UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

LEANDRO WRZECIONEK DE BRITO

EQUAÇÃO DO CALOR BIDIMENSIONAL COM CONDIÇÕES DE CONTORNO DE NEUMANN: SOLUÇÕES ANALÍTICA E NUMÉRICA

TOLEDO

2024

LEANDRO WRZECIONEK DE BRITO

EQUAÇÃO DO CALOR BIDIMENSIONAL COM CONDIÇÕES DE CONTORNO DE NEUMANN: SOLUÇÕES ANALÍTICA E NUMÉRICA

Two-dimensional heat equation with Neumann boundary conditions: analytical and numerical solutions

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Licenciado em Matemática do Curso de Licenciatura em Matemática da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Vanderlei Galina

Coorientadora: Profª. Drª. Jocelaine Cargnelutti

TOLEDO

2024

LEANDRO WRZECIONEK DE BRITO

EQUAÇÃO DO CALOR BIDIMENSIONAL COM CONDIÇÕES DE CONTORNO DE NEUMANN: SOLUÇÕES ANALÍTICA E NUMÉRICA

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Licenciado em Matemática do Curso de Licenciatura em Matemática da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 13 de junho de 2024

Vanderlei Galina
Doutorado
Universidade Tecnológica Federal do Paraná

Jocelaine Cargnelutti
Doutorado
Universidade Tecnológica Federal do Paraná

Marcia Regina Piovesan
Doutorado
Universidade Tecnológica Federal do Paraná

Gustavo Henrique Dalposso
Doutorado
Universidade Tecnológica Federal do Paraná

TOLEDO

2024

Dedico este trabalho a todos que me
ensinaram Matemática.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por iluminar meu caminho, dar-me forças e ser minha fonte constante de esperança em todos os momentos, contra todas as incertezas - que não foram poucas - ao longo desta jornada.

Reconheço também os meus orientadores, Prof. Dr. Vanderlei Galina e Profª. Drª. Jocelaine Cargnelutti pela sabedoria, ideias, conselhos e conversas ao longo desta trajetória.

Gratulo as colegas de sala, Luana e Isadora, que também fizeram a iniciação científica em EDP comigo e, portanto, me acompanharam no estudo do assunto deste trabalho.

Ainda, sou grato a todos os professores de Matemática que tive ao longo da minha vida. Ao me deparar com outras realidades, percebi que esses profissionais me proporcionaram conhecimentos que não se encontram em todo lugar, o que com certeza corroborou neste trabalho.

Gostaria de deixar registrado também o meu reconhecimento à minha família, que sempre me aponhou em seguir o estudo da Matemática. Sem meus pais me ensinando operações elementares antes mesmo da escola, talvez meu caminho teria sido diferente e sou muito grato por essa influência.

Outra influência familiar, quando criança, significativa neste trabalho foram os ensinamentos em programação do meu primo, Diego. Sem ele, não teria contato com progamação antes da faculdade, o que muito provavelmente dificultaria meu aprendizado em Python.

Já mais recentemente, uma influência na programação que gracio foi do meu irmão, Cesar, que iniciou seus estudos no curso de Sistemas para Internet e contribuiu nas soluções de erros em meus códigos computacionais - que não foram poucos.

E, com certeza, agradeço a minha namorada, Ana, por todo o suporte enquanto escrevia este trabalho, pela compreensão nos momentos de ausência, por ouvir minhas tentativas falhas de explicar o que estava estudando, o que ajudou a melhorar minha explicação.

Enfim, agradeço a todos os que por algum motivo contribuíram para a realização desta pesquisa e que eventualmente eu tenha esquecido de citá-los.

"'Even in science,' Contemplation said, 'faith plays a role. Each experiment done, each step on the path of knowledge, is achieved by striking out into the darkness. You can't know what you will find, or that you will find anything at all. It is faith that drives us - faith in answers that must exist.'"
(Brandon Sanderson)

RESUMO

Neste trabalho, abordou-se a equação do calor bidimensional com condições de contorno de Neumann, a qual descreve a difusão do calor ao longo de um sólido. Esta equação tem grande importância em projetos que envolvem sistemas térmicos. Fez-se a dedução algébrica da equação, e então, foram obtidas as soluções analítica e numérica. A solução analítica foi obtida pelo método da separação de variáveis. Por outro lado, para obter a solução numérica, aplicou-se o Método das Diferenças Finitas, que foi utilizado inicialmente por Leonhard Euler (1707 – 1783) e é largamente utilizado em simulações numéricas. Os resultados alcançados, tanto numérico quanto analítico, foram implementados por meio da linguagem de programação Python. Por fim, faz-se a análise da solução por meio de gráficos 2D e avaliação do erro relativo. Conclui-se que a solução numérica é uma boa aproximação para a solução analítica.

Palavras-chave: equações diferenciais parciais; método analítico da separação de varáiveis; métodos numéricos; aproximação numérica.

ABSTRACT

In this article, we addressed the two-dimensional heat equation with Neumann boundary conditions, which describes the diffusion of heat through a solid. This equation holds significant importance in projects involving thermal systems. We performed the algebraic derivation of the equation, and subsequently, obtained both analytical and numerical solutions. The analytical solution was obtained using the method of separation of variables. On the other hand, to obtain the numerical solution, we applied the Finite Difference Method, which was initially utilized by Leonhard Euler (1707 – 1783) and is widely used in numerical simulations. The results achieved, both numerical and analytical, were implemented using the Python programming language. Finally, we conducted an analysis of the solution through 2D graphs and evaluated the relative error. We conclude that the numerical solution provides a good approximation to the analytical solution.

Keywords: two-dimensional heat equation; partial differential equations; Neumann boundary conditions; numerical methods.

LISTA DE FIGURAS

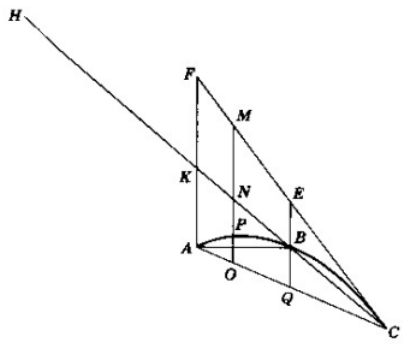
SUMÁRIO

1 INTRODUÇÃO

O cálculo diferencial e integral é composto por definições, teoremas e propriedades que fundamentam diversas aplicações, principalmente na área das ciências exatas e está atrelado a diversas descobertas, como o telefone, computadores, rádio, ultrassom, GPS, dividido o átomo, desvendado o genôma humano ou sequer ido à Lua (STROGATZ, 2019). Mesmo que seja uma pequena parte da elaboração dessas contribuições, são notórios os avanços proporcionados pelo cálculo.

A teoria do cálculo pode ser observada em pergaminhos antigos como, por exemplo, no Palimpsesto de Arquimedes, descoberto em uma biblioteca de Constantinopla em 1899. Nele, há um trecho em que Arquimedes, no século III a.C., explica o Método das Alavancas (Figura 1) para o cálculo de uma área limitada superiormente por uma parábola e inferiormente por uma reta (STROGATZ, 2019).

Figura 1 – Método das Alavancas



Fonte: Adaptado de Strogatz (2019).

Na figura 1, a parábola está sendo representada pelo arco AC e a reta, ao intersectar a parábola, forma o segmento AC. O triângulo AFC é formado de forma que seu lado FC tangencie a parábola. Toma-se um segmento HC para ser a alavanca, de modo que K seja ponto médio do segmento HC. Arquimedes então demonstra que, para qualquer haste curta, limitada na área formada pela parábola e segmento, a exemplo da haste OP, ao ser levada no ponto H, ficará em equilíbrio com a haste longa OM - com o mesmo se aplicando para qualquer fatia vertical. Disso, Arquimedes toma todas as hastes longas, que são infinitamente numerosas, e as substitui pelo ponto de gravidade do triângulo. Este ponto de gravidade fica 3 vezes mais perto do ponto K do que o ponto H. Segue disso que o peso do segmento parabólico deve ser de $\frac{1}{3}$ do peso do triângulo AFC, que é 4 vezes maior que ABC. Logo, a área delimitada pela parábola e pelo segmento segue uma proporção de $\frac{4}{3}$ do triângulo ABC. De fato, o Método das Alavancas é uma forma de se integrar uma região que já utilizava o conceito de infinito potencial (STROGATZ, 2019).

Passado alguns séculos sem muitos avanços nesse sentido, no início do século XVIII d.C., após o Renascimento Científico, é possível afirmar que os matemáticos Gottfried Wilhem Leibniz (1646-1716) e Sir Isaac Newton (1642-1726) estavam quase em guerra pela autoria do cálculo diferencial e integral (BARDI, 2008).

Suas descobertas só foram possíveis graças aos avanços de Fermat (1607-1665) e Descartes (1596-1650) no desenvolvimento de um sistema de coordenadas, conhecido como Plano Cartesiano, e a criação de métodos para calcular retas tangentes a uma curva dada, área em que Fermat se destacou (STROGATZ, 2019).

Newton desenvolveu sua teoria sobre cálculo no ano de 1666 porém não a publicou. Já Leibniz debruçou-se sobre o cálculo em 1675, criando, inclusive, suas próprias notações e símbolos, os quais são utilizados até hoje. Hoje, tanto Leibniz quanto Newton são considerados co-inventores independentes do cálculo (BARDI, 2008).

Nascem aí também as Equações Diferenciais, que são equações que envolvem funções e suas derivadas. As soluções para essas equações, no entanto, não eram tão fáceis. Segundo Alitolef (2010), Leibniz foi um dos primeiros a desenvolver métodos para esse tipo de equação, sendo um deles o Método de Separação de Variáveis. Newton e os irmãos Bernoulli também resolveram Equações Diferenciais advindas da Geometria e Mecânica.

Agora, cabe destacar a existência de dois tipos de Equações Diferenciais: Ordinárias e Parciais (FARLOW, 1993). As Equações Diferenciais Parciais (EDPs), são tipos de equações que possuem derivadas parciais, ou seja, diferentemente de Equações Diferenciais Ordinárias (EDOs) que possuem apenas uma variável, as EDPs podem possuir duas ou mais variáveis e suas derivadas, e elas serão o alvo de estudo neste trabalho.

Dentre as EDPs mais populares, a Equação do Calor foi alvo de estudo por Fourier (1768-1830). Ele desenvolveu um modelo físico que descrevia a propagação do calor utilizando séries trigonométricas, que ficaram conhecidas como Séries de Fourier (PIFER; AURANI, 2015). Essas séries auxiliaram na solução de EDPs pois funções trigonométricas são facilmente integráveis e deriváveis, possibiliando a solução de diversos tipos de EDPs.

Falando sobre soluções, essas equações podem ser abordadas de três formas principais: analítica, numérica e experimental. Enquanto a solução analítica busca uma formulação exata, muitas vezes é limitada a problemas simples ou bem definidos. As soluções experimentais, por outro lado, dependem de dados empíricos e podem não ser viáveis para todos os cenários. Portanto, as soluções numéricas assumem grande importância, especialmente quando se trata de problemas complexos para os quais as soluções analíticas são inacessíveis (ZILL; CULLEN, 2009). Uma das técnicas numéricas mais relevantes é o Método das Diferenças Finitas (MDF), que discretiza as equações, oferecendo uma aproximação prática e eficiente.

Ao longo da história, esse estudo sobre o método de separação de variáveis e o método de diferenças finitas foi expandido, sendo possível resolver a Equação do Calor para n dimensões e com variadas condições de contorno (ZILL, 2000).

Nota-se assim que as EDPs são um caso particular de modelagem matemática, a qual tem o objetivo de expressar situações por meio de modelos matemáticos, utilizando conceitos teóricos para fundamentar ou estimar soluções de problemas presentes no dia a dia (BEAN, 2001).

Sendo assim, este trabalho[1] terá como intuito verificar os resultados de uma Equação Bidimensional do Calor com Condições de Neumann - ou seja, como se propaga o calor em objetos bidimensionais com lados isolados e que não está perdendo calor para o ambiente, a fim de medir seu calor ao longo do tempo. Para tanto, basta que se aplique as seguintes Condições de Contorno (ZILL, 2000),

$$u(x, y, 0) = f(x, y), 0 < x < a, 0 < y < b, t > 0$$

$$u_x(0, y, t) = u_x(a, y, t) = 0$$

$$u_y(x, 0, t) = u_y(x, b, t) = 0$$

Obtendo sua solução analítica, por meio do método de separação de variáveis, e sua solução numérica, pelo método de diferenças finitas.

1.1 Estrutura do trabalho

Este trabalho se estrutura com capítulos descrevendo os objetivos, a justificativa, seguido da metodologia e referencial teórico. A partir disso, apresenta-se um capítulo para o método de solução analítica de separação de variáveis e outro capítulo com o método numérico de diferenças finitas e, por fim, a comparação entre os resultados.

1.2 Objetivos

1.2.1 Objetivo geral

Modelar a equação bidimensional do calor com condições de contorno de Neumann, obter a solução analítica da equação governante e comparar com a solução numérica obtida pelo método das diferenças finitas.

[1] Artigo publicado na Revista Eletrônica Científica Inovação e Tecnologia (RECIT), v. 14, n. 35 (2023). ISSN: 2175-1846.

1.2.2 Objetivos específicos

Baseando-se na Taxionomia de Bloom (BELHOT e FERRAZ, 2010), segue os objetivos específicos:

- Realizar um estudo sobre as equações diferenciais ordinárias e parciais;
- Reconhecer problemas de valor de contorno;
- Estudar o método de resolução analítico que será aplicado para resolver a equação do calor bidimensional;
- Compreender o problema de valor de contorno que representa a distribuição de temperatura em uma placa;
- Obter a solução analítica da equação do calor bidimensional aplicada a uma placa;
- Compreender as características e aplicabilidade do método das diferenças finitas;
- Obter a solução numérica por meio do método das diferenças finitas;
- Comparar a solução analítica e numérica utilizando a linguagem de programação Python.

1.3 Justificativa

É inegável a importância crescente dos métodos numéricos no âmbito da matemática aplicada, especialmente diante da complexidade dos problemas contemporâneos em ciências e engenharia. Ao longo da história, como demonstrado pelo Método das Alavancas de Arquimedes e as contribuições de Newton e Leibniz para o cálculo, a matemática tem sido uma ferramenta indispensável para o avanço científico e tecnológico. Entretanto, muitas equações diferenciais que surgem na realidade são intratáveis por métodos analíticos, especialmente quando envolvem geometrias ou condições de contorno mais complexas.

Sendo assim, os métodos numéricos, particularmente o MDF, oferecem uma alternativa confiável para obter soluções aproximadas para essas EDPs. Este método não apenas permite a modelagem de situações realísticas, mas também a simulação de cenários variados, possibilitando análises que seriam impossíveis ou impraticáveis por métodos puramente analíticos ou experimentais. Além disso, com o aumento da capacidade computacional disponível, esses métodos numéricos tornaram-se ainda mais essenciais e acessíveis para a solução de problemas multidimensionais complexos, como a Equação Bidimensional do Calor com Condições de Neumann abordada neste trabalho.

Assim, este estudo visa não só demonstrar a aplicabilidade e eficácia do MDF em um contexto prático específico, mas também contribuir para a compreensão mais ampla de como os

métodos numéricos podem ser utilizados para explorar, entender e resolver fenômenos físicos e de engenharia mais complexos.

2 MATERIAL E MÉTODOS

Inicialmente, foi feito um levantamento bibliográfico em revistas, livros, artigos, teses e trabalhos. A partir disso, estudou-se conceitos como Equações Diferenciais Parciais, Problema de Valor de Contorno, Python e suas bibliotecas, Método da Separação de Variáveis e o Método de Diferenças Finitas, além da própria Equação do Calor, alvo do trabalho.

Para tanto, é preciso que se estude com mais detalhes a Equação do Calor, incluindo sua classficação, exemplos, Condições de Contorno habituais e sua demonstração, para um melhor entendimento da natureza da Equação.

A partir disso, foi aplicado o Método de Separação de Variáveis para se encontrar a solução analítica com as Condições de Contorno dadas - a qual é uma solução exata. Já para a solução numérica - que é uma aproximação - foi utilizado o Método de Diferenças Finitas, calculado por meio do software PyCharm, uma interface de desenvolvimento integrado para programação em Python.

De posse da solução analítica e das equações que representam a aproximação numérica pelo MDF, fez-se a comparação entre as soluções. Essas soluções foram representadas graficamente, além da apresentação do erro relativo para critério de comparação.

Quanto à natureza desta pesquisa, é possível afirmar que é de natureza aplicada e com objetivo exploratório, pois tem como fim o aprimoramento do estudo da Equação do Calor Bidimensional com Condições de Contorno de Neumann, sendo assim também um Estudo de Caso (GIL, 2017). Além disso, o problema é abordado de maneira quantitativa, pois há o interessse de se encontrar as soluções analítica e numérica para a Equação do Calor com as condições dadas.

3 REFERENCIAL TEÓRICO

3.1 Equações Diferenciais Parciais

Equações Diferenciais Parciais (EDPs) são tipos de equações que possuem derivadas parciais. Ou seja, diferentemente de Equações Diferenciais Ordinárias (EDOs) que possuem apenas uma variável, EDPs podem possuir duas ou mais variáveis e suas derivadas.

Alguns exemplos de EDPs são (FARLOW, 1993):

$$u_t = u_{xx} \text{ (Equação Unidimensional do Calor)}$$
$$u_t = u_{xx} + u_{yy} \text{ (Equação Bidimensional do Calor)}$$
$$u_{tt} = u_{xx} + u_{yy} + u_{uu} \text{ (Equação Tridimensional da Onda)}$$
$$u_{tt} = u_{xx} + \alpha u_t + \beta u \text{ (Equação do Telégrafo)}$$

Pelos nomes das equações, percebe-se que as EDPs podem representar fenômenos físicos. Isso se dá porque, segundo Farlow (1993), as derivadas aparecem em fenômenos físicos pois elas descrevem problemas que envolvem velocidade, aceleração, força, fricção, fluxo e corrente. Sendo assim, é natural que, ao tentar encontrar uma função que descreva um fenômeno físico, se recaia em uma EDP.

3.2 Tipos de EDPs

Uma EDP pode ser classificada quanto à Ordem, Número de Variáveis, Linearidade, Homogeneidade, Coeficientes e Tipos.

- Ordem: é a ordem da maior derivada parcial da EDP.
- Número de Variáveis: é o número de variáveis independentes.
- Linearidade: seja u uma função de duas varíaveis da forma,

$$Au_{xx} + Bu_{xy} + Cu_{yy} + Du_x + Eu_y + Fu = G$$

  Se os coeficientes A, B, C, D, E, F e G são constantes, então u é denominada linear.

- Homogeneidade: se $\forall(x, y)/G(x, y) = 0$
- Coeficientes: os coeficientes podem ser constantes ou variáveis.
- Tipos: existem três tipos de EDPs para uma equação linear de 2 duas variáveis independentes:

  a) Parabólica, se $B^2 - 4AC = 0$

b) Hiperbólica, se $B^2 - 4AC > 0$

c) Elíptica, se $B^2 - 4AC < 0$

3.3 Problemas de Valor de Contorno

Uma EDP pode possuir infinitas soluções gerais. Dada uma função $u(x,t)$, obtém-se que a EDP $u_{tt} = 0$ teria a solução dada por $\Omega(x)t + \Gamma(x)$, sendo Ω e Γ funções arbitrárias da variável x. Dessa forma, para que se obtenha soluções particulares, é preciso que haja condições específicas - essas condições são chamadas de condições de contorno. O sistema formado com a EDP e as condições de contorno é chamado de Problema de Valor de Contorno (PVC) (IVRII, 2021).

Como será observado adiante, a solução de uma EDP é um tanto quanto meticulosa e, por seguinte, é preciso que se resolva uma única EDP com PVC bem definido. Dessa forma, considera-se a equação do calor bidimensional,

$$\frac{\partial u}{\partial t} = \alpha \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \tag{1}$$

Com condições de contorno de Neumann,

$$u_x(0, y, t) = u_x(a, y, t) = 0 \tag{2}$$

$$u_y(x, 0, t) = u_y(x, b, t) = 0 \tag{3}$$

$$u(x, y, 0) = f(x, y), \qquad 0 < x < a, 0 < y < b, t > 0 \tag{4}$$

3.4 Demonstração da Equação do Calor

Seja uma região arbitrária 3D subregião $V \subseteq \mathbb{R}$, com temperatura $u(X, t)$, definida em todos os pontos $X = (x, y, z)$.

Tem-se também que a energia do calor é dada por $c\rho u$, em que c é o calor específico, ρ é a densidade do objeto e u é a função que descreve o calor da subregião V ao longo do tempo (HANCOCK, 2006). Ao multiplicar $c\rho u$ por dV, procura-se a variação de energia do calor em uma pequena região de V,

$$\text{Variação de energia em uma região} = c\rho u \; dV \tag{5}$$

Ao integrar a nova expressão, soma-se essas variações de energias do calor, encontrando a energia total de V,

$$\text{Variação de Energia do calor em V} = \int\int\int_V \rho \, dV \tag{6}$$

Sabendo que,

$$\text{Taxa de mudança da energia do calor em } V = \text{Energia do calor dos contornos } S \text{ de } V + \text{energia do calor gerada no sólido por unidade de tempo} \tag{7}$$

É possível obter, assumindo $\phi(X,t) = -k_0\Delta u$, um vetor que define o caminho do fluxo de calor nos contornos e n a direção normal. Considerando que o calor transita do quente para o frio, o sinal dos contornos em S de V é negativo pois, se a energia do calor dos contornos é positiva, significa que a taxa de mudança de energia do calor em V está negativa - está saindo calor da superfície em V. Se a energia dos contornos em S de V é negativa, significa que está tendo uma variação positiva no interior da superfície V. Seja $Q(X,t)$ uma função que define o calor gerado no sólido.

$$\frac{d}{dt}\int\int\int_V c\rho u \, dV = -\int\int_S \phi \cdot n dS + \int\int\int_V Q \, dV \tag{8}$$

Pelo Teorema da Divergência de Gauss (HANCOCK, 2006),

$$\int\int\int_V \Delta \cdot A = \int\int_S A \cdot n dS \tag{9}$$

Sendo assim,

$$\frac{d}{dt}\int\int\int_V c\rho u \, dV = -\int\int\int_V \Delta \cdot \phi dV + \int\int\int_V Q \, dV \tag{10}$$

Procurando simplificar a expressão das integrais, subtrai-se os termos à esquerda da igualdade.

$$\int\int\int_V \left(c\rho \left(\frac{\partial u}{\partial t} \right) + \Delta \cdot \phi - Q \right) \, dV = 0 \tag{11}$$

Como a integral descrita em (11) é igual a 0, conclui-se que a expressão vale para qualquer volume V. Simplifica-se, assim,

$$c\rho \frac{\partial u}{\partial t} + \Delta \cdot \phi - Q = 0 \tag{12}$$

Como $\Delta \cdot \phi = -k_0\Delta^2 u$,

$$c\rho \frac{\partial u}{\partial t} = k_0\Delta^2 u + Q \tag{13}$$

Dividindo cp em ambos os termos,

$$\frac{\partial u}{\partial t} = \alpha \Delta^2 u + \frac{Q}{c\rho} \tag{14}$$

com $\alpha = \dfrac{k_0}{cp}$.

Considerando que não há calor gerado no sólido, $Q = 0$,

$$\frac{\partial u}{\partial t} = \alpha \Delta^2 u \tag{15}$$

4 SOLUÇÕES

4.1 Solução Analítica

Para obter a solução analítica da equação do calor bidimensional, equação (1), com condições de contorno de Neumann, equações (2), (3) e (4), aplicou-se o método de separação de variáveis.

4.1.1 Método da Separação de Variáveis

Suponha-se que a função u seja separável da seguinte forma (ZILL; CULLEN, 2009),

$$u(x, y, t) = F(x)G(y)H(t) \tag{16}$$

Em que F, G e H são funções de uma variável: x, y e t, respectivamente.
Sendo assim,

$$\frac{\partial u}{\partial t} = F(x)G(y)\frac{dH(t)}{dt} \tag{17}$$

$$\frac{\partial^2 u}{\partial x^2} = \frac{dF^2(x)}{dx^2}G(y)H(t) \tag{18}$$

$$\frac{\partial^2 u}{\partial y^2} = F(x)\frac{dG^2(y)}{dy^2}H(t) \tag{19}$$

Substituindo as expressões (17), (18) e (19) em (1), tem-se que,

$$F(x)G(y)\frac{dH(t)}{dt} = \alpha\left(\frac{dF^2(x)}{dx^2}G(y)H(t) + F(x)\frac{dG^2(y)}{dy^2}H(t)\right) \tag{20}$$

Considerando $\frac{dH(t)}{dt} = H_t$, $\frac{dF^2(x)}{dx^2} = F_{xx}$ e $\frac{dG^2(y)}{dy^2} = G_{yy}$ e dividindo ambos os lados de (20) por $\alpha F(x)G(y)H(t) \neq 0$, obtém-se,

$$\frac{H_t(t)}{\alpha H(t)} = \frac{F_{xx}(x)}{F(x)} + \frac{G_{yy}(y)}{G(y)} \tag{21}$$

Como o primeiro termo da igualdade está em função de t e o segundo termo está em função de x e y, é possível afirmar que ambos os lados são iguais a uma constante $-\lambda$. Sendo assim,

$$\frac{H_t(t)}{\alpha H(t)} = \frac{F_{xx}(x)}{F(x)} + \frac{G_{yy}(y)}{G(y)} = -\lambda \tag{22}$$

Separando (22) em duas expressões e, por conveniência, adotando $-\lambda = -\sigma^2$, é possível obter as seguintes equações,

$$\frac{H_t(t)}{\alpha H(t)} = -\sigma^2 \tag{23}$$

$$\frac{F_{xx}(x)}{F(x)} + \frac{G_{yy}(y)}{G(y)} = -\sigma^2 \tag{24}$$

Já na segunda expressão, em (24), nota-se que é possível deixar o primeiro termo da igualdade em função de x e o segundo termo em função de y, resultando novamente em uma constante,

$$\frac{F_{xx}(x)}{F(x)} = -\sigma^2 - \frac{G_{yy}(y)}{G(y)} = -\omega \tag{25}$$

Por conveniência, será adotado $-\omega$ = $-\xi^2$. Sendo assim , substituindo em (25),

$$\frac{F_{xx}(x)}{F(x)} = -\sigma^2 - \frac{G_{yy}(y)}{G(y)} = -\xi^2 \tag{26}$$

De (26), tomando $\xi^2 - \sigma^2 = -\gamma^2$ é possível obter 2 EDOs,

$$F_{xx}(x) = -\xi^2 F(x) \tag{27}$$

$$G_{yy}(y) = -\gamma^2 G(y) \tag{28}$$

Com isso, os problemas de Sturm-Liouville associados com as equações (27) e (28), considerando as condições de contorno, são,

$$F''(x) = -\xi^2 F(x),\ F'(0) = 0,\ F'(a) = 0. \tag{29}$$

$$G''(y) = -\gamma^2 G(y),\ G'(0) = 0,\ G'(a) = 0. \tag{30}$$

Recaindo, portanto, em EDOs.

4.1.2 Aplicando as condições de contorno

Analisando a solução de (29), verificam-se três casos: $-\xi^2 < 0$, $-\xi^2 = 0$ e $-\xi^2 > 0$. Para $-\xi^2 = 0$,

$$F''(x) = 0,\ F'(0) = 0,\ F'(a) = 0 \tag{31}$$

Segue daí que a solução da EDO é $F(x) = c_1 + c_2 x$. Mas a condição de contorno $F'(0) = 0$ implica que $c_2 = 0$. Nota-se que a segunda condição de contorno $F'(a) = 0$ não tem influência sobre o resultado constante. Sendo assim, $F(x) = c_1$ é uma solução não trivial.

Agora, para $\xi^2 < 0$,

$$F''(x) + \xi^2 F(x) = 0,\ F'(0) = 0,\ F'(a) = 0 \tag{32}$$

Assim, a solução da EDO é $F(x) = c_3 e^{\xi x} + c_4 e^{-\xi x}$. No entanto, ao aplicar a condição $F'(0) = 0$, percebe-se que $c_3 \xi e^{\xi x} - c_4 \xi e^{-\xi x} = 0$ implica em $c_3 = 0$ e $c_4 = 0$, já que $\xi \neq 0$. Assim, $F(x) = 0$, que é solução trivial.

Já para $\xi^2 > 0$, tem-se a expressão,

$$F''(x) + \xi^2 F(x) = 0,\ F'(0) = 0,\ F'(a) = 0 \tag{33}$$

Segue disso que a solução da EDO é $F(x) = c_5 \sin(\xi x) + c_6 \cos(\xi x)$.

Aplicando $F'(0) = 0$, obtém-se $F'(0) = c_5 \xi \cos(0) - c_6 \xi \sin(0)$, logo, $c_5 = 0$. Sendo assim, $F(x) = c_6 \cos(\xi x)$. Aplicando a segunda condição de contorno, $F'(a) = 0$, com $c_6 \xi \sin(\xi a) = 0$. $c_6 = 0$ resulta em solução trivial e, portanto, não há interesse em adotar $c_6 = 0$. Sendo assim, considerasse $c_6 \neq 0$. Como $\xi \neq 0$ por hipótese, então segue que $\sin(\xi a) = 0$. Assim, $\xi a = \pi m$ ou $\xi = \frac{\pi m}{a}$, em que $m = 1, 2, 3, ...$ Ou seja, a função tem soluções não triviais quando $-\omega_m = \xi_m^2 = \frac{\pi^2 m^2}{a^2}$. Logo, é preciso que $F(x)$, para $\xi^2 > 0$ seja,

$$F(x) = c_6 \cos\left(\frac{m\pi x}{a}\right),\ m = 1, 2, 3, ... \tag{34}$$

Logo, as soluções para $F(x)$ são,

$$F(x) = c_1 \text{ e } F(x) = c_6 \cos\left(\frac{m\pi x}{a}\right),\ m = 1, 2, 3, ... \tag{35}$$

De maneira análoga segue-se para $G(y)$ as duas soluções não triviais,

$$G(y) = c_7 \text{ e } G(y) = c_8 \cos\left(\frac{n\pi y}{b}\right),\ n = 1, 2, 3, ... \tag{36}$$

Além disso, $\gamma_n^2 = \frac{\pi^2 n^2}{b^2}$.

Já para obter $H(t)$, de (23), basta separá-las, obtendo $H(t) = k \cdot e^{-\sigma^2 \alpha t}$. Mas, considerando $\xi^2 - \sigma^2 = -\gamma^2$, $\xi_m^2 = \frac{\pi^2 m^2}{a^2}$ e $\gamma_n^2 = \frac{\pi^2 n^2}{b^2}$, tem-se que $H(t)$ é denotada por,

$$H(t) = k \cdot e^{-\left(\frac{\pi^2 m^2}{a^2} + \frac{\pi^2 n^2}{b^2}\right)\alpha t} = k \cdot e^{-\alpha \pi^2 \left(\frac{m^2}{a^2} + \frac{n^2}{b^2}\right) t} \tag{37}$$

Sendo assim, foi possível obter soluções para $F(x), G(y)$ e $H(t)$. Em seguida, basta unificá-las em uma combinação linear - para isso, aplica-se o princípio da superposição.

4.1.3 Princípio da Superposição

O Princípio da Superposição estabelece que, se $u_1, u_2, ..., u_k$ são soluções de uma equação diferencial parcial linear homogênea, então a combinação linear entre elas também é uma solução (ZILL; CULLEN, 2009). Sendo assim, combinando (35), (36) e (37), obtém-se,

$$\begin{aligned} u_{mn}(x,y,t) = A_{00} &+ A_{m0}\cos\left(\frac{m\pi x}{a}\right) e^{-\alpha\pi^2\left(\frac{m^2}{a^2}\right)t} \\ &+ A_{0n}\cos\left(\frac{n\pi y}{b}\right) e^{-\alpha\pi^2\left(\frac{n^2}{b^2}\right)t} \\ &+ A_{mn}\cos\left(\frac{n\pi y}{b}\right) e^{-\alpha\pi^2\left(\frac{m^2}{a^2}+\frac{n^2}{b^2}\right)t} \end{aligned} \tag{38}$$

Pelos dois conjuntos de autovalores, m e n, tem-se as seguintes somas,

$$\begin{aligned} u(x,y,t) = A_{00} &+ \sum_{m=1}^{\infty} A_{m0}\cos\left(\frac{m\pi x}{a}\right) e^{-\alpha\pi^2\left(\frac{m^2}{a^2}\right)t} \\ &+ \sum_{n=1}^{\infty} A_{0n}\cos\left(\frac{n\pi y}{b}\right) e^{-\alpha\pi^2\left(\frac{n^2}{b^2}\right)t} \\ &+ \sum_{n=1}^{\infty}\sum_{m=1}^{\infty} A_{mn}\cos\left(\frac{m\pi x}{a}\right)\cos\left(\frac{n\pi y}{b}\right) e^{-\alpha\pi^2\left(\frac{m^2}{a^2}+\frac{n^2}{b^2}\right)t} \end{aligned} \tag{39}$$

Segundo Cullen e Zill (2009), dado que na condição inicial tem-se $u(x,y,0)$, é possível representá-la com uma função $f(x,y)$, com apenas duas variáveis espaciais, já que $t = 0$. Sendo assim, é possível descrever a condição inicial simplificando (39) para,

$$\begin{aligned} F(x,y) = A_{00} &+ \sum_{m=1}^{\infty} A_{m0}\cos\left(\frac{m\pi x}{a}\right) + \sum_{n=1}^{\infty} A_{0n}\cos\left(\frac{n\pi y}{b}\right) \\ &+ \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} A_{mn}\cos\left(\frac{m\pi x}{a}\right)\cos\left(\frac{n\pi y}{b}\right) \end{aligned} \tag{40}$$

em um retângulo definido por $0 \leq x \leq a$ e $0 \leq y \leq b$.

Integrando em (40) sobre o retângulo definido,

$$\int_0^a \int_0^b F(x,y)dxdy = A_{00}\int_0^a \int_0^b dxdy + \int_0^a \int_0^b \sum_{m=1}^{\infty} A_{m0}\cos\left(\frac{m\pi x}{a}\right)dxdy + \int_0^a \int_0^b \sum_{n=1}^{\infty} A_{0n}\cos\left(\frac{n\pi y}{b}\right)dxdy + \int_0^a \int_0^b \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} A_{mn}\cos\left(\frac{m\pi x}{a}\right)\cos\left(\frac{n\pi y}{b}\right)dxdy \tag{41}$$

Como o conjunto das funções trigonométricas é ortogonal no intervalo de [-p, p] (ZILL; CULLEN, 2009), segue-se então que $\cos\left(\frac{m\pi x}{a}\right)$ e $\cos\left(\frac{n\pi y}{b}\right)$ são ortogonais no intervalo, para $n, m \geq 1$, reduzindo o lado direito a um único termo,

$$\int_0^a \int_0^b F(x,y)dxdy = A_{00}\int_0^a \int_0^b dxdy = A_{00}ab \tag{42}$$

Isolando A_{00},

$$A_{00} = \frac{1}{ab}\int_0^a \int_0^b F(x,y)dxdy \tag{43}$$

Agora, multiplicando (40) por $\cos\left(\frac{m\pi x}{a}\right)$, $\cos\left(\frac{n\pi y}{b}\right)$ e $\cos\left(\frac{m\pi x}{a}\right)\cos\left(\frac{n\pi y}{b}\right)$, respectivamente, integrando sobre o intervalo definido, tem-se,

$$A_{m0} = \frac{2}{ab}\int_0^a \int_0^b F(x,y)\cos\left(\frac{m\pi x}{a}\right)dxdy \tag{44}$$

$$A_{0n} = \frac{2}{ab}\int_0^a \int_0^b F(x,y)\cos\left(\frac{n\pi y}{b}\right)dxdy \tag{45}$$

$$A_{mn} = \frac{4}{ab}\int_0^a \int_0^b F(x,y)\cos\left(\frac{m\pi x}{a}\right)\cos\left(\frac{n\pi y}{b}\right)dxdy \tag{46}$$

Com isso, obteve-se os parâmetros necessários para fazer a solução de uma equação do calor bidimensional com condições de contorno de Neumann para qualquer $F(x,y)$ dada.

4.1.4 Definindo uma Condição Inicial

Até o presente momento, a solução analítica desenvolvida não está aplicada a uma função $f(x,y)$ definida. Dessa forma, tomando, por exemplo,

$$u(x,y,0) = t_0 \tag{47}$$

Neste caso, calculando os coeficientes através de (43), (44), (45) e (46),

$$A_{00} = t_0, A_{m0} = A_{0n} = A_{mn} = 0 \tag{48}$$

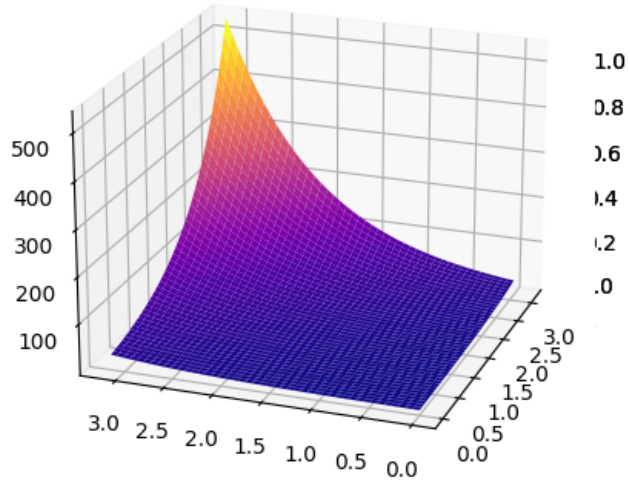
Sendo assim, $u(x, y, t)$ será dado por,

$$u(x, y, t) = t_0 \tag{49}$$

O que faz sentido com as condições de contorno do problema, já que os lados da placa estão isolados. Se a condição inicial é uma constante, então a função que define a temperatura da placa ao longo do tempo também deve ser constante, independentemente do tempo t, pelo Princípio da Conservação de Energia (O'NEIL, 2011).

Assumindo a condição inicial (Figura 2),

Figura 2 – Condição inicial



Fonte: Os autores (2023).

definida por,

$$u(x, y, 0) = e^{x+y} \tag{50}$$

Calculam-se os coeficientes por meio das equações (43), (44), (45) e (46),

$$A_{00} = \frac{1}{\pi^2} \cdot (e^{\pi} - 1)^2 \tag{51}$$

$$A_{m0} = \frac{2}{\pi^2} \cdot \frac{(e^{\pi} - 1) \cdot (e^{\pi}(-1)^m - 1)}{m^2 + 1} \tag{52}$$

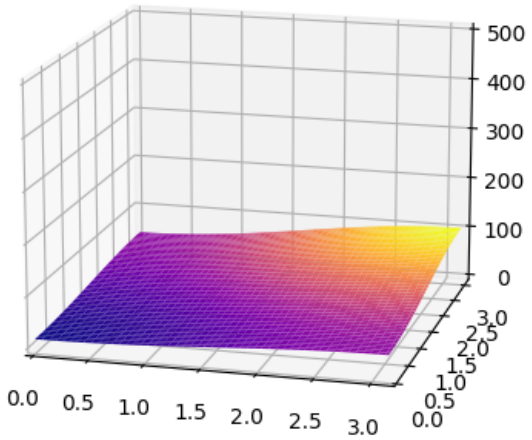
$$A_{0n} = \frac{2}{\pi^2} \cdot \frac{(e^{\pi} - 1) \cdot (e^{\pi}(-1)^n - 1)}{n^2 + 1} \tag{53}$$

$$A_{mn} = \frac{4}{\pi^2} \cdot \frac{(e^{\pi}(-1)^m - 1) \cdot (e^{\pi}(-1)^n - 1)}{(n^2 + 1) \cdot (m^2 + 1)} \tag{54}$$

Portanto, $u(x, y, t)$ será dado pela substituição desses coeficientes na função encontrada no Princípio da Superposição. Por exemplo, tomando $t = 100$, com parâmetros $m = 10$,

$n = 10$ e $\alpha = 0{,}01$, obtém-se, utilizando o código computacional do Apêndice A, (Figura 3 e 4)[1],

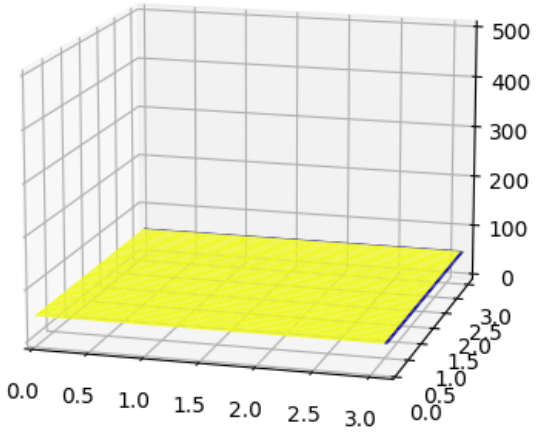
Figura 3 – Distribuição da temperatura para $t = 100$



Fonte: Os autores (2023).

A medida que $t \to \infty$ (Figura 4),

Figura 4 – Distribuição da temperatura para $t \to \infty$



Fonte: Os autores (2023).

a temperatura tende a se estabilizar em $u(x, y, t) = 49{,}66868$, já que, devido as condições iniciais, os lados estão isolados e não há troca de calor com o ambiente.

4.2 Solução Numérica

A solução numérica foi obtida pela aplicação do método das diferenças finitas explícito, descrito como segue.

Dada a equação bidimensional do calor, descrita em (1),

[1] Os códigos computacionais utilizados para gerar as imagens também podem ser encontrados em https://github.com/LeandroWrzecionek/2dHeatEquation/blob/main/Analítico.py

$$\frac{\partial u}{\partial t} = \alpha \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \tag{55}$$

em que α é o coeficiente de difusão.

A expansão em série de Taylor do valor de uma função $u(x)$ em $x = x_i + h$ em torno do valor $x = x_i$, é dada por (WROBEL, 1989),

$$u(x_i + h) = u(x_i) + h \left(\frac{du}{dx} \right)_{x=x_i} + \frac{h^2}{2} \left(\frac{d^2 u}{dx^2} \right)_{x=x_i} + \frac{h^3}{6} \left(\frac{d^3 u}{dx^2} \right)_{x=x_i} + \dots \tag{56}$$

De mesmo modo, obtém-se também, para $x = x_i - h$,

$$u(x_i - h) = u(x_i) - h \left(\frac{du}{dx} \right)_{x=x_i} + \frac{h^2}{2} \left(\frac{d^2 u}{dx^2} \right)_{x=x_i} - \frac{h^3}{6} \left(\frac{d^3 u}{dx^2} \right)_{x=x_i} + \dots \tag{57}$$

Subtraindo (57) de (56), resulta,

$$u(x_i + h) - u(x_i - h) = 2h \left(\frac{du}{dx} \right)_{x=x_i} + \frac{h^3}{3} \left(\frac{d^3 u}{dx^2} \right)_{x=x_i} + \dots \tag{58}$$

Isolando $\frac{du}{dx}$ determina-se a aproximação para a derivada primeira, chamada de diferença central, dada por,

$$\left(\frac{du}{dx} \right)_{x=x_i} = \frac{u(x_i + h) - u(x_i - h)}{2h} \tag{59}$$

que tem O(h^2) como ordem de convergência.[2]

O interesse de se utilizar a diferença central existe pois ela oferece uma precisão maior. Enquanto outros tipos de diferenças, como a progressiva e regressiva, têm uma precisão de primeira ordem O(h), a diferença central possui uma precisão de segunda ordem O(h^2), o que tende a produzir uma estimativa mais precisa da derivada.

Agora, para encontrar a aproximação de diferença central para a derivada de segunda ordem, soma-se (56) e (57),

$$u(x_i + h) + u(x_i - h) = 2u(x_i) + h^2 \left(\frac{d^2 u}{dx^2} \right)_{x=x_i} + \frac{h^4}{12} \left(\frac{d^4 u}{dx^4} \right)_{x=x_i} + \dots \tag{60}$$

Tendo como resultado,

[2] O termo $O(h^2)$ significa que, quando se diminui o tamanho do passo h, o erro diminui de acordo com a segunda potência de h

$$\left(\frac{d^2u}{dx^2}\right)_{x=x_i} = \frac{u(x_i+h) - 2u(x_i) + u(x_i-h)}{h^2} \tag{61}$$

com um erro O(h^2).

O desenvolvimento para determinar as derivadas parciais decorre da mesma forma, obtendo-se,

$$\left(\frac{\partial u(x,t)}{\partial x}\right)_{x=x_i} = \frac{u(x_i+h,t) - u(x_i-h,t)}{2h} \tag{62}$$

$$\left(\frac{\partial^2 u(x,t)}{\partial x^2}\right)_{x=x_i} = \frac{u(x_i+h,t) - 2u(x_i,t) + u(x_i-h,t)}{h^2} \tag{63}$$

com um erro O(h^2).

4.2.1 Método de Diferenças Finitas Explícito

Pode-se aproximar $u(x,y,t)$ pela função discreta $u_{i,j}^{(n)}$ em que $x = i\Delta x, y = j\Delta y$ e $t = n\Delta t$. Aplicando as equaçãos de diferenças (62) e (63), tem-se o método de diferenças finitas explícito de (1),

$$\frac{u_{i,j}^{(n+1)} - u_{i,j}^{(n)}}{\Delta t} = \alpha\left[\frac{u_{i+1,j}^{(n)} - 2u_{i,j}^{(n)} + u_{i-1,j}^{(n)}}{(\Delta x)^2} + \frac{u_{i,j+1}^{(n)} - 2u_{i,j}^{(n)} + u_{i,j-1}^{(n)}}{(\Delta y)^2}\right] \tag{64}$$

Logo, o estado do sistema com passo $n+1$ é dado por,

$$u_{i,j}^{(n+1)} = u_{i,j}^{(n)} + \alpha\Delta t\left[\frac{u_{i+1,j}^{(n)} - 2u_{i,j}^{(n)} + u_{i-1,j}^{(n)}}{(\Delta x)^2} + \frac{u_{i,j+1}^{(n)} - 2u_{i,j}^{(n)} + u_{i,j-1}^{(n)}}{(\Delta y)^2}\right] \tag{65}$$

Tomando $\Delta x = \Delta y = h$,

$$u_{i,j}^{(n+1)} = u_{i,j}^{(n)} + \alpha\Delta t\left[\frac{u_{i+1,j}^{(n)} + u_{i-1,j}^{(n)} - 4u_{i,j}^{(n)} + u_{i,j+1}^{(n)} + u_{i,j-1}^{(n)}}{h^2}\right] \tag{66}$$

Ou ainda,

$$u_{i,j}^{(n+1)} = u_{i,j}^{(n)} \cdot \left(1 - \frac{4\alpha\Delta t}{h^2}\right) + \alpha\Delta t\left[\frac{u_{i+1,j}^{(n)} + u_{i-1,j}^{(n)} + u_{i,j+1}^{(n)} + u_{i,j-1}^{(n)}}{h^2}\right] \tag{67}$$

O método de diferenças finitas explícito para a equação bidimensional do calor só é estável quando (ALIYU; OLATOYINBO, 2021),

$$1 - \frac{4\alpha\Delta t}{h^2} \geq 0 \tag{68}$$

Segue que,

$$1 \geq \frac{4\alpha\Delta t}{h^2} \tag{69}$$

Ou ainda,

$$\frac{h^2}{4\alpha} \geq \Delta t \tag{70}$$

Agora, considerando as condições de contorno de Neumann, (2) e (3), é possível discretizá-las,

$$u_x = \frac{f(x + \Delta x, y, t) - f(x - \Delta x, y, t)}{2\Delta x} = 0 \tag{71}$$

Logo, para $x = 0$,

$$u_x = \frac{f(0 + \Delta x, y, t) - f(0 - \Delta x, y, t)}{2\Delta x} = 0 \tag{72}$$

$$\frac{f(\Delta x, y, t) - f(-\Delta x, y, t)}{2\Delta x} = 0 \tag{73}$$

$$\frac{f(\Delta x, y, t) - f(-\Delta x, y, t)}{2\Delta x} = 0 \tag{74}$$

$$f(\Delta x, y, t) - f(-\Delta x, y, t) = 0 \tag{75}$$

$$f(-\Delta x, y, t) = f(\Delta x, y, t) \tag{76}$$

Ou seja, $u_{-1,j}^{(n)} = u_{1,j}^{(n)}$, para $x = 0$.
Sendo assim, em (66), para $x = 0$, isso resulta em,

$$u_{0,j}^{(n+1)} = u_{0,j}^{(n)} \cdot \left(1 - \frac{4\alpha\Delta t}{h^2}\right) + \alpha\Delta t \left[\frac{2u_{1,j}^{(n)} + u_{0,j+1}^{(n)} + u_{0,j-1}^{(n)}}{h^2}\right] \tag{77}$$

De modo análogo para $y = 0$,

$$f(x, -\Delta y, t) = f(x, \Delta y, t) \tag{78}$$

Assim, em (77),

$$u_{0,0}^{(n+1)} = u_{0,0}^{(n)} \cdot \left(1 - \frac{4\alpha\Delta t}{h^2}\right) + \alpha\Delta t \left[\frac{2u_{1,0}^{(n)} + 2u_{0,1}^{(n)}}{h^2}\right] \tag{79}$$

Os demais pontos da fronteira são feitos de maneira análoga. Dados $(x = K, y = L)$, para o canto superior direito , tem-se,

$$u_{M,N}^{(n+1)} = u_{M,N}^{(n)} \left(1 - \frac{4\alpha\Delta t}{h^2}\right) + \alpha\Delta t \left(\frac{2u_{M-1,N}^{(n)} + 2u_{M,N-1}^{(n)}}{h^2}\right) \tag{80}$$

Já para o canto inferior direito $(x = L, y = 0)$, obtém-se,

$$u_{M,0}^{(n+1)} = u_{M,0}^{(n)} \left(1 - \frac{4\alpha\Delta t}{h^2}\right) + \alpha\Delta t \left(\frac{2u_{M-1,0}^{(n)} + 2u_{M,1}^{(n)}}{h^2}\right) \tag{81}$$

Por fim, o canto superior esquerdo $(x = 0, y = L)$ é dado por,

$$u_{0,N}^{(n+1)} = u_{0,N}^{(n)} \left(1 - \frac{4\alpha\Delta t}{h^2}\right) + \alpha\Delta t \left(\frac{2u_{1,N}^{(n)} + 2u_{0,N-1}^{(n)}}{h^2}\right) \tag{82}$$

Dessa forma, ficam estabelecidas as equações de diferenças finitas para os ponteiros interiores e da fronteira do domínio de resolução.

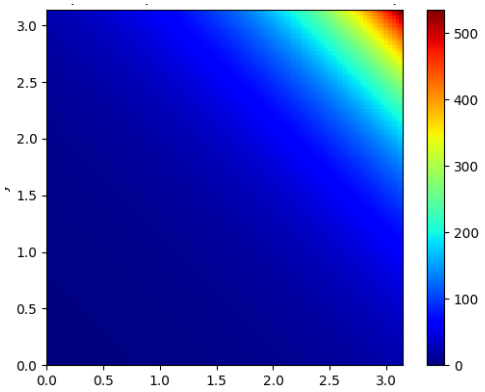
4.2.2 Obtenção da solução numérica

A solução numérica toma uma discretização finita dos pontos da placa. Então, a fim de se obter uma malha de pontos mais refinada, com um número grande mas ainda finito de pontos, utilizou-se o software *PyCharm* para o desenvolvimento dos códigos computacionais, na linguagem *Python*, que fizesse a solução numérica por meio do método de diferenças finitas explícito.

Para solucionar (67), portanto, adotou-se $h = \frac{\pi}{100}$, para discretizar a placa quadrada de medida π em uma malha de 100x100 pontos, $\alpha = 0{,}01$ para que o calor se propague lentamente e $\Delta t = 0{,}01$, para que se tenha um menor erro. Disso, tem-se, utilizando o código computacional em *Python* do Apêndice B, com os tempos $t = 0$ e $t = 10$ para que a condição de estabilidade (70) seja respeitada. Assim, para tempo $t = 0$ obtém-se (Figura 5) [3],

[3] Os códigos computacionais utilizados para gerar as imagens também podem ser encontrados em https://github.com/LeandroWrzecionek/2dHeatEquation/blob/main/Numérico%20Explícito.py

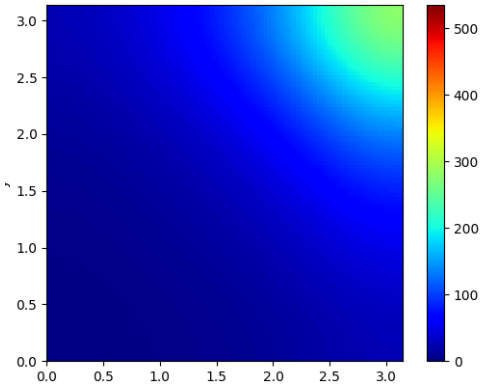
Figura 5 – Exemplo do Método de Diferenças Finitas Explícito para $t = 0$



Fonte: Os autores (2023).

Já para $t = 10$, tem-se (Figura 6),

Figura 6 – Exemplo do Método de Diferenças Finitas Explícito para $t = 10$



Fonte: Os autores (2023).

5 COMPARAÇÃO DOS RESULTADOS

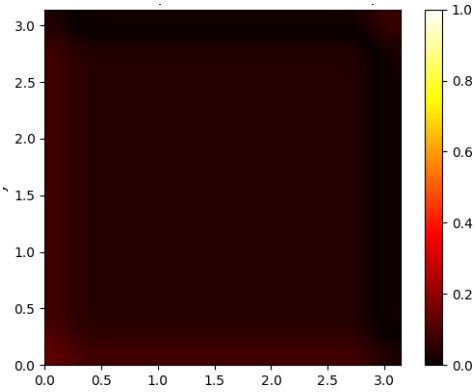
Para comparar os resultados, foi adotado o erro relativo percentual entre os valores, de cada ponto, dado por,

$$ER_x = \frac{|x - \overline{x}|}{\overline{x}} \cdot 100 \tag{83}$$

Sendo, neste caso, $\overline{x}$ a aproximação numérica e x a solução analítica.

Considerando $h = \frac{\pi}{100}$, $\alpha = 0{,}01$ e $\Delta t = 0{,}005$, para obter, como anteriormente, uma malha com um grande número de pontos, utilizando o código computacional do Apêndice C, tem-se para $t = 1$ (Figura 7) [4],

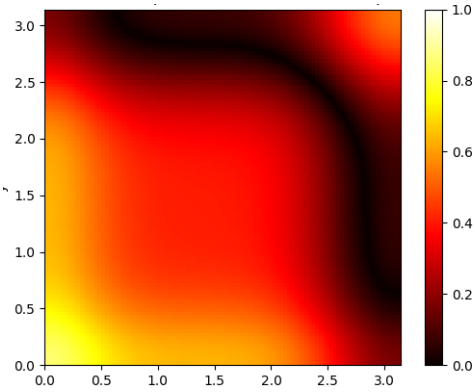
Figura 7 – Erro relativo para $t = 1$



Fonte: Os autores (2023).

E para $t = 10$ (Figura 8),

Figura 8 – Erro relativo para $t = 10$



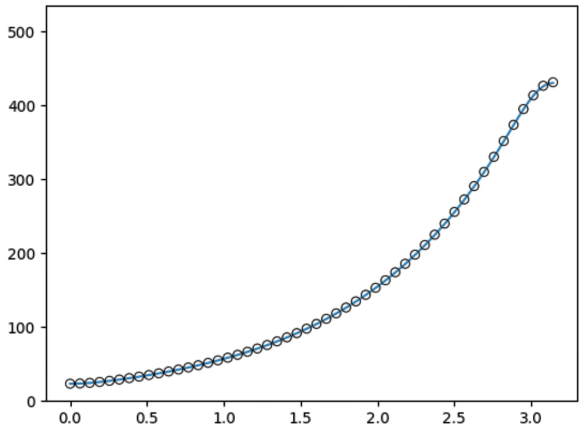
Fonte: Os autores (2023).

Observa-se que o erro relativo inicial tem valores próximos a 0,25%, no entanto, o erro acumulado ao longo do tempo faz com que, para $t = 10$, haja pontos com erro relativo em 0,8% para os parâmetros escolhidos.

[4] Os códigos computacionais utilizados para gerar as imagens podem ser encontrados também em: https://github.com/LeandroWrzecionek/2dHeatEquation/blob/main/Erro%20Relativo.py.

Ainda, é possível comparar os resultados de ambas as soluções por meio de seus gráficos. Fixando $y = \pi$, tem-se para $t = 1$ (Figura 9), com solução analítica em azul e solução numérica representada por circunferências pretas,

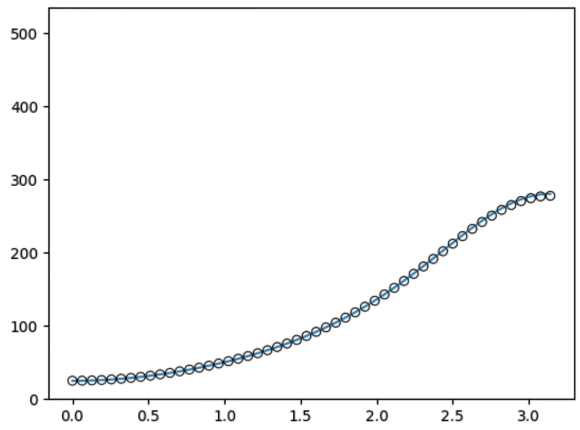
Figura 9 – Erro relativo para $y = \pi$ e $t = 1$



Fonte: Os autores (2023).

Já para $t = 10$ (Figura 10),

Figura 10 – Erro relativo para $y = \pi$ e $t = 10$



Fonte: Os autores (2023).

Nota-se, graficamente, que há grande concordância entre as soluções analítica e numérica, o que é confirmado pelo baixo erro relativo.

6 CONCLUSÃO

Dado o exposto, nota-se que o a solução numérica pelo método de diferenças finitas é uma excelente aproximação para a solução analítica do PVC proposto, podendo ser ainda mais precisa quando se considera mais pontos na discretização. Apesar disso, percebe-se ainda que o método de diferenças finitas explícito acumula o erro ao longo do tempo, o que pode gerar um erro relativo considerável para grandes quantidades de tempo, principalmente nos cantos da placa. Sugere-se também avaliar a forma implícita do MDF na busca de obter uma solução mais precisa, com a implementação do MDF em uma linguagem compilada, como C, Fortran ou C#, já que a demanda computacional se torna grande por conta da resolução de um sistema linear a cada iteração.

REFERÊNCIAS

ALITOLEF, S. dos S. História das equações diferenciais e algumas de suas aplicações. **X SEMANA DE MATEMÁTICA**, p. 74, 2010.

ALIYU, B.; OLATOYINBO, S. **Explicit and Implicit Solutions to 2-D Heat Equation**. 2021.

BARDI, J. S. A guerra do cálculo. 2008.

BEAN, D. O que é modelagem matemática? **Educação matemática em revista**, Sociedade Brasileira de Educação Matemática, v. 8, n. 9/10, p. 49–57, 2001.

BECKER, T. W.; KAUS, B. J. Numerical modeling of earth systems. **University of Southern California**, v. 1, 2016.

FARLOW, S. J. **Partial differential equations for scientists and engineers**. [*S.l.*]: Courier Corporation, 1993.

GIL, A. C. **Como elaborar projetos de pesquisa**. 6. ed. São Paulo, SP: Atlas, 2017.

HANCOCK, M. J. **The heat and wave equations in 2D and 3D**. [*S.l.*]: MIT: Fall, 2006.

IVRII, V. **Partial Differential Equations**. [*S.l.*: *s.n.*], 2021. 2021. Disponível em: http://www.math.toronto.edu/ivrii/PDE-textbook. Acesso em 24 mar. 2023.

O'NEIL, P. V. **Beginning partial differential equations**. [*S.l.*]: John Wiley & Sons, 2011. v. 85.

PIFER, A.; AURANI, K. M. A teoria analítica do calor de joseph fourier: uma análise das bases conceituais e epistemológicas. **Revista Brasileira de Ensino de Física**, SciELO Brasil, v. 37, p. 1603, 2015.

PRUNZEL, T. P. Equação do calor unidimensional: resolução analítica e computacional. 2021.

SANDERSON, B. W. **The Sunlit Man**. Dragonsteel Entertainment, 2023. ISBN 9781938570414. Disponível em: https://www.google.com.br/books/edition/The_Sunlit_Man/uomkEAAAQBAJ?hl=pt-BR&gbpv=0. Acesso em: 10 out. 2023.

STROGATZ, S. **O poder do infinito**: Como o cálculo revela os segredos do universo. [*S.l.*]: Sextante, 2019.

WROBEL, L. C. **Métodos numéricos em recursos hídricos**. [*S.l.*]: Associação Brasileira de Recursos Hídricos, 1989.

ZILL, D. G.; CULLEN, M. R. **Matemática Avançada para Engenharia**. [*S.l.*]: Bookman Editora, 2009. v. 1.

APÊNDICE A – Código Computacional da Solução Analítica

```
1  import numpy as np
2  import  pandas
3  import math as c
4  import matplotlib.pyplot as plt
5  ax = plt.axes(projection='3d')
6  #número de pontos
7  pt = 30
8  #alpha da função, velocidade que o calor se espalha
9  alpha = 0.01
10 #tempo
11 t = 10
12 #limita de série
13 series=50
14 #pontos do período de 0 a pi em x a y
15 x = np.linspace(0, c.pi, pt)
16 y = np.linspace(0, c.pi, pt)
17 #é preciso do mesmo número de pontos psoteriormente, por isso
      crio 2 ndarray vazias
18 x2 = np.linspace(0, 0, pt)
19 y2 = np.linspace(0, 0, pt)
20 #criação das ndarray
21 X, Y = np.meshgrid(x, y)
22 X2, Y2 = np.meshgrid(x2, y2)
23 #A00 definido por Fourier
24 A00 = (1/(c.pi**2))*(c.e**c.pi -1)**2
25 #Série com Am0 Fourier
26 Xsum = []
27 for p in range(0, pt):
28     Xsoma = 0
29     for m in range(1, series):
```

```
30          Am0 = (2/(c.pi**2))*(((c.e**(c.pi) - 1)*(c.e**(c.pi)
    *(-1)**(m)-1))/(m**(2)+1))*c.cos(X[0][p]*m)*c.e**(-alpha*m
    **(2)*t)
31          Xsoma = Xsoma + Am0
32      Xsum.append(Xsoma)
33  #Série com A0n Fourier
34  Ysum = []
35  for q in range(0, pt):
36      Ysoma = 0
37      for n in range(1, series):
38          A0n = (2/(c.pi**2))*(((c.e**(c.pi) - 1)*(c.e**(c.pi)
    *(-1)**(n)-1))/(n**(2)+1))*c.cos(Y[q][0]*n)*c.e**(-alpha*n
    **(2)*t)
39          Ysoma = Ysoma + A0n
40      Ysum.append(Ysoma)
41  #Série com Amn Fourier. Aqui que preciso da ndarray vazio,
42  #para conseguir criar uma tabela com pt x pt e preenchê-la com
    seus respectivos valores de z
43  XYsum = X2
44  for r in range(0, pt):
45      for s in range(0,pt):
46          XYsoma = 0
47          for n in range(1, series):
48              for m in range(1, series):
49                  Amn = (4/(c.pi**2))*(((c.e**(c.pi)*(-1)**(m)-1)
    *((c.e**(c.pi)*(-1)**(n)-1)))/((n**(2)+1)*(m**(2)+1)))*c.cos
    (X[0][s]*m)*c.cos(Y[r][0]*n)*c.e**(-alpha*(m**2+n**2)*t)
50                  XYsoma = XYsoma + Amn
51          XYsum[s][r] = XYsoma
52  #soma de todos os valores
53  SOMAFINAL = Y2
54  for t in range (0, pt):
```

```
55      for u in range (0, pt):
56          SOMAFINAL[u][t] = A00 + Xsum[u] + Ysum[t] +XYsum[u][t]
57  ax.plot_surface(X, Y, SOMAFINAL, cmap='hot')
58  ax.set_zlim(0, 500)
59  ax.set_xlim(0, c.pi)
60  ax.set_ylim(0, c.pi)
61  plt.show()
```

APÊNDICE B – Código Computacional da Solução Numérica

```
1  import numpy as np
2  import  pandas
3  import math as c
4  import matplotlib.pyplot as plt
5  import matplotlib.animation as animation
6  from matplotlib.animation import FuncAnimation
7  #tamanha da placa, alpha, número de pontos, Dt, etc
8  max_iter_time = 1001
9  pt = 100
10 plate_length = pt
11 alpha = 0.01
12 Dt = 0.01
13 h = c.pi/pt
14 if (1 - (4*alpha*Dt)/h**2) <= 0:
15     print("Valores errados para o método explícito")
16 else:
17     #pontos do período de 0 a pi em x a y
18     x = np.linspace(0, c.pi, pt)
19     y = np.linspace(0, c.pi, pt)
20     X, Y = np.meshgrid(x, y)
21     u = np.empty((max_iter_time, plate_length, plate_length))
22     #valores da temperatura na condição inicial
23     u[0] = c.e**(X+Y)
24     def calculate(u):
25         #pontos internos
26         for l in range (0, max_iter_time-1, 1):
27             for i in range (0, pt-1):
28                 for j in range (0, pt-1):
29                     u[l+1][i][j] = u[l][i][j]*(1 - (4*alpha*Dt)
   /h**2) + alpha * Dt * ((u[l][i+1][j]+u[l][i][j+1]+u[l][i-1][
   j]+u[l][i][j-1])/h**2)
30             #bordas sem cantos
```

```
31              for k in range (0, pt-1):
32                  u[l+1][0][k] = u[l][0][k] * (1 - (4 * alpha *
    Dt) / h ** 2) + alpha * Dt * ((2*u[l][1][k] + u[l][0][k+1] +
     u[l][0][k - 1]) / h ** 2)
33                  u[l+1][pt-1][k] = u[l][pt-1][k] * (1 - (4 *
    alpha * Dt) / h ** 2) + alpha * Dt * ((2 * u[l][pt-2][k] + u
    [l][pt-1][k + 1] + u[l][pt-1][k - 1]) / h ** 2)
34                  u[l+1][k][0] = u[l][k][0] * (1 - (4 * alpha *
    Dt) / h ** 2) + alpha * Dt * ((u[l][k+1][0] + 2* u [l][k][1]
     + u[l][k-1][0]) / h ** 2)
35                  u[l+1][k][pt-1] = u[l][k][pt-1] * (1 - (4 *
    alpha * Dt) / h ** 2) + alpha * Dt * ((u[l][k + 1][pt-1] + 2
     * u[l][k][pt-2] + u[l][k - 1][pt-1]) / h ** 2)
36              #4 cantos
37              u[l+1][0][0]= u[l][0][0]*(1 - (4*alpha*Dt)/h**2) +
    alpha * Dt * ((2*u[l][1][0]+2*u[l][0][1])/h**2)
38              u[l+1][pt-1][pt-1]= u[l][pt-1][pt-1]*(1 - (4*alpha*
    Dt)/h**2) + alpha * Dt * ((2*u[l][pt-2][pt-1]+2*u[l][pt-1][
    pt-2])/h**2)
39              u[l+1][pt-1][0]= u[l][pt-1][0]*(1 - (4*alpha*Dt)/h
    **2) + alpha * Dt * ((2*u[l][pt-2][0]+2*u[l][pt-1][1])/h**2)
40              u[l+1][0][pt-1]= u[l][0][pt-1]*(1 - (4*alpha*Dt)/h
    **2) + alpha * Dt * ((2*u[l][0][pt-2]+2*u[l][1][pt-1])/h**2)
41          return u
42      u = calculate(u)
```

APÊNDICE C – Erro Relativo entre a Solução Analítica e Numérica

```
1  import numpy as np
2  import  pandas
3  import math as c
4  import matplotlib.pyplot as plt
5  import matplotlib.animation as animation
6  #número de pontos
7  pt = 100
8  #alpha da função, velocidade que o calor se espalha
9  alpha = 0.01
10 #tempo
11 t = 10
12 #limita de série
13 series = 100
14 #Delta T
15 Dt = 0.005
16 #max_iter_time tem que ser verificado para os valores de Dt e T
17 max_iter_time = int(t/Dt)+1
18 plate_length = pt
19 h = c.pi/pt
20 #pontos do período de 0 a pi em x a y
21 x = np.linspace(0, c.pi, pt)
22 y = np.linspace(0, c.pi, pt)
23 #é preciso do mesmo número de pontos psoteriormente, por isso
      crio 2 ndarray vazias
24 x2 = np.linspace(0, 0, pt)
25 y2 = np.linspace(0, 0, pt)
26 #criação das ndarray
27 X, Y = np.meshgrid(x, y)
28 X2, Y2 = np.meshgrid(x2, y2)
29 #A00 definido por Fourier
30 A00 = (1/(c.pi**2))*(c.e**c.pi -1)**2
31 #Série com Am0 Fourier
```

```
Xsum = []
for p in range(0, pt):
    Xsoma = 0
    for m in range(1, series):
        Am0 = (2/(c.pi**2))*(((c.e**(c.pi) - 1)*(c.e**(c.pi)
   *(-1)**(m)-1))/(m**(2)+1))*c.cos(X[0][p]*m)*c.e**(-alpha*m
   **(2)*t)
        Xsoma = Xsoma + Am0
    Xsum.append(Xsoma)
#Série com A0n Fourier
Ysum = []
for q in range(0, pt):
    Ysoma = 0
    for n in range(1, series):
        A0n = (2/(c.pi**2))*(((c.e**(c.pi) - 1)*(c.e**(c.pi)
   *(-1)**(n)-1))/(n**(2)+1))*c.cos(Y[q][0]*n)*c.e**(-alpha*n
   **(2)*t)
        Ysoma = Ysoma + A0n
    Ysum.append(Ysoma)
#Série com Amn Fourier. Aqui que preciso da ndarray vazio,
#para conseguir criar uma tabela com pt x pt e preenchê-la com
   seus respectivos valores de z
XYsum = X2
for r in range(0, pt):
    for s in range(0,pt):
        XYsoma = 0
        for n in range(1, series):
            for m in range(1, series):
                Amn = (4/(c.pi**2))*(((c.e**(c.pi)*(-1)**(m)-1)
   *((c.e**(c.pi)*(-1)**(n)-1)))/((n**(2)+1)*(m**(2)+1)))*c.cos
   (X[0][s]*m)*c.cos(Y[r][0]*n)*c.e**(-alpha*(m**2+n**2)*t)
                XYsoma = XYsoma + Amn
```

```
57          XYsum[s][r] = XYsoma
58  #soma de todos os valores
59  Algébrico = Y2
60  for tn in range (0, pt):
61      for un in range (0, pt):
62          Algébrico[un][tn] = A00 + Xsum[un] + Ysum[tn] +XYsum[un
        ][tn]
63  if (1 - (4*alpha*Dt)/h**2) <= 0:
64      print("Valores errados para o método explícito")
65  else:
66      u = np.empty((max_iter_time, plate_length, plate_length))
67      #valores da temperatura na condição inicial
68      u[0] = c.e**(X+Y)
69      def calculate(u):
70          #pontos internos
71          for l in range (0, max_iter_time-1, 1):
72              for i in range (0, pt-1):
73                  for j in range (0, pt-1):
74                      u[l+1][i][j] = u[l][i][j]*(1 - (4*alpha*Dt)
        /h**2) + alpha * Dt * ((u[l][i+1][j]+u[l][i][j+1]+u[l][i-1][
        j]+u[l][i][j-1])/h**2)
75              #bordas sem cantos
76              for k in range (0, pt-1):
77                  u[l+1][0][k] = u[l][0][k] * (1 - (4 * alpha *
        Dt) / h ** 2) + alpha * Dt * ((2*u[l][1][k] + u[l][0][k+1] +
         u[l][0][k - 1]) / h ** 2)
78                  u[l+1][pt-1][k] = u[l][pt-1][k] * (1 - (4 *
        alpha * Dt) / h ** 2) + alpha * Dt * ((2 * u[l][pt-2][k] + u
        [l][pt-1][k + 1] + u[l][pt-1][k - 1]) / h ** 2)
79                  u[l+1][k][0] = u[l][k][0] * (1 - (4 * alpha *
        Dt) / h ** 2) + alpha * Dt * ((u[l][k+1][0] + 2* u [l][k][1]
         + u[l][k-1][0]) / h ** 2)
```

```
80                  u[l+1][k][pt-1] = u[l][k][pt-1] * (1 - (4 *
    alpha * Dt) / h ** 2) + alpha * Dt * ((u[l][k + 1][pt-1] + 2
     * u[l][k][pt-2] + u[l][k - 1][pt-1]) / h ** 2)
81                  #4 cantos
82                  u[l+1][0][0]= u[l][0][0]*(1 - (4*alpha*Dt)/h**2) +
    alpha * Dt * ((2*u[l][1][0]+2*u[l][0][1])/h**2)
83                  u[l+1][pt-1][pt-1]= u[l][pt-1][pt-1]*(1 - (4*alpha*
    Dt)/h**2) + alpha * Dt * ((2*u[l][pt-2][pt-1]+2*u[l][pt-1][
    pt-2])/h**2)
84                  u[l+1][pt-1][0]= u[l][pt-1][0]*(1 - (4*alpha*Dt)/h
    **2) + alpha * Dt * ((2*u[l][pt-2][0]+2*u[l][pt-1][1])/h**2)
85                  u[l+1][0][pt-1]= u[l][0][pt-1]*(1 - (4*alpha*Dt)/h
    **2) + alpha * Dt * ((2*u[l][0][pt-2]+2*u[l][1][pt-1])/h**2)
86              return u
87       Numérico = calculate(u)
88  relativeerror = ((np.absolute(Algébrico - Numérico[int(t/Dt)]))
    /Numérico[int(t/Dt)])*100
89  def plotheatmap(relativeerror_k, k):
90              # Clear the current plot figure
91              plt.clf()
92              plt.title(f"Erro relativo para t = {t} unidade de tempo
    ")
93              plt.xlabel("x")
94              plt.ylabel("y")
95              plt.xlim(0, c.pi)
96              plt.ylim(0, c.pi)
97              # This is to plot u_k (u at time-step k)
98              plt.imshow(relativeerror_k, cmap='hot', vmin=0, vmax=1,
99                         extent=[0, c.pi, c.pi, 0])  # Definindo os
    limites dos eixos x e y
100             plt.colorbar()
101             return plt
```

```
102 def animate(k):
103     plotheatmap(relativeerror, k)
104 anim = animation.FuncAnimation(plt.figure(), animate, interval
    =10, frames=max_iter_time, repeat=False)
105 plt.show()
106 print("Done!")
```