Kernel: SageMath 10.6

In [0]:

GA - Trabalho 2

Nome: Cintia da Silva Bulcao

Ra: 145107

1. A rotina número 01 deverá ser tal que: (a) O usuário deverá inserir as coordenadas inteiras da equação geral ax + by + cz + d = 0 de um plano qualquer que não passe pela origem, que não seja paralelo a algum plano coordenado nem paralelo a algum eixo coordenado. Essa inserção deverá ser feita de uma só vez, na ordem em que aparecem os coeficientes da equação geral, com os valores separados por vírgulas. (b) A rotina deverá, após o registro do plano, receber as coordenadas de um ponto P = (a,b,c) inseridas pelo usuário. (c) Em seguida, a rotina deverá verificar se o ponto pertence ou não ao plano fornecido e deverá imprimir essa informação na tela. (d) Por final, a rotina deverá perguntar se o usuário quer testar a pertinência de outro ponto ou não. (e) Se o usuário escolher que não quer testar mais pontos, a rotina deve perguntar se ele quer inserir os coeficientes de outro plano ou não.

In [3]:

```
seguir = 'y'
while seguir == 'y':
    plano_correto = False
    while plano_correto == False:
        entrada_texto = input('Digite os coeficientes a,b,c,d: ')
        partes = entrada_texto.split(',')

        if len(partes) != 4:
            print('Erro: insira 4 valores.')
            continue

        i = 0
        valores = []
        while i < 4:
            parte = partes[i].strip()
            if parte == '' or (parte[0] not in '-0123456789'):
```

```
            break
        j = 0
        valido = True
        while j < len(parte):
            if j == 0 and parte[j] == '-':
                j += 1
                continue
            if parte[j] < '0' or parte[j] > '9':
                valido = False
                break
            j += 1
        if valido:
            num = 0
            k = 0
            negativo = False
            if parte[0] == '-':
                negativo = True
                k = 1
            while k < len(parte):
                num = num * 10 + (ord(parte[k]) - ord('0'))
                k += 1
            if negativo:
                num = -num
            if num == 0:
                print('Coeficientes não podem ser zero.')
                break
            valores.append(num)
            i += 1
        else:
            break

    if len(valores) == 4:
        plano_correto = True
        a = valores[0]
        b = valores[1]
        c = valores[2]
        d = valores[3]
    else:
        print('Erro: insira apenas números inteiros diferentes de zero.')

checar_ponto = 'y'
```

```
while checar_ponto == 'y':
    entrada_ponto = input('Digite as coordenadas x,y,z: ')
    partes = entrada_ponto.split(',')

    if len(partes) != 3:
        print('Erro: insira 3 valores.')
        continue

    ponto = []
    i = 0
    while i < 3:
        texto = partes[i].strip()
        if texto == '' or (texto[0] not in '-0123456789'):
            break
        ponto_valido = True
        ponto_decimal = False
        j = 0
        while j < len(texto):
            if texto[j] == '.' and not ponto_decimal:
                ponto_decimal = True
                j += 1
                continue
            if j == 0 and texto[j] == '-':
                j += 1
                continue
            if texto[j] < '0' or texto[j] > '9':
                ponto_valido = False
                break
            j += 1
        if ponto_valido:
            ponto.append(float(texto))
            i += 1
        else:
            break

    if len(ponto) == 3:
        x = ponto[0]
        y = ponto[1]
        z = ponto[2]
        total = a * x + b * y + c * z + d
        if total == 0:
```

```
                print('O ponto está no plano.')
            else:
                print('O ponto não está no plano.')
        else:
            print('Erro: insira apenas números válidos.')

        checar_ponto = input('Outro ponto? (y/n): ').strip().lower()

    seguir = input('Novo plano? (y/n): ').strip().lower()

print('Encerrado.')
```

Out[3]:

```
Digite os coeficientes a,b,c,d: 1,3,6,3
Digite as coordenadas x,y,z: 1,3,1
O ponto não está no plano.
Outro ponto? (y/n): y
Digite as coordenadas x,y,z: 3,6,9
O ponto não está no plano.
Outro ponto? (y/n): n
Novo plano? (y/n): y
Digite os coeficientes a,b,c,d: 3,8,1,1
Digite as coordenadas x,y,z: 2,25,12,0
Erro: insira 3 valores.
Digite as coordenadas x,y,z: 25,12,0
O ponto não está no plano.
Outro ponto? (y/n): n
Novo plano? (y/n): n
Encerrado.
```

2. A rotina número 02 deverá ser tal que: (a) O usuário deverá inserir as coordenadas inteiras da equação geral ax + by + cz + d = 0 de um plano qualquer que não passe pela origem, que não seja paralelo a algum plano coordenado nem paralelo a algum eixo coordenado. Essa inserção deverá ser feita de uma só vez, na ordem em que aparecem os coeficientes da equação geral, com os valores separados por vírgulas. (b) A rotina deverá, após a captura dos coeficientes, imprimir a equação geral cujos coeficientes foram inseridos. Isso serve para conferência do usuário. (c) A rotina deverá determinar os pontos A, B e C, respectivamente pontos de interseção do plano com os eixos coordenados Ox, Oy e Oz. Deverá, também, imprimir as coordenadas desses pontos na tela. (d) A rotina deverá calcular a área do triângulo ABC e imprimir essa informação na tela. (e) A rotina deverá calcular a altura do triângulo ABC relativa ao lado xOz que pertence ao plano, ou seja, altura relativa ao lado AC. Deverá, também, imprimir essa informação na tela. (f) A rotina deverá calcular o volume do tetraedro limitado pelo plano fornecido e pelos planos coordenados, ou seja, o volume do tetraedro OABC. Deverá imprimir adequadamente essa informação na tela. (g) A rotina deverá mostrar parte do plano fornecido, mostrar os pontos A, B e C, e mostrar o triângulo em cor destacada. (h) Ao final, deverá perguntar ao usuário se ele quer inserir os coeficientes de outra equação geral para novos cálculos ou se quer encerrar o programa.

In [4]:

```
x, y, z = var('x y z')
continuar = 'y'


TOLERANCIA = 1e-3

while continuar == 'y':
    entrada = input('Digite a,b,c,d: ').replace(' ', '').split(',')

    while len(entrada) != 4:
        print('Erro: insira 4 valores separados por vírgula.')
        entrada = input('Digite a,b,c,d: ').replace(' ', '').split(',')

    try:
        a, b, c, d = map(float, entrada)
    except ValueError:
        print('Erro: insira valores numéricos válidos.')
        continue

    if a == 0 and b == 0 and c == 0:
        print('Erro: pelo menos um dos coeficientes a, b ou c deve ser diferente de zero.')
        continue

    print(f'\nPlano: ({a})x + ({b})y + ({c})z + ({d}) = 0')
```

```
A = (-d/a, 0, 0) if a != 0 else None
B = (0, -d/b, 0) if b != 0 else None
C = (0, 0, -d/c) if c != 0 else None

print('\nInterseções:')
print(f'A (eixo x): {A if A else "Inexistente"}')
print(f'B (eixo y): {B if B else "Inexistente"}')
print(f'C (eixo z): {C if C else "Inexistente"}')

pontos_validos = [p for p in [A, B, C] if p is not None]
if len(pontos_validos) < 3:
    print('Não é possível formar triângulo com menos de 3 interseções nos eixos.')
    continuar = input('\nNovo plano? (y/n): ').strip().lower()
    continue

A, B, C = pontos_validos


AC = [C[i] - A[i] for i in range(3)]
BC = [C[i] - B[i] for i in range(3)]


prod = [
    AC[1]*BC[2] - AC[2]*BC[1],
    AC[2]*BC[0] - AC[0]*BC[2],
    AC[0]*BC[1] - AC[1]*BC[0]
]


modulo = sum([comp**2 for comp in prod]) ** 0.5

area = modulo / 2
print(f'\nÁrea do triângulo ABC: {round(area, 4)} u²')


base = sum([comp**2 for comp in AC]) ** 0.5
altura = 2 * area / base if base != 0 else 0
print(f'Altura relativa ao lado AC: {round(altura, 4)} u')
```

```
    OC = [-C[i] for i in range(3)]
    prod_misto = sum([OC[i]*prod[i] for i in range(3)])
    volume = abs(prod_misto) / 6
    print(f'Volume do tetraedro OABC: {round(volume, 4)} u³')


    plano_eq = a*x + b*y + c*z + d == 0


    intervalo_x = [x, -1, max(A[0], 3) + 1]
    intervalo_y = [y, -1, max(B[1], 3) + 1]
    intervalo_z = [z, -1, max(C[2], 3) + 1]

    fig1 = implicit_plot3d(plano_eq, intervalo_x, intervalo_y, intervalo_z, opacity=0.2)
    fig2 = polygon([A, B, C], color='red', axes=True)
    fig3 = points([A, B, C], size=25, color='black')
    show(fig1 + fig2 + fig3)

    px = float(input('Digite x do ponto...'))
    py = float(input('Digite y do ponto...'))
    pz = float(input('Digite z do ponto...'))

    test_val = abs(a*px + b*py + c*pz + d)
    if test_val <= TOLERANCIA:
        print(f"O ponto ({px},{py},{pz}) pertence ao plano dentro da tolerância {TOLERANCIA}.")
    else:
        print(f"O ponto ({px},{py},{pz}) NÃO pertence ao plano (diferença: {test_val:.6f} >
{TOLERANCIA}).")

    continuar = input('\nNovo plano? (y/n): ').strip().lower()

print('\nEncerrado.')
```

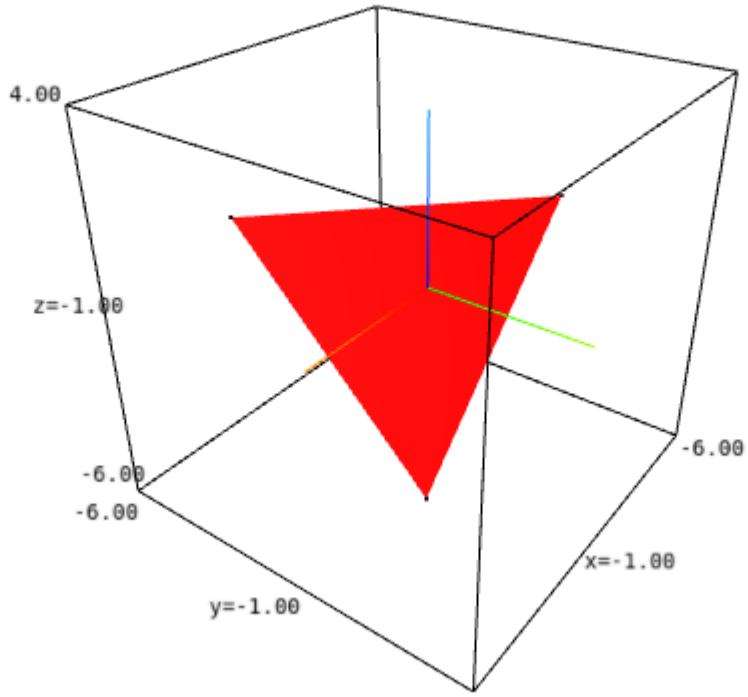
Out[4]:
```
Digite a,b,c,d: 1,1,1,6


Plano: (1.0)x + (1.0)y + (1.0)z + (6.0) = 0

Interseções:
A (eixo x): (-6.0, 0, 0)
```

```
B (eixo y): (0, -6.0, 0)
C (eixo z): (0, 0, -6.0)

Área do triângulo ABC: 31.1769 u²
Altura relativa ao lado AC: 7.3485 u
Volume do tetraedro OABC: 36.0 u³
```




```
Digite x do ponto... 2
Digite y do ponto... 1
Digite z do ponto... 10
O ponto (2.0,1.0,10.0) NÃO pertence ao plano (diferença: 19.000000 > 0.00100000000000000).

Novo plano? (y/n): n
```

```
Encerrado.
```