Kernel: SageMath 10.6

Trabalho Avaliativo - GA

Aluno(a): Cintia da Silva Bulcão

RA: 145107

Data de Entrega: 15/06/25

01) Crie uma rotina com a definição de três funções, uma que calcule o produto escalar em 3D, uma que calcule o produto vetorial e uma que calcule o produto misto. Crie um menu para que o usuário escolha o que quer calcular. Se escolher o produto escalar ou o vetorial, peça que ele insira os dois vetores 3D necessários, se ele escolher o produto misto, peça que ele insira os três vetores 3D. Não há necessidade de criar ilustrações.

In [3]:
```
def produto_escalar(v1: list, v2: list) -> float:
    return sum([v1[i] * v2[i] for i in range(3)])

def produto_vetorial(v1: list, v2: list) -> list:
    x = v1[1] * v2[2] - v1[2] * v2[1]
    y = -(v1[0] * v2[2] -v1[2] * v2[0])
    z = v1[0] * v2[1] - v1[1] * v2[0]
    return [x, y, z]

def produto_misto(v1: list, v2: list, v3: list) -> float:
    from sage.all import matrix, RR
    mat = matrix(RR, 3, 3, v1 + v2 + v3)
    return round(mat.determinant(), 2)

def solicitar_vetor(nome: str) -> list:
    while True:
        entrada = input(f'Digite as 3 coordenadas do vetor {nome} separadas por vírgulas: ')
        vetor = list(map(float, entrada.strip().split(',')))
        if len(vetor) == 3:
            return vetor
```

```
        print(f'O vetor informado possui {len(vetor)} coordenadas. Tente novamente.')

print('\n--- Calculadora de Produtos entre Vetores 3D ---\n')

opcao = 0
while opcao != 4:
    print('Escolha a operação desejada:')
    print('1 - Produto Escalar')
    print('2 - Produto Vetorial')
    print('3 - Produto Misto')
    print('4 - Sair')

    try:
        opcao = int(input('Digite sua opção: '))
    except ValueError:
        print('Por favor, insira um número válido.\n')
        continue

    if opcao == 1:
        print('\n[Produto Escalar Selecionado]')
        a = solicitar_vetor('A')
        b = solicitar_vetor('B')
        resultado = produto_escalar(a, b)
        print(f'Resultado: o produto escalar entre {a} e {b} é {resultado}\n')

    elif opcao == 2:
        print('\n[Produto Vetorial Selecionado]')
        a = solicitar_vetor('A')
        b = solicitar_vetor('B')
        resultado = produto_vetorial(a, b)
        print(f'Resultado: o produto vetorial entre {a} e {b} é {resultado}\n')

    elif opcao == 3:
        print('\n[Produto Misto Selecionado]')
        a = solicitar_vetor('A')
        b = solicitar_vetor('B')
        c = solicitar_vetor('C')
        resultado = produto_misto(a, b, c)
        print(f'Resultado: o produto misto entre {a}, {b} e {c} é {resultado}\n')

    elif opcao == 4:
        print('\nSaindo do programa. Até logo!\n')
```

```
    else:
        print('\nOpção inválida. Por favor, escolha uma das opções disponíveis.\n')
```

Out[3]:

```
--- Calculadora de Produtos entre Vetores 3D ---

Escolha a operação desejada:
1 - Produto Escalar
2 - Produto Vetorial
3 - Produto Misto
4 - Sair

Digite sua opção: 3

Digite as 3 coordenadas do vetor A separadas por vírgulas: 

[Produto Misto Selecionado]

Digite as 3 coordenadas do vetor B separadas por vírgulas: 1,5,4

Digite as 3 coordenadas do vetor C separadas por vírgulas: 1,4,9

Resultado: o produto misto entre [1.0, 2.0, 5.0], [1.0, 5.0, 4.0] e [1.0, 4.0, 9.0] é 14.0

Escolha a operação desejada:
1 - Produto Escalar
2 - Produto Vetorial
3 - Produto Misto
4 - Sair

Digite sua opção: 4

Saindo do programa. Até logo!
```

02)Crie uma rotina que receba as coordenadas de três vetores u, v e w do espaço e retorne:

. (i) a área do paralelogramo determinado por u e v e sua ilustração.

. (ii) o volume do paralelepípedo determinado por u, v e w e sua ilustração.

. (iii) a altura do paralelepípedo relativa à base determinada por u e v (sem ilustração).

In [4]:

```
from sage.all import arrow, show, polygon, plot, Polyhedron, vector, sqrt

def produto_vetorial(u: list, v: list) -> list:
    return [
        u[1]*v[2] - u[2]*v[1],
        -(u[0]*v[2] - u[2]*v[0]),
        u[0]*v[1] - u[1]*v[0]
    ]

def modulo_vetor(v: list) -> float:
    return sqrt(sum([v[i]**2 for i in range(3)]))

def produto_misto(u: list, v: list, w: list) -> float:
    from sage.all import matrix, RR
    mat = matrix(RR, 3, 3, u + v + w)
    return round(mat.determinant(), 2)

def pedir_vetor(nome: str) -> list:
    while True:
        entrada = input(f'Informe as 3 coordenadas do vetor {nome} separadas por vírgulas: ')
        vetor = list(map(float, entrada.strip().split(',')))
        if len(vetor) == 3:
            return vetor
        print(f'O vetor {nome} possui {len(vetor)} coordenadas. Tente novamente.\n')

print('\nIniciando análise geométrica com 3 vetores: u, v e w.')
print('Serão calculados: área do paralelogramo (u x v), volume do paralelepípedo (u, v, w), e altura relativa.\n')

while True:
    u = pedir_vetor('u')
    v = pedir_vetor('v')
    w = pedir_vetor('w')

    if produto_misto(u, v, w) == 0:
        print('\nOs vetores fornecidos estão num mesmo plano. Tente vetores não coplanares.\n')
        continue
    break
```

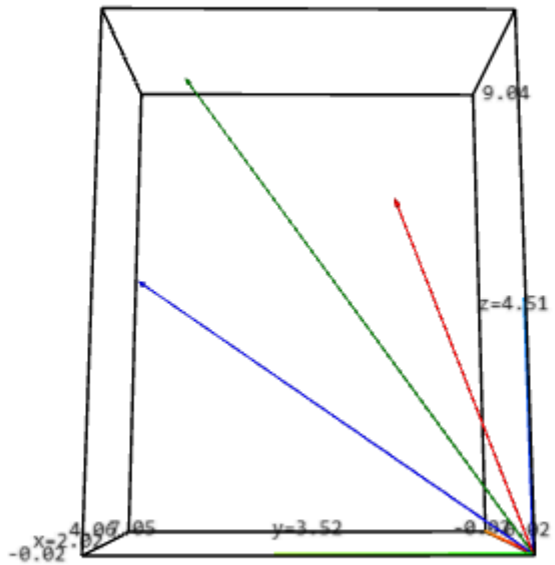
```
ponto_origem = [0, 0, 0]
seta_u = arrow(ponto_origem, u, arrowsize=2, color='red')
seta_v = arrow(ponto_origem, v, arrowsize=2, color='blue')
seta_w = arrow(ponto_origem, w, arrowsize=2, color='green')
show(seta_u + seta_v + seta_w, axes=True)


vet_cross = produto_vetorial(u, v)
area_base = modulo_vetor(vet_cross)
print(f'\nÁrea do paralelogramo formado por u e v: {area_base} unidades².')


uv = [u[i] + v[i] for i in range(3)]
base_paralelo = polygon([ponto_origem, u, uv, v], color='gray')
show(seta_u + seta_v + base_paralelo, axes=True)


volume = abs(produto_misto(u, v, w))
print(f'Volume do paralelepípedo formado por u, v e w: {volume} unidades³.')


uw = [u[i] + w[i] for i in range(3)]
vw = [v[i] + w[i] for i in range(3)]
uvw = [uv[i] + w[i] for i in range(3)]

forma3d = Polyhedron(vertices=[ponto_origem, u, v, w, uv, uw, vw, uvw])
grafico_final = plot(forma3d, alpha=0.2, color='blue', axes=True)
show(seta_u + seta_v + seta_w + base_paralelo + grafico_final, axes=True)


altura = volume / area_base
print(f'Altura do paralelepípedo em relação à base (u x v): {altura:.2f} unidades.')
```

Out[4]:

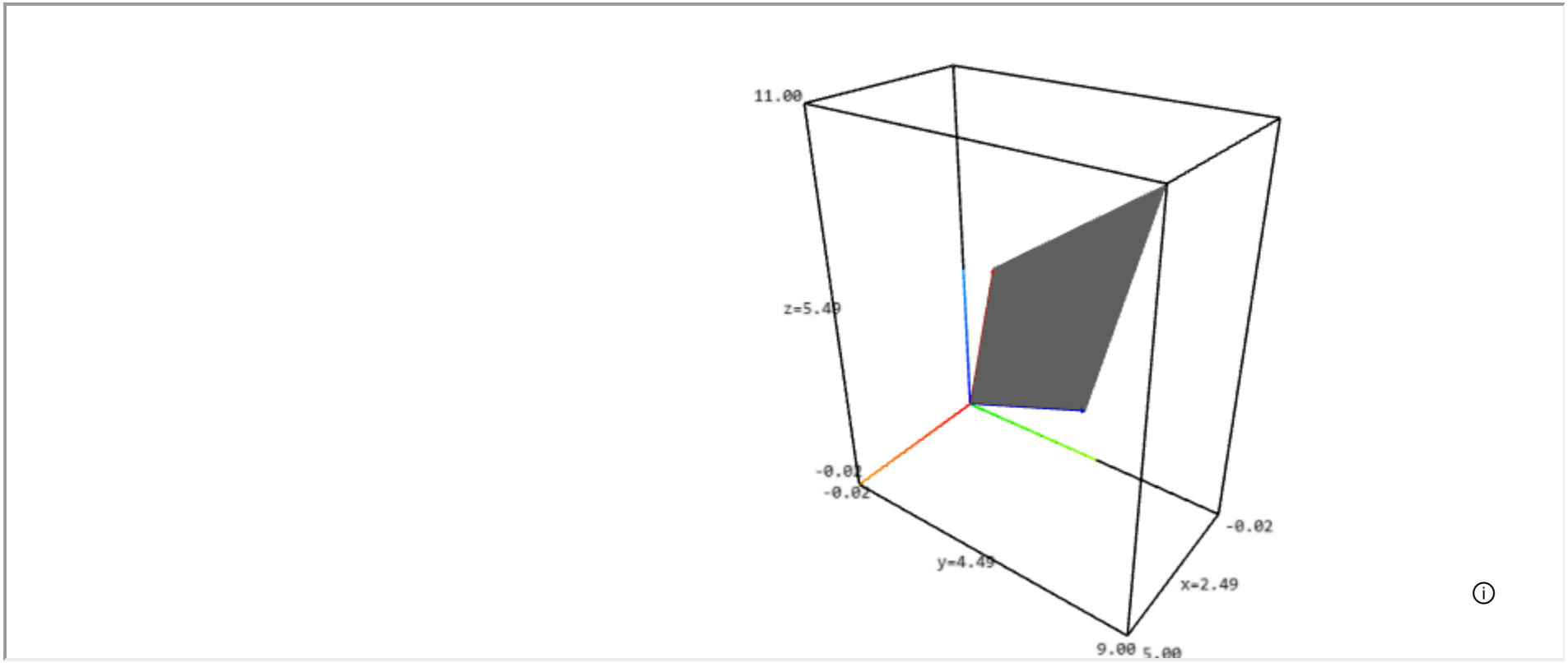
```
Iniciando análise geométrica com 3 vetores: u, v e w.
Serão calculados: área do paralelogramo (u x v), volume do paralelepípedo (u, v, w), e altura relativa.

Informe as 3 coordenadas do vetor u separadas por vírgulas: 1,2,6

Informe as 3 coordenadas do vetor v separadas por vírgulas: 4,7,5
```

Informe as 3 coordenadas do vetor w separadas por vírgulas: 3,6,9

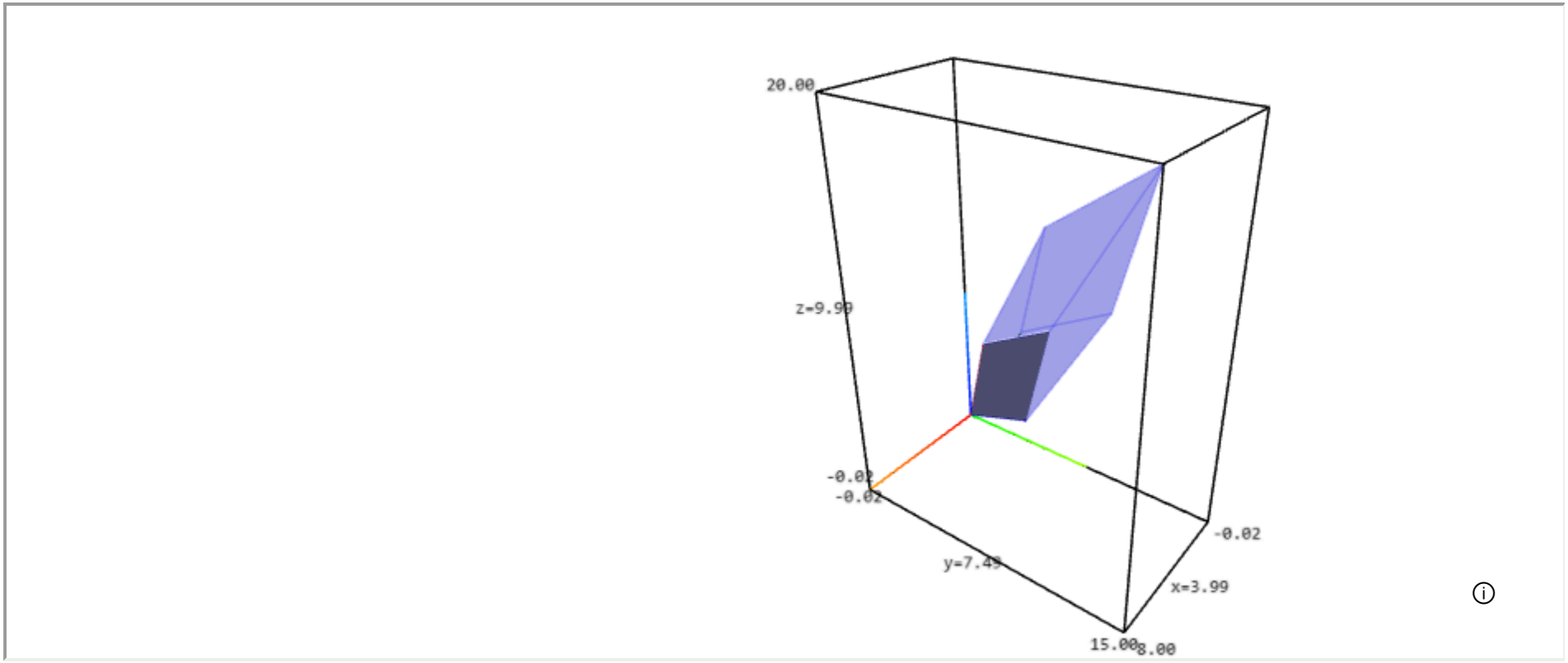
```
/ext/sage/10.6/local/var/lib/sage/venv-python3.12.5/lib/python3.12/site-packages/scikits/__init__.py:1: DeprecationWarning:
pkg_resources is deprecated as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html
  __import__("pkg_resources").declare_namespace(__name__)
/ext/sage/10.6/local/var/lib/sage/venv-python3.12.5/lib/python3.12/site-packages/scikits/__init__.py:1: DeprecationWarning:
Deprecated call to `pkg_resources.declare_namespace('scikits')`.
Implementing implicit namespace packages (as specified in PEP 420) is preferred to `pkg_resources.declare_namespace`. See
https://setuptools.pypa.io/en/latest/references/keywords.html#keyword-namespace-packages
  __import__("pkg_resources").declare_namespace(__name__)
```



Área do paralelogramo formado por u e v: 37.22902093797257 unidades².



Volume do paralelepípedo formado por u, v e w: 9.0 unidades³.




```
Altura do paralelepípedo em relação à base (u x v): 0.24 unidades.
```

03) Crie um programa no qual o usuário consiga inserir quantos vetores de 3 dimensões ele desejar. Impeça que ele insira vetores de dimensões diferentes de 3. Seu programa deve procurar conjuntos de três vetores coplanares na lista fornecida e imprimir os resultados.

```
def produto_misto(v1: list, v2: list, v3: list) -> float:
    from sage.all import matrix, RR
    mat = matrix(RR, 3, 3, v1 + v2 + v3)
    return round(mat.determinant(), 2)

def pedir_vetor_3d(index: int) -> list:
    while True:
        entrada = input(f'→ Vetor #{index + 1}: informe 3 coordenadas separadas por vírgula: ')
        vetor = list(map(float, entrada.strip().split(',')))
        if len(vetor) == 3:
```

```
            return vetor
        print(f'Vetor com {len(vetor)} coordenadas. Por favor, insira apenas 3.\n')

print('\nVerificador de Conjuntos Coplanares de Vetores 3D\n')

try:
    total = int(input('Quantos vetores deseja inserir? '))
except ValueError:
    print('Entrada inválida. Encerrando programa.')
    exit()

lista_vetores = []
for i in range(total):
    vetor = pedir_vetor_3d(i)
    lista_vetores.append(vetor)

print('\nVetores inseridos:')
for idx, vetor in enumerate(lista_vetores):
    print(f'  V{idx+1}: {vetor}')

coplanares = []

for i in range(len(lista_vetores)):
    for j in range(i + 1, len(lista_vetores)):
        for k in range(j + 1, len(lista_vetores)):
            v1 = lista_vetores[i]
            v2 = lista_vetores[j]
            v3 = lista_vetores[k]
            if produto_misto(v1, v2, v3) == 0:
                coplanares.append((v1, v2, v3))

if coplanares:
    print('\nConjuntos de vetores coplanares encontrados:')
    for grupo in coplanares:
        print(f'  → {grupo[0]}, {grupo[1]}, {grupo[2]}')
else:
    print('\nNenhum conjunto de três vetores coplanares foi encontrado.')
```

Out[6]:

```
Verificador de Conjuntos Coplanares de Vetores 3D
```

```
Quantos vetores deseja inserir? 3
→ Vetor #1: informe 3 coordenadas separadas por vírgula: 1,2,3
→ Vetor #2: informe 3 coordenadas separadas por vírgula: 4,5,6
→ Vetor #3: informe 3 coordenadas separadas por vírgula: 7,8,9

Vetores inseridos:
  V1: [1.0, 2.0, 3.0]
  V2: [4.0, 5.0, 6.0]
  V3: [7.0, 8.0, 9.0]

Conjuntos de vetores coplanares encontrados:
  → [1.0, 2.0, 3.0], [4.0, 5.0, 6.0], [7.0, 8.0, 9.0]
```