UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA

ELLIAN DOS SANTOS RODRIGUES

PROJETO ÁLGEBRA VETORIAL LINEAR PARA COMPUTAÇÃO

ÁLGEBRA VETORIAL

Recife, 2024

SUMÁRIO

Recife, 2024

1. INTRODUÇÃO

Algebra vetorial é uma das importantes na área da computação no geral, pois estuda formas de manipulação de vetores. Grandezas escalares como massa, tempo e comprimento podem ser descritas por um número e uma unidade de medida. Já as grandezas vetoriais, além de um número e uma unidade de medida, precisam de uma direção e sentido.

No projeto Python, a ideia é mostrar graficamente operações como soma e subtração de vetores no R2 no plano cartesiano, além de mostrar alguns resultados como a projeção do vetor 1 com o vetor 2, norma,produto escalar, medida angular e área do paralelogramo

2. VETORES

A representação geométrica de um vetor é feita por uma seta. O comprimento da seta indica o módulo ou intensidade do vetor. Já o sentido é mostrado pela ponta da seta, enquanto a direção é determinada pelas retas que são paralelas a ela.

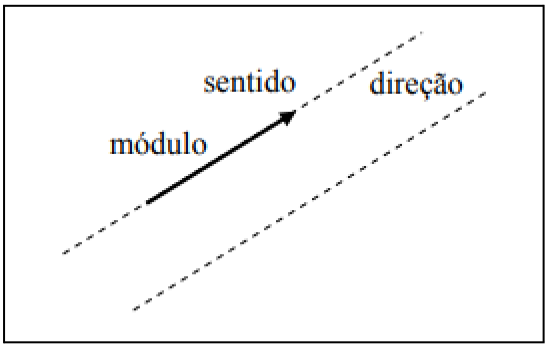


FIGURA 1 - Representação geométrica do vetor

Para denotar vetores, é usado uma seta em cima da letra (u→, v→ e w→). Para que seja definido, as seguinte propriedades devem ser satisfeitas:

$$
\begin{aligned}
\vec{u} + \vec{v} &= \vec{v} + \vec{u}, && \text{(Comutatividade da soma)}\\
\vec{u} + \left(\vec{v} + \vec{w}\right) &= \left(\vec{v} + \vec{u}\right) + \vec{w}, && \text{(Associatividade da soma)}\\
(\alpha + \beta)\,\vec{u} &= \alpha\,\vec{u} + \beta\,\vec{v}, && \text{(Distributividade da multiplicação)}\\
\alpha\left(\vec{u} + \vec{v}\right) &= \alpha\,\vec{u} + \alpha\,\vec{v}, && \text{(Distributividade da soma)}\\
\alpha\left(\beta\,\vec{u}\right) &= (\alpha\beta)\,\vec{u}, && \\
\vec{0} + \vec{v} &= \vec{v}, && \text{(Existência do vetor nulo)}\\
0\,\vec{v} &= \vec{0}, && \\
1\,\vec{v} &= \vec{v}. && \text{(Elemento neutro)}
\end{aligned}
$$

FIGURA 2 - Propriedades dos vetores

3. SOMA DE VETORES

A representação de da soma de dois vetores é dada por uma reta que vai do ponto inicial do primeiro vetor com a extremidade do último vetor:

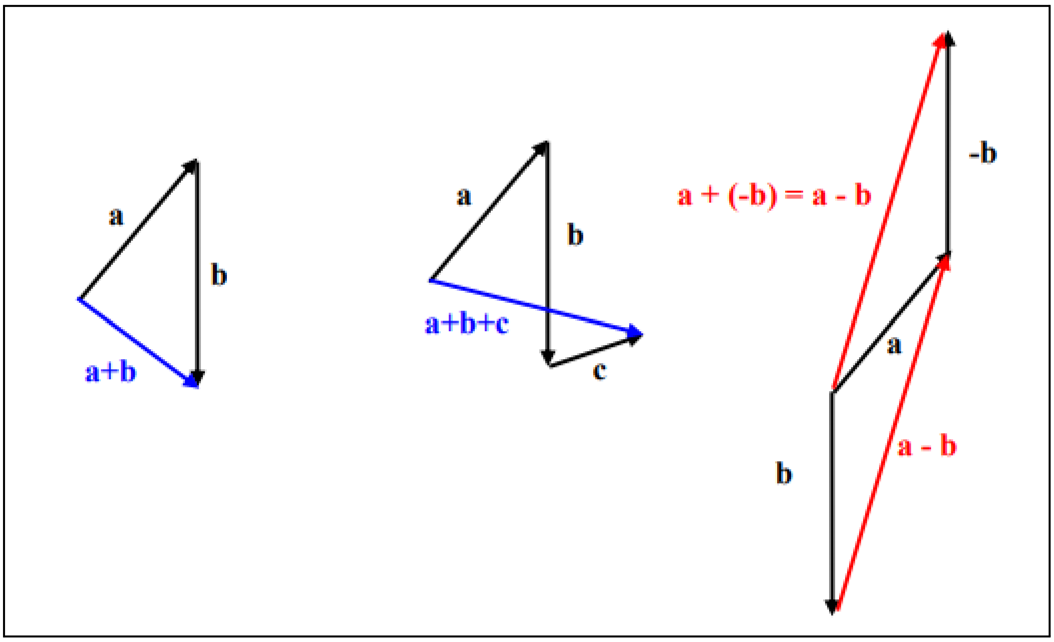


FIGURA 3 - Representação da soma de vetores

As propriedades comutativa e associativa da soma vetorial também podem ser mostradas geometricamente:

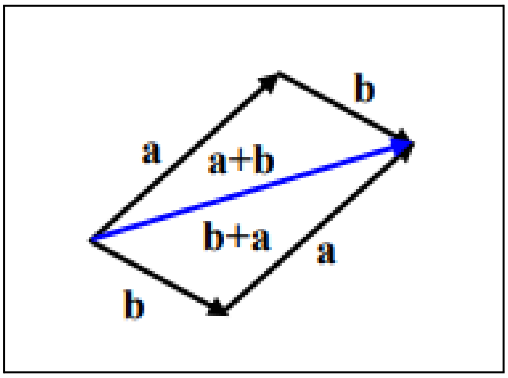


FIGURA 4 - Representação de propriedades associativas de soma

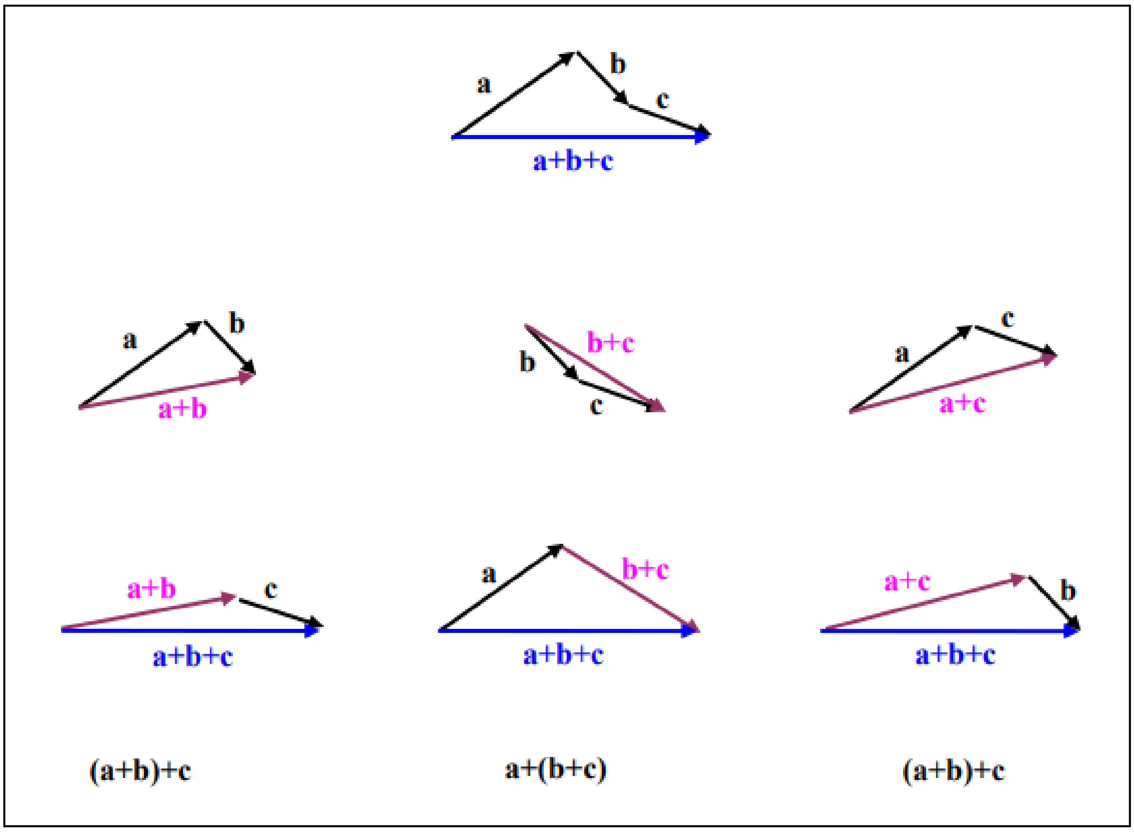
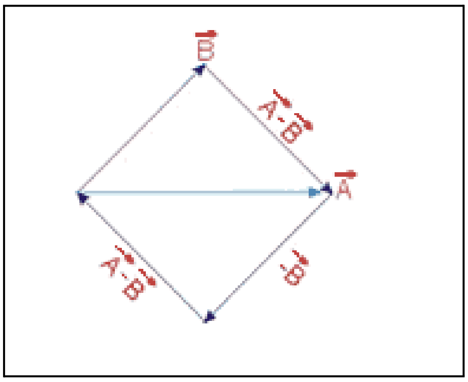


FIGURA 5 - Representação de propriedades associativas de soma 2

4. SUBTRAÇÃO DE VETORES

Dados 2 vetores A e B o vetor subtração $\vec{A}$ - $\vec{B}$ é como uma soma de $\vec{A}$ com o vetor oposto de $\vec{B}$.

$$\vec{A} - \vec{B} = \vec{A} + \left(-\vec{B}\right)$$



5. PROJEÇÃO ORTOGONAL

Dados os vetores →v e → , onde →u diferente de 0 , a projeção ortogonal de →v sobre →u :

$$\vec{v} := \frac{\vec{v} \cdot \vec{u}}{||\vec{u}||^2}\,\vec{u} = \frac{\vec{v} \cdot \vec{u}}{\vec{u} \cdot \vec{u}}\,\vec{u}.$$

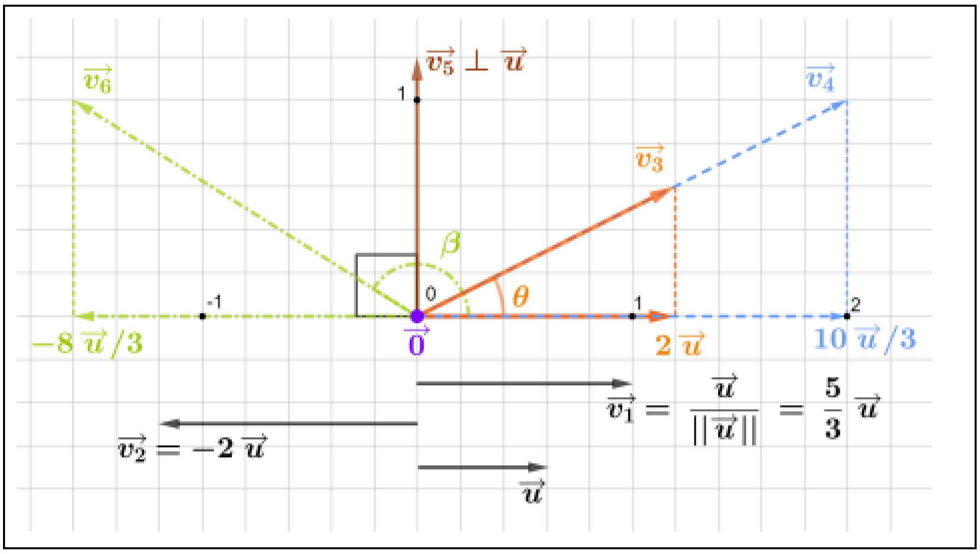


FIGURA 6 - Representação de projeções

7. PROJETO EM PYTHON

O projeto em python visa demonstrar no plano cartesiano manipulações simples com dois vetores, com soma e multiplicação, além de retornar o resultado da projeção do vetor 1 com o vetor 2.

Primeiro criamos a função de gerar o plano cartesiano, os valores são altos para caberem uma variação maior de vetores.

```
Codeium: Refactor | Explain | Generate Docstring | X
def plano_cartesiano(canvas):
    largura = canvas.winfo_width()
    altura = canvas.winfo_height()
    x = largura // 2
    y = altura // 2

    canvas.create_line(0, y, largura, y, fill="black")
    canvas.create_line(x, 0, x, altura, fill="black")

    for i in range(0, largura, 50):
        canvas.create_line(i, y - 5, i, y + 5, fill="black")
        canvas.create_text(i, y + 15, text=f'{i - x}', font=("Arial", 8), fill="black")
    for i in range(0, altura, 50):
        canvas.create_line(x - 5, i, x + 5, i, fill="black")
        canvas.create_text(x + 15, i, text=f'{y - i}', font=("Arial", 8), fill="black")
```

FIGURA 7 - Função cartesiana

Em seguida, com funções menores, temos como mostrar os vetores no canvas (plano cartesiano) no qual iremos usar para demonstrar os vetores. O produto escalar que calcula o vetor 1 com o vetor 2. Com projetar, fazemos o cálculo de projetar o vetor 1 no vetor 2, respeitando os conceitos já discutidos.

```python
Codeium: Refactor | Explain | Generate Docstring | X
def mostrar_vetor(canvas, origem, vetor, cor):
    canvas.create_line(origem[0], origem[1], origem[0] + vetor[0], origem[1] - vetor[1], arrow=tk.LAST, fill=cor)
    return (origem[0] + vetor[0], origem[1] - vetor[1])

Codeium: Refactor | Explain | Generate Docstring | X
def produto_escalar(vetor1, vetor2):
    return np.dot(vetor1, vetor2)

Codeium: Refactor | Explain | Generate Docstring | X
def projetar(vetor_a, vetor_b):
    escala = produto_escalar(vetor_a, vetor_b) / np.dot(vetor_b, vetor_b)
    return escala * vetor_b
```

FIGURA 8 - Funções menores

A medida angular, norma e área do paralelogramo também não ficam de fora do programa, sendo simples pois faço utilizando a biblioteca numpy para auxiliar nos cálculos.

```python
def medida_angular(vetor1, vetor2):
    produto = produto_escalar(vetor1, vetor2)
    normaV1 = np.linalg.norm(vetor1)
    normaV2 = np.linalg.norm(vetor2)
    cos_theta = produto / (normaV1 * normaV2)
    radiano = np.arccos(cos_theta)
    graus = np.degrees(radiano)
    return graus

Codeium: Refactor | Explain | Generate Docstring | X
def norma(vetor):
    return np.linalg.norm(vetor)

Codeium: Refactor | Explain | Generate Docstring | X
def area_paralelogramo(vetor1, vetor2):
    produto_vetorial = np.cross(vetor1, vetor2)
    area = np.linalg.norm(produto_vetorial)
    return area
```

FIGURA 9 - Funções menores parte 2

A maior função ficará com a soma (e subtração), que pega os valores dado pelo usuário, mostra eles no plano, faz o cálculo e novamente mostra ele no plano, além claro de apagar todos os vetores anteriores.

```
def calcular_soma():
    try:
        v1_x = float(entry_v1_x.get())
        v1_y = float(entry_v1_y.get())
        v2_x = float(entry_v2_x.get())
        v2_y = float(entry_v2_y.get())

        testeV1 = np.array([v1_x, v1_y])
        testeV2 = np.array([v2_x, v2_y])

        resultado_soma = testeV1 + testeV2

        canvas.delete('all')
        plano_cartesiano(canvas)

        origem = (350, 350)
        mostrar_vetor(canvas, origem, testeV1, cor='red')
        mostrar_vetor(canvas, origem, testeV2, cor='green')
        mostrar_vetor(canvas, origem, resultado_soma, cor='blue')

        resultado_projecao = projetar(testeV1, testeV2)
        canvas.create_text(100, 20, text=f'Projeção: {resultado_projecao}', font=("Arial", 10), fill="black")
```

FIGURA 10 - Função de soma

```
def calcular_subtracao():
    try:
        v1_x = float(entry_v1_x.get())
        v1_y = float(entry_v1_y.get())
        v2_x = float(entry_v2_x.get())
        v2_y = float(entry_v2_y.get())

        testeV1 = np.array([v1_x, v1_y])
        testeV2 = np.array([v2_x, v2_y])

        resultado_subtracao = testeV1 - testeV2

        canvas.delete('all')
        plano_cartesiano(canvas)

        mostrar_vetor(canvas, (350, 350), testeV1, cor='red')
        mostrar_vetor(canvas, (350, 350), testeV2, cor='green')
        mostrar_vetor(canvas, (350, 350), resultado_subtracao, cor='blue')

    except ValueError:
        canvas.create_text(100, 20, text='Sem valores', font=("Arial", 10), fill="red")
```

FIGURA 11 - Função de subtração

Para criar a interface e os gráficos utilizei o tkinter e numpy.

8. REFERÊNCIAS

- Escrita:

  https://www.ifi.unicamp.br/~mauro/F315/Revisao_Vetores.pdf

  https://www.educabras.com/aula/algebra-vetorial

  https://www.cin.ufpe.br/~brgccf/archive/Algebra%20Linear%20Boldrini.pdf

  https://www.ufrgs.br/reamat/Calculo/livro-cfvv/xv.html

  https://www.youtube.com/watch?v=l0PnHtUKDBY&list=PLFdqTtnbHmDPnF5TN6f010-PiPENLm0PA

- Código Python:

  https://docs.python.org/3/library/tkinter.html

  https://www.tutorialspoint.com/python/tk_canvas.htm

  https://realpython.com/python-gui-tkinter/

  https://www.geeksforgeeks.org/how-to-create-a-vector-in-python-using-numpy/

  https://jakevdp.github.io/PythonDataScienceHandbook/

  https://www.youtube.com/watch?v=9cF3Nougasg&list=PLOQU3c_3DSpK4D0k3OLj3HLZ7LMhb9GvB&index=14