
# ANÁLISE DE CONDUÇÃO DE CALOR EM COMPONENTES ELETRÔNICOS DE MATERIAL HETEROGÊNEO UTILIZANDO MÉTODO DE ELEMENTOS FINITOS

Victor Guido Pinheiro

Projeto de Graduação apresentado ao Curso de Engenharia Mecânica da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientador: Gustavo Rabello dos Anjos, Ph.D.

Rio de Janeiro

Março de 2022

*UNIVERSIDADE FEDERAL DO RIO DE JANEIRO*

Departamento de Engenharia Mecânica

DEM/POLI/UFRJ

# ANÁLISE DE CONDUÇÃO DE CALOR EM COMPONENTES ELETRÔNICOS DE MATERIAL HETEROGÊNEO UTILIZANDO MÉTODO DE ELEMENTOS FINITOS

Victor Guido Pinheiro

PROJETO FINAL SUBMETIDO AO CORPO DOCENTE DO DEPARTAMENTO DE ENGENHARIA MECÂNICA DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO MECÂNICO.

Aprovada por:

Prof. Gustavo Rabello dos Anjos, Ph.D.,

Prof. Gabriel Lisbôa Verissimo, D.Sc.

Prof. Daniel Alves Castello, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

MARÇO DE 2022

Guido Pinheiro, Victor

ANÁLISE DE CONDUÇÃO DE CALOR EM COMPONENTES ELETRÔNICOS DE MATERIAL HETEROGÊNEO UTILIZANDO MÉTODO DE ELEMENTOS FINITOS/ Victor Guido Pinheiro. – Rio de Janeiro: UFRJ/Escola Politécnica, 2022.

X, 76 p.: il.; 29,7cm.

Orientador: Gustavo Rabello dos Anjos, Ph.D.

Projeto de Graduação – UFRJ/ Escola Politécnica/ Curso de Engenharia Mecânica, 2022.

1. Condução de Calor. 2. Material Heterogêneo. 3. Método de Elementos Finitos. I. Ph.D., Gustavo Rabello dos Anjos,. II. Universidade Federal do Rio de Janeiro, UFRJ, Curso de Engenharia Mecânica. III. ANÁLISE DE CONDUÇÃO DE CALOR EM COMPONENTES ELETRÔNICOS DE MATERIAL HETEROGÊNEO UTILIZANDO MÉTODO DE ELEMENTOS FINITOS.

# Agradecimentos

Agradeço primeiramente à Deus, por cada porta aberta e oportunidade concedida, por ser meu Senhor e Salvador.

Agradeço aos meus pais, Delce Guido Pinheiro Jr. e Tatiana Fischer Guio Pinheiro por todo amor, apoio, cuidado e sacrifícios durante todos esses anos. Agradeço também ao meu irmão, Mateus Guido Pinheiro, por todos os conselhos e momentos de companheirismo.

Agradeço também à Camilla Martins de Almeida Figueiredo Rangel, por tudo nesses quase dez anos juntos, por todo o apoio nessa jornada, não apenas na parte acadêmica, mas em todas as áreas da minha vida.

Agradeço também ao meu orientador Gustavo Rabello dos Anjos por todo auxílio, paciência, compreensão e ensinamentos essenciais que tornaram este trabalho possível.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro Mecânico

# ANÁLISE DE CONDUÇÃO DE CALOR EM COMPONENTES ELETRÔNICOS DE MATERIAL HETEROGÊNEO UTILIZANDO MÉTODO DE ELEMENTOS FINITOS

Victor Guido Pinheiro

Março/2022

Orientador: Gustavo Rabello dos Anjos, Ph.D.

Departamento: Engenharia Mecânica

O objetivo desse trabalho é o desenvolvimento de uma simulação computacional para a análise da transferência de calor por condução no interior de componentes eletrônicos compostos de materiais heterogêneos. Para isso, foi implementado o Método de Elementos Finitos na linguagem de programação *Python*, com o intuito de resolver as equações de condução de calor em duas dimensões, uma vez que a espessura da CPU pode ser considerada desprezível. Para modelar as correntes elétricas encontrando resistência dentro dos componentes eletrônicos, foi considerada a presença de geração interna de calor. Desse modo foi possível definir o calor gerado para o funcionamento ideal da CPU, assim como uma equação que relaciona temperatura máxima registrada e calor interno gerado.

**Palavras-Chave**: Condução de Calor, Material Heterogêneo, Método de Elementos Finito.

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Mechanical Engineer

ANALYSIS OF HEAT CONDUCTION IN MICROCHIPS OF HETEROGENIAL MATERIALS WITH FINITE ELEMENT METHOD

Victor Guido Pinheiro

March/2022

Advisor: Gustavo Rabello dos Anjos, Ph.D.

Department: Mechanical Engineering

The objective of this work is to develop a computational simulation to analyze the phenomenon of heat transfer by conduction in the interior of a microchip composed by heterogenous material. To that end, the Finite Element Method was implemented in the programming language of *Python*, aiming to solve the heat conduction equations in two dimensions once the microchip thickness can be disregarded. To take in consideration the electric currents finding resistance inside the microchip, it was considered that was internal heat generation. Therefore, it was possible to determine the internal heat generation for the ideal operation of the microchip, as well as an equation that connects the maximum registered temperature with the internal heat generation.
**Key-words**: Heat Conduction, Heterogeneous Material, Finite Element Method.

# Sumário

# Lista de Figuras

# Capítulo 1

# Introdução

”Microchips”de computador, também conhecidos como Unidades de Processamento Central (CPU), são pequenos componentes eletrônicos retangulares de materiais semicondutores cristalinos (silício, na maioria das vezes) cobertos de pequenos transistores e outros eletrônicos. Sua utilidade é de transformar eletricidade em dados binários, para, assim, alcançar um alto desempenho em cálculos e processamento de informações, tendo se tornado a base de todos os eletrônicos desde a década de 70. De acordo com dados da ASML, mais de 634 bilhões de CPUs foram fabricados no ano de 2019.

Esses componentes estão se tornando exponencialmente mais complexos com o passar dos anos. Como observado por Gordon E. Moore, (mais tarde batizada de “Lei de Moore”, embora não seja uma lei da física, e sim uma relação empírica observada), o número de transistores de uma CPU dobra a cada 18 meses pelo mesmo custo.

Com esse aumento da complexidade do número de transistores e da corrente elétrica que passa por esses componentes, a questão de superaquecimento das CPUs também se torna mais relevante. Portanto, aumenta-se também a importância da análise da transferência de calor no projeto de componentes eletrônicos. No entanto, essa análise não é trivial, uma vez que frequentemente as equações de transferência de calor não apresentam soluções analíticas simples, de modo que é necessário recorrer a soluções numéricas aproximadas.

Desta forma, o objetivo desse trabalho é desenvolver um código da linguagem *Python* de programação que utilize o Método de Elementos Finitos para

resolver as equações de transferência de calor por condução, e assim permitindo analisar de modo mais eficiente a distribuição de temperatura na CPU durante o seu funcionamento, levando em consideração os diversos materiais utilizados em sua confecção, e a geração interna de calor causada pelas correntes elétricas nos transistores.

## 1.1 Hipóteses, Considerações e Organização do Trabalho

Para o desenvolvimento deste trabalho, utilizou-se a equação condução de calor em coordenadas cartesianas, considerando apenas duas dimensões. Isso se deve ao fato de que a espessura da CPU foi considerada desprezível quando comparada às outras duas dimensões. Também foi considerado que os materiais que compõem a CPU são isotrópicos. Para a derivada temporal, utilizou-se método de diferenças finitas de Crank-Nicolson para a discretização.

Para a solução numérica, utilizou-se o Método dos Elementos Finitos, e para a discretização espacial utilizou-se o Método de Galerkin, sobre uma malha com elementos triangulares produzida no software *Gmsh*. Uma vez que o estudo é feito em uma CPU constituída de diferentes materiais, as difusividades térmicas foram atribuídas aos elementos da malha de acordo com a sua posição. Nas fronteiras entre dois ou mais materiais, foi adotado o valor referente à média aritmética das difusividades térmicas de cada material.

Para resolver os sistemas lineares apresentados no Método de Elementos Finitos, foi utilizada a biblioteca Numpy disponível em *Python*. Para visualizar os resultados, os gráficos foram produzidos por outra biblioteca, *Matplotlib*, assim como no software *Microsoft Excel*.

A validação da simulação foi realizada através da comparação de exemplos encontrados na bibliografia e em outros trabalhos e monografias. A simulação realizada considerando como base o modelo de CPU apresentado no estudo de TU [1]. Embora esse estudo considere um modelo em três dimensões, o escopo foi mantido apenas na distribuição de temperatura em duas dimensões do componente eletrônico. Portanto, na Figura 1.1 apresentamos a imagem de uma CPU

padrão, como foi mostrado no estudo.

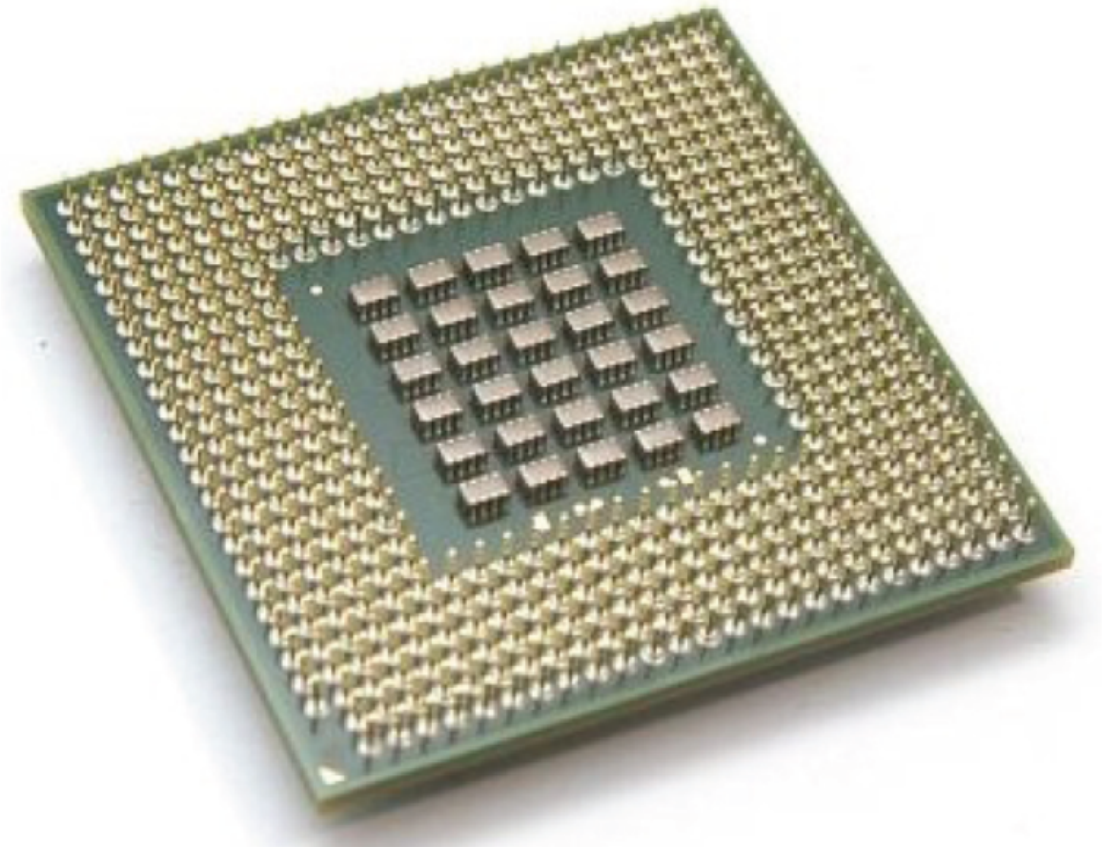


Figura 1.1: Ilustração do ”microchip”utilizado como referência [1]

Estre trabalho foi organizado em sete capítulos, como definido abaixo:

- Capítulo 1 - Introdução
- Capítulo 2 - Revisão Bibliográfica
- Capítulo 3 - Metodologia
- Capítulo 4 - Algoritmo
- Capítulo 5 - Validação
- Capítulo 6 - Resultados
- Capítulo 7 - Conclusão

No Capítulo 2, foram listadas as referências utilizadas para o desenvolvimento das equações de calor e do Método de Elementos Finitos, assim como

uma breve introdução desses temas. No Capítulo 3, apresentamos a metodologia utilizada para a equação de calor por condução no regime transiente em duas dimensões, assim como a aplicação do Método de Elementos Finitos para essa equação.

No Capítulo 4, o código utilizado em *Python* foi apresentado e explicado, para cada processo individual. Já no Capítulo 5, demonstramos a validação do código exibido por meio da comparação com resultados encontrados em outros trabalhos. Por fim, os Capítulos 6 e 7 apresentam, respectivamente, os resultados da simulação final e as considerações finais do trabalho.

# Capítulo 2

# Revisão Bibliográfica

## 2.1 Transferência de Calor

De acordo com ÖZISIK [2], podemos estudar o conceito de calor através da ótica de duas ciências. Primeiramente, a termodinâmica, que trata principalmente da relação entre o calor e outras formas de energia. Por outro lado, temos a ciência da transferência de calor, que analisa especialmente a taxa de transferência de calor em um sistema. Como definido por INCROPERA [3], a termodinâmica lida apenas com os estados iniciais e finais dos processos, e não providencia nenhuma informação sobre a natureza da interação ou a taxa de tempo com que ocorre o processo. Embora esse fluxo não possa ser medido diretamente, ele está relacionado à grandeza física chamada temperatura, que, por sua vez, pode ser diretamente mensurada.

O livro HAHN [4] descreve essa relação de forma que, quando houver diferenças de temperatura em um sistema, também haverá fluxo de calor, sempre fluindo da região de alta temperatura para a de baixa temperatura. Portanto, desde que o fluxo de calor suceda a presença de um gradiente de temperatura em um sistema, essa conexão se torna essencial para o estudo dos fenômenos de transferência de calor. Isso porque, ao conhecermos a distribuição da temperatura no sistema, utilizamos a lei que relaciona esse gradiente de temperatura com o fluxo de calor (quantidade de calor transferido por unidade de tempo por unidade de área).

A transferência de calor pode ocorrer de três modos diferentes: condução,

convecção e radiação. No entanto, essa divisão é bem definida apenas na teoria. Trabalhando mais perto da realidade, qualquer transferência de calor é formada por combinações dessas três modalidades, em proporções diferentes, dependendo das condições. Podemos, entretanto considerar um dos modos separadamente quando o calor transferido pelos outros for desprezível quando comparado.

A transferência de calor por meio de convecção ocorre quando há uma diferença de temperatura entre a superfície de um sólido e o fluido escoando sobre ele. É possível subsequentemente fragmentar essa classificação ao levar em consideração a origem do movimento do fluido. Quando o escoamento do fluido é influenciado apenas por meios naturais, definimos a convecção livre (ou natural). Por outro lado, quando o movimento é manipulado por meios externos (por um ventilador ou uma bomba), caracteriza-se a convecção forçada, como exemplificado na Figura 2.1.

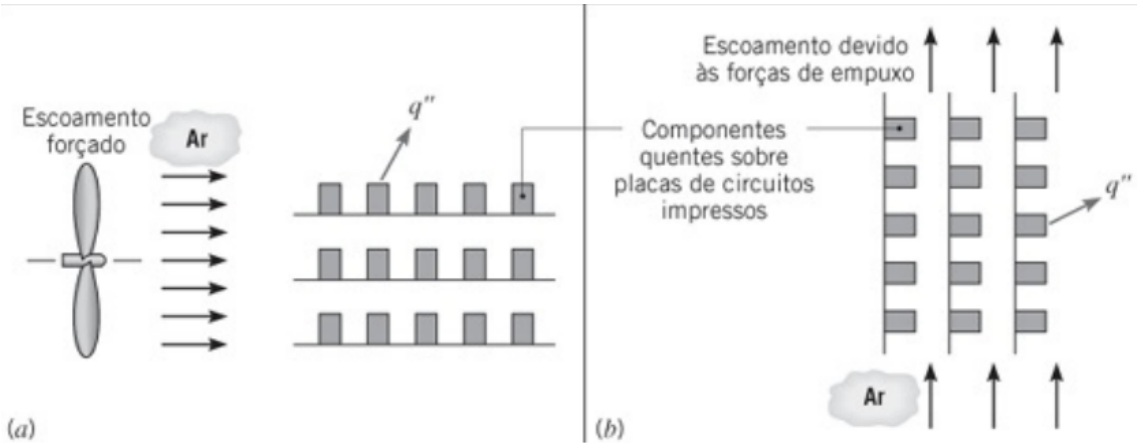



Figura 2.1: a) Exemplo de convecção forçada b) Exemplo de convecção livre [3]

Teoricamente, como a determinação da distribuição de temperaturas no fluido depende da movimentação do fluido, os cálculos de transferência de calor por convecção podem ser consideravelmente complexos, dependendo de numerosas variáveis não triviais. No entanto, como mencionado em ÖZISIK[2], para aplicações de engenharia podemos realizar um cálculo simplificado e conhecendo as temperaturas médias do fluído ($T_f$) e da superfície por onde ele escoa ($T_w$), podemos estimar o fluxo de calor como exibido na Equação 2.1.

$$q_x = h(T_f - Tw) \tag{2.1}$$

É importante notar que a Equação 2.1 descreve o fenômeno quando a temperatura do fluido é maior que a da superfície, e portanto há fluxo de calor do fluido para a dita superfície. Caso o inverso ocorra, com fluxo de calor da superfície para o fluido, será necessário inverter o sinal da Equação 2.1, como ilustrado na Equação 2.2.

$$q_x = h(T_w - Tf) \tag{2.2}$$

Definimos também nessas equações o termo $h$ como o coeficiente de transferência de calor (W/m²K). Essa constante depende de múltiplos fatores, como o tipo de fluxo do fluido (laminar ou turbulento), a geometria do corpo, a área de escoamento, a posição na superfície do corpo, a temperatura média e se a convecção estudada é forçada ou livre. Na maioria dos casos, assim como com as temperaturas, um valor médio pode ser considerado para a realização dos cálculos.

A radiação é o fenômeno de transferência de calor que ocorre quando um corpo emite energia de seu interior em forma de ondas eletromagnéticas através de sua superfície, em virtude de sua temperatura. Quando a radiação atinge a superfície de um outro corpo, ela aprofunda-se no meio, sendo absorvida (ou atenuada). Se uma grande parte dessa radiação for absorvida a um intervalo relativamente pequeno da superfície, podemos considerar que foi atenuada pela própria superfície.

Qualquer meio através do qual a radiação se propaga causa essa atenuação; por isso, a radiação se propaga sem perdas apenas através do vácuo. No entanto, para aplicações de engenharia, as perdas podem ser consideradas desprezíveis dependendo das condições. ÖZISIK[2] considera, por exemplo, que o ar atmosférico de uma sala, para finalidades práticas, é transparente à radiação térmica.

Utilizando a Lei de Stefan-Boltzmann, podemos calcular o fluxo máximo de transferência de calor emitido por um corpo por radiação como mostrado na Equação 2.3.

$$E_b = \sigma T^4 \tag{2.3}$$

Onde $T$ é a temperatura absoluta (em Kelvins), $\sigma$ é a constante de Stefan-Boltzmann (5.6697 x $10^{-8} W/m^2K^4$). $E_b$ é definido como emitância de corpo negro (W/m²), ou seja, um radiador ideal, o único corpo capaz de emitir radiação de acordo com a Equação 2.3. Já o cálculo do fluxo de transferência de calor em um corpo real teria é mostrado na Equação 2.4.

$$q = \epsilon E_b \tag{2.4}$$

Onde $\epsilon$ é a emissividade do corpo (variando entre 0 e 1), que define qual a proporção do fluxo máxima que o corpo é capaz de emitir.

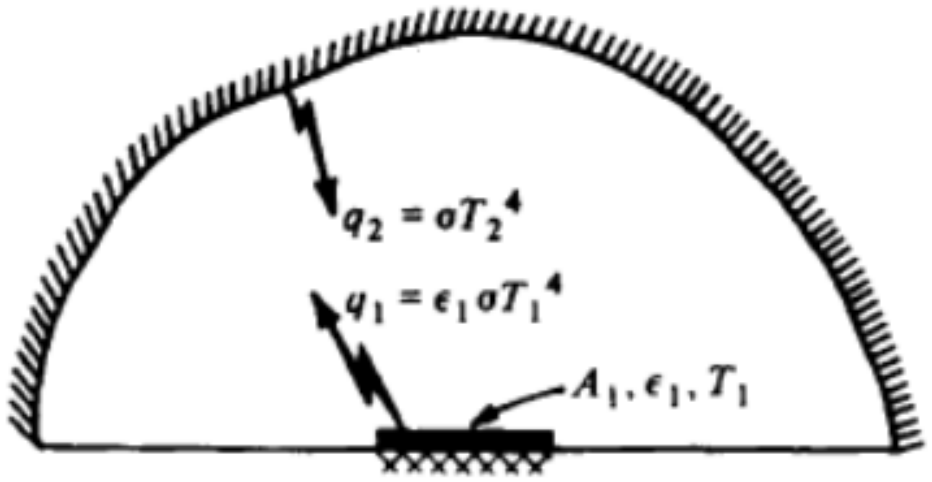



Figura 2.2: Exemplo de dois corpos emitindo radiação, adaptado de ÖZISIK [2]

Por último, temos a condução, o foco principal deste trabalho. Esse fenômeno é caracterizado quando calor é transferido pelo choque direto entre as moléculas (para fluidos) ou movimento de elétrons (para metais). Como a

presença de muitos elétrons se movendo em rede caracteriza alta condução de eletricidade, normalmente materiais que são bons condutores elétricos também são bons condutores térmicos. A imagem na Figura 2.3 ajuda a ilustrar melhor o fenômeno.

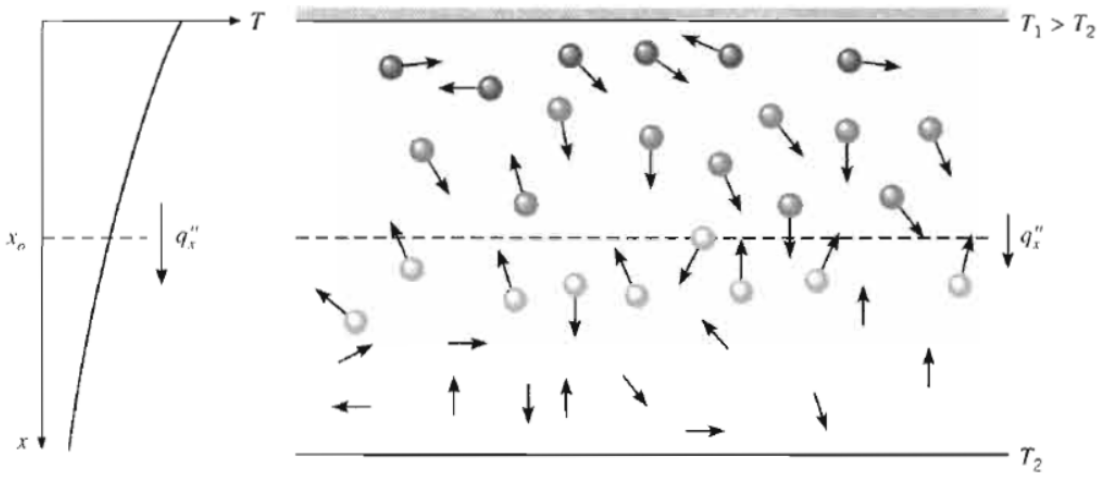



Figura 2.3: Condução de calor expressada por colisões entre moléculas [3]

Para calcular o fluxo de calor por condução, utilizamos a Lei de Fourier, como descrita na Equação 2.5.

$$q_x = -k\frac{dT}{dx} \tag{2.5}$$

Onde $q_x$ é o fluxo de calor no sentido positivo de x, e $k$ é a condutividade térmica do material. A direção x foi escolhida arbitrariamente, e a equação é análoga para as outras coordenadas. A equação também pode ser escrita de maneira mais geral como:

$$q_{condução} = -k\nabla T \tag{2.6}$$

## 2.2 Método de Elementos Finitos

De acordo com LOGAN [5], o Método de Elementos Finitos é uma ferramenta que nos permite determinar soluções numéricas aproximadas para diversos problemas da engenharia para os quais, por envolverem geometrias com-

plexas, distribuições de propriedades de materiais não uniformes ou outras complicações, não é possível obter soluções analíticas.

Como exposto em REDDY [6], ao realizar uma descrição física de um fenômeno ou processo criamos o que é chamado de "modelo matemático", que representa de maneira abstrata um fenômeno real e concreto. Esses modelos são desenvolvidos tomando como base suposições de acordo com conhecimentos científicos, axiomas ou leis que governem o processo estudado. Desse modo, antes do advento do computador eletrônico, esses modelos precisavam ser drasticamente simplificados para que suas soluções analíticas pudessem ser encontradas.

Soluções analíticas são aquelas obtidas por expressões matemáticas que resultam em valores que são atribuídos à variável desconhecida estudada em qualquer posição do corpo analisado, sendo assim válidas para um número infinito de posições no corpo. Normalmente essas soluções necessitam da resolução de equações integrais ou diferenciais (ordinárias ou parciais) que, por causa das complicações listadas anteriormente, podem não ser alcançáveis.

No entanto, nas últimas quatro décadas, a evolução dos computadores nos permitiu, combinando modelos matemáticos com métodos numéricos, resolver de modo satisfatório muitos problemas práticos de engenharia. O uso de um método numérico e um computador para avaliar o modelo matemático de um processo e assim estimar suas características e propriedades é chamado de "simulação numérica". A aplicação do Método de Elementos Finitos neste trabalho é uma dessas simulações.

LOGAN [5] define que o Método de Elementos Finitos consiste em dividir o sistema contínuo estudado em múltiplos elementos menores ("*elementos finitos*") equivalentes ao todo. Isso resulta em um conjunto de equações algébricas simultâneas, de muito mais simples resolução do que as equações diferenciais, e que podem ser organizadas de forma matricial para facilitar a resolução. REDDY [6] cita que uma das grandes vantagens do Método de Elementos Finitos sobre outros métodos numéricos (como o Método de Diferenças Finitas) é sua generalidade e poder em aplicações em problemas práticos encontrados no cotidiano da engenharia.

De acordo com ZIENKIEWICZ [7], o desenvolvimento de métodos que

utilizam discretizações teve seus primeiros trabalhos originados no fim do século XIX e início e meados do século XX. Por exemplo, o conceito do chamado "Método Direto de Rigidez"(a implementação mais comum e simples do Método de Elementos Finitos) foi introduzido por Claude Navier ainda no século XIX, e mais tarde retrabalhado e reapresentado em sua forma mais moderna por R. F. Clebsch e outros cientistas. FELLIPA [8] cita que grande parte do desenvolvimento desses métodos se originou na área de tecnologia aeroespacial. Durante a Segunda Guerra Mundial as pesquisas foram incentivadas na busca de avanços tecnológicos para aeronaves, mas restrições de publicações tornam difícil rastrear a evolução dos trabalhos.

Embora o termo "Elementos Finitos" tenha sido introduzido apenas em 1960 por R.W. Clough em CLOUGH [9], as raízes matemáticas do método se estendem por até décadas antes, como comentado nos parágrafos anteriores. O trabalho de ALLAIRE [10] cita que o método teve seu objetivo original em auxiliar estudos complexos de análises estruturais e mecânica dos sólidos, O início de seu desenvolvimento se deu na década de 40, por diversos nomes como R. Courant, A. Hrennikoff e I. Argyris (COURANT [11] e HRENNIKOFF [12]). Esse desenvolvimento foi um grande exemplo de cooperação interdisciplinar entre engenheiros, físicos e matemáticos aplicados, que gerou imenso progresso na área de simulações numéricas (concomitantemente com os avanços computacionais da segunda metade do século XX).

Essa colaboração de múltiplos profissionais que contribuíram entre si, trouxeram referências de trabalhos anteriores e construíram bases para que outros cientistas pudessem refinar gradativamente a teoria do que chegou a se tornar o Método de Elementos Finitos é muito bem representada no gráfico apresentado em ZIENKIEWICZ [7], reproduzido na Figura 2.4.

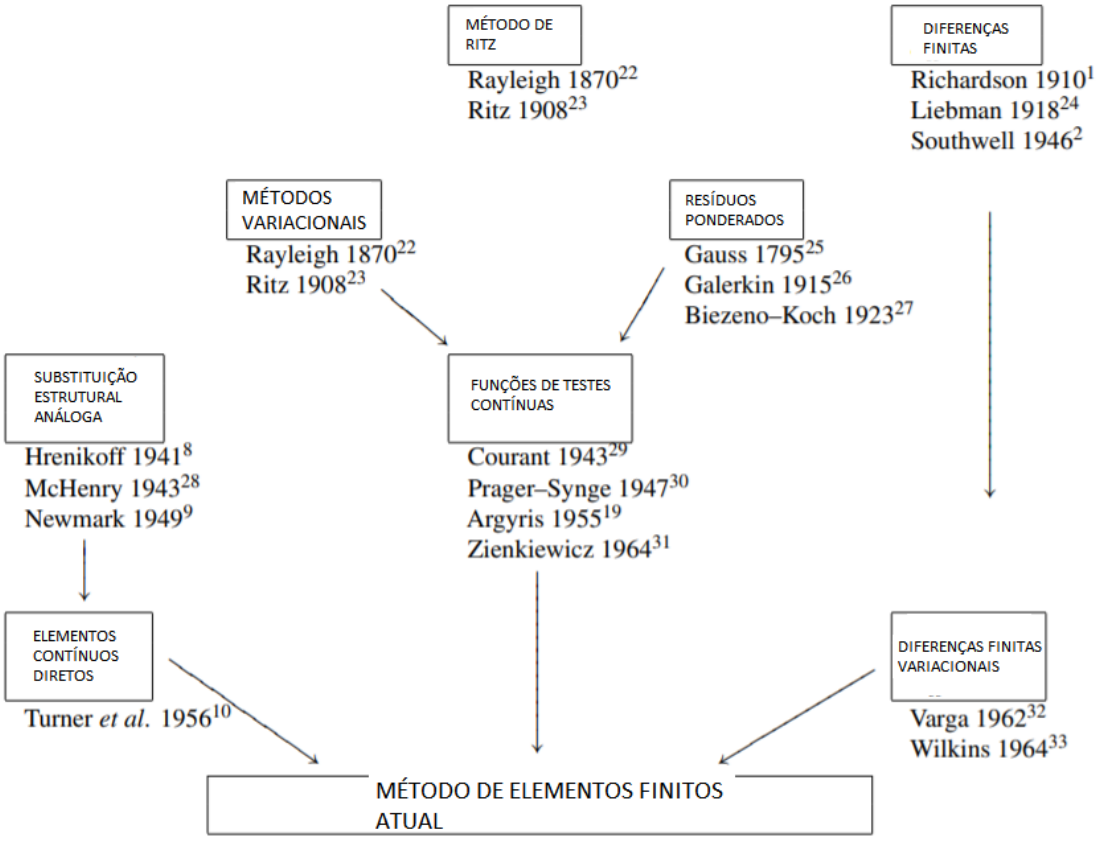



Figura 2.4: Evolução e contribuições do Método de Elementos Finitos, adaptado de ZIENKIEWICZ [7]

Posteriormente, como citado em AGUIAR [13] as contribuições de Boris Galerkin abriram novas possibilidades para a aplicação do Método de Elementos Finitos na Engenharia, com o desenvolvimento da Formulação (ou Método) de Galerkin. Esse método consistiu na utilização do Método de Resíduos Ponderados para determinar as constantes da formulação variacional, e as mesmas funções bases foram utilizadas nas funções peso.

A partir da década de 60, as aplicações do Método de Elementos Finitos se expandiram, não sendo mais limitadas apenas à área de análises estruturais. Suas utilizações se multiplicaram, abrangendo os campos da transferência de calor, dinâmica dos fluídos e até eletromagnetismo, proporcionando aproximações numéricas em casos em que as condições e geometrias complexas impediam a determinação das soluções analíticas.

De acordo com LEWIS [14], podemos aplicar o Método de Elementos Fini-

tos seguindo os seguintes passos do processo:

1. Discretização do contínuo
2. Escolha das funções de forma
3. Formular equações para os elementos
4. Reunir equações de elementos de forma a obter um sistema de equações simultâneas
5. Resolver o sistema de equações

Cada um dos passos será apresentado de forma mais detalhadas no capítulo a seguir, de forma a melhor descrever a aplicação do método. A imagem a seguir fornece uma visualização da sequência dos processos que compõe a aplicação das etapas do Método.

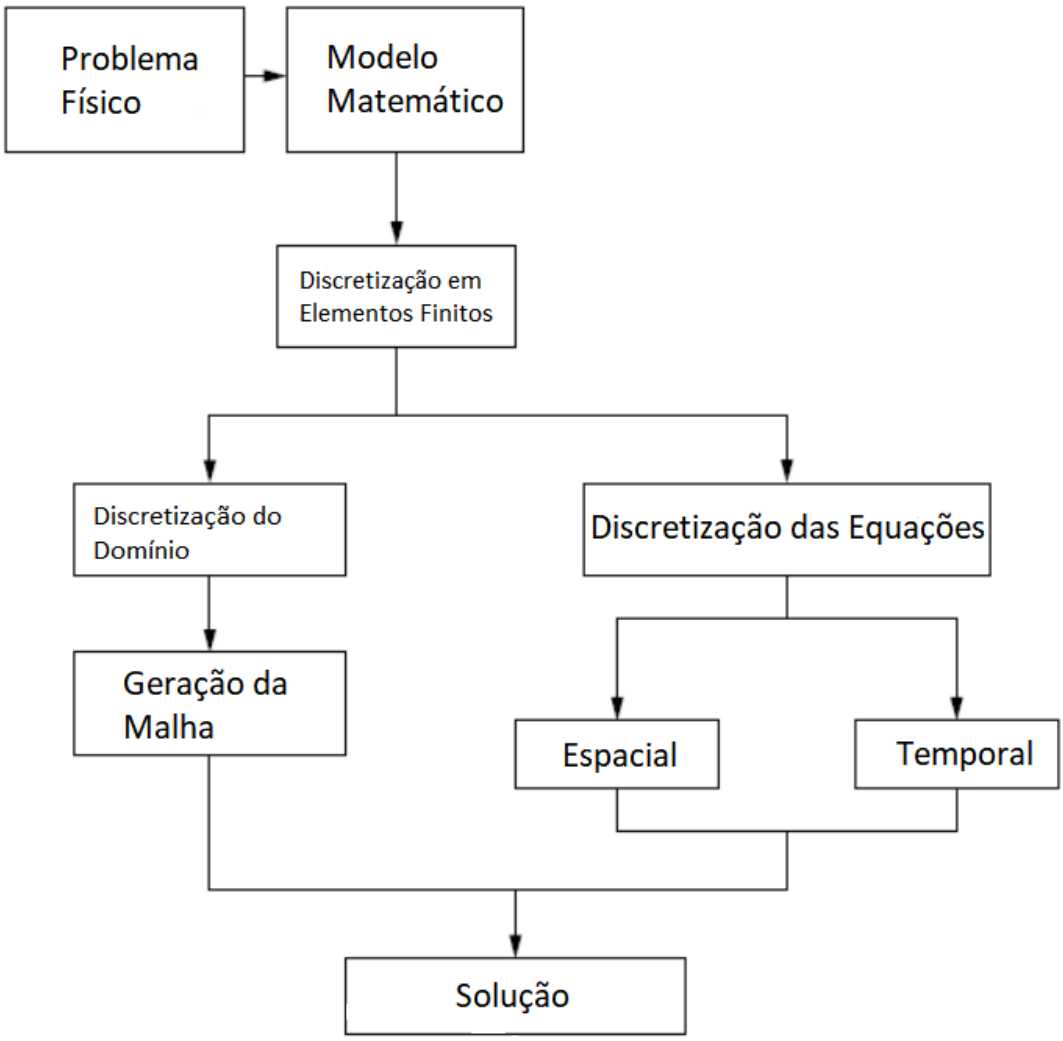



Figura 2.5: Esquematização do Método de Elementos Finitos [14]

# Capítulo 3

# Metodologia

Nessa seção será explicado, de forma mais detalhada, a aplicação do Método de Elementos Finitos (apresentado na seção anterior), à equação de condução de calor.

Como já descrito antes, consideramos a CPU com apenas duas dimensões, constituído de materiais isotrópicos (embora partes diferentes de componentes eletrônicos possam ser feitas de materiais diferentes). Também foi considerado apenas a condição de contorno de temperatura prescrita em todo contorno (condição de contorno de Dirichlet). Portanto, não será considerada transferência de calor por convecção, já que a ideia central deste trabalho é compreender a distribuição de temperatura no interior da CPU para as mesmas condições de contorno. Transferência de calor por radiação também foi considerada desprezível.

## 3.1 Modelo Teórico da Condução de Calor

De acordo com OZISIK [2], para determinarmos a equação bidimensional para a condução de calor, podemos considerar um elemento de volume de espessura $\Delta$x e altura $\Delta$y, com áreas $A_x$ e $A_y$ normais aos eixos coordenados x e y, respectivamente, como ilustrado na Figura 3.1.

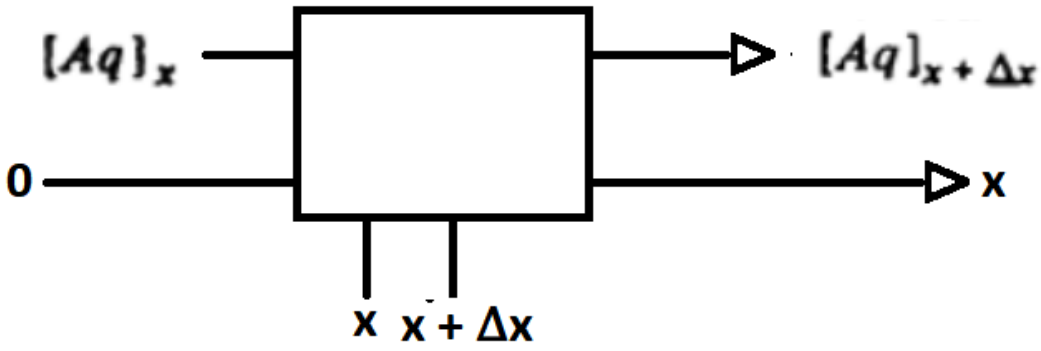



Figura 3.1: Esquema elemento infinitesimal x

O balanço de energia a ser usado como base, é mostrado na Equação 3.1.

(Taxa de Ganho Calor por Condução) + (Taxa de geração de energia) =
(Taxa de aumento de energia interna) (3.1)

Primeiramente, podemos escrever a taxa de calor por condução que flui através da área $A_x$ do elemento na posição x como exibido na Equação 3.2.

$$[Aq_x]_x \tag{3.2}$$

Onde $q$ é o fluxo de calor na posição x, no sentido positivo de x na superfície $A_x$ do elemento.

Da mesma forma, a mesma taxa de calor, mas na posição x +Δx é exibida na Equação 3.3.

$$[Aq_x]_{x+\Delta x} \tag{3.3}$$

Portanto, a taxa líquida de ganho de calor por condução (a primeiro termo da equação 3.1), pode ser escrita, na direção x, como mostrado na Equação 3.4.

$$[Aq_x]_x - [Aq_x]_{x+\Delta x} \tag{3.4}$$

Exibimos o raciocínio análogo para y na Equação 3.5.

$$[Aq_y]_y - [Aq_y]_{y+\Delta y} \tag{3.5}$$

Para o segundo termo, podemos definir que a geração de energia como demonstrado na Equação 3.6.

$$(A_x \Delta x + A_y \Delta y)g \tag{3.6}$$

Onde g é a geração interna de energia por unidade de volume.

Por último, para o terceiro termo, escrevemos que a taxa de aumento de energia interna, que é resultante da variação da temperatura com o tempo, é escrita como exibido na Equação 3.7.

$$(A_x \Delta x + A_y \Delta y)\rho c_p \frac{dT(x,y,t)}{dt} \tag{3.7}$$

Onde $c_p$ é o calor específico do material (J/Kg ºC) e $\rho$ é massa específica do material (kg/m³). Substituindo todas as definições na Equação 3.1, chegamos na Equação 3.8.

$$\frac{1}{A_x}\frac{[Aq_x]_x - [Aq_x]_{x+\Delta x}}{\Delta x} + \frac{1}{A_y}\frac{[Aq_y]_y - [Aq_y]_{y+\Delta y}}{\Delta y} + Q = \rho c_p \frac{dT(x,y,t)}{dt} \tag{3.8}$$

Considerando então que Δx e Δy tendem à zero, os primeiros dois termos tornam-se as derivadas de $[Aq]_x$ e $[Aq]_y$ em relação à x e y, respectivamente, como mostrado na Equação 3.9.

$$\frac{1}{A_x}\frac{\partial(A_x q_x)}{\partial x} + \frac{1}{A_y}\frac{\partial(A_y q_y)}{\partial y} + Q = \frac{\partial T(x,y,t)}{\partial t} \tag{3.9}$$

Substituindo os fluxos de calor $q_x$ e $q_y$, como definido na equação (2.5), e considerando que nas coordenadas cartesianas as áreas $A_x$ e $A_y$ são constantes, chegamos na Equação 3.10.

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right) + Q = \rho c_p \frac{\partial T(x,y,t)}{\partial t} \tag{3.10}$$

Considerando também que, para o escopo deste trabalho, a condutividade térmica $k$ é constante, escrevemos a Equação 3.11.

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{Q}{k} = \frac{\rho c_p}{k}\frac{\partial T(x,y,t)}{\partial t} \tag{3.11}$$

Podemos definir a grandeza $\alpha$ da difusividade térmica (m²/s) como escrito na Equação 3.12

$$\alpha = \frac{k}{c_p \rho} \tag{3.12}$$

Escrevemos finalmente a equação de transferência de calor por condução transiente em sólidos de duas dimensões como mostrado na Equação 3.13.

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{Q}{k} = \frac{1}{\alpha}\frac{\partial T(x,y,t)}{\partial t} \tag{3.13}$$

## 3.2 Modelo Teórico do Método de Elementos Finitos

### 3.2.1 Discretização do contínuo

Como definido em ANJOS [15], primeiramente devemos dividir o domínio contínuo em um número finito de subdomínios conhecidos como "elementos finitos", que são formados por nós (também referenciados como pontos nodais), que podem ser conectados de formas variadas, como triângulos ou quadriláteros (que não necessariamente precisam ser equiláteros ou regulares). O conjunto de todos esses nós forma um domínio discreto referenciado como malha. Essa é uma das grandes vantagens do Método de Elementos Finitos: a discretização por malhas com geometrias complexas exige o mesmo esforço que seria aplicado para geometrias mais simples e regulares.

### 3.2.2 Escolher funções de forma

Para definir as variações das variáveis a serem estudadas em cada elemento, utilizamos uma sequência combinatória constituída pelo produto de uma função por uma constante, ambas contida em um somatório finito, que é limitado justamente pelo número de elementos que compões o domínio discretizado definido no passo anterior. A função utilizada nessa sequência é chamada de função de forma, ou de interpolação.

Podemos então, para uma função *f(x)* qualquer, reescrevê-la de modo que:

$$f(x) = \sum_{i=1}^{np} f_i N_i(x) \tag{3.14}$$

Onde $f_i$ é a constante a ser determinada, $N_i$ é a função de forma escolhida e $np$ é o número de pontos nodais (ou nós).

A seleção da função de forma pode ser feita livremente, mas é necessário considerar os impactos que essa escolha trará para a aproximação numérica. Por exemplo, utilizar polinômios de ordem mais baixa, de acordo com ANJOS [15], consome menos custo computacional, e podem apresentar aproximações razoáveis dependendo das características do domínio. Já polinômios de graus

mais altos, embora em teoria proporcionassem aproximações melhores, aumentam consideravelmente o custo computacional, e podem ainda introduzir instabilidades numéricas na simulação. A Figura 3.2 exibe exemplos de polinômios de primeira e segunda ordem.

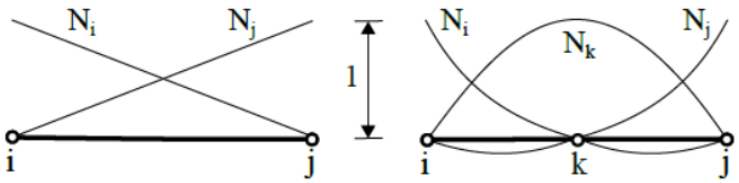



Figura 3.2: Funções de forma possíveis a serem utilizadas, com polinômio de primeira ordem à direita, e de segunda ordem à esquerda [15]

### 3.2.3 Formular equações para elementos

Para o próximo passo, utilizamos o Método do Resíduo ponderado, que consiste em ponderar a equação a ser resolvida através de uma função peso w, e integrar a equação ponderar no domínio do problema a ser resolvido ($\Omega$). Como definido em REDDY [6], após esse processo, encontramos o que é definido como a forma fraca do problema. O problema diferencial original, sem ponderação e integração, é definido, portanto, como forma forte. Considerando a Equação 3.15 (em sua forma forte).

$$g''(x) = 0 \tag{3.15}$$

Podemos escrever sua forma fraca como exibido na Equação 3.16.

$$\int_{\Omega} w(x)g''(x) = 0 \tag{3.16}$$

É válido ressaltar que a função peso é nula ($w = 0$) em quaisquer pontos onde as condições de contorno definidas para a função g sejam do tipo Dirichlet. Essa condição de contorno é definida, como mostrado por STRANG [16], quando os valores da solução no contorno são especificados.

Após esse processo, aplica-se o processo de integração por partes no termo com a derivada de ordem mais alta de modo a reduzir esse termo, e, consequentemente, possibilitando a escolha de funções de forma de ordens mais baixa, de modo a diminuir o custo computacional. Isso pode ser obtido com a aplicação do Teorema de Green, como demonstrado em CHASKALOVIC [17]. Para o exemplo utilizado acima, podemos definir as aproximações das Equações 3.17 e 3.18 com as funções de forma, de modo que sejam aplicadas para cada elemento discretizado.

$$g(x) = \sum_{i=1}^{np} g_i N_i(x) \tag{3.17}$$

$$w(x) = \sum_{j=1}^{np} w_j N_j(x) \tag{3.18}$$

Por último, ao utilizarmos o Método de Galerkin, definimos que ambas as funções de forma escolhidas para $g$ e $w$ são iguais, como mostrado na Equação 3.19.

$$N_i(x) = N_j(x) \tag{3.19}$$

### 3.2.4 Reunir equações de elementos de forma a obter um sistema de equações simultâneas

Substituindo então as equações (2.6) e (2.7) na equação (2.5), e arranjando de modo organizado os somatórios dentro das integrais, podemos definir a equação como um sistema linear, que por sua vez pode ser escrito em forma matricial, de modo que sua solução computacional se torna relativamente simples. BATHE [18] define uma das maiores vantagens do Método de Elementos Finitos

como sendo a resolução de um conjunto de múltiplas equações simples para encontrar a solução de um problema complexo, umas vez que a o comportamento dos elementos reflete o comportamento geral do sistema. Nesse passo, também realizamos a imposição das condições de contorno definidas no problema.

### 3.2.5 Resolver o sistema de equações

Podemos agora então utilizar qualquer um dos muitos métodos matemáticos para a resolução de sistemas lineares para encontrarmos os valores das variáveis de campo de cada um dos elementos, como no exemplo deste trabalho, a temperatura.

## 3.3 Modelo Numérico

A equação (3.13) descrita na seção anterior, é a forma forte da equação a ser resolvida, como definimos no Capítulo 2. Como definido no início deste capítulo, consideramos as condições de contorno para essa equação diferencial como sendo de temperatura constante em o todo o contorno da CPU. Isso caracteriza a condição de contorno de Dirichlet, o que facilitará o desenvolvimento da equação mais à frente, enquanto estivermos aplicando o Método de Elementos Finitos.

Seguindo então o método descrito no Capítulo anterior, primeiramente iremos discretizar o domínio contínuo. Como a geometria do problema é relativamente simples, utilizamos a forma geométrica de menor complexidade para aproximar a superfície: o triângulo, que, de acordo com LEWIS [14], é uma das formas de elementos mais utilizadas em aproximações de elementos finitos.

Desse modo, as funções de forma são definidas do modo representado adiante, nas Equações 3.20 à 3.22.

$$N_i = \frac{1}{2A}(a_i + b_i x + c_i y) \tag{3.20}$$

$$N_j = \frac{1}{2A}(a_j + b_j x + c_j y) \tag{3.21}$$

$$N_k = \frac{1}{2A}(a_k + b_k x + c_k y) \tag{3.22}$$

Definimos A como a área do elemento, calculada como mostrado na Equação 3.23.

$$A = \frac{1}{2}\det\begin{bmatrix} 1 & x_i & y_i \\ 1 & x_j & y_j \\ 1 & x_k & y_k \end{bmatrix} \tag{3.23}$$

Em seguida as constantes $a_i$, $a_j$, $a_k$, $b_i$, $b_j$, $b_k$, $c_i$, $c_j$ e $c_k$ são definidas nas Equações de 3.24 à 3.33.

$$a_i = x_j y_k - x_k y_j \tag{3.24}$$

$$a_j = x_k y_i - x_i y_k \tag{3.25}$$

$$a_k = x_i y_j - x_j y_i \tag{3.26}$$

$$b_i = y_j - y_k \tag{3.27}$$

$$b_j = y_k - y_i \tag{3.28}$$

$$b_k = y_i - y_j \tag{3.29}$$

$$c_i = x_k - x_j \tag{3.30}$$

$$c_j = x_i - x_k \tag{3.31}$$

$$c_k = x_j - x_i \tag{3.32}$$

$$\tag{3.33}$$

Como próximo passo, devemos agora escrever a forma fraca da equação 3.13. Definimos então a função peso *w(x,y)*, e integramos o produto dela pela forma forte da equação 3.13 no domínio Ω, como realizado na Equação 3.34.

$$\int_\Omega w\left[\frac{1}{\alpha}\frac{\partial T(x,y,t)}{\partial t} - \nabla^2 T - \frac{Q}{k}\right] d\Omega = 0 \tag{3.34}$$

Definimos o domínio $\Omega$ como exibido na Equação 3.35.

$$T : \Omega \rightarrow \mathrm{R}, 0 < x < x_t \cup 0 < y < y_t \tag{3.35}$$

Onde $x_t$ e $y_t$ são, respectivamente, o comprimento e altura da CPU.

Podemos então abrir a integral, e aplicar a identidade de Green no termo derivada de ordem mais alta, de modo a obter duas outras integrais de ordens reduzidas como mostrado nas Equações 3.36 e 3.37.

$$\int_{\Omega} w \frac{\partial T(x,y,t)}{\partial t} d\Omega - \int_{\Omega} w \alpha \nabla^2 T \, d\Omega - \int_{\Omega} w \frac{Q}{\rho c_p} d\Omega = 0 \tag{3.36}$$

$$\int_{\Omega} w \frac{\partial T(x,y,t)}{\partial t} d\Omega - \int_{\Gamma} w \alpha \, d\Gamma + \int_{\Omega} \nabla w \alpha (\nabla T) d\Omega - \int_{\Omega} w \frac{Q}{\rho c_p} d\Omega = 0 \tag{3.37}$$

Onde $\Gamma$ define todo o contorno do domínio $\Omega$.

Agora, definimos então as aproximações das funções *T(x,y,t)*, *w(x,y)* e *Q(x,y)* na forma dos somatórios discutidos no capítulo anterior, para realizar a discretização da parte espacial. A utilizando o Método de Galerkin é mostrado nas Equações 3.38, 3.39 e 3.40.

$$T(x,y) \simeq \sum_{i=0}^{n} N_i(x,y) a_i \tag{3.38}$$

$$w(x,y) \simeq \sum_{j=0}^{n} N_j(x,y) b_i \tag{3.39}$$

$$Q(x,y) \simeq \sum_{i=0}^{n} N_i(x,y) Q_i \tag{3.40}$$

O método de Galerkin adotado se caracteriza principalmente por igualar as funções de forma $N_i$ e $N_j$, como mostrado na Equação 3.41.

$$N_i(x,y) = N_j(x,y) \tag{3.41}$$

Podemos então, finalmente realizar as substituições das aproximações definidas na equação 3.21, de modo a obter a Equação 3.42.

$$\sum_{i=0}^{n}\sum_{j=0}^{n}\int_{\Omega} N_i N_j \frac{da_i}{dt} d\Omega + \sum_{i=0}^{n}\left[\sum_{j=0}^{n} - \int_{\Gamma} w(\alpha \nabla N_i) d\Gamma + \int_{\Omega} (\alpha \nabla N_i)\nabla N_j d\Omega\right] a_i = \int_{\Omega} N_i N_j \frac{Q}{\rho c_p} d\Omega \tag{3.42}$$

Considerando que o problema analisado é de natureza apenas térmica e que, como enunciado anteriormente, o mesmo apresenta apenas condições de contorno do tipo Dirichlet, a integral referente ao contorno ($\Gamma$) tem valor nulo, uma vez que a função peso tem valor igual à zero quando consideramos condições de contorno do tipo Dirichlet. Temos então que a equação 3.42 se torna a Equação 3.43.

$$\sum_{i=0}^{n}\sum_{j=0}^{n}\int_{\Omega} N_i N_j \frac{da_i}{dt} d\Omega + \sum_{i=0}^{n}\left[\sum_{j=0}^{n}\int_{\Omega} (\alpha \nabla N_i)\nabla N_j d\Omega\right] a_i = \int_{\Omega} N_i N_j \frac{Q}{\rho c_p} d\Omega \tag{3.43}$$

Podemos agora seguir para o próximo passo, que consiste em definir a equação 3.27 como um sistema linear, e escrevê-lo em sua forma matricial, de acordo com LEWIS [[14]], como mostrado na Equação 3.44.

$$\mathbf{M}_{ij}\frac{da_i}{dt} + \mathbf{K}_{ij}a_i = \mathbf{M}_{ij}Q_i + \mathbf{C} \tag{3.44}$$

Onde:

1. $\mathbf{M}_{ij}$ é definida como a matriz de massa

2. $\mathbf{K}_{ij}$ é definida como a matriz de rigidez

3. $\mathbf{C}$ é definido como o vetor que contém as condições de contorno

É importante notar que o termo referente à derivada temporal permanece igual. A discretização temporal será abordada em breve. Primeiro, serão definidos as matrizes de massa e de rigidez.

A matriz de massa $\mathbf{M}$ é o resultado do produto das funções de forma $N_i$ e $N_j$ dentro da integral no domínio $\Omega$. Desse modo podemos escrever a matriz de massa, para cada elemento, como mostrado na Equação 3.45.

$$\mathbf{M}^e = \frac{A}{12}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{3.45}$$

Já a matriz de rigidez $\mathbf{K}$ é encontrada com o produto dos gradientes das funções de forma ($\nabla N_i$ e $\nabla N_j$), uma vez que o coeficiente de condutividade térmica ($k$) é constante, e pode ser retirado da integral. Consideramos também a hipótese de isotropia para esse coeficiente; a condutividade térmica não é uma função da posição (x e y). No entanto, levando em consideração que o objetivo deste trabalho é o estudo de materiais heterogêneos, existem intervalos bem definidos das dimensões x e y com condutividades térmicas diferentes. Essa organização é obtida multiplicando a matriz de rigidez de cada elemento pela coeficiente relativo ao material em que esse elemento se encontra.

Em conclusão, podemos então definir a matriz de rigidez, para cada elemento individual como mostrado na Equação 3.46.

$$\mathbf{K}^e = \frac{1}{4A}\begin{bmatrix} b_i^2 + c_i^2 & b_i b_j + c_i c_j & b_i b_k + c_i c_k \\ b_j b_i + c_j c_i & b_j^2 + c_j^2 & b_j b_k + c_j c_k \\ b_k b_i + c_k c_i & b_k b_j + c_k c_j & b_k^2 + c_k^2 \end{bmatrix} \tag{3.46}$$

Por último, a matriz **C** contém as condições de contorno do problema estudado. Como já definido anteriormente, todas as condições são do tipo Dirichlet, caracterizando temperatura constante em todo o contorno do domínio estudado.

Consideraremos agora a derivada temporal. Embora fosse possível aplicar também o Método de Elementos Finitos, foi decidido aplicar o Método de Diferenças finitas para a parte temporal, no intuito de simplificar o desenvolvimento. Esse método consiste em aproximar a derivada por uma subtração e uma divisão, discretizando assim o processo. Encontramos então a Equação 3.47.

$$\frac{da_i}{dt} \simeq \frac{a_i^{n+1} - a_i^n}{\Delta t} \tag{3.47}$$

No entanto, podemos observar que a matriz de rigidez (**K**), proveniente do termo do laplaciano, também multiplica a constante $a_i$.Portanto, é necessário que esse fator também seja discretizado. Temos três opções para realizar essa discretização. Primeiramente, temos o métodos explícito (termo do laplaciano é discretizado como $a_i^n$) e o método implícito (termo do laplaciano é discretizado como $a_i^{n+1}$). O método explícito requer menos esforço computacional para ser utilizado, no entanto frequentemente o passo de tempo $\Delta t$ requerido para evitar instabilidade numérica é tão pequeno que torna o processo inviável. Já o método implícito permite passo de tempo mais flexível e prático, mas consome mais esforço computacional do que o explícito (uma vez que é necessário a realização de um cálculo adicional em cada iteração), mas é comparativamente mais estável. Teríamos, portanto, a equação 3.44 reescrita para o caso explícito na Equação 3.48.

$$\left(\frac{\mathbf{M}_{ij}}{\Delta t} + \mathbf{K}_{ij}\right) a_i^{n+1} = \frac{\mathbf{M}_{ij}}{\Delta t} a_i^n + \mathbf{M}_{ij} Q_i + \mathbf{C} \tag{3.48}$$

E para o caso implícito na Equação 3.49.

$$\frac{\mathbf{M}_{ij}}{\Delta t} a_i^{n+1} = \left( \frac{\mathbf{M}_{ij}}{\Delta t} - \mathbf{K}_{ij} \right) a_i^n + \mathbf{M}_{ij} Q_i + \mathbf{C} \tag{3.49}$$

Por último, temos uma alternativa viável para os dois métodos mencionados, chamado de Método de Crank–Nicolson, que consiste em uma média aritmética dos métodos explícitos e implícitos. Como esse método é formado por uma combinação dos outros dois, é um método de segunda ordem, apresenta estabilidade considerável, e um custo computacional razoável e prático. Esse foi o método implementado neste trabalho. Finalmente, podemos reescrever a equação 3.44 e chegar ao final do processo, como exibido na Equação 3.50.

$$\left( \frac{\mathbf{M}_{ij}}{\Delta t} + \frac{1}{2}\mathbf{K}_{ij} \right) a_i^{n+1} = \left( \frac{\mathbf{M}_{ij}}{\Delta t} - \frac{1}{2}\mathbf{K}_{ij} \right) a_i^n + \mathbf{M}_{ij} Q_i + \mathbf{C} \tag{3.50}$$

Sobre a escolha da função de forma (ou função de interpolação), foi definido um polinômio de primeiro grau, de acordo com o estabelecido no Capítulo anterior, que funções de ordens mais baixas produzem resultados razoáveis com baixo custo computacional. Um exemplo de um elemento triangular é exibido na Figura 3.3.

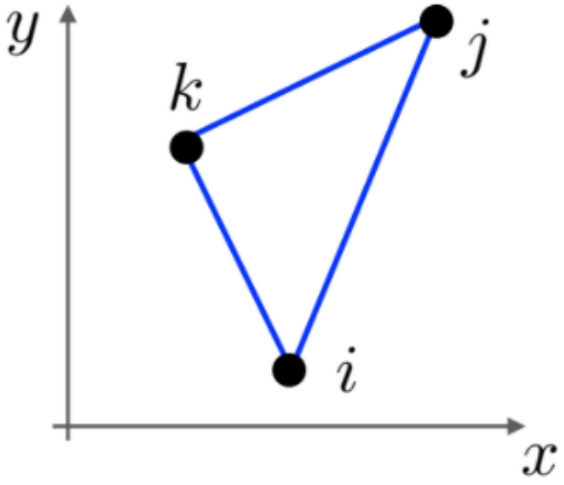



Figura 3.3: Elemento Triangular Linear [15]

# Capítulo 4

# Algoritmo

Neste capítulo, será abordado o algoritmo computacional que executa o método numérico descrito nos capítulos anteriores. O código foi implementado na linguagem *Python* de programação (versão 2.7), sendo uma linguagem de fácil acesso e comumente utilizada no meio acadêmico. Primeiramente, será apresentado o processo da geração da malha utilizada para o Método de Elementos Finitos, a partir do arquivo gerado no software *Gmsh*. Em seguida, abordaremos a montagem das matrizes de rigidez **K** e de massa M, assim como a implementação das condições de contorno de Dirichlet. Por último, apresentaremos também a solução do sistema linear de equações de condução de calor.

## 4.1 Geração da Malha no *Gmsh*

Como já definido anteriormente, a malha é o resultado da discretização do domínio contínuo estudado, onde elementos de forma triangular são formados por três nós (ou pontos nodais). As malhas utilizadas para este trabalho foram geradas pelo software *Gmsh* (versão 3.06), como mostrado nas Figuras 4.1 e 4.2, e exportadas como arquivos do tipo ".msh", como mostrado na Figura 4.3.

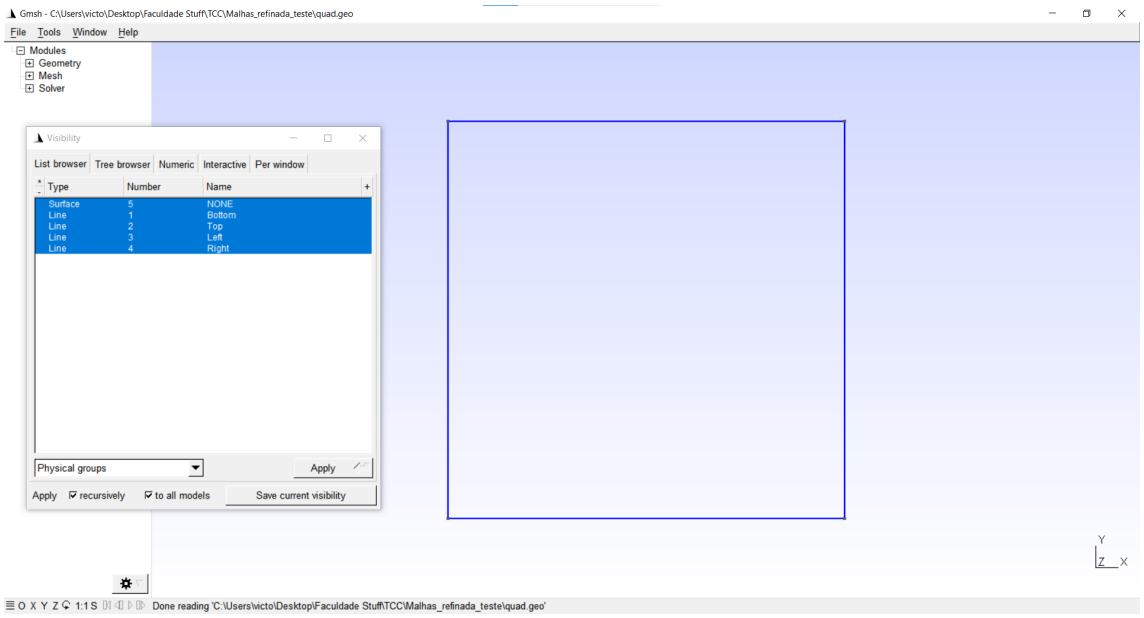



Figura 4.1: Modelagem de uma CPU quadrada simples no *Gmsh*, com os ”Physical Groups”definidos

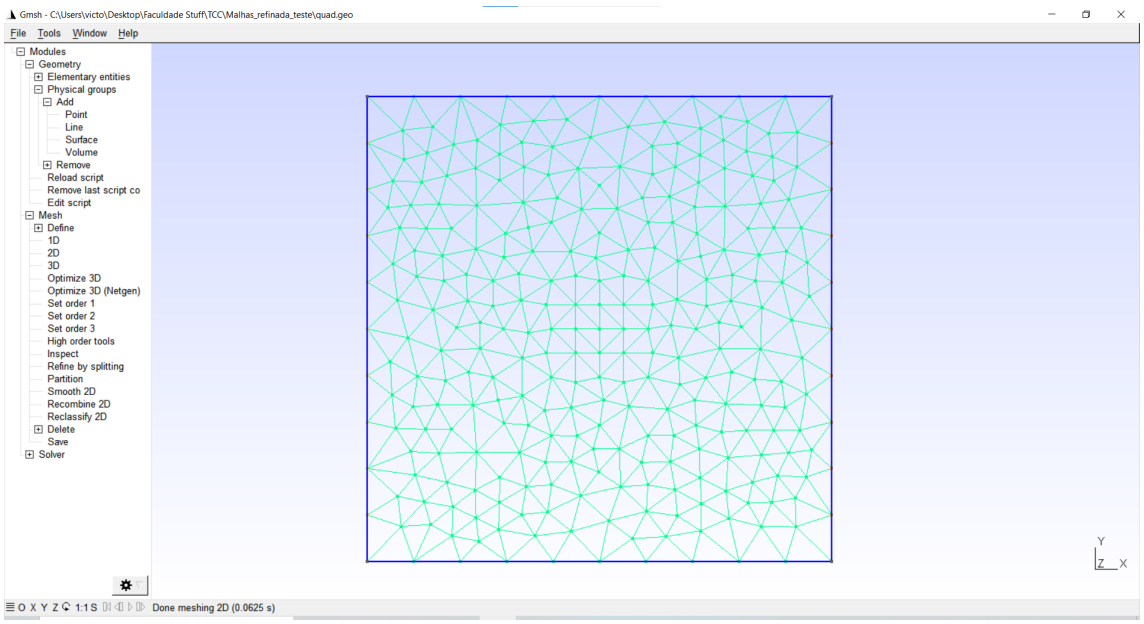



Figura 4.2: Malha gerada sobre o modelo

Esse tipo de arquivo pode ser lido pelo *Python* como um arquivo de texto normal (”.txt”). Esse arquivo gerado é dividido em três partes: primeiramente, com as definições dos ”Physical Names”(”nomes físicos”, em tradução livre). Nada mais são do que um método de se definir com mais facilidade as condições de contorno. Utilizamos as nomenclaturas ”Bottom”, ”Top”, ”Left”e ”Right”para

indicar todos os lados do retângulo que serve de modelo para a CPU, e que foram seu contorno. Todos os outros pontos são definidos como ”None”, indicando que não fazem parte do contorno.

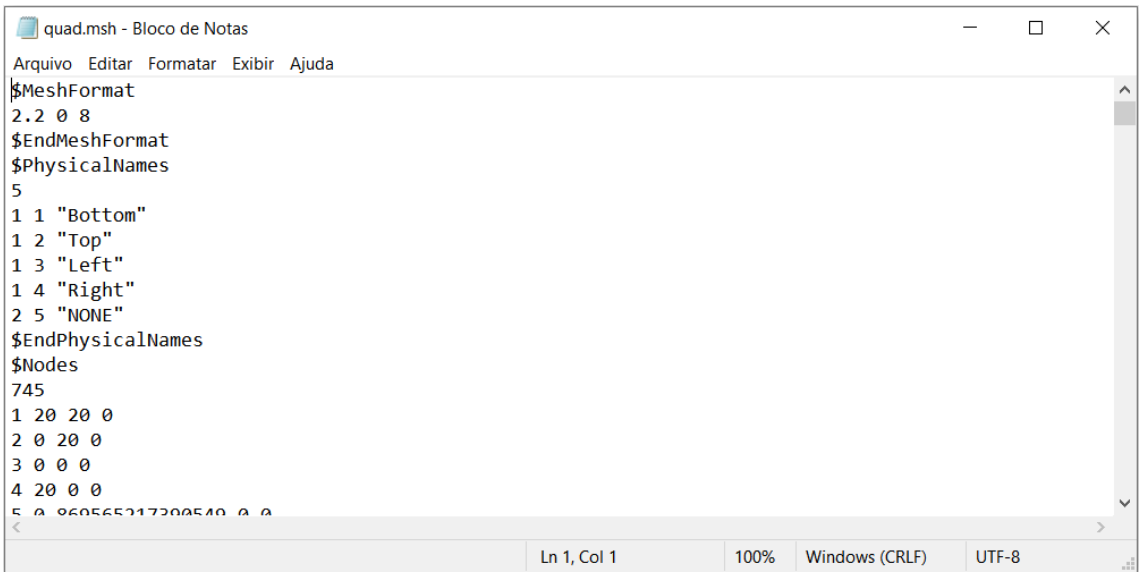



Figura 4.3: Exemplo do layout do arquivo de texto

As outras duas partes do arquivo são as listagens dos nós com suas coordenadas $x$, $y$ e $z$ (notando que, como estamos analisando apenas duas dimensões, o valor da coordenada $z$ é sempre nulo). Por último temos as listagens dos elementos triangulares, com o detalhamento de a partir de quais nós o elemento é formado.

## 4.2 Leitura do Arquivo e Geração da Malha no *Python*

Para a explicação mais detalhada do código implementado em *Pyhton*, referenciaremos o algoritmo esposto no Apêndice A deste trabalho. Primeiramente, a Parte A.1 do código mostra a a função que realiza a leitura do arquivo gerado pelo *Gmsh* e cria as listas onde as informações serão inseridas. Também foi definida a lista ”Chaves”, de modo que o código possa identificar as informações nas seções corretas dos arquivos. Criamos também um dicionário para identificar as regiões de contorno (neste caso, os quatro lados do retângulo).

Na segunda metade da função, o código irá percorrer todo o texto proveniente do arquivo ".msh"e, de acordo com as posições das informações com referência à chave, preencherá as matrizes e o dicionário criados anteriormente. No fina, retornará as listas com os pontos, e a informação do IEN geral e das condições de contorno.

Para auxiliar na construção da lista com os nós, criamos a classe "Ponto-Nodal"(que é referenciada na função mencionada). Ela armazena o índice do nó e suas coordenadas em $x$ e $y$.

Por fim, foi implementada também a classe "Malha", que se utiliza da função definida anteriormente para efetivamente construir a estrutura da malha para a execução do Método de Elementos Finitos.

## 4.3 Montagem das Matrizes

Uma vez que a estrutura da malha está pronta, construiremos as matrizes de rigidez e de massa, como demonstrado na Parte A.2. Para facilitar a confecção destas, foram declaradas duas matrizes nulas, com o tamanho referente ao número de nós da malha, que posteriormente serão preenchidas à medida que os cálculos forem realizados.

Para o próximo passo sabemos que, como detalhado no capítulo anterior, as matrizes **K** e **M** necessitam da área A e de seus elementos $a_i$, $a_j$, $a_k$, $b_i$, $b_j$, $b_k$, $c_i$, $c_j$ *e* $c_k$ para cada um de seus elementos da malha. Realizaremos, portanto, o cálculo dessas variáveis dentro de um *loop*, de modo que as operações se repetem para cada elemento da malha.Em sequência, definidas as variáveis necessárias, iniciaremos o cálculo das matrizes individuais dos elementos (ainda dentro do *loop*).

Para a montagem da matriz de de rigidez **K**, utilizamos uma notação alternativa para escrevê-la, embora o resultado matemático seja o mesmo, como mostrado na Equação 4.1 e 4.2.

$$\mathbf{K}^e = A\mathbf{B}^T\mathbf{B} \tag{4.1}$$

$$\mathbf{B} = \frac{1}{2A}\begin{bmatrix} b_i & b_j & b_k \\ c_i & c_j & c_k \end{bmatrix} \tag{4.2}$$

Definimos também, para auxiliar nos cálculos, uma função (*mult-por-escalar*), que realiza a multiplicação de um escalar por uma matriz. Embora a biblioteca *Numpy* tenha uma função análoga, ela depende da utilização de um *array*. Uma vez que estamos definindo as matrizes como listas dentro de uma lista, foi considerado mais prático a criação de uma função específica para isso, uma vez que não foi um processo trabalhoso, e necessitou de poucas linhas. Deste modo, temos a certeza de que o escalar multiplica cada um dos elementos da matriz.

Finalizando, portanto, a montagem das matrizes, criamos outros dois *loops* dentro do primeiro, e realizamos a montagem elemento por elemento. Outro detalhe importante a se atentar é de que, uma vez que este trabalho tem como objetivo estudar superfícies heterogêneas, regiões diferentes da área da CPU apresentarão propriedades diferentes, incluindo a difusividade térmica ($\alpha$). Desse modo, não podemos multiplicar toda a matriz de rigidez **K** pela mesma constante. Criamos então a matriz "Alpha", definindo o valor da difusividade térmica para cada ponto, de acordo com sua posição na CPU. Como já observado antes, calculamos o valor das propriedades de elementos que apresentem pontos na fronteira entre dois materiais diferentes com uma média aritmética simples das propriedades de cada material. Também definimos a lista "Q", que contém os valores da geração de calor interna da CPU. O valor do calor gerado é divido por $\rho c_p$, como mostrado na equação 3.43. Para "Alpha"e "Q", uma vez que as propriedades dos materiais dependem da posição, é necessário verificar a posição dos pontos para atribuir o valor correto no momento da montagem.

Por último, temos a montagem da matriz que contém as condições de contorno (**C**). Como já havíamos identificado os nós que constituem o contorno, um simples *loop* percorrendo o número de nós e verificando a descrição de cada nó é suficiente, como mostrado no Apêndice. Novamente, como estamos estudando

apenas o caso com condições de contorno do tipo Dirichlet de temperatura constante, são essas temperaturas que são atribuídas à matriz.

## 4.4 Solução do Sistema Linear de Equações

Finalmente, na Parte A.3, para resolvermos o sistema linear de equações de condução de calor criado na última seção, é necessário definir algumas algumas variáveis. Primeiramente, definiremos o valor de passo de tempo adotado (”dt”), e, consequentemente, do número de iterações a ser utilizado (”nint”). Essas duas variáveis dependerão das características do problema e da densidade da malha gerada, levando em consideração o poder de processamento computacional disponível.

Em seguida, definimos os vetores $a_i^{n+1}$ e $a_i^n$, que serão provisoriamente chamados de *T2* e *T1*, respectivamente. O vetor *T2* será recalculado em cada iteração, mas será necessário informar as condições iniciais do problema no vetor *T1*, como mostrado no Ap endice.

Se inicialmente a temperatura no interior da CPU não for a mesma em toda a região, podemos criar outro *loop* no momento de atribuir as temperaturas inciais, criando condições de acordo com as posições de cada elemento.

Uma vez definidos os vetores e variáveis, podemos resolver o sistema. Criamos também as matrizes auxiliares **A** e **B**, para facilitar os cálculos.

De modo que termos uma equação matricial simples, como mostrado na Equação 4.3.

$$\mathbf{A}T2 = \mathbf{B} \tag{4.3}$$

Como mencionado anteriormente, caso haja presença de calor interno, criamos a matriz **Q** definindo um *loop* simples, atribuindo um valor de geração de calor para os pontos nodais que estejam nas coordenadas onde ocorre a geração de calor.

Finalizando o processo, criamos um último *loop* com o número de iterações definido, e utilizamos a função da biblioteca *Numpy* ”linalg.solve”para resolver a equação matricial 4.3.

Uma vez que para cada iteração atribuímos o valor de *T2* para *T1*, é necessário recalcular a matriz **B** em todas as iterações com exeção da primeira.

Para visualizar os resultados obtidos, utilizamos a biblioteca *MatPlotLib* disponível para *Python*, para gerar gráficos que representam a distribuição de temperatura no componente eletrônico em uma escala de cores. Também utilizamos a ferramenta *Microsoft Exfel* para gerar alguns gráficos a partir dos dados obtidos pelo código. Alguns desses gráficos serão exibidos no próximo capítulo.

# Capítulo 5

# Validação

Neste capítulo será abordada a validação e verificação do código numérico. Serão apresentados resultados de resoluções de problemas com a utilização do Método de Elementos Finitos, e sua comparação com as soluções disponíveis. Em dois dos casos, foi possível obter a solução analítica (o que concede um alto de grau de confiabilidade para o método utilizado), e no último caso, mais complexo, para qual não há solução analítica disponível, foi realizada uma comparação com outra solução numérica apresentada na bibliografia [19] (de modo que realizamos uma verificação, e não uma validação).

Mas, primeiramente, iremos analisar a convergência da malha utilizada para realizar os problemas.

## 5.1 Convergência de Malha

Como já mencionado anteriormente, no momento de gerar a malha que será utilizada para discretizar o domínio, é necessário definir o número de elementos que comporão essa malha. Um número maior de elementos resulta em uma malha mais refinada e uma solução melhor, no entanto demanda um custo de processamento menor. Além disso, existe um momento em que o aumento do número de elementos causa cada vez menos influência na acurácia da solução. Para definir esse momento, é necessário realizar um estudo de convergência da malha.

Para isso, utilizamos o problema que será apresentado na seção a seguir

(5.2). O problema foi resolvido para malhas com números de elementos gradativamente maiores, e foi realizado o cálculo do maior erro absoluto registrado para cada caso, com relação à solução analítica. Para melhor enxergar a relação entre as duas grandezas, o resultado é mostrado na Figura 5.1.

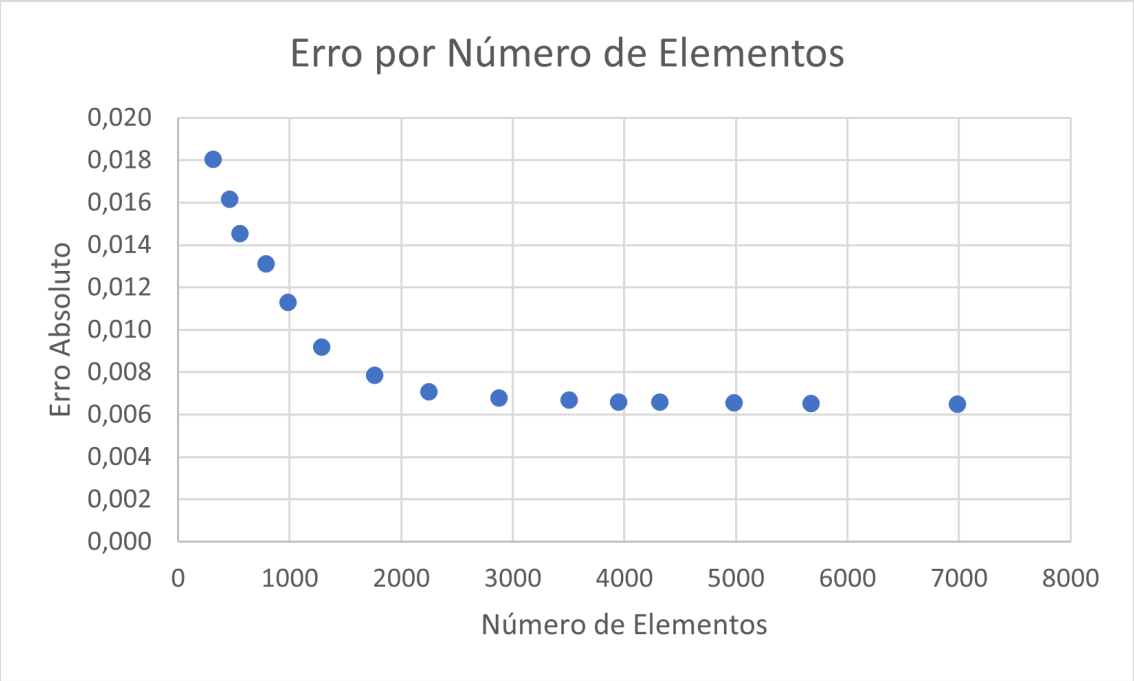



Figura 5.1: Relação entre número de elemento e o erro absoluto

Desse modo é possível visualizar que, após aproximadamente 4000 elementos, um aumento do número de elementos não traz mudanças significativas para a acurácia da solução apresentada. O número de elementos escolhido portanto, foi de 4316. Esse número foi utilizado para os três problemas apresentados neste capítulo, e para os resultados apresentados no Capítulo 6.

## 5.2 Validação com Condução de Calor em Região Homogênea

O primeiro problema a ser resolvido, apresentado em RAMOS [19], é relativamente simples, descrevendo a condução de calor em uma placa quadrada homogênea, com difusividade térmica unitária, que inicialmente se encontra à temperatura de 0ºC. A partir de $t > 0$, a face esquerda do quadrado é mantida

à 0ºC, e a face direita à 100ºC, enquanto todo o resto do domínio (incluindo as margens superior e inferior), são mantidas à 0ºC, como detalhado na Figura 5.2.

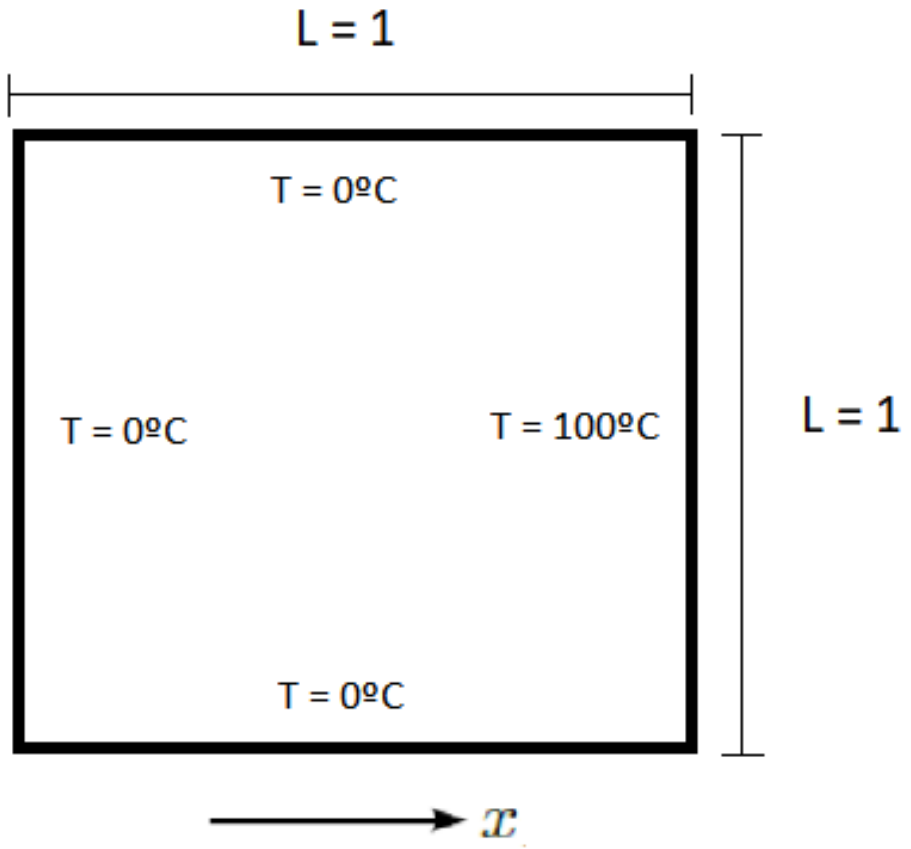



Figura 5.2: Esquema do problema 2 e suas condições de contorno [19]

A solução analítica, como demonstrada em CARSLAW [20], é mostrada nas Equações 5.1, 5.2 e 5.3.

$$T(x,t) = T(x=1)\sum_{m=0}^{\infty}\left\{\operatorname{erfc}\left[\frac{(2m+1)-x}{2\sqrt{kt}}\right] - \operatorname{erfc}\left[\frac{(2m+1)+x}{2\sqrt{kt}}\right]\right\} \tag{5.1}$$

$$\operatorname{erfc}(\phi) = 1 - \operatorname{erf}(\phi) \tag{5.2}$$

$$\text{erf}(\phi) = \frac{2}{\sqrt{\pi}} \int_0^{\phi} e^{-\omega^2} d\omega \tag{5.3}$$

Para a solução numérica, utilizamos um passo de tempo ($\Delta t$) de 0,0004 segundos, com 400 iterações. O cálculo foi realizado para a posição x = 0,5 (metade exata da placa). Para a solução analítica, variamos $m$ em um intervalo de 0 até 1000. O resultado pode ser melhor visualizado no gráfico da Figura 5.3.

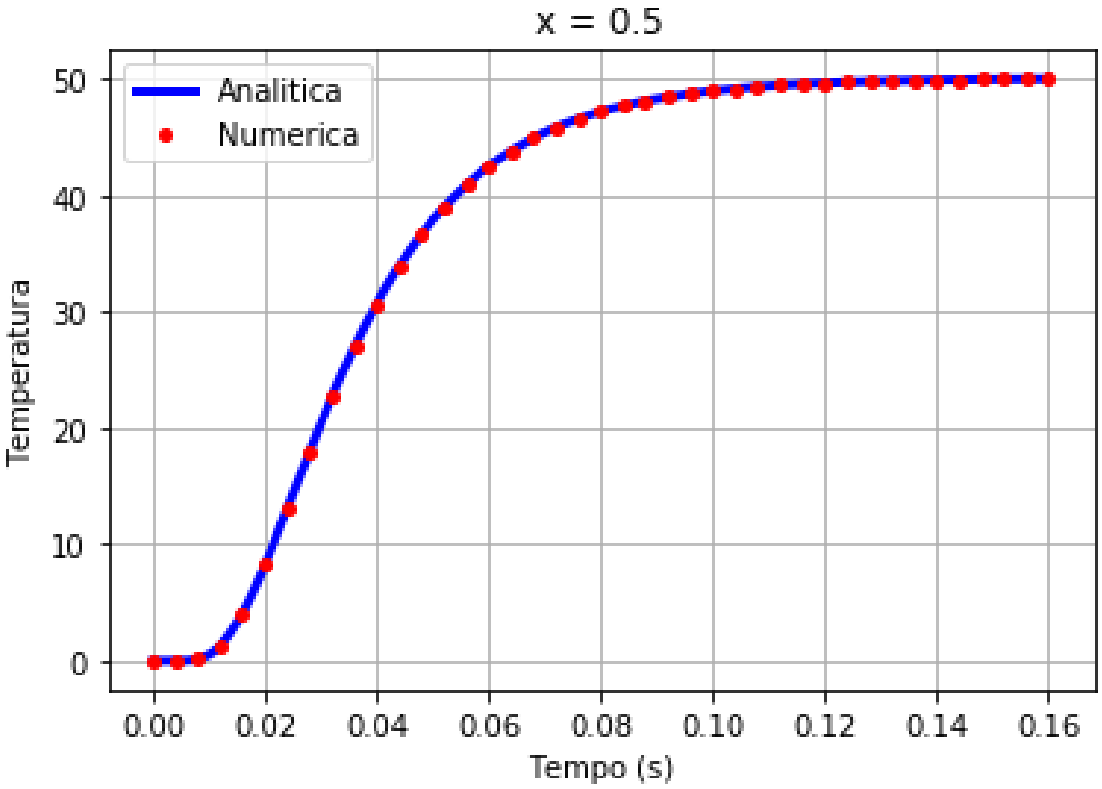



Figura 5.3: Soluções analítica e numérica encontradas para x = 0,5

## 5.3 Validação com Condução de Calor em Região Heterogênea

O segundo problema a ser resolvido, apresentado em DA CUNHA [21], envolve o estudo da condução de calor em uma placa quadrada composta de material heterogêneo. A divisão da placa é realizada de maneira simétrica, e cada

região apresenta valores diferentes para a difusividade térmica; a parte superior possuí um décimo da difusividade da parte inferior. A geometria e as condições são melhor explicadas na Figura 5.4, retiradas do trabalho de DA CUNHA [21].

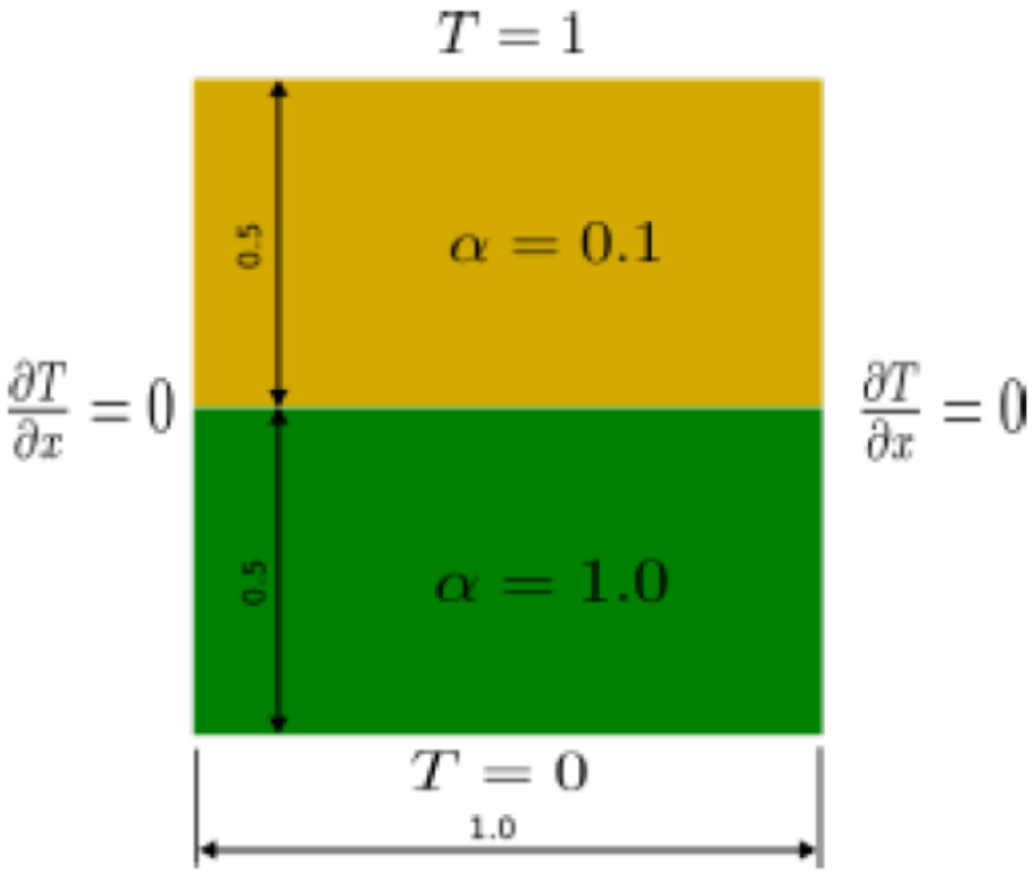



Figura 5.4: Esquema do problema 2 e suas condições de contorno [21]

A solução analítica é apresentada na Equação 5.4. Uma vez que não há fluxo de calor na direção horizontal(x), a solução depende apenas do eixo vertical (y). As equações foram encontradas aplicando a lei de Fourier (ou lei da condução de calor, apresentada no Capítulo 2) ao balanço do fluxo de calor na direção vertical. Para a comparação com a solução numérica, o problema foi resolvido na posição $x = 0,5$, no eixo que corta a placa em sua metade exata.

$$T(y) = \begin{cases} \frac{2}{11}y, & 0 \leq y \leq 0,5 \\ \frac{20}{11}y - \frac{9}{11}, & 0,5 < y \leq 1,0 \end{cases} \tag{5.4}$$

Foram utilizados passo de tempo ($\Delta t$) de 0,05 segundos, para 400 iterações do método. Os resultados são mostrados abaixo; no primeiro gráfico (Figura 5.5) podemos ver a comparação entre as soluções numérica e analítica, e no segundo (Figura 5.6) foi exibido o erro absoluto.

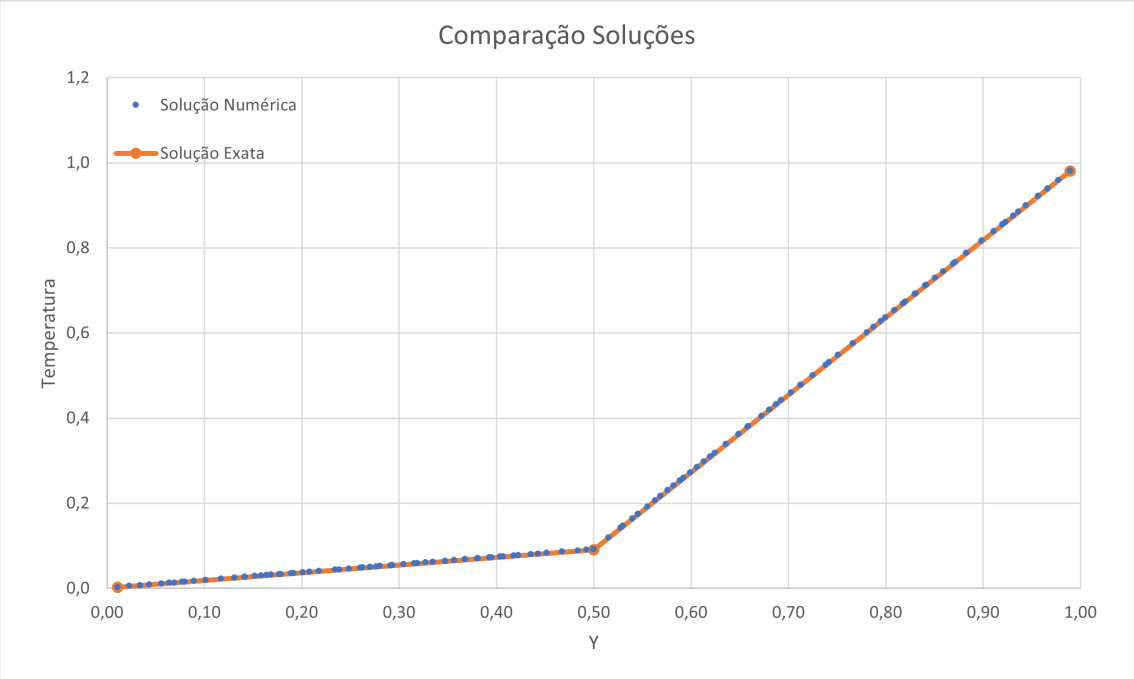



Figura 5.5: Soluções analítica e numérica encontrada para x = 0,5

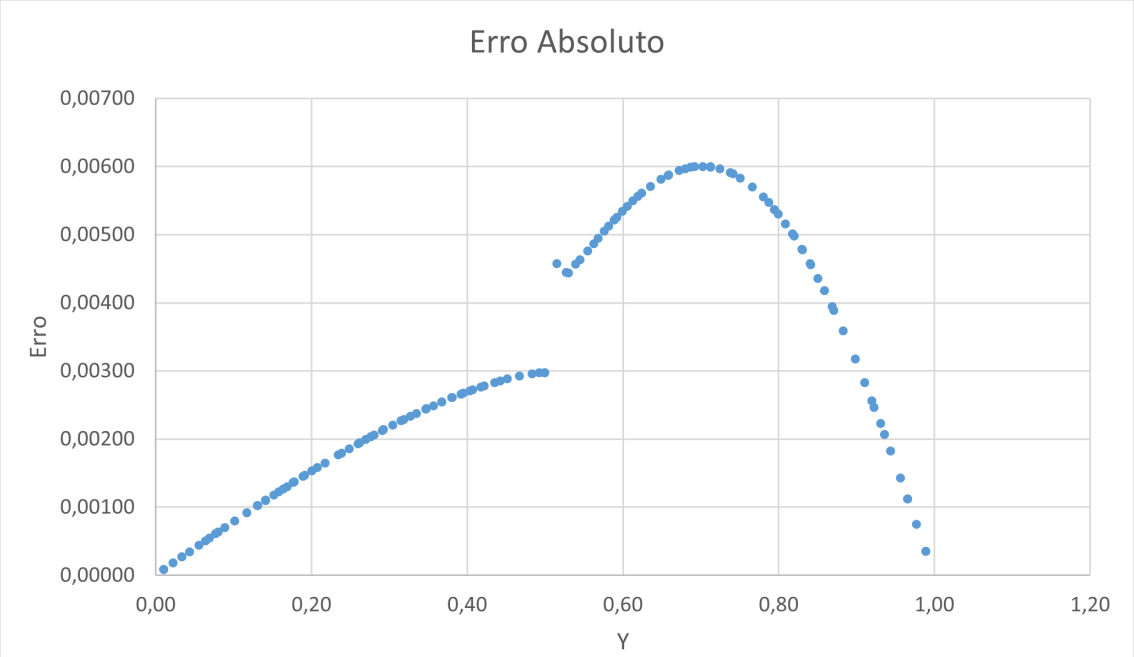



Figura 5.6: Erro absoluto da solução numérica

## 5.4 Verificação com Condução de Calor em Região Heterogênea Não-Linear

O terceiro e último problema a ser resolvido, também mostrado em apresentado em RAMOS [19], abrange novamente a a condução de calor em domínio heterogêneo; no entanto, não há uma divisão bem definida entre duas áreas do domínio com materiais diferentes com propriedades distintas. Neste caso, levaremos em consideração que as propriedades do material (mais especificamente, a condutividade térmica e a capacidade térmica) variam de acordo com a temperatura, como indicado nas Equações 5.5 e 5.6.

$$c_p = 0,002T + 0,5, \quad 0 \leq T \leq 100 \tag{5.5}$$

$$k = 0,01T + 1, \quad 0 \leq T \leq 100 \tag{5.6}$$

Ademais, todas as outras características do problema permanecem idênticas ao primeiro problema apresentado na seção 5.2, incluindo a geometria e condições de contorno. Para o cálculo da difusividade térmica, o valor da densidade utilizado foi de 0,5 kg/m³. Utilizamos novamente um passo de tempo ($\Delta t$) de 0,0004 segundos, com 400 iterações, e os cálculos foram realizados com referência à posição x = 0,5.

Para o cálculo da solução numérica, uma vez que as propriedades variam com a temperatura, após cada iteração do cálculo será necessário recalcular as matrizes. A implementação dessa correção não apresenta dificuldades, com a adição de um **loop** que incorpora todo o processo de montagem das matrizes e algumas alterações pequenas; no entanto, isso aumenta consideravelmente o custo computacional do processo. Para um exemplo simples como esse, foi possível obter uma solução, mas para casos mais complexos talvez fossem encontradas dificuldades. Outro detalhe a ser comentado, é que na Equação 3.11, consideramos a condutividade térmica constante, de modo a sair da integral e simplificar os

cálculos. Nesse caso, tal consideração não é aplicável, e foi realizado o desenvolvimento da equação de condução de calor sem essa simplificação.

Por último, outro ponto importante a se considerar é a ausência de uma solução analítica para a comparação com o resultado numérico encontrado. Como substituto, realizamos a comparação com os resultados encontrados por RAMOS [19]; o trabalho apresenta duas soluções numéricas diferentes, o Método de Multiescala e o Método de Elementos Finitos. Como pode ser observado nos gráficos exibidos nas Figuras 5.7 e 5.8, após uma comparação qualitativa visual, é possível observar que os métodos encontram soluções compatíveis e suficientemente próximas. Nesse caso, como não há solução analítica, realizamos uma verificação, e não uma validação.

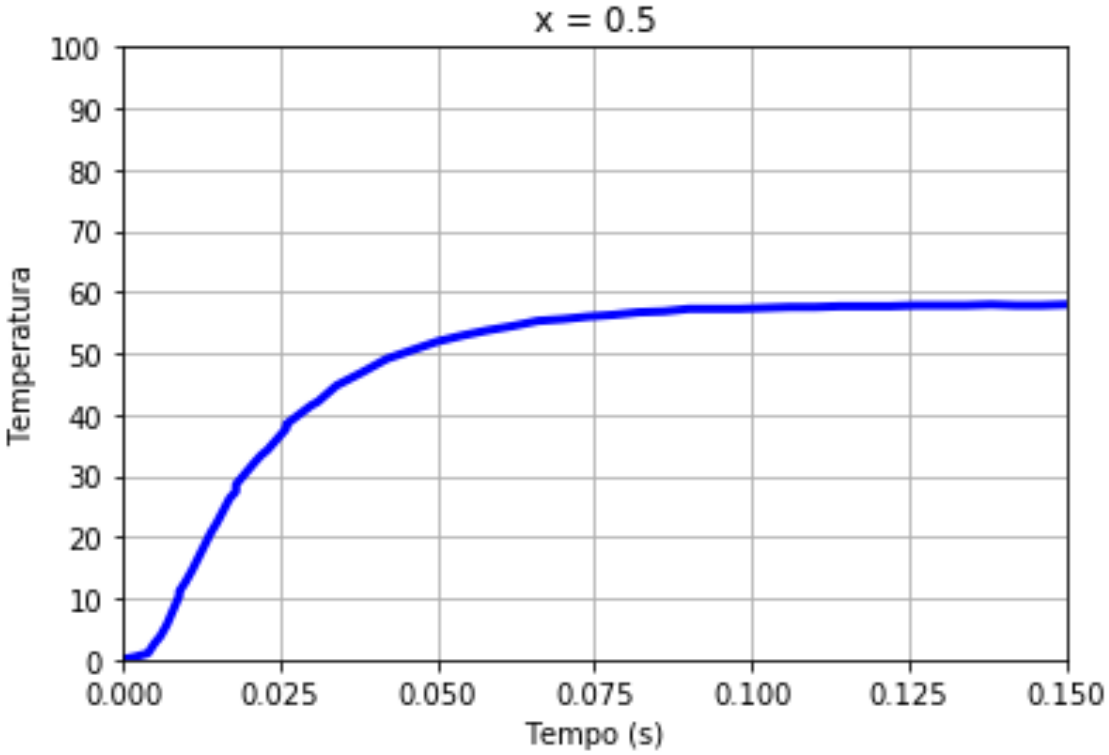



Figura 5.7: Solução numérica encontrada para x = 0,5

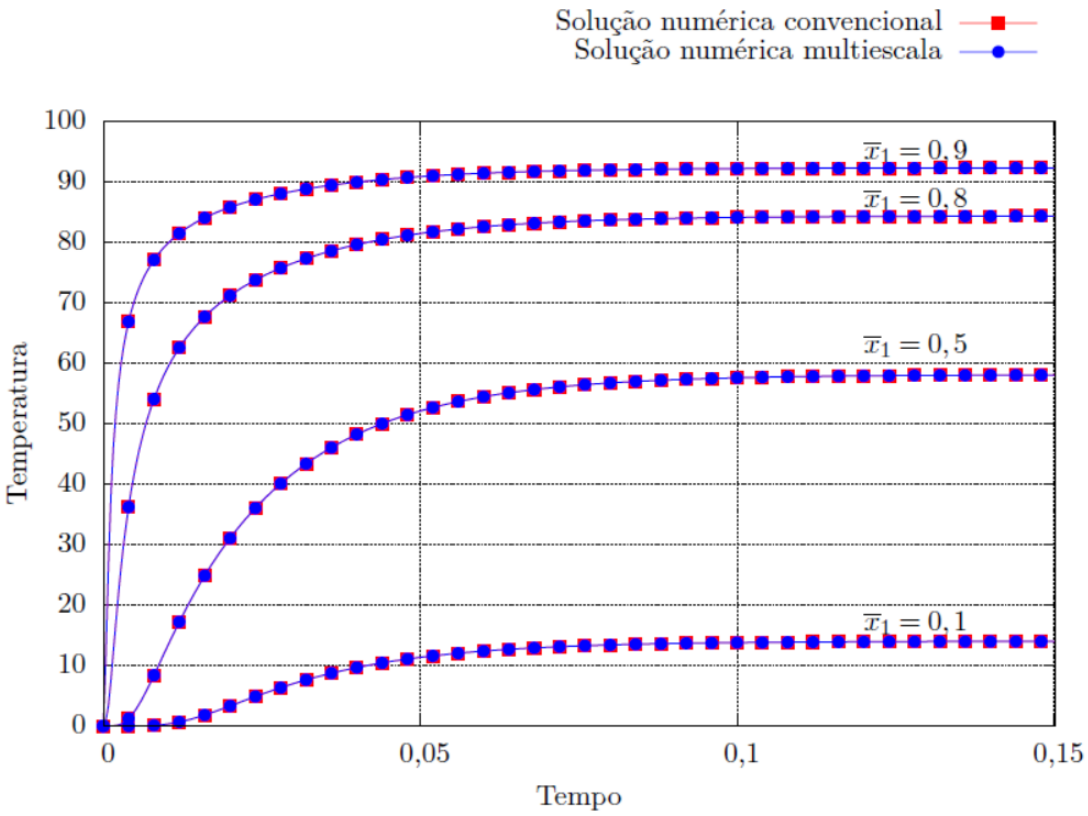



Figura 5.8: Soluções encontradas com o Método Multiescala e Método de Elementos Finitos (convencional) encontrados por RAMOS [19]

Concluindo, após a demonstração em três exemplos distintos que as soluções encontradas pelo Método de Elementos Finitos implementado neste trabalho são suficientemente corretas, podemos considerar o código numérico como vaidade. Portanto, seguimos para o próximo Capítulo, onde apresentaremos a simulação final da CPU, que é o objetivo primário deste trabalho.

# Capítulo 6

# Resultados

Uma vez explicitados os métodos utilizados para a obtenção de resultados, e comprovando a acurácia do método comparando com outros problemas, podemos enfim aplicar o Método de Elementos Finitos para estudar o comportamento da temperatura em uma CPU composta de materiais heterogêneos.

Como já mencionado na Introdução deste trabalho, utilizamos como referência o modelo de Unidade Central de Processamento apresentado no estudo de TU [1]. Embora esse artigo exponha objetivos e metodologias consideravelmente diferentes dos apresentados aqui, ainda assim oferece dados de grande significância sobre a forma, composição e comportamento de "microchips".

Primeiramente, é importante determinar os materiais constituem a composição da CPU mostrada na Figura 1.1. A parte central, onde efetivamente ocorre o processamento realizado pelo componente eletrônico no funcionamento do computador, é composta de múltiplas placas de silício, através dos quais passa a corrente elétrica.

Ao redor da unidades onde ocorre o processamento, temos uma breve camada de pasta de solda, substância constantemente utilizada em componentes eletrônicos, placas de processamento, e outros produtos similares. Este produto tem como papel a proteção da superfície da CPU, evitar a oxidação, eliminar impurezas, entre outras funções. Como a pasta utilizada não foi especificada no artigo, consideramos a utilização da SAC 305, um tipo de pasta frequentemente utilizada na composição de CPUs, composta de 96,5% de estanho, 3% de prata 0,5% de cobre.

Por último, ao redor de toda a periferia da CPU temos uma grande concentração de pequenos cilindros de cobre. Esses cilindros tem como função principal ampliar a capacidade de dissipação do calor gerado pelo processamento que ocorre na parte central da CPU. Uma vez que o cobre apresenta uma alta difusividade térmica (que pode ser definida como a capacidade do material de transferir calor da área com maior temperatura para a área de menor temperatura) em relação aos demais materiais da CPU, os cilindros cumprem de maneira satisfatória seu propósito.

Portanto, uma vez apresentados os materiais da CPU, e suas características, podemos discutir as considerações e aproximações utilizadas para a simulação propriamente dita. Primeiramente, tornamos as áreas características de cada material como contínuas. Deste modo, ao invés de múltiplas seções compostas de silício no centro da CPU, consideramos uma área quadrada interrupta de silício. O mesmo para os cilindros de cobre no contorno do componente eletrônico; uma região continuamente composta de cobre, em oposição à múltiplos cilindros individuais. Para o escopo deste trabalho, consideramos que tais aproximações não devem interferir de maneira apreciável no resultado final.

Sobre a a geometria da CPU, definimos que este seria constituído como um quadrado de 20mm de lado. As regiões de diferentes materiais seriam organizadas de modo a serem quadrados concêntricos. O quadrado composto por silício teria lado de 6mm, e a área de SAC 305 seria de um quadrado de lado de 8mm. As Figuras 6.1 e 6.2 ilustram o esquema de maneira mais apropriada.

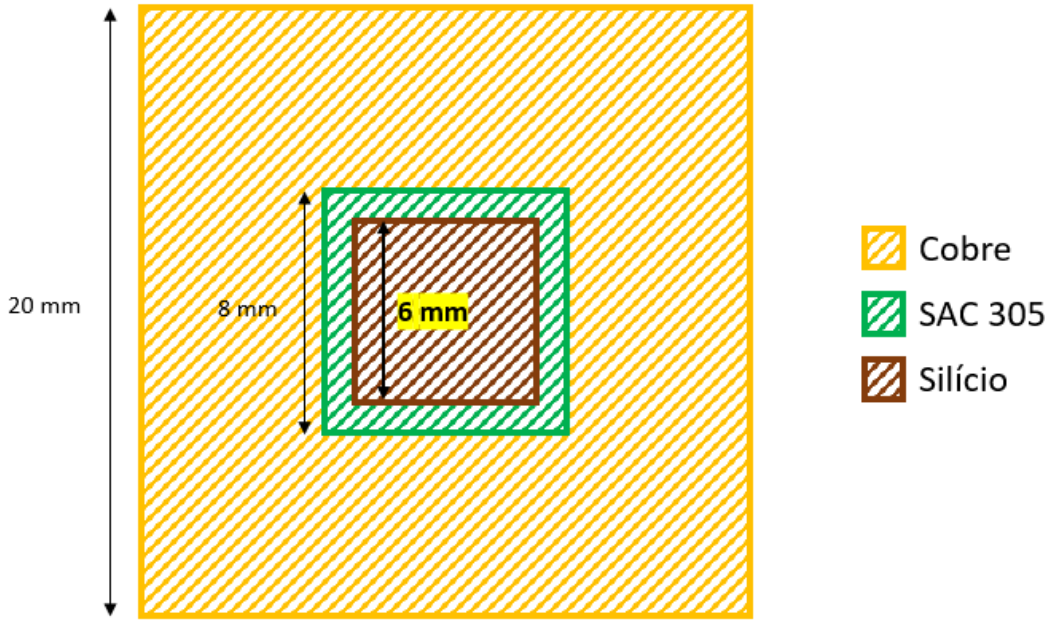



Figura 6.1: Esquema da geometria e materiais considerados para a simulação

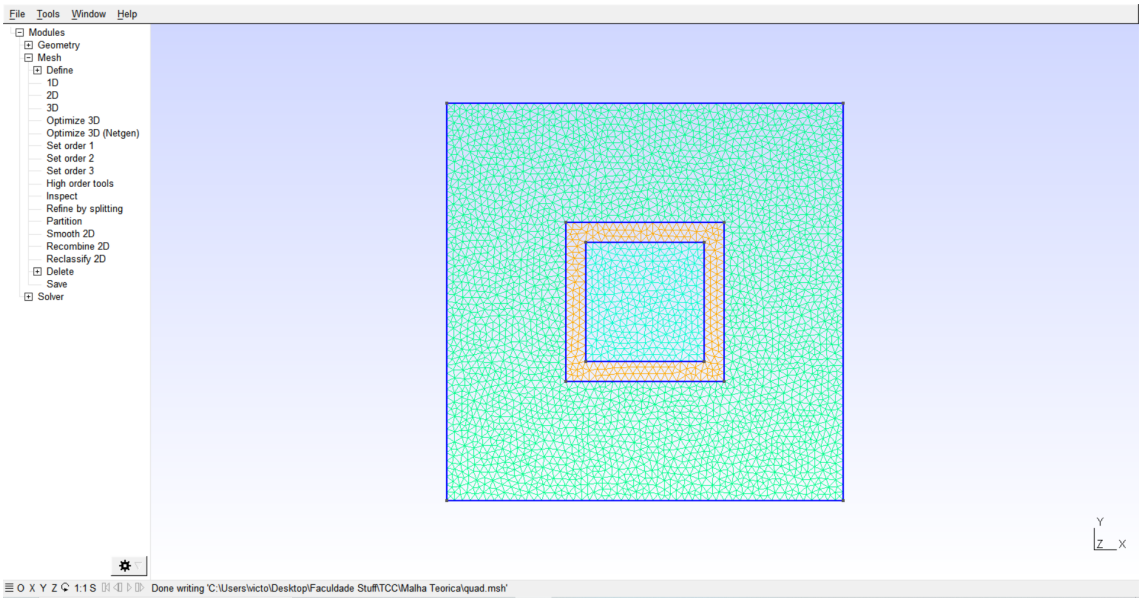



Figura 6.2: Imagem da malha utilizada para a modelagem do problema

Uma vez tendo definido as características geométricas do problema, vamos às demais considerações. Como o processamento ocorre nos processadores de silício localizados no centro da CPU, levaremos em consideração que será apenas nessa área onde ocorrerá geração interna de calor. Utilizaremos também a

hipótese de que a largura da parte de cobre, onde ocorrerá a dissipação de calor, é grande o suficiente para isolar a CPU de influências externas (como transmissão de calor por convecção). Deste modo, consideramos para condições de contorno que a temperatura em toda a borda é constante.

Outra informação importante proveniente do artigo de TU [1], é a faixa de temperatura na qual uma CPU pode trabalhar em condições ideais: entre 40°C e 50°C. Portanto, utilizaremos esses dados como referência para mensurar se os resultados que descrevem a distribuição de calor na CPU são aceitáveis ou não. Utilizando também como referência o artigo, supomos que a geração de calor no processador seria entre 20W e 70W. Desse modo, realizamos seis simulações, com valores para a geração de calor de 20W, 30W, 40W, 50W, 60W e 70W. Todas as demais características e variáveis do problema permaneceram idênticas.

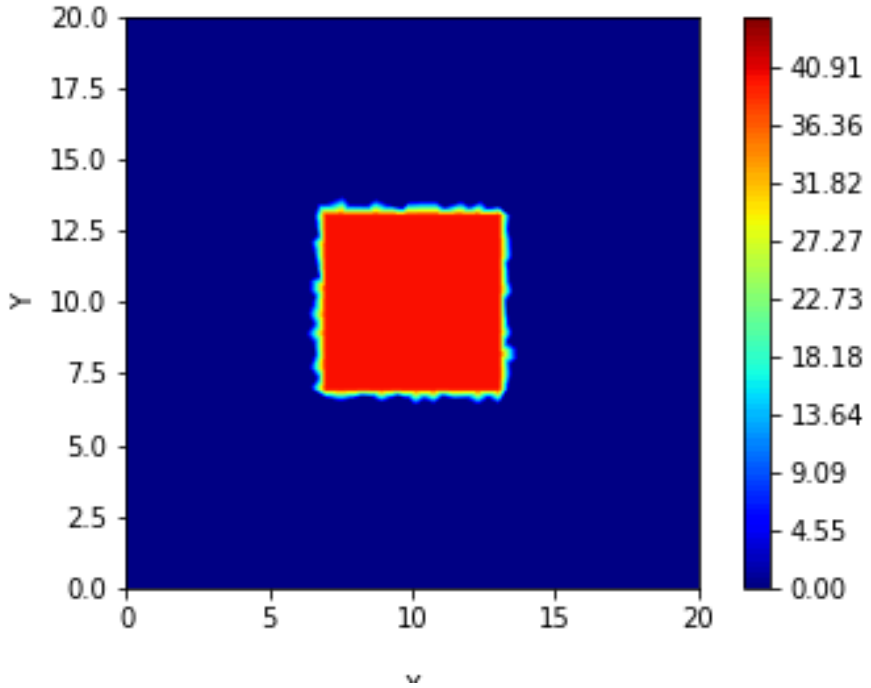



Figura 6.3: Distribuição da geração interna de calor para o caso de 40W (de t =0s à t = 1.5s)

Consideramos, como condições inciais (para todos os problemas a serem resolvidos) então: que todo o componente eletrônico estaria à temperatura ambiente de 20°C, ou seja:

$$T(x, y, t = 0) = 20°C \tag{6.1}$$

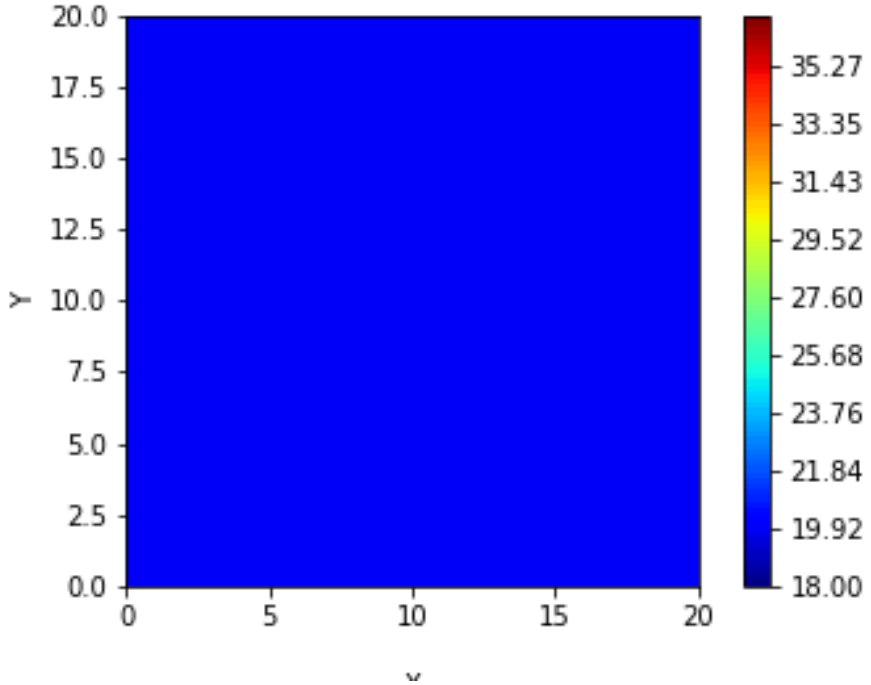



Figura 6.4: Temperatura na CPU tem t = 0s (condições iniciais)

Como mencionado anteriormente, haverá geração interna de calor na região central composta de silício (como mostrado na Figura 6.2).

Como também mencionado anteriormente, para as condições de contorno, consideramos que a temperatura em toda a borda da CPU é constante, se mantendo à temperatura ambiente, ou seja, 20°C.

Utilizamos um passo de tempo $\Delta t$ de 0,0001 segundos, com um total de 15.000 iterações, totalizando assim 1.5 segundos para observar o comportamento da transferência de calor.

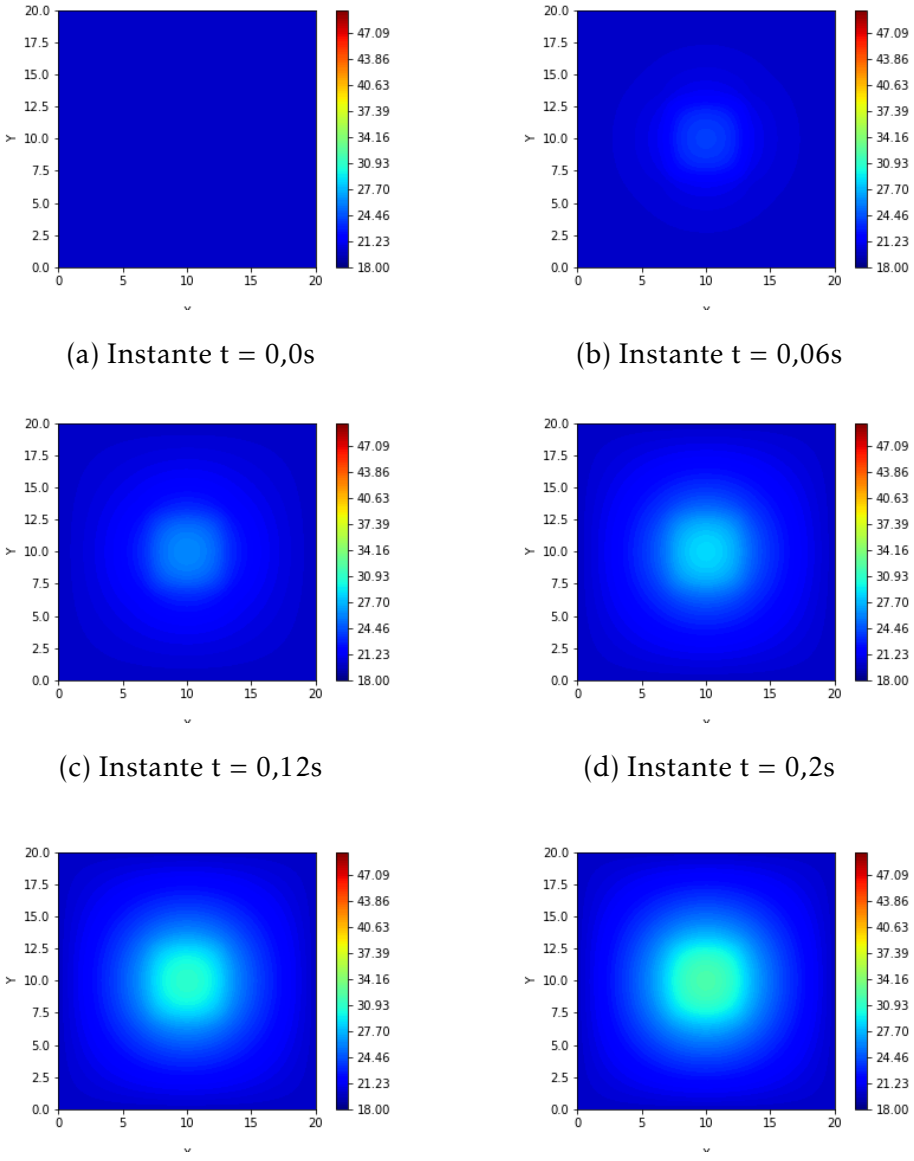
Os valores da difusividade térmica dos materiais que compõe a CPU são mostrados abaixo:

$$\alpha_{Cobre} = 440mm^2/s \tag{6.2}$$

$$\alpha_{SAC305} = 35mm^2/s \tag{6.3}$$

$$\alpha_{Silicio} = 88mm^2/s \tag{6.4}$$

Podemos então, finalmente, apresentar os resultados encontrados nas simulações. Uma vez que estamos utilizando como referência a temperatura ideal para o funcionamento da CPU, para cada caso de geração interna de calor plotamos em um gráfico a temperatura máxima registrada (sempre ocorrendo no centro) ao longo do tempo decorrido. Desse modo, podermos verificar se o componente eletrônico poderia funcionar com esse nível de geração de energia. Exibimos também, para melhor visualização do fenômeno, a distribuição de temperatura em seis instantes de tempo: t = 0s, t= 0,06s, t= 0,12s, t= 0,2s, t= 0,4s t = 1,0s. Paramos em t = 1,0s pois, como pode ser observado nos gráficos, por esse instante de tempo que a temperatura se torna constante, uma vez que a derivada temporal não tem mais efeito.




(a) Instante t = 0,0s

(b) Instante t = 0,06s

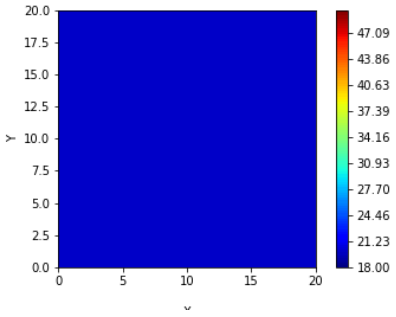
(c) Instante t = 0,12s

(d) Instante t = 0,2s

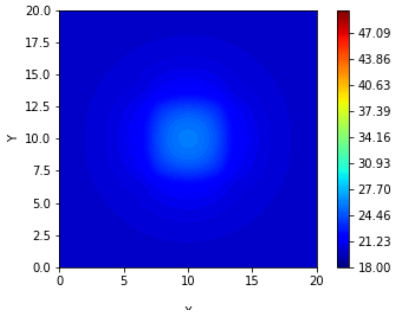
(e) Instante t = 0,4s

(f) Instante t = 1,0s

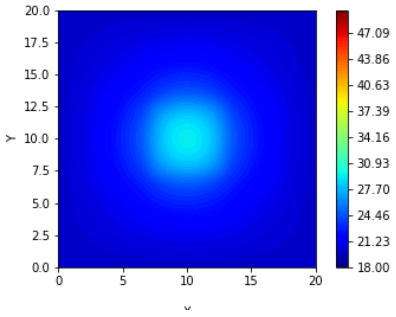
Figura 6.5: Distribuições de temperatura para Q = 20W



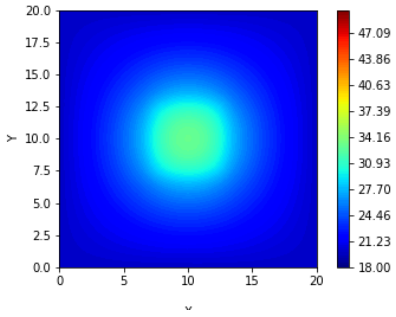

(a) Instante t = 0,0s



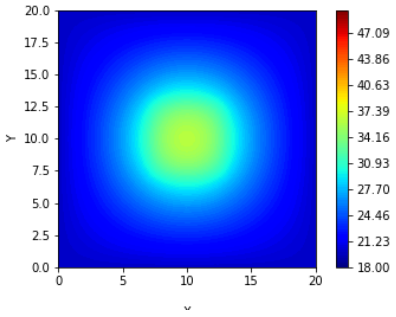

(b) Instante t = 0,06s



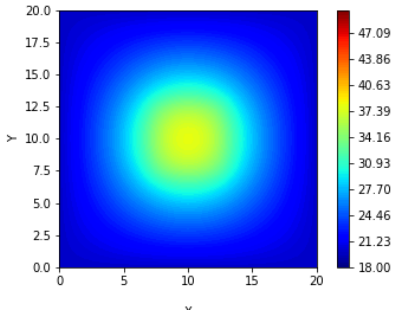

(c) Instante t = 0,12s




(d) Instante t = 0,2s

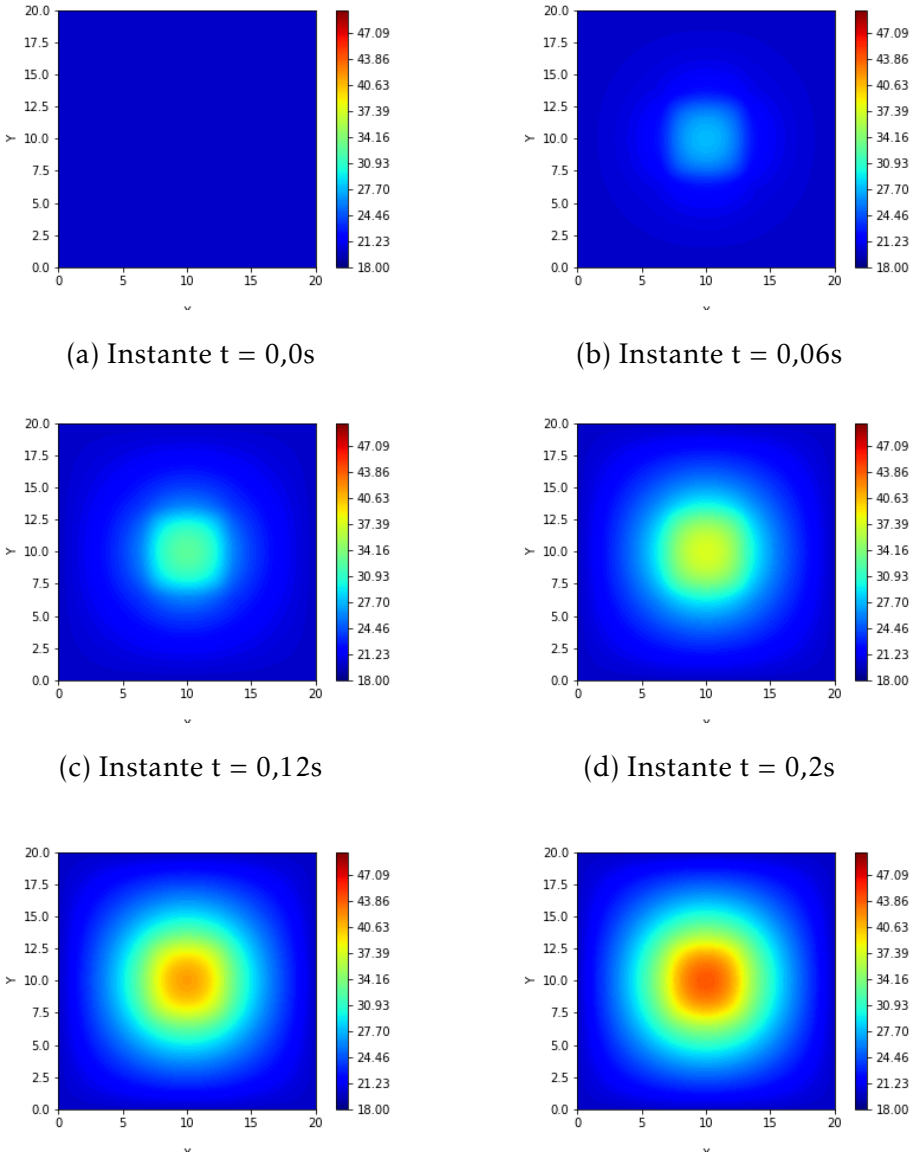



(e) Instante t = 0,4s




(f) Instante t = 1,0s

Figura 6.6: Distribuições de temperatura para Q = 30W




(a) Instante t = 0,0s

(b) Instante t = 0,06s

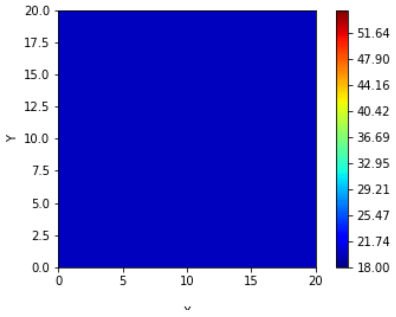
(c) Instante t = 0,12s

(d) Instante t = 0,2s

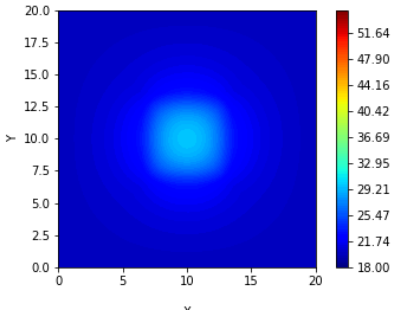
(e) Instante t = 0,4s

(f) Instante t = 1,0s

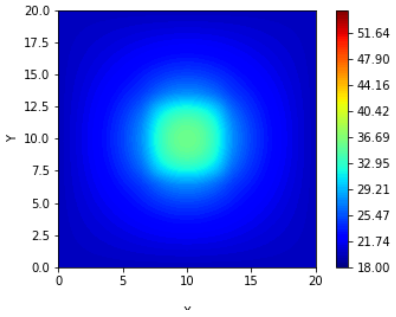
Figura 6.7: Distribuições de temperatura para Q = 40W



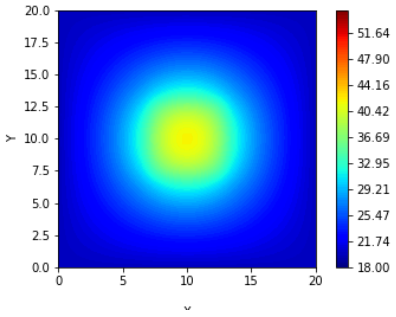

(a) Instante t = 0,0s



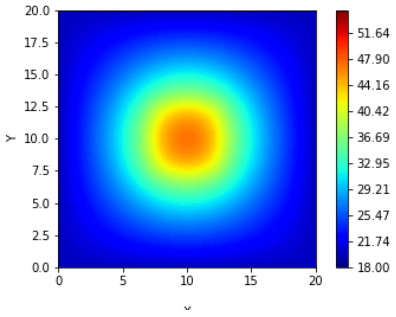

(b) Instante t = 0,06s



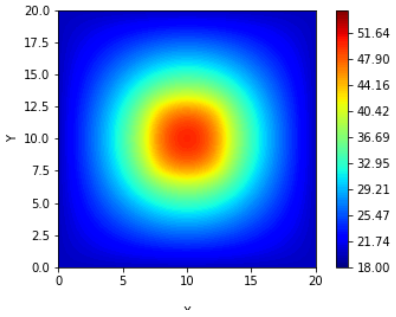

(c) Instante t = 0,12s




(d) Instante t = 0,2s

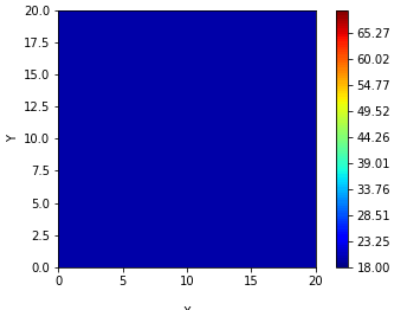



(e) Instante t = 0,4s

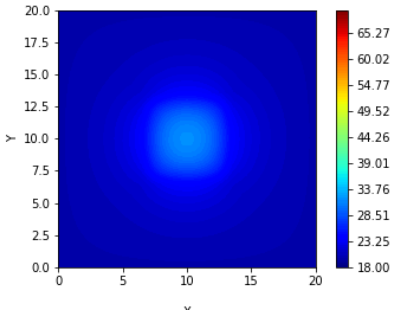



(f) Instante t = 1,0s

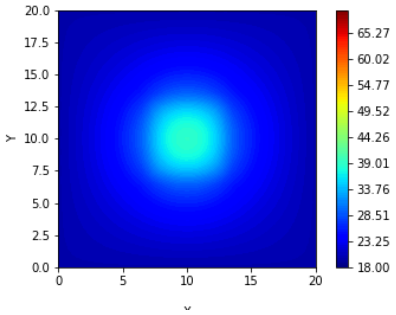
Figura 6.8: Distribuições de temperatura para Q = 50W



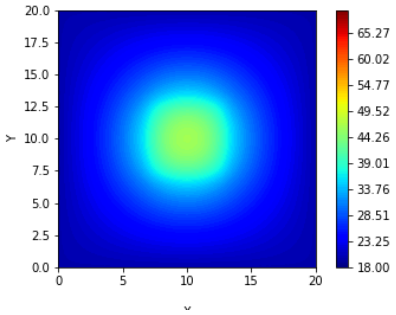

(a) Instante t = 0,0s



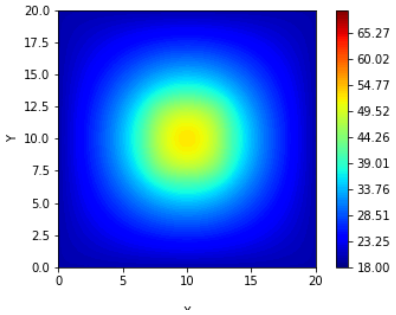

(b) Instante t = 0,06s



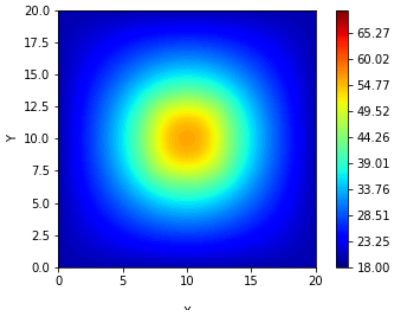

(c) Instante t = 0,12s




(d) Instante t = 0,2s

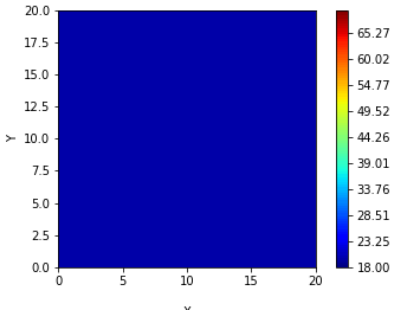



(e) Instante t = 0,4s

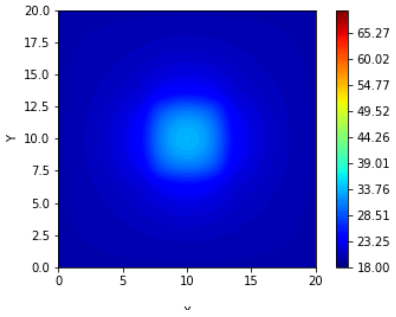



(f) Instante t = 1,0s

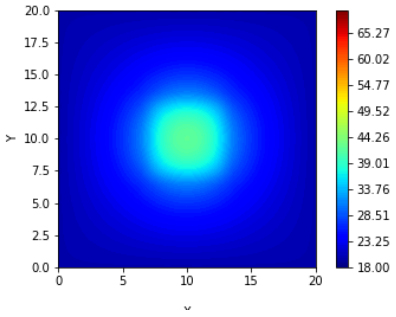
Figura 6.9: Distribuições de temperatura para Q = 60W



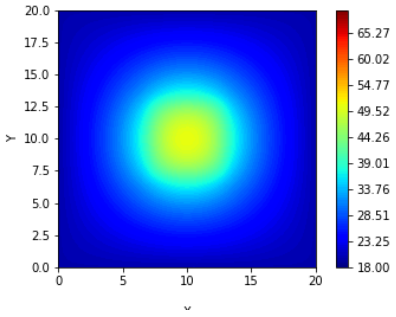

(a) Instante t = 0,0s



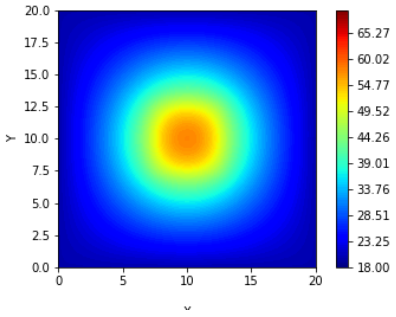

(b) Instante t = 0,06s



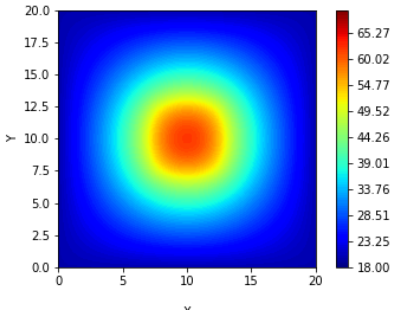

(c) Instante t = 0,12s




(d) Instante t = 0,2s

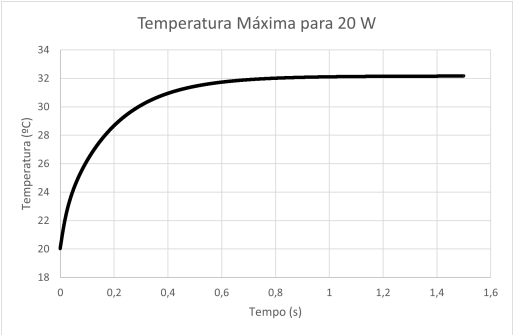



(e) Instante t = 0,4s

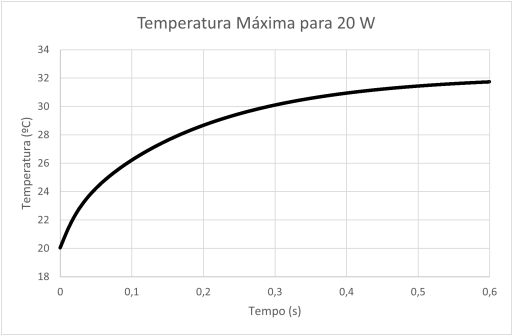



(f) Instante t = 1,0s

Figura 6.10: Distribuições de temperatura para Q = 70W

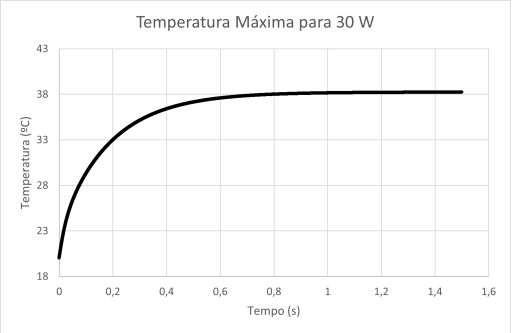



(a) Período de tempo de t = 0s até t = 1,5s

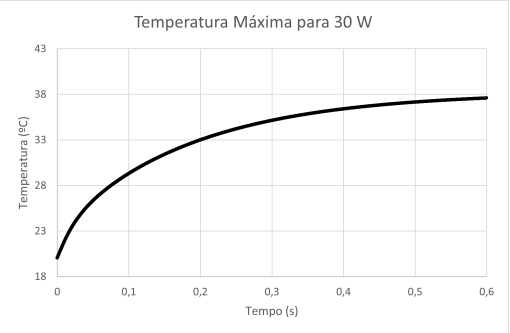



(b) Período de tempo de t = 0s até t = 0,6s

Figura 6.11: Temperatura máxima para Q = 20W

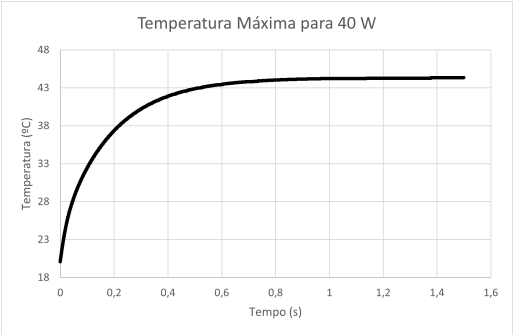



(a) Período de tempo de t = 0s até t = 1,5s

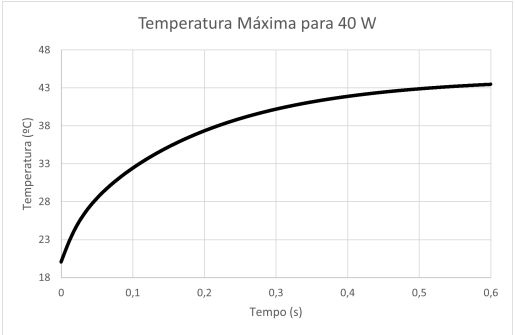



(b) Período de tempo de t = 0s até t = 0,6s

Figura 6.12: Temperatura máxima para Q = 30W

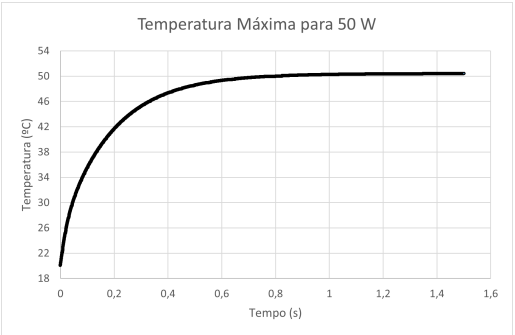



(a) Período de tempo de t = 0s até t = 1,5s

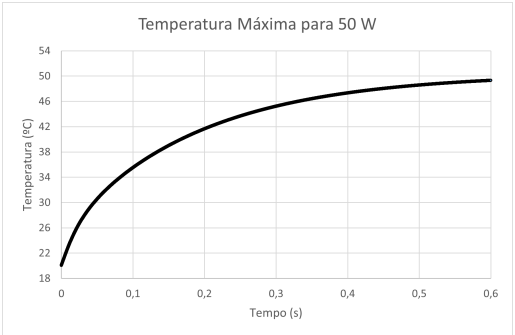



(b) Período de tempo de t = 0s até t = 0,6s

Figura 6.13: Temperatura máxima para Q = 40W

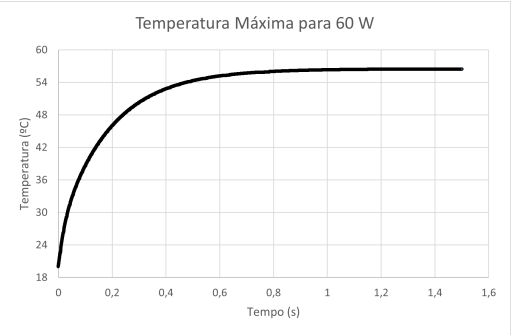



(a) Período de tempo de t = 0s até t = 1,5s

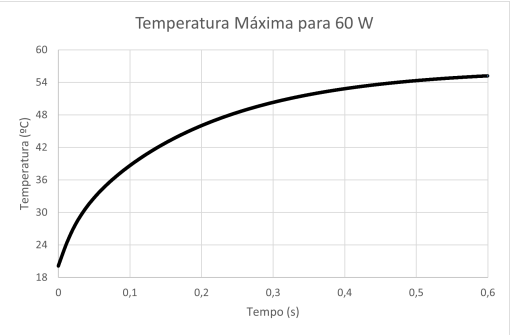



(b) Período de tempo de t = 0s até t = 0,6s

Figura 6.14: Temperatura máxima para Q = 50W

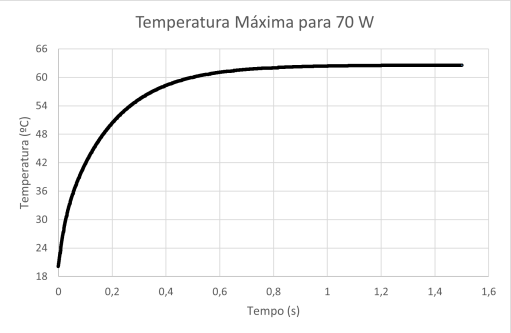



(a) Período de tempo de t = 0s até t = 1,5s

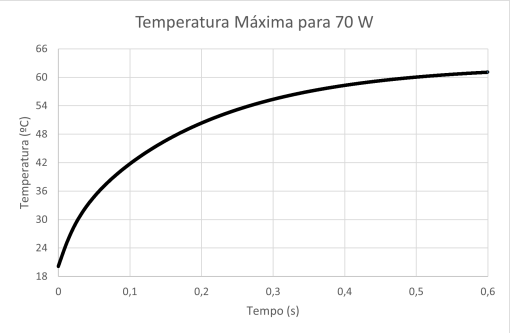



(b) Período de tempo de t = 0s até t = 0,6s

Figura 6.15: Temperatura máxima para Q = 60W




(a) Período de tempo de t = 0s até t = 1,5s




(b) Período de tempo de t = 0s até t = 0,6s

Figura 6.16: Temperatura máxima para Q = 70W

Na Figura 6.5, podemos observar o comportamento da distribuição de temperatura na CPU ao longo do tempo, para um calor interno gerado de 20W. Como observado anteriormente, após aproximadamente o instante t = 1.0s, a distribuição de temperatura se torna constante, uma vez que a derivada temporal não tem mais efeito, e entramos na parte permanente. Nesse caso, a temperatura máxima registrada é de aproximadamente 32°C, de modo que a CPU pode operar com tranquilidade. A temperatura máxima pode ser melhor observada na Figura 6.6. Na parte a) podemos ver a temperatura máxima em todo o espaço de tempo da simulação, enquanto na parte b) temos uma visão mais detalhada do período

de tempo entre t = 0s e t = 0,6s, provendo mais informações sobre a influência da derivada temporal.

A Figura 6.7 é referente ao caso com geração interna de 30W. Podemos observar a presença de temperaturas relativamente maiores do que no caso de 20W, mas ainda dentro do limite de segurança da CPU de 40°C a 50°C. Como é possível verificar na Figura 6.8, a temperatura máxima de 38°C é atingida aproximadamente no instante t = 1.0s, confirmando o padrão já referenciado da influência da derivada temporal.

Na Figura 6.9 é exibido o perfil de temperatura na CPU é para um calor interno de 40W. É possível observar um aumento expressivo de temperaturas com relação ao caso de 20W. Na figura 6.10, vemos que a temperatura máxima registrada de 44°C já está dentro da faixa limite para um funcionamento saudável da CPU, mas sem entrar em território proibitivo.

Considerando agora a Figura 6.11, para o caso de 50W, vemos que a temperatura aumenta expressivamente. O limite da palheta de cores, que nas Figuras 6.5, 6.7 e 6.9 eram de 50°C, agora passa a ser de 55°C. A Figura 6.12 apresenta mais detalhes, informando que a temperatura máxima registrada é de aproximadamente 50°C, ou seja, o limite apresentado anteriormente. O funcionamento por da CPU nessa temperatura por um período de tempo extenso não é recomendado, e poderia causar danos permanentes aos componentes eletrônicos.

Para a Figura 6.13 e 6.15, aumentamos novamente o limite da barra de cores, desta vez para 70°C, para melhor detalhar o perfil de temperatura. Para o caso de 60W, a Figura 6.14 nos informa uma temperatura máxima de 56°C, que excede o limite estabelecido, e é proibitivo para o funcionamento da CPU.

Por fim, a Figura 6.15 nos mostra o último caso, com a simulação realizada para 70W. A temperatura máxima registrada na CPU, de aproximadamente 63°C como axibido na Figura 6.16, torna o funcionamento do modelo de CPU utilizado neste trabalho inviável.

Portanto podemos concluir que, dentro das considerações e especificações para o funcionamento da CPU, o mesmo poderá operar de maneira segura entre 20W e 40W gerados no processamento sem riscos de superar a temperatura limite de 50°C. É possível observar, (especialmente na parte f das imagens, que mostra

a distribuição de temperatura no regime permanente) que os casos de 50W, 60W e 70W superam a temperatura máxima de 50°C.

É importante também reforçar que, aproximadamente após o instante t = 1.0s, a temperatura atinge seu máximo e permanece constante, o que comprova que o comportamento da derivada temporal permanece o mesmo em todos os casos. Para os valores superiores de geração de calor (de 50W a 70W e além), não seria recomendada a operação, com risco de danificar o componente eletrônico.

Utilizando então os dados obtidos nas simulações das Figuras 6.6 à 6.16, podemos plotar em um gráfico os valores das temperaturas máximas obtidas para cada valor de geração interna de calor. O resultado é mostrado na Figura 6.17.

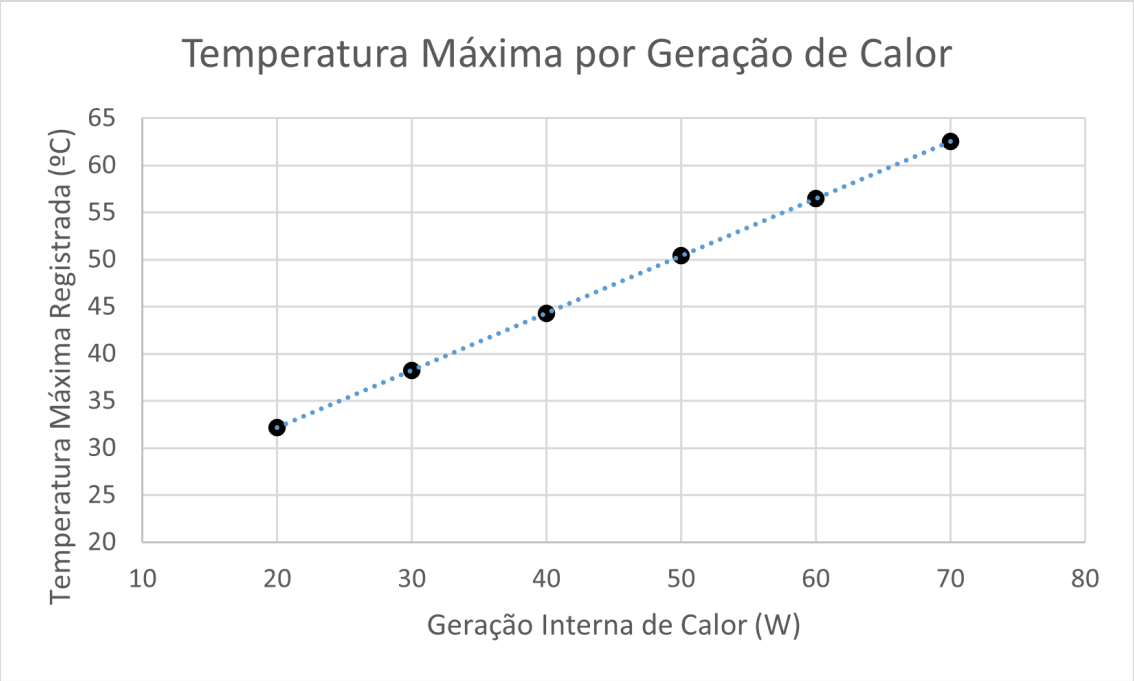



Figura 6.17: Temperatura máxima registrada na CPU por taxa de calor gerado

Podemos observar que as duas grandezas se comportam de maneira linear uma em relação a outra. Desse modo, podemos escrever uma função que descreva essa relação, possibilitando que seja possível descobrir a temperatura máxima que será registrada na configuração apresentada da CPU para um dado calor interno, demonstrada pela Equação 6.5.

$$T_{max} = 0,6076Q + 20 \tag{6.5}$$

Onde $T_{max}$ é a temperatura máxima registrada e Q é a geração interna de

calor.

# Capítulo 7

# Conclusão

Examinando os resultados exibidos nos últimos dois Capítulos, é possível observar que a aplicação do Método de Elementos Finitos para a solução de problemas de transferência de calor por condução apresenta resultados satisfatórios, levando em consideração a complexidade das equações e o custo computacional de processamento exigido para que os resultados fossem encontrados.

Definimos que, para o modelo de CPU apresentado, pode funcionar idealmente (apresentando uma temperatura máxima de até 50°C) com um calor interno gerado de até 50W; calor interno esse gerado pela corrente elétrica que aciona os processadores de silício. Foi possível também determinar uma relação linear entre a temperatura máxima registrada, e esse calor interno gerado no centro da CPU.

Os resultados mostrados na Seção 5.3 do Capítulo 5 mostram, por exemplo, que o erro absoluto máximo registrado foi de aproximadamente 0.006. Desse modo, quando aproximações com esse grau de precisão forem suficientes, a aplicação do Método de Elementos Finitos é uma ótima escolha, uma vez que a aplicação código em *Python* é relativamente simples e, uma vez pronto, é necessário apenas realizar a modelagem da geometria necessária e implementar as condições de contorno características do problema para que resultados sejam encontrados.

Outras consideração notável é que, uma vez que esse código base esteja funcional e exibindo resultados satisfatórios, realizar alterações para lidar com problemas mais complexos não exigirá esforços desnecessariamente árduos. Por

exemplo, para a análise de problemas em três dimensões, ao contrário das duas dimensões consideradas neste trabalho, apenas exigirá mudanças na montagem das matrizes e na leitura do arquivo de texto gerado pelo *Gmsh*. O restante da lógica e estrutura não necessita de mudanças, possivelmente apenas de pequenos ajustes.

Outro exemplo de melhorias seria a expandir o modo de transferência de calor para incluir também convecção, uma vez que neste trabalho consideramos apenas a presença de condução. Diferente da inclusão de uma dimensão, abranger outro modo de transferência de calor causaria mudanças nas equações modeladas, mas as alterações necessárias no código não seriam drasticamente extensas.

Portanto, levando em consideração as reflexões expostas acima, podemos afirmar que os objetos propostos por este trabalho foram alcançados.

# Bibliografia


[1] Jianping Tu, Walter Yuen e Yishu Gong. "An Assessment of Direct Chip Cooling Enhancement Using Pin Fins". Em: *Heat Transfer Engineering* 33 (2012), pp. 1–9.

[2] M. Necati Ozisik. *Heat transfer: A basic approach*. New York: McGraw-Hill, 1985.

[3] Frank P. Incropera e David P. DeWitt. *Fundamentals of heat and mass transfer*. New York: J. Wiley, 2002.

[4] David W. Hahn e M. Necati Ozisik. *Heat conduction*. Hoboken, N.J. : Wiley, 2012.

[5] D. L. Logan. *A first course in the finite element method*. Boston: PWS Engineering, 1986.

[6] J.N. Reddy. *An Introduction to The Finite Element Method*. McGraw-Hill, 2006.

[7] O.C. Zienkiewicz, R.L. Taylor e J.Z. Zhu. *The Finite Element Method: Its Basis and Fundamentals*. Butterworth-Heinemann, 2005.

[8] C.A. FELLIPA. "A Historical Outline of Matrix Structural Analysis: A Play in Three Acts". Em: *Department of Aerospace Engineering Sciences and Center for Aerospace Structures; University of Colorado* (2000).

[9] R.W. Clough. "The finite element method in plane stress analysis". Em: *Proceedings of 2nd ASCE Conferenceon Electronic Computation* (1960).

[10] Gregoire Allaire. *Numerical Analysis and Optimization*. Oxford University Press, 2007.

[11] R. Courant. “Variational methods for the solution of problems of equilibrium and vibrations”. Em: *Bulletin of the American Mathematical Society* 49 (1943), pp. 1–23.

[12] A. Hrennikoff. “Solution of problems of elasticity by the framework method”. Em: *Journal of Applied Mechanics* 8 (1941), pp. 169–175.

[13] Gabriel da Silva Oliveira Nunes de Aguiar. “Simulação numérica da trasnferência de calor em um disco de freio durante frenagem usando o método de elementos finitos”. Em: *Universidade Federal do Rio de Janeiro* (2020).

[14] Roland W. Lewis, Perumal Nithiarasu e Kankanhalli N. Seetharamu. *Fundamentals of the Finite Element Method for Heat and Fluid Flow*. John Wiley e Sons, 2004.

[15] Gustavo R. Anjos. *Computaçãoo Científica para Engenheiros*. PEM/COPPE/UFRJ: Rio de Janeiro, 2019.

[16] G. Strang e Fix G.J. *An Analysis of The Finite Element Method*. Prentice Hall, 1973.

[17] J. Chaskalovic. *Finite Element Methodsfor Engineering Sciences*. Springer-Verlag Berlin Heidelberg, 2008.

[18] K-J. Bathe e E.L. Wilson. *Numerical Methods in Finite Element Analysis*. New Jersey: Brentice-Hall,INC., 1976.

[19] Gustavo Roberto Ramos. “MÉTODO MULTIESCALA PARA MODELAGEM DA CONDUÇÃO DE CALOR TRANSIENTE COM GERAÇÃO DE CALOR: TEORIA E APLICAÇÃO”. Tese de dout. Universidade Federal do Rio Grande do Sul, 2015.

[20] H.S. Carslaw e Jaeger J.C. *Conduction of Heat in Solids*. Oxford, 1959.

[21] Luís Henrique Carnevale da Cunha. “Formulação Corrente-Vorticidade em Problemas de Transferência de Calor Conjugado usando MEFs”. Em: *Universidade do Estado do Rio de Janeiro* (2019).

# Apêndice A

# Algoritmo

## A.1 Parte 1

```
def file_reader(file_name):
    file_name_format = file_name + ".msh"

    with open(file_name_format, 'r') as arqv:
        content = arqv.read()
    arqv.close()

    read_lines = content.split('\n')

    Chave = [["$PhysicalNames", "$EndPhysicalNames"],
             ["$Nodes", "$EndNodes"],
             ["$Elements", "$EndElements"]]
    IndexChave = 0
    status = False

    CC = dict() # Dicionário das condições de contorno
    nos = list() # Nós
    IEN = list()  # IEN
    IENcc = list() # IEN das condições de contorno
```

```
for line in read_lines:
    if(line == Chave[IndexChave][0]):
        status = True
        continue

    elif(line == Chave[IndexChave][1]):
        IndexChave += 1
        status = False
        if(IndexChave == 3):
            break
        continue

    if(status is True):
        content = line.split(" ")
        if(IndexChave == 0): # Physical Names
            if(len(content) > 1):
                CC[ int(content[1]) ] = content[2][1:-1:1]
            else:
                pass

        elif(IndexChave == 1): # Nós
            if(len(content) > 1):
                dados = PontoNodal((int(content[0])-1),
                                   float(content[1]),
                                   float(content[2]))
                nos.append(dados)

        elif(IndexChave == 2): # Elementos
            if(len(content) == 7): # Condições de Contorno

                pontos[int(content[-2]) - 1].cc = CC[int(content[-
```

```
4])]
                IENcc.append(int(content[-2]) - 1)

            elif(len(content) == 8): #IEN
                dados2 = [(int(content[-3])-1),(int(content[-2])-
1),                 (int(content[-1])-1)]
                IEN.append(dados2)
    else:
        pass
return nos, IEN, IENcc


class PontoNodal():
    def __init__(self, name_p, x_p, y_p, z_p):
        self.name = name_p
        self.x = x_p
        self.y = y_p


class Malha():
    def __init__(self, file_name):

        self.nos, self.IEN, self.IENcc = file_reader( file_name )

        self.nosNumber = len(self.nos) # Número de nós
        self.elementsNumber = len(self.IEN)  # Número de elementos

        # Definir lista as coordenadas x:
        self.X = [self.nos[i].x for i in range(self.nosNumber)]
        # Definir lista as coordenadas y:
        self.Y = [self.nos[i].y for i in range(self.nosNumber)]
```

```
        self.x_min = min(self.X)
        self.x_max = max(self.X)
        self.y_min = min(self.Y)
        self.y_max = max(self.Y)

        malha = Malha(nome_arquivo)
```

## A.2 Parte 2

```
def mult-por-escalar(escalar, matriz):
    for i in range(len(matriz)):
        for j in range(len(matrizt[i])):
            matriz[i][j] *= escalar
    return matriz


K = [ [0 for i in range(malha.nosNumber)] for i in range(malha.nosNumber)]
M = [ [0 for i in range(malha.nosNumber)] for i in range(malha.nosNumber)]

Alpha = [ [0 for i in range(3)] for i in range(malha.elementsNumber)]


    # Criar matriz dos alfas

    for g in range(malha.elementsNumber):
        for h in range(3):
            x = malha.X[malha.IEN[g][h]]
            y = malha.Y[malha.IEN[g][h]]
```

```
# Centro de Silício
if x > 7.0 and x < 13.0 and y > 7.0 and y < 13.0:
    Alpha[g][h] = AlfaSil

# Borda de Cobre
elif x < 6.0 or x > 14.0 or y < 6.0 or y > 14.0:
    Alpha[g][h] = AlfaCobre

# Fronteira entre SAC e Cobre
elif x == 6.0 and y > 6.0 and y < 14.0:
    Alpha[g][h] = AlfaSacCobre

elif x == 14.0 and y > 6.0 and y < 14.0:
    Alpha[g][h] = AlfaSacCobre

elif y == 6.0 and x > 6.0 and x < 14.0:
    Alpha[g][h] = AlfaSacCobre

elif y == 14.0 and x > 6.0 and x < 14.0:
    Alpha[g][h] = AlfaSacCobre

# Fronteira entre SAC e Silício

elif x == 7.0 and y > 7.0 and y < 13.0:
    Alpha[g][h] = AlfaSacSil

elif x == 13.0 and y > 7.0 and y < 13.0:
    Alpha[g][h] = AlfaSacSil

elif y == 7.0 and x > 7.0 and x < 13.0:
    Alpha[g][h] = AlfaSacSil
```

```
            elif y == 13.0 and x > 7.0 and x < 13.0:
                Alpha[g][h] = AlfaSacSil

            else:
                Alpha[g][h] = AlfaSac

Q = []

for t in range(malha.pointsNumber):
    px = malha.X[t]
    py = malha.Y[t]

    if px > 7.0 and px < 13.0 and py > 7.0 and py < 13.0:
        Q.append(Qint/RhoCv_Silicio)

    else:
        Q.append(0.0)

for e in range(malha.elementsNumber):

        # Listagem dos indices dos nós que compoem o elemento:
        v = malha.IEN[e]

        # Cálculo do determinante:
        det =  malha.X[v[2]]*( malha.Y[v[0]] - malha.Y[v[1]])
        det += malha.X[v[0]]*( malha.Y[v[1]] - malha.Y[v[2]])
        det += malha.X[v[1]]*(-malha.Y[v[0]] + malha.Y[v[2]])

        #Cálculo da área:
        area = det/2.0

        #Cálculo dos elementos das matrizes
```

```
ai = malha.X[v[1]]*malha.Y[v[2]] - malha.X[v[2]]*malha.Y[v[1]]
aj = malha.X[v[2]]*malha.Y[v[0]] - malha.X[v[0]]*malha.Y[v[2]]
ak = malha.X[v[0]]*malha.Y[v[1]] - malha.X[v[1]]*malha.Y[v[0]]

bi = malha.Y[v[1]] - malha.Y[v[2]]
bj = malha.Y[v[2]] - malha.Y[v[0]]
bk = malha.Y[v[0]] - malha.Y[v[1]]

ci = malha.X[v[2]] - malha.X[v[1]]
cj = malha.X[v[0]] - malha.X[v[2]]
ck = malha.X[v[1]] - malha.X[v[0]]

# Cálculo da matriz de massa
Me = mult-por-escalar(area/12.0, [[2.0, 1.0, 1.0], [1.0, 2.0, 1.0],
                                  [1.0, 1.0, 2.0]])
# Cálculo da Matriz B
B = mult-por-escalar((1.0/(2.0*area)), [[b1, b2, b3], [c1, c2, c3]])

# Matriz B Transposta
BT = [list(line) for line in zip(*B)]

# Cálculo da matriz de rigidez
Ke = area*np.dot(BT, B)

for i in range(3):
    ii = malha.IEN[e][i]
    for j in range(3):
        jj = malha.IEN[e][j]

        K[ii][jj]  += Alpha[e][i]*Ke[i][j]
        M[ii][jj]  += Me[i][j]
```

```
# Matriz nula que conterá as condições de contorno:
C = [0 for i in range(malha.nosNumber)]

# Matriz é preenchida:
for i in range(malha.nosNumber):
    cc = malha.nos[i].cc
    if(cc == "Bottom"):
        C[i] = T_Bottom
    elif(cc == "Top"):
        C[i] = T_Top
   elif(cc == "Left"):
        C[i] = T_Left
    elif(cc == "Right"):
        C[i] = T_Right
```

## A.3 Parte 3

```
# Matriz temperatura inicial
T2 = []

# Matriz temperatura inicial
T1 = []

# Definindo temperaturas inciais
for k in range(len(malha.X)):
    cc = malha.nos[k].cc
    if(cc == "Bottom"):
        T1.append(T_Bottom)
    elif(cc == "Top"):
        T1.append(T_Top)
```

```
        elif(cc == "Left"):
            T1.append(T_Left)
        elif(cc == "Right"):
            T1.append(T_Right)
        else:
            T1.append(T_inicial)


    A = np.add(mult_por_escalar(1/dt,M), mult_por_escalar(1/2.0,K))

    B1 = np.add(mult_por_escalar(1/dt,M), mult_por_escalar(-
1/2.0,K))

    B2 = np.matmul(B1,T1)

    B3 = np.add(B2, np.matmul(M,Q))

    B = np.add(B3,C)

    for k in range(nint+1):
        if k !=0:

            B2 = np.matmul(B1,T1)

            B3 = np.add(B2, np.matmul(M,Q))

            B = np.add(B3,C)

        T2 = np.linalg.solve(A,B)

        T1 = T2
```