Relatório 3 - Manobras Orbitais (AGA0521)

Kethelin Parra Ramos - 9898349

I. INTRODUÇÃO

É comum a utilização do Sistema de Equatorial de Coordenadas na Astronomia. No caso do Sistema Equatorial Universal, utilizamos a declinação (δ) e ascensão reta (α), sendo que é comum o uso das siglas DEC e RA para representá-las. Dada as coordenadas cartesianas arbitrárias $\vec{r} = (x, y, z)$, é possível convertê-las para coordenadas equatoriais δ e α.

O versor $\hat{u}_r$ de um vetor $\vec{r}$ pode ser calculado a partir de:

$$\hat{u}_r = \frac{\vec{r}}{r}, \tag{1}$$

onde r é o módulo do vetor $\vec{r}$.

E a relação deste versor $\hat{u}_r$ com a ascensão reta e declinação é dada por:

$$\hat{u}_r = \cos\delta\cos\alpha\hat{i} + \cos\delta\sin\alpha\hat{j} + \sin\delta\hat{k}. \tag{2}$$

Por fim, igualando as equações 1 e 2, conseguimos determinar os valores de δ e α.

II. MÉTODOS

Para resolver os problemas propostos desta semana foi desenvolvido um único programa em Python. O código foi "quebrado" em duas partes para que todos os passos das atividades fossem descritos de forma satisfatória.

A atividade desta semana solicitou a elaboração de um programa que utilize a posição (x, y, z) de um corpo em órbita da Terra para determinar sua ascensão reta (α) e declinação (δ). Para isto, foi escrito as funções *Ref_ EquatorialGeo()* e *quadrantes()*, sendo esta última responsável em verificar a solução no quadrante correto. As funções foram testadas utilizando os dados da Tab. I e o código delas se encontra abaixo:

```
#Bibliotecas
import numpy as np

"""                                    Funções                                    """
#Função que avalia a solução correta (quadrante correto)
def quadrantes(uVersor,RA1,RA2,dec):
    #Calculo dos senos e cossenos para verificação do sinal
    seno_alfa = uVersor[1]/np.cos(np.radians(dec))
    cos_alfa = uVersor[0]/np.cos(np.radians(dec))

    #Primeiro quadrante
    if cos_alfa>0 and seno_alfa>0:
        if RA1>0 and RA1<90: #Se o RA1 estiver dentro do quadrante
           RA = RA1
        elif RA2>0 and RA2<90: #Se o RA2 estiver dentro do quadrante
           RA = RA2
        else: #Caso nenhum dos dois estejam no quadrante e um deles seja negativo
            if RA1<0:
               RA = 180+np.fabs(RA1)
            elif RA2<0:
               RA = 180+np.fabs(RA2)

    #Segundo quadrante
    if cos_alfa<0 and seno_alfa>0:
        if RA1>90 and RA1<180: #Se o RA1 estiver dentro do quadrante
            RA = RA1
        elif RA2>90 and RA2<180: #Se o RA2 estiver dentro do quadrante
            RA = RA2
        else: #Caso nenhum dos dois estejam no quadrante e um deles seja negativo
            if RA1<0:
               RA = 180+np.fabs(RA1)
            elif RA2<0:
               RA = 180+np.fabs(RA2)
    #Terceiro quadrante
    if cos_alfa<0 and seno_alfa<0:
        if RA1>180 and RA1<270:#Se o RA1 estiver dentro do quadrante
           RA = RA1
```

```
        elif RA2>180 and RA2<270:#Se o RA2 estiver dentro do quadrante
            RA = RA2
        else: #Caso nenhum dos dois estejam no quadrante e um deles seja negativo
            if RA1<0:
                RA = 180+np.fabs(RA1)
            elif RA2<0:
                RA = 180+np.fabs(RA2)

    #Quarto quadrante
    if cos_alfa>0 and seno_alfa<0:
        if RA1>270 and RA1<360:#Se o RA1 estiver dentro do quadrante
            RA = RA1
        elif RA2>270 and RA2<360:#Se o RA2 estiver dentro do quadrante
            RA = RA2
        else: #Caso nenhum dos dois estejam no quadrante e um deles seja negativo
            if RA1<0:
                RA = 180+np.fabs(RA1)
            elif RA2<0:
                RA = 180+np.fabs(RA2)
    return RA  #Retorna a solução correta

#Função que recebe as coordenadas (x,y,z) [km] e calcula a ascensão reta e declinação
def Ref_EquatorialGeo(coordenadas):
    #Calculo do modulo de r e do versor de r
    modulo_r=np.sqrt(coordenadas[0]**2 + coordenadas[1]**2 + coordenadas[2]**2)
    uVersor = coordenadas/modulo_r

    #Calculando a declinação e ascensão reta (rad)
    dec = np.arcsin(uVersor[2])
    RA1 = np.arccos(uVersor[0]/np.cos(dec))
    RA2 = np.arcsin(uVersor[1]/np.cos(dec))

    #Conversão para graus
    dec = np.degrees(dec)
    RA1 = np.degrees(RA1)
    RA2 = np.degrees(RA2)

    #Verifica qual é a solução correta
```

```
    RA = quadrantes(uVersor,RA1,RA2,dec)

    return RA,dec #Retorna as coordenadas ascensão reta e declinação em graus decimais
```

Tabela I: Coordenadas utilizada na atividade 3.

Questão	**Coordenadas (km)**
1	(-5368.0, -1784.0, 3691.0)
2	(-4834.0, 2500.0, 4000.0)

III. RESULTADOS

Os resultados do testes realizados com os dados descritos na Seção II se encontram na Tab.II e código utilizado se encontra abaixo:

```
"""                                 Programa Principal                                  """

#Questão 1 - Exemplo da aula
r = np.array([-5368.,-1784.,3691.]) #A posição da ISS em km
coordenadas = Ref_EquatorialGeo(r)  #Calculando as coordenadas equatoriais

print("Questão 1:")
print('RA = {:f}°'.format(coordenadas[0]))
print('Dec = {:f}° '.format(coordenadas[1]))


#Questão 2 - Enunciado
r = np.array([-4834.,2500.,4000.]) #A posição da ISS em km
coordenadas = Ref_EquatorialGeo(r) #Calculando as coordenadas equatoriais

print("\nQuestão 2:")
print('RA = {:f}°'.format(coordenadas[0]))
print('Dec = {:f}° '.format(coordenadas[1]))
```

Tabela II: Resultado dos cálculos da ascensão reta (α) e declinação (δ).

Questão	α (°)	δ (°)
1	198.383700	33.124543
2	152.653349	36.315763

IV. CONCLUSÃO

Como o resultado da questão 1 coincidiu com o exemplo de aula, conclui que o programa está funcionando corretamente.