MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

ESTUDO DA PERTURBAÇÃO DA ÓRBITA DE SATÉLITES ARTIFICIAIS DEVIDO À AÇÃO DA ATRAÇÃO LUNI-SOLAR

RELATÓRIO FINAL DE PROJETO DE INICIAÇÃO CIENTÍFICA
(PIBIC/INPE/CNPq)

Erick de Souza Fernandes (UBC, Bolsista PIBIC/CNPq)
E-mail: erick.souza281355@gmail.com

Hans-Ulrich Pilchowski (INPE-ETE/DMC, Orientador)
E-mail: hans.pilchowski@inpe.br

Julho de 2018

"Dedico este trabalho primeiramente a Deus, por ser essencial em minha vida, autor de meu destino, meu guia, socorro presente na hora da angústia, ao meu pai Sidnei de Oliveira Fernandes, minha mãe Giscele Cristina Machado e aos meus irmãos."

AGRADECIMENTO

"Aos meus pais, amigos Anderson Bartholomeu de Oliveira e Rodolfo Lyu Shimotsu, INPE e o CNPq pelo incentivo e ao Prof. Hans-Ulrich Pilchowski, pela sua orientação, seu grande desprendimento em ajudar-nos e amizade sincera."

RESUMO

O presente trabalho tem como foco a elaboração de um algoritmo para a obtenção da perturbação da órbita dos satélites artificiais, devida à ação da atração lunar. Para obter o entendimento do funcionamento das leis da mecânica celeste e aplicá-las sobre órbitas de satélites artificiais. Para efetuar aplicação foi necessário o uso do método inverso e do método direto, consecutivamente. Assim, a primeira parte a considerar no problema dos dois corpos, para obter posição, velocidade e os elementos keplerianos, subsequentes, da orbita do satélite, ao longo do tempo, considerando o geopotencial da Terra, até o nível J2. Inicialmente, o algoritmo não considerou a perturbação, para obter o vetor de estado. Porém, para aplicá-lo ao problema dos três corpos, onde entra a perturbação, fez-se necessário obter uma constante, a partir da posição da Terra em relação à Lua. E então aplicá-la na variação de Gauss, que fornecerá a perturbação dos elementos keplerianos em cada ponto da órbita individual e simultaneamente. O algoritmo desenvolvido, tem como finalidade fornecer essas perturbações automaticamente, colocando-o em forma de sub-rotina, passível de ser inserida em algoritmos computacionais mais abrangentes, de forma que seus resultados possam ser somados a outras perturbações Orbitais e utilizá-las na correção orbital, sempre que necessário.

Palavras-chave: Determinação da órbita de satélites, perturbação da órbita, atração luni-solar.

SUMÁRIO

1 INTRODUÇÃO 8
2 OBJETIVO 9
2.1 OBJETIVOS GERAIS 9
2.2 OBJETIVOS ESPECIFICOS 9
3 FUNDAMENTAÇÃO TEÓRICA 9
3.1 ELEMENTOS KEPLERIANO 9
3.2 VARIAÇÃO DE GAUSS 10
3.3 ATRAÇÃO LUNAR 10
4 MATERIAIS E METODOS 12
5 ANÁLISES E RESULTADO 12
6 CONCLUSÃO 17
REFERÊNCIAS BIBLIOGRÁFICAS 18
APÊNDICE 19

LISTA DE ILUSTRAÇÕES

Figura 1 - Atração lunar que influencia nos satélites terrestres. 11
Figura 3 - dados para o posicionamento das três posições. 13
Figura 4 - Dados do posicionamento direto. 13
Figura 5 - Dados do posicionamento inverso. 14
Figura 6 - Propagação orbital de um satélite Terrestre. 14
Figura 7 - Comportamento da ascensão da reta perturbada devido a atração lunar ao longo do tempo. 15
Figura 8 - Comportamento da inclinação perturbada devido a atração lunar ao longo do tempo. 16
Figura 9 - Comportamento do argumento do perigeu perturbada devido a atração lunar ao longo do tempo. 16

LISTA DE SIMBOLOS E ABREVIATURAS

a – Semi-eixo maior

h – Momento angular

e – Excentricidade

i – Inclinação

Ω – Ascensão da reta

ω – Argumento do perigeu

f – Anomalia verdadeira

μ – Constante gravitacional

μL – Constante gravitacional lunar

$\boldsymbol{F}(r)$ – Força Atrativa

G – Constante gravitacional universal

M – Massa do corpo maior

m – Massa do corpo menor

r – Distancia entre os dois corpos

1 INTRODUÇÃO

Como quase todos têm a curiosidade de contemplar o céu e pensar em seus enigmas, foi devido essa fascinação que possibilitou entender uma parte da lógica empregada no cosmo, que como base desses conhecimentos, é desenvolvido os estudos astronômicos atuais que representam esses fenômenos.

A partir desses estudos, como o movimento dos corpos celestes, que consiste no comportamento dos planetas em relação ao Sol, corpo com maior massa, que seguindo as teorias de Isaac Newton, no conceito de massas e a sua força de atração, em conjunto com as leis de Kepler, onde foram utilizados os conceito para colocar em prática o estudo da interferência que ocorre nos satélites artificiais, devido à ação da força gravitacional influenciada pela atração da Lua.

Assumindo que o leitor tenha conhecimento básico de mecânica celeste, o presente trabalho consiste no desenvolvimento de um algoritmo, que deve ser aplicado na determinação de órbitas e da perturbação dessas em satélites artificiais. Assim, foram utilizados os métodos Direto e Inverso para a propagação orbital. E como base do projeto foi desenvolvido um programa visando a perturbação da atração lunar, que irá verificar a interferência nos elementos keplerianos ao longo do tempo, os quais continuariam constantes caso não houvessem perturbações, como a gravitacional devida à não homogeneidade de distribuição de massa e à forma não esférica da Terra [2].

2 OBJETIVO

2.1 OBJETIVOS GERAIS

O projeto de pesquisa tem como objetivo desenvolver um algoritmo computacional que forneça dados da **perturbação orbital** de satélites artificiais terrestres, **devida à ação da atração lunar**. Assim, visa-se alcançar o objetivo de obter um algoritmo que possa ser inserido em programas computacionais mais abrangentes, destinados ao controle de órbitas.

2.2 OBJETIVOS ESPECIFICOS

- Analisar os dados extraídos pelo método direto e inverso, comparando-os com os dados já existentes.
- Analisar graficamente a propagação orbital do satélite sem uma perturbação.
- Analisar graficamente a perturbação devido a atração lunar nos elementos keplerianos individualmente.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 ELEMENTOS KEPLERIANO

Os elementos keplerianos ou elementos orbitais, são os componentes que definem uma orbita e é responsável por determinar a forma de movimento, posição e sua velocidade, dentre eles tem [2]:

Momento angular (h) → Momento angular ou quantidade de movimento angular de um corpo é uma grandeza física associada à rotação desse corpo.
Excentricidade (e) → Representa o tipo de orbita (hiperbólica, elíptica, circular).
Inclinação (i) → É o ângulo formado entre o plano da órbita e o plano de referência.
Anomalia verdadeira (f) → Ângulo entre as direções foco da elipse - periastro e foco da elipse.
Argumento do perigeu (ω) → Ponto mais próximo do foco.
Nodo ascendente (Ω) → Ponto onde a órbita cruza o Equador, do hemisfério sul ao norte.

3.2 VARIAÇÃO DE GAUSS

Uma pergunta importante é como a perturbação afeta os elementos orbitais; ou seja, eles deixam de ser constantes, e passam a variar no tempo. A partir do conhecimento dos elementos orbitais $(h, e, f, \Omega, i, \omega)$, como uma função temporal podemos determinar a posição e a velocidade de um objeto, prevendo a órbita resultante, que tende a ficar degenerada ao longo do tempo, devido a uma aceleração perturbadora que tende a afeta essas constantes [1], abaixo tem os elementos keplerianos devido à aceleração perturbadora em cada elemento ao longo do tempo:

$$\frac{dh}{dt} = r \cdot \rho_s$$

$$\frac{de}{dt} = \frac{h}{\mu}\sin f \rho_s + \frac{1}{\mu h}[(h^2 + \mu r)\cos f + \mu e r]\rho_s$$

$$\frac{di}{dt} = \frac{r}{h}\cos(\omega + f)\rho_w$$

$$\frac{df}{dt} = \frac{h}{r^2} + \frac{1}{eh}\left[\frac{h^2}{\mu}\cos f \rho_r - \left(r + \frac{h^2}{\mu}\right)\sin f \rho_s\right]$$

$$\frac{d\Omega}{dt} = \frac{r}{h \sin i}\sin(\omega + f)\rho_w$$

$$\frac{d\omega}{dt} = -\frac{1}{eh}\left[\frac{h^2}{\mu}\cos f \rho_r - \left(r + \frac{h^2}{\mu}\right)\sin f \rho_s\right] - \frac{r \sin(\omega + f)}{h \tan i}\rho_w$$

Para os elementos orbitais perturbados, basta integrar os elementos citados acima em função do tempo, e obter os dados individuais que irá afetar a órbita.

3.3 ATRAÇÃO LUNAR

Um satélite quando está orbitando em torno da Terra fica sujeito a diversas forças, onde uma das forças predominantes é de origem gravitacional. Se a Terra fosse esférica e homogênea teria uma orbita elíptica, mas devido ao geopotencial apresentar deformações, tende a afetar a órbita [3]. Outra força que perturba a órbita é a ação gravitacional da Luar e

do Sol, onde segundo a teoria de Newton, uma massa m é atraído por uma massa M, e vice-versa, apresentando uma força de interação entre os dois corpos que varia de acordo com a massa e a distância, como pode-se observar na Eqn 3.1.

$$\boldsymbol{F}(r) = \frac{GMm}{r^2}\hat{\boldsymbol{r}} \tag{3.1}$$

Em uma aceleração gravitacional lunar, leva-se em consideração o problema de três copos, que considera a posição dos copos, no caso da lua os dados são obtidos a partir do almanaque astronômico e levadas em relação à uma origem referencial XYZ num momento de inercia, onde as três massas sofrem uma influência atrativa simultânea [1], por exemplo a Terra e a Lua atrai o satélite, como pode-se observar na figura 3.1.

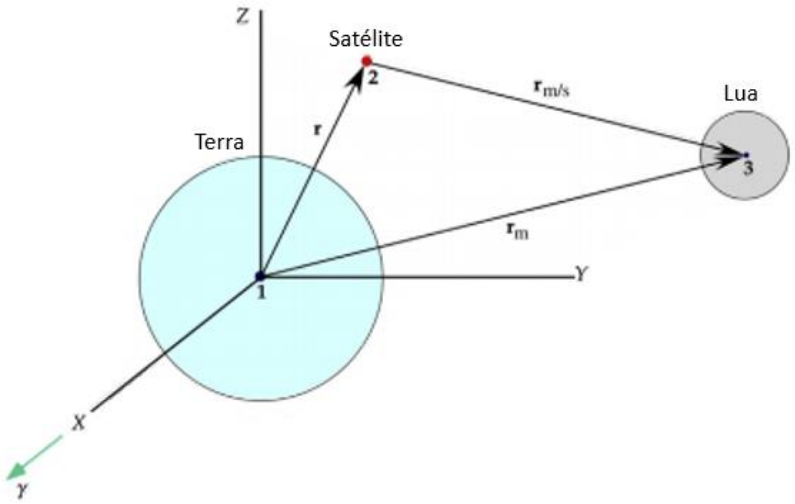

Figura 1 - Atração lunar que influencia nos satélites terrestres.
Fonte: CURTIS H. D

Para obter a aceleração perturbadora $\boldsymbol{\rho}$ influenciada pela gravidade lunar tem-se que levar em consideração a constante gravitacional lunar $\mu L = 4903\ km^3/s^2$, para aplicar na equação de movimento de Kepler, Eqn. (3.2).

$$\boldsymbol{\rho} = \mu L\left(\frac{\boldsymbol{r}_{\frac{m}{s}}}{r_{\frac{m}{s}}^3} - \frac{\boldsymbol{r}_m}{r_m^3}\right) \tag{3.2}$$

Calcule o vetor unitário nos eixos rsw:

$$\hat{\boldsymbol{r}} = \frac{\boldsymbol{r}}{\|\boldsymbol{r}\|} \qquad \hat{\boldsymbol{w}} = \frac{\boldsymbol{r} \times \boldsymbol{v}}{\|\boldsymbol{r} \times \boldsymbol{v}\|} \qquad \hat{\boldsymbol{s}} = \frac{\dot{\boldsymbol{w}} \times \dot{\boldsymbol{r}}}{\|\dot{\boldsymbol{w}} \times \dot{\boldsymbol{r}}\|}$$

Depois de obter o vetor unitário nos pontos rsw, aplicar a aceleração perturbadora em cada ponto para colocar na equação de Gauss.

$$\rho_r = \boldsymbol{\rho} \cdot \hat{\boldsymbol{r}} \qquad \rho_s = \boldsymbol{\rho} \cdot \hat{\boldsymbol{s}} \qquad \rho_w = \boldsymbol{\rho} \cdot \hat{\boldsymbol{w}}$$

4 MATERIAIS E METODOS

O projeto de pesquisa tem como foco inicial, a partir de uma linguagem de programação elaborar um algoritmo que venha a fornecer os dados dos métodos usados para determinar uma órbita, o primeiro passo consiste no método das três posições em conjunto com a propagação orbital em função do tempo, o procedimento é a partir do sinal emitido pelas três posições proporcionais do satélite, que é enviado para as antenas quando este entra no plano atmosférico, a partir deste método obtém-se a velocidade correspondente de cada posição, para usá-lo no próximo passo na propagação orbital para determinar a nova posição e velocidade ao longo do tempo. Depois de obter a determinação da órbita, foi feito um algoritmo que forneça a perturbação ocasionada pela atração lunar.

5 ANÁLISES E RESULTADO

Primeiramente o código foi implementado em linguagem Python, por ser de fácil interpretação e propicio a trabalhar com dados científicos, onde os resultados obtidos foram comparados com dados encontrados por CURTIS (2014).

Para validar o algoritmo de determinação de órbitas, foram comparados os dados obtidos dos métodos das três posições, direto e inverso com os já existentes, como os dados mostram mesmo comportamento, os algoritmos são considerados validos, abaixo pode-se observar os resultados nas Figura (3 – 5).

Os valores para validação dos dados para método das três posições, usado por CURTIS (2014):

$Posição\ 1: -294{,}32\hat{\boldsymbol{i}} + 4265{,}1\hat{\boldsymbol{j}} + 5986{,}7\hat{\boldsymbol{k}}\ (km)$

$Posição\ 2: -1365{,}5\hat{\boldsymbol{i}} + 3637{,}6\hat{j} + 6346{,}8\hat{\boldsymbol{k}}\ (km)$

$Posição\ 3: -2940{,}3\hat{\boldsymbol{i}} + 2473{,}7\hat{\boldsymbol{j}} + 6555{,}8\hat{\boldsymbol{k}}\ (km)$

```
12 Velocidade para a posição (dois): [-6.21740189 -4.01216524  1.59898473]
13
14 Os elementos orbitais que representa a posição e a velocidade (dois)
15 Semi-eixo maior :   7813.0  Km
16 Excentricidade:   0.10010369281339042
17 Inclinação:   60.000470277369566  graus
18 Ascensão do nodo:   40.00144177286778  graus
19 Argumento do Perigeu:   30.074116831547773  graus
20 Anomalia verdadeira:   49.92565926551782  graus
21
22
23 Process finished with exit code 0
```

Figura 2 - dados para o posicionamento das três posições.
Fonte: o Autor

Os valores para validação de dados no método direto, usado por CURTIS (2014):

$Momento\ angular\ (h)$: 80.000 km^2/s

$Excentricidade\ (e)$: 1.4

$Inclinação\ (i)$: 30º

$Ascensão\ da\ reta\ (\Omega)$: 40º

$Argumento\ do\ perigeu\ (\omega)$: 60º

$Anomalia\ verdadeira\ (f)$: 30º

```
11  posição: [-4039.8959232   4814.56048018  3628.62470217]
12  velocidade: [-10.38598762  -4.77192164   1.743875  ]
13
14 Process finished with exit code 0
```

Figura 3 - Dados do posicionamento direto.
Fonte: o Autor

Os valores para validação de dados no método inverso, usado por CURTIS (2014):

$Posição: -6045\hat{\boldsymbol{i}} - 3490\hat{\boldsymbol{j}} + 2500\hat{\boldsymbol{k}}\ (km)$

$Velocidade: -3{,}457\hat{\boldsymbol{i}} + 6{,}618\hat{\boldsymbol{j}} + 2.533\hat{\boldsymbol{k}}\ (km/s)$

```
12 Momento angular :   58311.66993185606  Km²/s
13 Excentricidade:   0.17121234628445342
14 Inclinação:   153.2492285182475  graus
15 Ascensão do nodo:   104.72071466560382  graus
16 Argumento do Perigeu:   20.068316650582467  graus
17 Anomalia verdadeira:   28.445628306614967  graus
18
19
20 Process finished with exit code 0
```

Figura 4 - Dados do posicionamento inverso.
Fonte: o Autor

Para colocar em prática a propagação orbital ao longo do tempo, foram utilizados os métodos citados a cima em conjunto, na Figura 6 mostra uma simulação de como o programa apresentaria graficamente os satélites orbitando a Terra.

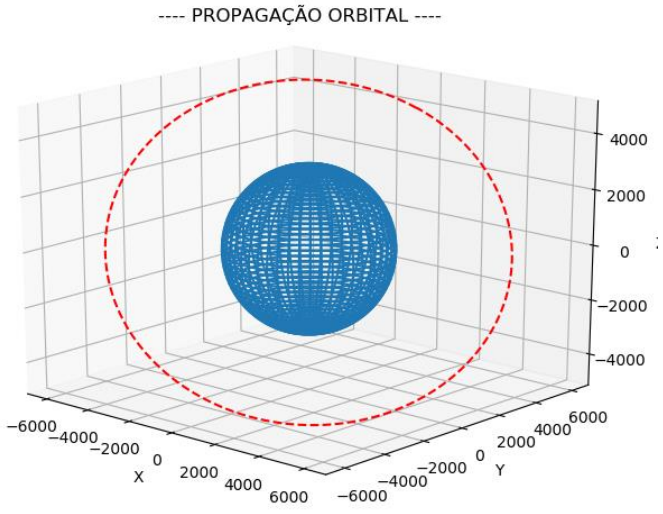

Figura 5 - Propagação orbital de um satélite Terrestre.
Fonte: o Autor

Para simulação das perturbações devido as ações ocasionadas pela atração lunar, foram usados os mesmos dados encontrados por CURTIS (2014), seguem abaixo os dados usados:

$$Semi - eixo\ maior\ = 26553.4$$
$$Excentricidade\ = 0.741$$
$$Argumento\ do\ perigeu\ = 270°$$
$$Ascensão\ da\ reta\ = 0$$
$$Inclinação\ = 63.4°$$
$$Anomalia\ verdadeira\ = 0$$
$$Data\ Juliana\ = 2454283$$

Nas Figuras (7 – 9), é possível observar as perturbações nos elementos orbitais mais significativos em uma órbita elíptica, levando em consideração a força de atração lunar no problema dos três corpos durante sessenta dias, comparando-os com dados já existentes.

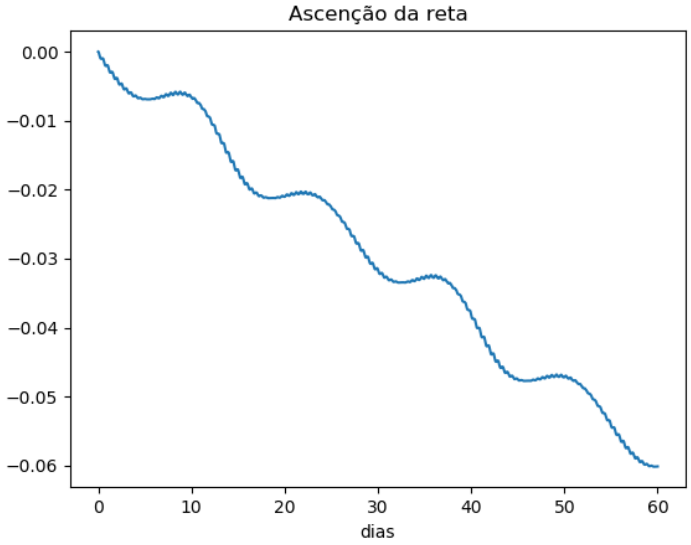

Figura 6 - Comportamento da ascensão da reta perturbada devido a atração lunar ao longo do tempo.
Fonte: o Autor

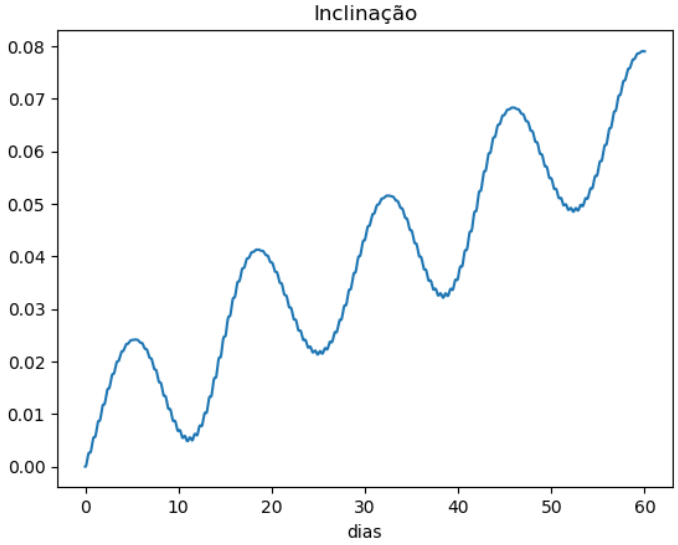

Figura 7 - Comportamento da inclinação perturbada devido a atração lunar ao longo do tempo.
Fonte: o Autor

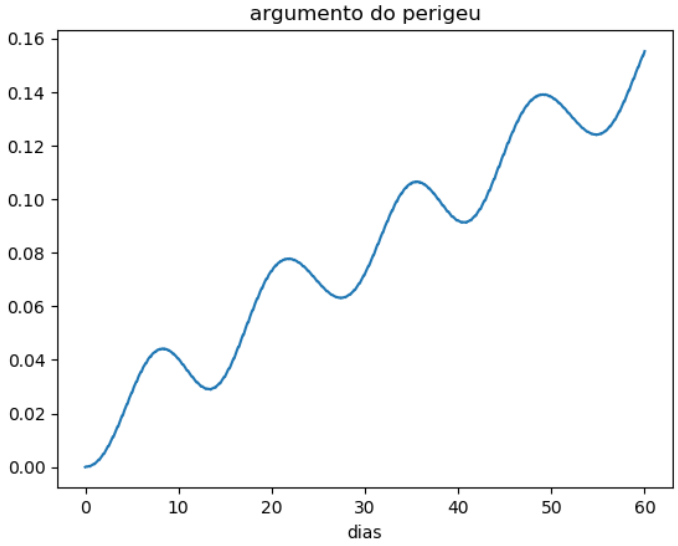

Figura 8 - Comportamento do argumento do perigeu perturbada devido a atração lunar ao longo do tempo.
Fonte: o Autor

6 CONCLUSÃO

Os resultados indicam que os algoritmos foram implementados com êxitos, se aproximando dos dados encontrados por CURTIS (2014). Nas perturbações, os elementos keplerianos mais significativos (f, i, ω), se dá pelo fato de os outros elementos não variarem muito no tempo, levando em consideração que nas outras perturbações tem um efeito mais significativo, por esse motivo não foi computado.

O tipo de linguagem usada para o tratamento dos dados, apresenta mais casas decimais se aproximando dos dados reais, infelizmente o tempo de processamento não está sendo hábil, sendo umas das possíveis melhorias futuras, junto da implantação de uma interface gráfica que facilite a comunicação entre o homem e a máquina, deixando mais rápido e eficiente.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] - CURTIS, Howard D. **Orbital Mechanics: for Engineering Students**. Third Edition. United States: Elsevier Ltd. 2014.

[2] - PILCHOWSKI, H.U.; SILVA, W.C.C. & FERREIRA, L.D.D. **Introdução à mecânica celeste**. São José dos Campos, São Paulo, Brasil, 1981. (INPE-2126-RPE/350)

[3] - FERREIRA, L.D.D.; SILVA, W.C.C. & PILCHOWSKI, H.U. **Notas sobre sistemas de coordenadas e tempo**. São José dos Campos, São Paulo, Brasil, 1979. (INPE-1634-RPE/039)

[4] - BROUWER, D. & CLEMENCE, G.M. **Methods of celestial mechanics**. New York, N.Y., Academic, 1961.

[5] - ESCOBAL, P.R. **Methods of orbital determination**. New York, N.Y., John Wiley & Sons, 1965.

[6] - PRUSSING, J. E.; CONWAY, B. A. **Orbital Mechanics**. Oxford, University Press, 1993.
BOND, V. R.; ALLMAN, M. C. **Modern Astrodynamics: Fundamentals and Perturbation Methods**. Princeton University Press, 1996.

[7] - KAPLAN, M. H. **Modern spacecraft dynamic & control**. New York: John Wiley & Sons, 1976.

[8] - LARSON, W. J.; WERTZ, J. R. **Space mission analysis and design**. Torrance, California: Space Technology Series, 1992.

[9] - WIE, B. **Space vehicle dynamics and control**. Reston, Virginia: AIAA Education Series, 1998.

[10] - NOAA/NASA/USAF, 1976. **U.S. Standard Atmosphere**, 1976. GPO.

APÊNDICE

Este apêndice lista os scripts do python onde foram implementados os algoritmos citados no texto. Este programa usa uma linguagem simples, onde foi desenvolvido a lógica do problema da perturbação luni-solar.

Algoritmo para o procedimento do método das três posições:

```
class three_pos(object):
    def __init__(self, mi, pos):
        self.mi = mi
        self.pos = pos
        self.tol = 1e-4

    def três_posições(self):
        Pos1 = self.pos[0]
        Pos2 = self.pos[1]
        Pos3 = self.pos[2]

        pos1 = np.sqrt(Pos1[0] ** 2 + Pos1[1] ** 2 + Pos1[2] ** 2)
        pos2 = np.sqrt(Pos2[0] ** 2 + Pos2[1] ** 2 + Pos2[2] ** 2)
        pos3 = np.sqrt(Pos3[0] ** 2 + Pos3[1] ** 2 + Pos3[2] ** 2)

        C12 = np.cross(Pos1, Pos2)
        C23 = np.cross(Pos2, Pos3)
        C31 = np.cross(Pos3, Pos1)

        c23 = np.sqrt(C23[0] ** 2 + C23[1] ** 2 + C23[2] ** 2)

        if abs(np.dot(Pos1, C23)/pos1/c23) > self.tol:
            print('\n  Os vetores posição não são coplanares.\n\n')

        else:
            N = pos1*C23 + pos2*C31 + pos3*C12
            n = np.sqrt(N[0] ** 2 + N[1] ** 2 + N[2] ** 2)

            D = C12 + C23 + C31
            d = np.sqrt(D[0] ** 2 + D[1] ** 2 + D[2] ** 2)

            S = Pos1*(pos2 - pos3)+Pos2*(pos3 - pos1)+Pos3*(pos1 - pos2)

            Vel2 = np.sqrt(self.mi/n/d) * (np.cross(D, Pos2) / pos2 + S)

            print('\nResultados\nVelocidade para a posição (dois):',Vel2,
                  "\n\nOs elementos orbitais que representa a posição e a
velocidade (dois)")
```

Algoritmo para o procedimento do método direto:

```
class direto(object):
    R: np.ndarray
    V: np.ndarray
    def __init__(self, elk, mi):
        self.elk = elk
        self.mi = mi

    def pos_dir(self):
        momA = self.elk[0]
        exce = self.elk[1]
        incl = self.elk[2]
        AscR = self.elk[3]
        ArgP = self.elk[4]
        AnoV = self.elk[5]

        """ posição """
        rx = pow(momA, 2)/self.mi*(1/(1+exce*np.cos(AnoV)))*np.cos(AnoV)
        ry = pow(momA, 2)/self.mi*(1/(1+exce*np.cos(AnoV)))*np.sin(AnoV)
        rz = pow(momA, 2)/self.mi*(1/(1+exce*np.cos(AnoV)))*0

        """ velocidade """
        vx = self.mi/momA*-np.sin(AnoV)
        vy = self.mi/momA*(exce + np.cos(AnoV))
        vz = self.mi/momA*0

        Qx_1 = np.array([[np.cos(ArgP), np.sin(ArgP), 0],
                         [-np.sin(ArgP), np.cos(ArgP), 0],
                         [0, 0, 1]])

        Qx_2 = np.array([[1, 0, 0],
                         [0, np.cos(incl), np.sin(incl)],
                         [0, -np.sin(incl), np.cos(incl)]])

        Qx_3 = np.array([[np.cos(AscR), np.sin(AscR), 0],
                         [-np.sin(AscR), np.cos(AscR), 0],
                         [0, 0, 1]])

        Qx1 = np.dot(Qx_2, Qx_3)
        Qx = np.dot(Qx_1, Qx1)
        Qx_T = np.matrix.transpose(Qx)

        self.R = np.dot(Qx_T, np.array([rx, ry, rz]))
        self.V = np.dot(Qx_T, np.array([vx, vy, vz]))

        return self.R, self.V
```

Algoritmo para o procedimento do método inverso:

```
class inverso(object):
    def __init__(self, pos, vel):
        self.pos = pos
        self.vel = vel
        self.mi = 398600

    def pos_vel(self):
        """ posição (r)"""
        r_e1 = np.asscalar(pow(self.pos[0], 2)+pow(self.pos[1], 2)+
pow(self.pos[2], 2))
        self.r_e = np.sqrt(r_e1)

        """ velocidade (v)"""
        v_e1 = np.asscalar(pow(self.vel[0], 2) + pow(self.vel[1], 2) +
pow(self.vel[2], 2))
        self.v_e = np.sqrt(v_e1)

        """ multiplicação vetorial de posição com a velocidade"""
        vr_0 = np.dot(self.vel, self.pos)
        self.vr = (vr_0 / self.r_e)

    def mom_ang(self):
        self.pos_vel()

        """ Magnetude do momento angular (h)"""
        self.h = np.cross(self.pos, self.vel)
        h_e0 = np.asscalar(pow(self.h[0], 2) + pow(self.h[1], 2) +
pow(self.h[2], 2))
        return np.sqrt(h_e0)

    def incl(self):
        self.mom_ang()

        """ inclinação (i)"""
        i_1 = (self.h[2] / self.mom_ang())
        incl = np.arccos(i_1)
        '''
        if np.degrees(incl) > 90:
            print('\t\033[1;31mOrbita retrograda\033[m')
        '''
        """ multiplicação vetorial de K com h """
        k = np.array([0, 0, 1])
        self.n1 = np.cross(k, self.h)
        n1_e0 = np.asscalar(pow(self.n1[0], 2) + pow(self.n1[1], 2) +
pow(self.n1[2], 2))
        self.n1_e = np.sqrt(n1_e0)
        return incl

    def asc_ret(self):
```

```
        self.incl()
        """ ascenção da reta """
        return mth.acos((self.n1[0] / self.n1_e))

    def excen(self):
        self.asc_ret()

        """ excentricidade """
        self.e = (1 / self.mi * (((pow(self.v_e, 2) - self.mi / self.r_e)
* self.pos) - self.r_e * self.vr * self.vel))
        e_e0 = np.asscalar(pow(self.e[0], 2) + pow(self.e[1], 2) +
pow(self.e[2], 2))
        return np.sqrt(e_e0)

    def arg_per(self):
        self.excen()

        """ argumento do perigeu """
        return mth.acos(np.dot(self.n1, self.e) / (self.n1_e *
self.excen()))

    def anom_verd(self):
        self.arg_per()

        """ anomalia Verdadeira """
        return mth.acos(np.dot(self.e, self.pos) / (self.excen() *
self.r_e))

    def sem_Ex_maior(self):
        """ Raio do Perigeu """
        rp = pow(self.mom_ang(), 2) / self.mi * (1 / (1 + self.excen() *
np.cos(0)))
        """ Raio do Apogeu """
        ra = pow(self.mom_ang(), 2) / self.mi * (1 / (1 + self.excen() *
np.cos(180)))
        """ Semi eixo maior """
        return 0.5 * (int(rp + ra))

    def anom_exce(self):
        """ Anomalia Excentrica """
        return 2 * mth.atan(np.tan(self.anom_verd() / 2) / np.sqrt((1 +
self.excen()) / (1 - self.excen())))

    def anom_media(self):
        """ Anomalia Media """
        return self.anom_exce() - self.excen() * np.sin(self.anom_exce())

    def periodo(self):
        """ periodo """
        return 2 *np.pi/np.sqrt(self.mi)*pow(self.sem_Ex_maior(),1.5)
```

```
    def mov_med(self):
        """ Movimento Médio """
        return 2 * np.pi / self.periodo()

    def anom_excI(self):
        """ Anomalia Excentrica Inicial """
        return mth.atan(np.sqrt((1 - self.excen()) / (1 + self.excen()))
* (np.tan(self.anom_verd() / 2))) * 2

    def tempo_inic(self):
        """ Tempo Inicial """
        return (self.anom_excI() - self.excen() *
np.sin(self.anom_excI())) / self.mov_med()
```

Algoritmo para determinar a posição da Lua:

```
class pos_Lunar:
    To: float
    Obq: float
    LonL: float
    LatL: float
    HP: float
    def __init__(self, data):
        self.data = data
        self.RT = 6378

    def ALS_lunar_lat_lon(self):
        T = (self.data - 2451545.0)/36525

        long = 218.32 + 481267.881 * T\
        + 6.29 * np.sin(np.radians(135.0 + 477198.87 * T)) - 1.27 *
np.sin(np.radians(259.3 - 413335.36 * T))\
        + 0.66 * np.sin(np.radians(235.7 + 890534.22 * T)) + 0.21 *
np.sin(np.radians(269.9 + 954397.74 * T))\
        - 0.19 * np.sin(np.radians(357.5 + 35999.05 * T)) - 0.11 *
np.sin(np.radians(186.5 + 966404.03 * T))

        e_long = np.mod(long, 360)
        lat = 5.13 * np.sin(np.radians(93.3 + 483202.02 * T)) + 0.28 *
np.sin(np.radians(228.2 + 960400.89 * T))\
        - 0.28 * np.sin(np.radians(318.3 + 6003.15 * T)) - 0.17 *
np.sin(np.radians(217.6 - 407332.21 * T))

         e_lat = np.mod(lat, 360)
        h_p = 0.9508\
        + 0.0518 * np.cos(np.radians(135.0 + 477198.87 * T)) + 0.0095 *
np.cos(np.radians(259.3 - 413335.36 * T))\
        + 0.0078 * np.cos(235.7 + 890534.22 * T) + 0.0028 * np.cos(269.9
+ 954397.74 * T)
```

```
        h_par = np.mod(h_p, 360)
        obliquity = 23.439291 - 0.0130042 * T

        l = np.cos(np.radians(e_lat)) * np.cos(np.radians(e_long))
        m = np.cos(np.radians(obliquity)) * np.cos(np.radians(e_lat)) *
np.sin(np.radians(e_long)) - \
            np.sin(np.radians(obliquity)) * np.sin(np.radians(e_lat))
        n = np.sin(np.radians(obliquity)) * np.cos(np.radians(e_lat)) *
np.sin(np.radians(e_long)) + \
            np.cos(np.radians(obliquity)) * np.sin(np.radians(e_lat))
        dist = self.RT / np.sin(np.radians(h_par))
        r_lua = dist * np.array([l, m, n])
        return r_lua
```

Algoritmo para determinar a atração lunar:

```
class pert_lunar(object):
    def __init__(self, elk, JD0):
        self.hors = 3600
        self.days = 24*self.hors
        self.mi = 398600
        self.mL = 4903
        self.RE = 6378
        self.a0 = elk[0]          # semimajor axis(km)
        self.e0 = elk[1]          # eccentricity
        self.w0 = elk[2]          # argument of perigee(rad)
        self.RA0 = elk[3]         # right ascension(rad)
        self.i0 = elk[4]          # inclination(rad)
        self.TA0 = elk[5]         # true anomaly(rad)
        self.JD0 = JD0            # Julian Day

    def solveit(self):
        momA0 = np.sqrt(self.mi * self.a0 * (1 - self.e0 ** 2))
        coe0 = [momA0, self.e0, self.i0, self.RA0, self.w0, self.TA0]
        t0 = 0
        tf = 60 * self.days
        nout = 400
        mu = self.mi
        y0 = coe0

        def deriv(t, f):
            h = f[0]
            e = f[1]
            i = f[2]
            RA = f[3]
            w = f[4]
            TA = f[5]
```

```
            phi = w + TA
            elk = np.array([h, e, i, RA, w, TA])
            R, V = direto(elk, self.mi).pos_dir()
            r = np.sqrt(R[0] ** 2 + R[1] ** 2 + R[2] ** 2)
            ur = R / r
            W = np.cross(R, V)
            uh = W / np.sqrt(W[0] ** 2 + W[1] ** 2 + W[2] ** 2)
            S = np.cross(uh, ur)
            us = S / np.sqrt(S[0] ** 2 + S[1] ** 2 + S[2] ** 2)

            DJ = self.JD0 + t / self.days

            R_m = atraçãoLunar(DJ).ALS_lunar_lat_lon()
            r_m = np.sqrt(R_m[0]** 2 + R_m[1]** 2 + R_m[2]** 2)
            R_rel = R_m - R
            r_rel = np.sqrt(R_rel[0]** 2 + R_rel[1]** 2 + R_rel[2]** 2)
            q = np.dot(R, (2 * R_m - R)) / r_m ** 2
            F = (q ** 2 - 3 * q + 3) * q / (1 + (1 - q) ** 1.5)
            ap = self.mL / r_rel ** 3 * (F * R_m - R)

            """ Perturbação da aceleração """
            apr = np.dot(ap, ur)
            aph = np.dot(ap, uh)
            aps = np.dot(ap, us)

            hdot = r*aps
            edot = h/mu*np.sin(TA)*apr + 1/mu/h*((h**2 + mu*r)*np.cos(TA)
+ mu*e*r)*aps
            RAdot = r/h/np.sin(i)*np.sin(phi)*aph
            idot = r/h*np.cos(phi)*aph
            wdot = - h*np.cos(TA)/mu/e*apr + (h**2 +
mu*r)/mu/e/h*np.sin(TA)*aps - r*np.sin(phi)/h/np.tan(i)* aph
            TAdot = h/r**2 + 1/e/h*(h**2/mu*np.cos(TA)*apr - (r +
h**2/mu)*np.sin(TA)*aps)
            return [hdot, edot, idot, RAdot, wdot, TAdot]

        coe = rk45(deriv, [t0, tf], y0)

        h_f, e_f, i_f, RA_f, w_f, TA_f = coe.y
        t_f = coe.t

        return i_f - self.i0, RA_f - self.RA0, w_f - self.w0
```