Trabalho 1: Distribuição de Calor

Curso: Bacharelado em Ciências de Computação

Disciplina: SME104 - Cálculo Numérico

Entrega: 23/03/2018

Professor: Fabricio Simeoni de Sousa

Aluno:

- Leonardo Meireles Murtha Oliveira: 4182085

Sumário

1. Descrição do Problema

Dada uma placa quadrada de lado $1\,m^2$, já com os valores de temperatura estabelecidos na fronteira, fazer uma rotina em *Python* que calcule visualize a distribuição de temperaturas nesta placa usando o Método de Gauss-Seidel e um grid de resolução $n\,x\,n$, considerando que a temperatura de um ponto é a média dos quatros pontos vizinhos a ele e que a temperatura dos pontos onde as fronteiras se encontram é a média entre as temperaturas das mesmas.

2. Método Utilizado

O método utilizado para a resolução do sistema linear de equações foi o método de Gauss-Seidel é um método iterativo bastante baseado no método de Jacobi. Ademais, esse método utiliza de operações matriciais para encontrar uma solução aproximada usando um chute inicial de forma iterativa, o critério de convergência utilizado é o de Sassenfeld derivado do critério das linhas.

A fórmula utilizada para a implementação do método é $x^{(k+1)} = Cx^{(k)} + g$ onde, $C = -L^{-1}Rx$ e $g = L^{-1}b$. L e R são matrizes triangulares feitas a partir da matriz inicial A.

3. Implementação

A primeira parte do programa é adquirir o input da dimensão do grid e das temperaturas de fronteira, logo em seguida o grid é inicializado com as temperaturas de fronteira depois dos passos de inicialização é chamada uma função que monta a partir do grid a matriz de coeficientes do sistema linear dos pontos internos da placa.

Por conseguinte, é chamada uma função que cria um chute inicial e depois que todas as variáveis já estão preenchidas é chamado o método de Gauss-Seidel que retorna o vetor de soluções do sistema, logo em seguida o grid é atualizado com as temperaturas internas.

Imagem 1: Glossários de funções implementadas

```
# Retorna a matriz de coeficientes e o vetor B
def createLinearSystem(grid, n):

# Inicializa o grid com as temp. fronteiras
def initGrid(dR, n):

# Atualiza o grid com a solução
def updateGrid(grid, xf, n):

# Cria automaticamente um chute inicial
def initialX(A, b):

# Soluciona o sistema via Gauss-Seidel
def solve(A, b, xi, t):
```

4. Gráficos

Aqui nesta seção serão apresentados diferentes gráficos para diferentes valores de temperaturas e tamanhos de grids.

Imagem 2: Grid(4x4), Temperaturas(25,20,20,30)

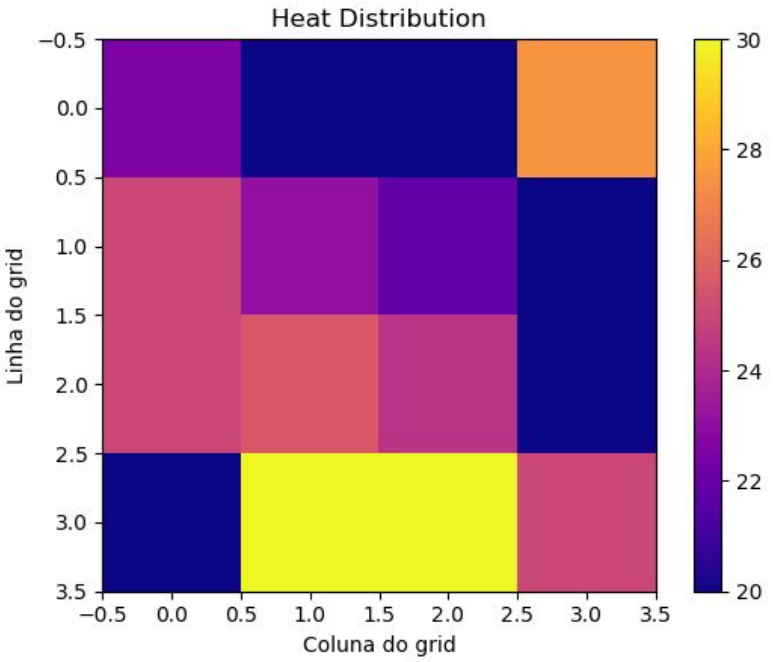

Imagem 3: Grid(9x9), Temperaturas(9,6,9,6)

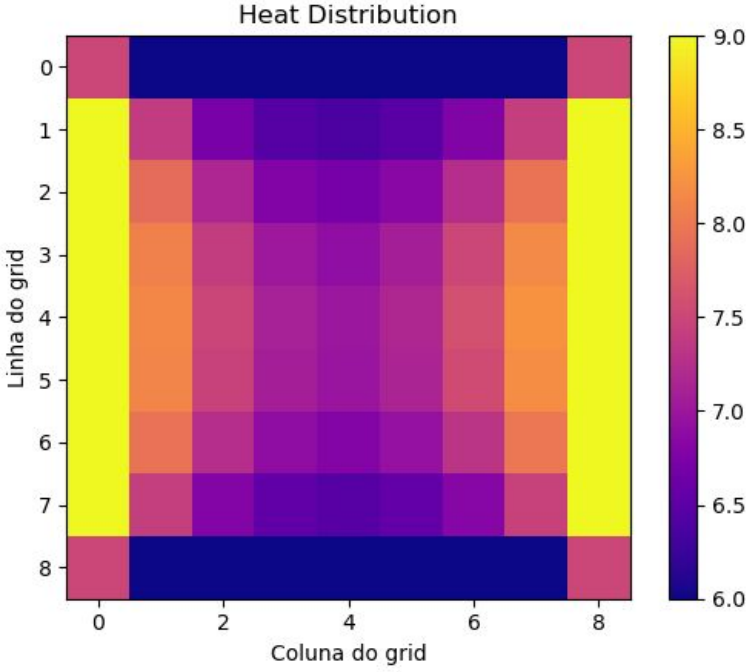

Imagem 4: Grid(40x40), Temperaturas(32.5,34,36,31.5)