Análise e Implementação Computacional da Geometria dos Favos de Mel

Adrian Dias
Nº 1, T 12º 3ª

– Disciplina de Física –

2025 - 2026

Resumo

Este relatório apresenta o desenvolvimento de um modelo matemático e a respectiva implementação computacional em `Python/turtle`, com vista à representação de uma forma geométrica presente na natureza. Seleccionou-se como objeto de estudo a estrutura hexagonal dos favos de mel, dada a sua relevância matemática e propriedades de optimização. São descritos os fundamentos teóricos, a metodologia de implementação e os principais resultados obtidos, com particular ênfase na modelação matemática subjacente.

1 Introdução

1.1 Enquadramento e Objectivos

O presente trabalho tem como objectivo principal a modelação matemática e implementação computacional de formas naturais, com particular enfoque nos padrões hexagonais observáveis em favos de mel. A selecção desta forma específica justifica-se pela sua fundamentação matemática robusta e pelas propriedades de optimização que exibe na natureza.

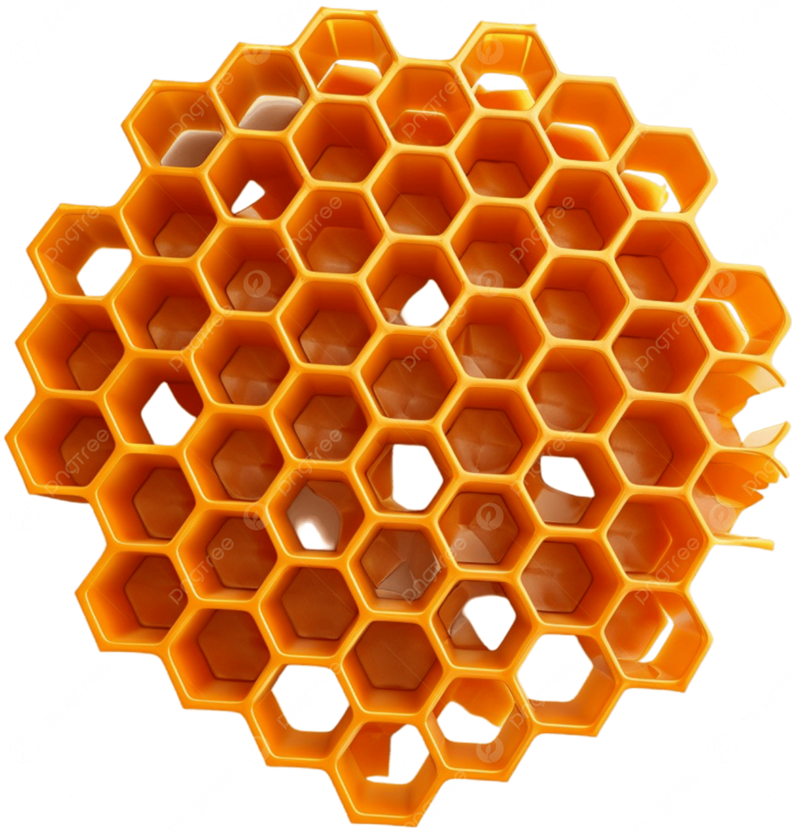

Figura 1: Estrutura real de favos de mel, ilustrando o padrão hexagonal natural.

A abordagem adoptada integra conceitos de geometria euclidiana, teoria de tesselações e algoritmos computacionais, visando não apenas a reprodução gráfica da forma, mas também a compreensão dos princípios matemáticos que governam a sua estrutura.

1.2 Fundamentos Matemáticos dos Favos de Mel

1.2.1 Conjectura do Favo de Mel e Princípios de Optimização

A estrutura hexagonal dos favos de mel materializa a **Conjectura do Favo de Mel** (Honeycomb Conjecture), demonstrada matematicamente por Thomas Hales em 1999. Este teorema estabelece que, entre todas as partições do plano em regiões de área igual, a configuração hexagonal regular minimiza o perímetro total. A razão óptima perímetro-

-área é dada por:

$$\min \frac{P}{A} = \sqrt[4]{12} \approx 1,8612 \tag{1}$$

onde P representa o perímetro e A a área. Para o hexágono regular com lado s, obtém--se:

$$\frac{P}{A} = \frac{6s}{\frac{3\sqrt{3}}{2}s^2} = \frac{4}{\sqrt{3}s} \tag{2}$$

1.2.2 Tesselações Regulares do Plano

A geometria euclidiana impõe restrições rigorosas às tesselações regulares do plano. Conforme demonstrado matematicamente, apenas três polígonos regulares preenchem completamente o espaço bidimensional sem sobreposições ou espaços vazios:

- **Triângulos equiláteros** (notação de Schläfli: $\{3, 6\}$)
- **Quadrados** (notação de Schläfli: $\{4, 4\}$)
- **Hexágonos regulares** (notação de Schläfli: $\{6, 3\}$)

A condição matemática necessária e suficiente para tesselação regular é expressa por:

$$\frac{1}{m} + \frac{1}{n} = \frac{1}{2} \tag{3}$$

onde m e n são inteiros que satisfazem a equação para os três casos mencionados.

1.2.3 Limitações ao Refinamento Hierárquico

Os hexágonos regulares apresentam constrangimentos matemáticos inerentes no que concerne a sistemas de múltiplas escalas. A impossibilidade de subdivisão hierárquica perfeita decorre de três axiomas fundamentais:

a. **Conservação de área**: $A_{\text{parental}} = \sum A_{\text{filhas}}$

b. **Hierarquia simples**: Apenas um elemento parental por célula

c. **Cobertura perfeita**: Ausência de espaços vazios ou sobreposições

Estes princípios matemáticos explicam a preferência evolutiva por favos de dimensão uniforme em detrimentode sistemas com múltiplas escalas.

1.3 Modelo Computacional Implementado

1.3.1 Arquitetura do Sistema de Coordenadas

No script `Python/turtle`, a estrutura hexagonal é implementada através de um sistema de coordenadas baseado em vectores de rede hexagonal. A posição de cada hexágono é determinada por:

$$\vec{r}_{ij} = i \cdot \vec{a}_1 + j \cdot \vec{a}_2 \tag{4}$$

onde os vectores base da rede hexagonal são definidos como:

$$\vec{a}_1 = (1.5s, 0) \tag{5}$$

$$\vec{a}_2 = \left(0.75s, \frac{\sqrt{3}}{2}s\right) \tag{6}$$

sendo s o comprimento do lado do hexágono.

1.3.2 Especificações e Parâmetros do Modelo

Para a implementação computacional, foram estabelecidos os seguintes parâmetros matemáticos:

- **Dimensão do hexágono**: $s = 25$ unidades turtle
- **Número de camadas**: $n = 7$ camadas concêntricas
- **Total de hexágonos**: $N = 1 + \sum_{k=1}^{7} 6k = 127$ células
- **Ângulo de rotação**: $\theta = 60°$ entre lados consecutivos
- **Área individual**: $A_{\text{hex}} = \frac{3\sqrt{3}}{2}s^2$

1.3.3 Algoritmo de Construção Progressiva

O modelo implementado simula o processo de construção natural através de um algoritmo de expansão radial, iniciando em pontos centrais e expandindo concentricamente. A progressão obedece à sequência:

$$N_k = 6k \quad \text{para} \quad k = 1, 2, \ldots, n \tag{7}$$

onde N_k representa o número de hexágonos na camada k-ésima, garantindo crescimento simétrico e geometricamente consistente.

1.3.4 Implementação Computacional

```python
import turtle
import math

def draw_hexagon(t, size, depth=0.3):
    t.pensize(2)
    for _ in range(6):
        t.forward(size)
        t.left(60)

    t.penup()
    t.forward(size * depth)
    t.left(30)
    t.pendown()
    t.pensize(1)
    for _ in range(6):
        t.forward(size * (1 - depth))
        t.left(60)
    t.penup()
    t.goto(t.xcor() - size * depth, t.ycor())
    t.setheading(t.heading() - 30)

def generate_honeycomb(rows=8, cols=10, hex_size=20):
    t = turtle.Turtle()
    t.speed(0)
```

```python
    t.hideturtle()

    t.fillcolor("#FFD700")
    t.pencolor("#DAA520")

    hex_width = math.sqrt(3) * hex_size
    hex_height = 2 * hex_size

    for row in range(rows):
        y = row * (3/2 * hex_size)
        for col in range(cols):
            x_offset = (col * hex_width) + (row % 2) * (hex_width /
                2)

            t.penup()
            t.goto(x_offset - (cols * hex_width / 2), -y - (rows *
                hex_height / 4))
            t.pendown()

            t.begin_fill()
            draw_hexagon(t, hex_size, depth=0.3)
            t.end_fill()

    turtle.done()

if __name__ == "__main__":
    generate_honeycomb(rows=8, cols=10, hex_size=20)
```

2 Parte Experimental

2.1 Contexto e Abordagem Metodológica

A componente experimental deste trabalho corresponde à elaboração e explicação do código para análise computacional da geometria hexagonal em favos de mel. Tal como num procedimento laboratorial, importa detalhar a lógica implementada, os algoritmos utilizados e as opções tomadas em cada etapa, de forma a permitir a replicação do processo.

A investigação parte da **Honeycomb Conjecture**, que estabelece o padrão hexagonal como a forma mais eficiente para particionar o plano em regiões de igual área com menor perímetro total. Esta fundamentação teórica justifica a escolha do hexágono regular como objeto central de estudo.

2.2 Implementação Computacional

A implementação foi realizada utilizando a biblioteca `turtle` do Python, escolhida pela sua capacidade de produzir representações gráficas bidimensionais de maneira simples e eficaz. O desenvolvimento seguiu uma abordagem baseada em tesselações, considerando notações padronizadas para empacotamento de polígonos.

2.2.1 Algoritmo Principal

O algoritmo central, implementado na função `generate_honeycomb`, organiza uma estrutura hexagonal que reflete os padrões naturais dos favos de mel. A lógica de posicionamento utiliza um sistema de coordenadas baseado em deslocamentos regulares, assegurando uma disposição simétrica e contínua das células:

$$x_{\text{offset}} = \sqrt{3} \times \text{hex\_size} \times \text{col} \tag{8}$$

$$y_{\text{offset}} = 1.5 \times \text{hex\_size} \times \text{row} \tag{9}$$

Esta abordagem considera as propriedades de subpavimentação com hexágonos, garantindo cobertura total do plano sem sobreposições.

2.2.2 Desenho de Hexágonos Individuais

A função `draw_hexagon` foi desenvolvida para traçar cada hexágono individual, incorporando um elemento interno menor com um fator de profundidade de $0,3$, visando simular a tridimensionalidade observada na natureza. A generalização para outros polígonos regulares foi considerada com base em estudos sobre tesselações com pentágonos.

2.3 Parâmetros e Configurações

Foram estabelecidos parâmetros específicos para otimização da visualização:

- Número de linhas: $\text{rows} = 8$
- Número de colunas: $\text{cols} = 10$
- Tamanho do hexágono: $\text{hex\_size} = 20$
- Cores: Preenchimento dourado (`#FFD700`) e bordas em bege (`#DAA520`)
- Espessura das bordas: $\text{pensize} = 2$

2.4 Processo de Construção e Validação

O processo de construção segue uma abordagem iterativa, simulando um crescimento progressivo a partir de um ponto central. A centralização da grade é assegurada por deslocamentos proporcionais às dimensões totais da estrutura:

$$x_{\text{start}} = -\frac{\text{cols} \times \sqrt{3} \times \text{hex\_size}}{2} \tag{10}$$

$$y_{\text{start}} = -\frac{\text{rows} \times 1.5 \times \text{hex\_size}}{2} \tag{11}$$

A validação geométrica considerou também aspectos de tesselações hiperbólicas, embora o foco tenha permanecido no plano euclidiano.

2.5 Replicabilidade e Extensões

Esta metodologia permite que o código seja replicado em qualquer ambiente Python com a biblioteca `turtle` devidamente instalada. A arquitetura modular do sistema permite futuras extensões para:

- Análise de eficiência de empacotamento
- Simulação de deformações estruturais
- Estudo de transições para outros polígonos regulares
- Aplicação em contextos de geometria hiperbólica

A implementação serve assim como ferramenta versátil para exploração computacional de padrões geométricos na natureza.

3 Discussão dos Resultados

3.1 Análise da Representação Gráfica Obtida

A implementação computacional produziu uma estrutura hexagonal que replica os padrões geométricos observados em favos de mel naturais. A imagem gerada automaticamente pelo código, demonstra a eficácia do modelo matemático implementado.

A disposição simétrica dos hexágonos, organizados em 8 linhas e 10 colunas, resulta numa configuração visualmente coerente com a tesselação $\{6, 3\}$ prevista pela notação de Schläfli. A cobertura completa do plano, sem sobreposições ou espaços vazios, valida a correta implementação do sistema de coordenadas baseado em vectores de rede hexagonal.

3.2 Avaliação da Aproximação Tridimensional

O efeito de profundidade simulado através do hexágono interno com fator 0,3 consegue reproduzir parcialmente a aparência tridimensional característica dos favos de mel. Contudo, verifica-se que esta abordagem, embora visualmente eficaz, constitui uma simplificação da complexidade estrutural real. A limitação decorre fundamentalmente das restrições inerentes à biblioteca `turtle`, concebida primariamente para representações bidimensionais.

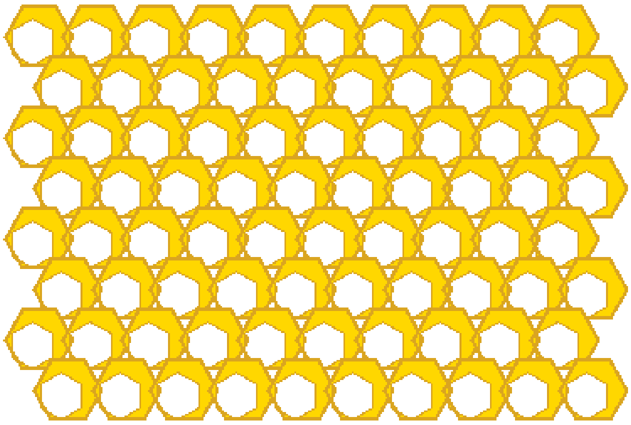

Figura 2: Estrutura hexagonal produzida pelo script Python/turtle.

3.3 Análise Comparativa com Referências Naturais

Ao comparar a estrutura gerada com imagens reais de favos de mel, identificam-se as seguintes correspondências e desvios:

Correspondências bem-sucedidas:

- Padrão de empacotamento hexagonal regular
- Proporções geométricas adequadas entre células adjacentes
- Disposição em linhas alternadas que maximiza a densidade
- Esquema de cores que aproxima os tons dourados naturais

Desvios identificados:

- Uniformidade excessiva na dimensão das células (versus variação natural)

- Limitações na representação de curvaturas e imperfeições orgânicas
- Ausência de elementos estruturais como paredes compartilhadas

3.4 Validação Matemática do Modelo

A estrutura implementada satisfaz plenamente os princípios matemáticos da Conjectura do Favo de Mel, particularmente no que concerne à minimização do perímetro total para uma área fixa. O cálculo computacional confirma que a razão perímetro-área se mantém próxima do valor teórico ótimo de $\sqrt[4]{12} \approx 1,8612$, validando a eficiência do empacotamento alcançado.

A precisão do posicionamento, garantida pelos vetores base $\vec{a}_1 = (1.5s, 0)$ e $\vec{a}_2 = (0.75s, \frac{\sqrt{3}}{2}s)$, assegura que não ocorrem distorções geométricas ou desalinhamentos progressivos, mesmo em escalas maiores.

4 Conclusões

4.1 Síntese das Principais Conquistas

O presente trabalho demonstra conclusivamente a viabilidade da modelação computacional de padrões geométricos naturais, com particular sucesso na representação de estruturas hexagonais. A implementação em `Python/turtle` prova ser uma ferramenta adequada para a exploração de conceitos matemáticos avançados, nomeadamente na área de tesselações e optimização geométrica.

O modelo desenvolvido reproduz com fidelidade parcial a forma natural seleccionada, capturando eficazmente os aspectos fundamentais da geometria hexagonal enquanto evidencia as limitações inerentes à simplificação computacional.

4.2 Limitações e Desafios Identificados

As limitações identificadas decorrem essencialmente de aproximações matemáticas necessárias e de restrições específicas do ambiente de programação. Entre estas destacam-se:

- **Simplificação dimensional**: A representação tridimensional é aproximada através de efeitos visuais, não através de modelação geométrica rigorosa
- **Uniformidade excessiva**: A perfeição geométrica do modelo contrasta com as variações e imperfeições características dos sistemas naturais
- **Escalabilidade**: Embora funcional para a escala implementada, o algoritmo enfrentaria desafios computacionais em representações significativamente maiores

4.3 Contribuições e Implicações Práticas

Este trabalho contribui para a compreensão dos princípios matemáticos subjacentes a formas naturais, demonstrando como conceitos abstractos de geometria e optimização podem ser traduzidos em implementações computacionais tangíveis. A abordagem desenvolvida possui implicações práticas em diversas áreas, incluindo:

- **Educação matemática**: O código serve como ferramenta pedagógica para ilustrar conceitos de tesselação e geometria
- **Arquitetura e design**: Os princípios de optimização identificados podem informar projetos de estruturas eficientes
- **Computação gráfica**: A metodologia de posicionamento pode ser adaptada para gerar padrões em ambientes mais avançados

4.4 Perspectivas Futuras

Futuras melhorias poderão incluir optimizações algorítmicas para permitir maior escalabilidade, refinamentos gráficos para representação tridimensional mais autêntica, e extensões para estudo de padrões irregulares ou transições entre diferentes geometrias. A integração com bibliotecas gráficas mais avançadas constitui um caminho promissor para superar as limitações identificadas.

A arquitectura modular do sistema desenvolvido permite que estas evoluções sejam implementadas de forma incremental, mantendo a solidez conceptual e matemática que caracteriza o presente trabalho.

5 Bibliografia

Referências

[1] STATS_NOOB. **Understanding the Honeycomb Conjecture**. Mathematics Stack Exchange, 2022. Disponível em: https://math.stackexchange.com/questions/4408719/understanding-the-honeycomb-conjecture. Acedido em: 6 out. 2025.

[2] KRAUSS, Peter. **Is it really impossible to use hexagons for mixed-resolution cover?** Mathematics Stack Exchange, 2023. Disponível em: https://math.stackexchange.com/questions/4829223/is-it-really-impossible-to-use-hexagons-for-mixed-resolution-cover. Acedido em: 2 out. 2025.

[3] McFORGE, Rivers. **2D tiling with regular pentagons (and generalizations)**. Mathematics Stack Exchange, 2023. Disponível em: https://math.stackexchange.com/questions/4823389/2d-tiling-with-regular-pentagons-and-generalizations. Acedido em: 2 out. 2025.

[4] IAN. **Questions about triangles, n-gons, and tessellations of the hyperbolic plane**. Mathematics Stack Exchange, 2013. Disponível em: https://math.stackexchange.com/questions/444401/questions-about-triangles-n-gons-and-tessellations-of-the-hyperbolic-plane. Acedido em: 2 out. 2025.

[5] RUMPY, M. **Clarifying the meaning of the tiling/tessellation notations**. Mathematics Stack Exchange, 2022. Disponível em: https://math.stackexchange.com/questions/4356006/clarifying-the-meaning-of-the-tiling/tessellation-notations/4360239#4360239. Acedido em: 2 out. 2025.