Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Departamento de Matemática

Métodos Numéricos Aplicados a Dinâmica Orbital

Autor: *Lucas Vinicius Antunes*

Orientador: *Sávio Brochini Rodrigues*

Disciplina: Trabalho de Conclusão do Curso B

Curso: Licenciatura em Matemática

Professores Responsáveis: Adriana Ramos Pereira
Luis Antonio Carvalho dos Santos
Wladimir Seixas

São Carlos, 29 de novembro de 2021.

Métodos Numéricos Aplicados a Dinâmica Orbital

Autor: *Lucas Vinicius Antunes*

Orientador: *Sávio Brochini Rodrigues*

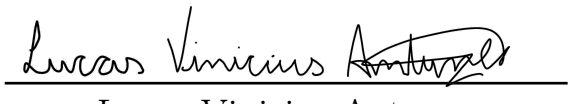
Disciplina: Trabalho de Conclusão do Curso B

Curso: Licenciatura em Matemática

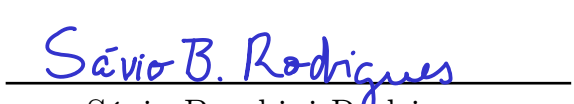
Professores Responsáveis: Adriana Ramos Pereira
Luis Antonio Carvalho dos Santos
Wladimir Seixas

Instituição: Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Departamento de Matemática

São Carlos, 29 de novembro de 2021.



Lucas Vinicius Antunes



Sávio Brochini Rodrigues

Dedico esse trabalho à todas as pessoas que me apoiaram e motivaram em todo meu percurso na graduação.

Agradecimentos

Gostaria de agradecer primeiramente ao Professor Doutor Sávio Brochini Rodrigues, que como meu orientador me deu todo apoio necessário para a realização desse trabalho, trazendo criticas importantes, agradeço também pela paciência e disponibilidade.

Aos meus amigos e familiares também sou muito grato por todo apoio, incentivo e ajuda que recebi dentro do que estava ao alcance de cada um. Em especial, gostaria de deixar um agradecimento ao meu caro amigo Pã Montenegro, por ter se disposto voluntariamente a realizar a revisão ortográfica deste trabalho.

Gostaria de deixar um agradecimento especial ao Carlos Eduardo Rufino, que como um dos autores de um artigo base deste trabalho, em um momento decisivo foi muito solicito em ajudar com duvidas relacionadas a um dos processos presentes no trabalho.

Resumo

Nesta dissertação temos como objetivo estudar um pouco sobre aplicações de métodos numéricos na área de dinâmica orbital, dentre eles os principais são o método de Runge-Kutta de 4ª ordem e a depuração via solução manufaturada.

Será realizada uma breve abordagem da relação de força entre dois corpos no espaço, uma vez que ela será utilizada posteriormente para a modelagem do problema principal.

Além disso, trabalharemos com algoritmos desenvolvidos em Python, buscando aplicar os conceitos estudados resolvendo e depurando uma EDO (Equação Diferencial Ordinária) e, ao final, será apresentada uma simulação orbital entre os corpos Sol-Terra-Lua.

Sumário

Lista de Figuras

Lista de Tabelas

Prefácio

A exploração espacial vem crescendo a cada dia mais e mais, com isso, as demandas tecnológicas aumentam proporcionalmente e consequentemente o uso de métodos numéricos nessas tecnologias se tornam vitais para o sucesso das missões espaciais, seja para realizar simulações e encontrar melhores órbitas de acordo com os objetivos, ou para realizar manobras orbitais ou ainda, até mesmo para o desenvolvimento dos processos autônomos dos equipamentos, visando diminuir ao máximo a necessidade de interferências externas por meio de comunicações com o equipamento.

Para que tudo isso seja passível de ocorrer é preciso a combinação de mais de um método numérico. Neste trabalho, dividido em 4 capítulos, abordaremos alguns deles visando desenvolver uma simulação ao final.

A princípio iniciaremos com uma abordagem sobre o método de Runge-Kutta (R-K), realizando uma breve demonstração do método de segunda ordem, utilizando as expansões de Taylor para uma e duas variáveis. Em seguida, o método de Runge-Kutta de quarta ordem será apresentado mas não demonstrado, por conta de sua complexidade que não caberia no tempo hábil deste trabalho.

Ainda no primeiro capítulo, será feita uma aplicação dos métodos R-K de ordem 2 e 4, no mesmo PVI (Problema de Valor Inicial) e com mesmo passo, buscando mostrar na prática a diferença na velocidade de convergência entre os dois e assim justificando a escolha do método de ordem 4 para o uso em nosso problema final.

Em seguida, no capítulo 2, abordaremos o método de verificação via solução manufaturada, o qual é importante de ser utilizado por verificar se a configuração dos algoritmos numéricos foi feita corretamente. De forma simplificada, o método insere no algoritmo do PVI, uma função de solução exata que, por sua vez, ao ser extraída do resultado final obtido deixará os resquícios do erro de aproximação, e este estará diretamente relacionado ao passo usado para a aproximação, de modo que, poderemos calcular e conferir a ordem numérica para o método.

O capítulo 3 apresentará uma breve abordagem da relação de força entre dois corpos no espaço e, em seguida, apresentaremos um algoritmo genérico dessa relação, buscando realizar uma primeira implementação de um problema de dois corpos, apenas para fins de testagem, para que posteriormente possamos adaptar a modelagem do problema real.

No capítulo seguinte uniremos todos os conceitos numéricos vistos e testados até então,

a fim de modelar um problema de três corpos bastante conhecido, que neste caso será o problema Sol-Terra-Lua, onde, por meio de processos de adimensionalização e redução de ordem das relações de força, adaptaremos os dados ao algoritmo desenvolvido no capítulo anterior, combinando com o algoritmo desenvolvido em [7] e, ao final teremos as 3 imagens representando as órbitas da Terra e da Lua individualmente e de forma plana e também, as órbitas desses mesmos dois corpos, combinadas no espaço, além de algumas outras representações que achamos de relevante inserção.

Por fim, apresentaremos uma breve conclusão deste trabalho, retomando os métodos que aqui foram estudados e ressaltando mais uma vez a importância que eles têm dentro dos estudos da dinâmica orbital e da exploração espacial.

Capítulo 1

Métodos Numéricos

1.1 Método de Runge-Kutta

Desenvolvido por Carl David Tolmé Runge e Martin Wilhelm Kutta, o método conhecido pela junção de seus sobrenomes, Runge-Kutta, tem o intuito de "imitar" o método de séries de Taylor, com o diferencial de que não seja necessário realizar derivações analíticas da EDO (Equação Diferencial Ordinária) original. Para que resolvêssemos um PVI (Problema de Valor Inicial) do tipo

$$\begin{cases} x' = f(t,x) \\ x(a) = x_a \end{cases}, \tag{1.1}$$

a partir do método de Taylor seria preciso encontrar as derivadas $x', x'', ...$ da função f. Essa limitação pode se tornar um problema ao usar esse método, uma vez que os cálculos das derivadas podem ser bastante trabalhosos, aumentando as chances de erro e também aumentando o tempo gasto para preparação de um possível código. Neste trabalho iremos supor que tanto a função $f(t,x)$ como a função $x(t)$ tem infinitas derivadas contínuas em um intervalo de tempo $[a, a+T]$ para $T > 0$ suficientemente grande para uma simulação.

Para fins de exposição *a priori* apresentaremos o método de Runge-Kutta de ordem 2, que apesar de ter uma baixa precisão para o uso científico, se torna uma interessante ferramenta para compreender melhor o método de ordem superior, no caso, o Runge-Kutta de ordem 4, que por sua vez será apresentado posteriormente.

1.1.1 Série de Taylor Para Funções de Duas Variáveis

Antes de adentrarmos na apresentação do método de Runge-Kutta de ordem 2, é importante discutirmos a série de Taylor para funções de duas variáveis, pois será importante

para sua demonstração. A série infinita é dada da seguinte forma:

$$f(x+h, y+k) = \sum_{i=0}^{\infty} \frac{1}{i!} \left(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y} \right)^i f(x,y), \tag{1.2}$$

sendo ela análoga à série de Taylor para funções de uma variável, descrita por

$$f(x+h) = \sum_{k=0}^{n} \frac{f^k(x)}{k!} h^k + E_{n+1},$$

onde

$$E_{n+1} = \frac{f^{n+1}(\bar{x})}{(n+1)!} h^{n+1}$$

para algum $\bar{x} \in [x, x+h]$, onde f tenha $n+1$ derivadas contínuas.

Em 1.2 os termos dentro dos parênteses são interpretados da seguinte forma

$$\begin{aligned}
&\left(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y} \right)^0 f(x,y) = f, \\
&\left(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y} \right)^1 f(x,y) = h\frac{\partial f}{\partial x} + k\frac{\partial f}{\partial y}, \\
&\left(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y} \right)^2 f(x,y) = h^2\frac{\partial^2 f}{\partial x^2} + 2kh\frac{\partial^2 f}{\partial x \partial y} + k^2\frac{\partial^2 f}{\partial y^2}, \\
&\qquad\qquad \vdots
\end{aligned}$$

onde as derivadas parciais de f estão em função de (x, y).

Assim como no caso da série para funções de uma variável, se a série de Taylor para funções de duas variáveis for truncada, é preciso inserir um termo para representar o erro de aproximação e assim manter a igualdade. Sendo assim, a equação deverá ficar da seguinte forma:

$$f(x+h, y+k) = \sum_{i=0}^{n-1} \frac{1}{i!} \left(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y} \right)^i f(x,y) + \frac{1}{n!} \left(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y} \right)^n f(\bar{x}, \bar{y}). \tag{1.3}$$

O ponto $(\bar{x}, \bar{y})$ pertence ao segmento que une os pontos (x, y) e $(x+h, y+k)$ no plano.

Visando adotar uma notação menos carregada, utilizaremos as seguintes alterações para representar as derivadas parciais:

$$f_x = \frac{\partial f}{\partial x} \qquad\qquad f_t = \frac{\partial f}{\partial t} \tag{1.4}$$

1.1.2 Método de Runge-Kutta de Ordem 2

No método de Runge-Kutta de ordem 2 sua fórmula contém dois tipos especiais de função como mostra o teorema a seguir.

Teorema 1.1. *Considerando um PVI da forma apresentada em 1.1 o valor $\eta(t+h)$ calculado através da seguinte equação:*

$$\eta(t+h) = x(t) + \frac{1}{2}(K_1 + K_2),$$

onde

$$\begin{cases} K_1 = hf(t,x) \\ K_2 = hf(t+h, x+K_1), \end{cases}$$

é tal que $x(t+h) = \eta(t+h) + O(h^3)$.

Observação: Dizemos que o *erro de truncamento local* é de ordem 3, e dizemos que o *erro do método*, também chamado de erro global, é de ordem 2, uma unidade a menos que o erro de truncamento local. Ou seja, $\eta(t+h)$ é uma aproximação de ordem 2 para $x(t+h)$.

Demonstração. No método de Runge-Kutta de ordem 2 utilizamos a combinação de duas formas especiais da função, sendo elas

$$\begin{cases} K_1 = hf(t,x) \\ K_2 = hf(t+\alpha h, x+\beta K_1), \end{cases}$$

que ao realizar uma combinação linear entre elas obteremos

$$\eta(t+h) = x(t) + \omega_1 K_1 + \omega_2 K_2 \iff$$

$$\eta(t+h) = x(t) + \omega_1 hf(t,x) + \omega_2 hf(t+\alpha h, x+\beta hf(t,x)). \tag{1.5}$$

Precisamos determinar ω_1, ω_2, α e β para que a equação 1.5 seja o mais precisa possível. Para isso, iremos expandir o máximo possível de termos utilizando a série de Taylor

$$x(t+h) = x(t) + hx'(t) + \frac{1}{2!}h^2x''(t) + \frac{1}{3!}h^3x'''(t) + ... \tag{1.6}$$

na qual estamos supondo que tanto $x(t)$ como $f(x(t),t)$ são funções suaves em t.

Comparando as equações 1.5 e 1.6, uma forma de fazer com que elas sejam semelhantes seria optando por adotar $\omega_1 = 1$ e $\omega_2 = 0$, uma vez que o sistema 1.1 nos dá que $x' = f$. Deste modo, obteríamos apenas a forma simples do método de Euler, que por sua vez tem ordem 1 de precisão.

Para alcançar uma precisão que chegue até o termo h^2 é preciso uma escolha de parâmetros um pouco mais complexa. Usaremos então uma aplicação da série de Taylor para funções de duas variáveis no termo final da equação 1.5. Adotando $n = 2$ no método

de Taylor mencionado, dado pela equação 1.3, os termos t, αh, x e βhf substituirão os termos x, h, y e k respectivamente, de modo que obteremos o seguinte:

$$f(t+\alpha h, x+\beta hf(t,x)) = f + \alpha h f_t + \beta h f f_x + \frac{1}{2}\left(\alpha h \frac{\partial}{\partial t} + \beta h f \frac{\partial}{\partial x}\right)^2 f(\bar{x},\bar{y}).$$

substituindo essa nova equação em 1.5 ela ficará dessa forma

$$\eta(t+h) = x(t) + (\omega_1+\omega_2)hf + \alpha\omega_2 h^2 f_t + \beta\omega_2 h^2 f f_x + \mathcal{O}(h^3). \tag{1.7}$$

A partir da equação 1.1 a equação 1.6 recebe uma nova forma. Uma vez que $x' = f$ temos que

$$x'' = \frac{dx'}{dt} = \left(\frac{df(t,x)}{dt}\right) = \left(\frac{\partial f}{\partial t}\right)\left(\frac{dt}{dt}\right) + \left(\frac{\partial f}{\partial x}\right)\left(\frac{dx}{dt}\right) = f_t + f_x f \Longrightarrow$$

$$x(t+h) = x(t) + hf + \frac{1}{2}h^2 f_t + \frac{1}{2}h^2 f f_x + \mathcal{O}(h^3). \tag{1.8}$$

Comparando as equações 1.7 e 1.8 obteremos as seguintes relações:

$$(\omega_1+\omega_2) = 1, \tag{1.9}$$

$$\alpha\omega_2 = \frac{1}{2}, \tag{1.10}$$

$$\beta\omega_2 = \frac{1}{2}. \tag{1.11}$$

Logo, uma solução conveniente para os termos acima seriam $\alpha = 1$, $\beta = 1$, $\omega_1 = \frac{1}{2}$ e $\omega_2 = \frac{1}{2}$.

Sendo assim, o resultado obtido para $x(t+h)$ aplicando o método de Runge-Kutta de ordem 2 será da forma:

$$\eta(t+h) = x(t) + \frac{1}{2}(K_1 + K_2),$$

onde

$$\begin{cases} K_1 = hf(t,x) \\ K_2 = hf(t+h, x+K_1). \end{cases}$$

Este método de Runge-Kutta é conhecido como método de Euler modificado, mas ele por sua vez não é o único método de R-K de ordem 2, pois podemos fazer outras escolhas que satisfazem as equações 1.9, 1.10 e 1.11. Usando α como parâmetro, temos as seguintes equações

$$\beta = \alpha, \qquad \omega_2 = \frac{1}{2\alpha}, \qquad \omega_1 = 1 - \frac{1}{2\alpha}.$$

□

O método de Runge-Kutta pode também ser de ordem 3 ou superior contendo três ou mais estágios consequentemente, sendo o de ordem 4 o mais utilizado.

1.1.3 Método de Runge-Kutta de Ordem 4

Diferente do algoritmo de ordem 2, o método de Runge-Kutta de ordem 4 é muito mais comum no meio científico para trabalhar com PVIs do tipo 1.1, pois, por se tratar de uma ordem maior, existem mais processos a serem realizados no caminho de convergência para a solução aproximada, e com isso temos um ganho de precisão e velocidade de convergência. Por conta disso, esse método é muito mais utilizado e consiste na seguinte equação:

$$\eta(t+h) = x(t) + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4) \tag{1.12}$$

onde

$$\begin{cases} K_1 = hf(t,x) \\ K_2 = hf(t+\frac{1}{2}h, x+\frac{1}{2}K_1) \\ K_3 = hf(t+\frac{1}{2}h, x+\frac{1}{2}K_2) \\ K_4 = hf(t+h, x+K_3). \end{cases}$$

Aqui não realizaremos a demonstração deste método, pois se trata de um longo processo que não caberia no tempo de planejamento deste trabalho, entretanto a mesma pode ser encontrada na referência [3].

1.1.4 Aplicação

Para exemplificar a diferença de precisão entre os dois métodos, realizaremos uma implementação deles na linguagem de programação denominada Python, de modo a aproximar um resultado para $x(2)$ partindo do seguinte PVI, com um passo de $0,01$:

$$\begin{cases} x' = 1 + x^2 + t^3 \\ x(1) = -4 \end{cases}, \tag{1.13}$$

Aplicação Runge-Kutta de Ordem 2

```
#Definição da função
def f(t, x):
    return 1 + x**2 + t**3

#Condição inicial x(a)
x = -4

#número de passos
n = 100

#intervalo de integração
a = 1
b = 2

#Passo
h = (b-a)/n

#Aplicação do Runge-Kutta de Ordem 2 iniciando em t = a e
#integrando n vezes para obter uma solução aproximada de x(b)
t = a
for k in range(0,n):
    k1 = h * f(t, x)
    k2 = h * f(t + (h / 2), x + (k1 / 2))
    x = x + (1/2)*(k1 + k2)
    t = a + ((k+1)*h)
print(x)
```

Ao término das integrações do código acima obtém-se que $x(2) \approx 4,3004174$.

Aplicação Runge-Kutta de Ordem 4

Se substituirmos no código anterior o laço descrito dentro do *for* pelo laço a seguir, obteremos então um código para solucionar o PVI 1.13 através do método de Runge-Kutta de ordem 4 e então poderemos comparar os resultados obtidos ao utilizar os mesmos parâmetros para as duas variações do método.

```
#Aplicação do Runge-Kutta de Ordem 4 iniciando em t = a e
#integrando n vezes para obter uma solução aproximada de x(b)
t = a
for k in range(0,n):
```

```
        k1 = h * f(t, x)
        k2 = h * f(t + (h / 2), x + (k1 / 2))
        k3 = h * f(t + (h / 2), x + (k2 / 2))
        k4 = h * f(t + h, x + k3)
        x = x + (1/6)*(k1 + (2*k2) + (2*k3) + k4)
        t = a + ((k+1)*h)
    print(x)
```

Feita a alteração, obtém-se que $x(2) \approx 4,3712208$. Para fins de parâmetros de comparação adotaremos $x(2) \approx 4,371221866$ como sendo uma aproximação mais precisa com relação à solução exata, que pode ser verificada por meio de uma análise mais aprofundada do PVI, entretanto a intenção aqui é apenas mostrar a diferença de precisão entre os dois métodos apresentados anteriormente perante um resultado mais próximo do ideal. Temos então os seguintes valores:

R-K 2ª Ordem	**R-K 4ª Ordem**	**Aprox. Precisa**
$x(2) \approx 4,3004174$	$x(2) \approx 4,3712208$	$x(2) \approx 4,371221866$

Analisando os resultados obtidos e comparando com o valor mais preciso podemos claramente observar que o método de ordem 4 é muito mais preciso que o de ordem 2, para melhor visualizar a diferença vejamos o erro relativo de cada um com relação ao valor de maior precisão:

Erro R-K 2ª Ordem	**Erro R-K 4ª Ordem**
$\approx 7,08 \times 10^{-2}$	$\approx 1,066 \times 10^{-6}$

Com isso temos a garantia de que utilizar o método de ordem 4 nos beneficiará com maior precisão e consequentemente maior velocidade de convergência, ou seja, será preciso menos integrações para obter um resultado de maior relevância de precisão.

Capítulo 2

Verificação Por Solução Manufaturada

Para verificarmos se o método numérico foi devidamente configurado e não contém erros de digitação ou erros de lógica, utilizaremos uma estratégia de depuração, que utiliza aproximações de erro para diferentes passos de integração do método em questão, aplicado a um problema que tenha solução exata conhecida $x(t)$. Essa estratégia é denominada verificação por solução manufaturada.

Se tomarmos, por exemplo, um passo h e um instante de tempo $t = t_f$ fixo, devemos então calcular as soluções numéricas $\eta(t_f, h)$ e $\eta\left(t_f, \frac{h}{2}\right)$, ou seja, para o instante de tempo t_f aplicaremos os passos h e $\frac{h}{2}$ no código em questão. Se h for suficientemente pequeno, os erros podem ser modelados com a seguinte aproximação

$$|e(t_f, h)| = |\eta(t_f, h) - x(t_f)| \approx |\mathcal{C}_{(t_f)}||h^p|, \tag{2.1}$$

$$|e\left(t_f, \frac{h}{2}\right)| = |\eta\left(t_f, \frac{h}{2}\right) - x(t_f)| \approx |\mathcal{C}_{(t_f)}||\left(\frac{h}{2}\right)^p|, \tag{2.2}$$

de modo que $\mathcal{C}_{(t_f)}$ é uma constante com relação a h, mas pode depender de t_f, sendo que p representa a ordem indicada pelo método que está sendo utilizado, ou seja, como neste trabalho abordamos os métodos de Runge-Kutta de ordens 2 e 4, então para o código de cada um deles o valor de p deve ser aproximadamente 2 e 4, respectivamente.

Vale ressaltar que o problema escolhido para fazer a verificação deve ter solução com número de derivadas superior à ordem do método. Considerando que temos a solução exata $x(t)$, para $t = t_f$ na equação 2.1, desejamos então nos certificar de que a ordem do método configurado tem a ordem prevista em teoria. Para isso, é preciso determinar os valores $\eta(t_f, h)$ e $\eta\left(t_f, \frac{h}{2}\right)$, utilizando os passos h e $\frac{h}{2}$, respectivamente, sendo $h > 0$. Feito isso, temos que o valor da razão entre 2.1 e 2.2 representará o seguinte:

$$\left|\frac{\eta(t_f, h) - x(t_f)}{\eta\left(t_f, \frac{h}{2}\right) - x(t_f)}\right| \approx \left|\frac{\mathcal{C}_{(t_f)}h^p}{\mathcal{C}_{(t_f)}\left(\frac{h}{2}\right)^p}\right| = 2^p, \tag{2.3}$$

ou seja,

$$\bar{p} = \log_2 \left| \left(\frac{\eta(t_f, h) - x(t_f)}{\eta\left(t_f, \frac{h}{2}\right) - x(t_f)} \right) \right|. \tag{2.4}$$

Ao aplicarmos esse método sucessivamente para passos de integração progressivamente menores, obteremos através da equação 2.4, uma sequência $\bar{p}_1, \bar{p}_2, \bar{p}_3, \bar{p}_4, ...$, que converge para o valor de p previsto na ordem do método numérico utilizado. Isso acontecerá somente se a implementação do método tiver sido realizada corretamente e se o problema utilizado tiver solução suficientemente diferenciável.

Como afirmado na referência [8], é de suma importância que sejam realizados procedimentos de verificação por solução manufaturada, pois faz parte das boas práticas de um bom programador ao aplicar um método numérico, buscando solucionar algum problema em questão. Usualmente esse procedimento deve ser realizado para várias soluções manufaturadas de complexidade variada.

2.1 Depuração do Método de Runge-Kutta de 2ª e 4ª Ordem

No capítulo anterior apresentamos dois códigos configurados em Python e realizamos uma comparação de suas precisões utilizando o mesmo passo h, entretanto, surge o questionamento: Como confiar que tudo foi corretamente configurado e que os resultados apresentados de fato justificam a escolha do método de 4ª ordem? Para isso, utilizamos a depuração por solução manufaturada para ter a garantia de que tudo havia sido feito corretamente. Antes de apresentarmos os códigos depurados e os resultados aproximados obtidos, descreveremos como foi feita a escolha da função $g(t)$, que será inserida no código para que a depuração seja feita de forma precisa.

Primeiro escolhemos a seguinte solução exata para $x(t)$

$$x(t) = \frac{-4}{e} e^t. \tag{2.5}$$

Em seguida, buscaremos descobrir quem é $g(t)$ (termo forçante), de modo que a solução exata 2.5 seja solução de

$$\begin{cases} x' = 1 + x^2 + t^3 + g(t) \\ x(1) = -4 \end{cases}, \tag{2.6}$$

substituindo 2.5 no sistema 2.6 e chamando $\frac{-4}{e}$ de $\mathcal{C}$, obteremos que

$$(\mathcal{C}e^t)' = 1 + (\mathcal{C}e^t)^2 + t^3 + g(t) \iff$$

$$g(t) = -1 - \mathcal{C}^2 e^{2t} + \mathcal{C}e^t - t^3. \tag{2.7}$$

2.1.1 Depuração do Runge-Kutta de Ordem 2

Dadas as definições descritas anteriormente, configuramos o código a seguir para o método R-K de ordem 2 e coletamos os dados referentes a cada erro relacionado ao número escolhido de passos que, por sua vez, começamos com um valor baixo e fomos dobrando até que obtivéssemos a convergência dos valores da ordem aproximada para a ordem do método em questão.

```
import numpy as np
# Definição da função g(t) tal que x(t) tem a solução exata escolhida
C = -4/(np.exp(1))
def g(t):
    return C*np.exp(t) - 1 - (C**2)*np.exp(2*t) - t**3

# Definição da função do PVI
def f(t, x):
    return 1 + x**2 + t**3

# Condição inicial x(a)
x = -4

# número de passos que deve ser dobrado a cada integração
# até que se observe a convergência do erro tender a 0
# e a ordem tender a ordem do método que está sendo aplicado
n = 12

# Intervalo de integração
a = 1
b = 2

# Passo
h = (b - a) / n

# Aplicação do Runge-Kutta de Ordem 2 utilizando a função g(t)
# para calcular a solução manufaturada, iniciando em t = a e
# integrando n vezes para obter o valor do erro de aproximação,
# ao realizar a diferença entre a solução aproximada de x(t)
```

Tabela 2.1: Depuração por solução manufaturada do algoritmo do R-K de 2ª ordem.

Número de Passos (n)	Erro aproximado	Razão	$\log_2(Razão)$
12	$-8{,}0 \times 10^{-2}$	$\approx 10{,}00$	$\approx 3{,}32$
24	$-8{,}0 \times 10^{-3}$	$\approx 5{,}30$	$\approx 2{,}41$
48	$-1{,}5 \times 10^{-3}$	$\approx 4{,}70$	$\approx 2{,}23$
96	$-3{,}2 \times 10^{-4}$	$\approx 4{,}20$	$\approx 2{,}10$
192	$-7{,}6 \times 10^{-5}$	$\approx 4{,}20$	$\approx 2{,}10$
384	$-1{,}8 \times 10^{-5}$	$\approx 4{,}00$	$\approx 2{,}00$
768	$-4{,}5 \times 10^{-6}$	$\approx 4{,}00$	$\approx 2{,}00$

```
# e a solução exata de g(t) no print ao final do laço "for"
t = a
for k in range(0,n):
    k1 = h * (f(t, x) + g(t))
    k2 = h * (f(t + h, x + k1 ) + g(t+h))
    x = x + (1 / 2) * (k1 + k2)
    t = a + ((k+1)*h)
print(x)
print(t)
print(x-C*np.exp(t))
```

A tabela 2.1, foi construída a partir dos dados coletados a cada integração do código para um número diferente de passos, onde indicamos os números de passos (integrações), aproximação do erro associado, a razão, calculada conforme descrito em 2.3 e suas respectivas aproximações de ordem calculadas segundo a equação 2.4.

Como podemos observar na tabela 2.1, a convergência para a ordem do método ocorreu perfeitamente, o que indica que o método foi configurado corretamente ao ser realizada a preparação do algoritmo.

2.1.2 Depuração do Runge-Kutta de Ordem 4

Para fazer a depuração do algoritmo do método de R-K de 4ª ordem, substituímos o código apresentado a seguir no algoritmo anterior, na etapa equivalente a configuração do método de ordem 2, e da mesma forma, realizamos as alterações no valor número de passos para obter as aproximações dos erros.

```
# Aplicação do Runge-Kutta de Ordem 4 utilizando a função g(t)
# para calcular a solução manufaturada, iniciando em t = a e
# integrando n vezes para obter o valor do erro de aproximação,
# ao realizar a diferença entre a solução aproximada de x(t)
# e a solução exata de g(t) no print ao final do laço "for"
```

Tabela 2.2: Depuração por solução manufaturada do algoritmo do R-K de 4ª ordem.

Número de Passos (n)	Erro aproximado	Razão	$\log_2(\widetilde{Razão})$
12	$-5{,}3 \times 10^{-3}$	$\approx 24{,}09$	$\approx 4{,}50$
24	$-2{,}2 \times 10^{-4}$	$\approx 18{,}33$	$\approx 4{,}20$
48	$-1{,}2 \times 10^{-5}$	$\approx 18{,}46$	$\approx 4{,}20$
96	$-6{,}5 \times 10^{-7}$	$\approx 16{,}66$	$\approx 4{,}05$
192	$-3{,}9 \times 10^{-8}$	$\approx 16{,}25$	$\approx 4{,}02$
384	$-2{,}4 \times 10^{-9}$	$\approx 16{,}00$	$\approx 4{,}00$
768	$-1{,}5 \times 10^{-10}$	$\approx 16{,}00$	$\approx 4{,}00$

```
for k in range(0,n):
    k1 = h * (f(t, x) + g(t))
    k2 = h * ((f(t + (h / 2), x + (k1 / 2)) + g(t + h / 2)))
    k3 = h * ((f(t + (h / 2), x + (k2 / 2)) + g(t + h / 2)))
    k4 = h * (f(t + h, x + k3) + g(t + h))
    x = x + (1/6)*(k1 + (2*k2) + (2*k3) + k4)
    t = a + ((k+1)*h)
print(x)
print(t)
print(x-C*np.exp(t))
```

A tabela 2.2, assim como a anterior, foi construída a partir dos dados coletados a cada integração do código para um número de passos diferente, onde indicamos os números de passos e suas respectivas aproximações de ordem.

Como podemos observar, a convergência para a ordem do método ocorreu perfeitamente, o que indica que o método foi configurado corretamente ao ser realizada a preparação do algoritmo.

Capítulo 3

Relação de Força Entre Dois Corpos no Espaço

Neste capítulo vamos supor que dois corpos de massas m_1 e m_2 se encontram no espaço tridimensional, e cada um tem uma velocidade inicial arbitrária. Considerando que seus movimentos são regidos pela Lei da Gravitação Universal e que estão em um sistema inercial.

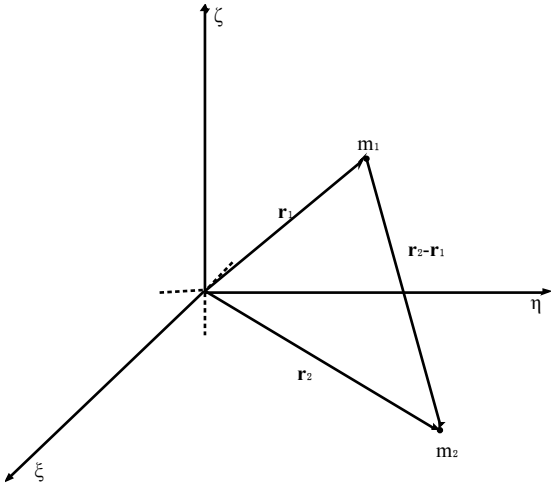


Figura 3.1: Posições das Massas m_1 e m_2.

Portanto, para obter a relação de força do problema de dois corpos, necessitaremos de uma sequência de mudanças de coordenadas. Iniciaremos com um sistema inercial arbitrário no qual um vetor $\mathbf{r}$ qualquer pode ser descrito da seguinte forma:

$$\mathbf{r} = \xi\hat{\xi} + \eta\hat{\eta} + \zeta\hat{\zeta}, \tag{3.1}$$

onde $\hat{\xi}$, $\hat{\eta}$ e $\hat{\zeta}$ são os vetores unitários do sistema escolhido e ξ, η e ζ representam a posição de um objeto qualquer. Sendo assim, os vetores $\mathbf{r}_1$ e $\mathbf{r}_2$ representarão a posição das massas m_1 e m_2, respectivamente. Conforme a figura 3.1, [1].

Partindo agora da segunda Lei de Newton em sua forma vetorial:

$$\mathbf{F} = m\frac{d^2\mathbf{r}}{dt^2}, \tag{3.2}$$

chamaremos de $\mathbf{F}_{12}$ a força gravitacional exercida sob m_1 devido à m_2 e de $\mathbf{F}_{21}$ a força gravitacional exercida sob m_2 devido à m_1, onde

$$\mathbf{F}_{12} = k^2 m_1 m_2 \frac{\mathbf{r}_2 - \mathbf{r}_1}{|\ \mathbf{r}_2 - \mathbf{r}_1\ |^3} \tag{3.3}$$

$$\mathbf{F}_{21} = -k^2 m_1 m_2 \frac{\mathbf{r}_2 - \mathbf{r}_1}{|\ \mathbf{r}_2 - \mathbf{r}_1\ |^3}. \tag{3.4}$$

Vale ressaltar que a diferença de sinais entre as forças $\mathbf{F}_{12}$ e $\mathbf{F}_{21}$ remetem ao sentido em que estão sendo exercidas.

Afim de compreender melhor o funcionamento das equações que descrevem as relações de força no espaço e compor um primeiro esboço de um código que será melhor trabalhado posteriormente com dados reais, desenvolvemos um código com dados genéricos, de modo que pudéssemos obter algum desenho orbital, averiguando se o método de desenvolvimento das relações de força havia sido configurado corretamente. O mesmo será explicado no capítulo a seguir, ao final da modelagem do problema Sol-Terra-Lua, nas equações 4.10 à 4.12.

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



#x[0], x[1] e x[2] posição do corpo 1, coordenadas x, y, z respectivamente
#x[3], x[4] e x[5] posição do corpo 2, coordenadas x, y, z respectivamente
#x[6], x[7] e x[8] velocidade do corpo 1, coordenadas x, y, z respectivamente
#x[9], x[10] e x[11] Velocidade do corpo 2, coordenadas x, y, z respectivamente

kq = 3.5 #Constante k^2 da lei da gravitação escolhida de forma arbitrária

m1 = 1  #Massa do corpo 1
m2 = 2  #Massa do corpo 2

# condição inicial dos objetos genéricos
x = np.array([-1,0,0,1,0,0,0,-2,0,0,1,0])

#composição das equações de força de dois objetos no espaço
```

```
def f(t, x):
  d3 = (np.sqrt((x[0]-x[3])**2 + (x[1] - x[4])**2 + (x[2] - x[5])**2))**3

  f6 = kq*m2*(x[3] - x[0])/d3
  f7 = kq*m2*(x[4] - x[1])/d3
  f8 = kq*m2*(x[5] - x[2])/d3
  f9 = -kq*m1*(x[3] - x[0])/d3
  f10 = -kq*m1*(x[4] - x[1])/d3
  f11 = -kq*m1*(x[5] - x[2])/d3
  return np.array([x[6], x[7], x[8], x[9], x[10], x[11],
                   f6, f7, f8, f9, f10, f11])



#Condições iniciais do método R-K
n = 1000       #número de passos
a = 0          #inicio do intervalo de integração
b = 50         #fim do intervalo de integração
h = (b-a)/n    #Passo
t = a          #Variável de tempo

#Criação da matriz de vetores, onde cada linha receberá as posições calculadas
v=np.zeros((n+1,12))

#Inserção das condições iniciais na primeira linha do vetor
v[0, :] = x

#Método de Runge-Kutta de 4ª ordem
for k in range(0,n):
    k1 = h * f(t, x)
    k2 = h * f(t + (h / 2), x + (k1 / 2))
    k3 = h * f(t + (h / 2), x + (k2 / 2))
    k4 = h * f(t + h, x + k3)
    #Composição dos K_{i}s para estimar a próxima posição dos objetos
    x = x + (1/6)*(k1 + (2*k2) + (2*k3) + k4)
    t = a + ((k+1)*h) #Avanço no tempo
    v[k+1,:] = x      #Registro da nova posição dos objetos



#2D
matplotlib.pyplot.plot(v[:, 0], v[:, 1])
```

```
matplotlib.pyplot.plot(v[:, 3], v[:, 4])
plt.axis('equal')
matplotlib.pyplot.show()
```

Como resultado deste código, obtivemos as imagens 3.2 e 3.3 que mostram o movimento entre os dois corpos genéricos descritos no código. É possível observar que os dois orbitam um centro de massa associado as massas de ambos. Vale destacar, que na imagem 3.3 adicionamos apenas uma velocidade no eixo Z, para ilustrar qual seria o real movimento dos corpos no espaço ao longo do tempo. Tendo tudo isso em vista, comprovamos assim, que o código desenvolvido cumpriu com o objetivo de testar o funcionamento das equações de relação de força e também de testar a estrutura do código. Deste modo, no próximo capítulo adaptaremos o algorítimo para trabalhar com uma modelagem real.

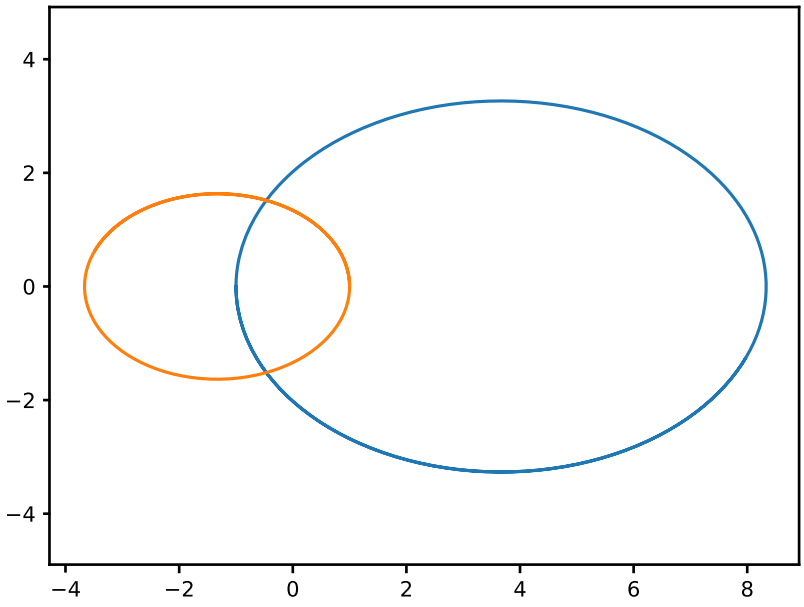


Figura 3.2: Sistema de dois corpos genéricos orbitando um centro de massa.

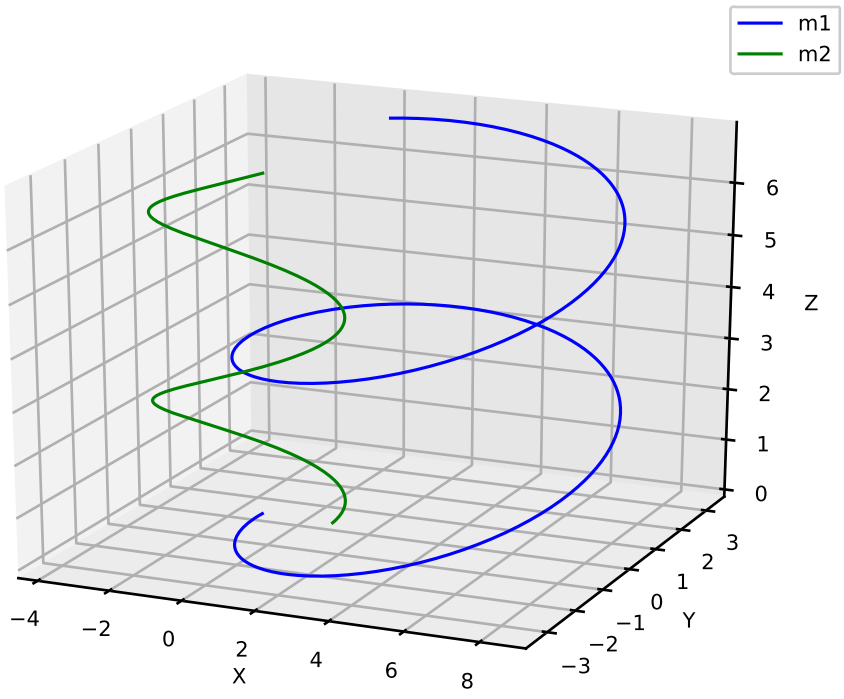


Figura 3.3: Sistema de dois corpos genéricos orbitando um centro de massa com movimento no espaço.

Capítulo 4

Modelagem do Problema de 3 Corpos Sol-Terra-Lua

Antes de modelarmos nosso problema Sol-Terra-Lua, temos que ajustar alguns parâmetros nas equações 3.3 e 3.4 apresentadas anteriormente, cujo desenvolvimento está mais detalhado na referência [5]. Nelas encontramos a constante k^2, que na física é equivalente a constante gravitacional G que, por sua vez, equivale à $6,673 \times 10^{-11} m^3/(kg \cdot s^2)$.

Uma outra consideração a ser realizada é a de que o Sol será colocado como o centro do sistema, uma vez que sua massa é muito superior as massas da Terra e da Lua, sendo assim, ele será o referencial inercial do sistema, deste modo reduzimos o problema de três corpos a um problema de dois corpos, que pode ser mais facilmente solucionado. Para facilitar, consideraremos que a Terra e a Lua serão massas pontuais. A figura a seguir ilustra o sistema.

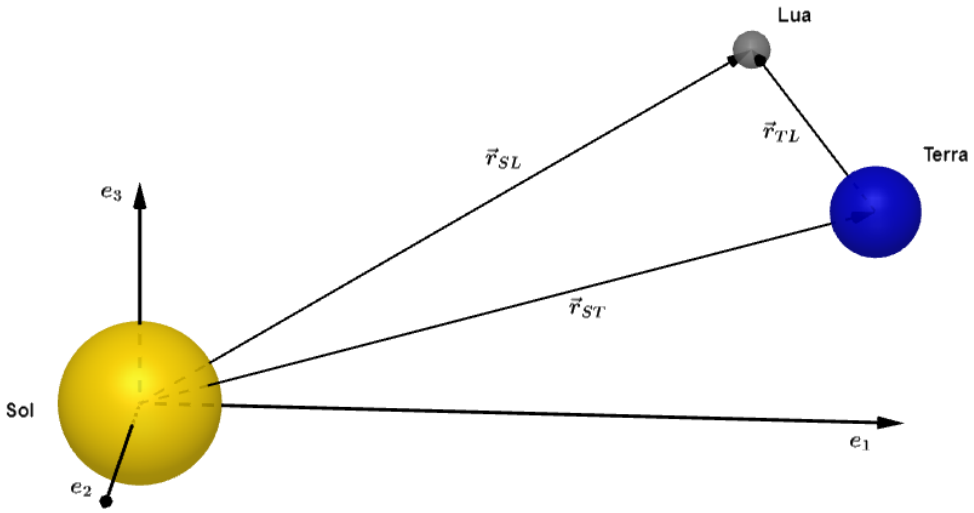


Figura 4.1: Configuração do sistema Sol, Terra, Lua, [7].

Utilizando os subíndices S, T e L, para fazer referência ao Sol, Terra e Lua, respectivamente, descreveremos a seguir as relações de força que agem sob cada corpo a partir do que foi descrito em 3.3 e 3.4:

A força exercida pelo Sol sob a Terra é dada por

$$\mathbf{F}_{ST} = -G\frac{m_T M_S}{r_{ST}^3}\mathbf{r}_{ST}. \tag{4.1}$$

A força exercida pelo Sol sob a Lua é dada por

$$\mathbf{F}_{SL} = -G\frac{m_L M_S}{r_{SL}^3}\mathbf{r}_{SL}. \tag{4.2}$$

A força exercida pela Terra sob a Lua é dada por

$$\mathbf{F}_{TL} = -G\frac{m_T m_L}{r_{TL}^3}\mathbf{r}_{TL}. \tag{4.3}$$

E por fim, a força exercida pela Lua sob a Terra é dada por

$$\mathbf{F}_{LT} = -G\frac{m_T m_L}{r_{LT}^3}\mathbf{r}_{LT} = -\mathbf{F}_{TL}. \tag{4.4}$$

Onde m_T, m_L e M_S representam as massas da Terra, Lua e Sol, respectivamente, as componentes $\mathbf{r}_{ST}$, $\mathbf{r}_{SL}$, $\mathbf{r}_{TL}$ e $\mathbf{r}_{LT}$ representam os vetores Sol-Terra, Sol-Lua, Terra-Lua e Lua-Terra (conforme apresentado na figura 4.1), de modo que os denominadores das frações representam o módulo de cada um desses vetores, por isso não estão com o termo r em destaque.

Relacionando as equações apresentadas com a equação de força desenvolvida por Newton e aplicando o princípio da superposição de forças, que segundo descrito na referência [2], nos permite calcular a força total a que uma partícula está submetida, através da soma vetorial das forças que todas as demais partículas exercem sobre ela. Sendo assim, obtemos o seguinte sistema:

$$\begin{cases} m_T \cdot \mathbf{a}_T = -G\frac{m_T M_S}{r_{ST}^3}\mathbf{r}_{ST} - G\frac{m_T m_L}{r_{LT}^3}\mathbf{r}_{LT} \\ m_L \cdot \mathbf{a}_L = -G\frac{m_L M_S}{r_{SL}^3}\mathbf{r}_{SL} - G\frac{m_T m_L}{r_{TL}^3}\mathbf{r}_{TL} \end{cases}$$

onde $\mathbf{a}_T$ e $\mathbf{a}_L$ representam a aceleração da Terra e da Lua. Reescrevendo a equação acima, temos que

$$\begin{cases} \frac{d^2}{dt^2}\mathbf{r}_T = GM_s\left(-\frac{\mathbf{r}_{ST}}{r_{ST}^3} - \frac{m_L}{M_S}\frac{\mathbf{r}_{LT}}{r_{LT}^3}\right) \\ \frac{d^2}{dt^2}\mathbf{r}_L = GM_s\left(-\frac{\mathbf{r}_{SL}}{r_{SL}^3} - \frac{m_T}{M_S}\frac{\mathbf{r}_{TL}}{r_{TL}^3}\right) \end{cases} \tag{4.5}$$

Seja $\mathbf{r}_i$ o vetor tridimensional dado por:

$$\mathbf{r}_i = \sum_{j=1}^{3} x_{ij} \cdot \widehat{e}_j,$$

ou seja, $\mathbf{r}_i$ representa todos os vetores posição relacionados ao sistema Terra, Lua e Sol, que por sua vez, a partir de agora, usaremos os índices 1, 2 e 3, respectivamente, para melhor descrever suas representações no algoritmo que apresentaremos posteriormente. A componente $\widehat{e}_j$, representa os eixos ortonormais do sistema tridimensional e a componente x_{ij} representa os parâmetros das coordenadas em cada eixo do sistema.

Buscando evitar trabalhar com números muito grandes, realizaremos um processo chamado de adimensionalização, que consiste em escolher um parâmetro base e em seguida o relacionar aos demais parâmetros semelhantes a ele, ou seja, no nossos caso realizaremos duas adimensionalizações diferentes, uma em relação a distância Sol-Terra e a outra será a adimensionalização do tempo, que consistirá em normalizar as constantes que aparecerão nas equações que serão desenvolvidas a seguir.

Vamos agora introduzir os parâmetros reais R e t' que serão usados na adimensionalização. Sejam

$$\begin{cases} x_{ij} = Ru_{ij} \\ t = t' \cdot \widetilde{t} \end{cases}, \tag{4.6}$$

onde R é escolhido como o raio médio entre o Sol e a Terra em metros. O valor de t' em segundos, é escolhido de acordo com a equação 4.8. Assim, u_{ij} são as novas coordenadas computacionais adimensionais e $\widetilde{t}$ o tempo computacional adimensional. Tudo isso dito, podemos reescrever as equações presentes em 4.5 da seguinte forma

$$\frac{1}{t'^2}\frac{d^2}{d\widetilde{t}^2}\sum_{j=1}^{3} u_{ij}\cdot\widehat{e}_j = \frac{GM_S}{R^3}\sum_{\substack{k=1,\\ k\neq i}}^{3}\sum_{j=1}^{3}\left(-\frac{M_k}{M_S}\frac{(u_{ij}-u_{kj})}{(\sum_{l=1}^{3}(u_{il}-u_{kl})^2)^{\frac{3}{2}}}\cdot\widehat{e}_j\right), \tag{4.7}$$

onde k, é conjunto de todos os corpos presentes no problema e i é o conjunto dos corpos que estão sob a ação das forças aplicadas, logo $i = 1, 2$, uma vez que o Sol, representado pelo índice 3, foi escolhido como centro do sistema e, por sua vez, não consideraremos as influências por ele sofridas, pois são minímas e isso apenas dificultaria a modelagem.

Nas equações anteriores, realizamos o processo de adimensionalização das distâncias conforme fora descrito, agora, para garantir que toda a equação seja adimensionalizada, t' será escolhido de modo que a constante que multiplica a equação 4.7 seja normalizada, logo

$$t' = \sqrt{\frac{R^3}{GM_S}}, \tag{4.8}$$

sendo assim a equação 4.7 assumirá a forma

$$\frac{d^2}{d\widetilde{t}^2}\sum_{j=1}^{3} u_{ij}\cdot\widehat{e}_j = \sum_{\substack{k=1,\\ k\neq i}}^{3}\sum_{j=i}^{3}\left(-\frac{M_k}{M_S}\frac{(u_{ij}-u_{kj})}{(\sum_{l=1}^{3}(u_{il}-u_{kl})^2)^{\frac{3}{2}}}\cdot\widehat{e}_j\right). \tag{4.9}$$

Agora realizaremos o processo de redução de ordem do problema modelado. Desta forma simplificaremos a preparação do algoritmo que será apresentado em breve. Inicia-

remos a partir da seguinte definição

$$\begin{cases} x_{ij0} \equiv u_{ij} \\ x_{ij1} = \frac{d}{dt} u_{ij}, \end{cases} \tag{4.10}$$

onde x_{ij0} representa a posição dos objetos modelados e x_{ij1} a velocidade, sendo ela a primeira derivada da posição. Substituindo 4.10 em 4.9, obtemos que

$$\begin{cases} \frac{d}{dt}\sum_{j=1}^{3} x_{ij0}\cdot\widehat{e}_j = \sum_{j=1}^{3} x_{ij1}\cdot\widehat{e}_j \\ \frac{d}{dt}\sum_{j=1}^{3} x_{ij1}\cdot\widehat{e}_j = \sum_{\substack{k=1,\\ k\neq i}}^{3}\sum_{j=i}^{3}\left(-\frac{M_k}{M_S}\frac{(x_{ij0}-x_{kj0})}{(\sum_{l=1}^{3}(x_{il0}-x_{kl0})^2)^{\frac{3}{2}}}\cdot\widehat{e}_j\right), \end{cases} \tag{4.11}$$

onde $i = 1, 2$ totalizando um sistema de 12 equações escalares.

Por fim, podemos reescrever o sistema 4.11, de modo que em suma, descreve os movimentos da Terra e da Lua através das seguintes EDOs de primeira ordem

$$\begin{cases} \frac{d}{dt}\sum_{j=1}^{3} x_{1j0}\cdot\widehat{e}_j = \sum_{j=1}^{3} x_{1j1}\cdot\widehat{e}_j \\ \frac{d}{dt}\sum_{j=1}^{3} x_{2j0}\cdot\widehat{e}_j = \sum_{j=1}^{3} x_{2j1}\cdot\widehat{e}_j \\ \frac{d}{dt}\sum_{j=1}^{3} x_{1j1}\cdot\widehat{e}_j = \sum_{k=2}^{3}\sum_{j=1}^{3}\left(-\frac{M_k}{M_S}\frac{(x_{1j0}-x_{kj0})}{(\sum_{l=1}^{3}(x_{1l0}-x_{kl0})^2)^{\frac{3}{2}}}\cdot\widehat{e}_j\right) \\ \frac{d}{dt}\sum_{j=1}^{3} x_{2j1}\cdot\widehat{e}_j = \sum_{\substack{k=1,\\ k\neq 2}}^{3}\sum_{j=1}^{3}\left(-\frac{M_k}{M_S}\frac{(x_{2j0}-x_{kj0})}{(\sum_{l=1}^{3}(x_{2l0}-x_{kl0})^2)^{\frac{3}{2}}}\cdot\widehat{e}_j\right), \end{cases} \tag{4.12}$$

onde as funções $x_{1js}(t)$ e $x_{2js}(t)$ são referentes aos corpos Terra e Lua respectivamente.

4.1 Implementação do Problema em Python

Cada vez mais a programação numérica vem se tornando não só uma grande aliada, mas essencial para a exploração espacial e para a compreensão do desenvolvimento do universo e consequentemente do nosso sistema solar

Combinando as informações apresentadas até o momento, adaptamos uma simulação encontrada na referência [7], para a estrutura do algoritmo que viemos desenvolvendo durante todo o trabalho. Utilizando o método de Runge-Kutta de ordem 4 que foi depurado via solução manufaturada e utilizando o algoritmo desenvolvido para o problema de dois corpos genéricos. Esse código, possivelmente pode ser melhor otimizado no caso de desejar uma abordagem científica mais precisa, entretanto para os fins desse trabalho está sob medida, buscamos detalhar cada passo nele presente através de comentários, para deixá-lo o mais didático possível, o mesmo ainda pode ser acessado através da referência [6].

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

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# m^3/(kg*s^2) é a constante gravitacional
G = 6.673*10**(-11)

#Terra/Lua/Sol em quilogramas
m1 = 5.9724*10**24  #terra
m2 = 0.07346*10**24  #lua
m3 = 1.98892*10**30  #sol

R = 1.4709*10**11  #Raio Terra no periélio
V = 30.29 *10**3   #Velocidade da terra no periélio
Rm = 1.496*10**11  #Raio médio Sol-Terra
Vm = 29.78*10**3   #Velocidade média da trajetória

RL = R + 0.3633*10**9   #Raio da Lua no perigeu
VL = V + 1.076*10**3    #Velocidade da Lua no perigeu
Rml = Rm + 0.3633*10**9 #Raio médio Sol-Lua
Vml = 1.02207*10**3     #Velocidade média da trajetória

#Adimensionalização das distâncias com relação a distância Terra-Sol
Rt = R/Rm   #Raio Sol-Terra
Vt = V/Vm   #Velocidade Terra

Rl = RL/Rm #Raio Sol-Lua
Vl = VL/Vm #Velocidade Lua

#Admensionalização do tempo
A =  1                          #Ano
As = 31536000                   #1 ano em segundos
Tmet = np.sqrt((Rm**3)/(G*m3))  #Tempo métrico
Tcomp = As/Tmet                 #Tempo computacional / normal izado
Ti = 0                          #Tempo inicial
Tf = A*Tcomp                    #Tempo final
dt = 1000                       #intervalo dt em segundos
n = int (A*As/dt)               #Número de divisões

#Condições iniciais
x = np.array([Rt,0,0,Rl,0,0,0,Vt,0,0,Vl,0.0033])
```

```
#x[0], x[1] e x[2] posição da Terra, coordenadas x, y, z respectivamente
#x[3], x[4] e x[5] posição da Lua, coordenadas x, y, z respectivamente
#x[6], x[7] e x[8] velocidade da Terra, coordenadas x, y, z respectivamente
#x[9], x[10] e x[11] Velocidade da Lua, coordenadas x, y, z respectivamente

#Constantes das Equações
C1 = m2/m3 #Constante das equações da Lua
C2 = m1/m3 #Constante das equações da Terra

#Função das relações de força entre os objetos
def f(t, x):
  d3 =  (np.sqrt(( x[0]-  x[3])**2 + (x[1] - x[4])**2 + (x[2] - x[5])**2) )**3
  dt3 = (np.sqrt((x[0])**2 + (x[1])**2 + (x[2])**2))**3
  dl3 = (np.sqrt((x[3])**2 + (x[4])**2 + (x[5])**2))**3

  #Movimento da Terra
  f6 =  ((-x[0])/dt3) - C1*(x[0] - x[3])/  d3
  f7 =  ((-x[1])/dt3) - C1*(x[1] - x[4])/  d3
  f8 =  ((-x[2])/dt3) - C1*(x[2] - x[5])/  d3

  #movimento da Lua
  f9 =  ((-x[3])/dl3) - C2*(x[3] - x[0])/  d3
  f10 = ((-x[4])/dl3) - C2*(x[4] - x[1])/  d3
  f11 = ((-x[5])/dl3) - C2*(x[5] - x[2])/  d3

  return np.array([x[6], x[7], x[8], x[9], x[10], x[11], f6, f7, f8, f9, f10, f11])

#Vetor u que receberá as novas posições calculadas no método R-K
#Criação da matriz de vetores, onde cada linha receberá as posições calculadas
u=np.zeros((n+1,12))
#Inserção das condições iniciais na primeira linha do vetor u
u[0, :] = x


#Condições iniciais do método R-K
a = Ti          #Condição do tempo
t = a           #Variável de tempo
h = (Tf-Ti)/n   #Passo de integração
```

```
#Método de Runge-Kutta de 4ª ordem
for k in range(0,n):
    k1 = h * f(t, x)
    k2 = h * f(t + (h / 2), x + (k1 / 2))
    k3 = h * f(t + (h / 2), x + (k2 / 2))
    k4 = h * f(t + h, x + k3)
    #Composição dos K_{i}s para estimar a próxima posição da Terra e da Lua
    x = x + (1/6)*(k1 + (2*k2) + (2*k3) + k4)
    t = a + ((k+1)*h) #Avanço no tempo
    u[k+1,:] = x      #Registro da nova posição da Terra e da Lua


#Função para plotagem das órbitas
def plot(titulo, X, Y, planeta):

    plt.figure(titulo)
    plt.plot(X , Y, label = titulo)
    plt.xlabel( r' $x_{%s}$'%planeta )
    plt.ylabel( r' $y_{%s}$'%planeta )
    plt.grid()
    plt.legend (loc = 1)

plot( u'Orbita da Terra 2D' ,u[:, 0], u[:, 1], 'T' )
plot( u'Orbita da Lua 2D' ,u[:, 3] - u[:, 0], u[:, 4]-u[:,1], 'L_R' )
fig = plt.figure()

#composição 3D das órbitas da terra e da Lua
ax = Axes3D(fig)
ax.plot(u[:, 0], u[:, 1],u[:, 2],'b-',label='Terra')
ax.plot(u[:, 3], u[:, 4],u[:, 5],'g-',label='Lua')
ax.legend()
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')


plt.show()
```

4.1.1 Resultados obtidos

Após rodar o código, ele nos apresentou as seguintes imagens das órbitas da Terra e da Lua no plano e também de casos diferentes da composição das duas órbitas no espaço, com variações no tempo orbital.

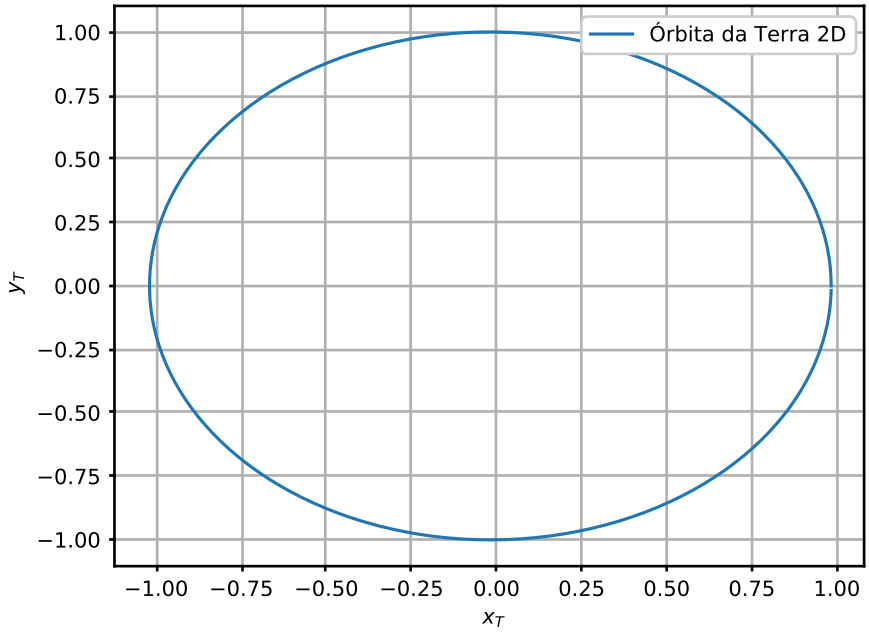


Figura 4.2: Órbita da Terra adimensionalizada [7].

As imagens 4.3 e 4.4 são referentes a órbita da Lua adimensionalizada. A intenção de apresentá-las é mostrar que a órbita da lua, assim como as de todos outros objetos celestes, tem suas variações ao longo do tempo, podendo ser mais ou menos evidente. Como o período orbital da Lua é de 27 dias, quando olhamos seu movimento ao longo de 1 ano completo, nota-se uma variação conforme a apresentada na imagem 4.4.

As imagens 4.5 e 4.6 nos mostram a combinação dos movimentos da Lua e da Terra no espaço, nelas é possível notar com mais evidencia que a Lua não está no mesmo plano em que a terra e o sol. É importante observar também, que por conta dessa evidenciação, o movimento da lua no eixo Z não está em escala. Observando mais atentamente, outro fator que podemos notar, é de que a órbita da Terra sofre uma leve variação no espaço por consequência desse movimento da Lua, uma vez que, os dois objetos orbitam o centro de massa do sistema Terra-Lua, e ele por sua vez, está levemente deslocado do centro de massa da Terra.

Por fim, a imagem 4.7 apresenta um outro evento interessante, onde claramente podemos observar que o movimento da Lua no espaço, para períodos maiores que do que 1 ano, não segue o mesmo trajeto, havendo assim uma pequena variação de um ano para outro.

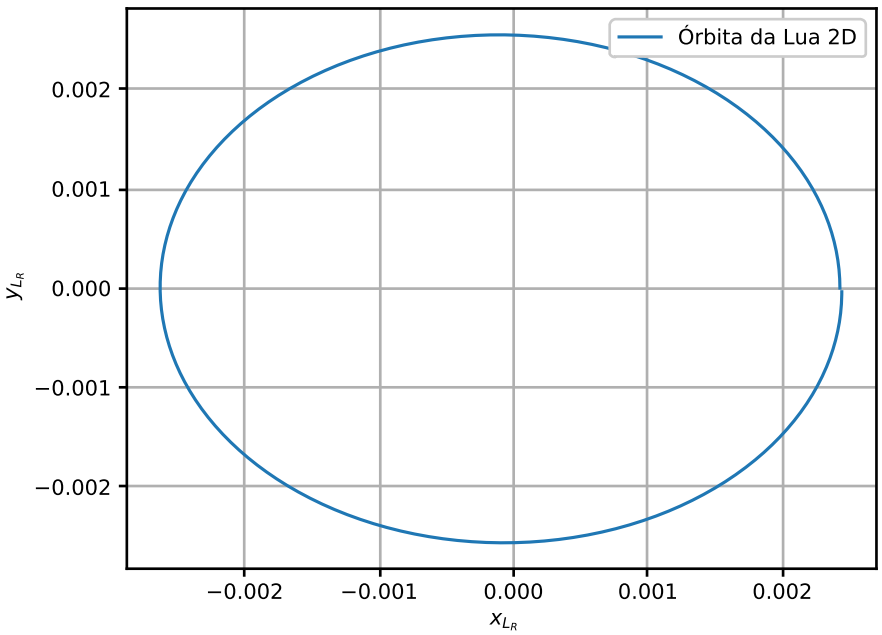


Figura 4.3: Órbita da Lua adimensionalizada [7].

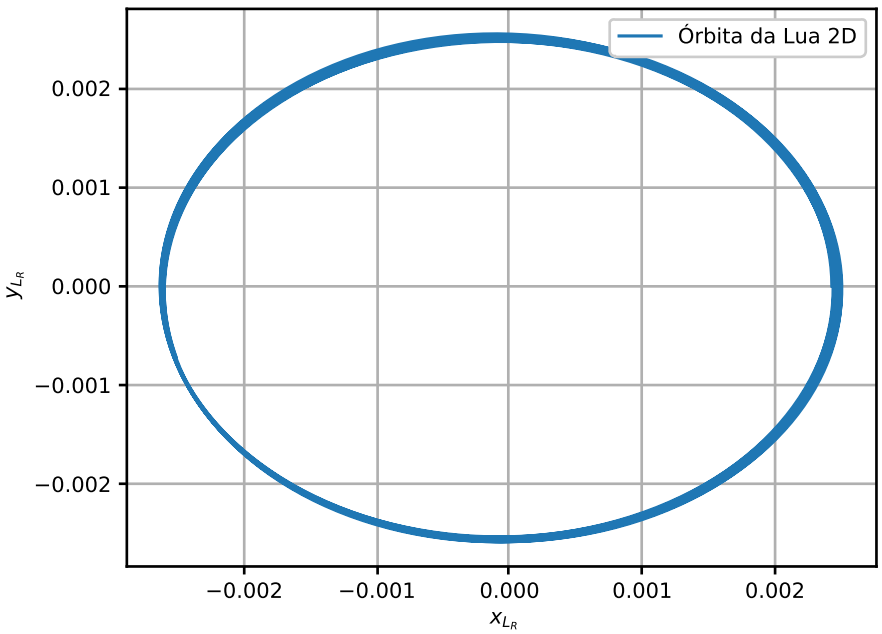


Figura 4.4: Órbita da Lua adimensionalizada: Período de 1 ano.

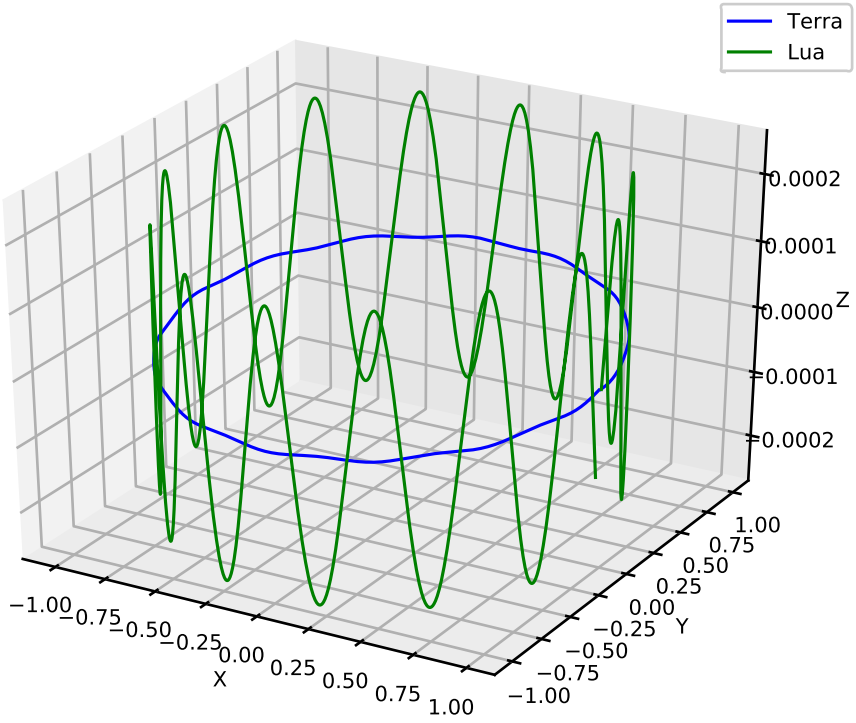


Figura 4.5: Composição tridimensional das órbitas da Terra e da Lua adimensionalizadas [7].

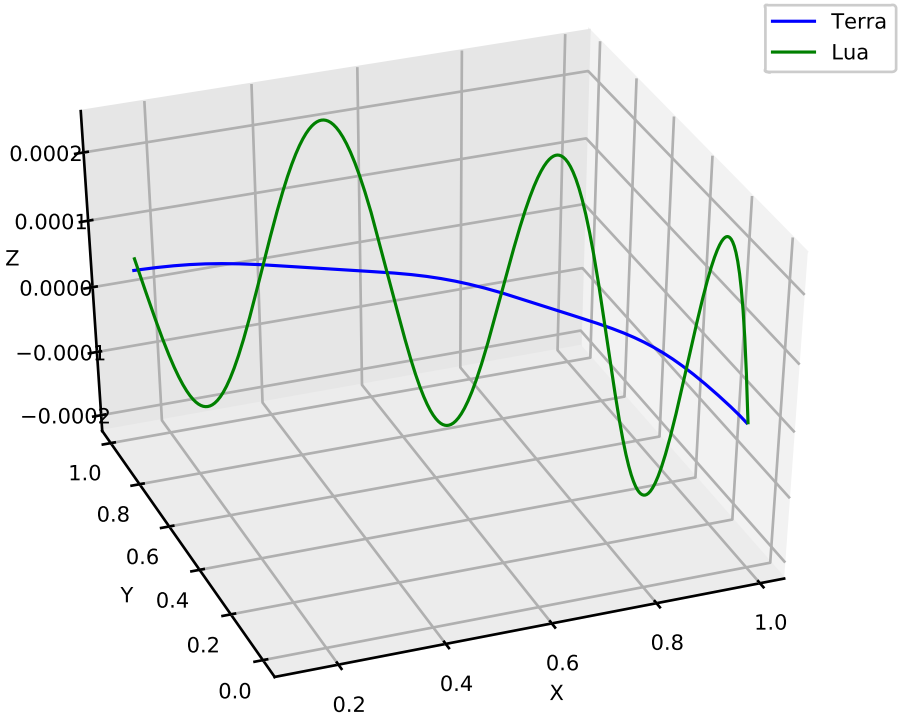


Figura 4.6: Composição tridimensional das órbitas da Terra e da Lua adimensionalizadas: Período de 3 meses.

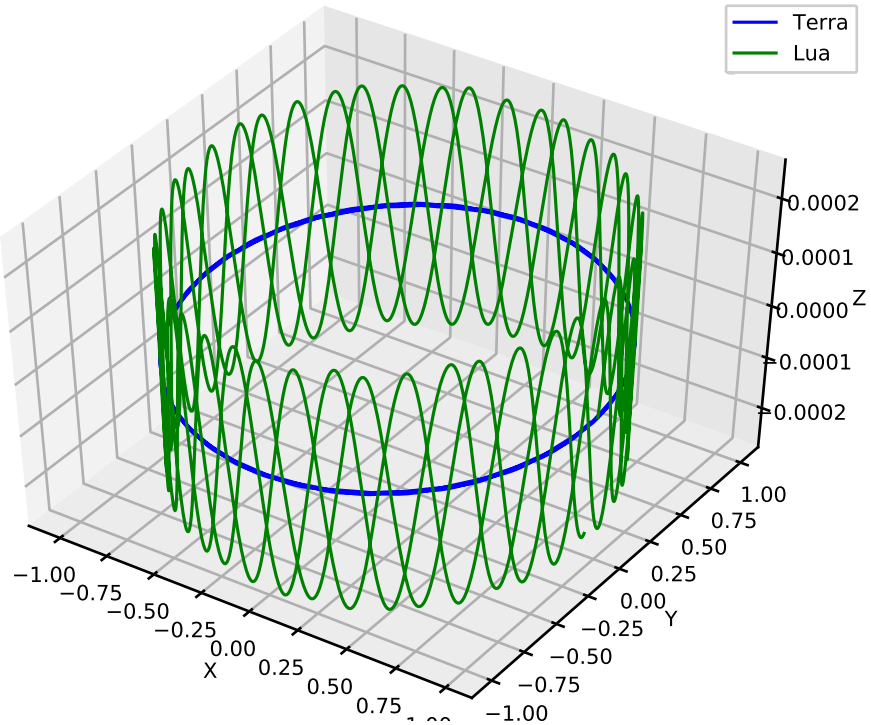


Figura 4.7: Composição tridimensional das órbitas da Terra e da Lua adimensionalizadas: Período de 3 anos.

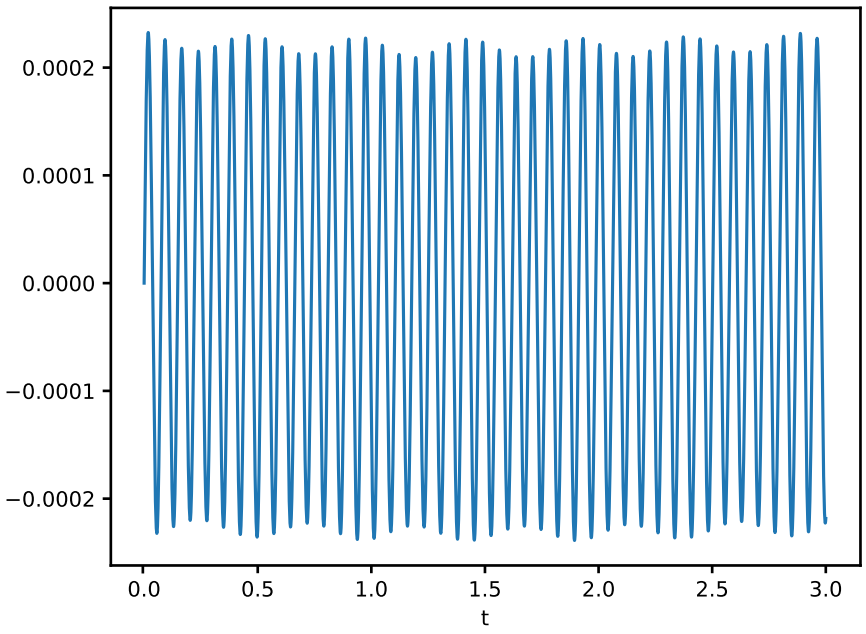


Figura 4.8: Variação na amplitude da coordenada Z da Lua: Período de 3 anos.

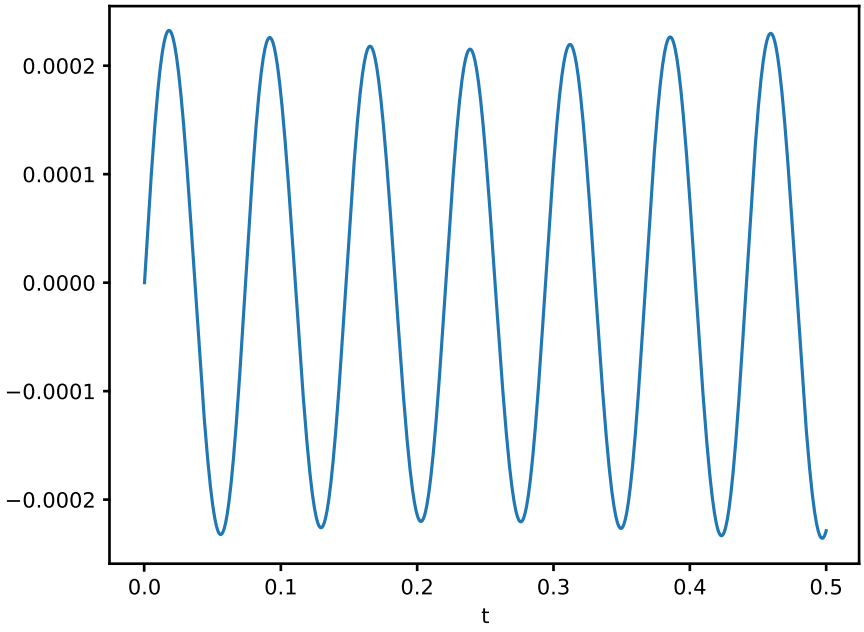


Figura 4.9: Período de variação na amplitude da coordenada Z da Lua: 6 meses

De modo complementar, na imagem 4.8 observamos que a amplitude do movimento da lua no eixo Z, sofre uma variação periódica, porém se observarmos a imagem 4.9, veremos que o período é menor do que meio ano, sendo assim, isso já implica no fato de a trajetória da Lua não se repetir no período de 3 anos da imagem 4.7.

Para uma análise um pouco mais detalhada das órbitas obtidas, recomenda-se a leitura da referência [7], uma vez que o nosso foco era fazer a modelagem e apresentar uma simulação, faltando tempo hábil para tal.

Capítulo 5

Conclusão

O presente trabalho teve como intuito apresentar uma cadeia de métodos numéricos que, computacionalmente, podem ser aplicados a modelagem de diversos problemas matemáticos. Aqui em especial apresentamos aplicações voltadas para soluções de PVIs, tendo como foco principal a modelagem de problemas de dinâmica orbital.

A princípio, apresentamos o método de Runge-Kutta, que foi percursor em todas as etapas de desenvolvimento deste trabalho, em especial o método de $4^{\underline{a}}$ ordem, sendo ele responsável por trazer a precisão necessária para que a modelagem final pudesse ocorrer.

Para verificarmos se nossos algoritmos haviam sido configurados corretamente, apresentamos o método de verificação via solução manufaturada, que após as testagens a ordem apresentada pelo algoritmo era equivalente a ordem do método configurado, garantindo assim sua correta aplicação.

Após uma breve retomada da relação de forças entre dois corpos no espaço, realizamos uma primeira configuração do algoritmo que posteriormente seria adaptado para a modelagem de um problema real. O intuito nesta etapa, foi justamente de entender como montaríamos a estrutura base e se a redução de ordem do problema havia sido realizada corretamente.

Por fim, adentramos na modelagem de um caso especifico de três corpos na mecânica celeste, uma vez que não existe ainda uma solução geral de viável aplicação, deste modo após algumas considerações importantes, reduzimos o problema a um problema de dois corpos, onde o Sol foi considerado o referencial inercial do sistema e as distâncias foram parametrizadas segundo a distância Terra-Sol, fazendo o processo de adimensionalização, o que nos faria trabalhar com números pequenos. Da mesma forma parametrizamos o tempo, para que este fosse o fator de adimensionalização das constantes, tornando por fim o problema passível de uma configuração mais simples.

Concluímos então a modelagem do problema de três corpos, com um algoritmo que a partir dos dados reais, adaptado da referência [7], que ao fim, nos apresenta o caminho das órbitas da Terra e da Lua, tendo o Sol como referencial inercial. Deste modo, atingimos

o objetivo principal deste trabalho, que era realizar aplicações numéricas a um problema de dinâmica orbital.

Referências Bibliográficas

[1] Vieira Neto, E. *Introdução à Mecânica Celeste*, XXIV Escola de Verão em Dinâmica Orbital e Planetologia.

[2] Halliday, D.; Resnick, R.; Walker, J. *Fundamentos de física, volume 2: gravitação, ondas e termodinâmica*, tradução Ronaldo Sérgio de Biasi, 10ª Edição, Rio de Janeiro: LTC, 2016.

[3] Hairer, E.; Norsett, S. P.; Wanner, G. *Solving Ordinary Differential Equations I: Nonstiff problem*, Springer-Verlag Berlin Heidelberg, 1997.

[4] Kincaid, D.; Cheney, W. *Numerical Mathematics and Computing*, Thomson Brooks/-Cole, 6ª Edição.

[5] Antunes, L. V. *Problema de Dois Corpos na Mecânica Celeste*, Trabalho de Conclusão de Curso A, Universidade Federal de São Carlos, São Carlos - SP, 2021.

[6] Antunes, L. V. *Sol-Terra-Lua-Modelo*, disponível em <`https://colab.research.google.com/drive/1Qobq0vCrC7OdiXnFQ2QFPsQ64aEbq8hU?usp=sharing`>, ultimo acesso: 03/11/2021.

[7] Silva, C. E. R.; Silva, R. R.; Batista, L. A. L.; *O sistema Sol-Terra-Lua: Uma breve simulação numérica usando Python*, Revista Ciencias Exatas e Naturais - RECEN, Unicentro/PR, Brasil, 2018.

[8] *Modelagem usando EDOs*, Ime - USP, 2019, disponível em <`https://www.ime.usp.br/~map3121/2019/map3121/programas/EP2-edos.pdf`>, ultimo acesso: 03/11/2021.