MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

ANÁLISE E SIMULAÇÃO DE REENTRADAS ATMOSFÉRICAS CONTROLADAS

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

Bolsista - Ariane de Oliveira Braga (ETEP Faculdades, Bolsista PIBIC/CNPq)

E-mail: arianebraga01@hotmail.com

Orientador - Dr. Marcelo Lopes de Oliveira e Souza (DMC/ETE/INPE)

E-mail: marcelo@dem.inpe.br

Julho de 2007

SUMÁRIO

LISTA DE FIGURAS E TABELAS

Páginas

CAPÍTULO 1

Introdução

BREVE HISTÓRICO: No período de Fevereiro de 2006 até Julho de 2006, foram realizados: 1) Nossa introdução ao tema: "Reentradas Atmosféricas Controladas", tratando do retorno controlado de um veículo espacial para dentro da atmosfera da Terra; 2) Uma análise de alguns casos recentes como o do "Compton Gama Ray Observatory-CGRO" para melhor entendimento do trabalho; 3) Um estudo sobre a transferência inversa de Hohmann e a transferência inversa de Breakwell, dentre diversas estratégias para realizar o decaimento orbital controlado envolvendo órbitas circulares e elípticas usando modelos já existentes como o do CGRO. Os resultados e conclusões estão no Relatório Final "Análise e Simulação de Reentradas Atmosféricas Controladas" e foram apresentados no Seminário de Iniciação Científica (SICINPE 2006), nos dias 11 e 12 de Julho de 2006.

OBJETIVOS E ESTRUTURA DESTE RELATÓRIO: O objetivo deste Relatório é descrever os trabalhos realizados no período de Agosto de 2006 até o início de Julho de 2007, dando continuidade aos resultados da pesquisa anterior incluindo: 1) programas do Matlab para simular o decaimento orbital controlado usando a transferência inversa de Hohmann e a transferência inversa de Breakwell; 2) o cálculo da reentrada simplificada do Observatório Compton de Raios Gama que foi um dos casos recentes apresentados no Relatório Final de Julho de 2006; 3) o estudo de transferência orbital com força de arrasto atmosférico,simulando programas no Matlab.

CAPÍTULO 2 - Metodologia

2.1 - Transferência Inversa de Hohmann

A Transferência Inversa de Hohmann com decaimento orbital controlado é a partir de uma órbita circular, que são aplicados dois impulsos necessários para desacelerar o movimento orbital do satélite e levá-lo a outra órbita circular menor. Para simular essa tranferência foi realizado alguns programas no Matlab como modelos para melhor compreensão, que são mostrado a seguir:

O programa “orb_hoh2_old ” (vide Apêndice 1, Programa 1), simula a aplicação de dois impulsos. Numa órbita inicialmente circular, é aplicado ao satélite o impulso no ponto da órbita que representa o apogeu da órbita elíptica de transferência que começa a ser percorrida pelo satélite. Quando satélite atinge o perigeu dessa órbita elíptica de transferência outro impulso é aplicado de modo a torná-la circular novamente no raio final desejado, como mostrado na Figura 1 abaixo:

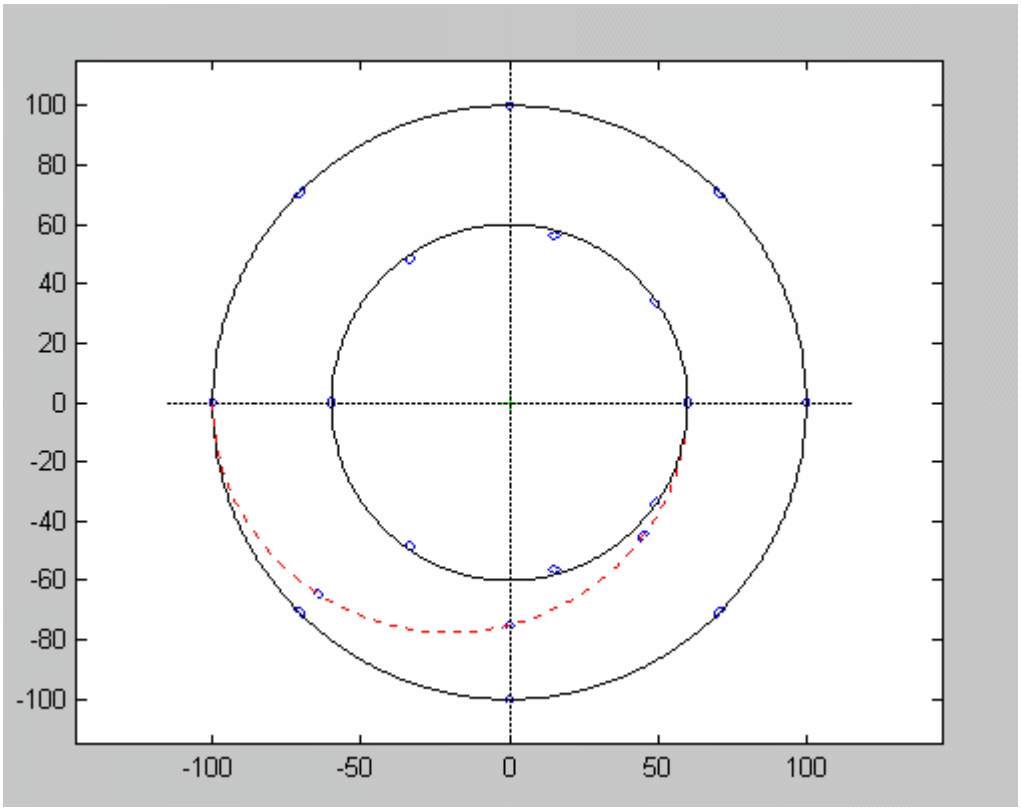

Figura 1- Uma Transferência Inversa de Hohmann.

O programa “espiral_elipsecte ”(vide Apêndice 1, Programa 2), simula a parte final de uma reentrada com uma trajetória final em espiral até atingir uma região segura da Terra, como mostrado na Figura 2 abaixo:

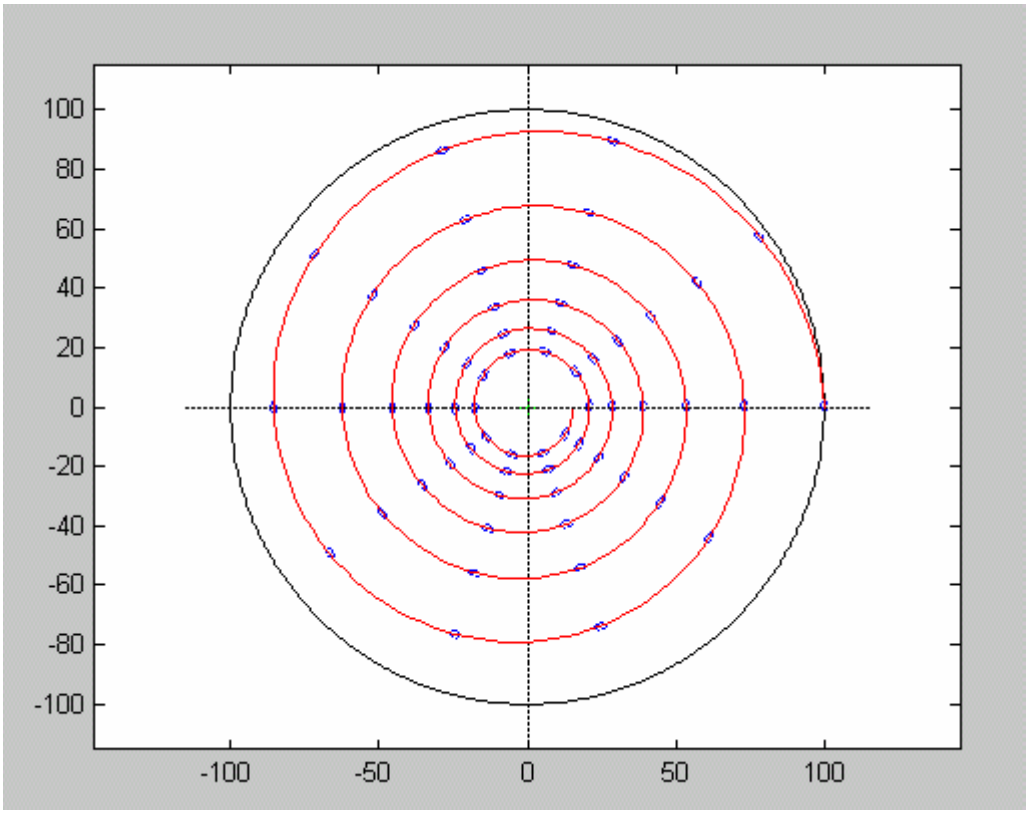

Figura 2- Trajetória Final em Espiral

O programa “orb_hoh4 ”(vide Apêndice 1, Programa 3), é formado pelos programas anteriores “orb_hoh2_old” e “espiral_elipsecte ” que simula aplicação dos dois impulsos necessários para desacelerar o movimento orbital do satélite e levá-lo a outra órbita circular menor, finalizando a órbita final com espiral visando impactar uma região segura da superfície da Terra, como mostra a Figura 3 abaixo:

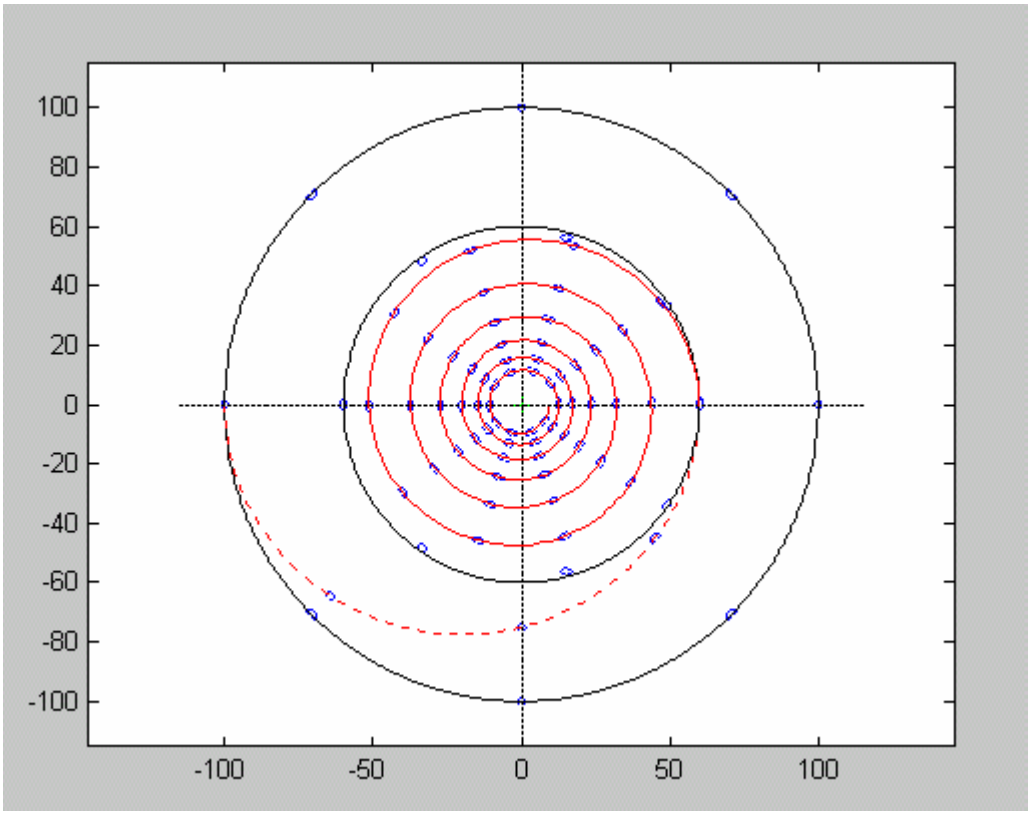

Figura 3 – Uma Transferência Inversa de Hohmann e Trajetória Final em Espiral.

O programa do Matlab “orb_hoh2”(vide Apêndice 1, Programa 4), simula duas transferências inversas de Hohmann levando o satélite para uma órbita circular menor e finalizando a órbita com o programa “espiral_elipsecte ” com a órbita espiral, como mostra a Figura 4 abaixo:

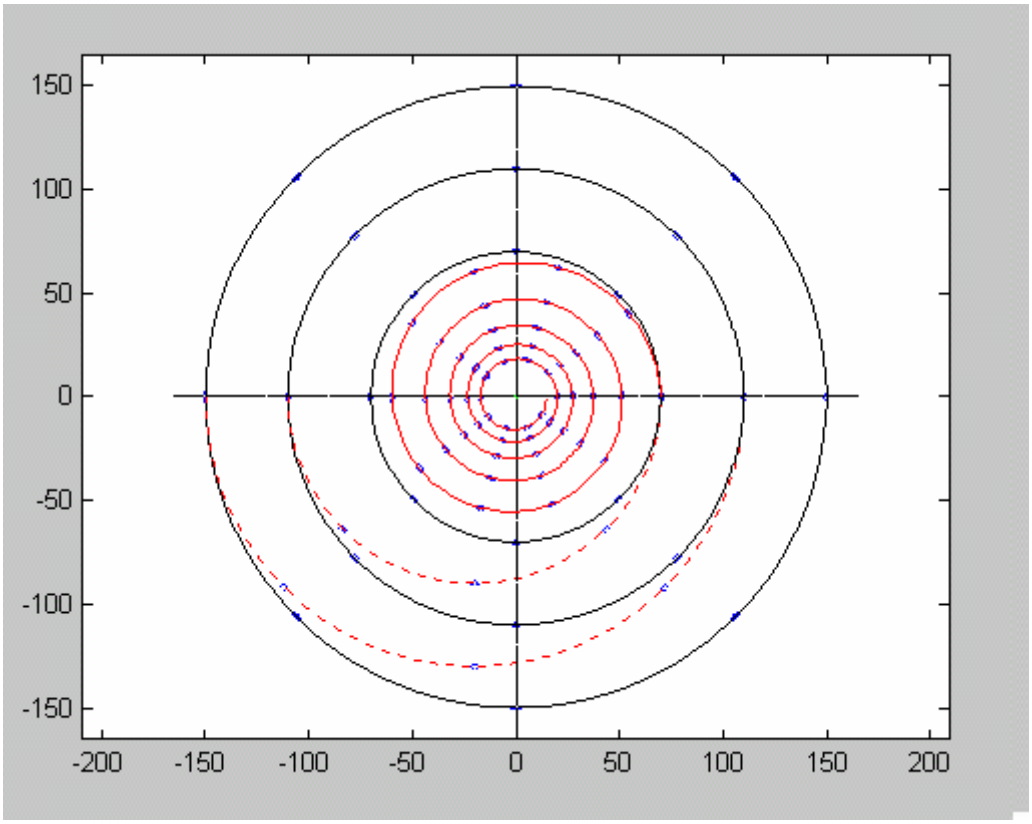

Figura 4- Duas Transferências Inversas de Hohmann e Trajetória Final em Espiral.

O programa do Matlab “orb_hoh22 ”(vide Apêndice 1, Programa 5), simula três transferências inversas de Hohmann levando o satélite para uma órbita circular menor e finalizando a órbita com o programa “espiral_elipsecte ” com a órbita espiral, como mostra a Figura 5 abaixo:

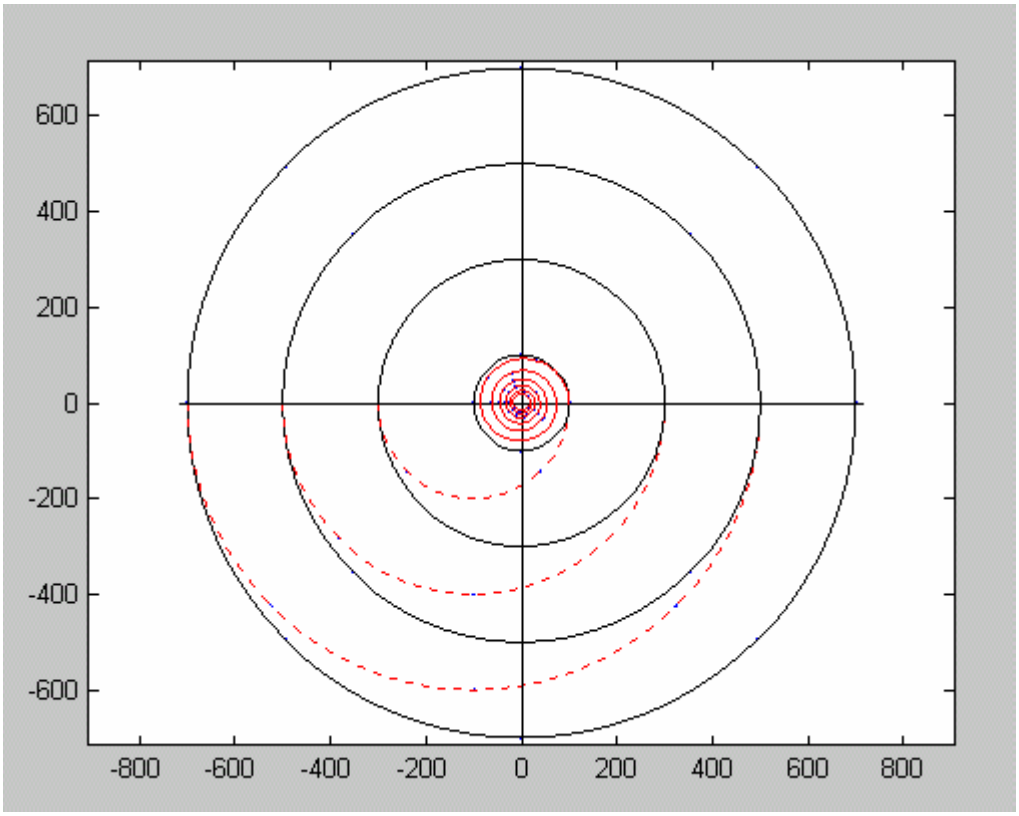

Figura 5 - Três Transferências Inversas de Hohmann e Trajetória Final em Espiral.

Para determinar os impulsos utilizamos as seguintes equações:

A partir da órbita inicial, o primeiro impulso é aplicado quando o veículo está no apogeu da órbita elíptica de transferência, de acordo com as equações:

$$\frac{\Delta V_i}{Vc_i} = \sqrt{\frac{\frac{2 r_f}{r_i}}{1 + \frac{r_f}{r_i}}} - 1$$

Portanto:

$$\frac{\Delta V_i}{Vc_i} = \sqrt{\frac{2 r_f}{r_i + r_f}} - 1$$

O segundo impulso é aplicado quando o veículo está no perigeu da órbita elíptica de transferência, e esse impulso circulariza a órbita no raio final desejado, de acordo com as equações:

$$\frac{\Delta V_f}{Vc_i} = \sqrt{\frac{1}{\frac{r_f}{r_i}}} - \sqrt{\frac{\frac{2r_f}{r_i}}{\frac{r_f}{r_i}(1+\frac{r_f}{r_i})}}$$

Portanto:

$$\frac{\Delta V_f}{Vc_i} = \sqrt{\frac{r_i}{r_f}} - \sqrt{\frac{2 r_i}{r_i + r_f}}$$

Onde: Vc_i é velocidade inicial na órbita circular.

ΔV_i e ΔV_f são os impulsos nas órbitas inicial e final.

r_i e r_f são raios da órbita inicial e da órbita final.

Utilizando as equações anteriores, temos a seguir a simulação de um modelo da Transferência Inversa de Hohmann.

O programa do Matlab “orb_raio0 ”(vide Apêndice 1, Programa 6), simula a aplicação de dois impulsos, onde temos o raio inicial, raio final das órbitas circulares e o delta velocidade que são os impulsos aplicados nas órbitas iniciais e finais, como mostra a Figura 6 abaixo:

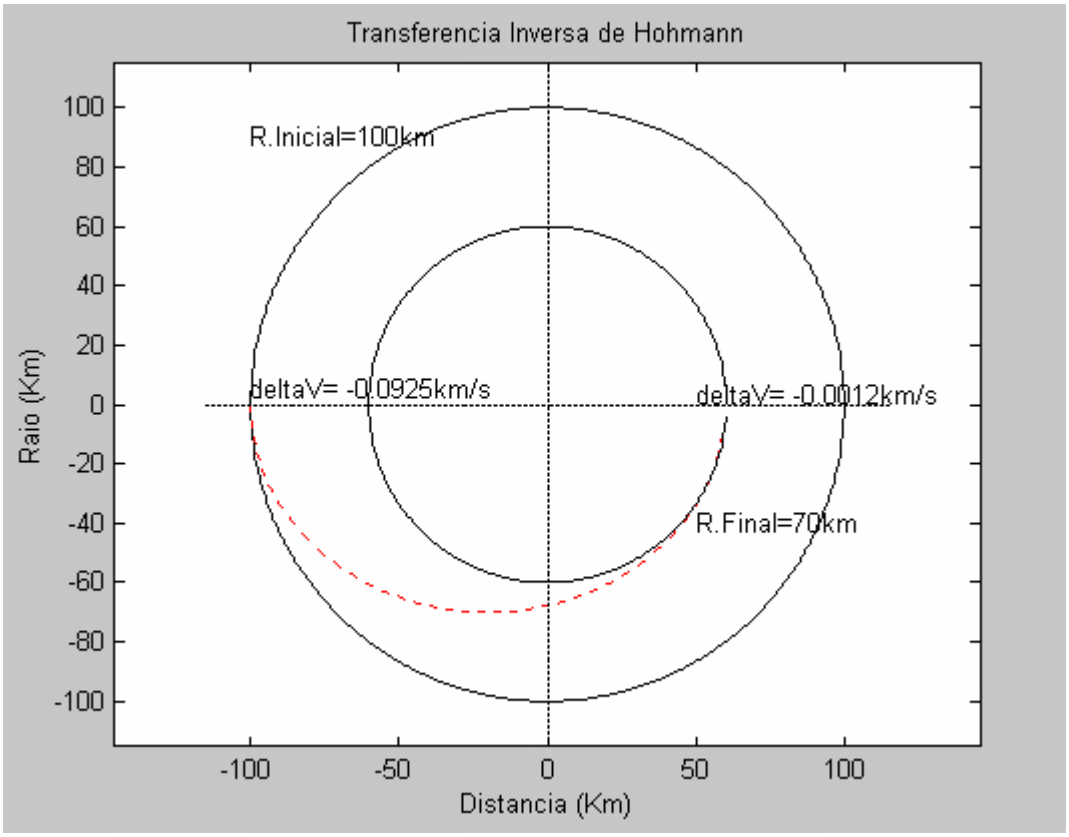

Figura 6 –Transferência Inversa de Hohmann

2.2 - Transferência Inversa de Breakwell

A Transferência Inversa de Breakwell é uma órbita elíptica, onde é aplicado um impulso necessários para desacelerar o movimento orbital do satélite e levá-lo a outra órbita elíptica. A partir da órbita inicial, o impulso é aplicado quando o veículo está no apogeu da órbita elíptica inicial, reduzindo o perigeu e alterando a excentricidade. Para simular essa transferência foi realizado alguns programas no Matlab como modelos para melhor compreensão, que são mostrado a seguir:

O programa “orb_elip_ff” (vide Apêndice 2, Programa 1), simula repetidamente a aplicação de um impulso quando o veículo está no apogeu da órbita inicial=1 e reduzindo o perigeu da órbita final=2, como mostra a Figura 7 abaixo:

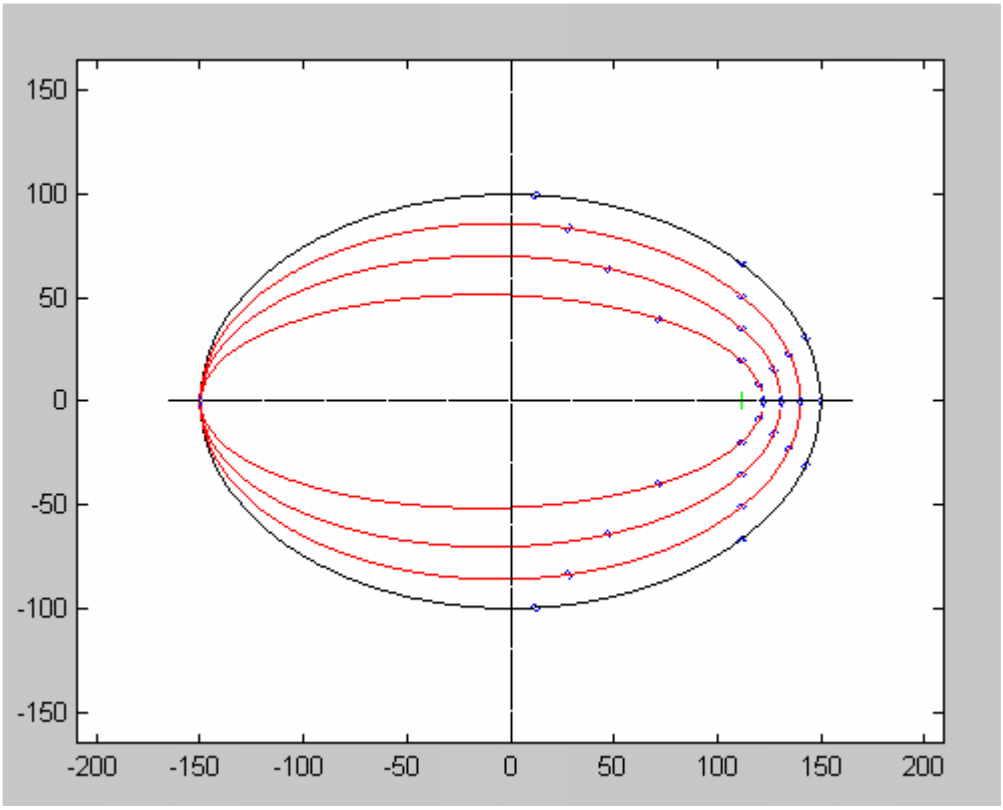

Figura 7- Três Transferências Inversas de Breakwell.

O programa “espiral_breakwell ”(vide Apêndice 2, Programa 2), simula a parte final de uma reentrada com uma trajetória final em espiral até atingir uma região segura da Terra, como mostra a Figura 8 abaixo:

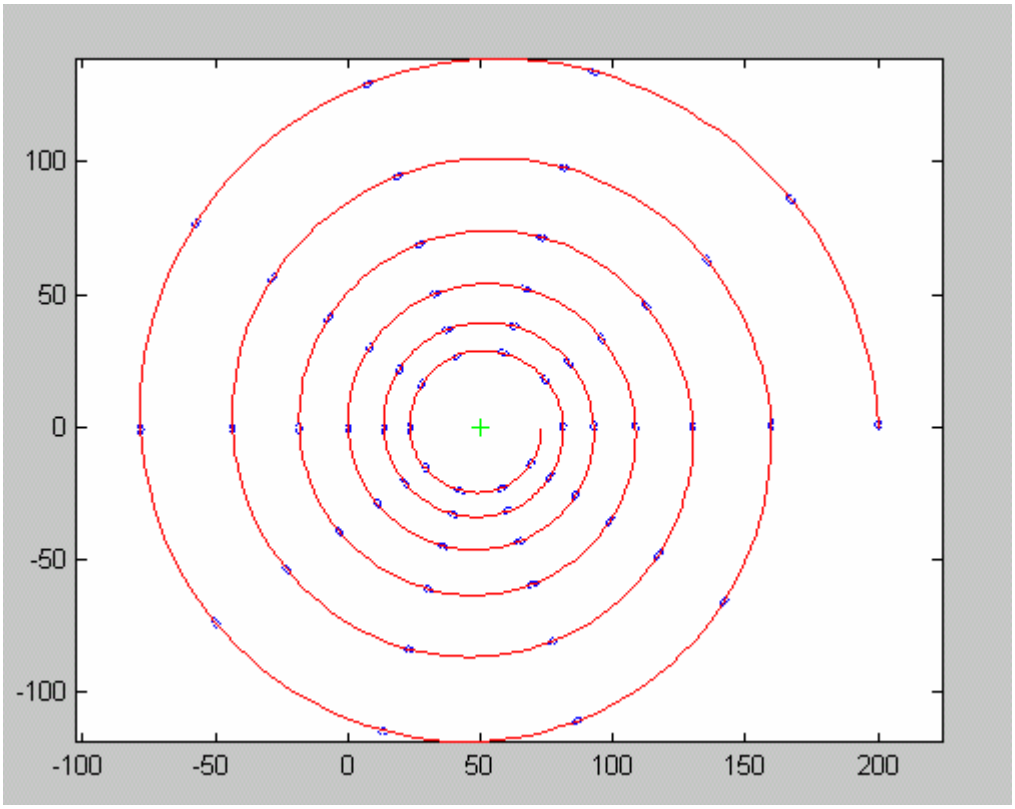

Figura 8- Trajetória Final em Espiral.

O programa “breakwell_auto1 ”(vide Apêndice 2, Programa 3), é formado pelos programas anteriores “orb_elip_ff ”e“espiral_breakwell ” que simula repetidamente a aplicação de um impulso quando o veículo está no apogeu da órbita inicial=1 e reduzindo o perigeu da órbita final=2, finalizando a órbita com o programa “espiral_breakwell ” com a órbita espiral até atingir uma região segura da superfície da Terra, como mostra a Figura 9 abaixo:

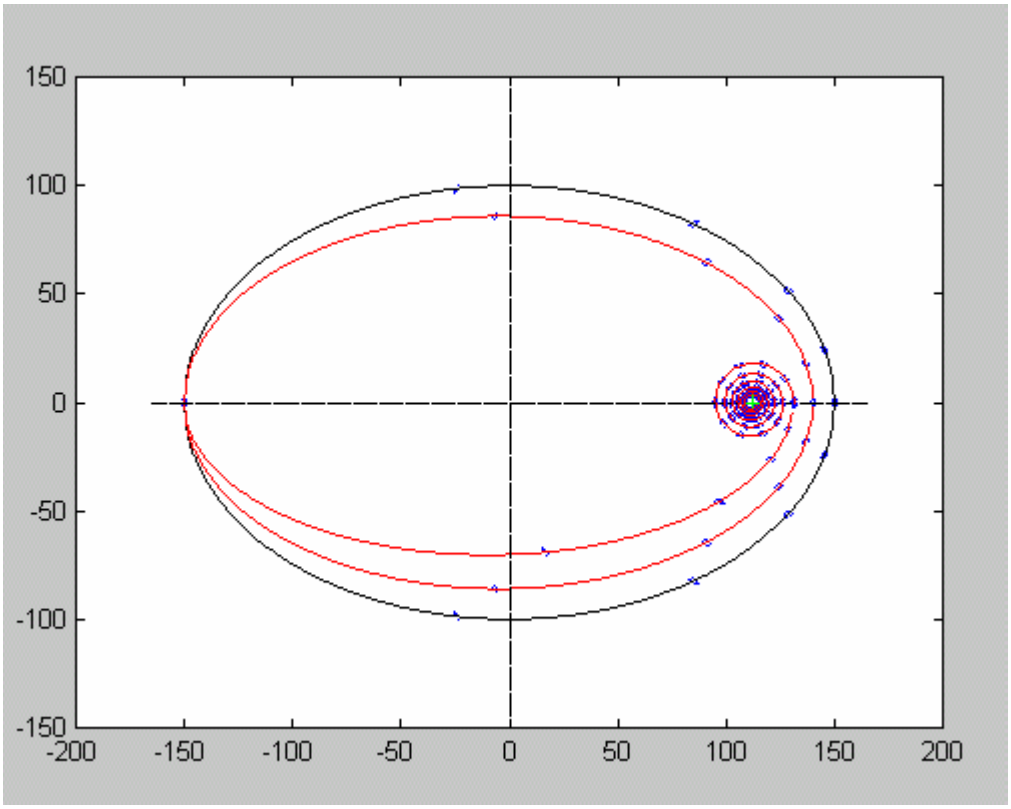

Figura 9- Duas Transferências Inversas de Breakwell e Trajetória Final em Espiral.

O programa “breakwell1”(vide Apêndice 2, Programa 4), simula repetidamente a aplicação de alguns impulsos que mostra os valores do semi eixo maior, semi eixo menor e também os valores da excentricidade que vai variando de acordo com cada impulso, como mostra a Figura 10 abaixo:

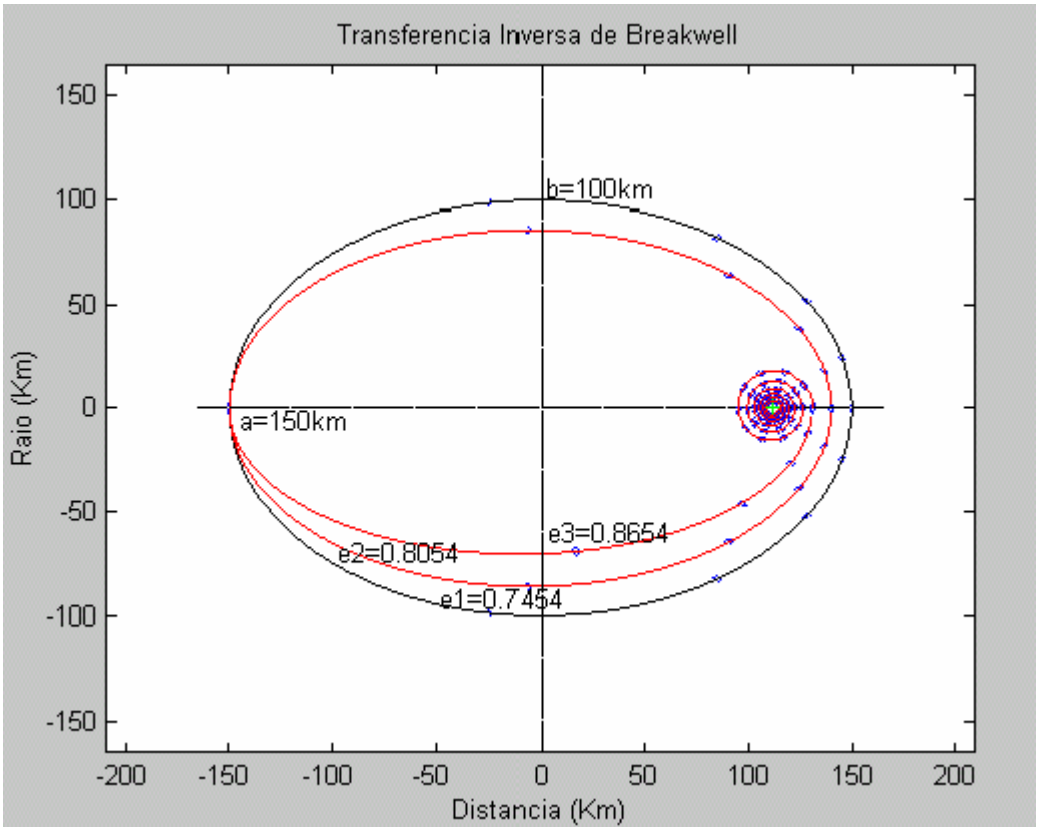

Figura 10- Transferências Inversas de Breakwell

2.3 - Simulações Numéricas

No Relatório Final de Julho de 2006 foi realizado uma pesquisa sobre o Observatório Compton de Raios Gama que foi lançado em abril de 1991 e teve sua reentrada na atmosfera da Terra de forma controlada em 4 de Junho de 2000, após encerrar suas operações devido a falhas de um giroscópio.

Seqüência Simplificada da Reentrada do CGRO

Em 28 de maio 2000

- As cargas desnecessárias da configuração/ verificação geral da nave espacial (instrumentos) foram desligadas.
- A engenharia testa a execução de comandos
- Vida de uma órbita: diversos anos

Em 30 de maio 2000 - Impulso número 1

- O impulso1 diminuiu o perigeu de 510 km para 350 km
- Vida de uma órbita: ~1 ano

Em 31 de maio 2000 - Impulso número 2

- O impulso 2 diminuiu o perigeu de 350 km para 250 km
- Vida de uma órbita: ~80 dias

Em 4 de junho 2000 - Impulso número 3

- O impulso 3 diminuiu o perigeu de 250 km de 150 km
- Vida de uma órbita: ~3 dias

Em 4 de junho 2000 - Impulso número 4

- A órbita seguinte, executa o impulso 4 para reentrar na atmosfera.
- A nave espacial reentra no alvo

De acordo com os dados obtidos da seqüência simplificada da reentrada do Observatório Compton de Raios Gama, e utilizando as equações abaixo:

$$a_2 = \frac{a_1(1+e_1)}{(1+e_2)} \qquad V^2 = \mu\left(\frac{1}{r} - \frac{1}{a}\right)$$

Obtivemos os seguintes valores:

Tabela 1- Sequência Simplificada da Reentrada do CGRO

Sequência Simplificada da Reentrada do CGRO			
Data	**Altura do Perigeu (R_t =6378 km)**	**Exc. e Altura Finais**	**ΔV (km/s)**
30 de Maio de 2000	510km para 350km	e= 0,186 a= 430 km	ΔV= -0,045
31 de Maio de 2000	350km para 250km	e= 0,168 a= 300 km	ΔV= -0,106
04 de Junho de 2000	250km para 150km	e= 0,25 a= 200 km	ΔV= -0,057
04 de Junho de 2000	150km para 84km	e= 0,282 a= 117 km	ΔV= -0,069

2.4 - Transferência Orbital com Força de Arrasto Atmosférico

Para demonstrar as transferências orbitais com arrasto atmosférico integraremos numericamente as equações obtidas a partir das Leis de Newton envolvendo as forças gravitacional e de arrasto atmosférico, a saber:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = -\frac{\mu}{|\vec{r}|^3}\begin{bmatrix} x \\ y \end{bmatrix} - \frac{Fa}{|\vec{v}|}\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}$$

Onde: μ=398600 km é a constante gravitacional da Terra;

$|\vec{r}| = (x^2 + y^2)^{1/2}$ é o vetor posição ;

$|\vec{v}| = (\dot{x}^2 + \dot{y}^2)^{1/2}$ é o vetor velocidade;

$Fa = \frac{1}{2} C_\rho A \rho v^2$ equação da força de arrasto atmosférico, onde $C\rho$ é o coeficiente de arrasto; A é a área do satélite; ρ é a densidade atmosférica; v vetor velocidade.

O programa " transf_arrasto_modif2 " (vide Apêndice 3, Programa 1), simula as transferências orbitais com força de arrasto atmosférico. Para realizar o programa foi utilizado as equações anteriores, os comandos de ode com o método de Runge-Kutta 4 ou 5 no Matlab, densidade (constante) e a densidade variando em função da altitude. Desenvolvendo assim o decaimento da orbital, como mostra as Figuras 11 abaixo:

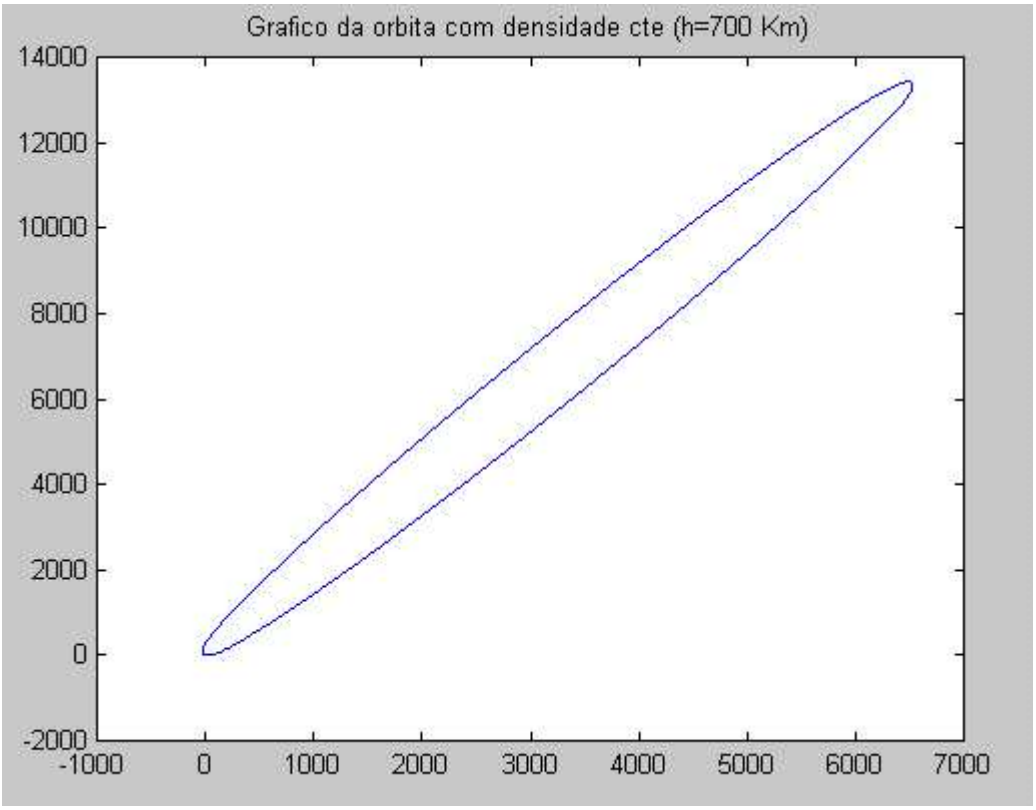

Figura 11- Decaimento com Força de Arrasto Atmosférico

A seguir temos o programa anterior que foi ampliado para melhor visualização do decaimento orbital, como mostra a Figura 12 abaixo:

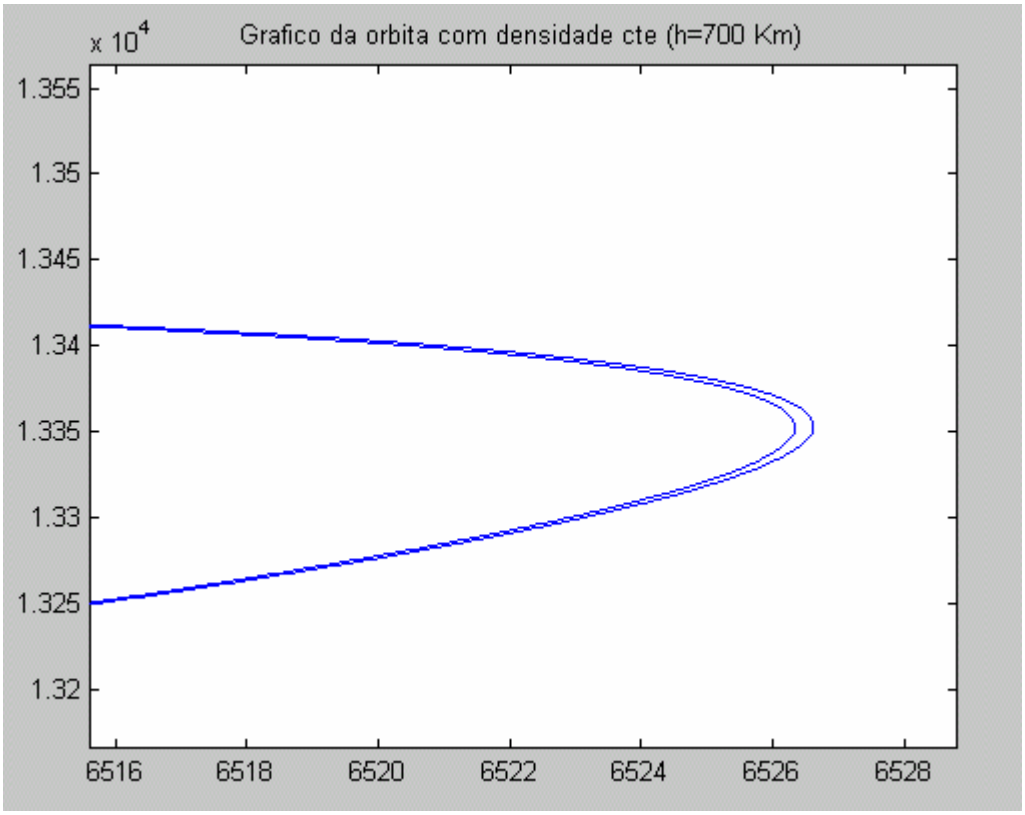

Figura 12-Decaimento Orbital Ampliado

A densidade atmosférica em função da altitude como mostra a tabela 2 :

Tabela 2 - Densidade Atmosférica em Função da Altitude

Altitude (Km)	Densidade (km/m^3)	Altitude (Km)	Densidade (km/m^3)	Altitude (Km)	Densidade (km/m^3)
0	1.225*10^0	80	1.905*10^-5	200	2.789*10^-10
25	3.899*10^-2	85	8.337*10^-6	250	7.248*10^-11
30	1.774*10^-2	90	3.396*10^-6	300	2.418*10^-11
35	8.279*10^-3	95	1.343*10^-6	350	9.158*10^-12
40	3.972*10^-3	100	5.597*10^-7	400	3.725*10^-12
45	1.995*10^-3	110	9.661*10^-8	450	1.585*10^-12
50	1.057*10^-3	120	2.438*10^-8	500	6.967*10^-13
55	5.821*10^-4	130	8.484*10^-9	600	1.454*10^-13
60	3.206*10^-4	140	3.845*10^-9	700	3.614*10^-14
65	1.718*10^-4	150	2.070*10^-9	800	1.170*10^-14
70	8.770*10^-5	160	1.244*10^-9	900	5.245*10^-15
75	4.178*10^-5	180	5.464*10^-10	1000	3.019*10^-15

A seguir temos a Figura 13 que simula o decaimento orbital onde é usado os dados da tabela 2 que representa a densidade atmosférica variando em função da altitude.

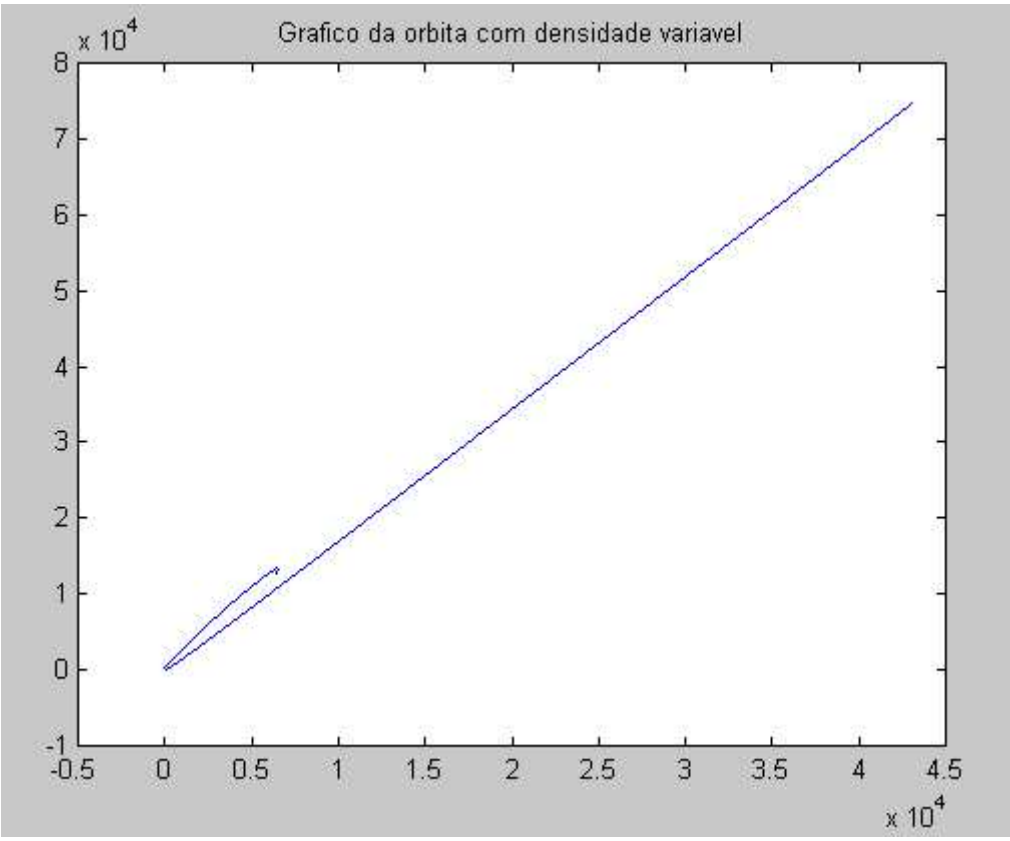

Figura 13- Transferência Orbital com Densidade Variando

A seguir temos a Figura 14 que está comparando as duas orbitas anteriores para uma melhor compreensão do decaimento.

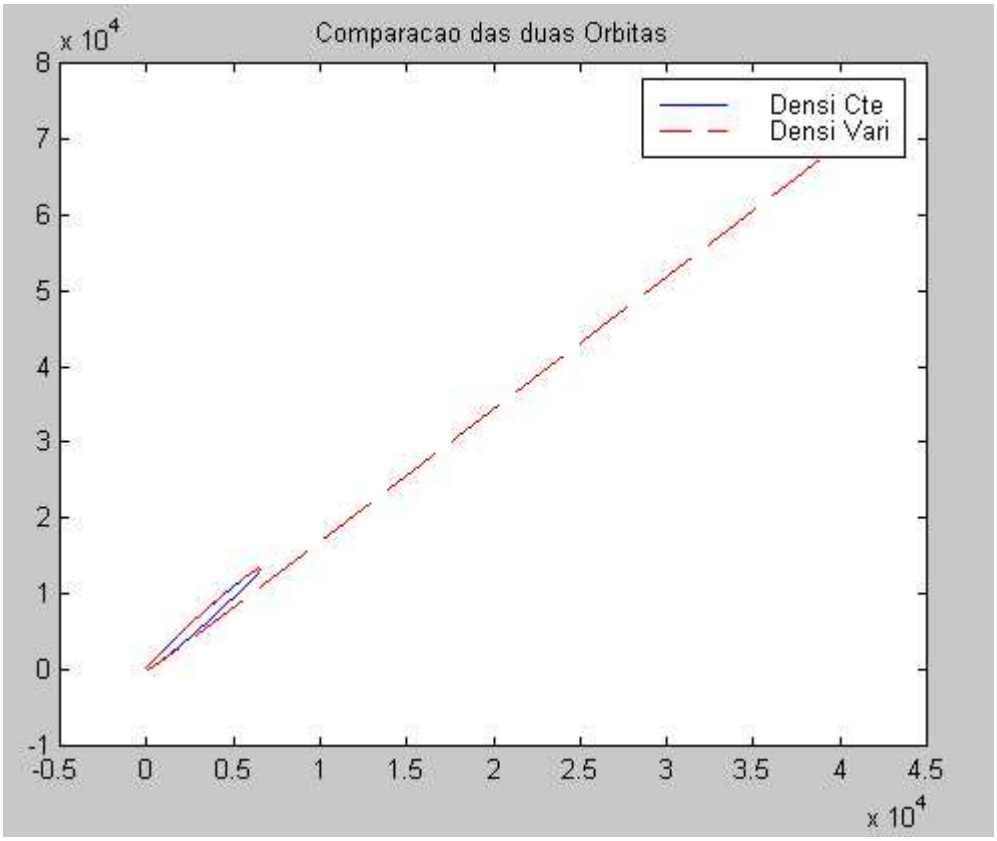

Figura 14- Comparação entre Densidade Constante e Variando

A seguir temos a Tabela 3 que representa o modelo exponencial da densidade:

$$\rho = e^{\frac{-h}{6.8}} \qquad \text{com} \quad 0 \leq h \prec 110 \text{, onde } h \text{ é altitude.}$$

Tabela 3 – Modelo Exponencial

Altitude (Km)	Densidade da Tabela (km/m^3)	Densidade (km/m^3)
0	1.225*10^0	1
25	3.899*10^-2	2.53*10^-2
30	1.774*10^-2	1.21*10^-2
35	8.279*10^-3	5.816*10^-3
40	3.972*10^-3	2.788*10^-3
45	1.995*10^-3	1.336*10^-3
50	1.057*10^-3	6.407*10^-4
55	5.821*10^-4	3.071*10^-4
60	3.206*10^-4	1.472*10^-4
65	1.718*10^-4	7.057*10^-5
70	8.770*10^-5	3.383*10^-5
75	4.178*10^-5	1.621*10^-5
80	1.905*10^-5	7.774*10^-6
85	8.337*10^-6	3.726*10^-6
90	3.396*10^-6	1.786*10^-6
95	1.343*10^-6	8.563*10^-7
100	5.597*10^-7	4.105*10^-7
110	9.661*10^-8	9.432*10^-8

O programa “arrast” (vide apêndice 3, programa 2), simula o decaimento orbital com o modelo exponencial da densidade atmosférica utilizando os dados da tabela anterior, como mostra a Figura 15 abaixo:

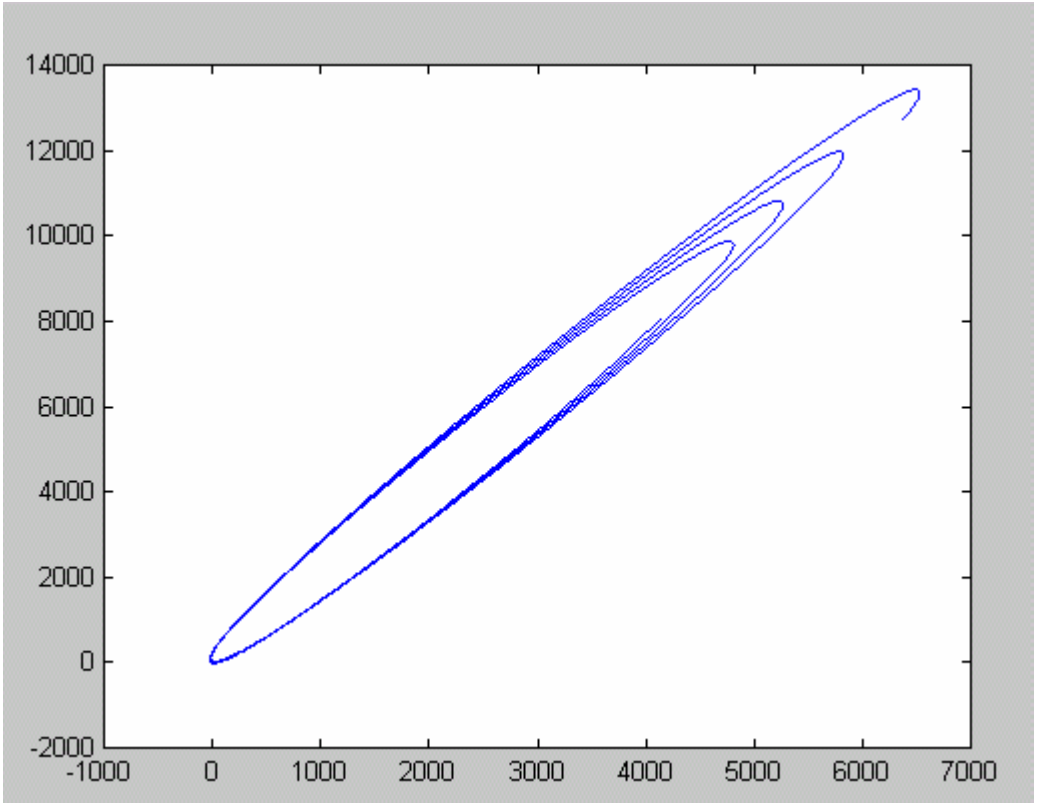

Figura 15- Transferência Orbital com Modelo Exponencial

CAPÍTULO 3

Conclusões, Comentários e Sugestões para o Prosseguimento do Trabalho

O trabalho anterior realizou análise e simulação de reentradas atmosféricas controladas de um satélite em final de órbita e início, utilizando o estudo de manobras de atitude e de transferência orbital visando otimizar o decaimento orbital controlado de um satélite e também o estudo da sua reentrada inteira ou de seus fragmentos, visando impactar uma região segura da superfície da Terra. Para o decaimento orbital controlado usamos a transferência inversa de Hohmann e a transferência inversa de Breakwell; e também a transferência com força de arrasto atmosférico. No relatório foram listados os programas em Matlab das transferências orbitais para melhor compreensão do trabalho. Com base nestes programas e neste modelo, objetiva-se, posteriormente, estudar as propriedades básicas desse processo. Assim, será possível analisar os problemas de colisão e interferência dos detritos espaciais com outros objetos encontrados no espaço como satélites, ônibus espaciais, e estações espaciais.

REFERÊNCIAS BIBLIOGRÁFICAS

1) KUGA, RAO, *Introdução à Mecânica Orbital,* INPE, São José dos Campos - SP, 1995.

.

2) JOHNSON, N.L., & MCKINIGHT, D.S. *Artificial Space Debris (Updated Edition)*. Krieger Pub. Co., Malabar, FL, USA, 1991.

3)CHOBOTOV, V.A. (ed.) *Orbital Mechanics (2 Ed.)* Reston, VA, USA, AIAA, 1996.

4) SOUZA, M.L.O., NUNES, D., *Forecasting Space Debris Distribution: A Measure Theory Approach,* 51st. International Astronautical Congress – IAC. Rio de Janeiro - RJ, 2-6 Out.2000, Paper IAA-00-IAA.6.4.07.

5) ROSSER, J.B. (ED.) *Space Mathematics, Part I.* American Mathematical Society, New York, NY, USA, 1996.

6) CHANDRASEKHAR, S. *Principles of Stellar Dynamics*. Chicago Univ. Press, Chigaco, IL, USA, 1942; e Dover Pub., New York, NY, USA, 1960.

7) CARNAHAN, Brice; LUTHER, H. A.; WILKES, James O. *Applied Numerical Methods/*. New York : John Wiley, c1969 604 p. : ISBN 471.135070 (enc.)

8) (Apostila) Curso de MATLAB 5.1*; Introdução à Solução de Problemas de Engenharia*; 2ª edição; Programa Prodenge / Sub-Programa Reenge; Universidade do Estado do Rio de Janeiro.

9) WERTZ, J.R. *Spacecraft Attitude Determination and Control*, Kluwer Academic Publishers, Dordrecht, Holland, 1978.

10) HANSELMAN,D., & LITTLEFIELD,B. *MatLab 5: Guia do Usuário, Versão do Estudante*, Makron Books,1999.

Apêndice A - Programas e simulações da órbita elíptica (Transferência de Hohmann) desenvolvido por Flávio Francesco Soares Schmidt em Matlab:

Programa 1:

```
function orb_hoh2_old(n,v,raio)
%valores recebidos
%n=8; n. de elipses em 1 volta
%v=2; n. de voltas
%raio inicial da orbita circular

a=raio; % -> semi-eixo horizontal
b=raio; % -> semi-eixo vertical

%armazena o valor de xo para manter fixo o perigeu
xp=a; %era xo;
%armazena o valor de yo para alterar a orbita
yp=b;  %era yo;

%definicao das variaveis da elipse (foco, excentricidade e
distancia)
%    a=xo;
%    b=yo;
    c=sqrt(a.^2-b.^2);
    e=c/a;
    daf=a.*(1+e);  %distancia do apogeu ao foco
%inicia o laço de repetiçao para fazer o giro completo na orbita
%************************************************
%** laço retirado, primeiro farei manualmente. **
%************************************************
%for rf=a:-20:raio-20*(v-1)
%inicio da manobra de Hohmann (percorrer a orbita circular
inicial)
    rf=a;

%definicao da orbita eliptica
    e=(daf./rf)-1;
    b=sqrt(rf.^2-(e.*rf).^2);

%*************************************************
%**      laço das elipses em orbita removido      **
%** por enquanto, as elipses serao desconsideradas **
%*************************************************
    for t=1:1:(n)
    w=2.*pi/(n);
    %raio=10;
    teta=w.*t;

%definicao da distancia do foco ao centro da elipse em orbita
    r=(rf.*(1-e.^2))/(1+e.*cos(teta));
```

```
%define o centro de cada elipse na orbita eliptica
    xe=e.*rf+(r.*cos(teta));%-(xp-xo);
    ye=r.*sin(teta);%-(yp-yo);

%giro equivalente ao angulo percorrido da orbita
giro=-teta;
%definiçao do crescimento da elipse
ey=2;%+0.02*t*(red+1);
ex=1;%+0.01*t*(red+1);

%elipse base
x=(-ex):0.05:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);

%elipse rotacionada e reposicionada de acordo com o centro (ex,ey)
x1=xe+x*cos(giro)+ysup.*sin(giro);
x2=xe+x*cos(-giro)+ysup.*sin(-giro);
ysup1=ye-x.*sin(giro)+ysup.*cos(giro);
yinf1=ye-x.*sin(giro)-ysup.*cos(giro);

%plotagem da figura de cada elipse modificada
plot(x1-(xp-rf),ysup1);
hold on;
plot(x2-(xp-rf),yinf1);
hold on;
end

%plotagem da orbita para referencia
ey=b;
ex=rf;
x=(-ex):0.5:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);
plot(x-(xp-rf),ysup,'-k');
hold on;
plot(x-(xp-rf),yinf,'-k');
hold on;
%plotagem do foco de cada orbita
plot(e.*rf-(xp-a),0,'+g');
hold on;
%end %laco for

%segundo passo da manobra de Hohmann (1o. tiro)
    rf=rf-20;
%definicao da orbita eliptica
    e=(daf./rf)-1;
    b=sqrt(rf.^2-(e.*rf).^2);

%plotagem das elipses na orbita eliptica de transiçao
    for t=1:1:(n/2)
    w=2.*pi/(n);
    %raio=10;
```

```
    teta=pi+w.*t;

%definicao da distancia do foco ao centro da elipse em orbita
    r=(rf.*(1-e.^2))/(1+e.*cos(teta));

%define o centro de cada elipse na orbita eliptica
    xe=e.*rf+(r.*cos(teta));%-(xp-xo);
    ye=r.*sin(teta);%-(yp-yo);

%giro equivalente ao angulo percorrido da orbita
giro=-teta;
%definiçao do crescimento da elipse
ey=2;%+0.02*t*(red+1);
ex=1;%+0.01*t*(red+1);

%elipse base
x=(-ex):0.05:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);

%elipse rotacionada e reposicionada de acordo com o centro (ex,ey)
x1=xe+x*cos(giro)+ysup.*sin(giro);
x2=xe+x*cos(-giro)+ysup.*sin(-giro);
ysup1=ye-x.*sin(giro)+ysup.*cos(giro);
yinf1=ye-x.*sin(giro)-ysup.*cos(giro);

%plotagem da figura de cada elipse modificada
plot(x1-(xp-rf),ysup1);
hold on;
plot(x2-(xp-rf),yinf1);
hold on;
end

%plotagem da orbita para referencia
ey=b;
ex=rf;
x=(-ex):0.5:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2); %somente percorre 1/2 orbita
%plot(x-(xp-rf),ysup,'r'); %demonstraçao da continuidade da orbita
eliptica nao percorrida
hold on;
plot(x-(xp-rf),yinf,':r');
hold on;
%plotagem do foco de cada orbita
plot(e.*rf-(xp-rf),0,'+g');
hold on;

%terceiro passo da manobra de Hohmann (2o. tiro)
%retorno a orbita circular, agora com raio final
    b=rf*(1-e);
    rf=b;
```

```
%plotagem das elipses na orbita circular final
        for t=1:1:(n)
    w=2.*pi/(n);
    %raio=10;
    teta=w.*t;

%definicao da distancia do foco ao centro da elipse em orbita
    r=(rf.*(1-e.^2))/(1+e.*cos(teta));

%define o centro de cada elipse na orbita eliptica
  xe=e.*rf+(r.*cos(teta));%-(xp-xo);
  ye=r.*sin(teta);%-(yp-yo);

%giro equivalente ao angulo percorrido da orbita
giro=-teta;
%definiçao do crescimento da elipse
ey=2;%+0.02*t*(red+1);
ex=1;%+0.01*t*(red+1);

%elipse base
x=(-ex):0.05:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);

%elipse rotacionada e reposicionada de acordo com o centro (ex,ey)
x1=xe+x*cos(giro)+ysup.*sin(giro);
x2=xe+x*cos(-giro)+ysup.*sin(-giro);
ysup1=ye-x.*sin(giro)+ysup.*cos(giro);
yinf1=ye-x.*sin(giro)-ysup.*cos(giro);

%plotagem da figura de cada elipse modificada
plot(x1,ysup1);
hold on;
plot(x2,yinf1);
hold on;
end

%plotagem da orbita para referencia
ey=b;
ex=rf;
x=(-ex):0.5:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);
plot(x,ysup,'-k');
hold on;
plot(x,yinf,'-k');
hold on;
%plotagem dos eixos x e y para orientaçao, respectivamente
plot(-(xp+15):1:(xp+15),0,'k');
plot(0,-(xp+15):1:(xp+15),'k');
%igualando a escala dos eixos x e y
axis equal
function end
```

Programa 2 :

```
function espiral_elipsecte(v,n,raio,b)

%v=numero de voltas
%n=numero de elipses
%raio=raio inicial
%b=taxa de decaimenro do espiral
fi=0:pi/60:2*pi*v;
r=raio*exp(b*fi);
%polar(r,fi);
%hold on;
for(i=0:1:2*v*60)
    xs=r.*cos(fi);
    ys=r.*sin(fi);
end
%desenha espiral no fim

%inicia o laço de repetiçao para fazer o giro completo na orbita
for t=1:100:(100*n*v)
    w=2*pi/(100*n);
    %raio=10;
    teta=w.*t;
re=raio*exp(b*teta); %calculando espiral das elipses
%define o centro de cada elipse
    xe=re.*cos(teta);
    ye=re.*sin(teta);

%giro equivalente ao angulo percorrido da orbita
giro=-teta;
%definiçao da elipse
ey=2;
ex=1;

%elipse base
x=(-ex):0.05:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);

%elipse rotacionada e reposicionada de acordo com o centro (ex,ey)
x1=xe+x*cos(giro)+ysup.*sin(giro);
x2=xe+x*cos(-giro)+ysup.*sin(-giro);
ysup1=ye-x.*sin(giro)+ysup.*cos(giro);
yinf1=ye-x.*sin(giro)-ysup.*cos(giro);

%plotagem da figura de cada elipse modificada
plot(x1,ysup1);
hold on;
plot(x2,yinf1);
hold on;

%plotagem da orbita circular para referencia
```

```
ey=raio;
ex=raio;
x=(-ex):0.5:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);
plot(x,ysup,'-k');
hold on;
plot(x,yinf,'-k');
hold on;
%plotagem do centro da orbita (Centro de atraçao gravitacional)
plot(0,0,'+g');
hold on;
%plotagem das linhas de raio do centro da orbita ate o centro de
cada elipse
%plot(0:1:xe,0:ye/xe:ye,'r'); %lado direito, onde xe>0
hold on;
%plot(xe:1:0,ye:ye/xe:0,'r'); %lado esquerdo, onde xe<0
hold on;
%plotagem dos eixos x e y para orientaçao, respectivamente
plot(-(raio+15):1:(raio+15),0,'k');
plot(0,-(raio+15):1:(raio+15),'k');
%desenhando o espiral
plot(xs,ys,'r');

%igualando a escala dos eixos x e y
axis equal
end
```

Programa 3 :

```
function orb_hoh4(n,v,raio)

%valores recebidos
%n=8; n. de elipses em 1 volta
%v=2; n. de voltas
%raio inicial da orbita circular

a=raio; % -> semi-eixo horizontal
b=raio; % -> semi-eixo vertical

%armazena o valor de xo para manter fixo o perigeu
xp=a; %era xo;
%armazena o valor de yo para alterar a orbita
yp=b;  %era yo;

%definicao das variaveis da elipse (foco, excentricidade e
distancia)
%    a=xo;
%    b=yo;
    c=sqrt(a.^2-b.^2);
    e=c/a;
```

```
    daf=a.*(1+e);  %distancia do apogeu ao foco

%inicia o laço de repetiçao para fazer o giro completo na orbita
%************************************************
%** laço retirado, primeiro farei manualmente. **
%************************************************
%for rf=a:-20:raio-20*(v-1)
%inicio da manobra de Hohmann (percorrer a orbita circular
inicial)
    rf=a;

%definicao da orbita eliptica
    e=(daf./rf)-1;
    b=sqrt(rf.^2-(e.*rf).^2);
%************************************************
%**      laço das elipses em orbita removido       **
%** por enquanto, as elipses serao desconsideradas **
%************************************************
    for t=1:1:(n)
    w=2.*pi/(n);
    %raio=10;
    teta=w.*t;

%definicao da distancia do foco ao centro da elipse em orbita
    r=(rf.*(1-e.^2))/(1+e.*cos(teta));

%define o centro de cada elipse na orbita eliptica
    xe=e.*rf+(r.*cos(teta));%-(xp-xo);
    ye=r.*sin(teta);%-(yp-yo);

%giro equivalente ao angulo percorrido da orbita
giro=-teta;
%definiçao do crescimento da elipse
ey=2;%+0.02*t*(red+1);
ex=1;%+0.01*t*(red+1);

%elipse base
x=(-ex):0.05:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);

%elipse rotacionada e reposicionada de acordo com o centro (ex,ey)
x1=xe+x*cos(giro)+ysup.*sin(giro);
x2=xe+x*cos(-giro)+ysup.*sin(-giro);
ysup1=ye-x.*sin(giro)+ysup.*cos(giro);
yinf1=ye-x.*sin(giro)-ysup.*cos(giro);

%plotagem da figura de cada elipse modificada
plot(x1-(xp-rf),ysup1);
hold on;
plot(x2-(xp-rf),yinf1);
hold on;
end
```

```
%plotagem da orbita para referencia
ey=b;
ex=rf;
x=(-ex):0.5:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);
plot(x-(xp-rf),ysup,'-k');
hold on;
plot(x-(xp-rf),yinf,'-k');
hold on;
%plotagem do foco de cada orbita
plot(e.*rf-(xp-a),0,'+g');
hold on;
%end %laco for

%segundo passo da manobra de Hohmann (1o. tiro)
    rf=rf-20;
%definicao da orbita eliptica
    e=(daf./rf)-1;
    b=sqrt(rf.^2-(e.*rf).^2);

%plotagem das elipses na orbita eliptica de transiçao
    for t=1:1:(n/2)
    w=2.*pi/(n);
    %raio=10;
    teta=pi+w.*t;

%definicao da distancia do foco ao centro da elipse em orbita
    r=(rf.*(1-e.^2))/(1+e.*cos(teta));

%define o centro de cada elipse na orbita eliptica
    xe=e.*rf+(r.*cos(teta));%-(xp-xo);
    ye=r.*sin(teta);%-(yp-yo);

%giro equivalente ao angulo percorrido da orbita
giro=-teta;
%definiçao do crescimento da elipse
ey=2;%+0.02*t*(red+1);
ex=1;%+0.01*t*(red+1);

%elipse base
x=(-ex):0.05:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);

%elipse rotacionada e reposicionada de acordo com o centro (ex,ey)
x1=xe+x*cos(giro)+ysup.*sin(giro);
x2=xe+x*cos(-giro)+ysup.*sin(-giro);
ysup1=ye-x.*sin(giro)+ysup.*cos(giro);
yinf1=ye-x.*sin(giro)-ysup.*cos(giro);

%plotagem da figura de cada elipse modificada
```

```
plot(x1-(xp-rf),ysup1);
hold on;
plot(x2-(xp-rf),yinf1);
hold on;
end

%plotagem da orbita para referencia
ey=b;
ex=rf;
x=(-ex):0.5:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2); %somente percorre 1/2 orbita
%plot(x-(xp-rf),ysup,'r');%demonstraçao dacontinuidade da orbita
elipticanao
percorrida
hold on;
plot(x-(xp-rf),yinf,':r');
hold on;
%plotagem do foco de cada orbita
plot(e.*rf-(xp-rf),0,'+g');
hold on;

%terceiro passo da manobra de Hohmann (2o. tiro)
%retorno a orbita circular, agora com raio final
    b=rf*(1-e);
    rf=b;

%plotagem das elipses na orbita circular final
        for t=1:1:(n)
    w=2.*pi/(n);
    %raio=10;
    teta=w.*t;

%definicao da distancia do foco ao centro da elipse em orbita
    r=(rf.*(1-e.^2))/(1+e.*cos(teta));

%define o centro de cada elipse na orbita eliptica
    xe=e.*rf+(r.*cos(teta));%-(xp-xo);
    ye=r.*sin(teta);%-(yp-yo);

%giro equivalente ao angulo percorrido da orbita
giro=-teta;
%definiçao do crescimento da elipse
ey=2;%+0.02*t*(red+1);
ex=1;%+0.01*t*(red+1);

%elipse base
x=(-ex):0.05:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);

%elipse rotacionada e reposicionada de acordo com o centro (ex,ey)
x1=xe+x*cos(giro)+ysup.*sin(giro);
```

```
x2=xe+x*cos(-giro)+ysup.*sin(-giro);
ysup1=ye-x.*sin(giro)+ysup.*cos(giro);
yinf1=ye-x.*sin(giro)-ysup.*cos(giro);

%plotagem da figura de cada elipse modificada
plot(x1,ysup1);
hold on;
plot(x2,yinf1);
hold on;
end

%plotagem da orbita para referencia
ey=b;
ex=rf;
x=(-ex):0.5:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);
plot(x,ysup,'-k');
hold on;
plot(x,yinf,'-k');
hold on;
%plotagem dos eixos x e y para orientaçao, respectivamente
plot(-(xp+15):1:(xp+15),0,'k');
plot(0,-(xp+15):1:(xp+15),'k');
%igualando a escala dos eixos x e y
axis equal
espiral_elipsecte(6,10,60,-0.05)

function end
```

Programa 4 :

```
function orb_hoh2(n,v,raio)

%valores recebidos
%n=8; n. de elipses em 1 volta
%v=2; n. de voltas
%raio inicial da orbita circular

a=raio; % -> semi-eixo horizontal
b=raio; % -> semi-eixo vertical

%armazena o valor de xo para manter fixo o perigeu
xp=a; %era xo;
%armazena o valor de yo para alterar a orbita
yp=b;  %era yo;

%definicao das variaveis da elipse (foco, excentricidade e
distancia)
%    a=xo;
%    b=yo;
```

```
    c=sqrt(a.^2-b.^2);
    e=c/a;
    daf=a.*(1+e);  %distancia do apogeu ao foco

%inicia o laço de repetiçao para fazer o giro completo na orbita
%*********************************************
%** laço retirado, primeiro farei manualmente. **
%*********************************************
%for rf=a:-20:raio-20*(v-1)
%inicio da manobra de Hohmann (percorrer a orbita circular
inicial)
    rf=a;

%definicao da orbita eliptica
    e=(daf./rf)-1;
    b=sqrt(rf.^2-(e.*rf).^2);

%*************************************************
%**           laço das elipses em orbita          **
%*************************************************
    for t=1:1:(n)
    w=2.*pi/(n);
    %raio=10;
    teta=w.*t;

%definicao da distancia do foco ao centro da elipse em orbita
    r=(rf.*(1-e.^2))/(1+e.*cos(teta));

%define o centro de cada elipse na orbita eliptica
    xe=e.*rf+(r.*cos(teta));%-(xp-xo);
    ye=r.*sin(teta);%-(yp-yo);

%giro equivalente ao angulo percorrido da orbita
giro=-teta;
%definiçao do crescimento da elipse
ey=2;%+0.02*t*(red+1);
ex=1;%+0.01*t*(red+1);

%elipse base
x=(-ex):0.05:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);

%elipse rotacionada e reposicionada de acordo com o centro (ex,ey)
x1=xe+x*cos(giro)+ysup.*sin(giro);
x2=xe+x*cos(-giro)+ysup.*sin(-giro);
ysup1=ye-x.*sin(giro)+ysup.*cos(giro);
yinf1=ye-x.*sin(giro)-ysup.*cos(giro);

%plotagem da figura de cada elipse modificada
plot(x1-(raio-rf),ysup1);
hold on;
plot(x2-(raio-rf),yinf1);
```

```
hold on;
end

%plotagem da orbita para referencia
ey=b;
ex=rf;
x=(-ex):0.5:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);
plot(x,ysup,'-k');
hold on;
plot(x,yinf,'-k');
hold on;
%plotagem do foco de cada orbita
plot(e.*rf,0,'+g');
hold on;
%end %laco for

while rf > 100 %reduz orbita ate atingir 100km de distancia
%segundo passo da manobra de Hohmann (1o. tiro)
    rf=rf-20; %diminuiçao do raio da orbita. Dobro do valor
indicado, devido ao deslocamento
    daf=rf.*(1+e);  %distancia do apogeu ao foco
%definicao da orbita eliptica
    e=(daf./rf)-1;
    b=sqrt(rf.^2-(e.*rf).^2);

%plotagem das elipses na orbita eliptica de transiçao
    for t=1:1:(n/2)
    w=2.*pi/(n);
    %raio=10;
    teta=pi+w.*t;

%definicao da distancia do foco ao centro da elipse em orbita
    r=(rf.*(1-e.^2))/(1+e.*cos(teta));

%define o centro de cada elipse na orbita eliptica
    xe=e.*rf+(r.*cos(teta));
    ye=r.*sin(teta);

%giro equivalente ao angulo percorrido da orbita
giro=-teta;
%definiçao do crescimento da elipse
ey=2;%+0.02*t*(red+1);
ex=1;%+0.01*t*(red+1);

%elipse base
x=(-ex):0.05:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);

%elipse rotacionada e reposicionada de acordo com o centro (ex,ey)
x1=xe+x*cos(giro)+ysup.*sin(giro);
```

```
x2=xe+x*cos(-giro)+ysup.*sin(-giro);
ysup1=ye-x.*sin(giro)+ysup.*cos(giro);
yinf1=ye-x.*sin(giro)-ysup.*cos(giro);

%plotagem da figura de cada elipse modificada
plot(x1-(raio-rf),ysup1);
hold on;
plot(x2-(raio-rf),yinf1);
hold on;
end

%plotagem da orbita para referencia
ey=b;
ex=rf;
x=(-ex):0.5:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2); %somente percorre 1/2 orbita
%plot(x-(raio-rf),ysup,'r'); %demonstraçao da continuidade da
orbita eliptica nao percorrida
%hold on;
plot(x-(raio-rf),yinf,':r');
hold on;
%plotagem do foco de cada orbita
plot(e.*raio-(raio-rf),0,'+g');
hold on;

%terceiro passo da manobra de Hohmann (2o. tiro)
%retorno a orbita circular, agora com raio final
    b=(rf-20)*(1-e);
    rf=b;

%plotagem das elipses na orbita circular final
        for t=1:1:(n)
    w=2.*pi/(n);
    teta=w.*t;

%definicao da distancia do foco ao centro da elipse em orbita
    r=(rf.*(1-e.^2))/(1+e.*cos(teta));

%define o centro de cada elipse na orbita eliptica
    xe=e.*rf+(r.*cos(teta));%-(xp-xo);
    ye=r.*sin(teta);%-(yp-yo);

%giro equivalente ao angulo percorrido da orbita
giro=-teta;
%definiçao do crescimento da elipse
ey=2;%+0.02*t*(red+1);
ex=1;%+0.01*t*(red+1);

%elipse base
x=(-ex):0.05:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);
```

```
%elipse rotacionada e reposicionada de acordo com o centro (ex,ey)
x1=xe+x*cos(giro)+ysup.*sin(giro);
x2=xe+x*cos(-giro)+ysup.*sin(-giro);
ysup1=ye-x.*sin(giro)+ysup.*cos(giro);
yinf1=ye-x.*sin(giro)-ysup.*cos(giro);

%plotagem da figura de cada elipse modificada
plot(x1,ysup1);
hold on;
plot(x2,yinf1);
hold on;
end

%plotagem da orbita para referencia
ey=b;
ex=rf;
x=(-ex):0.5:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);
plot(x,ysup,'-k');
hold on;
plot(x,yinf,'-k');
hold on;
%define nova orbita inicial para proxima reduçao
raio=rf;
end %fim do while

%plotagem dos eixos x e y para orientaçao, respectivamente
plot(-(xp+15):1:(xp+15),0,'k');
plot(0,-(xp+15):1:(xp+15),'k');
%igualando a escala dos eixos x e y
axis equal
espiral_elipsecte(5,10,rf,-0.05)
function end
```

Programa 5 :

```
function orb_hoh22(n,raio,rfmax,redo,rede)

%valores recebidos
%n= numero de elipses em 1 volta
%raio= raio inicial da orbita circular
%rfmax= raio final maximo permitido, abaixo desse valor, inicia o
espiral
%redo= reduçao da orbita por impulso
%rede= taxa de decaimento do espiral

a=raio; % -> semi-eixo horizontal
b=raio; % -> semi-eixo vertical
```

```
%armazena o valor de xo para manter fixo o perigeu
xp=a; %era xo;
%armazena o valor de yo para alterar a orbita
yp=b;  %era yo;

%definicao das variaveis da elipse (foco, excentricidade e
distancia)
%    a=xo;
%    b=yo;
    c=sqrt(a.^2-b.^2);
    e=c/a;
    daf=a.*(1+e);  %distancia do apogeu ao foco

%inicia o laço de repetiçao para fazer o giro completo na orbita
%************************************************
%** laço retirado, primeiro farei manualmente. **
%************************************************
%for rf=a:-20:raio-20*(v-1)
%inicio da manobra de Hohmann (percorrer a orbita circular
inicial)
    rf=a;

%definicao da orbita eliptica
    e=(daf./rf)-1;
    b=sqrt(rf.^2-(e.*rf).^2);

%*************************************************
%**           laço das elipses em orbita          **
%*************************************************
    for t=1:1:(n)
    w=2.*pi/(n);
    %raio=10;
    teta=w.*t;

%definicao da distancia do foco ao centro da elipse em orbita
    r=(rf.*(1-e.^2))/(1+e.*cos(teta));

%define o centro de cada elipse na orbita eliptica
    xe=e.*rf+(r.*cos(teta));%-(xp-xo);
    ye=r.*sin(teta);%-(yp-yo);

%giro equivalente ao angulo percorrido da orbita
giro=-teta;
%definiçao do crescimento da elipse
ey=2;%+0.02*t*(red+1);
ex=1;%+0.01*t*(red+1);

%elipse base
x=(-ex):0.05:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);

%elipse rotacionada e reposicionada de acordo com o centro (ex,ey)
```

```
x1=xe+x*cos(giro)+ysup.*sin(giro);
x2=xe+x*cos(-giro)+ysup.*sin(-giro);
ysup1=ye-x.*sin(giro)+ysup.*cos(giro);
yinf1=ye-x.*sin(giro)-ysup.*cos(giro);

%plotagem da figura de cada elipse modificada
plot(x1-(raio-rf),ysup1);
hold on;
plot(x2-(raio-rf),yinf1);
hold on;
end

%plotagem da orbita para referencia
ey=b;
ex=rf;
x=(-ex):0.5:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);
plot(x,ysup,'-k');
hold on;
plot(x,yinf,'-k');
hold on;
%plotagem do foco de cada orbita
plot(e.*rf,0,'+g');
hold on;
%end %laco for

while rf > rfmax %reduz orbita ate atingir rfmax de distancia
%segundo passo da manobra de Hohmann (1o. tiro)
    rf=rf-redo/2; %diminuiçao do raio da orbita. Dobro do valor
indicado, devido ao deslocamento
    daf=rf.*(1+e);  %distancia do apogeu ao foco
%definicao da orbita eliptica
    e=(daf./rf)-1;
    b=sqrt(rf.^2-(e.*rf).^2);

%plotagem das elipses na orbita eliptica de transiçao
    for t=1:1:(n/2)
    w=2.*pi/(n);
    %raio=10;
    teta=pi+w.*t;

%definicao da distancia do foco ao centro da elipse em orbita
    r=(rf.*(1-e.^2))/(1+e.*cos(teta));

%define o centro de cada elipse na orbita eliptica
    xe=e.*rf+(r.*cos(teta));
    ye=r.*sin(teta);

%giro equivalente ao angulo percorrido da orbita
giro=-teta;
%definiçao do crescimento da elipse
ey=2;%+0.02*t*(red+1);
```

```
ex=1;%+0.01*t*(red+1);

%elipse base
x=(-ex):0.05:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);

%elipse rotacionada e reposicionada de acordo com o centro (ex,ey)
x1=xe+x*cos(giro)+ysup.*sin(giro);
x2=xe+x*cos(-giro)+ysup.*sin(-giro);
ysup1=ye-x.*sin(giro)+ysup.*cos(giro);
yinf1=ye-x.*sin(giro)-ysup.*cos(giro);

%plotagem da figura de cada elipse modificada
plot(x1-(raio-rf),ysup1);
hold on;
plot(x2-(raio-rf),yinf1);
hold on;
end

%plotagem da orbita para referencia
ey=b;
ex=rf;
x=(-ex):0.5:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2); %somente percorre 1/2 orbita
%plot(x-(raio-rf),ysup,'r'); %demonstraçao da continuidade da
orbita eliptica nao percorrida
%hold on;
plot(x-(raio-rf),yinf,':r');
hold on;
%plotagem do foco de cada orbita
plot(e.*rf,0,'+g');
hold on;

%terceiro passo da manobra de Hohmann (2o. tiro)
%retorno a orbita circular, agora com raio final
    b=(rf-redo/2)*(1-e);
    rf=b;

%plotagem das elipses na orbita circular final
        for t=1:1:(n)
    w=2.*pi/(n);
    teta=w.*t;

%definicao da distancia do foco ao centro da elipse em orbita
    r=(rf.*(1-e.^2))/(1+e.*cos(teta));

%define o centro de cada elipse na orbita eliptica
    xe=e.*rf+(r.*cos(teta));%-(xp-xo);
    ye=r.*sin(teta);%-(yp-yo);

%giro equivalente ao angulo percorrido da orbita
```

```
giro=-teta;
%definiçao do crescimento da elipse
ey=2;%+0.02*t*(red+1);
ex=1;%+0.01*t*(red+1);

%elipse base
x=(-ex):0.05:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);

%elipse rotacionada e reposicionada de acordo com o centro (ex,ey)
x1=xe+x*cos(giro)+ysup.*sin(giro);
x2=xe+x*cos(-giro)+ysup.*sin(-giro);
ysup1=ye-x.*sin(giro)+ysup.*cos(giro);
yinf1=ye-x.*sin(giro)-ysup.*cos(giro);

%plotagem da figura de cada elipse modificada
plot(x1,ysup1);
hold on;
plot(x2,yinf1);
hold on;
end

%plotagem da orbita para referencia
ey=b;
ex=rf;
x=(-ex):0.5:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);
plot(x,ysup,'-k');
hold on;
plot(x,yinf,'-k');
hold on;
%define nova orbita inicial para proxima reduçao
raio=rf;
end %fim do while

%plotagem dos eixos x e y para orientaçao, respectivamente
plot(-(xp+15):1:(xp+15),0,'k');
plot(0,-(xp+15):1:(xp+15),'k');
%igualando a escala dos eixos x e y
axis equal
espiral_elipsecte(6,10,rf,rede)
function end
```

Programa 6 :

```
function orb_raio0(ri,rf)
%valores recebidos
%ri=100 - raio inicial
%rf=70  - raio final
```

```
a=ri; % -> semi-eixo horizontal
b=ri; % -> semi-eixo vertical
%armazena o valor de xo para manter fixo o perigeu
xp=a; %era xo;
%armazena o valor de yo para alterar a orbita
yp=b;  %era yo;
%definicao das variaveis da elipse (foco, excentricidade e
distancia)
    c=sqrt(a.^2-b.^2);
    e=0;
    daf=a.*(1+e);  %distancia do apogeu ao foco
    w=a;
   %inicio da manobra de Hohmann (percorrer a orbita circular
inicial)
      e=0;
      b=sqrt(w.^2-(e.*w).^2);
%plotagem da orbita para referencia
ey=b;
ex=w;
x=(-ex):0.5:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);
plot(x-(xp-w),ysup,'-k');
hold on;
plot(x-(xp-w),yinf,'-k');
hold on;
axis equal

%segundo passo da manobra de Hohmann (1o. tiro)
   w=w-20;
   p=sqrt(((2.*rf)./(ri))/(1+(rf./ri)))-1;
   u=(p+w);
%definicao da orbita eliptica
    e=(daf./w)-1;
    b=sqrt(w.^2-(e.*w).^2);
%plotagem da orbita para referencia
ey=rf;
ex=u;
x=(-ex):0.5:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2); %somente percorre 1/2 orbita
%plot(x-(xp-w),ysup,'-r'); %demonstraçao da continuidade da orbita
eliptica nao percorrida
hold on;
plot(x-(xp-w),yinf,':r');
hold on;

%terceiro passo da manobra de Hohmann (2o. tiro)
%retorno a orbita circular, agora com raio final
    ri=w*(1-e);
    rf=ri+0.3;
    q=sqrt(ri./rf)-sqrt((2.*ri)/(ri+rf));
    s=rf+q;
```

```
%plotagem da orbita para referencia
ey=ri;
ex=s;
x=(-ex):0.5:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);
plot(x,ysup,'-k');
hold on;
plot(x,yinf,'-k');
hold on;

xlabel('Distancia (Km)')
ylabel('Raio (Km)')
title('Transferencia Inversa de Hohmann')
text(-100,5,'deltaV= -0.0925km/s')
text(50,3,'deltaV= -0.0012km/s')
text(-100,90,'R.Inicial=100km')
text(50,-40,'R.Final=70km')
%gtext('r=100')
%plotagem dos eixos x e y para orientaçao, respectivamente
plot(-(xp+15):1:(xp+15),0,'k');
plot(0,-(xp+15):1:(xp+15),'k');
%igualando a escala dos eixos x e y
axis equal
```

Apêndice B - Programas e simulações da órbita elíptica (Transferência de Breakwell) desenvolvido por Flávio Francesco Soares Schmidt em Matlab:

Programa 1:

```
 function orb_elip_ff(n,v,a,b)

%valores recebidos
%n=8; n. de elipses em 1 volta
%v=2; n. de voltas
%a -> semi-eixo horizontal
%b -> semi-eixo vertical

%armazena o valor de xo para manter fixo o perigeu
xp=a; %era xo;
%armazena o valor de yo para alterar a orbita
yp=b;  %era yo;

%definicao das variaveis da elipse (foco, excentricidade e
distancia)
%    a=xo;
%    b=yo;
    c=sqrt(a.^2-b.^2);
    e=c/a;
    daf=a.*(1+e);  %distancia do apogeu ao foco
   e1=e;

%inicia o laço de repetiçao para fazer o giro completo na orbita
for e2=e:0.06:e+6*(v-1)/100

%definicao da orbita eliptica
    a=a.*(1+e1)/(1+e2);
    b=sqrt(a.^2-(e2.*a).^2);

for t=1:1:(n)
    w=2.*pi/(n);
    %raio=10;
    teta=w.*t;

%definicao da distancia do foco ao centro da elipse em orbita
    r=(a.*(1-e2.^2))/(1+e2.*cos(teta));

%define o centro de cada elipse na orbita eliptica
xe=e2.*a+(r.*cos(teta));%-(xp-xo);
    ye=r.*sin(teta);%-(yp-yo);

%giro equivalente ao angulo percorrido da orbita
giro=-teta;
%definiçao do crescimento da elipse
ey=2;%+0.02*t*(red+1);
```

```
ex=1;%+0.01*t*(red+1);

%elipse base
x=(-ex):0.05:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);

%elipse rotacionada e reposicionada de acordo com o centro (ex,ey)
x1=xe+x*cos(giro)+ysup.*sin(giro);
x2=xe+x*cos(-giro)+ysup.*sin(-giro);
ysup1=ye-x.*sin(giro)+ysup.*cos(giro);
yinf1=ye-x.*sin(giro)-ysup.*cos(giro);

%plotagem dafigura de cada elipse modificada
plot(x1-(xp-a),ysup1);
hold on;
plot(x2-(xp-a),yinf1);
hold on;
end

%plotagem da orbita para referencia
ey=b;
ex=a;
x=(-ex):0.5:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);
plot(x-(xp-a),ysup,'-k');
hold on;
plot(x-(xp-a),yinf,'-k');
hold on;
%plotagem do foco de cada orbita
%plot(e2*a-(xp-a),0,'+g'); retirado para teste
plot(e2.*a-(xp-a),0,'+g');
hold on;
%plotagem do centro da orbita (Centro de atraçao gravitacional)
%plot(0,0,'+g');
%hold on;
%plotagem das linhas de raio do centro da orbita ate o centro de
cada elipse
%plot(0:1:xe,0:ye/xe:ye,'r'); %lado direito, onde xe>0
%hold on;
%plot(xe:1:0,ye:ye/xe:0,'r'); %lado esquerdo, onde xe<0
%hold on;

%define valor de e com ultimo valor calculado
e1=e2;
end %laco for
%plotagem dos eixos x e y para orientaçao, respectivamente
plot(-(xp+15):1:(xp+15),0,'k');
plot(0,-(xp+15):1:(xp+15),'k');
%igualando a escala dos eixos x e y
axis equal
function end
```

Programa 2:

```
function espiral_breakwell(v,n,raio,b,df)

%v=numero de voltas
%n=numero de elipses
%raio=raio inicial
%b=taxa de decaimenro do espiral
%df=distancia do foco
fi=0:pi/60:2*pi*v;
r=raio*exp(b*fi);
%polar(r,fi);
%hold on;
for(i=0:1:2*v*60)
    xs=r.*cos(fi);
    ys=r.*sin(fi);
end
%desenha espiral no fim

%inicia o laço de repetiçao para fazer o giro completo na orbita
for t=1:100:(100*n*v)
    w=2*pi/(100*n);
    %raio=10;
    teta=w.*t;
re=raio*exp(b*teta); %calculando espiral das elipses
%define o centro de cada elipse
    xe=re.*cos(teta);
    ye=re.*sin(teta);

%giro equivalente ao angulo percorrido da orbita
giro=-teta;
%definiçao da elipse
ey=2;
ex=1;

%elipse base
x=(-ex):0.05:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);

%elipse rotacionada e reposicionada de acordo com o centro (ex,ey)
x1=xe+x*cos(giro)+ysup.*sin(giro);
x2=xe+x*cos(-giro)+ysup.*sin(-giro);
ysup1=ye-x.*sin(giro)+ysup.*cos(giro);
yinf1=ye-x.*sin(giro)-ysup.*cos(giro);

%plotagem da figura de cada elipse modificada
plot(x1+df,ysup1);
hold on;
plot(x2+df,yinf1);
hold on;
```

```
%plotagem da orbita circular para referencia
%ey=raio;
%ex=raio;
%x=(-ex):0.5:(ex);
%ysup=ey.*sqrt(1-((x)/ex).^2);
%yinf=-ey.*sqrt(1-((x)/ex).^2);
%plot(x+df,ysup,'-k');
%hold on;
%plot(x+df,yinf,'-k');
%hold on;
%plotagem do centro da orbita (Centro de atraçao gravitacional)
plot(df,0,'+g');
hold on;
%plotagem dos eixos x e y para orientaçao, respectivamente
%plot(-(raio+15):1:(raio+15),0,'k');
%plot(0,-(raio+15):1:(raio+15),'k');
%desenhando o espiral
plot(xs+df,ys,'r');

%igualando a escala dos eixos x e y
axis equal
end
```

Programa 3 :

```
function breakwell_auto1(n,a,b,af)
%valores recebidos
%n=8; n. de elipses em 1 volta
%a -> semi-eixo horizontal
%b -> semi-eixo vertical
%af -> semi-eixo horizontal maximo para entrar em espiral

%armazena o valor de xo para manter fixo o perigeu
xp=a; %era xo;
%armazena o valor de yo para alterar a orbita
yp=b;  %era yo;

%definicao das variaveis da elipse (foco, excentricidade e
distancia)
%    a=xo;
%    b=yo;
    c=sqrt(a.^2-b.^2);
    e=c/a;
    daf=a.*(1+e);  %distancia do apogeu ao foco
    dpf=a.*(1-e); %distancia do perigeu ao foco
    e1=e;

%inicia o laço de repetiçao para fazer o giro completo na orbita
%for e2=e:0.06:e+6*(v-1)/100
    e2=e; %condiçao inicial
```

```
    %define cor da órbita inicial
    cor='-k';
while dpf > af  %executa a manobra ate que alcance o limite do a
definido (a final)

%definicao da orbita eliptica
    a=a.*(1+e1)/(1+e2);
    b=sqrt(a.^2-(e2.*a).^2);

for t=1:1:(n)
    w=2.*pi/(n);
    %raio=10;
    teta=w.*t;

%definicao da distancia do foco ao centro da elipse em orbita
    r=(a.*(1-e2.^2))/(1+e2.*cos(teta));

%define o centro de cada elipse na orbita eliptica
xe=e2.*a+(r.*cos(teta));%-(xp-xo);
    ye=r.*sin(teta);%-(yp-yo);

%giro equivalente ao angulo percorrido da orbita
giro=-teta;
%definiçao do crescimento da elipse
ey=2;%+0.02*t*(red+1);
ex=1;%+0.01*t*(red+1);

%elipse base
x=(-ex):0.05:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);

%elipse rotacionada e reposicionada de acordo com o centro (ex,ey)
x1=xe+x*cos(giro)+ysup.*sin(giro);
x2=xe+x*cos(-giro)+ysup.*sin(-giro);
ysup1=ye-x.*sin(giro)+ysup.*cos(giro);
yinf1=ye-x.*sin(giro)-ysup.*cos(giro);

%plotagem da figura de cada elipse modificada
plot(x1-(xp-a),ysup1);
hold on;
plot(x2-(xp-a),yinf1);
hold on;
end

%plotagem da orbita para referencia
ey=b;
ex=a;
x=(-ex):0.5:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);
plot(x-(xp-a),ysup,cor);
hold on;
```

```
plot(x-(xp-a),yinf,cor);
hold on;
%define cor das órbitas de transferência
cor='-r';
%plotagem do foco de cada orbita
plot(e2.*a-(xp-a),0,'+g');
hold on;

%define valor de e com ultimo valor calculado
e1=e2;
e2=e2+0.06; %reduz excentricidade para reduzir a orbita
dpf=a.*(1-e2); %nova distancia do perigeu ao foco
end %laco while

%definiçao e plotagem da ultima orbita antes de entrar em espiral
%definicao da orbita eliptica
    a=a.*(1+e1)/(1+e2);
    b=sqrt(a.^2-(e2.*a).^2);
for t=1:1:(n/2)
    w=2.*pi/(n);
    %raio=10;
    teta=pi+(w.*t);
%definicao da distancia do foco ao centro da elipse em orbita
    r=(a.*(1-e2.^2))/(1+e2.*cos(teta));
%define o centro de cada elipse na orbita eliptica
xe=e2.*a+(r.*cos(teta));%-(xp-xo);
    ye=r.*sin(teta);%-(yp-yo);
%giro equivalente ao angulo percorrido da orbita
giro=-teta;
%definiçao do crescimento da elipse
ey=2;%+0.02*t*(red+1);
ex=1;%+0.01*t*(red+1);
%elipse base
x=(-ex):0.05:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);
%elipse rotacionada e reposicionada de acordo com o centro (ex,ey)
x1=xe+x*cos(giro)+ysup.*sin(giro);
x2=xe+x*cos(-giro)+ysup.*sin(-giro);
ysup1=ye-x.*sin(giro)+ysup.*cos(giro);
yinf1=ye-x.*sin(giro)-ysup.*cos(giro);
%plotagem da figura de cada elipse modificada
plot(x1-(xp-a),ysup1);
hold on;
plot(x2-(xp-a),yinf1);
hold on;
end
%plotagem da orbita para referencia
ey=b;
ex=a;
x=(-ex):0.5:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);
```

```
%plot(x-(xp-a),ysup,'-k');
%hold on;
plot(x-(xp-a),yinf,'-r');
hold on;
%plotagem do foco de cada orbita
plot(e2.*a-(xp-a),0,'+g');
hold on;
%fim da ultima orbita

%plotagem dos eixos x e y para orientaçao, respectivamente
plot(-(xp+15):1:(xp+15),0,'k');
plot(0,-(xp+15):1:(xp+15),'k');
%igualando a escala dos eixos x e y
axis equal
%dpf=a.*(1-e1); retirado pois alterou o ultimo para orbita < af
c=sqrt(xp.^2-yp.^2); %foco inicial - ultimo termo da funçao
espiral, substituido para teste
espiral_breakwell(6,10,dpf,-0.05,e2.*a-(xp-a))

axis([-200 200 -150 150])
function end
```

Programa 4 :

```
function breakwell1(n,a,b,af)
%valores recebidos
%n=8; n. de elipses em 1 volta
%a  -> semi-eixo horizontal
%b  -> semi-eixo vertical
%af -> semi-eixo horizontal maximo para entrar em espiral
%armazena o valor de xo para manter fixo o perigeu
xp=a;  %era xo;
%armazena o valor de yo para alterar a orbita
yp=b;  %era yo;
%definicao das variaveis da elipse (foco, excentricidade e
distancia)
    c=sqrt(a.^2-b.^2);
    e=c/a;
    daf=a.*(1+e);  %distancia do apogeu ao foco
    dpf=a.*(1-e);  %distancia do perigeu ao foco
    e1=e;
%inicia o laço de repetiçao para fazer o giro completo na orbita
    e2=e; %condiçao inicial
          %define cor da órbita inicial
    cor='-k';
while dpf > af  %executa a manobra ate que alcance o limite do a
definido (a final)
%definicao da orbita eliptica
    a=a.*(1+e1)/(1+e2);
    b=sqrt(a.^2-(e2.*a).^2);
for t=1:1:(n)
```

```
    w=2.*pi/(n);
    teta=w.*t;
%definicao da distancia do foco ao centro da elipse em orbita
    r=(a.*(1-e2.^2))/(1+e2.*cos(teta));
%define o centro de cada elipse na orbita eliptica
xe=e2.*a+(r.*cos(teta));%-(xp-xo);
    ye=r.*sin(teta);%-(yp-yo);
%giro equivalente ao angulo percorrido da orbita
giro=-teta;
%definiçao do crescimento da elipse
ey=2;%+0.02*t*(red+1);
ex=1;%+0.01*t*(red+1);
%elipse base
x=(-ex):0.05:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);
%elipse rotacionada e reposicionada de acordo com o centro (ex,ey)
x1=xe+x*cos(giro)+ysup.*sin(giro);
x2=xe+x*cos(-giro)+ysup.*sin(-giro);
ysup1=ye-x.*sin(giro)+ysup.*cos(giro);
yinf1=ye-x.*sin(giro)-ysup.*cos(giro);
%plotagem da figura de cada elipse modificada
plot(x1-(xp-a),ysup1);
hold on;
plot(x2-(xp-a),yinf1);
hold on;
end
%plotagem da orbita para referencia
ey=b;
ex=a;
x=(-ex):0.5:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);
plot(x-(xp-a),ysup,cor);
hold on;
plot(x-(xp-a),yinf,cor);
hold on;
%define cor das órbitas de transferência
cor='-r';
%plotagem do foco de cada orbita
plot(e2.*a-(xp-a),0,'+g');
hold on;
%define valor de e com ultimo valor calculado
e1=e2;
e2=e2+0.06;    %reduz excentricidade para reduzir a orbita
dpf=a.*(1-e2); %nova distancia do perigeu ao foco
end %laco while

%definiçao e plotagem da ultima orbita antes de entrar em espiral
%definicao da orbita eliptica
    a=a.*(1+e1)/(1+e2);
    b=sqrt(a.^2-(e2.*a).^2);
for t=1:1:(n/2)
```

```
    w=2.*pi/(n);
    teta=pi+(w.*t);
%definicao da distancia do foco ao centro da elipse em orbita
    r=(a.*(1-e2.^2))/(1+e2.*cos(teta));
%define o centro de cada elipse na orbita eliptica
xe=e2.*a+(r.*cos(teta));%-(xp-xo);
    ye=r.*sin(teta);%-(yp-yo);
%giro equivalente ao angulo percorrido da orbita
giro=-teta;
%definiçao do crescimento da elipse
ey=2;%+0.02*t*(red+1);
ex=1;%+0.01*t*(red+1);
%elipse base
x=(-ex):0.05:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);
%elipse rotacionada e reposicionada de acordo com o centro (ex,ey)
x1=xe+x*cos(giro)+ysup.*sin(giro);
x2=xe+x*cos(-giro)+ysup.*sin(-giro);
ysup1=ye-x.*sin(giro)+ysup.*cos(giro);
yinf1=ye-x.*sin(giro)-ysup.*cos(giro);
%plotagem da figura de cada elipse modificada
plot(x1-(xp-a),ysup1);
hold on;
plot(x2-(xp-a),yinf1);
hold on;
end
%plotagem da orbita para referencia
ey=b;
ex=a;
x=(-ex):0.5:(ex);
ysup=ey.*sqrt(1-((x)/ex).^2);
yinf=-ey.*sqrt(1-((x)/ex).^2);
%plot(x-(xp-a),ysup,'-k');
%hold on;
plot(x-(xp-a),yinf,'-r');
hold on;
%plotagem do foco de cada orbita
plot(e2.*a-(xp-a),0,'+g');
hold on;
%fim da ultima orbita
%plotagem dos eixos x e y para orientaçao, respectivamente
plot(-(xp+15):1:(xp+15),0,'k');
plot(0,-(xp+15):1:(xp+15),'k');
%igualando a escala dos eixos x e y
axis equal
c=sqrt(xp.^2-yp.^2); %foco inicial -ultimo termo da funçao
espiral, substituindo para teste
espiral_breakwell(6,10,dpf,-0.05,e2.*a-(xp-a))
axis equal

xlabel('Distancia (Km)')
ylabel('Raio (Km)')
```

```
title('Transferencia Inversa de Breakwell')
text(2,103,'b=100km')
text(-145,-5,'a=150km')
text(-48,-92,'e1=0.7454')
text(-99,-70,'e2=0.8054')
text(2,-60,'e3=0.8654')
function end
```

Apêndice C – Programas e simulações com força de arrasto atmosférico desenvolvido por Rolf Henry Vargas Valdivia em Matlab:

Programa 1:

```
%transferencia com arrasto atmosferico
clear all
close all

opt = odeset('AbsTol',0.00001,'RelTol', 0.0000001);

x0 = [6378;12756 ; 0.5; 1.5];    % espaço xo inicial
mu = 398600;                     % a unidade de espaço e o km
r  = [x0(1); x0(2)];               %x(1)= x = posiçao  , x(4)=y =
posicao
mr = sqrt(x0(1)^2 + x0(2)^2);      % modulo do vetor posiçao
v  = [x0(3); x0(4)];               %x(3)= xponto=velocidade  ,
x(4)=yponto=velocidade
mv = sqrt(x0(3)^2+x0(4)^2);        % modulo do vetor velocidade
a  = mr*mu/(2*mu-mr*mv^2);       % semi-eixo maior 'a'

periodo = 2*pi*sqrt((a^3)/398600);
tspan   = [0:0.5:2.*periodo];

 [t,x]  = ode45('arrasto1',tspan,x0,opt);

 [t2,x2]  = ode23s('arrasto2',tspan,x0);

 figure(1)
 plot(x(:,1),x(:,2));
 title('Grafico da orbita com densidade cte (h=700 Km)')
 zoom on

 figure(2)
 plot(x2(:,1),x2(:,2));
 title('Grafico da orbita com densidade variavel')
 zoom on

 figure(3)
 plot(x(:,1),x(:,2),'b',x2(:,1),x2(:,2),'r--');
 title('Comparacao das duas Orbitas');
 legend('Densi Cte','Densi Vari')
 zoom on
```

Programa 1.1 :

```
%arrasto1
function dx = f(t,x);
```

```
mu    = 398600; % a unidade de espaço e o km(parametro gravt da
terra)
r     = sqrt(x(1)^2 + x(2)^2); % modulo do vetor posiçao
%v     = [x(3); x(4)];          %x(3)= xponto=velocidade  ,
x(4)=yponto=velocidade
mv    = sqrt(x(3)^2+x(4)^2);   % modulo do vetor velocidade
%rt    = 6378;                  % raio da terra
raio  = 2;                     % raio do satelite
h     = 700;                   %  altura  %h= ra-rt;  ra= h+rt;
cd    = 0.5;                   % coeficiente de arrasto
area  = (pi.*(raio^2));        % area do satelite
densi =3.614*10^-14;           % densidade correspondente a altura
700


f = 0.5.*cd.*area.*densi.*mv^2; % equaçao da força de arrasto
atmosférico

dx    =[x(3);
        x(4);
        ((-mu/r^3).*x(1))-(f.*x(3)/mv);
        ((-mu/r^3).*x(2))-(f.*x(4)/mv)];      %   vetor coluna
```

Programa 1.2 :

```
%Arrasto2
function dx = f(t,x);

mu    = 398600; % a unidade de espaço e o km(parametro gravt da
terra)
r     = sqrt(x(1)^2 + x(2)^2); % modulo do vetor posiçao
%v     = [x(3); x(4)];          %x(3)= xponto=velocidade  ,
x(4)=yponto=velocidade
mv    = sqrt(x(3)^2+x(4)^2);   % modulo do vetor velocidade
%rt    = 6378;                  % raio da terra
raio  = 2;                     % raio do satelite
cd    = 0.5;                   % coeficiente de arrasto
area  = (pi.*(raio^2));        % area do satelite
h     = r-6378;

if h > 1000
    densi = 3.019*10^-15;
end
if h >900
    if h<=1000
        densi = 5.245*10^-15;
    end
end
if h >800
    if h<=900
        densi = 1.17*10^-14;
```

```
    end
end
if h >700
    if h<=800
        densi = 3.614*10^-14;
    end
end
if h >600
    if h<=700
        densi = 1.454*10^-13;
    end
end
if h >500
    if h<=600
        densi = 6.967*10^-13;
    end
end
if h >400
    if h<=500
        densi = 3.725*10^-12;
    end
end
if h >300
    if h<=400
        densi = 2.418*10^-11;
    end
end
if h >200
    if h<=300
        densi = 2.789*10^-10;
    end
end
if h >180
    if h<=200
        densi = 5.464*10^-10;
    end
end
if h >160
    if h<=180
        densi = 1.244*10^-9;
    end
end
if h >150
    if h<=160
        densi = 2.070*10^-9;
    end
end
if h >140
    if h<=150
        densi = 3.845*10^-9;
    end
end
if h >130
```

```
    if h<=140
        densi = 8.484*10^-9;
    end
end
if h >120
    if h<=130
        densi = 2.438*10^-8;
    end
end
if h >110
    if h<=120
        densi = 9.661*10^-8;
    end
end
if h >100
    if h<=110
        densi = 5.297*10^-7;
    end
end
if h<=100
    densi = 1.343*10^-6;
end

f = 0.5.*cd.*area.*densi.*mv^2; % equaçao da força de arrasto
atmosférico

dx    =[x(3);
        x(4);
        ((-mu/r^3).*x(1))-(f.*x(3)/mv);
        ((-mu/r^3).*x(2))-(f.*x(4)/mv)];      %   vetor coluna
```

Programa 2 :

```
%arrast

opt = odeset('AbsTol',0.00001,'RelTol', 0.0000001);

x  = [6378;12756 ; 0.5; 1.5];    % espaço xo inicial
mu = 398600;                     % a unidade de espaço e o km
r  = [x(1); x(2)];               %x(1)= x = posiçao  , x(4)=y =
posicao
mr = sqrt(x(1)^2 + x(2)^2);      % modulo do vetor posiçao
v  = [x(3); x(4)];               %x(3)= xponto=velocidade  ,
x(4)=yponto=velocidade
mv = sqrt(x(3)^2+x(4)^2);        % modulo do vetor velocidade
a  = mr*mu/(2*mu-mr*mv^2);       % semi-eixo maior 'a'

periodo = 2*pi*sqrt((a^3)/398600);
tspan   = [0:3.*periodo];

 [t,x]  = ode45('forma',tspan,x,opt);
```

```
 plot(x(:,1),x(:,2))
zoom on
```

Programa 2.1 :

```
%forma
function dx = f(t,x);

mu    = 398600;                % a unidade de espaço e o
km(parametro gravt da terra)
r     = sqrt(x(1)^2 + x(2)^2); % modulo do vetor posiçao
v     = [x(3); x(4)];          %x(3)= xponto=velocidade  ,
x(4)=yponto=velocidade
mv    = sqrt(x(3)^2+x(4)^2);   % modulo do vetor velocidade
rt    = 6378;                  % raio da terra
raio  = 2;                     % raio do satelite
%h    = 700;                   %  altura  %h= ra-rt;  ra= h+rt;
cd    = 0.5;                   % coeficiente de arrasto
area  = (pi.*(raio^2));        % area do satelite

%h=[0;
25;30;35;40;45;50;55;60;65;70;75;80;85;90;95;100;110;120;130;140;
 %
150;160;180;200;250;300;350;400;450;500;600;700;800;900;1000];%
altura

%D=[8.44;6.49;6.75;7.07;7.47;7.83;7.95;7.73;7.29;6.81;6.33;6.00;5.
70;5.41;5.38;5.74;6.15;
 %
8.06;11.6;16.1;20.6;24.6;26.3;33.2;38.5;46.9;52.5;56.4;59.4;62.2;6
5.8;79;109;164;225;
 %  268]; % escala de altitude

densi =exp(-100/6.15);
f = 0.5.*cd.*area.*densi.*mv^2; % equaçao da força de arrasto
atmosférico

dx =    [x(3);
        x(4);
        ((-mu/r^3).*x(1))-(f.*x(3)/mv);
        ((-mu/r^3).*x(2))-(f.*x(4)/mv)];      %   vetor coluna
```

Apêndice D - Estudo sobre a Integração Numérica de Equações Diferenciais Ordinárias

D.1-Introdução

Este estudo foi baseado no capítulo 6 do livro Applied Numerical Methods pelos autores Carnaham B. , Luther H. A. e Wilkes J. O, 1969. Para um melhor entendimento dos métodos de integração numérica para a solução de equações diferenciais ordinárias, estudamos o 1º método de Euler e os métodos de Runge-Kutta de terceira e quarta ordem.

D.2- 1° Método de Euler

Este é um método simples, responsável pela análise da propagação de erro, o passo para o método de Euler pode ser discutido em alguns detalhes, mesmo que precisão limitada impede seu uso para problemas práticos.

$$y_1 = y(x_0) + hf(x_0, y(x_0))$$

$$y_{i+1} = y_i + hf(x_i, y_i) = y_i + hf_i \qquad i \geq 1$$

A solução através do intervalo [x0 ,x1] é assumido seguindo a linha tangente de y(x) em x0. Quando o 1º método de Euler é aplicado repetidamente através de vários intervalo em seqüência, a solução numérica desenha sobre a curva, um segmento com declive fi, i=0,1,2,...,n-1.
A figura C.1 mostra os conceitos apresentados pelo método:

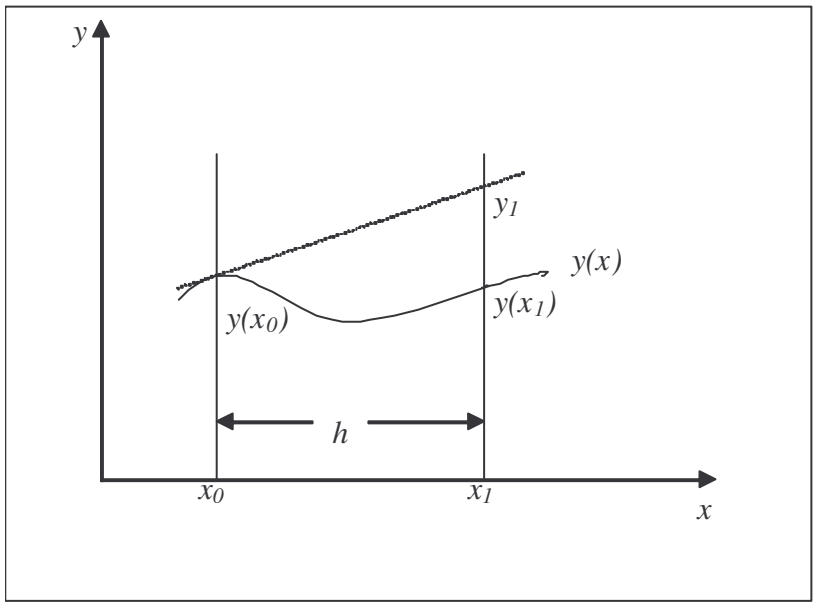

Figura D.1 – 1° Método de Euler

Temos como exemplo, a equação diferencial :

$$\frac{dy}{dx} = f(x, y) = x + y \qquad (1)$$

Para a condição inicial x0= 0, y(x0) = y0 = 0. Temos:

$$y = e^{x} - x - 1 \qquad (2)$$

A solução pelo 1º Método de Euler é mostrada na Tabela D.1, usando o passo de h=0.1 e o limite de integração x10= 1.0 .Utilizando a seguintes equações:

$$y_1 = y(x_0) + hf(x_0, y(x_0)) \qquad (3)$$

$$y_{i+1} = y_i + hf(x_i, y_i) = y_i + hf_i \qquad i \geq 1$$

De acordo com a tabela :

- A coluna 1 mostra o valor de i .
- A coluna 2 mostra o valor de xi .
- A coluna 3 mostra o valor de yi calculado pelas equações (3).
- A coluna 4 mostra o cálculo com equação (1) da derivada f(xi,yi)
- A coluna 5 mostra a verdadeira solução analítica arredondada de y(xi) para quatro algarismos, usando a equação y =ex - x - 1.
- A coluna 6 mostra o erro global calculado por Ei = yi - y(xi) .
- A coluna 7 mostra o truncamento de yi para quatro algarismos.
- A coluna 8 mostra o arredondamento de yi para quatro algarismos.
- A coluna 9 mostra o valor calculado pela equação (3) usando y(xi -1), com o valor verdadeiro y(xi).
- A coluna 10 mostra o erro de truncamento calculado por Ei = y(xi –1) - y(xi), com o valor verdadeiro y(xi) .
- A coluna 11 mostra o valor do erro de truncamento local calculado por:

$$|e_t|_{\max} = \frac{h^2}{2!} e^{x_i + 1} \qquad \text{Sendo } x = x_{i+1} \qquad (4)$$

Tabela D.1 – 1o Método de Euler

$y(0)=0$						$h=0.1$				
i (1)	x_i (2)	y_i (3)	$f(x_i,y_i)$ (4)	$y(x_i)$ (5)	$E_i = y_i - y(x_i)$ (6)	y_i (7)	y_i (8)	$y(x_i-1)$ (9)	$y(x_i-1)$ (10)	e^x_{max} (11)
0	0.0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	-	-
1	0.1	0.0000	0.1000	0.0052	-0.0052	0.0000	0.0000	0.0000	-0.0052	0.0055
2	0.2	0.0100	0.2100	0.0214	-0.0114	0.0100	0.0100	0.0157	-0.0057	0.0061
3	0.3	0.0310	0.3310	0.0499	-0.0189	0.0310	0.0310	0.0435	-0.0064	0.0067
4	0.4	0.0641	0.4641	0.0918	-0.0277	0.0641	0.0641	0.0849	-0.0069	0.0075
5	0.5	0.11051	0.61051	0.1487	-0.0382	0.1105	0.1105	0.1410	-0.0077	0.0082
6	0.6	0.171561	0.771561	0.2221	-0.0505	0.1715	0.1715	0.2136	-0.0085	0.0091
7	0.7	0.2487171	0.9487171	0.3138	-0.0651	0.2486	0.2488	0.3043	-0.0095	0.0101
8	0.8	0.34358881	1.14358881	0.4255	-0.0819	0.3434	0.3437	0.4152	-0.0103	0.0111
9	0.9	0.457947691	1.357947691	0.5596	-0.1017	0.4577	0.4581	0.5480	-0.0116	0.0123
10	1.0	0.5937424601	1.5937424601	0.7183	-0.1246	0.5934	0.5939	0.7056	-0.0127	0.0136

Na Tabela D.2 foram realizados os mesmos cálculos da tabela anterior, desenvolvidos para um melhor entendimento e acompanhamento do 1º Método de Euler usando o Microsoft Excel versão 97.

Tabela D.2 - 1º Método de Euler calculado pelo Excel

$y(0)=0$						$h=0.1$				
i (1)	x_i (2)	y_i (3)	$f(x_i,y_i)$ (4)	$y(x_i)$ (5)	$E_i = y_i - y(x_i)$ (6)	y_i (7)	y_i (8)	$y(x_i-1)$ (9)	$y(x_i-1)$ (10)	e^x_{max} (11)
0	0	0	0	0	0	0	0	0	-	0.005
1	0.1	0	0.1	0.0052	-0.0052	0	0	0	-0.0052	0.0055
2	0.2	0.01	0.21	0.0214	-0.0114	0.01	0.01	0.0157	-0.0057	0.0061
3	0.3	0.031	0.331	0.0499	-0.0189	0.031	0.031	0.0435	-0.0064	0.0067
4	0.4	0.0641	0.4641	0.0918	-0.0277	0.0641	0.0641	0.0849	-0.0069	0.0075
5	0.5	0.11051	0.61051	0.1487	-0.0382	0.1105	0.1105	0.141	-0.0077	0.0082
6	0.6	0.171561	0.771561	0.2221	-0.0505	0.1715	0.1715	0.2136	-0.0084	0.0091
7	0.7	0.2487171	0.9487171	0.3138	-0.0651	0.2486	0.2488	0.3043	-0.0095	0.0101
8	0.8	0.34358881	1.14358881	0.4255	-0.0819	0.3434	0.3437	0.4152	-0.0103	0.0111
9	0.9	0.457947691	1.357947691	0.5596	-0.1017	0.4577	0.4581	0.5481	-0.0115	0.0123
10	1.0	0.59374246	1.59374246	0.7183	-0.1246	0.5934	0.5939	0.7056	-0.0127	0.0136

D.3 - Método de Runge-Kutta

É possível desenvolver um procedimento de um passo que envolve somente cálculos da derivada de primeira-ordem. Estes algoritmos são chamado de Runge-Kutta. As aproximações de segunda, terceira e quarta ordens (isso é, a aproximação com precisão equivalente à Expansão de Taylor de y(x) mantendo o termo em h2, h3 e h4, respectivamente), requer a estimação de f(x,y) em dois, três e quatro valores, respectivamente, de x no intervalo $xi \leq x \leq xi+1$.

Todo Método de Runge-Kutta tem algoritmo de forma:

$$y_{i+1} = y_i + h\phi(x_i, y_i, h) \qquad (5)$$

sendo ∅ o incremento da função para aproximação de f(x,y) sobre intervalo xi ≤ x ≤ xi+1.

Runge-Kutta de 3ª ordem:

Sendo K1, K2 e K3 aproximações da derivada em diversos pontos sobre o intervalo de integração [xi, xi+1].

$$\begin{aligned} y_{i+1} &= y_i + \frac{h}{6}(k_1 + 4k_2 + k_3) \\ k_1 &= f(x_1, y_i) \\ k_2 &= f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_1\right) \\ k_3 &= f(x_i + h, y_i + 2hk_2 - hk_1) \end{aligned} \qquad (6)$$

Runge-Kutta de 4ª ordem:

Sendo K1, K2 ,K3 e K4 aproximações da derivada em diversos pontos sobre o intervalo de integração

[xi, xi+1].

$$\begin{aligned} y_{i+1} &= y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ k_1 &= f(x_1, y_i) \\ k_2 &= f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_1\right) \\ k_3 &= f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_2\right) \\ k_4 &= f(x_i + h, y_i + hk_3) \end{aligned} \qquad (7)$$

Runge-Kutta de 4a ordem:

Sendo K1, K2 ,K3 e K4 aproximações da derivada em diversos pontos sobre o intervalo de integração [xi, xi+1].

$$\begin{aligned} y_{i+1} &= y_i + \frac{h}{8}(k_1 + 3k_2 + 3k_3 + k_4) \\ k_1 &= f(x_1, y_i) \\ k_2 &= f\left(x_i + \frac{1}{3}h, y_i + \frac{1}{3}hk_1\right) \\ k_3 &= f\left(x_i + \frac{2}{3}h, y_i - \frac{1}{3}hk_1 + hk_2\right) \\ k_4 &= f(x_i + h, y_i + hk_1 - hk_2 + hk_3) \end{aligned} \qquad (8)$$

Para um melhor desenvolvimento e entendimento do Método, na Tabela D.3 estão contidos todos os cálculos utilizando o Método Runge-Kutta usando o Microsoft Excel versão 97, considerando a equação diferencial:

$$\frac{dy}{dx} = f(x, y) = x + y$$

De acordo com a Tabela D.3:

- A coluna 1 mostra o valor de i .
- A coluna 2 mostra o valor de xi .
- A coluna 3 mostra o valor de yi calculado pelas equações(6) do Runge-Kutta de 3ª ordem .
- A coluna 4 mostra o yi de 3ª ordem arredondado para quatro algarismo, com equação(6) .
- A coluna 5 mostra o valor de yi calculado pelas equações(7) do Runge-Kutta de 4ª ordem .
- A coluna 6 mostra o yi de 4ª ordem arredondado para quatro algarismo, com equação(7).
- A coluna 7 mostra o valor de yi calculado pelo Runge-Kutta de 4ª ordem, com a equação (8).
- A coluna 8 mostra o yi de 4ª ordem arredondado para quatro algarismos, com a equação (8).
- A coluna 9 mostra o yi analítico.
- A coluna 10 mostra o yi analítico arredondado.

Tabela D.3 - Método de Runge-Kutta calculado pelo Excel

		$y(0)=0$						$h=0.1$	
i (1)	x_i (2)	y_i *3ª ordem (3)*	y_i *arred (4)*	y_i *4ª ordem (5)*	y_i *arred (6)*	y_i *4ª ordem (7)*	y_i *arred (8*	y_i *Analítico (9)*	y_i *Analítico arred (10)*
0	0	0	0	0	0	0	0	0	0
1	0.1	0,005166667	0,0052	0,005170833	0,0052	0,005170833	0,0052	0,005170918	0,0052
2	0.2	0,021393361	0,0214	0,021402571	0,0214	0,021402571	0,0214	0,021402758	0,0214
3	0.3	0,04984323	0,0498	0,049858497	0,0499	0,049858497	0,0499	0,049858808	0,0499
4	0.4	0,091801743	0,0918	0,09182424	0,0918	0,09182424	0,0918	0,091824698	0,0918
5	0.5	0,148689559	0,1487	0,148720639	0,1487	0,148720639	0,1487	0,148721271	0,1487
6	0.6	0,222076744	0,2221	0,222117962	0,2221	0,222117962	0,2221	0,2221188	0,2221
7	0.7	0,313698482	0,3137	0,313751627	0,3138	0,313751627	0,3138	0,313752707	0,3138
8	0.8	0,425472439	0,4255	0,425539563	0,4255	0,425539563	0,4255	0,425540928	0,4255
9	0.9	0,559517957	0,5595	0,559601414	0,5596	0,559601414	0,5596	0,559603111	0,5596
10	1.0	0,718177262	0,7182	0,718279744	0,7183	0,718279744	0,7183	0,718281828	0,7183

D.4-Conclusão

No Método de Runge-Kutta o erro está a partir da quarta casa decimal, no Método de Euler o erro está na primeira casa decimal, para o mesmo número de iterações. Conclui-se então que o método de Runge-Kutta tem uma melhor precisão em seus resultados.

Apêndice E - Estudo sobre MATLAB

E.1 - Introdução

Trabalhamos com o programa Matlab usando a apostila da versão 5.1 do MATLAB Introdução à Solução de Problemas de Engenharia, para simular e entender melhor os programas de transferência inversa de Hohmann e transferência inversa de Breakwell. Foi realizado um estudo na definição de matrizes, cálculos fundamentais e gráficos.

E.2 - Definição de Matrizes no MATLAB: Quando definimos uma matriz, os valores das linhas podem estar separados por vírgulas ou por espaços. Assim, as matrizes A, B, C e D usando o MATLAB serão representadas por:

```
A = [2,6,8,1];
B = [8,10,1,3];
C=[1,2,2;4,1,2;5,6,8];
D=[3,0,0;2,4,5;6,2,1];
```

E.3 - Cálculos Fundamentais: As operações de adição, subtração, multiplicação e divisão são a maioria das operações fundamentais. Estas operações podem ser executadas sobre um valor simples (um escalar), aplicadas a uma lista de valores (vetor), ou aplicadas a um grupo de valores armazenados em uma matriz.

Adição e Subtração de Matrizes:

```
» A=[2,6,8,1];
» B=[8,10,1,3];
A+B

ans =

   10    16     9     4

» A-B

ans =

   -6    -4     7    -2
```

Multiplicação de Matrizes:

```
» A=[2,6,8,1];
» B=[8,10,1,3];
» A.*B

ans =

   16    60     8     3
```

Divisão de Matrizes: O MATLAB tem dois operadores de divisão, o comando para divisão direita:

```
» A=[2,6,8,1];
» B=[8,10,1,3];
» A./B
ans =

0.2500    0.6000    8.0000    0.3333
```

O comando para divisão *esquerda*:

```
A.\B

ans =
   4.0000    1.6667    0.1250    3.0000
```

Determinante de Matrizes:

```
» C=[1,2,2;4,1,2;5,6,8];
» D=[3,0,0;2,4,5;6,2,1];
» det(C)

ans =

   -10

» det(C+D)

ans =

   -12

» det(C.*D)

ans =

  -264
```

E.4 - Gráficos X-Y : Usando o MATLAB para plotar gráficos, gerando um simples gráfico x-y de dados armazenados em dois vetores.Suponha que queira plotar os dados de temperatura a seguir coletados em uma experiência de física:

Tempo, s	Temperatura, °C
0	54.2
1	58.5
2	63.8
3	64.2
4	67.3
5	71.5
6	88.5
7	90.1
8	90.6
9	89.5
10	90.4

Para plotar estes pontos, simplesmente usamos o comando *plot* , onde *x e y* são vetores-linha ou vetores-coluna.

```
plot (x, y)

A =

         0   54.2000
    1.0000   58.5000
    2.0000   63.8000
    3.0000   64.2000
    4.0000   67.3000
    5.0000   71.5000
    6.0000   88.5000
    7.0000   90.1000
    8.0000   90.6000
    9.0000   89.5000
   10.0000   90.4000

» x=A(:,1)

x =

     0
     1
     2
     3
     4
     5
     6
     7
     8
     9
    10

» y=A(:,2)

y =

   54.2000
   58.5000
   63.8000
   64.2000
   67.3000
   71.5000
   88.5000
   90.1000
   90.6000
   89.5000
   90.4000

plot(x,y)
```

```
» title('Laboratório de Física-Experiência1')
» xlabel('Tempo,s')
» ylabel('Temperatura, graus Celsius')
» grid
```

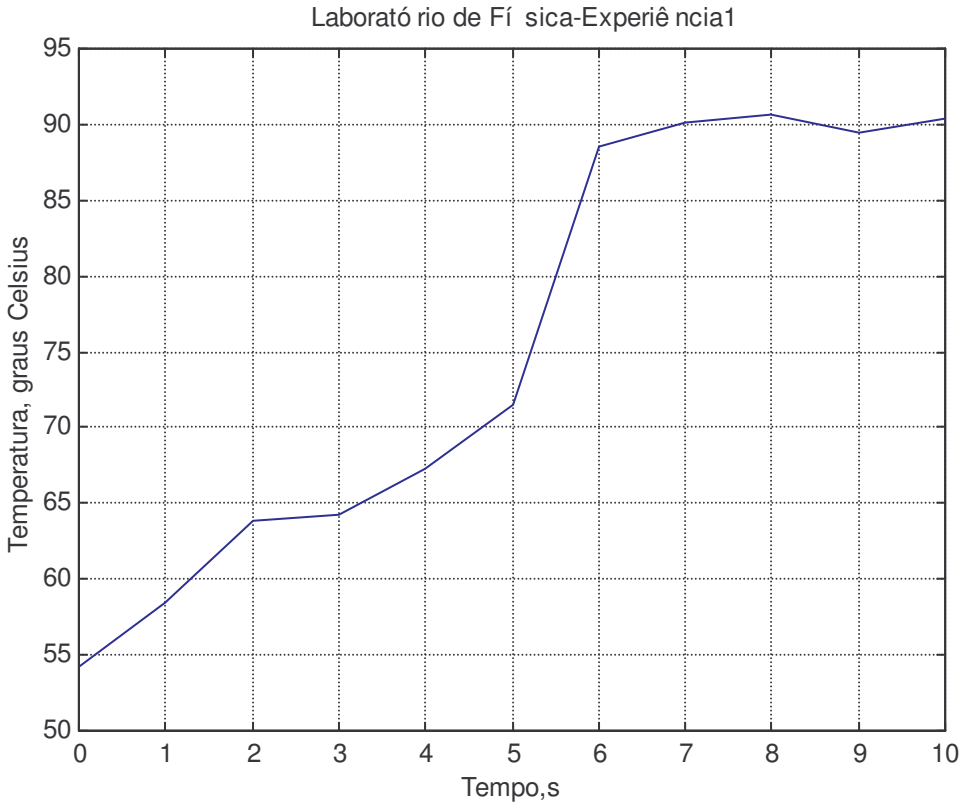

Figura E.1 – Gráfico X-Y

E.5 - Equações Diferenciais Ordinárias

A equação diferencial (ODE) é uma equação que pode ser escrita na seguinte forma:

$$y\ ' = \frac{dy}{dx} = f(x, y)$$

onde x é a variável independente.

A equação diferencial descreve como a taxa de variação das variáveis de um sistema são influenciadas por variáveis do sistema e por estímulos, ou seja, por entradas.

As técnicas numéricas mais comuns para resolver equações diferenciais ordinárias, são o método de Euler e o método de Runge-Kutta. Tanto o método de Euler quanto o método de Runge-Kutta aproximam a função utilizando-se da expansão em série de Taylor.

E.5.1 - Método de Runge - Kutta

Os métodos mais populares para a integração da equação diferencial de primeira ordem são os métodos de Runge - Kutta. Esses métodos de aproximação de uma função se usam da expansão por série de Taylor.

- Comando ode

O Matlab contém dois comandos para calcular soluções numéricas para equações diferenciais ordinárias: ode23 e ode45; o comando ode23 usa o método de Runge - Kutta para equações diferenciais de segunda e terceira ordem; o comando ode45 usa o método de Runge - Kutta para equações diferenciais de quarta e quinta ordem. Os comandos ode23 e ode45 possuem os mesmos tipos de argumentos.

A função ode45 é sempre a primeira candidata a ser usada na solução de um problema. Isso requer que escrevamos um arquivo M de função que retorna as derivadas, dados o tempo atual e os valores atuais do y_1 e y_2 .

Exemplo 1: Considerando a seguinte equação diferencial:

$$\frac{dy}{dx} = x + y$$

seja: $y_1 = x$ $y_2 = y$

então: $\frac{dy_1}{dx} = 1$ $\frac{dy_2}{dx} = x + y$

O arquivo M de função é mostrado a seguir:

```
function yponto=kutta(t,y)
%equação dy/dx=x+y
% ('= d/dx, ''=d^2/dx^2)
% façamos y(1)=x  e y(2)=y
%
% então   y(1)'=1
%         y(2)'= y(1)+y(2)

yponto=[1;y(1)+y(2)];    %   vetor coluna
```

Depois na tela do Matlab é dado o tempo e condições iniciais, a solução é calculada:

```
tspan=[0 1];
 y0=[0;0];
 [t,y]=ode45('kutta',tspan,y0)
 plot(t,y)
```

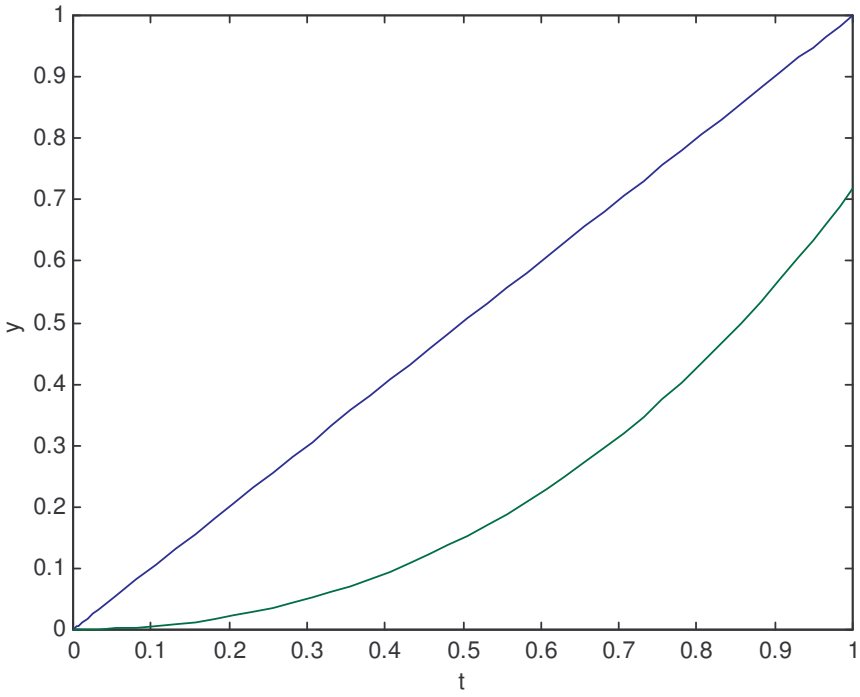

Figura E.2 – Resolução da Equação Diferencial

Exemplo 2: Vamos considerar a clássica equação diferencial de Van der Pol, que descreve um oscilador:

$$\frac{d^2x}{dt^2} - \mu\left(1 - x^2\right)\frac{dx}{dt} + x = 0$$

Para a resolução de equações diferenciais de ordem superior devem ser reescritas em termos de um conjunto equivalente de equações diferenciais de primeira ordem. Isso é obtido pela definição de duas novas variáveis:

$$y_1 = x \qquad\qquad y_2 = \frac{dx}{dt}$$

então:

$$\frac{dy_1}{dt} = y_2 \qquad\qquad \frac{dy_2}{dt} = \mu\left(1 - {y_1}^2\right) - y_1$$

Dessa forma, y_1 e y_2 são escritos como um vetor coluna y. Logo o arquivo M de função resultante é:

```
function yponto=vonder(t,y)
%equação van der Pol
% ('= d/dx, ''=d^2/dx^2)
% façamos y(1)=x  e y(2)=x'
%
% então   y(1)'=y(2)
%         y(2)'=mu*(1-y(1)^2)*y(2)-y(1)

mu=2;
yponto=[y(2);mu*(1-(y(1)).^2)*y(2)-y(1)]; % vetor coluna
```

Depois na tela do Matlab é dado o tempo e condições iniciais, a solução é calculada:

```
tspan=[0 20];
y0=[2;0];
[t,y]=ode45('vonder',tspan,y0);
plot(t,y)
```

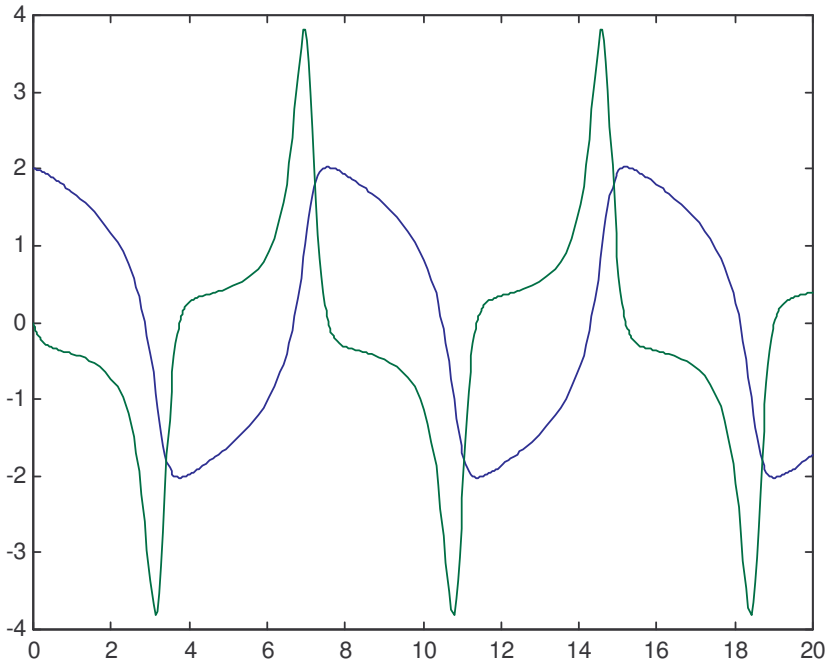

Figura E.3 – Equação Diferencial de Van Der Pol

E.6 - Conclusão

O estudo do Matlab foi realizado para uma maior compreensão dos programas que foram executados no trabalho, tendo como objetivo de aprender a desenvolver programas no Matlab.