Capítulo 1

Elaboração

Neste capítulo é apresentado a elaboração do programa, constituído pelo desenvolvimento teórico, modelagem numérica, identificação de pacotes e algoritmos adicionais relacionados ao *software*.

1.1 Análise de domínio

A análise de domínio, como parte da elaboração, tem o objetivo de entender e delimitar conceitos fundamentais, sob o qual o software é construído[?].

O presente trabalho pode ser dividido em quatro conceitos fundamentais:

1. Transferência de calor:

   Transferência de calor é uma subárea da termodinâmica, a qual é área da física. É responsável por tratar das três formas possíveis de transferência de calor: condução, convecção e radiação. Este projeto trata especificamente da condução de calor.

   A condução só pode ocorrer em meio material (fluidos ou sólidos), e sem que haja movi emento do próprio meio, característica da convecção [?].

2. Modelagem numérica:

   Métodos numéricos são algoritmos desenvolvidos com ajuda da matemática para resolver problemas complexos da natureza. São utilizados quando uma solução analítica é difícil de ser obtida, ou com condições de contorno complexas.

3. Programação orientada ao objeto com C++:

   O paradigma orientado ao objeto é um dos principais paradigmas da programação, utilizado especialmente na construção de grandes *softwares* devido à portabilidade, organização e delimitação de assuntos. C++ é uma das linguagens mais utilizadas atualmente, por ser mais rápida com muito suporte e por permitir a orientação ao objeto.

4. Renderização 3D:

   Renderização 3D é uma área com grande ascensão na indústria de jogos e softwares de engenharia profissional, torna prático que usuários consigam visualizar o objeto sob qualquer ótica. É necessário a utilização de vários conceitos da álgebra linear.

1.2 Formulação

1.2.1 Formulação teórica

A taxa de transferência de calor foi modelado empiricamente por Jean B. J. Fourier em 1822 ([?]). Posteriormente a teoria foi aprimorada até chegar na equação geral da difusão de calor (??). O desenvolvimento teórico para chegar nesta equação, pode ser acompanhado detalhadamente no [?].

Portanto, a seguir é apresentada a equação geral da difusão de calor em meios tridimensionais cartesianos:

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right)=\rho c_p\frac{\partial T}{\partial t} \tag{1.1}$$

Onde ρ é a massa específica em $[kg/m^3]$, c_p é a capacidade térmica em $[J/(kg\cdot K)]$, k é a condutividade térmica em $[W/(m\cdot K)]$.

Para resolver a equação geral da difusão térmica, será utilizado o método implícito de diferenças finitas BTCS, com malha em formato bloco centrado.

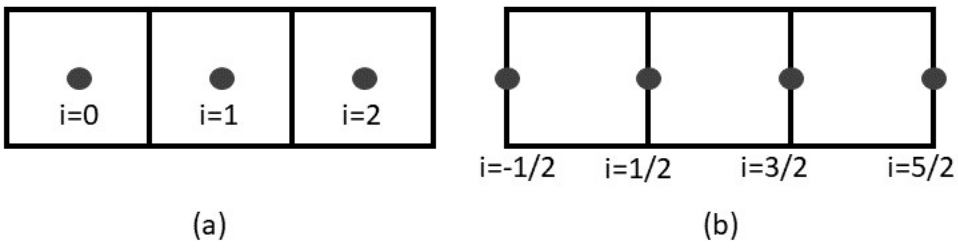

Figura 1.1: Tipos de malha, (a) bloco-centrado e (b) ponto-distribuído.

Conforme a Figura 1.1, existem dois tipos principais de malha: bloco-centrado, onde os pontos analisados estão nos centros de cada bloco, e ponto-distribuído, onde os pontos analisados estão nas fronteiras de cada bloco.

Com esses conceitos em mente, a equação geral é modelada por diferenças finitas, mantendo a condutividade térmica dentro da derivada espacial. Inicialmente, será modelado somente a derivada externa:

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)=\frac{\left(k\frac{\partial T}{\partial x}\right)_{i-\frac{1}{2},j,k}-\left(k\frac{\partial T}{\partial x}\right)_{i+\frac{1}{2},j,k}}{\Delta x} \tag{1.2}$$

Modelando as derivadas internas:

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) = \frac{k_{i-\frac{1}{2},j,k}\left(\frac{T_{i-1,j,k}-T_{i,j,k}}{\Delta x}\right) - k_{i+\frac{1}{2},j,k}\left(\frac{T_{i,j,k}-T_{i+1,j,k}}{\Delta x}\right)}{\Delta x} \tag{1.3}$$

Com um pouco de álgebra:

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) = \frac{k_{i-\frac{1}{2},j,k}\left(T_{i-1,j,k}-T_{i,j,k}\right) - k_{i+\frac{1}{2},j,k}\left(T_{i,j,k}-T_{i+1,j,k}\right)}{\Delta x^2} \tag{1.4}$$

Chegando na modelagem final para a derivada espacial ao longo do x:

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) = \frac{k_{i-\frac{1}{2},j,k}T_{i-1,j,k} - \left(k_{i-\frac{1}{2},j,k}+k_{i+\frac{1}{2},j,k}\right)T_{i,j,k} + k_{i+\frac{1}{2},j,k}T_{i+1,j,k}}{\Delta x^2} \tag{1.5}$$

Como as outras dimensões são simétricas:

$$\frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right) = \frac{k_{i,j-\frac{1}{2},k}T_{i,j-1,k} - \left(k_{i,j-\frac{1}{2},k}+k_{i,j+\frac{1}{2},k}\right)T_{i,j,k} + k_{i,j+\frac{1}{2},k}T_{i,j+1,k}}{\Delta y^2} \tag{1.6}$$

$$\frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right) = \frac{k_{i,j,k-\frac{1}{2}}T_{i,j,k-1} - \left(k_{i,j,k-\frac{1}{2}}+k_{i,j,k+\frac{1}{2}}\right)T_{i,j,k} + k_{i,j,k+\frac{1}{2}}T_{i,j,k+1}}{\Delta z^2} \tag{1.7}$$

A derivada temporal é atrasada no tempo:

$$\frac{\partial T}{\partial t} = \frac{T_{i,j}^{n+1} - T_{i,j}^{n}}{\Delta t} \tag{1.8}$$

Substituindo as diferenças finitas na equação geral:

$$\begin{gathered}\frac{k_{i-\frac{1}{2},j,k}T_{i-1,j,k} - \left(k_{i-\frac{1}{2},j,k}+k_{i+\frac{1}{2},j,k}\right)T_{i,j,k} + k_{i+\frac{1}{2},j,k}T_{i+1,j,k}}{\Delta x^2} + \\ \frac{k_{i,j-\frac{1}{2},k}T_{i,j-1,k} - \left(k_{i,j-\frac{1}{2},k}+k_{i,j+\frac{1}{2},k}\right)T_{i,j,k} + k_{i,j+\frac{1}{2},k}T_{i,j+1,k}}{\Delta y^2} + \\ \frac{k_{i,j,k-\frac{1}{2}}T_{i,j,k-1} - \left(k_{i,j,k-\frac{1}{2}}+k_{i,j,k+\frac{1}{2}}\right)T_{i,j,k} + k_{i,j,k+\frac{1}{2}}T_{i,j,k+1}}{\Delta z^2} = \\ \frac{T_{i,j}^{n+1} - T_{i,j}^{n}}{\Delta t}\end{gathered} \tag{1.9}$$

Onde a malha é homogênea na superfície, mas não entre os perfis, ou seja, $\Delta x = \Delta y \neq \Delta z$. Substituindo:

$$\begin{array}{c}\frac{k_{i-\frac{1}{2},j,k}T_{i-1,j,k}-\left(k_{i-\frac{1}{2},j,k}+k_{i+\frac{1}{2},j,k}\right)T_{i,j,k}+k_{i+\frac{1}{2},j,k}T_{i+1,j,k}}{\Delta x^2}+\\ \frac{k_{i,j-\frac{1}{2},k}T_{i,j-1,k}-\left(k_{i,j-\frac{1}{2},k}+k_{i,j+\frac{1}{2},k}\right)T_{i,j,k}+k_{i,j+\frac{1}{2},k}T_{i,j+1,k}}{\Delta x^2}+\\ \frac{k_{i,j,k-\frac{1}{2}}T_{i,j,k-1}-\left(k_{i,j,k-\frac{1}{2}}+k_{i,j,k+\frac{1}{2}}\right)T_{i,j,k}+k_{i,j,k+\frac{1}{2}}T_{i,j,k+1}}{\Delta z^2}=\\ c_p\rho\frac{T_{i,j}^{n+1}-T_{i,j}^{n}}{\Delta t}\end{array} \tag{1.10}$$

Multiplicando pelo múltiplo comum:

$$\begin{array}{c}\Delta z^2\left(k_{i-\frac{1}{2},j,k}^{n+1}T_{i-1,j,k}^{n+1}-\left(k_{i-\frac{1}{2},j,k}^{n+1}+k_{i+\frac{1}{2},j,k}^{n+1}\right)T_{i,j,k}^{n+1}+k_{i+\frac{1}{2},j,k}^{n+1}T_{i+1,j,k}^{n+1}\right)+\\ \Delta z^2\left(k_{i,j-\frac{1}{2},k}^{n+1}T_{i,j-1,k}^{n+1}-\left(k_{i,j-\frac{1}{2},k}^{n+1}+k_{i,j+\frac{1}{2},k}^{n+1}\right)T_{i,j,k}^{n+1}+k_{i,j+\frac{1}{2},k}^{n+1}T_{i,j+1,k}^{n+1}\right)+\\ \Delta x^2\left(k_{i,j,k-\frac{1}{2}}^{n+1}T_{i,j,k-1}^{n+1}-\left(k_{i,j,k-\frac{1}{2}}^{n+1}+k_{i,j,k+\frac{1}{2}}^{n+1}\right)T_{i,j,k}^{n+1}+k_{i,j,k+\frac{1}{2}}^{n+1}T_{i,j,k+1}^{n+1}\right)=\\ \frac{\Delta z^2\Delta x^2c_p\rho}{\Delta t}T_{i,j}^{n+1}-\frac{\Delta z^2\Delta x^2c_p\rho}{\Delta t}T_{i,j}^{n}\end{array} \tag{1.11}$$

Como a equação acima é complexa para ser reorganizada e resolvida por equações matriciais, será utilizado aproximações implícitas para resolver esse problema, para calcular a iteração $\nu+1$, será utilizado:

$$\begin{array}{c}\overset{\nu+1}{T_{i,j}^{n+1}}=\\ C_1\frac{\Delta z^2\Delta x^2c_p\rho}{\Delta t}\overset{\nu}{T_{i,j}^{n}}+\\ C_1\Delta z^2\left(k_{i-\frac{1}{2},j,k}^{\nu\,n+1}\overset{\nu}{T_{i-1,j,k}^{n+1}}+k_{i+\frac{1}{2},j,k}^{\nu\,n+1}\overset{\nu}{T_{i+1,j,k}^{n+1}}\right)+\\ C_1\Delta z^2\left(k_{i,j-\frac{1}{2},k}^{\nu\,n+1}\overset{\nu}{T_{i,j-1,k}^{n+1}}+k_{i,j+\frac{1}{2},k}^{\nu\,n+1}\overset{\nu}{T_{i,j+1,k}^{n+1}}\right)+\\ C_1\Delta x^2\left(k_{i,j,k-\frac{1}{2}}^{\nu\,n+1}\overset{\nu}{T_{i,j,k-1}^{n+1}}+k_{i,j,k+\frac{1}{2}}^{\nu\,n+1}\overset{\nu}{T_{i,j,k+1}^{n+1}}\right)\end{array} \tag{1.12}$$

Onde C_1é a constante:

$$\frac{1}{C_1}=\begin{array}{l}\frac{\Delta z^2\Delta x^2c_p\rho}{\Delta t}+\quad\Delta z^2\left(k_{i-\frac{1}{2},j,k}^{\nu\,n+1}+k_{i+\frac{1}{2},j,k}^{\nu\,n+1}\right)+\\ \qquad\Delta z^2\left(k_{i,j-\frac{1}{2},k}^{\nu\,n+1}+k_{i,j+\frac{1}{2},k}^{\nu\,n+1}\right)\quad+\Delta x^2\left(k_{i,j,k-\frac{1}{2}}^{\nu\,n+1}+k_{i,j,k+\frac{1}{2}}^{\nu\,n+1}\right)\end{array} \tag{1.13}$$

Agora, é necessário definir o cálculo das condutividades térmicas nas fronteiras. Para isso, será feito um análogo com a permeabilidade de rochas em série [**?**], mas utilizando as equações de calor:

$$q_x=-kA\frac{dT}{dx}=-\frac{kA}{L}\Delta T \tag{1.14}$$

Isolando a diferença de temperatura:

$$\Delta T = -\frac{Lq_x}{kA} \tag{1.15}$$

A Figura ?? mostra um caso de condutividades térmicas em série. O calor (q) que entra no sistema, é igual ao que sai. E a diferença de temperatura entre a esquerda (0) e a direita (2), é soma das diferenças nesse meio, ou seja:

$$T_0 - T_2 = (T_0 - T_1) + (T_1 - T_2) \tag{1.16}$$

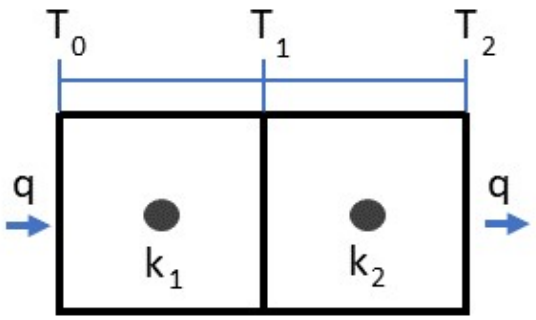

Figura 1.2: Representação de condutividade térmica em série.

Logo,

$$\Delta T_t = \Delta T_1 + \Delta T_2 \tag{1.17}$$

Onde as taxas de transferência de calor são:

$$-\frac{2Lq}{k_r A} = -\frac{Lq}{k_1 A} - \frac{Lq}{k_2 A} \tag{1.18}$$

Com alguns ajustes algébricos:

$$\frac{2}{k_r} = \frac{1}{k_1} + \frac{1}{k_2} \tag{1.19}$$

Ou, simplesmente:

$$k_r = \frac{2k_1 k_2}{k_1 + k_2} \tag{1.20}$$

É importante analisar a célula computacional, ou a região que é observada quando um ponto é calculado. Para isso, é apresentada a Figura ??, onde a esquerda é o tempo anterior T=n-1, e o ponto calculado está no tempo presente T=n. Para calcular o ponto vermelho, é utilizado o mesmo ponto, mas no tempo anterior, e uma célula em cada sentido.

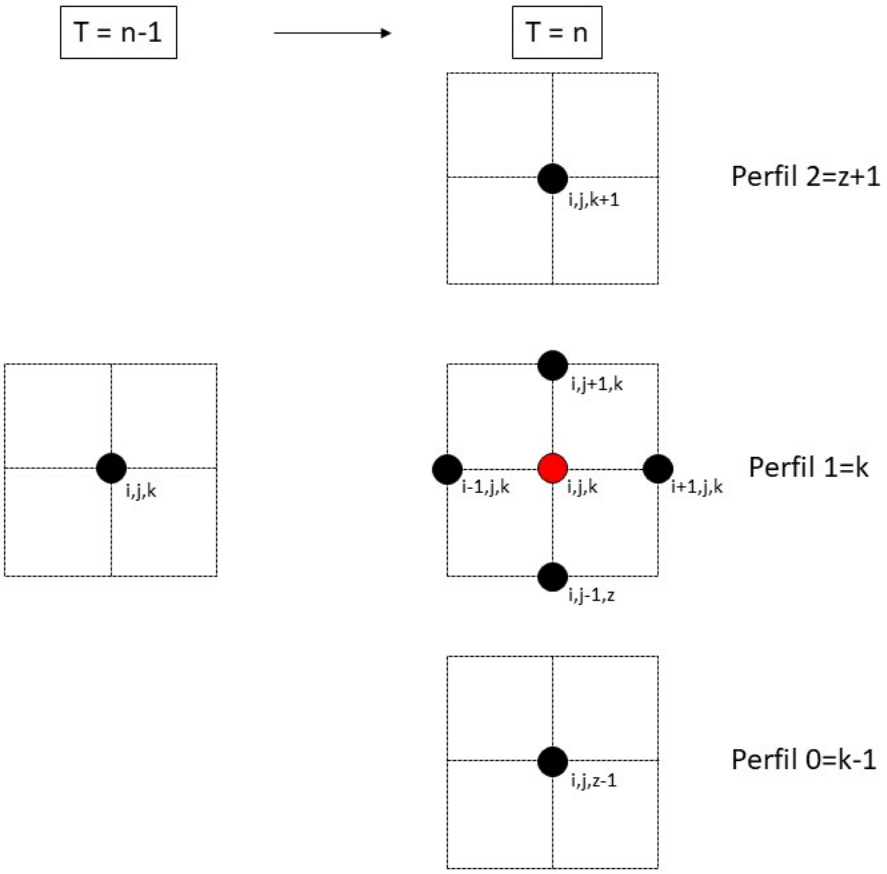

Figura 1.3: Malha utilizada para calcular um ponto de temperatura, cada ponto é o centro dos blocos.

A seguir, é resolvida a última etapa da modelagem do problema, a modelagem da condição de fronteira de Neumann

Condição de fronteira

Condição de fronteira, como o próprio nome diz, é a condição onde estão os limites materiais do objeto. Nessa região, a condução térmica é diferente do interior do objeto, pois não poderá conduzir calor em todos os sentidos, mas só onde existir material adjacente.

A condição de contorno de Neumann define que, nessa região, o objeto não troca calor com o meio externo. Na Figura **??**, a fronteira está na reta vermelha e, como o método modelado utilizaria o ponto à esquerda, é necessário encontrar um substituto real para esse termo.

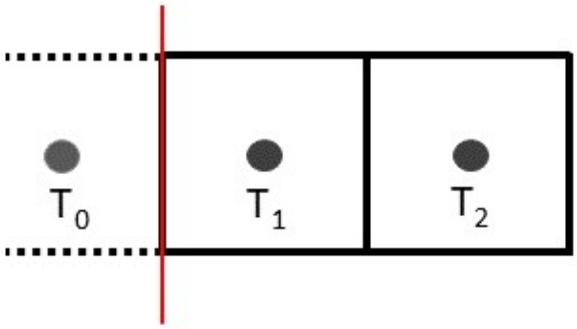

Figura 1.4: Análise da fronteira de Neumann.

Por isso, é importante modelar a condição de contorno, que pode ser modelada com diferenças finitas centradas como:

$$k\frac{\partial T}{\partial x}_{i-\frac{1}{2},j,k} = 0\frac{T^{n+1}_{i,j,k} - T^{n+1}_{i-1,j,k}}{\Delta x} = 0 \tag{1.21}$$

A equação acima possui duas soluções:

$$\begin{cases} k_{i-\frac{1}{2},j,k} & = 0 \\ \frac{\partial T}{\partial x}_{i-\frac{1}{2},j,k} & = 0 \end{cases} \tag{1.22}$$

Resolvendo a linha de baixo:

$$\frac{\partial T}{\partial x}_{i-\frac{1}{2},j,k} = \frac{T^{n+1}_{i,j,k} - T^{n+1}_{i-1,j,k}}{\Delta x} = 0 \tag{1.23}$$

$$T^{n+1}_{i-1,j,k} = T^{n+1}_{i,j,k} \tag{1.24}$$

Todas as seis fronteiras são simétricas, então:

$$\begin{array}{l} T^{n+1}_{i-1,j,k} = T^{n+1}_{i,j,k} \\ T^{n+1}_{i+1,j,k} = T^{n+1}_{i,j,k} \\ T^{n+1}_{i,j-1,k} = T^{n+1}_{i,j,k} \\ T^{n+1}_{i,j+1,k} = T^{n+1}_{i,j,k} \\ T^{n+1}_{i,j,k-1} = T^{n+1}_{i,j,k} \\ T^{n+1}_{i,j,k+1} = T^{n+1}_{i,j,k} \end{array} \tag{1.25}$$

As equações encontradas na Eq. **??** dizem que, se existir uma fronteira, a temperatura inexistente deve ser substituída pela temperatura do próprio ponto. Ou, como mostrado na Eq. **??**, a condutividade térmica na fronteira deve ser zero. Quaisquer dentre as duas opções resolvem o problema da condição de contorno de Neumann.

Demonstrações

Nesta parte, será analisado dois casos para validar as modelagens. Primeiro, será utilizado um objeto formado por uma única célula isolada no espaço. Posteriormente, será analisado o caso do objeto constituído por um único material, mas bidimensional.

Começando pelo objeto de única célula, todas as suas seis fronteiras devem ser aplicadas as condições de contorno de Neumann. Fisicamente, é esperado que o objeto, por estar isolado, não varie com sua temperatura interna ao longo do tempo. Então, partindo da equação geral:

$$\begin{gathered}\overset{\nu+1}{T}{}^{n+1}_{i,j} = \\ C_1\frac{\Delta z^2\Delta x^2 c_p\rho}{\Delta t}\overset{\nu}{T}{}^{n}_{i,j} + \\ C_1\Delta z^2\left(\overset{\nu}{k}{}^{n+1}_{i-\frac{1}{2},j,k}\overset{\nu}{T}{}^{n+1}_{i-1,j,k} + \overset{\nu}{k}{}^{n+1}_{i+\frac{1}{2},j,k}\overset{\nu}{T}{}^{n+1}_{i+1,j,k}\right) + \\ C_1\Delta z^2\left(\overset{\nu}{k}{}^{n+1}_{i,j-\frac{1}{2},k}\overset{\nu}{T}{}^{n+1}_{i,j-1,k} + \overset{\nu}{k}{}^{n+1}_{i,j+\frac{1}{2},k}\overset{\nu}{T}{}^{n+1}_{i,j+1,k}\right) + \\ C_1\Delta x^2\left(\overset{\nu}{k}{}^{n+1}_{i,j,k-\frac{1}{2}}\overset{\nu}{T}{}^{n+1}_{i,j,k-1} + \overset{\nu}{k}{}^{n+1}_{i,j,k+\frac{1}{2}}\overset{\nu}{T}{}^{n+1}_{i,j,k+1}\right)\end{gathered} \tag{1.26}$$

$$\frac{1}{C_1} = \begin{array}{ll}\frac{\Delta z^2\Delta x^2 c_p\rho}{\Delta t} + & \Delta z^2\left(\overset{\nu}{k}{}^{n+1}_{i-\frac{1}{2},j,k} + \overset{\nu}{k}{}^{n+1}_{i+\frac{1}{2},j,k}\right) + \\ & \Delta z^2\left(\overset{\nu}{k}{}^{n+1}_{i,j-\frac{1}{2},k} + \overset{\nu}{k}{}^{n+1}_{i,j+\frac{1}{2},k}\right) \quad + \Delta x^2\left(\overset{\nu}{k}{}^{n+1}_{i,j,k-\frac{1}{2}} + \overset{\nu}{k}{}^{n+1}_{i,j,k+\frac{1}{2}}\right)\end{array} \tag{1.27}$$

Mas, como demonstrado na Eq. **??**, quando houver fronteira, a condutividade térmica na fronteira é zero:

$$\begin{gathered}\overset{\nu+1}{T}{}^{n+1}_{i,j} = \\ C_1\frac{\Delta z^2\Delta x^2 c_p\rho}{\Delta t}\overset{\nu}{T}{}^{n}_{i,j} + \\ C_1\Delta z^2\left(\cancelto{0}{\overset{\nu}{k}{}^{n+1}_{i-\frac{1}{2},j,k}}\overset{\nu}{T}{}^{n+1}_{i-1,j,k} + \cancelto{0}{\overset{\nu}{k}{}^{n+1}_{i+\frac{1}{2},j,k}}\overset{\nu}{T}{}^{n+1}_{i+1,j,k}\right) + \\ C_1\Delta z^2\left(\cancelto{0}{\overset{\nu}{k}{}^{n+1}_{i,j-\frac{1}{2},k}}\overset{\nu}{T}{}^{n+1}_{i,j-1,k} + \cancelto{0}{\overset{\nu}{k}{}^{n+1}_{i,j+\frac{1}{2},k}}\overset{\nu}{T}{}^{n+1}_{i,j+1,k}\right) + \\ C_1\Delta x^2\left(\cancelto{0}{\overset{\nu}{k}{}^{n+1}_{i,j,k-\frac{1}{2}}}\overset{\nu}{T}{}^{n+1}_{i,j,k-1} + \cancelto{0}{\overset{\nu}{k}{}^{n+1}_{i,j,k+\frac{1}{2}}}\overset{\nu}{T}{}^{n+1}_{i,j,k+1}\right)\end{gathered} \tag{1.28}$$

$$\frac{1}{C_1} = \begin{array}{l} \frac{\Delta z^2 \Delta x^2 c_p \rho}{\Delta t} + \quad \Delta z^2 \left(\cancel{k^{n+1}_{i-\frac{1}{2},j,k}}^{\nu\ 0} + \cancel{k^{n+1}_{i+\frac{1}{2},j,k}}^{\nu\ 0} \right) + \\ \Delta z^2 \left(\cancel{k^{n+1}_{i,j-\frac{1}{2},k}}^{\nu\ 0} + \cancel{k^{n+1}_{i,j+\frac{1}{2},k}}^{\nu\ 0} \right) \quad + \Delta x^2 \left(\cancel{k^{n+1}_{i,j,k-\frac{1}{2}}}^{\nu\ 0} + \cancel{k^{n+1}_{i,j,k+\frac{1}{2}}}^{\nu\ 0} \right) \end{array} \tag{1.29}$$

Resultando em:

$$\overset{\nu+1}{T}{}^{n+1}_{i,j} = C_1 \frac{\Delta z^2 \Delta x^2 c_p \rho}{\Delta t} \overset{\nu}{T}{}^{n}_{i,j} \tag{1.30}$$

$$\frac{1}{C_1} = \frac{\Delta z^2 \Delta x^2 c_p \rho}{\Delta t} \tag{1.31}$$

Logo:

$$\overset{\nu+1}{T}{}^{n+1}_{i,j} = \frac{\Delta t}{\Delta z^2 \Delta x^2 c_p \rho} \frac{\Delta z^2 \Delta x^2 c_p \rho}{\Delta t} \overset{\nu}{T}{}^{n}_{i,j} \tag{1.32}$$

$$\overset{\nu+1}{T}{}^{n+1}_{i,j} = \overset{\nu}{T}{}^{n}_{i,j} \tag{1.33}$$

Mostrando que a temperatura não varia com o tempo.

Para a segunda demonstração, onde o objeto é constituído pelo mesmo material e mesma condutividade térmica, mas somente bidimensional. Partindo da equação geral:

$$\begin{array}{c} T^{n+1}_{i,j} = \\ C_1 \frac{\Delta z^2 \Delta x^2 c_p \rho}{\Delta t} T^{n}_{i,j} + \\ C_1 \Delta z^2 \left(k^{n+1}_{i-\frac{1}{2},j,k} T^{n+1}_{i-1,j,k} + k^{n+1}_{i+\frac{1}{2},j,k} T^{n+1}_{i+1,j,k} \right) + \\ C_1 \Delta z^2 \left(k^{n+1}_{i,j-\frac{1}{2},k} T^{n+1}_{i,j-1,k} + k^{n+1}_{i,j+\frac{1}{2},k} T^{n+1}_{i,j+1,k} \right) + \\ C_1 \Delta x^2 \left(k^{n+1}_{i,j,k-\frac{1}{2}} T^{n+1}_{i,j,k-1} + k^{n+1}_{i,j,k+\frac{1}{2}} T^{n+1}_{i,j,k+1} \right) \end{array} \tag{1.34}$$

$$\frac{1}{C_1} = \begin{array}{l} \frac{\Delta z^2 \Delta x^2 c_p \rho}{\Delta t} + \quad \Delta z^2 \left(k^{n+1}_{i-\frac{1}{2},j,k} + k^{n+1}_{i+\frac{1}{2},j,k} \right) + \\ \Delta z^2 \left(k^{n+1}_{i,j-\frac{1}{2},k} + k^{n+1}_{i,j+\frac{1}{2},k} \right) \quad + \Delta x^2 \left(k^{n+1}_{i,j,k-\frac{1}{2}} + k^{n+1}_{i,j,k+\frac{1}{2}} \right) \end{array} \tag{1.35}$$

Com a substituição de todas as condutividades térmicas nas interfaces por k, e simplificando para bidimensional:

$$\begin{array}{c} T^{n+1}_{i,j} = \\ C_1 \frac{\Delta z^2 \Delta x^2 c_p \rho}{\Delta t} T^{n}_{i,j} + \\ C_1 \Delta z^2 \left(k T^{n+1}_{i-1,j,k} + k T^{n+1}_{i+1,j,k} \right) + \\ C_1 \Delta z^2 \left(k T^{n+1}_{i,j-1,k} + k T^{n+1}_{i,j+1,k} \right) \end{array} \tag{1.36}$$

$$\frac{1}{C_1} = \frac{\Delta z^2 \Delta x^2 c_p \rho}{\Delta t} + \Delta z^2 (k + k) + \Delta z^2 (k + k) \tag{1.37}$$

Com alguns ajustes:

$$\begin{gathered}\frac{1}{C_1}T_{i,j}^{n+1} = \\ \frac{\Delta x^2 c_p \rho}{\Delta t}T_{i,j}^{n} + \\ k\left(T_{i-1,j,k}^{n+1} + T_{i+1,j,k}^{n+1}\right) + \\ k\left(T_{i,j-1,k}^{n+1} + T_{i,j+1,k}^{n+1}\right)\end{gathered} \tag{1.38}$$

$$\frac{1}{C_1} = \frac{\Delta x^2 c_p \rho}{\Delta t} + 2k + 2k \tag{1.39}$$

Logo:

$$\begin{gathered}\left(\frac{\Delta x^2 c_p \rho}{k\Delta t} + 2 + 2\right)T_{i,j}^{n+1} = \\ \frac{\Delta x^2 c_p \rho}{k\Delta t}T_{i,j}^{n} + \\ \left(T_{i-1,j,k}^{n+1} + T_{i+1,j,k}^{n+1}\right) + \\ \left(T_{i,j-1,k}^{n+1} + T_{i,j+1,k}^{n+1}\right)\end{gathered} \tag{1.40}$$

Chegando na equação:

$$\begin{gathered}\frac{c_p \rho}{k}\frac{T_{i,j}^{n+1} - T_{i,j}^{n}}{\Delta t} = \\ \frac{T_{i-1,j,k}^{n+1} - 2T_{i,j}^{n+1} + T_{i+1,j,k}^{n+1}}{\Delta x^2} + \\ \frac{T_{i,j-1,k}^{n+1} - 2T_{i,j}^{n+1} + T_{i,j+1,k}^{n+1}}{\Delta x^2}\end{gathered} \tag{1.41}$$

E essa é a igual implícita discretizada para um sistema homogêneo bidimensional, conforme [**?**].

1.2.2 Condutividade térmica variável

A condutividade térmica (k), no projeto desenvolvido, pode variar com o espaço, pelo objeto ser constituído por mais de um material, com condutividade térmica distinta. Mas também pode variar com a temperatura e, consequentemente, com o tempo.

Será fornecido aos usuários, três opções para calcular essas condutividades térmicas:

- Valores constantes;
- Correlação;
- Interpolação.

Para o primeiro caso, como o nome diz, a condutividade térmica será constante ao longo de todo o tempo, variando somente com a posição.

No segundo caso, será utilizado os modelos de correlação do *handbook Thermophysical Properties* [**?**]. O modelo proposto, é calculado, em geral, como:

$$k = C_0 + C_1 T - C_2 T^2 \tag{1.42}$$

Onde C_0, C_1e C_2 são constantes da correlação, específico para cada material.

O terceiro caso, é o cálculo pela interpolação e, como o nome diz, calcula a condutividade térmica pela interpolação linear entre valores obtidos em laboratório.

Assim, é finalizada as demonstrações físico-numérica do problema da difusão térmica. A partir de agora, será elaborado o paralelismo/multithreading, e a renderização 3D.

1.2.3 Paralelismos/multi-thread

Os chips de processadores atuais, são constituídos por vários processadores menores, o que permite que um mesmo processador consiga realizar tarefas distintas. A ideia é separar tarefas distintas, para que um processador não fique travado em uma única tarefa.

Uma analogia para melhorar a explicação é a dos estudantes. Uma sala cheia de estudantes, recebe uma tarefa de resolver uma lista de exercícios. Se todos os exercícios forem resolvidas por um único aluno, levará muito tempo para terminar a tarefa (caso sem paralelismo). Se os alunos dividirem as tarefas entre si, ela será resolvida muito mais rapidamente.

Similarmente ao cenário acima, foram implementados três casos de paralelismo, por questão de didática.

1. Sem paralelismo: uma única thread do processador resolve todos os cálculos.

2. Paralelismo por grid: cada thread resolve uma camada do objeto. Possuí certa otimização em relação ao anterior, mas, se só existir objeto em uma camada, outras threads ficam ociosas.

3. Paralelismo total: todas as threads do processador resolvem os cálculos de todo o objeto 3D, intercalando a posição com base no número da thread.

A figura ilustra melhor esses casos

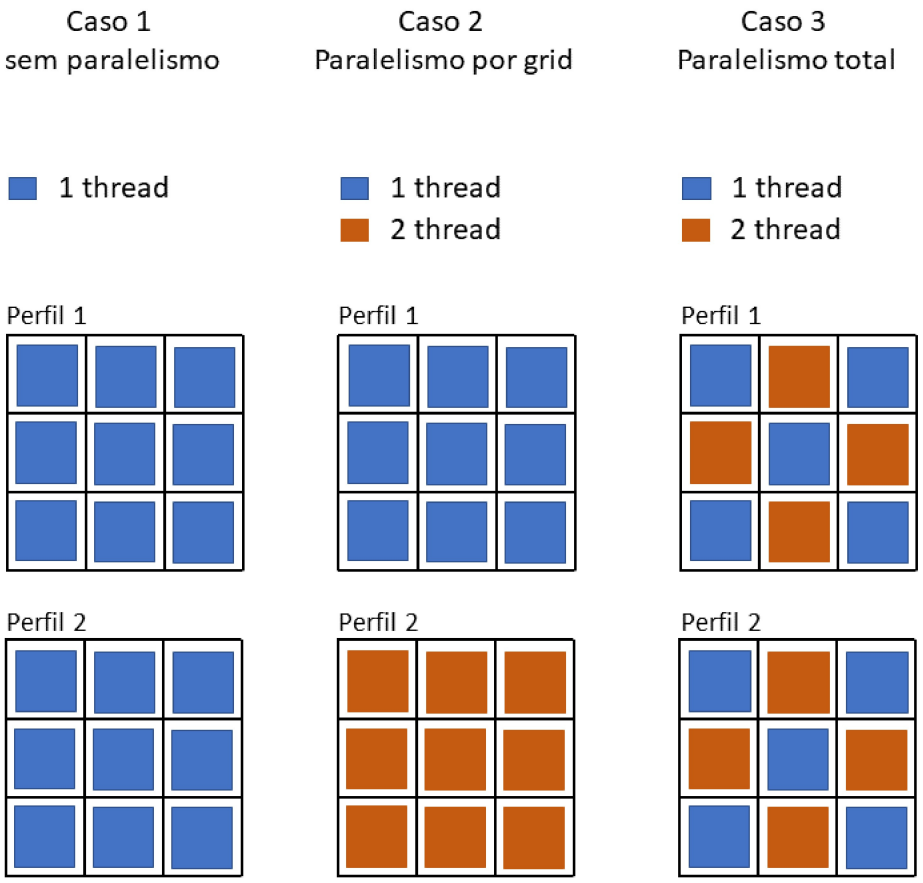

Figura 1.5: Figura ilustrando os três casos de paralelismo implementados para duas camadas com 9 células cada, e um processador com duas threads.

O algoritmo utilizado para o caso 3 é:
for(int i = NUM_THREAD; i < size; i+=MAX_THREADS)

Esse algoritmo diz que a thread "i", deverá começar a resolver as equações na posição "i". Quando finalizar, deve pular para a posição "i + números de threads".

1.2.4 Renderização 3D

Após o usuário desenhar algum objeto no software, pode ser de interesse observar como seria em renderização 3D. Portanto, é implementado algoritmos para essa renderização.

Inicialmente, é interessante observar a complexidade da renderização: um objeto 3D deve ser apresentado em uma tela 2D, com a ilusão de ótica que é um objeto com profundidade. Por exemplo, um cubo com arestas de tamanho 1 cm é mostrado nos quatro casos da figura abaixo:

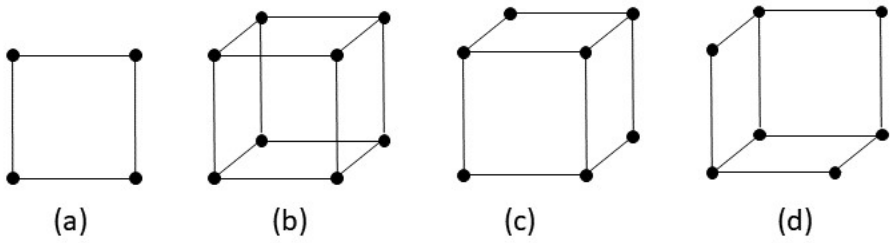

Figura 1.6: (a) Observador alinhado com uma das faces do cubo. (b) observador não está alinhado e não foram removidas arestas ocultas. O cérebro consegue interpretar que é um objeto 3D, mas fica confuso entre os casos (c) e (d).

Todos cantos do cubo da figura 1.2.4 estão na mesma posição, o que mudou foi o ângulo do observador com o objeto.

Portanto, tendo em mãos os pontos das arestas, é multiplicado esses vetores com a matriz de rotação do autor [?] mostrada em (1.43), a qual permite rotacionar qualquer ponto a partir dos três ângulos do observador.

$$R(\alpha,\beta,\gamma) = \begin{bmatrix} cos(\gamma)cos(\beta) & cos(\gamma)sin(\beta)sin(\alpha) - sin(\gamma)cos(\alpha) & cos(\gamma)sin(\beta)sin(\alpha) + sin(\gamma)cos(\alpha) \\ sin(\gamma)cos(\beta) & sin(\gamma)sin(\beta)sin(\alpha) + cos(\gamma)cos(\alpha) & sin(\gamma)sin(\beta)cos(\alpha) - cos(\gamma)sin(\alpha) \\ -sin(\beta) & cos(\beta) * sin(\alpha) & cos(\beta) * cos(\alpha) \end{bmatrix} \tag{1.43}$$

Ou seja, inicialmente, um cubo de aresta 3 cm, com uma margem de 1 cm, pode ser mostrado na tela (monitor) com os pontos do caso (a) da figura 1.2.4, onde o observador está alinhado com o objeto.

Conforme desejado, o objeto pode mudar seu ângulo com o observador, como no caso (b), onde os ângulos x e y passaram a ter o valor de 0.1 radianos. Não foi só os pontos de trás do cubo que aparecem (e mudaram seus valores), mas todos os pontos foram modificados.

Além disso, a aresta possui valor ligeiramente menor que 3, pois não é mais "de frente" que o observador está olhando, mas ligeiramente de lado. Mesmo que o objeto cubo tenha aresta de 3 centímetros.

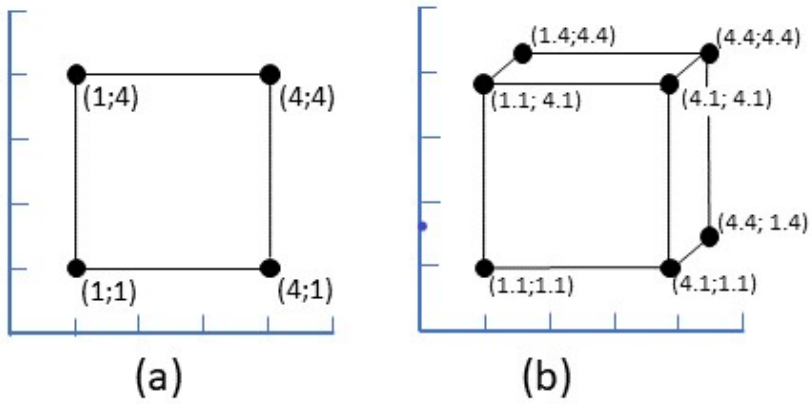

Figura 1.7: (a) o cubo está com ângulos nulos. (b) ângulo x e y estão com valor de 0.1 radianos.

Nos desenhos do simulador, cada pixel da figura, é uma célula com propriedades que serão calculadas, possuindo material, temperatura e volume. Como o usuário pode desenhar por pixel, a renderização 3D deve partir do princípio que cada pixel é um **potencial** objeto que deve ser renderizado.

Inicialmente, essa conclusão pode ficar vaga, pois todas as células do simulador devem ser renderizadas, mas, quando a simulação fica grande, é numeroso a quantidade de objetos renderizando ao mesmo tempo, tornando muito lenta a apresentação. Então algumas considerações são feitas no algoritmo para otimizar a renderização.

Primeiro, é desejável desenhar triângulos, e não pontos ou retas, por 2 motivos: geometria simples, possui normal e a biblioteca do Qt consegue desenhar e preencher a área com qualquer cor escolhida.

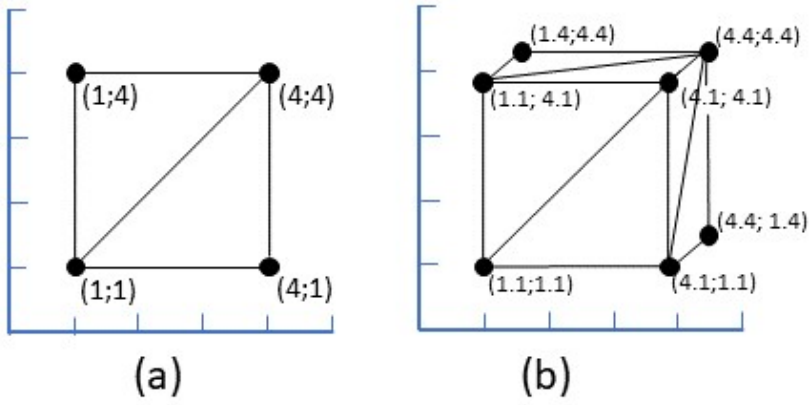

Figura 1.8: Mesmo desenho da figura anterior, mas agora renderizando a partir de triângulos.

O segundo motivo apresentado, é o mais importante dos três. Um triângulo possui três pontos, podendo ser reduzido para dois vetores (subtraindo o ponto de origem dos

outros dois pontos) e permite-se calcular a normal dessa superfície. Com isso, é obtido dos vetores $\boldsymbol{a} = \{a_1, a_2, a_3\}$ e o vetor $\boldsymbol{b} = \{b_1, b_2, b_3\}$ permitindo a realização do produto vetorial:

$$\boldsymbol{a} \times \boldsymbol{b} = \begin{bmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{bmatrix} \tag{1.44}$$

Ou simplesmente:

$$\boldsymbol{a} \times \boldsymbol{b} = (a_2 b_3 - a_3 b_2)\boldsymbol{i} - (a_1 b_3 - a_3 b_1)\boldsymbol{k} + (a_1 b_2 - a_2 b_1)\boldsymbol{j} \tag{1.45}$$

Utilizando a Regra da Mão Direita[1], é possível entender a utilidade da equação 1.45: o caso (a) da figura 1.2.4, mostra uma normal saindo do papel, em direção ao olho do leitor, logo, é um triângulo que deve ser renderizado. O caso (b) possui uma normal no sentido contrário, e não faz sentido desenhar esse triângulo, pois está na parte de trás do objeto.

1

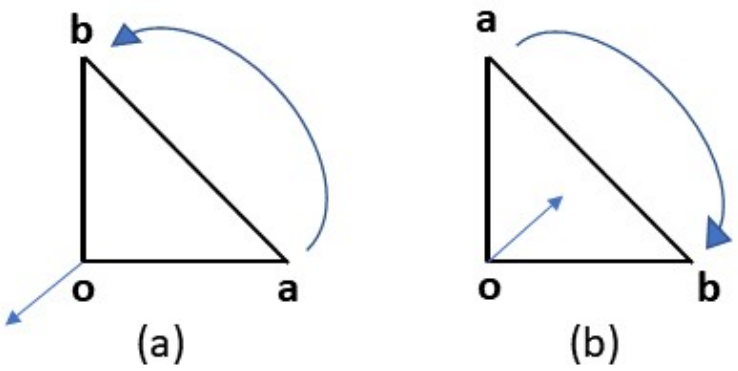

Figura 1.9: (a) mostra um caso onde a normal é na direção do leitor e (b) mostra um caso onde a normal é para dentro da folha.

Essa simples operação condicional do valor positivo/negativo de **j** da normal, reduz a renderização de objetos ocultos, e otimiza o software em duas vezes.

Uma outra condição implementada é a de avaliar se o objeto possui fronteira com outro objeto. Com isso, não é necessário renderizar 4 triângulos dessas duas superfícies em contato. Como estão em contato, não deve ser renderizada sob hipótese alguma.

Por fim, antes de renderizar os numerosos triângulos, eles são colocadas em ordem crescente com o valor de **j** da normal. Isso serve para ser desenhado primeiro o que está

[1]Para utilizar a Regra da Mão Direita, posicione o dedo polegar sobre o ponto **o**, e estique o indicador para o ponto **a**, agora, feche o indicador no sentido do ponto **b** (seta curvada mostra o sentido que a ponta do indicador deve realizar). No caso (a) da figura, o dedo polegar fica no sentido para fora do papel, e o caso (b), para dentro.

atrás, e depois desenhar o que está na frente, sobrescrevendo áreas que deveriam estar ocultas, evitando a criação de figuras confusas como no caso (b) da figura 1.2.4. É uma técnica lenta, mas de fácil implementação.

1.3 Identificação de pacotes – assuntos

- Pacote de malhas: organiza o objeto desenhado em vetores, facilita o acesso do simulador às propriedades de cada célula.
- Pacote de simulação: nela está presente o coração do simulador: o solver da equação da temperatura, discretizada por métodos numéricos, e resolvida por método iterativo.
- Pacote de interpolação: utilizado para realizar interpolação com propriedades termofísicas dos materiais, é acessado pelo simulador, e retorna as propriedades do material.
- Pacote de correlação: mesma função da linha acima, mas para método de correlação.
- Pacote de interface ao usuário: utilização da biblioteca Qt, para criar interface gráfica amigável. Fornece um ambiente onde o usuário pode enviar comandos para o simulador de maneira fácil, e apresenta os resultados.
- Pacote de gráficos: utilização da biblioteca qcustomplot, para montar os melhores gráficos para o problema. É solicitado ao pacote de malhas os resultados da temperatura. Está presente junto com o pacote de interface

1.4 Diagrama de pacotes – assuntos

Abaixo é apresentado o diagrama de pacotes (Figura 1.10).

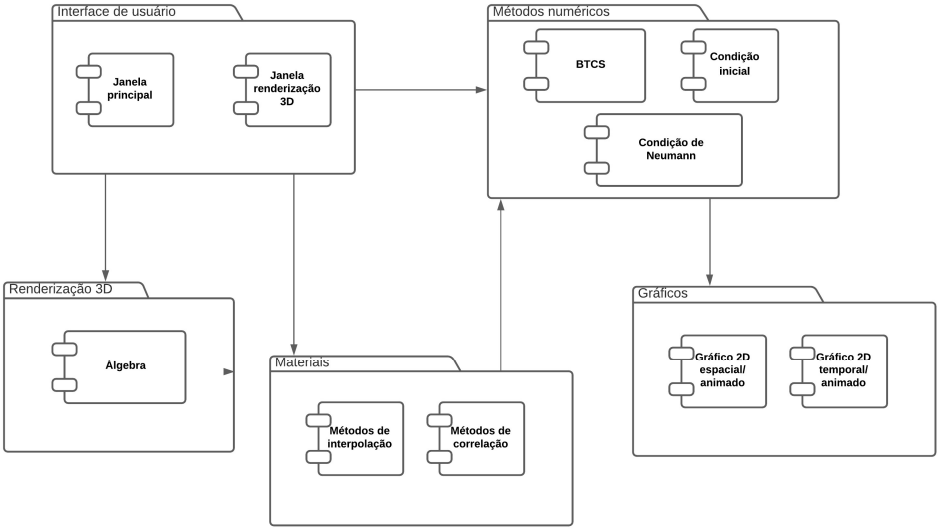

Figura 1.10: Diagrama de Pacotes