Introdução aos métodos computacionais em biomedicina

3º ano | LEBIOM

Relatório do trabalho nº 2

Realizado por:

Ana Rita Damas Mendes (99641) ritadmendes@tecnico.ulisboa.pt
Inês Pereira Marques (99674) ines.p.marques@tecnico.ulisboa.pt
Sofia Silvestre de Oliveira (99715) sofia.de.oliveira@tecnico.ulisboa.pt

Grupo 4

2022/2023 - 1º Semestre,P2

Conteúdo

1 Objetivo

Este projeto pretende resolver Equação Diferencial Ordinária(EDO's) de primeira ordem empregando o Método de Runge-Kutta de terceira ordem (RK3). Iremos testar o nosso programa para os casos específicos dados pelo enunciado incluindo uma descida de pára-quedas. Adicionalmente, faremos uma análise do erro do método numérico variando o 'step'. A ideia básica deste método é aproveitar as qualidades dos métodos da série de Taylor e ao mesmo tempo eliminar o seu maior defeito que é o cálculo de derivadas de f (x, y) que torna os métodos de série de Taylor computacionalmente ineficientes.

2 Introdução Teórica

Uma EDO é uma equação diferencial em que a função desconhecida depende apenas de uma variável independente e a sua ordem corresponde à maior ordem das derivadas que a constituem. No nosso caso apenas trabalharemos com EDO's de primeira ordem ($y' = f(x, y)$).Estamos perante uma equação diferencial com condição inicial quando se define um ponto (x_0, y_0),com $y(x_0) = y_0$, num intervalo $[a, b]$ contendo x_0. Os Métodos Runge-Kutta são variantes obtidos por meio da expansão da série de Taylor com o resto de Lagrange mas que evitam o cálculo de derivadas de ordem superiores a 1 e procuram substituir as derivadas parciais por outras avaliações pontuais em f. A função f(x) pode ser expandida numa série de Taylor na vizinhança de um ponto x da seguinte forma:

$$f(x) = \sum_{k=0}^{\infty} \frac{f^k(a)}{k!}(x-a)^k = f(a) + f'(a)\frac{x-a}{1!} + ... + f^{(k)}(a)\frac{(x-a)^k}{k!}. \tag{1}$$

Para x qualquer, existe um ponto b pertencente ao intervalo [a,x] que corresponde ao valor exato do erro da aproximação que será feita, segundo a fórmula do Resto de Lagrange:

$$R_k(a, x) = f^{(k+1)}(b)\frac{x - a^{(k+1)}}{(k+1)!} \tag{2}$$

Através do desenvolvimento de Taylor com resto de lagrange até k=3, substituindo em $f(x) + R_k(a, x)$,a por x_n e x por $x_{n+1} = x_n + h(= step)$ chegamos à seguinte equação:

$$y_{(x_{n+1})} = y(x_n) + hy'(x_n) + \frac{h^2}{2!}y''(x_n) + \frac{h^3}{3!}y'''(x_n) + \frac{h^4}{4!}y''''(c) \tag{3}$$

O valor da derivada a usar para aproximar a função no ponto i+1, é uma média ponderada dos três valores encontrados, sendo que se atribui mais peso ao valor intermédio, i.e. $K2$, pela fórmula corretora de Milne. Esta é uma variação do método RK3 tradicional que inclui um termo de correção para melhorar a precisão da solução.

$$y_{i+1} = y_i + h(ak_1 + bk_2 + ck_3) \tag{4}$$

onde a =1, b =4 e c =1. Nesse caso,

$$k_1 = f(x_i, y_i) \qquad k_2 = f(x_i + ph, y_i + phk_1) \qquad k_3 = f(x_i + rh, y_i + shk_2 + (r - s)hk_1) \tag{5}$$

onde k_1, k_2 e k_3 são aproximações das derivadas em vários pontos no intervalo de integração $[x_i, x_{i+1}]$. Para determinar as constantes a, b, c, p, r e s , expandem-se k_2 e k_3 em Taylor, em torno de (x_i, y_i) e obtêm-se as seguintes equações:

$$a + b + c = 1 \qquad bp + cr = \frac{1}{2} \qquad bp^2 + cr^2 = \frac{1}{2} \qquad cps = \frac{1}{6} \tag{6}$$

Duas das constantes são escolhidas arbitrariamente. Para um determinado conjunto de constantes escolhido por Kutta, tem-se o seguinte método de 3ª ordem(que equivale à Regra de Simpson se $\frac{dy}{dx} = f(x)$).

$$y_{i+1} = y_i + \frac{h}{6}(k_1 + 4k_2 + k_3) \tag{7}$$

$$k_1 = f(x_i, y_i) \qquad k_2 = f(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_1) \qquad k_3 = f(x_i + h, y_i + 2hk_2 - hk_1) \tag{8}$$

Assim, teoricamente, o método RK3 usado apresenta um erro local dado pelo resto de Lagrange $\frac{h^4}{(4)!}y''''(b)$, logo é de ordem $O(h^4)$, e portanto o erro máximo, que é fruto da acumulação dos erros locais apresentará ordem $O(h^3)$.

3 Resultados

3.1 Caso A,B e C

Caso A: $y' = t–y, \quad y(0) = 2, \quad t \in (0, 4];$
Solução da equação diferencial: $y(t) = 3e^{-t} + t - 1$
Caso B: $y' = 20t^2–20y + 2t \quad y(0) = 1, \quad t(0, 1];$
Solução da equação diferencial: $y(t) = t^2 + e^{-20t}$
Caso C: $y' = (2e^{-t}–1)y, \quad y(0) = 1, \quad t(0, 10];$
Solução da equação diferencial: $y(t) = e^{-t-2e^{-t}+2}$

A solução numérica de EDOs envolve dois tipos de erro:

1. Erros de truncamento: causados pela natureza das técnicas usadas para aproximar os valores de y. Os erros de truncamento são compostos por duas partes. O primeiro é um erro de truncamento local que resulta de uma aplicação do método em questão em uma única etapa. O segundo é um erro de truncamento propagado que resulta das aproximações produzidas durante as etapas anteriores. A soma dos dois é o erro global ou de truncamento global. [2]

2. Erros de arredondamento: causados pelo número limitado de dígitos significativos que podem ser retidos por um computador. [2]

Analisando os gráficos abaixo podemos verificar que diminuindo o h temos uma melhor aproximação à função exata, no entanto, mais tarde iremos verificar que não é tão linear assim.

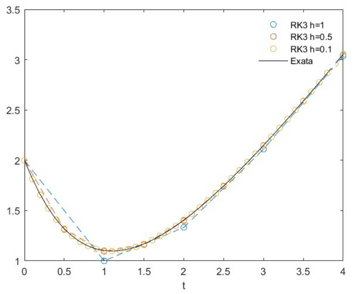

Figura 1: Caso 1

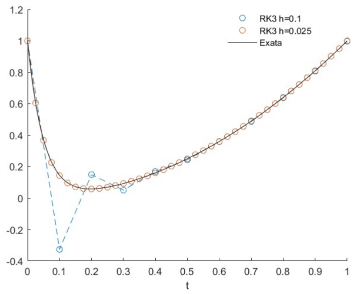

Figura 2: Caso 2

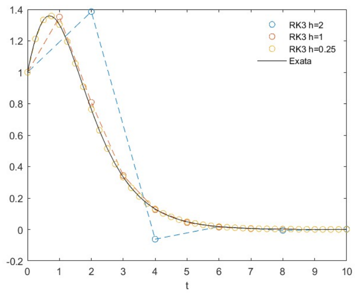

Figura 3: Caso 3

Em geral, a maneira que temos para minimizar os erros de aproximação será aumentar os algarismos significativos do computador, no entanto o MATLAB tem uma limitação de até 16 digitos de precisão. Já em relação aos erros de truncamento, temos de reduzir o passo (h) para que este diminua. No entanto, quanto menor o passo, mais cálculos computacionais serão feitos pelo que erros de aproximação irão aumentar. Assim, encontramos um dilema: até que ponto a diminuição do passo é vantajoso? Temos de determinar um tamanho de passo apropriado para determinada computação. O desafio é identificar o ponto onde o erro de arredondamento começa a anular os benefícios da redução do tamanho do passo. [2] Iremos estudar este erro numérico mais à frente, onde vamos considerar o erro do truncamento do método utilizado, como o erro máximo (max (abs (y - yExact))).

3.2 Caso D : Descida de um paraquedista

A descida de um paraquedista pode ser descrita por uma equação diferencial e, por isso, podemos utilizar o método Runge-Kutta para realizar uma simulação do movimento do paraquedista em função do tempo. Tendo em conta que durante a descida atuam duas forças no conjunto pára-quedas+paraquedista, a força gravítica e a força da resistência do ar, conseguimos obter a força resultante a que o sistema está sujeito e, a partir da segunda lei de Newton, obter a equação diferencial pretendida. A fórmula da força da resistência do ar é

$$F_{RA} = \frac{1}{2}\rho c_d A v^2 \tag{9}$$

ρ - densidade do ar (aproximadamente $1.22kg/m^3$)
A - área do pára-quedas
c_d - coeficiente de arrasto (*drag coefficient*), depende da geometria e das propriedades materiais do pára-quedas bem como da sua aerodinâmica.
v - velocidade.

Dado que c_d e A variam conforme o pára-quedas está aberto ou fechado, a resultante de arrasto $C_d = \frac{1}{2}\rho c_d A$ toma dois valores distintos durante a descida, $C_d \approx 0.5$ até ao instante de abertura do pára-quedas e $C_d 2 \approx 20$ depois da abertura, uma vez que a área de superfície sujeita à resistência do ar com o pára-quedas aberto é bastante superior. Logo, quando um corpo se desloca a uma velocidade elevada e se abre o pára-quedas, a força da resistência do ar será superior ao peso, e provocará a desacelaração do paraquedista.

Através da Segunda Lei de Newton, obteve-se a seguinte equação diferencial ordinária não linear que descreve a aceleração (i.e. a derivada da velocidade) de um paraquedista em descida:

$$\frac{dv}{dt} = -g + \frac{C_d}{m} \times v^2 \tag{10}$$

sendo g a aceleração gravítica (9.8 m s^{-2}), m a massa do conjunto paraquedista+pára-quedas e C_d constante de proporcionalidade que tem conta a resistência do ar, sendo diferente no caso do paraquedista em queda sem o pára-quedas (c_d≈0.5, A≈1m^2), e no caso do paraquedista com o mesmo (c_d2≈1.75, A≈19m^2). [1] Ao resolver esta EDO com o método RK3, tendo em conta o intervalo [0,50] s, g=9.8 ms^2; C_d=0.5; C_{d2}=20; m=90 Kg (70Kg+20Kg do pára-quedas); t_0=0 s ; v(0)=0 m/s; $t_{abertura}$=25 s, observou-se a Fig.4.

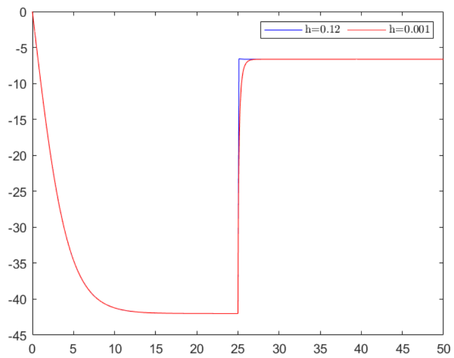

Figura 4: Evolução da velocidade de um paraquedista

Notou-se que para um h superior a 0.1 a desaceleração não descreve uma curva, pois é necessário um h inferior.

É possível observar um movimento acelerado do paraquedista, desde o seu lançamento (em t=0 s) até atingir a primeira velocidade terminal (em t≈ 15 s) de -42 m/s [1]. A partir dos 25 segundos, instante no qual abre o pára-quedas, constata-se um movimento desacelerado até atingir a segunda velocidade terminal (em $t \approx 30$ s) de -6.6408 m/s. Os parâmetros foram escolhidos de modo a traçar a descida de pára-quedas num contexto real. [3]

Para escolher o $t_{abertura}$ observou-se primeiramente o gráfico da equação com o pára-quedas fechado para conhecer a partir de que instante o sistema chegava à primeira velocidade terminal.

Tempo de chegada em função da altura e do instante de abertura do pára-quedas

Considerando os mesmos parâmetros utilizados na Fig.4 (à exceção de h, que se aumentou para 0.1 de forma a acelerar o processamento computacional), definimos um vetor com diferentes valores para $t_{abertura}$(1:h:75) e uma altura inicial de lançamento hi=3065 m. Utilizando 2 vezes consecutivas o método RK3, encontramos a função aproximada y(t) $((\frac{dv}{dt} \xrightarrow{RK3} \frac{dy}{dt} \xrightarrow{RK3} y(t))$ e a partir desta obter o tempo de chegada. É de notar que se utilizou interpolação polinomial (*polyfit* de grau 2) dos valores obtidos do método RK3 para que se pudedesse usar o método uma segunda vez, e novamente para encontrar uma aproximação da equação do movimento y(t) para antes e após a abertura do pára-quedas. Uma vez que as funções utilizadas para a

[1] Considerou-se que o paraquedista se movia ao longo do eixo vertical y, na direção contrária a este.

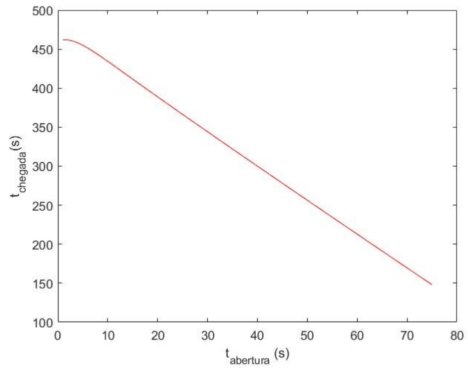

Figura 5: $t_{chegada}$ em função de $t_{abertura}$

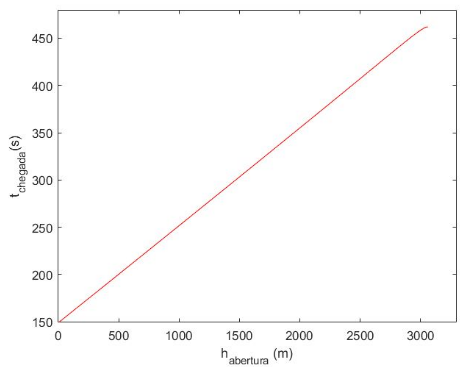

Figura 6: $t_{chegada}$ em função da $h_{abertura}$

obtenção dos gráficos abaixo são aproximadas, os valores apresentam um erro associado mas que não impede a visualização da relação entre as variáveis pretendidas.

Na Fig. 5, observa-se uma diminuição do tempo de chegada à medida que o instante de abertura do pára-quedas aumenta. Esta evolução está de acordo com o esperado pois quanto mais tempo o paraquedista estiver com a 1ª velocidade terminal (maior que a segunda), e por isso com um maior t$_{abertura}$, mais rápido este chega ao chão (i.e. menor o tempo de chegada).

Na Fig. 6, vemos que quanto maior a altura de abertura do pára-quedas, maior o tempo de chegada ao solo. Isto está de acordo com o previsto, uma vez que ao se abrir o pára-quedas a velocidade diminui bastante, pelo que quanto mais alto se abrir o paraquedas, mais tempo demorará a atingir o solo.

4 Erro numérico

Para aplicarmos o método de Runge-Kutta de ordem 3 é necessário ter em consideração a eficiência e precisão para encontrar um h ideal. No entanto, a simultaneidade de máxima eficiência e precisão são incompatíveis. Por isso, é necessário encontrar a região de estabilidade, que é alcançada quando o erro numérico causado pelo método é aceitável e não impede a obtenção de uma solução precisa.

No entanto, o valor do h ideal pode variar no decorrer de uma mesma função pois pode haver zonas da função em que esta não varia muito e outras em que varie abruptamente e seja necessário h mais pequeno para descrever a função. Existem várias formas de se avaliar a estabilidade num método de Runge-Kutta de ordem 3. Uma das formas é analisar o comportamento do erro numérico variando o valor de h . Quando o erro numérico é pequeno e não aumenta, diminuindo h, o método é considerado estável nesse intervalo de h. Para encontrarmos essa região foi utilizado o programa ploterropasso.m que nos indica o erro máximo para os diferentes h. (Considerámos a região de estabilidade aquele que o erro seria inferior a 10^{-14})

Na tabela abaixo, encontra-se as diferentes zonas de estabilidade para os diferentes casos:

Caso	Passo mínimo	Passo máximo
1	3.8147×10^{-6}	6.10352×10^{-5}
2	2.384×10^{-7}	3.8147×10^{-6}
3	4.7684×10^{-6}	7.62939×10^{-5}

Os seguintes gráficos mostram como evolui o erro com a diminuição do valor de h. Um erro mínimo é alcançado quando h é aproximadamente 10^{-6}. Além desse valor de h, o erro aumenta à medida que os erros arredondamentos dominam. É de notar que existe um limite computacional para o qual se pode diminuir o h (aproximadamente 10^{-8}) .

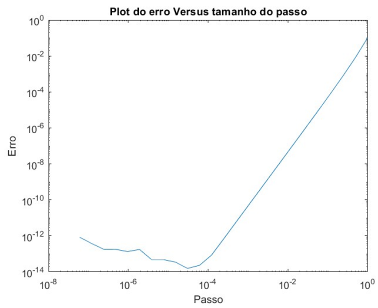

Figura 7: Caso 1

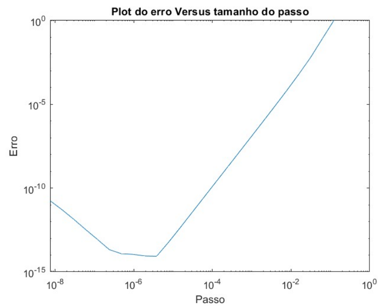

Figura 8: Caso 2

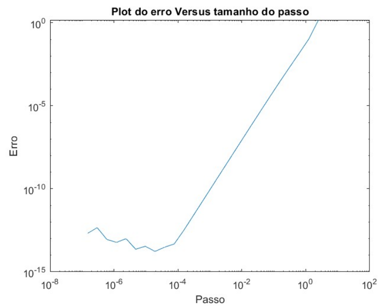

Figura 9: Caso 3

Através do estudo do comportamento assimptótico das funções pode-se verificar qual é a ordem do método utilizado. Partindo da expressão matemática:

$$E = Ch^{\alpha} \tag{11}$$

sendo E o erro máximo do método numérico utilizado em relação à solução exata, C uma constante, h o tamanho do passo e α é a ordem do método usado. Se logartizarmos a expressão acima temos que:

$$logE = logC + \alpha logh \tag{12}$$

Assim, foi utilizado a função *polyfit* para determinar os coeficientes do polinómio de primeiro grau que melhor interpola os valores de logaritmo de h e logaritmo de erro máximo. O declive dessa reta corresponde ao primeiro desses coeficientes. Foram utilizados os h no intervalo de $[\frac{b_{caso}}{8}\ ,\ \frac{b_{caso}}{8\times 2^{14}}\ ]$. Sendo bcaso = 4, 1, 10 , respetivamente.

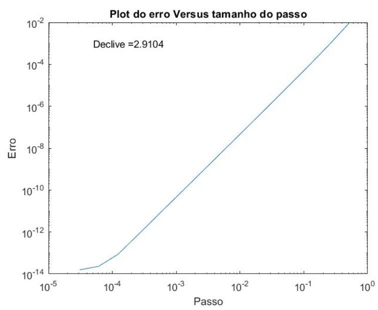

Figura 10: Caso 1

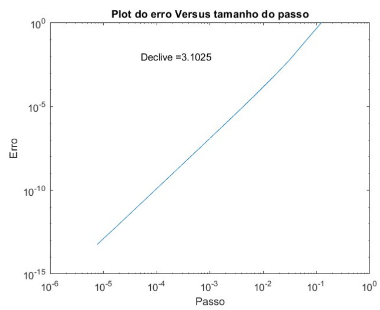

Figura 11: Caso 2

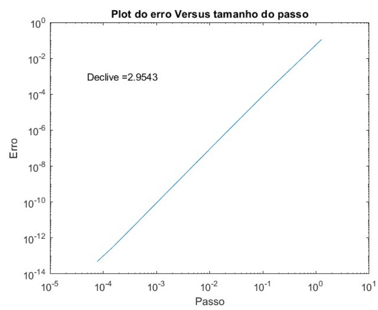

Figura 12: Caso 3

Observando os declives dos casos 1,2 e 3 verificamos que o método de Runge-Kutta verifica o comportamento assimptótico de 3ª ordem, que seria o esperado.

Referências

[1] Tom Benson. Velocity during recovery.

[2] Steven C. Chapra Raymond P. Canale. *Numerical Methods for Engineers*. McGraw-Hill,, 2010.

[3] Ronald Phoebus and Cole Reilly. Differential equations and the parachute problem, 2004.

Appendices

A Programas Principais

RK3.m: Método Runge-Kutta de ordem 3 . Utilizador tem de dar:

1. f (EDO de primeira ordem)
2. t (intervalo t = a:h:b)
3. y(0)

```
%RK3
function y = RK3(f, t, y0)
n=length(t);
y=nan(length(y0),n);
y(:,1)=y0(:);
for k=1:n-1
    h=t(k+1)-t(k);
    F1=f(t(k),y(:,k));
    F2=f(t(k)+h/2,y(:,k)+h*(F1)/2);
    F3=f(t(k)+h,y(:,k)+h*(2*F2-F1));
    y(:,k+1)=y(:,k)+h*(F1+4*F2+F3)/6; %corrigido
end
end
```

PlotErroPasso.m: Programa que devolve um gráfico log(Erro máximo) vs log(tamanho do passo). É possível visualizar no command window os valores h utilizados e o seu erro associado. Utilizador tem de dar:

1. f (EDO de primeira ordem)
2. a e b (intervalo de integração)
3. y(0)

É de salientar que a exata tem de ser alterada no código consoante a EDO escolhida. (linha 13)

```
1  %ERRO M XIMO VS TAMANHO DO PASSO
2  %mudar a exata consoante o caso
3  function PlotErroPasso(f,a,b,y0)
4  n=15; %  MAXIMO    POR VOLTA DOS 25
5  format long
6  H=zeros(1,n);
7  E=zeros(1,n);
8  h = b/8; %sempre 1/8 de B do caso
9  for i=1:n
10     H(i)=h;
11     t = a:h:b;
12     y =RK3(f, t, y0);
13     yExact = exp(-t-2.*exp(-t)+2); %colocar exata
14     E(i)=max(abs(y-yExact)); %erro maximo para o passo
```

```
    h=h/2;% Iterativamente, dividindo por 2
end
L=[H' E']';
fprintf(' Passo  Erro M ximo\n');
fprintf('%14.10f  %16.13f\n',L);
loglog(H,E),xlabel('Passo'),ylabel('Erro') %grafico loglog
title('Plot do erro Versus tamanho do passo')
coef=polyfit(log(H),log(E),1);
declive=coef(1);  %d  o declive
declive_string = num2str(declive);
declive_grafico = strcat('Declive = ', declive_string);
text(0.00005, 0.001, declive_grafico); %imprime o declive no gr fico

end
```

B Plots dos gráficos

GráficoPlot.m: Utilizado para fazer os gráfico das figuras 1-3.

```
%Escolher Casos
%CASO1
f = @(t,y) t-y;
a=0;
b=4;
y0=2;
% y(t)=3*exp(-t)+t-1 %exata

%CASO2
%f = @(t,y) 20*t^2 - 20*y+2*t;
%a=0;
%b=1;
%y0=1;
%%exata y(t)=t^2 + exp(-20*t);

%CASO3
%f = @(t,y) (2*exp(-t) - 1)*y;
%b=10;
%y0=1;
%%exata y(t)=exp(-t-2*exp(-t)+2);

%%%%%%%%%%%%%%%%%%%%%%%%%%% plot
h = 1 ;
t = a:h:b;
y =RK3(f, t, y0);
plot(t, y, 'o--')
hold on
%%%%%%%%%%%%%%%%%%%%%%%%%%% plot
h2 = 0.5 ;
t2 = 0:h2:b;
y2 =RK3(f, t2, y0);
plot(t2, y2, 'o--')
hold on
%%%%%%%%%%%%%%%%%%%%%%%%%%%plot
h3 = 0.1 ;
```

```
t3 = 0:h3:b;
y3 =RK3(f, t3, y0);
plot(t3, y3, 'o--')
hold on
%%%%%%%%%%%%%%%%%%%%%%%%%% fplot
yExact = @(t) 3*exp(-t)+t-1;
fplot(yExact,[0,b], 'k-')
xlabel('t')
```

parachute.m:Utilizado para fazer o gráfico da Fig. 4.

```
%Dados pelo utilizador
a=0;b=50;
cd=0.5;%Cd paraquedas fechado
cd2=20;%Cd2 paraquedas aberto
g=9.8;m=90;tabertura=25;v0=0;
func=@(t,y) (-g+((cd/m)*(y^2)));%equac paraquedas fechado
func2=@(t,y) (-g+((cd2/m)*(y^2)));%equac paraquedas aberto
vt1=-sqrt((g*m)/cd); %velocidade terminal 1


hs=[0.12,0.001];
for i = 1:length(hs)
h=hs(i);
t1=a:h:tabertura;
t2=tabertura:h:b;
v1=RK3(func,t1,v0); %solu o aproximada
v2=RK3(func2,t2,vt1); % solu o aproximada
plot(t1,v1,'-')
hold on
plot(t2,v2,'-')
hold on
end
```

tchegada.m: Utilizado para fazer o gráfico da Fig. 5 e 6.

```
%condi es iniciais
a=0;b=600;
hi=3065; %altura do lan amento
cd=0.5;cd2=20;
g=9.8;m=90;
vt1=-sqrt((g*m)/cd); %velocidade terminal 1
c=0;%velocidade inicial de lan amento

h=0.1;
func=@(t,y) (-g+(cd/m)*(y^2));
func2=@(t,y) (-g+(cd2/m)*(y^2));

tabertura=1:h:75;
tchegada=zeros(length(tabertura),1);
hfalta=zeros(length(tabertura),1);

for k=1:length(tabertura)
   %antes de abertura
   t1=a:h:tabertura(k);
   v1=RK3(func,t1,c);
   p1 = polyfit(t1,v1,2); %interpola o polinomial
```

```
    fv1= @(t,y) p1(1)*t^2 +p1(2)*t +p1(3); %eq velocidade v(t)
    y1=RK3(fv1,t1,hi);
    p4=polyfit(t1,y1,2);
    fy1= @(t,y) p4(1)*t^2 +p4(2)*t +p4(3); %eq posi  o y(t)
    hfalta(k)= fy1(tabertura(k)); %altura no instante que se abriu

    % ap s abertura
    t2=tabertura(k):h:b;
    v2=RK3(func2,t2,vt1);
    p2 = polyfit(t2,v2,2); %interpola  o polinomial
    fv2= @(t,y) p2(1)*t^2 +p2(2)*t +p2(3); %eq velocidade v(t)
    y2=RK3(fv2,t2,hfalta(k));
    p3 = polyfit(t2,y2,2);
    fy2= @(t) p3(1)*t^2 +p3(2)*t +p3(3); %eq posi  o y(t)
    tchegada(k) = fzero(fy2,hfalta(k)) + tabertura(k);
end
%plot(tabertura,tchegada,'r-')
plot(hfalta,tchegada,'r-')
```

GráficoErro.m: Utilizado para fazer os gráficos de 7-12.

```
%CASO1: exata y(t)=3.*exp(-t)+t-1
%f = @(t,y) t-y;
%a=0;
%b=4; %intervalo
%y0=2;

%CASO2: exata y(t)=t.^2 + exp(-20.*t);
%f = @(t,y) 20*t^2 - 20*y+2*t;
%a=0;
%b=1;
%y0=1;

%CASO3: exata y(t)=exp(-t-2.*exp(-t)+2);
f = @(t,y) (2*exp(-t) - 1)*y;
a=0;
b=10;
y0=1;

PlotErroPasso(f,a,b,y0)
```