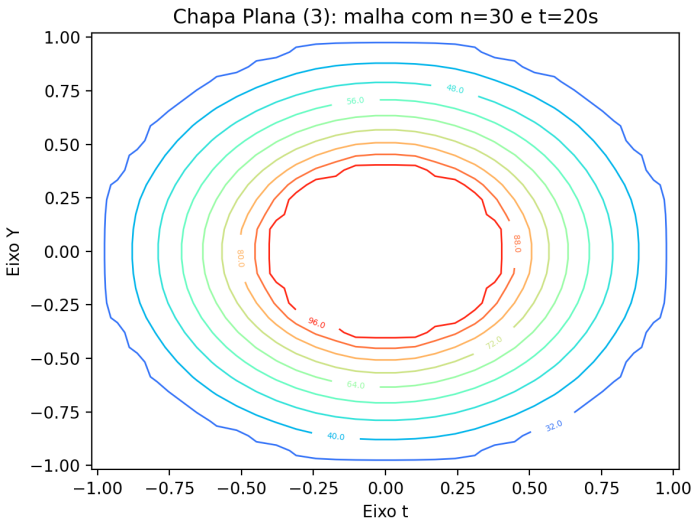
```
# código da chapa 3
import matplotlib.pyplot as plt
import numpy as np
# configurando variáveis
tempo_total = 30
Te = 20 # temperatura na extremidade
Ti = 30 # temperatura inicial da chapa
Tb = 100 # temperatura do buraco central
L = 1 # tamanho do lado da chapa
n = 30 # valor de n da malha
a = 0.016
s = 0.1
dx= 2*L/n
dy= dx
dt = s/(a*(1/dx**2 + 1/dy**2))
# criando os vetores e a matriz
x = np.linspace(-1, L, n)
y = np.linspace(-1, L, n)
T = Ti * np.ones([n,n])
# iniciar método de euler explícito
imag_10s = True
imag_20s = True
re = L # raio da chapa
ri = L*0.4 # raio do buraco
t = 0
while True:
    To = T.copy() # salva os valores da matriz T
    for j in range(1, n-1):
        for i in range(1, n-1):
            # aplicando as temperaturas no buraco da chapa, pois são
            # constantes
            r = np.sqrt(y[i]**2  + x[j]**2)
            if r < re and r > ri:
                T[i, j] = To[i, j] + dt*a*(((To[i-1, j] - 2*To[i, j] +
                          To[i+1, j])/dx**2) + ((To[i, j-1] - 2*To[i,j]+
                          To[i, j+1])/dy**2))
            if r >= re:
                T[i, j] = Ti
            if r <= ri:
                T[i, j] = Tb
    t += dt
    # verifica se chegou nos tempos para salvar o estado atual e depois
```

```
    # plotar
    if t >= 10 and imag_10s:
        imag_10s = False
        T10 = T.copy()
    elif t >= 20 and imag_20s:
        imag_20s = False
        T20 = T.copy()
    elif t >= tempo_total:
        T30 = T.copy()
        # saindo do loop pois t já chegou em 30s
        break
# por ser 3 gráficos decidi salvar como imagem para visualizar todos no
# final
# preparando plot
X, Y = np.meshgrid(x, y)
# função para facilitar a plotagem
def salvar_grafico(arquivo_nome, matrix, tempo):
    ax = plt.subplot()
    plt.xlabel('Eixo t')
    plt.ylabel('Eixo Y')
    plt.xlim([-1.02, 1.02])
    plt.ylim([-1.02, 1.02])
    plt.title(f'Chapa Plana (3): malha com n={n} e t={tempo}s')
    grafico = ax.contour(X, Y, matrix, 9, vmin=Te, vmax=Tb,
                         linewidths=1, cmap='rainbow')
    ax.clabel(grafico, fontsize=5, fmt ='%1.1f')
    plt.savefig(f'{arquivo_nome}{tempo}.png', dpi=200)
    ax.clear()
# salvando as imagens
salvar_grafico('chapa_03_t', T10, 10)
salvar_grafico('chapa_03_t', T20, 20)
salvar_grafico('chapa_03_t', T30, 30)
```

3.2 Imagens


Chapa Plana (3): malha com n=30 e t=10s
Eixo Y
Eixo t
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
96.0
88.0
80.0
72.0
64.0
56.0
48.0
40.0
32.0


Chapa Plana (3): malha com n=30 e t=30s
Eixo Y
Eixo t
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
96.0
88.0
80.0
72.0
64.0
56.0
48.0
40.0
32.0


Chapa Plana (3): malha com n=30 e t=20s
Eixo Y
Eixo t
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
96.0
88.0
80.0
72.0
64.0
56.0
48.0
40.0
32.0

3.3 Fluxograma