

UNIVERSIDADE FEDERAL DO ABC



ENGENHARIA AEROESPACIAL

LABORATÓRIO DE GUIAGEM, NAVEGAÇÃO E CONTROLE

Relatório 1 - O Problema de Dois Corpos

Discente:

Arthur Sarri Binelli

RA: 21043816

João Victor Pinho Pizoni

RA: 11201920704

Rebeca Sales Ribeiro Alves

RA: 11201920408

Docente:

Prof. Leandro Baroni

São Bernardo do Campo
2025

Resumo

A relação dada exclusivamente pela atração gravitacional mútua de dois corpos é definida como problema de dois corpos. Para o caso deste estudo, será tomado como corpo principal e fixo a Terra e como corpo secundário um corpo que orbita a Terra. Através das simulações, objetivou-se observar a trajetória que um corpo descreve ao redor da Terra através de integrações numéricas utilizando o *software* desenvolvido em Python e a influência que os parâmetros de integração possuem nos resultados da integração bem como no custo computacional

Sumário

Capítulo 1 Introdução.....	4
1.1 Objetivos.....	4
Capítulo 2 Fundamentação Teórica.....	5
2.1 O Problema de Dois Corpos.....	5
2.2 Período Orbital.....	7
2.3 O Sistema Geocêntrico Equatorial Inercial.....	8
2.4 Modelo do Elipsóide Terrestre.....	9
Capítulo 3 Procedimento.....	10
3.1 Versionamento e abordagem.....	10
3.2 Integrador numérico.....	10
3.3 Fluxograma do Programa.....	12
Capítulo 4 Resultados e Discussões.....	13
4. Análise da simulação numérica.....	15
Capítulo 5 Conclusão.....	20
Referências Bibliográficas.....	21
Apêndice A.....	22
Apêndice B.....	28

Capítulo 1

Introdução

O problema de dois corpos estuda a dinâmica presente entre dois corpos sujeitos exclusivamente a atração gravitacional mútua e, para isso, utiliza a 2^a Lei de Newton em conjunto com a Lei da Gravitação Universal desprezando a influência de quaisquer outras forças sobre os corpos em questão, sendo assim um problema conservativo em termos de energia e momento angular. Além disso, o problema de dois corpos assume que os corpos são esféricos e possuem massa pontual localizada em seus centros de massa.

Uma das possíveis aplicações do problema de dois corpos é na mecânica celeste, visto que a análise de um corpo de massa pontual (secundário) orbitando ao redor de outro corpo de maior massa (corpo central), e também pontual, pode ser estendida às análises de órbitas de detritos, satélites e outros veículos espaciais ao redor de planetas, como a Terra, sob a influência gerada pelo campo gravitacional do corpo fixo de maior massa.

Além disso, o fato de o problema de dois corpos apresentar solução analítica possibilita a validação da resposta obtida através de cálculos numéricos, permitindo concluir se os resultados obtidos são satisfatórios ou não.

1.1 Objetivos

Este relatório tem por objetivos realizar a observação da órbita de um veículo espacial ao redor da Terra utilizando simulação numérica através do *software* em Python e discutir a influência dos parâmetros de integração nos fenômenos físicos representados pelos resultados da simulação.

Capítulo 2

Fundamentação Teórica

Este capítulo tem por objetivo introduzir os conceitos e fundamentos relacionados ao problema de dois corpos e a simulação de um veículo orbitando o planeta Terra.

2.1 O Problema de Dois Corpos

O problema de dois corpos estuda o movimento de dois corpos sujeitos apenas a força gravitacional entre eles, conforme mencionado no Capítulo 1. Para a solução deste tipo de problema, é considerado que um corpo com massa desprezível encontra-se sob a ação de um campo gravitacional gerado por uma massa pontual fixa.

Além disso, o problema de dois corpos possui um papel relevante na astronáutica pois:

1. É o único problema em astrodinâmica para o qual há uma solução analítica, com exceção o de casos particulares de problema de três corpos.
2. Uma grande variedade de problemas práticos podem ser tratados como problema de dois corpos.
3. O efeito de outros corpos podem ser tratados como uma perturbação desse sistema.

Diversas forças atuam sobre o veículo espacial (V/E), dentre elas: arrasto atmosférico, perturbação gravitacional do 3º corpo, impulso e pressão de radiação solar direta e indireta

Sendo assim, da 2ª Lei de Newton:

$$\sum \vec{F}_{ext} = m \vec{a} \quad (2.1)$$

Como a solução para tal problema é complexa, considera-se apenas a força gravitacional no sistema. A Lei da Gravitação Universal de Newton descreve a força gravitacional que um corpo de massa M exerce sobre outro e é dada pela equação (2.2):

$$\vec{F}_g = GMm/r^2 (-\hat{r}) \quad (2.1)$$

onde G é a constante da gravitação universal, M é a massa do corpo de maior massa e m a massa do corpo de menor massa. Além disso, r é a distância radial entre os corpos e o vedor $(-\hat{r})$ indica que a natureza da força gravitacional é radial e negativa devido à atração.

Dessa forma,

$$\sum \vec{F}_{ext} = \vec{F}_g = m \vec{a} \quad (2.3)$$

$$\therefore m \ddot{\vec{r}} = -\frac{GMm}{r^2} (\hat{r}) \quad (2.4)$$

$$\therefore \ddot{\vec{r}} - \frac{GM}{r^2} (\hat{r}) = 0 \quad (2.5)$$

A solução da equação (2.5) fornece a trajetória do veículo espacial. A solução analítica para o problema de dois corpos é a equação das cônicas em um sistema de coordenadas polares fixo ao corpo primário dada por:

$$r = p / (1 + e \cos(\theta)) \quad (2.6)$$

onde p é o *semilatus rectum* - parâmetro que varia de acordo com o formato da órbita, que pode ser circular, elíptica, parabólica ou hiperbólica -, e é a excentricidade e θ é o ângulo polar.

A solução numérica para o problema também pode ser obtida em coordenadas cartesianas a partir de um vetor inicial de posição, \vec{r} , e de velocidade, \vec{v} , no sistema geocêntrico inercial (que será apresentado na Seção 2.3).

Considerando que $\mu = GM$ e partindo da equação (2.4),

$$-\frac{\mu}{r^2} \frac{\vec{r}}{r} = -\frac{\mu}{r^3} \vec{r} = -\frac{\mu}{(x^2 + y^2 + z^2)^{3/2}} (x, y, z) \quad (2.7)$$

Dessa forma, tem-se um sistema de 3 equações diferenciais ordinárias (EDO) de 2ª ordem, cuja solução dá o movimento do V/E sujeito apenas à atração gravitacional terrestre.

Reescrevendo a equação (2.7) em termos das variáveis computacionais, tem-se o seguinte sistema de EDOs:

$$\begin{cases} \ddot{x}_1 = -\frac{\mu}{(x_1^2 + x_2^2 + x_3^2)^{3/2}} x_1 \\ \ddot{x}_2 = -\frac{\mu}{(x_1^2 + x_2^2 + x_3^2)^{3/2}} x_2 \\ \ddot{x}_3 = -\frac{\mu}{(x_1^2 + x_2^2 + x_3^2)^{3/2}} x_3 \end{cases} \quad (2.8)$$

2.2 Período Orbital

Para obter a evolução da órbita do veículo ao redor da Terra, é necessário definir um período de integração que, no caso, será o período orbital (T).

Sabe-se que a energia mecânica do sistema é dada pela soma da energia cinética com a potencial, conforme a equação (2.9):

$$E = \frac{1}{2}mv^2 - \frac{\mu m}{r} \quad (2.9)$$

$$\therefore \varepsilon = \frac{v^2}{2} + \frac{\mu}{r} \quad (2.10)$$

onde ε é a energia mecânica específica dada pela razão E/m .

Além disso, a energia mecânica específica pode ser descrita em função do semi-eixo maior da órbita, a , conforme a equação (2.11):

$$\varepsilon = -\frac{\mu}{2a} \quad (2.11)$$

Com isso é possível determinar o valor do semi-eixo maior da órbita:

$$a = -\frac{\mu}{2\varepsilon} \quad (2.12)$$

Sabendo que a constante gravitacional da Terra, G , vale $6,67 \times 10^{-11} Nm^2/kg^2$ e que a massa da Terra, M , vale $5,972 \times 10^{24} kg$, é possível determinar μ e, por consequência, o movimento médio do corpo, n , através da equação (2.13):

$$n = \sqrt{\frac{\mu}{a^3}} \quad (2.13)$$

Por fim, o período orbital é dado por:

$$T = \frac{2\pi}{n} \quad (2.14)$$

2.3 O Sistema Geocêntrico Equatorial Inercial

Ao se trabalhar com problemas que envolvem navegação e guiagem, é necessário que haja conhecimento dos sistemas de coordenadas envolvidos na trajetória do veículo, pois as quantidades variam de acordo com o sistema considerado.

Um dos sistemas que será considerado no problema de dois corpos é o Sistema Geocêntrico Equatorial Inercial, ou também *Earth Centered Inertial* (ECI), que possui a origem centrada na Terra e a notação dada por $Ox_iy_i z_i$. O eixo Ox_i aponta para o equinócio vernal, o eixo Oz_i é paralelo ao eixo de rotação da Terra e aponta para o Norte geográfico e, por fim, o eixo Oy_i completa o sistema destrógiro. A Figura 2.1 exemplifica a representação desse sistema de coordenadas.

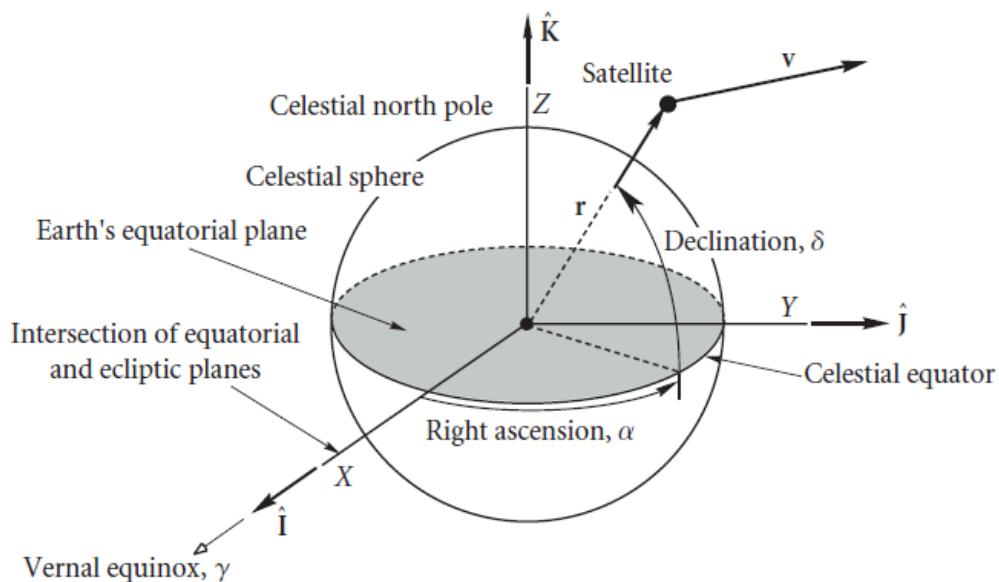


Figura 2.1: Representação do Sistema Geocêntrico Equatorial Inercial.

2.4 Modelo do Elipsóide Terrestre

Para a simulação, mostrada neste relatório, será utilizado um modelo simplificado de representação da Terra, visto que o planeta apresenta muitas irregularidades e complexidades. Dois possíveis modelos de elipsóides terrestres são: GRS80 e WGS84.

O GRS80 (*1980 Geodetic Reference System*) foi adotado na XVII Assembleia Geral da União Internacional de Geodesia e Geofísica (IUGG) em 1979 e foi originalmente usado pelo *World Geodetic System 1984* (WGS 84). O elipsóide de referência WGS84 agora difere ligeiramente do GRS80 devido a refinamentos.

O elipsóide utilizado no programa desenvolvido é o WGS84 com semieixo maior de 6378,137 km e semi eixo menor de 6356,752 km. A Figura 2.2 representa o elipsóide utilizado na simulação, cuja origem foi considerada como sendo $(x, y, z) = (0, 0, 0)$.

Terra WGS 84

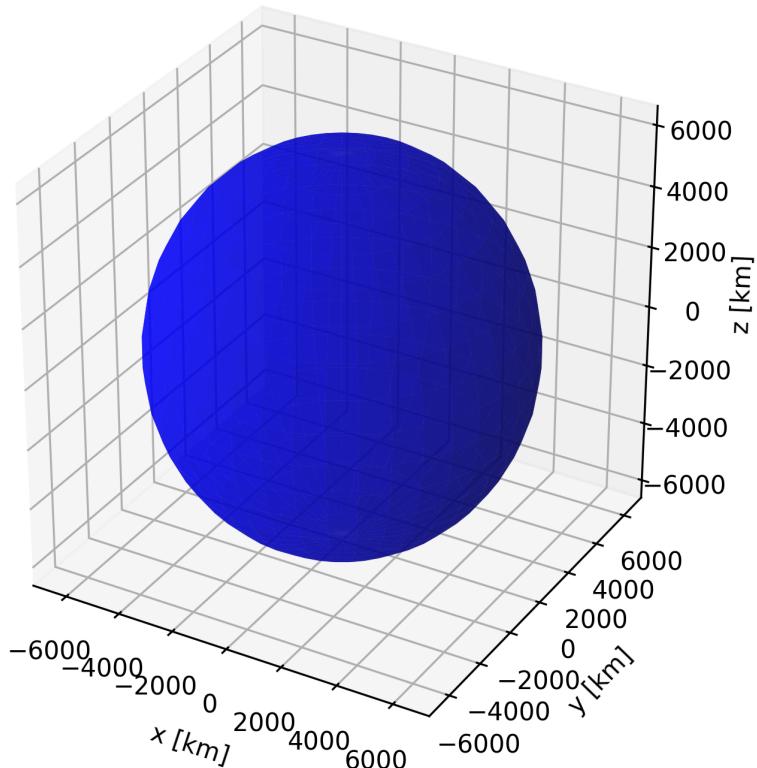


Figura 2.2: Modelo de elipsoide terrestre .

Capítulo 3

Procedimento

Este capítulo abordará a metodologia utilizada para descrever a órbita do veículo ao redor da Terra através de simulação numérica utilizando o integrador solve_ivp do *software* Python.

3.1 Versionamento e abordagem

Para o versionamento do projeto foi utilizado um repositório público no github, o qual contém um README.md contendo todas as indicações referente às etapas necessárias para execução e replicação da arquitetura desenvolvida no projeto.

- https://github.com/JoaoVPizoni/NAVEGACAO_GUIAGEM_CONTROLE_UFAB_C.git

3.2 Integrador numérico

Conforme mencionado, o integrador numérico utilizado na simulação da órbita é o solve_ivp. O solve_ivp é uma função que soluciona equações diferenciais ordinárias (ode, em inglês) e implementa o método de Runge-Kutta de ordem 4 com um intervalo de tempo variável (caso nenhum seja definido) para prover eficiência em termos computacionais. Matematicamente, o solve_ivp com argumento method='RK45' foi projetado para lidar com equações diferenciais ordinárias de 1^a ordem, conforme mostrado a seguir:

$$\frac{d\vec{x}}{dt} = \mathbf{f}(t, \vec{x}), \quad \vec{x}(0) = t_0 \quad (3.1)$$

em que t é a variável independente (tempo), \vec{x} é um vetor de variáveis dependentes que serão encontradas, $\mathbf{f}(t, x)$ é uma função de t e x e $x(0)$ são as condições iniciais do problema estudado no tempo inicial t_0 .

Para o problema de dois corpos, é necessário reduzir a ordem das EDOs (equação (2.8)) para que o solve_ivp seja capaz de calcular os vetores de posição e velocidade ao longo do tempo. Assim sendo, o sistema de 3 equações de 2^a ordem se tornará um sistema de 6

equações diferenciais ordinárias de 1^a ordem, que servirá de *input* para o integrador `solve_ivp`.

Dessa forma, ao considerar:

$$\begin{cases} x_1 = x \\ x_2 = y \\ x_3 = z \\ x_4 = \dot{x} \\ x_5 = \dot{y} \\ x_6 = \dot{z} \end{cases} \quad (3.2)$$

Tem-se:

$$\begin{cases} \dot{x}_1 = x_4 \\ \dot{x}_2 = x_5 \\ \dot{x}_3 = x_6 \\ \dot{x}_4 = -\frac{\mu}{(x_1^2 + x_2^2 + x_3^2)^{3/2}} x_1 \\ \dot{x}_5 = -\frac{\mu}{(x_1^2 + x_2^2 + x_3^2)^{3/2}} x_2 \\ \dot{x}_6 = -\frac{\mu}{(x_1^2 + x_2^2 + x_3^2)^{3/2}} x_3 \end{cases} \quad (3.3)$$

Com a redução de ordem das EDOs e com a aplicação das condições iniciais do vetor posição ($\rightarrow r = 10016, 34\hat{i} - 17012, 52\hat{j} + 7899, 28\hat{k}$ [km]) e do vetor velocidade ($\rightarrow v = 2, 50\hat{i} - 1, 05\hat{j} + 3, 88\hat{k}$ [km/s]) no sistema geocêntrico equatorial inercial é possível aplicar o método de integração numérica selecionando um intervalo de integração que será um múltiplo do período calculado na equação (2.14).

3.3 Fluxograma do Programa

Nesta seção é apresentado o raciocínio utilizado no desenvolvimento do programa que calcula a evolução da órbita do veículo ao redor da Terra (Figura 3.1).

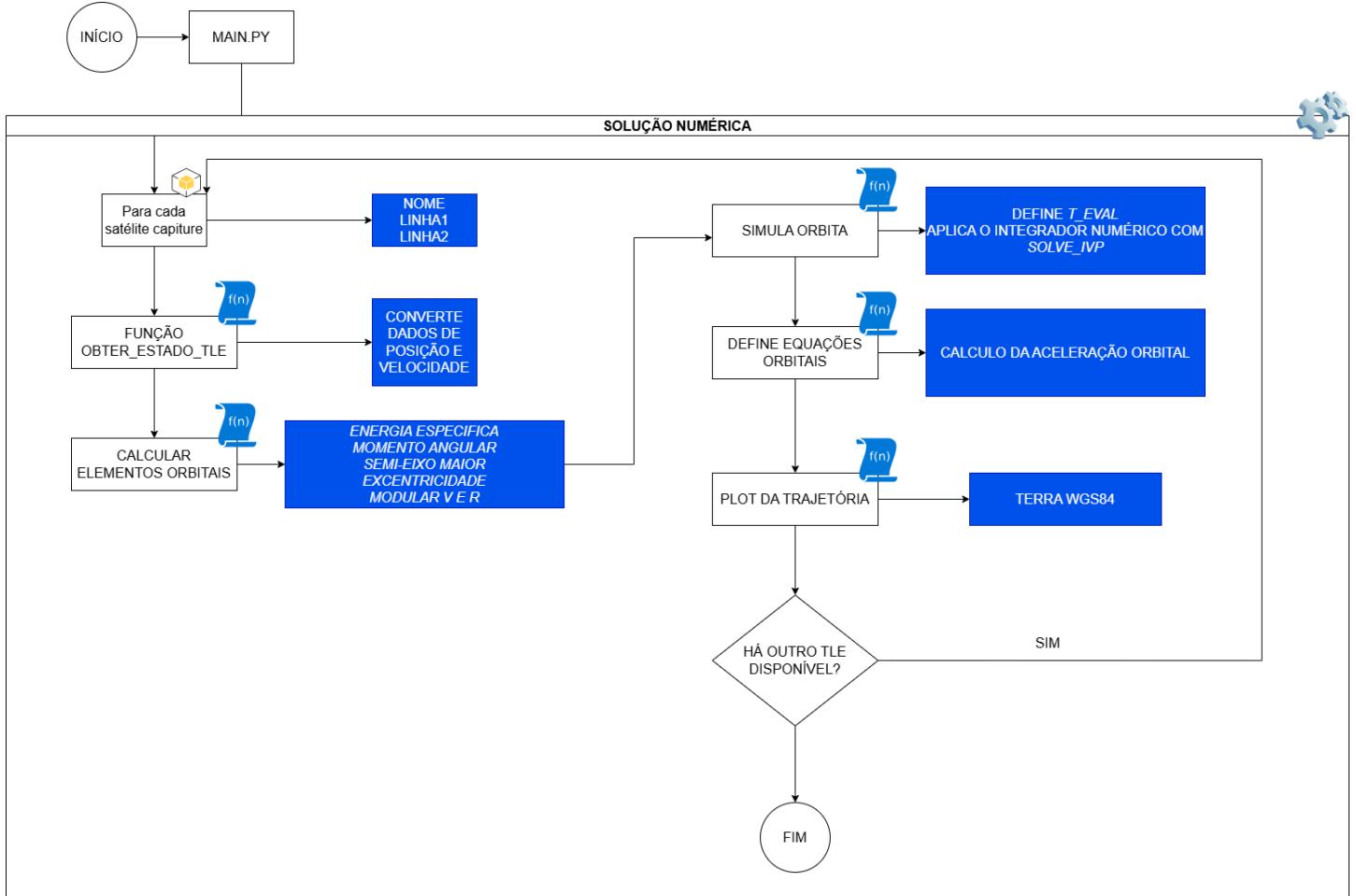


Figura 3.1: Fluxograma do programa.

Capítulo 4

Resultados e Discussões

Após a simulação, foram obtidos os seguintes resultados para os elementos orbitais de interesse.

Tabela 1: *Outputs* referentes aos elementos orbitais simulados.

Satélite	Energia (J)	Momento Angular (m^2/s)	Tipo de Órbita	Semi-eixo Maior (m)	Período (s)	Excentricidade
ISS	-29.351.015,89	[2.47e+10, -3.25e+10, 3.23e+10]	Elíptica	6.790.232,46	5.568,50	0.001472
CBERS-4A	-28.447.913,15	[-3.61e+10, 3.80e+10, -7.16e+09]	Elíptica	7.005.794,06	5.835,75	0.000772
MOLNIYA 1-91	-8.370.782,41	[-2.03e+10, -6.13e+10, 3.08e+10]	Elíptica	23.809.031,35	36.561,46	0.679013
STARONE D2	-4.726.580,14	[-4.67e+07, -6.48e+07, 1.30e+11]	Elíptica	42.165.839,78	86.169,21	0.000184

A análise dos dados orbitais dos satélites revela como as características de cada órbita estão diretamente relacionadas às finalidades operacionais dos veículos espaciais. Satélites como a ISS e o CBERS-4A operam em órbitas quase circulares e de baixa altitude, favorecendo observações constantes da Terra. Já o MOLNIYA 1-91, com alta excentricidade, é projetado para cobrir regiões de altas latitudes por longos períodos. Por fim, o STARONE D2 apresenta uma órbita geoestacionária com grande semi-eixo e período de 24 horas, ideal para comunicações contínuas. Assim, os parâmetros como energia, momento angular e período orbital evidenciam a diversidade de estratégias orbitais utilizadas conforme os objetivos de cada missão.

Inicialmente, foi feita a simulação aplicando uma tolerância absoluta ('AbsTol') na função 'simular_orbita' do integrador solve_ivp utilizando o método RK45 de 10^{-10} . O resultado pode ser visualizado na Figura 4.1.

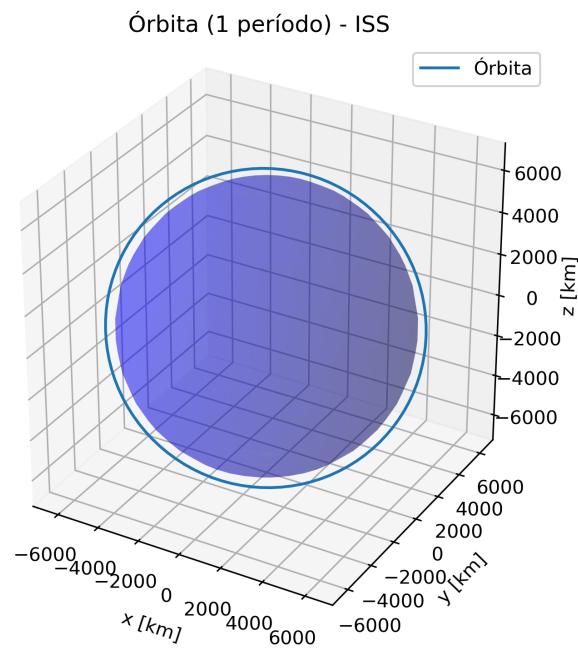


Figura 4.1: Representação de 1 órbita do V/E ao redor da Terra.

É válido afirmar que os parâmetros selecionados para a simulação estão longe de fornecer uma representação de qualidade para o movimento do corpo orbital. A resposta insatisfatória é amplificada conforme o veículo órbita mais vezes a Terra (Figura 4.2).

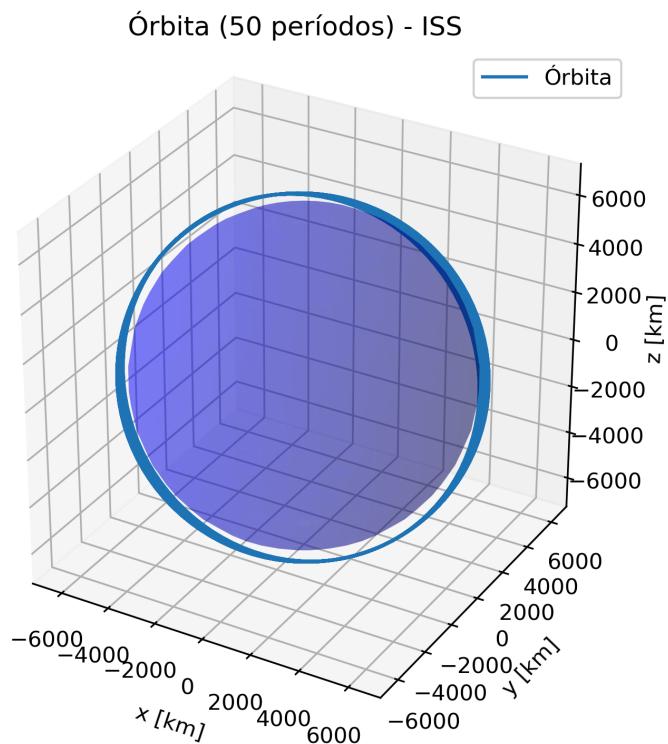


Figura 4.2: Representação de 50 órbitas do V/E ao redor da Terra.

4. Análise da simulação numérica

A simulação das órbitas demonstrou trajetórias compatíveis com os dados fornecidos pelos TLEs dos satélites analisados. As órbitas representadas em 3D refletem com precisão os elementos orbitais, como o semi-eixo maior, a inclinação e a excentricidade. Satélites como o ISS e o CBERS-4A apresentaram órbitas quase circulares e de baixa altitude, condizentes com suas respectivas missões — observação terrestre e tripulação humana. Por outro lado, o MOLNIYA 1-91 revelou uma órbita altamente excêntrica e alongada, característica de satélites projetados para cobrir regiões de altas latitudes, como os polos.

A modelagem tridimensional da Terra oblata, aliada à representação das trajetórias orbitais, permitiu observar com clareza o impacto da altitude e da inclinação das órbitas. A simulação de múltiplos períodos orbitais evidenciou a estabilidade dinâmica das trajetórias sob a influência da gravidade terrestre ajustada pelo termo de achatamento (J2). Todos os resultados observados foram fisicamente possíveis e consistentes com a mecânica orbital clássica. A energia total negativa indicou órbitas ligadas ao campo gravitacional da Terra, e os valores de excentricidade explicaram adequadamente as diferenças nas formas das órbitas.

Quanto ao aspecto numérico, foi utilizado o método de Runge-Kutta de ordem 4/5 por meio da função `solve_ivp`, uma escolha robusta para simulações orbitais. Apesar disso, em simulações prolongadas ou envolvendo órbitas muito excêntricas, erros numéricos e arredondamentos podem se acumular, resultando em pequenas distorções nas trajetórias. O uso de um passo fixo baseado no período orbital pode não ser ideal para todas as fases do movimento, especialmente em perigeus muito curtos ou apogeus muito distantes. Ajustes adaptativos no passo de integração e precisão dos parâmetros de entrada podem melhorar os resultados e a fidelidade da simulação.

Seguem os modelos 3D com o mesmo procedimento realizado para o CBERS-4A, Molniya 1-91 e o Starone D2 respectivamente:

Órbita (1 período) - CBERS-4A

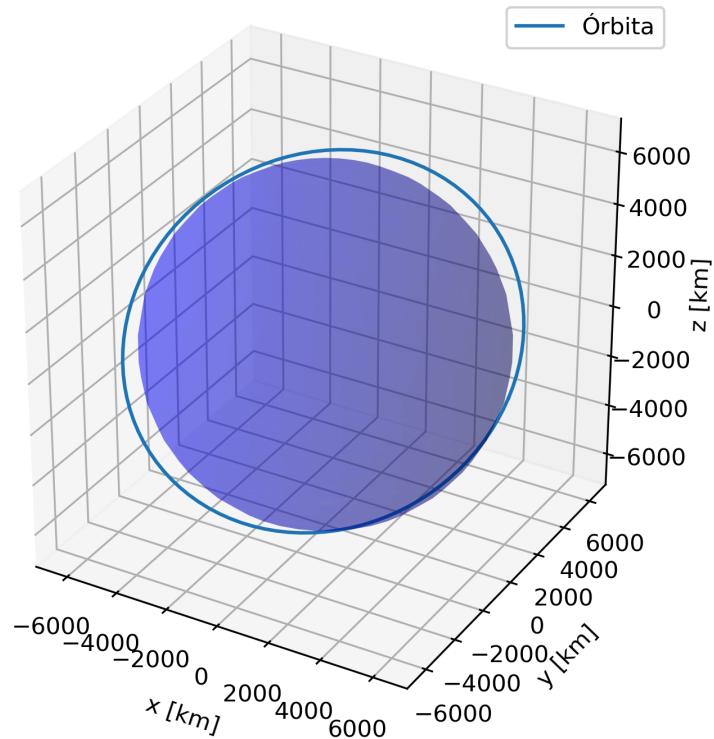


Figura 4.3: Representação de 1 órbita para CBERS-4A.

Órbita (50 períodos) - CBERS-4A

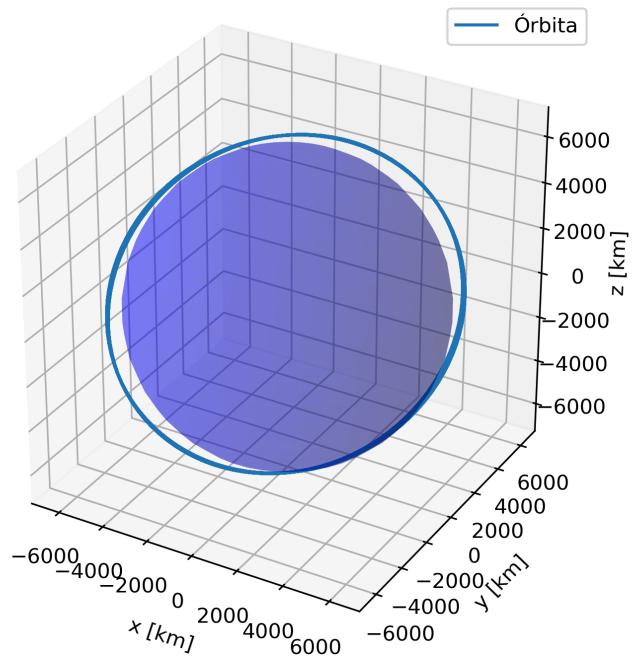


Figura 4.4: Representação de 50 órbitas para CBERS-4A.

Órbita (1 período) - MOLNIYA 1-91

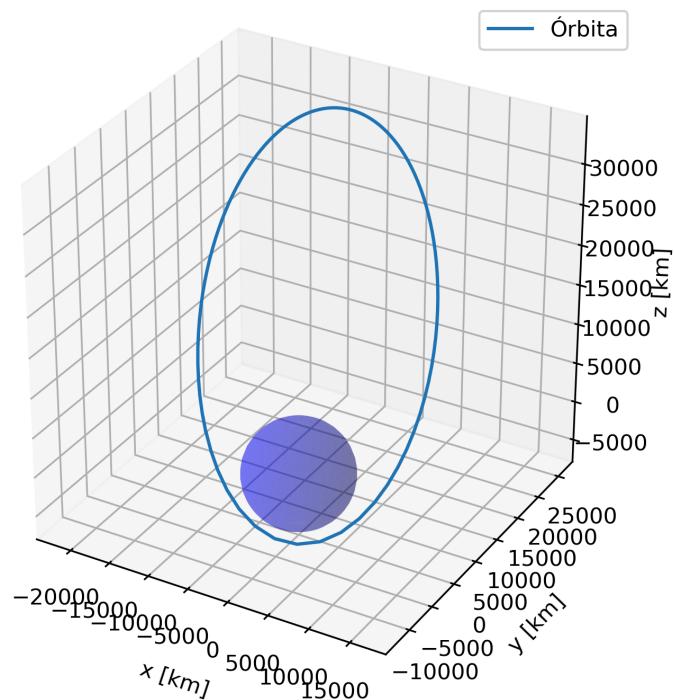


Figura 4.5: Representação de 1 órbita para o Molniya 1-91.

Órbita (50 períodos) - MOLNIYA 1-91

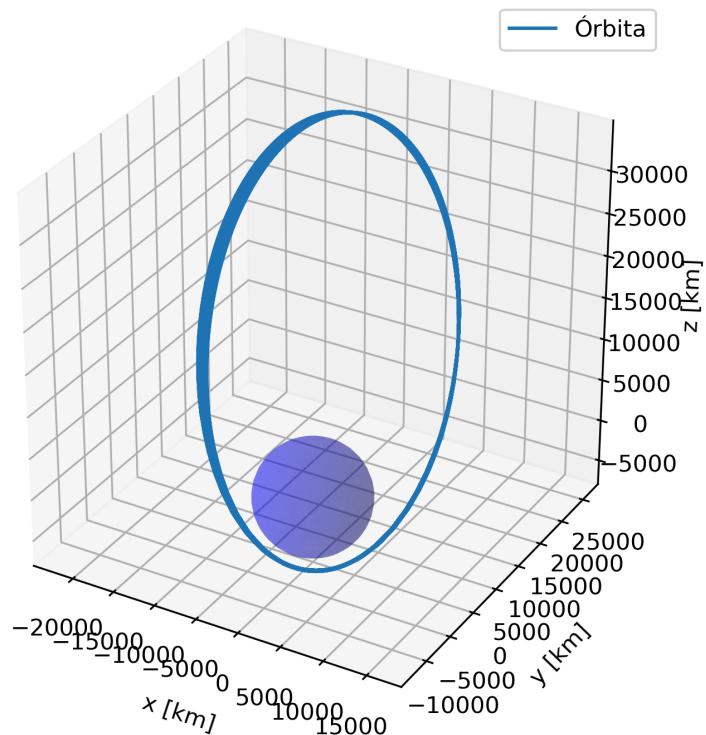


Figura 4.6: Representação de 50 órbitas para o Molniya 1-91.

Órbita (1 período) - STARONE D2

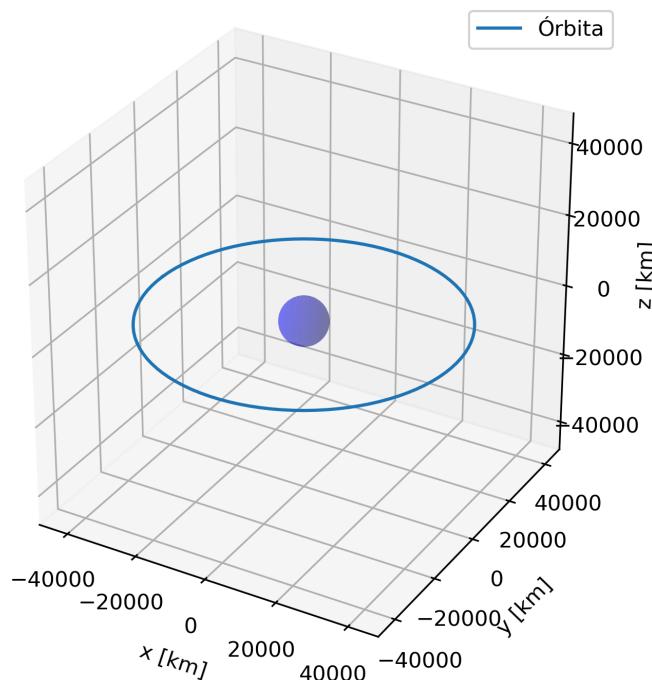


Figura 4.7: Representação de 1 órbita para o Starone D2.

Órbita (50 períodos) - STARONE D2

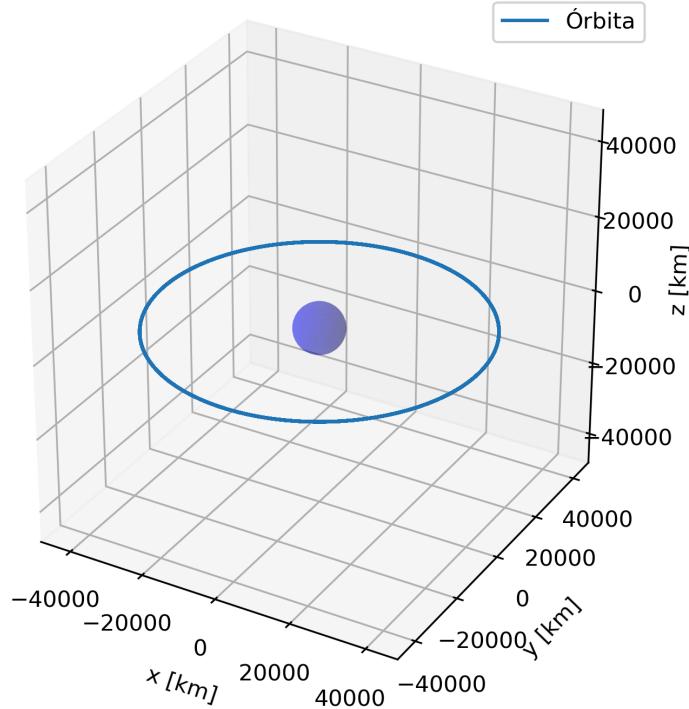


Figura 4.8: Representação de 50 órbitas para o Starone D2.

Vale ressaltar ainda que a solução numérica está de acordo com a solução analítica, visto que a órbita descreve uma elipse em que os pontos de início e fim do movimento orbital coincidem. A órbita visualizada é altamente elíptica, característica de satélites usados para transmissão de televisão, telecomunicações, comunicações, comunicações militares e monitoramento do tempo.

Capítulo 5

Conclusão

A atividade proposta permite concluir que, ao se trabalhar com simulações numéricas, os erros sempre estão presentes. Como forma de mitigar a presença de erros que afetem a observação do fenômeno físico, alguns parâmetros do integrador podem ser variados, como é o caso da tolerância e do passo de integração. Por outro lado, foi visto que há um *tradeoff* entre a qualidade da resposta obtida e o custo computacional exigido, isto é, apesar de valores menores de passo de integração e de tolerância possibilitarem respostas mais representativas, consomem também mais memória computacional, trazendo, portanto, a necessidade de avaliar a resposta com base em critérios físicos para saber se os resultados obtidos são satisfatórios.

No caso de um modelo que não considera a influência do arrasto atmosférico e de quaisquer outras forças que não a da atração gravitacional entre os dois corpos, é esperado que o semi-eixo maior da órbita não varie com o tempo, pois a variação do semi-eixo maior indicaria uma variação de energia no sistema, não condizente com a natureza conservativa do modelo representado.

Referências Bibliográficas

- [1] **CURTIS**, H. D. Orbital mechanics for engineering students. [S. l.]: Butterworth-Heinemann, 2013.
- [2] **CELESTINO**, C. C. *O Problema de Dois Corpos*.
- [3] **CELESTINO**, C. C. Dinâmica Orbital.
- [4] **ADCS** for Beginners. Earth-Centred Inertial Frame. Disponível em: <https://adcsforbeginners.wordpress.com/tag/earth-centred-inertial-frame/>. Acesso em: 15 de junho de 2024.
- [5] **Wikipedia**. Geodetic Reference System 1980. Disponível em: https://en.m.wikipedia.org/wiki/Geodetic_Reference_System_1980. Acesso em: 18 de junho de 2024.
- [6] **Space Legal Issues**. The Molniya Orbit and Satellites. Disponível em: <https://www.spacelegalissues.com/the-molniya-orbit-and-satellites/>. Acesso em: 12 de junho de 2024.

Apêndice A

Código em Python - Simulação Completa

```
from sgp4.api import Satrec, jday
from datetime import datetime
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt
import os

# Constantes - MODELO WGS 84
mu = 3.986004418e14 # [m³/s²] - Terra
req = 6378.137 # [km]
rp = 6356.752 # [km]
f = (req - rp) / req # Achatamento
e = np.sqrt(1 - (rp/req)**2) #Excentricidade

# -----
# Função para obter posição e velocidade de um TLE usando sgp4
def obter_estado_tle(linha1, linha2):
    sat = Satrec.twoline2rv(linha1, linha2)
    ano, mes, dia, hora, minuto, segundo = 2025, 5, 30, 0, 0, 0 # data de referência
    jd, fr = jday(ano, mes, dia, hora, minuto, segundo)
    e, r, v = sat.sgp4(jd, fr)
    if e != 0:
        raise RuntimeError("Erro ao processar TLE")
    return np.array(r) * 1000, np.array(v) * 1000 # posição e velocidade [m, m/s]

# -----
# EDOs do problema de dois corpos
def equacoes_orbitais(t, x):
    r_vec = x[:3]
    v_vec = x[3:]
    r = np.linalg.norm(r_vec)

    # Termo de J2 (achatamento da Terra)
    J2 = 1.08263e-3
    R = req * 1000 # converte para metros
    x_, y_, z_ = r_vec
```

```

fator_J2 = (3/2) * J2 * mu * R**2 / r**5
ax_J2 = fator_J2 * x_ * (1 - 5 * (z_**2) / r**2)
ay_J2 = fator_J2 * y_ * (1 - 5 * (z_**2) / r**2)
az_J2 = fator_J2 * z_ * (3 - 5 * (z_**2) / r**2)

a_vec = -mu * r_vec / r**3 + np.array([ax_J2, ay_J2, az_J2])

return np.concatenate((v_vec, a_vec))

# -----
# Resolução via solve_ivp
def simular_orbita(estado_inicial, T, passo, titulo, nome_sat):
    num_pontos = int(np.ceil(T / passo)) + 1
    t_eval = np.linspace(0, T, num_pontos)
    sol = solve_ivp(equacoes_orbitais, [0, T], estado_inicial, t_eval=t_eval,
                    method='RK45', atol=1e-10, rtol=1e-8) #RANGE KUTTA 45
    plotar_orbita(sol, titulo, nome_sat)

# -----
# Plotagem da órbita e da Terra
def plotar_orbita(sol, titulo, nome_sat):
    import os
    pasta = "Solucao_Problema_Proposto/Dois_Corpos/figuras"
    os.makedirs(pasta, exist_ok=True)

    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    x = sol.y[0] / 1000
    y = sol.y[1] / 1000
    z = sol.y[2] / 1000
    ax.plot(x, y, z, label='Órbita')

    u, v = np.mgrid[0:2*np.pi:40j, 0:np.pi:20j]
    xe = req * np.cos(u) * np.sin(v)
    ye = req * np.sin(u) * np.sin(v)
    ze = rp * np.cos(v)
    ax.plot_surface(xe, ye, ze, color='b', alpha=0.3)

    igualar_escalas(ax)
    ax.set_title(titulo)
    ax.set_xlabel("x [km]")

```

```

ax.set_ylabel("y [km]")
ax.set_zlabel("z [km]")
ax.legend()
ax.set_box_aspect([1, 1, 1])
plt.tight_layout()

# Salvar figura
nome_arquivo = os.path.join(pasta, f"{nome_sat.replace(' ', '_')}.png")
plt.savefig(nome_arquivo, dpi=300)
print(f"Figura salva em: {nome_arquivo}")

# Mostrar na tela
plt.show()
plt.close(fig)

# -----
# Igualar escalas para evitar distorção visual
def igualar_escalas(ax):
    xlim = ax.get_xlim3d()
    ylim = ax.get_ylim3d()
    zlim = ax.get_zlim3d()

    x_range = abs(xlim[1] - xlim[0])
    y_range = abs(ylim[1] - ylim[0])
    z_range = abs(zlim[1] - zlim[0])

    max_range = max(x_range, y_range, z_range)

    x_middle = np.mean(xlim)
    y_middle = np.mean(ylim)
    z_middle = np.mean(zlim)

    ax.set_xlim3d([x_middle - max_range/2, x_middle + max_range/2])
    ax.set_ylim3d([y_middle - max_range/2, y_middle + max_range/2])
    ax.set_zlim3d([z_middle - max_range/2, z_middle + max_range/2])

# -----
# Trajetória de acordo com a Energia Mecânica Específica e Excentricidade
def consultar_trajetoria(epsilon, e):
    if epsilon<0 and e==0:
        return 'Circular'
    elif epsilon<0:
        return 'Elíptica'

```

```

    elif epsilon==0:
        return 'Parabólica'
    else:
        return 'Hiperbólica'

# -----
# Simular para um satélite
def executar_para_tle(nome, linha1, linha2):
    print(f"Simulando para: {nome}")
    r, v = obter_estado_tle(linha1, linha2)
    estado_inicial = np.concatenate((r, v))
    mod_r = np.linalg.norm(r)
    mod_v = np.linalg.norm(v)
    epsilon = (mod_v**2) / 2 - mu / mod_r #Energia Mecânica Específica
    a = -mu / (2 * epsilon) # semi-eixo maior
    T = 2 * np.pi * np.sqrt(a**3 / mu) # período orbital
    h = np.cross(r, v) # Momento Angular Específico
    e_vec = np.cross(v, h) / mu - r / mod_r
    e = np.linalg.norm(e_vec) #Excentricidade da Órbita
    print(f"Energia Mecânica Específica (J): {epsilon:.2f}")
    print(f"Momento Angular Específico (m²/s): {h}")
    print(f"Posível Trajetória: {consultar_trajetoria(epsilon, e)}")
    print(f"Semi-eixo maior (m): {a:.2f}")
    print(f"Período orbital (s): {T:.2f}")
    print(f"Excentricidade: {e:.6f}")

    passo = T * 0.01

    simular_orbita(estado_inicial, T, passo, f"Órbita: {nome}", nome)

    # Simulação para 1 período orbital
    simular_orbita(estado_inicial, T, passo, f"Órbita (1 período) - {nome}",
    f"{nome}_1periodo")

    # Simulação para 50 períodos orbitais
    simular_orbita(estado_inicial, 50*T, passo, f"Órbita (50 períodos) - {nome}",
    f"{nome}_50periodos")

# -----
# TLEs dos satélites
satelites = {

```

```

    "ISS": (
        "1 25544U 98067A   25150.54603503 .00012878 00000-0 23439-3 0 9999",
        "2 25544   51.6399  34.4830 0002197 166.0379 262.3840 15.49859072512427"
    ) ,
    "CBERS-4A": (
        "1 44883U 19093E   25150.83982066 .00001032 00000-0 13795-3 0 9998",
        "2 44883   97.7932 224.3455 0001774   6.2371 353.8864 14.81582034294467"
    ) ,
    "MOLNIYA 1-91": (
        "1 25485U 98054A   25150.46666369 -.00000126 00000-0 00000-0 0 9999",
        "2 25485   64.4919 341.5937 6788400 287.6338 12.7988 2.36440192204520"
    ) ,
    "STARONE D2": (
        "1 49055U 21069A   25150.42901189 -.00000270 00000-0 00000+0 0 9997",
        "2 49055   0.0110 322.8887 0001735 134.5547 235.1911 1.00271589 14066"
    )
}

# -----
# Rodar simulações para todos
if __name__ == "__main__":
    for nome, (l1, l2) in satelites.items():
        executar_para_tle(nome, l1, l2)

#-----
# Fluxograma de execução
#
# [ Início (main) ]
#
#   |
#   v
#
# [ Para cada satélite
#   (nome, linha1, linha2) ]
#
#   |
#   v
#
# [ obter_estado_tle()
#   - Converte TLE para
#     posição e velocidade ]
#
#   |
#   v
#
# [ Calcular:

```

```
# | - módulo de r e v
# | - energia específica
# | - semi-eixo maior (a)
# | - período orbital (T)
# | - Excentricidade
# | - Momento Angular
#
# |   ↓
#
# |   ↓
#
# |   ↓
# | simular_orbita()
# | - define t_eval
# | - resolve EDO com solve_ivp
#
# |   ↓
# |   ↓
#
# |   ↓
# | equacoes_orbitais()
# | - calcula aceleração
# |   gravitacional (-μr/r³)
# | - (opcional: termo J2)
#
# |   ↓
# |   ↓
#
# |   ↓
# | plotar_orbita()
# | - Plota órbita 3D
# | - Plota Terra oblata
# | - Salva como imagem
#
# |   ↓
# |   ↓
#
# |   ↓
# | Fim do loop
# | (volta ao próximo satélite)
#
# |   ↓
#
# |   ↓
# | Fim do programa
```

Apêndice B

Código em Python - Terra WGS 84

```
import matplotlib.pyplot as plt
import numpy as np
import os

# Constantes - MODELO WGS 84
mu = 3.986004418e14 # [m³/s²] - Terra
req = 6378.137 # [km]
rp = 6356.752 # [km]
f = (req - rp) / req # Achatamento
e = np.sqrt(1 - (rp/req)**2) #Excentricidade

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

u, v = np.mgrid[0:2*np.pi:40j, 0:np.pi:20j]
xe = req * np.cos(u) * np.sin(v)
ye = req * np.sin(u) * np.sin(v)
ze = rp * np.cos(v)
ax.plot_surface(xe, ye, ze, color='b', alpha=0.65)
ax.set_title("Terra WGS 84")
ax.set_xlabel("x [km]")
ax.set_ylabel("y [km]")
ax.set_zlabel("z [km]")
ax.set_box_aspect([1, 1, 1])
plt.tight_layout()

# Salvar figura
pasta = "Solucao_Problema_Proposto/Dois_Corpos/figuras"
os.makedirs(pasta, exist_ok=True)
nome_arquivo = os.path.join(pasta, f"TERRA_WGS84.png")
plt.savefig(nome_arquivo, dpi=300)
print(f"Figura salva em: {nome_arquivo}")

plt.show()
```