

## Solução Numérica da Difusão de Calor 2D

Relatório do 2º Trabalho Computacional - Regime Permanente

Aluno: Romulo de Aguiar Beninca

### 1. Definição do Problema e Organização do Trabalho

O objetivo deste trabalho é desenvolver um código computacional para resolver a equação de difusão de calor bidimensional em regime permanente, utilizando o Método de Volumes Finitos (MVF). Conforme especificado na tarefa, o problema físico consiste em uma placa quadrada unitária onde a temperatura é prescrita na **face superior**, enquanto as demais faces são mantidas a  $0^{\circ}C$ .

Este relatório está organizado de maneira a apresentar a metodologia e os resultados de forma estruturada:

**Seção 2 (Desenvolvimento Matemático):** Detalha a discretização da equação governante, o tratamento das condições de contorno via volumes fictícios e a dedução da solução analítica por separação de variáveis.

**Seção 3 (Correspondência Algébrica):** Relaciona as equações matemáticas deduzidas com a lógica de programação implementada.

**Seção 4 (Resultados):** Apresenta a simulação interativa, gráficos de convergência, perfis de temperatura comparativos, mapas de calor e cálculos de integrais (temperatura média e fluxos de calor).

**Seção 5 (Listagem do Programa):** Disponibiliza o código fonte completo em JavaScript utilizado para a solução do problema.

#### ITENS SOLICITADOS NA ESPECIFICAÇÃO

- Item 1:** Número de iterações e gráfico da variação de  $T(1/2, 1/2)$  (escala log/decimal).
- Item 2:** Tabela com solução analítica, numérica e erro para o perfil vertical em  $X = 1/2$ .
- Item 3:** Gráfico de  $Y$  versus  $T$  para  $X = 1/2$  (Perfil Vertical).
- Item 4:** Tabela com solução analítica, numérica e erro para o perfil horizontal em  $Y = 1/2$ .
- Item 5:** Gráfico de  $T$  versus  $X$  para  $Y = 1/2$  (Perfil Horizontal).
- Item 6:** Temperatura média da placa (Analítica vs Numérica).
- Item 7:** Taxa de transferência de calor no contorno leste (Analítica vs Numérica).
- Item 8:** Taxa de transferência de calor no contorno norte (Analítica vs Numérica).
- Item 9:** Listagem impressa do programa computacional implementado.

# Dinâmica de fluidos computacionais I

Questão única: implementar um programa computacional para resolver com o método de volumes finitos o problema definido por:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0, \quad T(0, y) = T(1, y) = T(x, 0) = 0, \quad T(x, 1) = \sin(\pi x)$$

Dados:

$K = 1 \text{ W/m.K}$      $N_x = N_y = 13$  volumes de controle  
 solver: Gauss-Seidel    Estimativa inicial da temperatura:  $0^\circ$   
 Malha uniforme (dimensão unitária em  $z$ )  
 Condições de contorno aplicada com volumes fictícios  
 Funções de interpolação lineares (CDS) para  $T$

Interpretação do problema

O trabalho trata da difusão de calor 2D em regime permanente, abordada no capítulo 5 da unidade curricular CFD. Como ilustrado no material do capítulo 5, o problema consiste em calcular a distribuição de temperatura em uma placa quadrada, tendo como condição inicial a lado direito uma fonte de calor com distribuição  $T(x, y) = \sin(\pi x)$  e as demais bordas a  $0^\circ\text{C}$ .

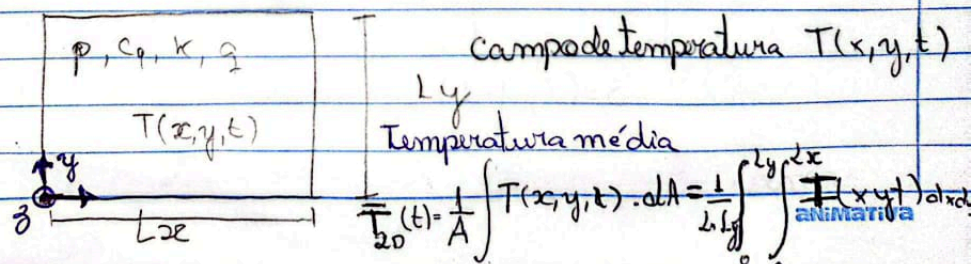


Figura 1: Definição dos parâmetros de malha ( $N = 13$ ) e equações iniciais.

## 2. Desenvolvimento Matemático (Dedução)

---

### 2.1. Interpretação Física e Malha

A discretização espacial é realizada dividindo o domínio em volumes de controle. O comportamento esperado é um fluxo de calor partindo da região de maior temperatura (fonte senoidal) em direção às regiões mais frias ( $0^{\circ}C$ ).



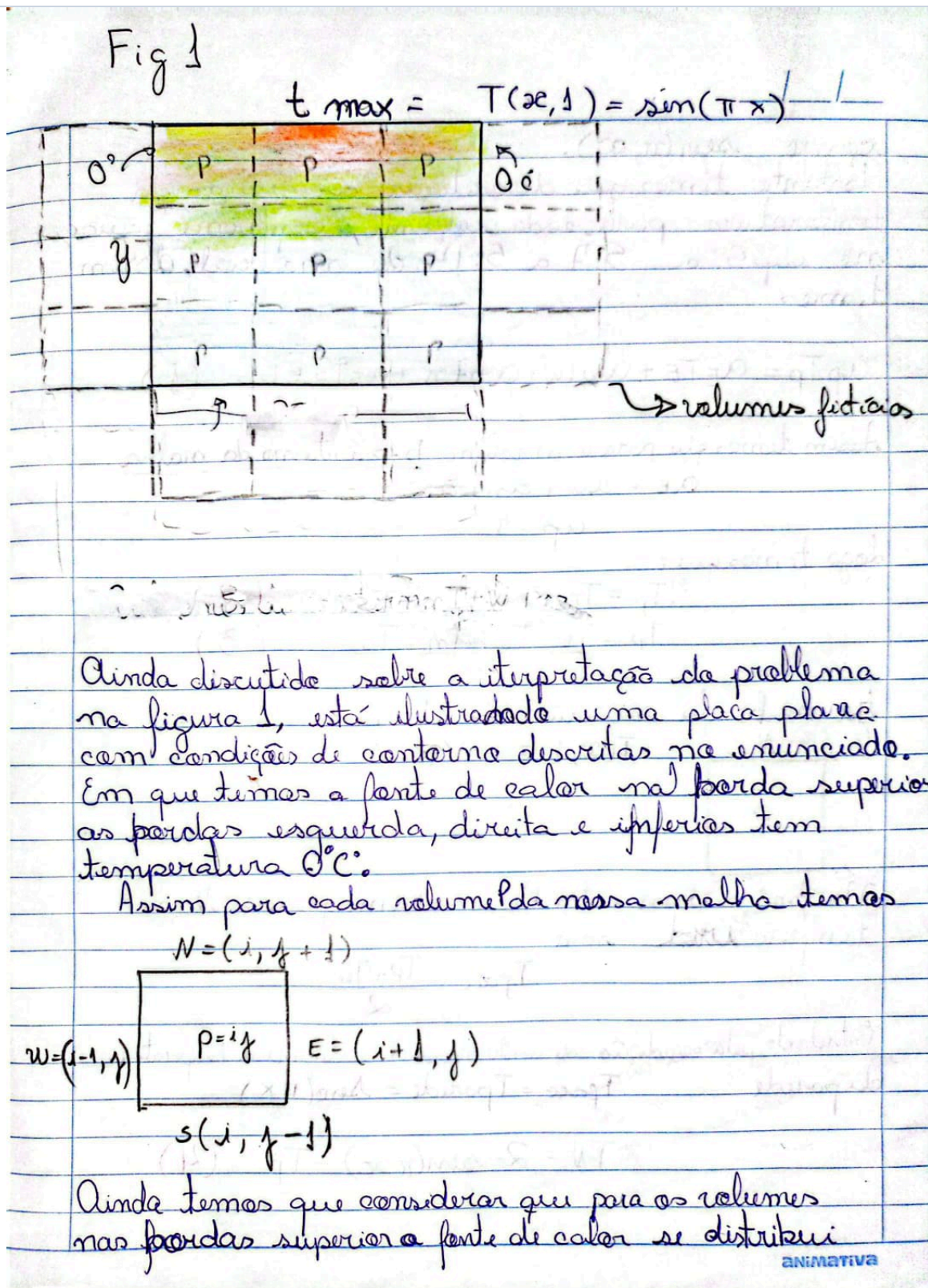


Figura 2: Interpretação física e esquema da malha com nomenclatura dos pontos cardeais (P, N, S, E, W).

## 2.2. Discretização para Volumes Internos

Para os volumes de controle internos, onde não há contato direto com as bordas prescritas, aplicamos o balanço de energia considerando condutividade constante e malha uniforme ( $\Delta x = \Delta y$ ). Isso resulta na simplificação onde a soma dos coeficientes vizinhos é igual ao coeficiente central.

$$\frac{1}{\pi} \sin(\pi x) = (1,95) T = x \text{ cm } \dagger$$

como  $\sin(\pi, x)$ .

Portanto temos que discretizar a equação da temperatura para cada volume, conforme equação no cap 5 a 5.7 a 5.14 do material, assim temos

$$A_p T_p = A_E T_E + A_W T_W + A_n T_n + A_s T_s + b_p \quad (1)$$

Assim temos que para um volume b-p interno da malha

$$A_E = A_W + A_n + A_s$$

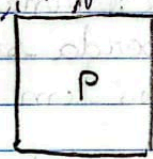
$$A_p = 4$$

Logo temos que:

$$4T_p = T_E + T_W + T_n + T_s \quad (2)$$

$$T_p = \frac{T_E + T_W + T_n + T_s}{4} \quad (3)$$

Na borda superior ~~temos~~ temos



$$T(x, 1) = \sin(\pi x)$$

Na face superior entre P e o volume fictício N a temperatura seria

$$T_{face} = \frac{T_p + T_n}{2}$$

Entretanto pela condição de contorno deve ser igual a temperatura da parede

$$T_{face} = T_{parde} = \sin(\pi x)$$

$$T_n = 2 \sin(\pi x) - T_p \quad (4)$$

Figura 3: Dedução da equação para nós internos, resultando na média aritmética dos 4 vizinhos.



### 2.3. Tratamento da Condição de Contorno

Para tratar a condição de contorno prescrita (fonte de calor), utiliza-se a técnica de volumes fictícios

Agora substituindo a equação 4 em 2 temos

$$4T_p = T_E + T_W + (2\sin(\pi x) - T_p) + T_S$$

$$4T_p = T_E + T_W + T_S + (2\sin(\pi x)) - T_p$$

$$5T_p = T_E + T_W + T_S + 2\sin(\pi x)$$

$$T_p = \frac{T_E + T_W + T_S + 2\sin(\pi x)}{5}$$

Para as demais bordas

$$T(0, y), T(x, 0), T(1, y) = 0$$

$$T_p = 0$$

Figura 4: Dedução do coeficiente modificado (divisor 5) utilizando volumes fictícios (Adaptação para o problema atual).

## 2.4. Resumo do Algoritmo (Gauss-Seidel)

Para a solução do sistema de equações algébricas, optou-se pelo método iterativo de Gauss-Seidel. O algoritmo varre a malha atualizando a temperatura  $T_P$  com base nos valores mais recentes dos vizinhos.

Agora usando Gauss Seidel para modelar o nosso modelo numérico de termos

Para os volumes internos

$$T_{i,j}^{(k+1)} = \frac{T_{i+1,j}^{(k)} + T_{i-1,j}^{(k+1)} + T_{i,j+1}^{(k)} + T_{i,j-1}^{(k+1)}}{4}$$

Para a borda direita

$$T_{1,j}^{(k+1)} = \frac{T_{2,j}^{(k+1)} + T_{1,j} + T_{1,j+1}^{(k+1)} + 2 \sin(\pi x f)}{5}$$

Para as demais bordas

$$T = 0$$

Para volume superior direito e volume inferior direito

$$T_p = \frac{T_W + T_S + 0 + 2 \sin(\pi x f)}{5}$$

$$T_p = \frac{T_W + T_N + 0 + 2 \sin(\pi x f)}{5}$$

Figura 5: Resumo das equações iterativas implementadas no código para varredura do domínio.



## 2.5. Solução Analítica

Para fins de validação dos resultados numéricos, deduziu-se a solução exata da equação nas condições de contorno apresentadas. O método utilizado foi a Separação de Variáveis, assumindo que  $T(x, y) = X(x)Y(y)$ .

O processo inicia-se dividindo a EDP em duas EDOs e aplicando as condições homogêneas nas bordas  $x = 0$ ,  $x = 1$  e  $y = 0$ .

Solução analítica

Considerando a equação

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \quad (1)$$

Com as condições de contorno

$$T(0, y) = 0$$

$$T(1, y) = 0$$

$$T(x, 0) = 0$$

$$T(x, 1) = \sin(\pi x) \quad (2)$$

Procuramos uma solução na forma

$$T(x, y) = X(x)Y''(y) = 0 \quad (3)$$

Dividindo por  $X(x)Y(y)$  ( $x \neq y$ )

$$\frac{X''(x)}{X(x)} + \frac{Y''(y)}{Y(y)} = 0 \quad (4)$$

$$-\lambda = \frac{X''(x)}{X(x)} \quad \lambda = \frac{Y''(y)}{Y(y)} \quad (5)$$

Figura 6: Início da dedução analítica: separação de variáveis e aplicação das condições de contorno homogêneas.

Em seguida, determina-se os autovalores  $\lambda_n$  e as autofunções associadas para  $X(x)$  e  $Y(y)$ , resultando em uma solução na forma de série infinita.

$$X''(x) + \lambda x(x) = 0 \quad (6)$$

Das condições de contorno em  $x$

$$T(0; y) = 0 \Rightarrow X(0)Y(y) = 0 \Rightarrow X(0) = 0 \quad (7)$$

$$\lambda_n = (n\pi)^2, \quad n = 1, 2, 3, \dots \quad (8)$$

$$x_n(x) = \sin(n\pi x) \quad (9)$$

$$\lambda_n = (n\pi)^2 \text{ e } y$$

$$y_n''(y) - (n\pi)^2 Y_n(y) = 0 \quad (10)$$

com solução geral

$$Y_n(y) = A_n \sin(n\pi y) + B_n \cosh(n\pi y) \quad (11)$$

Aplicando a condição geral temos,  $y=0$

$$T(x, 0) = 0 \Rightarrow X_n(x)Y_n(0) = 0 \quad (12)$$

$$Y_n(0) = A_n \sin(0) + B_n \cosh(0) = B_n \cdot 1$$

logo

$$Y_n(y) = A_n \sinh(n\pi y) \quad (13)$$

Como uma série

$$T(x, y) = \sum_{n=1}^{\infty} A_n \sinh(n\pi y) \sin(n\pi x) \quad (14)$$

ANIMATIVA

Figura 7: Determinação dos autovalores e da solução geral em série.



Por fim, aplica-se a condição de contorno não-homogênea na face superior ( $T(x, 1) = \sin(\pi x)$ ) para encontrar os coeficientes da série. Devido à natureza da função seno, a série colapsa em um único termo (modo fundamental  $n = 1$ ).

Entretanto na face superior temos  $T(x, 1) = \sin(\pi x)$

$$T(x, 1) = \sum_{n=1}^{\infty} A_n \sinh(n\pi) \sin(n\pi x) = \sin(\pi x) \quad (15)$$

$$\sin(\pi x) = \sum_{n=1}^{\infty} B_n \sin(n\pi x) \quad (16)$$

$$B_n = A_n \sinh(n\pi) \quad (17)$$

Logo  $A_1 \sinh(\pi) = 1 \Rightarrow A_1 = \frac{1}{\sinh(\pi)} \quad (18)$

$$A_n \sinh(n\pi) = 0 \Rightarrow A_n = 0 \quad (19)$$

Substituindo termos (18 em 14)

$$T(x, y) = A_1 \sinh(\pi y) \sin(\pi x) = \frac{\sinh(\pi y)}{\sinh(\pi)} \sin(\pi x)$$

$$T(x, y) = \frac{\sinh(\pi y)}{\sinh(\pi)} \cdot \sin(\pi x)$$

Solução analítica

animativa

CS Digitalizado com CamScanner

Figura 8: Aplicação da condição da face superior e obtenção da equação final fechada.

### 3. Correspondência Algébrica vs. Implementação

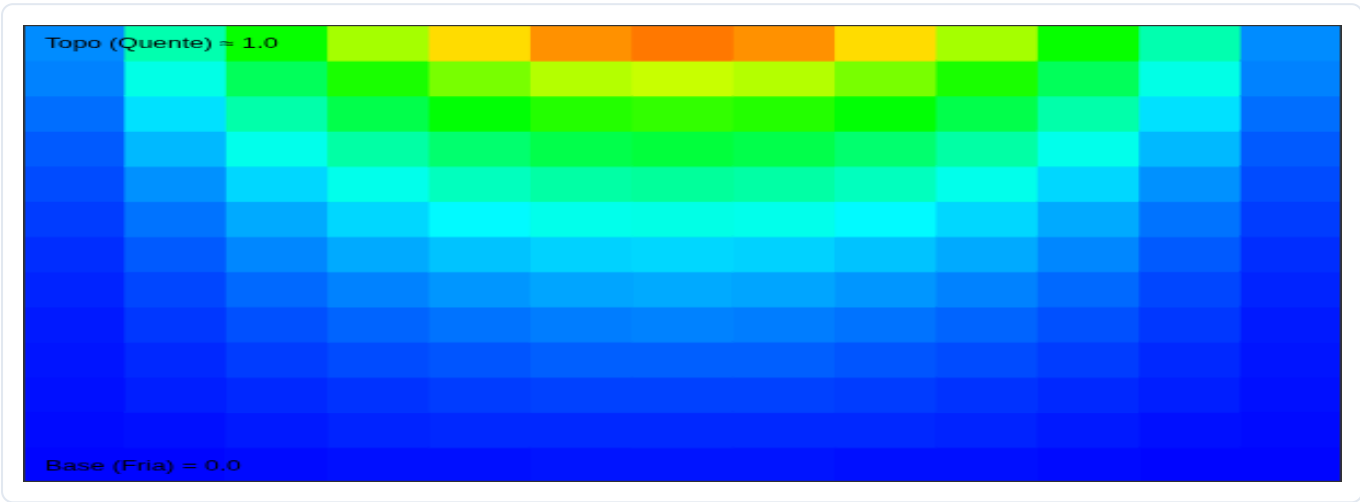
A implementação utiliza JavaScript puro. A tabela abaixo relaciona as equações deduzidas com a lógica de código utilizada no loop iterativo:

Tipo de Volume	Expressão Algébrica (Dedução)	Lógica no Código (JS)
<b>Interno</b> (Longe da borda superior)	$T_P^{(k+1)} = \frac{T_E^{(k)} + T_W^{(k+1)} + T_N^{(k)} + T_S^{(k+1)}}{4}$	<pre>T[i][j] = (T[i+1][j] + T[i-1][j] + T[i][j+1] + T[i][j-1]) / 4.0</pre>
<b>Borda Superior</b> ( $j = Ny - 1$ )	$T_P = \frac{T_E + T_W + T_S + 2\text{sen}(\pi x)}{5}$	<pre>fonte = 2.0 * Math.sin(Math.PI * x) T[i][j] = (T[i+1][j] + T[i-1][j] + T[i][j-1] + fonte) / 5.0</pre>

#### 4. Resultados da Simulação

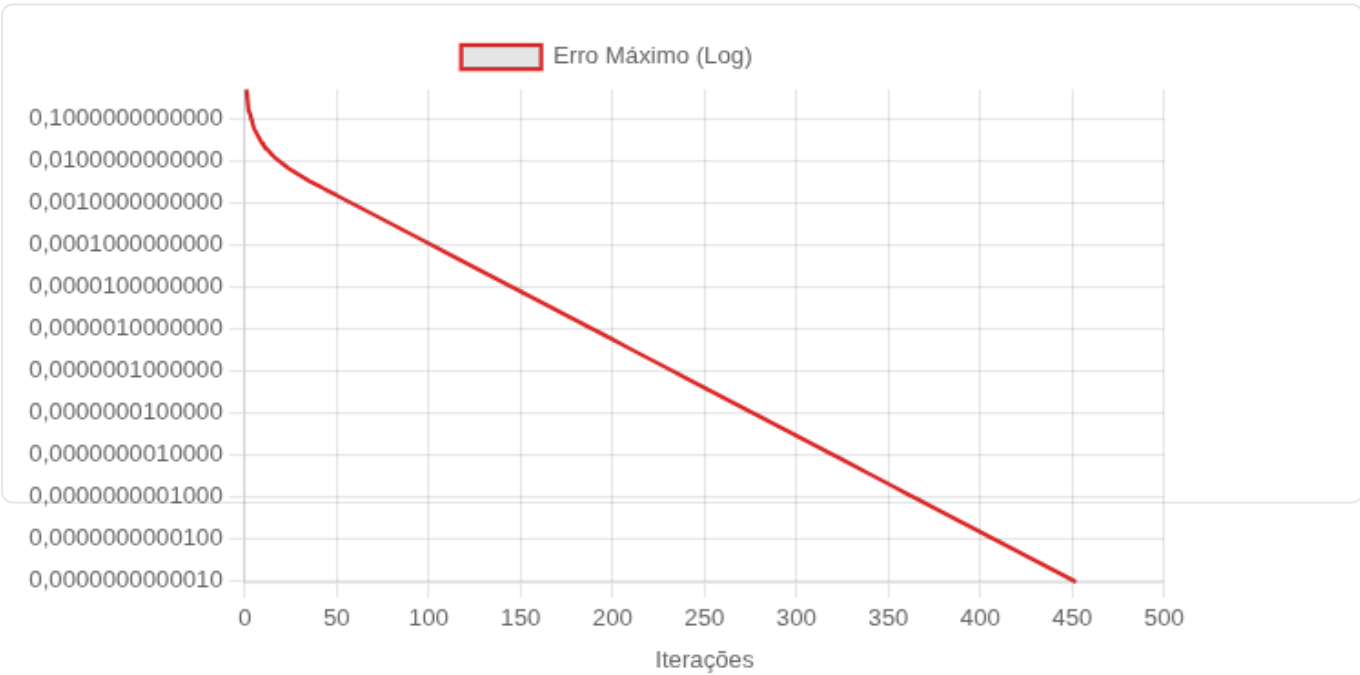
##### 4.1. Campo de Temperatura (Visualização)

Distribuição de temperatura na placa. O gradiente parte da face superior (quente) para as demais faces (frias).

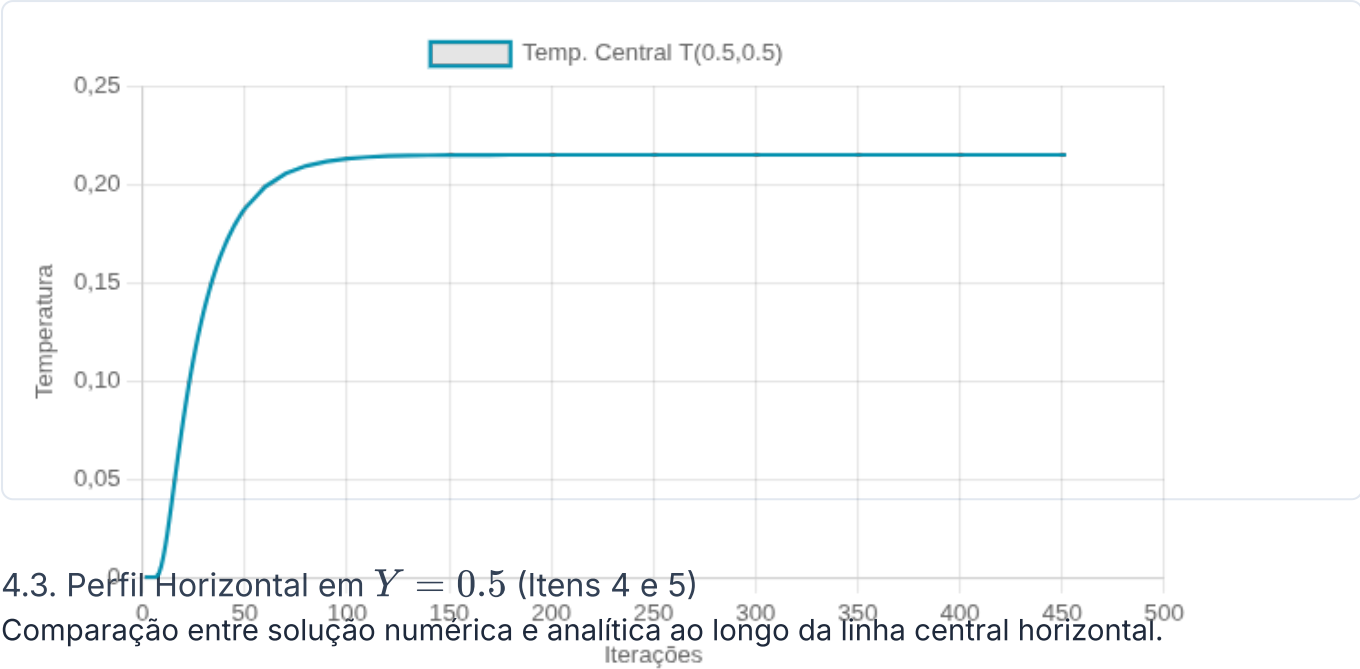


##### 4.2. Convergência e Monitoramento (Item 1 do trabalho)

Variação da temperatura no ponto central  $T(0.5, 0.5)$  e do erro máximo a cada iteração.







4.3. Perfil Horizontal em  $Y = 0.5$  (Itens 4 e 5)  
 Comparação entre solução numérica e analítica ao longo da linha central horizontal.

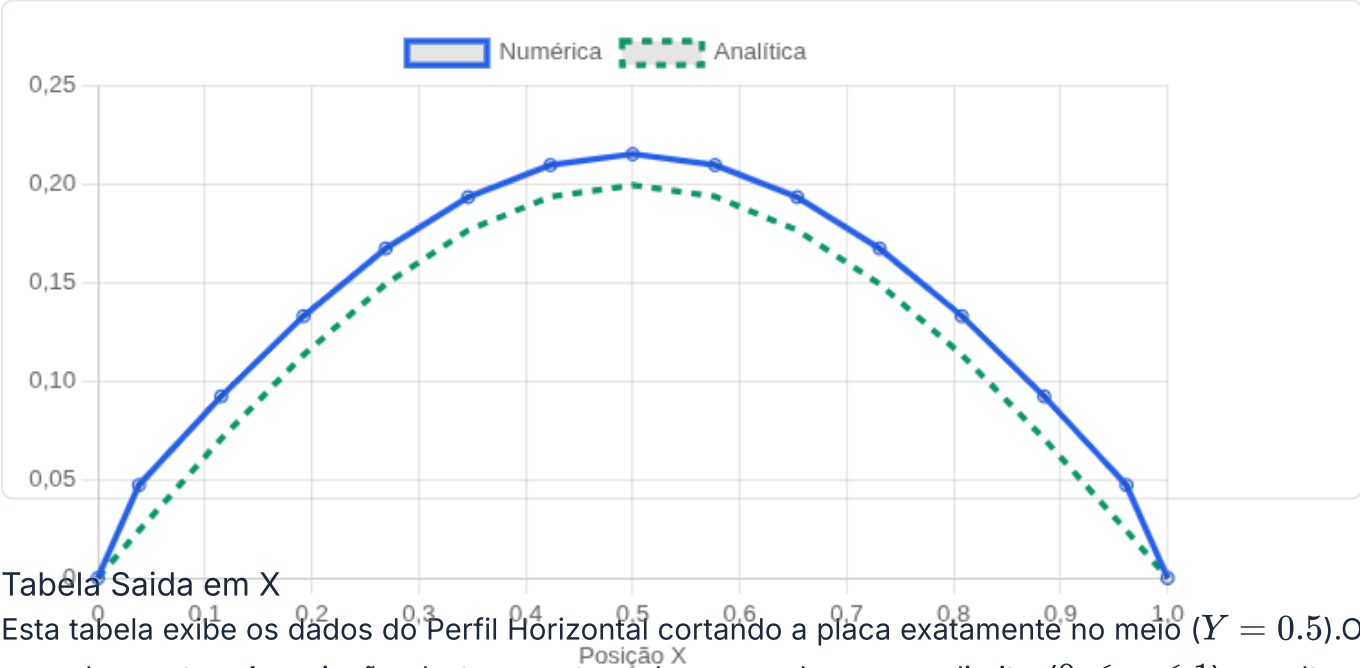


Tabela Saída em X  
 Esta tabela exibe os dados do Perfil Horizontal cortando a placa exatamente no meio ( $Y = 0.5$ ).O que ela mostra: A variação da temperatura da esquerda para a direita ( $0 \leq x \leq 1$ ) na altura média da placa.

i	x	Numérica	Analítica	Erro
BC	0.0000	0.000000	0.000000	0.00e+0
2	0.1154	0.092160	0.070662	2.15e-2
4	0.2692	0.167195	0.149155	1.80e-2

i	x	Numérica	Analítica	Erro
6	0.4231	0.209483	0.193478	1.60e-2
8	0.5769	0.209483	0.193478	1.60e-2
10	0.7308	0.167195	0.149155	1.80e-2
12	0.8846	0.092160	0.070662	2.15e-2
BC	1.0000	0.000000	0.000000	0.00e+0

#### 4.4. Perfil Vertical em $X = 0.5$ (Itens 2 e 3)

Comparação entre solução numérica e analítica ao longo da linha central vertical.

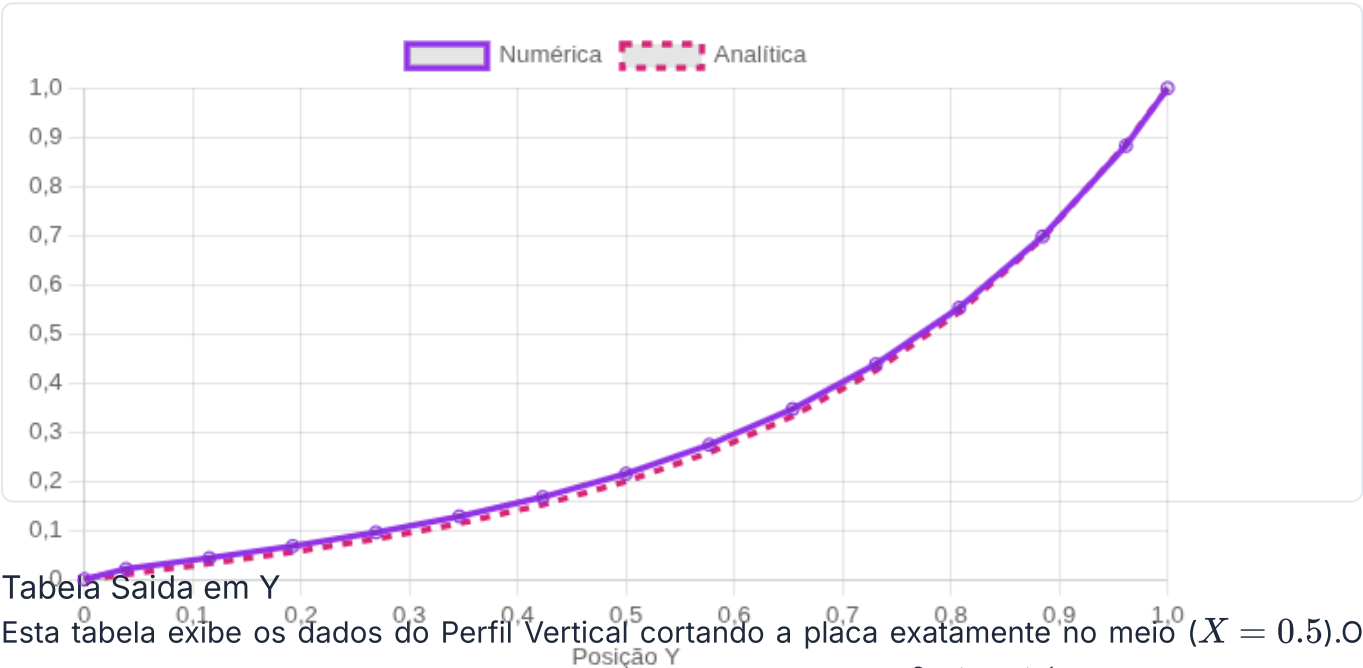


Tabela Saída em Y

Esta tabela exibe os dados do Perfil Vertical cortando a placa exatamente no meio ( $X = 0.5$ ).O que ela mostra: A variação da temperatura de baixo para cima ( $0 \leq y \leq 1$ ) na largura média da placa.

j	y	Numérica	Analítica	Erro
BC	0.0000	0.000000	0.000000	0.00e+0
2	0.1154	0.043344	0.032080	1.13e-2

j	y	Numérica	Analítica	Erro
4	0.2692	0.095540	0.082289	1.33e-2
6	0.4231	0.167326	0.152098	1.52e-2
8	0.5769	0.273751	0.258133	1.56e-2
10	0.7308	0.437972	0.425652	1.23e-2
12	0.8846	0.697919	0.694554	3.36e-3
BC	1.0000	1.000000	1.000000	0.00e+0

#### 4.5. Médias e Taxas de Transferência (Itens 6, 7 e 8)

##### ITEM 6: TEMPERATURA MÉDIA DA PLACA

Numérica: **0.201298** °C | Analítica: **0.185854** °C | Erro: **1.54e-2**

##### ITEM 7: TAXA DE CALOR NO CONTORNO LESTE (RESFRIADO)

Numérica: **1.542568** W/m | Analítica: **0.917152** W/m | Erro: **6.25e-1**

##### ITEM 8: TAXA DE CALOR NO CONTORNO NORTE (FONTE)

Numérica: **-1.730053** W/m | Analítica: **-2.007484** W/m | Erro: **2.77e-1**



## 5. Listagem do Programa (Item 9)

Código JavaScript completo utilizado para a simulação, traduzido e comentado.

```
(function() {  
    // Variáveis globais para os gráficos (para poder destruir/recriar)  
    let graficoConv = null, graficoCentro = null, graficoProfX = null, graficoProfY = null;  
  
    // --- SOLUÇÃO ANALÍTICA (FONTE NO TOPO) ---  
    //  $T(x,y) = \sin(\pi x) * \sinh(\pi y) / \sinh(\pi)$   
    function solucaoAnalitica(x, y) {  
        return (Math.sin(Math.PI * x) * Math.sinh(Math.PI * y)) / Math.sinh(Math.PI);  
    }  
  
    // Média Analítica: Integral dupla de  $T(x,y)$  dx dy  
    // Integral  $\sin(\pi x)$  de 0 a 1 =  $2/\pi$   
    // Integral  $\sinh(\pi y)$  de 0 a 1 =  $(\cosh(\pi)-1)/\pi$   
    // Constante  $1/\sinh(\pi)$   
    // Resultado:  $(2 * (\cosh(\pi) - 1)) / (\pi^2 * \sinh(\pi))$   
    function mediaAnalitica() {  
        return (2.0 * (Math.cosh(Math.PI) - 1.0)) / (Math.pow(Math.PI, 2) * Math.sinh(Math.PI));  
    }  
  
    // Fluxo Leste (x=1): Integral de  $-k * dT/dx$  dy de 0 a 1  
    //  $dT/dx = \pi * \cos(\pi x) * \sinh(\pi y) / \sinh(\pi)$   
    // Em x=1:  $\cos(\pi) = -1$ . Logo  $dT/dx = -\pi * \sinh(\pi y)/\sinh(\pi)$   
    // Fluxo  $q'' = -k * (-\pi * \dots) = k * \pi * \sinh(\pi y)/\sinh(\pi)$   
    // Integral dy de  $\sinh(\pi y)$  é  $(\cosh(\pi)-1)/\pi$   
    // Integral Total =  $k * (\cosh(\pi)-1)/\sinh(\pi) = k * \tanh(\pi/2)$   
    function fluxoLesteAnalitico(k=1.0) {  
        return k * Math.tanh(Math.PI / 2.0);  
    }  
  
    // Fluxo Norte (y=1): Integral de  $-k * dT/dy$  dx de 0 a 1  
    //  $dT/dy = \pi * \sin(\pi x) * \cosh(\pi y) / \sinh(\pi)$   
    // Em y=1:  $\cosh(\pi)$ . Logo  $dT/dy = \pi * \sin(\pi x) * \coth(\pi)$   
    // Fluxo  $q'' = -k * \pi * \sin(\pi x) * \coth(\pi)$   
    // Integral dx de  $\sin(\pi x)$  é  $2/\pi$   
    // Integral Total =  $-k * 2 * \coth(\pi)$   
    function fluxoNorteAnalitico(k=1.0) {  
        return -k * 2.0 * (1.0 / Math.tanh(Math.PI));  
    }  
  
    // Função Principal chamada pelo botão na UI  
    window.executarSimulacaoPeloUI = function() {  
        // Obter valores dos inputs do HTML  
        const Nx = parseInt(document.getElementById('entrada_Nx').value);  
        const Ny = parseInt(document.getElementById('entrada_Ny').value);  
        const maxIter = parseInt(document.getElementById('entrada_MaxIter').value);  
        const tol = parseFloat(document.getElementById('entrada_Tol').value);  
  
        // 1. Executar o Solver Numérico  
        const resultado = executarSolver(Nx, Ny, maxIter, tol);  
  
        // 2. Gerar Gráficos de Convergência (Item 1 do PDF)  
        plotarConvergencia(resultado.histErro);  
        plotarHistoricoCentro(resultado.histCentro);  
    }  
})
```

```

// 3. Gerar Perfis e Tabelas (Itens 2, 3, 4, 5 do PDF)
gerarPerfilHorizontal(resultado.T, Nx, Ny);
gerarPerfilVertical(resultado.T, Nx, Ny);

// 4. Desenhar Mapa de Calor (Heatmap)
desenharMapaCalor(resultado.T, Nx, Ny);

// 5. Calcular Integrais e Fluxos (Itens 6, 7, 8 do PDF)
calcularIntegrais(resultado.T, Nx, Ny);

// 6. Exibir o próprio código fonte na página (Item 9 do PDF)
document.getElementById('exibicao-codigo-fonte').textContent = document.getElementById('codigo-fonte').textContent;
}

// Solver Gauss-Seidel
function executarSolver(Nx, Ny, maxIter, tol) {
    // Inicializar matriz T com zeros
    let T = Array(Nx).fill(0).map(() => Array(Ny).fill(0.0));
    let erroMax = 1.0, iteracao = 0;
    let histErro = [], histCentro = [];

    let meioX = Math.floor(Nx/2);
    let meioY = Math.floor(Ny/2);

    // Loop Iterativo
    while(iteracao < maxIter && erroMax > tol) {
        erroMax = 0.0;
        // Varredura da malha
        for(let i=0; i<Nx; i++) {
            for(let j=0; j<Ny; j++) {
                let T_antigo = T[i][j];

                // Vizinhos (Se fora do domínio, T=0, exceto topo que é tratado na eq)
                let Tw = (i > 0) ? T[i-1][j] : 0.0; // Oeste
                let Te = (i < Nx-1) ? T[i+1][j] : 0.0; // Leste
                let Ts = (j > 0) ? T[i][j-1] : 0.0; // Sul
                let Tn = (j < Ny-1) ? T[i][j+1] : 0.0; // Norte

                let T_novo = 0;

                // Condição de Contorno no Topo (j = Ny-1)
                // Fonte:  $T(x,1) = \sin(\pi \cdot x)$ 
                if(j === Ny-1) {
                    let x = (i + 0.5) / Nx;
                    let fonte = 2.0 * Math.sin(Math.PI * x);
                    // Eq. deduzida:  $(Tw + Te + Ts + fonte) / 5$ 
                    // Nota: Tn é o vizinho fictício incorporado na fonte
                    T_novo = (Tw + Te + Ts + fonte) / 5.0;
                } else {
                    // Volumes Internos
                    // Eq. deduzida:  $(Tw + Te + Tn + Ts) / 4$ 
                    T_novo = (Tw + Te + Tn + Ts) / 4.0;
                }

                T[i][j] = T_novo; // Atualização Gauss-Seidel (imediata)

                let erroLocal = Math.abs(T_novo - T_antigo);
                if(erroLocal > erroMax) erroMax = erroLocal;
            }
        }
        iteracao++;
    }
}

```

```

    }
    iteracao++;

    // Salvar dados para os gráficos
    // Salva apenas a cada 10 iterações ou no final para performance
    if(iteracao < 50 || iteracao % 10 === 0 || erroMax ≤ tol) {
        histErro.push({x: iteracao, y: erroMax});
        histCentro.push({x: iteracao, y: T[meioX][meioY]});
    }
}
return {T, histErro, histCentro, iteracao};
}

// Cálculo das integrais (Média e Fluxos)
function calcularIntegrais(T, Nx, Ny) {
    let dx = 1.0/Nx, dy = 1.0/Ny;
    let k = 1.0; // Condutividade térmica

    // --- Item 6: Temperatura Média (Regra do Retângulo) ---
    let somaT = 0;
    for(let i=0; i<Nx; i++) {
        for(let j=0; j<Ny; j++) {
            somaT += T[i][j];
        }
    }
    // Área total = 1.0 * 1.0 = 1.0
    let mediaNum = (somaT * dx * dy) / 1.0;
    let mediaAna = mediaAnalitica();
    let erroMedia = Math.abs(mediaNum - mediaAna);

    document.getElementById('res-media').innerHTML =
        `Numérica: <b>${mediaNum.toFixed(6)}</b> °C | Analítica: <b>${mediaAna.toFixed(6)}</b> °C`;

    // --- Item 7: Fluxo Leste (x=1) ---
    // Fronteira fria (T=0).
    // q'' = -k dT/dx. Usando UDS (Two-point backward): dT/dx ~ (T_parede - T_P) / (dx/2)
    // T_parede = 0. T_P = T[Nx-1][j]. Distância = dx/2.
    let fluxoLeste = 0;
    for(let j=0; j<Ny; j++) {
        let T_parede = 0.0;
        let T_P = T[Nx-1][j];
        let gradiente = (T_parede - T_P) / (dx/2.0);
        fluxoLeste += (-k * gradiente) * dy; // Integral de q'' dy
    }
    let flAna = fluxoLesteAnalitico();
    let errFL = Math.abs(fluxoLeste - flAna);

    document.getElementById('res-fluxo-leste').innerHTML =
        `Numérica: <b>${fluxoLeste.toFixed(6)}</b> W/m | Analítica: <b>${flAna.toFixed(6)}</b> W/m`;

    // --- Item 8: Fluxo Norte (y=1) ---
    // Fronteira quente (T=sin(pi*x)).
    // q'' = -k dT/dy. Usando UDS: dT/dy ~ (T_parede - T_P) / (dy/2)
    // T_parede = sin(pi*x). T_P = T[i][Ny-1].
    let fluxoNorte = 0;
    for(let i=0; i<Nx; i++) {
        let x = (i + 0.5) * dx;
        let T_parede = Math.sin(Math.PI * x);
        let T_P = T[i][Ny-1];
        let gradiente = (T_parede - T_P) / (dy/2.0);
    }
}

```

```

        fluxoNorte += (-k * gradiente) * dx; // Integral de q' dx
    }
    let fnAna = fluxoNorteAnalitico();
    let errFN = Math.abs(fluxoNorte - fnAna);

    document.getElementById('res-fluxo-norte').innerHTML =
        `Numérica: <b>${fluxoNorte.toFixed(6)}</b> W/m | Analítica: <b>${fnAna.toFixed(6)}</b> W/m`
}

// Gerar Perfil Horizontal (T vs X)
function gerarPerfilHorizontal(T, Nx, Ny) {
    let meioY = Math.floor(Ny/2);
    let coordY = (meioY + 0.5) / Ny;
    let htmlTabela = "<table style='width:100%'><thead><tr><th>i</th><th>x</th><th>Numérica</th><th>Analítica</th></tr></thead><tbody>";
    let dadosNum = [], dadosAna = [];

    // Adicionar contorno esquerdo (x=0, T=0)
    htmlTabela += `<tr><td>BC</td><td>0.0000</td><td>0.000000</td><td>0.000000</td><td>0.000000</td></tr>`;
    dadosNum.push({x:0, y:0}); dadosAna.push({x:0, y:0});

    for(let i=0; i<Nx; i++) {
        let x = (i + 0.5) / Nx;
        let valNum = T[i][meioY];
        let valAna = solucaoAnalitica(x, coordY);
        dadosNum.push({x:x, y:valNum}); dadosAna.push({x:x, y:valAna});
        htmlTabela += `<tr><td>${i+1}</td><td>${x.toFixed(4)}</td><td>${valNum.toFixed(6)}</td><td>${valAna.toFixed(6)}</td><td>${errFN.toFixed(6)}</td></tr>`;
    }

    // Adicionar contorno direito (x=1, T=0)
    htmlTabela += `<tr><td>BC</td><td>1.0000</td><td>0.000000</td><td>0.000000</td><td>0.000000</td></tr>`;
    dadosNum.push({x:1, y:0}); dadosAna.push({x:1, y:0});

    document.getElementById('saida-tabela-x').innerHTML = htmlTabela;

    const ctx = document.getElementById('graficoPerfilX').getContext('2d');
    if(graficoProfX) graficoProfX.destroy();
    graficoProfX = new Chart(ctx, {
        type: 'line',
        data: { datasets: [
            { label:'Numérica', data:dadosNum, borderColor:'#2563eb', pointRadius:3 },
            { label:'Analítica', data:dadosAna, borderColor:'#059669', borderDash:[5,5], pointRadius:3 }
        ]},
        options: { responsive:true, maintainAspectRatio:false, scales:{x:{type:'linear', ticks:10}, y:{type:'linear', ticks:10}}
    });
}

// Gerar Perfil Vertical (T vs Y)
function gerarPerfilVertical(T, Nx, Ny) {
    let meioX = Math.floor(Nx/2);
    let coordX = (meioX + 0.5) / Nx;
    let htmlTabela = "<table style='width:100%'><thead><tr><th>j</th><th>y</th><th>Numérica</th><th>Analítica</th></tr></thead><tbody>";
    let dadosNum = [], dadosAna = [];

    // Contorno Inferior (y=0, T=0)
    dadosNum.push({x:0, y:0}); dadosAna.push({x:0, y:0});
    htmlTabela += `<tr><td>BC</td><td>0.0000</td><td>0.000000</td><td>0.000000</td><td>0.000000</td></tr>`;

    for(let j=0; j<Ny; j++) {
        let y = (j + 0.5) / Ny;
        let valNum = T[meioX][j];
        let valAna = solucaoAnalitica(coordX, y);
        dadosNum.push({x:j, y:y}); dadosAna.push({x:j, y:y});
        htmlTabela += `<tr><td>${j+1}</td><td>${y.toFixed(4)}</td><td>${valNum.toFixed(6)}</td><td>${valAna.toFixed(6)}</td><td>${errFN.toFixed(6)}</td></tr>`;
    }

    document.getElementById('saida-tabela-y').innerHTML = htmlTabela;

    const ctx = document.getElementById('graficoPerfilY').getContext('2d');
    if(graficoProfY) graficoProfY.destroy();
    graficoProfY = new Chart(ctx, {
        type: 'line',
        data: { datasets: [
            { label:'Numérica', data:dadosNum, borderColor:'#2563eb', pointRadius:3 },
            { label:'Analítica', data:dadosAna, borderColor:'#059669', borderDash:[5,5], pointRadius:3 }
        ]},
        options: { responsive:true, maintainAspectRatio:false, scales:{x:{type:'linear', ticks:10}, y:{type:'linear', ticks:10}}
    });
}

```



```

        let valAna = solucaoAnalitica(coordX, y);
        dadosNum.push({x:y, y:valNum}); dadosAna.push({x:y, y:valAna});
        htmlTabela += `<tr><td>${j+1}</td><td>${y.toFixed(4)}</td><td>${valNum.toFixed(6)}
    }

    // Contorno Superior (y=1, T=sin(pi*x))
    let tParede = Math.sin(Math.PI * coordX);
    dadosNum.push({x:1, y:tParede}); dadosAna.push({x:1, y:tParede});
    htmlTabela += `<tr><td>BC</td><td>1.0000</td><td>${tParede.toFixed(6)}</td><td>${tPar

    document.getElementById('saida-tabela-y').innerHTML = htmlTabela;

    const ctx = document.getElementById('graficoPerfilY').getContext('2d');
    if(graficoProfY) graficoProfY.destroy();
    graficoProfY = new Chart(ctx, {
        type: 'line',
        data: { datasets: [
            { label: 'Numérica', data: dadosNum, borderColor: '#9333ea', pointRadius: 3 },
            { label: 'Analítica', data: dadosAna, borderColor: '#db2777', borderDash: [5, 5],
        ]},
        options: { responsive: true, maintainAspectRatio: false, scales: { x: { type: 'linear',
    }
});

// Gráfico de Convergência do Erro
function plotarConvergencia(dados) {
    const ctx = document.getElementById('graficoConvergencia').getContext('2d');
    if(graficoConv) graficoConv.destroy();
    graficoConv = new Chart(ctx, {
        type: 'line',
        data: { datasets: [{ label: 'Erro Máximo (Log)', data: dados, borderColor: '#dc2626',
        options: { responsive: true, maintainAspectRatio: false, scales: { x: { type: 'linear',
    }
});

// Gráfico do Histórico do Ponto Central
function plotarHistoricoCentro(dados) {
    const ctx = document.getElementById('graficoPontoCentral').getContext('2d');
    if(graficoCentro) graficoCentro.destroy();
    graficoCentro = new Chart(ctx, {
        type: 'line',
        data: { datasets: [{ label: 'Temp. Central T(0.5,0.5)', data: dados, borderColor: '#dc2626',
        options: { responsive: true, maintainAspectRatio: false, scales: { x: { type: 'linear',
    }
});

// Mapa de Calor (Heatmap)
function desenharMapaCalor(T, Nx, Ny) {
    const c = document.getElementById('canvasMapaCalor');
    const ctx = c.getContext('2d');
    const w = c.parentElement.clientWidth;
    const h = c.parentElement.clientHeight;
    c.width = w; c.height = h;
    const cw = w/Nx, ch = h/Ny;

    for(let i=0; i<Nx; i++) {
        for(let j=0; j<Ny; j++) {
            let val = T[i][j];
            // Escala de cor: Azul (frio) → Vermelho (quente)
            // 0 máximo analítico é 1.0 (seno de pi/2 * sinh/sinh)

```

```
        let matiz = (1.0 - val) * 240;
        ctx.fillStyle = `hsl(${matiz}, 100%, 50%)`;
        // Inverter Y para desenhar de baixo para cima
        ctx.fillRect(i*cw, h - (j+1)*ch, cw+1, ch+1);
    }
}

// Borda
ctx.strokeStyle = "#333"; ctx.lineWidth=2; ctx.strokeRect(0,0,w,h);

// Legenda Simples
ctx.fillStyle = "black"; ctx.font = "12px sans-serif";
ctx.fillText("Topo (Quente)  $\approx$  1.0", 10, 20);
ctx.fillText("Base (Fria) = 0.0", 10, h - 10);
}

// Executar automaticamente ao carregar a página
setTimeout(window.executarSimulacaoPeloUI, 500);
})();
```