Referência Bibliográfica

1 SANTOS, A. V. **Projeto e Controle de Estabilidade de um Sistema Robótico Anfíbio para Sensoriamento Remoto**. Rio de Janeiro, 2007.120p. Dissertação de Mestrado - Departamento de Engenharia Mecânica. Pontifícia Universidade Católica do Rio de Janeiro.

2 IAGNEMMA, K.; DUBOWSKY, S. **Mobile Robot in Rough Terrain:** Estimation, Motion Planning, and Control with Application to Planetary Rovers. Berlim: Springer, 2004.110p.

3 SIMEÓN, T.;DACRE-WRIGHT, B. **A Practical Motion Planner for All-terrain Mobile Robots**. In IEEE International Conference on Intelligent Robots and Systems, Yokohama (Japão),1993.

4 TAROKH, M.; et al. **Kinematic Modeling of a High Mobility Mars Rover**. In IEEE International Conference in Robotics & Automation, Detroit (EUA), 1999.

5 BALARAM, J. **Kinematic state estimation for a Mars Rover**. Robotica, v. 18, 2000, pp. 251-262.

6 CALTABIANO, D.; MUSCATO, G. **A Comparison Between Different Traction control Methods For a Field Robot**. In International Conference on Intelligent Robots and Systems, Proceedings of the 2002 IEEE, Lausanne (Suíça).

7 ALBAGUL, A.; WAHYUDI. **Dynamic Modelling and Adaptive Traction Control for Mobile Robots**. International Journal of Advanced Robotic Systems, v. 1, n.3, pp. 149-154, 2004.

8 GRAND, C.; et al. **Decoupled control of posture and trajectory of the hybrid wheel-legged robot Hylos**. In IEEE International Conference on Robotics & Automation. New Orleans (EUA), 2004.

9 ANWAR, S. **Brake-Based Vehicle Traction Control via Generalized Predictive Algorithm**. SAE International, 2003.

10 SAKAI, S.; SADO, H.; HORI, Y. **Motion Control in an Electric Vehicle with Four Independently Driven In-Wheel Motors**. IEEE/ASME Transactions on Mechatronics, v. 4, n.1, pp. 9 -16, 1999.

11 LAMON, P.; SIEGWART, R. **Wheel Torque Control in Rough Terrain**: Modeling and Simulation. In IEEE International Conference on Robotics & Automation, Barcelona (Espanha), 2005.

12 CALTABIANO, D.; CIANCITTO, D.; MUSCATO, G. **Experimental Results on a Traction Control Algorithm for Mobile Robots in Volcano Environment**. In IEEE International Conference on Robotics & Automation. New Orleans (EUA), 2004.

13 SARKAR, N.; YUN, X. **Traction Control of Wheeled Vehicles Using Dynamic Feedback Approach**. In IEEE International Conference on Intelligent Robots and Systems, Victoria (Canadá), 1998.

14 BURG, J.; BLAZEVIC, P. **Anti-Lock Braking and Traction Control Concept for All-Terrain Robotic Vehicles**. In IEEE International Conference on Robotics and Automation, Albuquerque (EUA), 1997.

15 CHATILA, R. et al. A. **Motion control for a Planetary Rover**. In International Conference on Field and Service Robotics, Pittsburgh (EUA), 1999.

16 DEMIDOVITCH, B. **Problemas e Exercícios de Análise Matemática**. 5 ed., Moscou: Editora MIR, 1977. pp. 85-106.

17 ANTON, H.;RORRES, C. **Álgebra Linear com Aplicações**. Tradução Claus Ivo Doering, 8. ed, Porto Alegre: Bookman, 2001, p. 195.

18 KREYSZIG, E. **Matemática Superior**, v. 2, Rio de Janeiro: editora LTC, 1974, p. 325.

Apêndice A:
Prova das Razões da Força de Atrito pela Força Normal (Fat/N) para a Análise Estática

Considerando a função R dada por:

$$R = \max(|G_1|, |G_2|) \tag{1}$$

onde:

$$G_1 = \frac{Fat_1}{N_1} \tag{2}$$

$$G_2 = \frac{Fat_2}{N_2} \tag{3}$$

As normais são dadas, de acordo com (48) e (49), por:

$$N_1 = e_1 - S_1 \cdot Fat_1 \tag{4}$$

$$N_2 = e_2 - S_2 \cdot Fat_2 \tag{5}$$

E as forças de atrito estão relacionadas , conforme a equação (54) por:

$$a_1 \cdot Fat_1 + a_2 \cdot Fat_2 = a \tag{6}$$

$$Fat_2 = -\frac{a_1}{a_2} \cdot Fat_1 + \frac{a}{a_2} \tag{7}$$

Assim, de acordo com as equações (4) a (7) acima a função R será função apenas de Fat_1. No entanto, Fat_1 é limitada pelas seguintes restrições:

a) $|Fat_1| \le F_{sat}$;

b) $|Fat_2| \le F_{sat}$;

c) $N_1 > 0$;

d) $N_2 > 0$.

Desta forma, haverá um intervalo $I_1 = [fat1\_\min, fat1\_\max]$ de possíveis valores de Fat_1, sendo que pretende-se encontrar o valor de $Fat_1 \in I_1$ que minimize R. Para isso, faz-se necessário analisar o crescimento (decrescimento) de G_1 e G_2 , através do cálculo da derivada dessas funções em relação a Fat_1 e Fat_2, respectivamente.

Assim, tem-se que:

$$\frac{dG_1}{dFat_1} = \frac{(e_1 - S_1 \cdot Fat_1) - Fat_1 \cdot (-S_1)}{(e_1 - S_1 \cdot Fat_1)^2} = \frac{e_1}{(e_1 - S_1 \cdot Fat_1)^2} \tag{8}$$

De forma análoga, a derivada de G_2 em relação a Fat_2 é dada por:

$$\frac{dG_2}{dFat_2} = \frac{e_2}{(e_2 - S_2 \cdot Fat_2)^2} \tag{9}$$

Analisando as equações (8) e (9), nota-se que G_1 e G_2 serão monotônicas, ou seja, crescentes ou decrescentes. Como os possíveis valores de Fat_1 resultam sempre normais positivas, os módulos de G_1 e G_2 irão mudar a sua monotonicidade em $Fat_1 = 0$ e $Fat_2 = 0$, respectivamente.

Para esta análise serão considerados os casos em que as forças de atrito possuem o mesmo sinal, sem perda de generalidade.

Assim, têm-se duas possibilidades para o mínimo de R, a saber:

A. Caso em que $|G_1|$ e $|G_2|$ têm a mesma monotonicidade, ou seja, são ambas crescentes ou decrescentes. Se ambas forem crescentes, então o mínimo será em $Fat_1 = fat_1\_\min$. Já se ambas forem decrescentes, o mínimo será em $Fat_1 = fat_1\_\max$.

B. Caso em que $|G_1|$ e $|G_2|$ têm monotonicidades diferentes.

   Considerando neste caso, sem perda de generalidade, que $|G_1|$ é crescente e $|G_2|$ é decrescente. Supondo $Fat_1 = \overline{F}at_1$ o valor que minimiza R e que neste valor $|G_1| \neq |G_2|$, então há duas possibilidades para R:

   i) $R = \left|G_1(\overline{F}at_1)\right|$, sendo então que $\left|G_1(\overline{F}at_1)\right| > \left|G_2(\overline{F}at_1)\right|$. Tomando um apropriado número real $\varepsilon > 0$ tal que $\left|G_1(\overline{F}at_1 - \varepsilon)\right| > \left|G_2(\overline{F}at_1 - \varepsilon)\right|$. Resulta que em $Fat_1 = \overline{F}at_1 - \varepsilon$; $R = \left|G_1(\overline{F}at_1 - \varepsilon)\right|$. Como $|G_1|$ é crescente implica que $\left|G_1(\overline{F}at_1)\right| > \left|G_1(\overline{F}at_1 - \varepsilon)\right|$ e $Fat_1 = \overline{F}at_1 - \varepsilon$ será o valor da força de atrito que minimiza R, o que é um absurdo, de acordo com a hipótese feita.

ii) $R = \left| G_2(\overline{F}at_1) \right|$, de forma análoga ao item (i) acima, pode-se encontrar $\overline{\varepsilon} > 0$ tal que $\left| G_2(\overline{F}at_1 + \varepsilon) \right| > \left| G_1(\overline{F}at_1 + \varepsilon) \right|$, e como $|G_2|$ é decrescente $Fat_1 = \overline{F}at_1 + \overline{\varepsilon}$ gera um valor menor para R, o que invalida a hipótese de que $Fat_1 = \overline{F}at_1$ seja o valor da força de atrito da roda 1 que minimiza R.

Procedimento semelhante ao utilizado em (i) e (ii) acima pode ser usado em todo caso que considere $|G_1| \neq |G_2|$, assim sendo o valor da força de atrito na roda 1 que minimiza R é tal que gera $|G_1| = |G_2|$

Portanto, os valores de Fat_1 candidatos a minimizar a função R serão:

- $Fat_1 = fat_1\_\min$;
- $Fat_1 = fat_1\_\max$;
- $Fat_1 = \overline{F}at_1 / \left| G_1(\overline{F}at_1) \right| = \left| G_2(\overline{F}at_1) \right|$, se $\overline{F}at_1 \in I_1$.

O valor de Fat_1 acima que substituído em R resultar o menor valor é o que minimiza R no intervalo I_1.

Anexo I:
Códigos Fonte

Os programas desenvolvidos para a simulação do sistema utilizaram a linguagem MatLab.

Abaixo está o código fonte do programa que calcula a curva de centro e a armazena junto com a curva do perfil do terreno em Tab_terreno.

```
%curva_centros
%programa que gera curva do centro das rodas
%dado o perfil do terreno-x,fx
%o programa calcula os pontos (x_i,fx_i) associados com cada ponto do
%perfil do terreno
%e gerada a tabela Tab_terreno que possui os seguintes campos
%x  x_i fx  fx_i
echo off
%_____________________
global Tab_terreno
%declaracao de Tab_terreno
Tab_terreno=[];
%declaracao de dx
dx=x(2)-x(1);
%loop que faz uma varredura ao longo dos pontos do perfil do terreno
for i=1:size(x,2)
      %calculo da derivada no ponto (x(i),fx(i))
   if (i~=1)&(i~=size(x,2))
     dfi=fx(i+1)-fx(i);          %diferenca em y progressiva
     dfi_1=fx(i)-fx(i-1);        %diferenca em y regressiva
     %df_dx=(dfi-dfi_1)/(2*dx); %derivada tomando a media das diferencas
     df_dx=dfi/dx;
```

```
elseif i==1
   df_dx=(fx(2)-fx(1))/dx;
   dfi=df_dx*dx;
   dfi_1=df_dx*dx;
elseif i==size(x,2)
   df_dx=(fx(size(x,2))-fx(size(x,2)-1))/dx;
   dfi=df_dx*dx;
   dfi_1=df_dx*dx;
end
%--------------------------
%calculo dos pontos da curva de centro
    x_i=x(i)-r*df_dx/sqrt(1+df_dx^2);
 fx_i=fx(i)+r/sqrt(1+df_dx^2);

 %verificacao das descontinuidade
 if (i>1)&(i~=size(x,2))
     %verificaçao de descontinuidadade convexa(roda gira em torno do
%ponto x(i),fx(i))
    if (abs(dfi-dfi_1)>.2*r)
       alfa1=atan2(dfi_1,(x(i)-x(i-1)));      %angulo da tangente calcula com
%derivada regressiva
       alfa2=atan2(dfi,(x(i+1)-x(i)));        %angulo da tangente calcula com
%derivada progressiva

       if    ((sign(alfa1)==sign(alfa2))&(alfa1>alfa2))|(alfa1>=0&alfa2<=0)
%verifica se houve descontinuidade convexa
          n_pts=20;        %numero de pontos na curva de centros para este
%arco
          ang_teta=linspace((alfa1+pi/2),((alfa2+pi/2)),n_pts);
%calculo do angulo do arco da curva de centro
         %armazenamento em Tab_terreno

Tab_terreno=[Tab_terreno;[x(i)+0*(1:n_pts)]',[x(i)+r*cos(ang_teta)]',[fx(i)+0*(1:
n_pts)]',*sin(ang_teta)]';
```

```
                clear ang_teta;
                continue
              end %verificacao e armazenamento
            end %verificacao da diferença em y
            %-----------------------------------
             %verificaçao de descontinuidadade concava(dois pontos de contato)
            if (x_i<Tab_terreno(end,2))|(abs(dfi-dfi_1)>.2*r) %verifica se houve
%retorno de x_i ou se ha descontinuidade
              %calculo do ponto da curva de centros
              alfa1=atan2(dfi_1,(x(i)-x(i-1)));          %angulo da tangente calcula
%com derivada regressiva
              alfa2=atan2(dfi,(x(i+1)-x(i)));        %angulo da tangente calcula com
%derivada progressiva
              Psi=(pi-abs(alfa1-alfa2))/2;
              H=r/sin(Psi);

              x_i=x(i)-H*cos(Psi-alfa1);
              fx_i=fx(i)+H*sin(Psi-alfa1);

              %variaveis auxiliares
              k=size(Tab_terreno,1);
              fim=0;

 %faz varredura ate encontrar o valor de x_i da tabela que nao e maior que o atual
%x_i
               while (~fim)&(k>1)
                  if Tab_terreno(k,2)>x_i
                    k=k-1;
                  else
                    fim=1;
                  end%if (while)
                end%while
               %armazenamento
```

```
            Tab_terreno(k:end,:)=[x((i-(size(Tab_terreno,1)-k)):i)',[0*x((i-
(size(Tab_terreno,1)-k)):i)+x_i]',...
                    fx((i-(size(Tab_terreno,1)-k)):i)',[0*fx((i-(size(Tab_terreno,1)-
k)):i)+fx_i]'];
            Tab_terreno=[Tab_terreno;x(i),x_i,fx(i),fx_i];
            continue
        end%if ajuste descontinuidade

        %armazenamento para o caso em que nao ha descontinuidade
        Tab_terreno=[Tab_terreno;x(i),x_i,fx(i),fx_i];

      else  %armazena
        Tab_terreno=[Tab_terreno;x(i),x_i,fx(i),fx_i];
      end %primeiro if
    end%laco
    clear x_i fx_i k dfi dfi_1 df_dx k fim Psi alfa1 alfa2
    ----------------------------------------------------------------------------------
```

A seguir estão os códigos fontes dos programas utilizados na simulação do modelo flexível.

```
    %simula_ODE
    %arquivo que simula a dinamica de um veiculo robotico com suspensao
%flexivel utilizando a funçao ODE45
    echo off;close all;clear all
    global Tab_n Tab_fat Tab_pot Vd Tempo
    inicializa_dados_ODE;       %arquivo que inicializa os dados da simulaçao
    %velocidade desejada no centro de massa
    Vd=.5;   %m/s
    st_T0=[xc,yc,alfa,dX(1),dX(2),dX(3)]';      %estado inicial
    temp=[0 .3];                    %vetor com tempo inicial e final da simulacao
opcao=odeset('RelTol',1e-4); %parametros que fixa tolerancia relativa em 10^4
[t,e_t]=ode45(@Xp,temp,st_T0,opcao); %execucao da integracao do problema
                                %utilizando ode45
```

```
view_sim_ODE                    %visualizacao da simulacao
salva_dados                     %arquivo que salva dados da simulação
```

```
% inicializa_dados_ode
%programa que iniciliza dados para a simulacao dinamica 2D do veiculo
%robotico
global D_eta_1 D_eta_2 Fat
dados_veiculo_ODE       %carrega dados do veiculo
load ../terrenos/ramp_02.mat      %carrega arquivo com o perfil do solo
dt=0.1*sqrt(m/K);  %intervalo de tempo de cada passo da simulacao
%--------------------
%dados iniciais
inicio=500;                 %indice do pto inicial do centro da roda 1[c1]
alfa=0;                     %chute inicial para alfa
c1=[Tab_terreno(inicio,2),Tab_terreno(inicio,4)];   %fixa a roda 1
%deformaçoes das suspensoes e suas taxas de variaçao no tempo
eta_1=-P/(2*K);      %deformaçao da suspensao da roda 1
eta_2=-P/(2*K);      %deformaçao da suspensao da roda 2
D_eta_1=0;          %taxa de deformaçao da suspensao da roda 1
D_eta_2=0;          %taxa de deformaçao da suspensao da roda 2

%posicionamento do veiculo atraves do calcuclo das coordenadas dos
%centros das rodas (c1,c2) e de xc,yc e alfa
posiciona_vehicle

%*** velocidades iniciais do centro de massa***
dX=[0;0;0];
%iniciliza as Fat's e as forcas normais
Fat1=0;Fat2=0;Fat=[Fat1;Fat2];N=[1;1];
```

%**dados_veiculo_ODE**

```
%arquivo que entra com os dados do veiculo
global r h1 h2 L1 L2 K c mi P I m Tsat
%---------dados entrada veiculo
r=0.3;      %raio da roda
m=120;      %massa veiculo em kg
I=15.22;       %momento de inercia do veiculo ao longo do eixo z em kg.m^2
g=9.8;      %aceleracao da gravidade em m/s^2
P=m*g;      %peso do veiculo
K=1e7;      %rigidez transversal em N/m
c=2*.7*sqrt(K*m);   %coef. de amortecimento
h1=0.3;       %distancia do centro da roda ate o centro de massa na vertical para
%suspensao relaxada
h2=0.3;     %idem para roda 2
L1=0.7;     %distancia longitudinal ate o CG a partir do centro da roda 1 com mola
%relaxada
L2=0.7;     %idem para roda 2
%max. razao entre força de traçao e força normal
mi=0.6;
%Torque de saturacao do motor Tsat em N.m
Tsat=96;
```

%**posiciona_vehicle**

```
%posicionamento do veiculo recalculando o centro de massa e o angulo de
%pitch(alfa)
L=sqrt((L1+L2)^2+(h2-h1+eta_2-eta_1)^2);
if (eta_1<0)&(eta_2<0)
   c2=[c1(1)+L*cos(alfa);c1(2)+L*sin(alfa)];
   max_iter=900;              %const. com o maximo de iteracoes
   iter=1;                    %var. que controla o numero de iteracoes
   max_erro=1e-6;                %const. com o valor do max. erro permitido
   erro=(valor_gx(c2(1))-c2(2));   %var. que controla o erro
```

```
    if isempty(erro)
       'Erro no posicionamento'
       return
    end
     psi=alfa;
    %loop
    while (abs(erro)>max_erro)&(iter<=max_iter)
         if abs(erro+c2(2)-c1(2))<=L
           c2(2)=c2(2)+erro;
           c2(1)=c1(1)+sqrt(L^2-(c1(2)-c2(2))^2);
        else
           psi=subspace(c2-c1,[c2(1);valor_gx(c2(1))]-c1);
            c2=c1+[cos(psi),-sin(psi);sin(psi),cos(psi)]*(c2-c1);
         end
         erro=(valor_gx(c2(1))-c2(2));
        iter=iter+1;
     end

    psi=atan((h2-h1+eta_2-eta_1)/(L1+L2));

    if (c1(1)-c2(1))~=0
       teta=atan((c1(2)-c2(2))/(c1(1)-c2(1)));
    else
       teta=pi/2;
    end
     alfa=psi+teta;
     xc=c1(1)+L1*cos(alfa)-(h1+eta_1)*sin(alfa);
    yc=c1(2)+L1*sin(alfa)+(h1+eta_1)*cos(alfa);
elseif eta_2<0
    xc=c2(1)-L2*cos(alfa)-(h2+eta_2)*sin(alfa);
    yc=c2(2)-L2*sin(alfa)+(h2+eta_2)*cos(alfa);
     c1=[xc;yc]+[-L1*cos(alfa)+h1*sin(alfa);-L1*sin(alfa)-h1*cos(alfa)];
```

```
else
   xc=c1(1)+L1*cos(alfa)-(h1+eta_1)*sin(alfa);
   yc=c1(2)+L1*sin(alfa)+(h1+eta_1)*cos(alfa);
    c2=[xc;yc]+[L2*cos(alfa)+h2*sin(alfa);L2*sin(alfa)-h2*cos(alfa)];
end
```

%param_control_ODE

```
%funcao que calcula os parametros para o calculo das Fat's controladas
function
[a1_x,a2_x,a_x,a1_y,a2_y,a_y,a1_alfa,a2_alfa,a_alfa,e1,S1,e2,S2,eta_1,eta_2]=pa
ram_control_ODE(xc,yc,alfa,V_xc,V_yc,W_alfa)

global D_eta_1 D_eta_2 h1 h2 Fat K c P mi L1 L2 m I r Tab_N Tab_Fat
%calculo dos parametros do estado
%centro das rodas
c1=posiciona_centro(xc,yc,alfa,1);
c2=posiciona_centro(xc,yc,alfa,2);
%calcula angulos de contato
gama1=contact_angle(c1(1));
gama2=contact_angle(c2(1));
if isempty(gama1)|isempty(gama2)
   F_x=[];
   disp('Erro angulo de contato')
   return
end
if abs(gama1-gama2)<1e-6    %verifica se gama1 e gama2 sao iguais
   gama1=gama2;
end
%calculo das deformacoes
eta_1=-(xc-c1(1))*sin(alfa)+(yc-c1(2))*cos(alfa)-h1;   %deformacao da suspensao
%da roda 1
eta_2=-(xc-c2(1))*sin(alfa)+(yc-c2(2))*cos(alfa)-h2;   %deformacao da suspensao
%da roda 2
```

```
%calculo das velocidades de deformacoes D_eta's
D_x1=V_xc+L1*sin(alfa)*W_alfa;  %velocidade em x do pto 1
D_y1=V_yc-L1*cos(alfa)*W_alfa;  %velocidade em y do pto 1
D_x2=V_xc-L2*sin(alfa)*W_alfa;  %velocidade em x do pto 2
D_y2=V_yc+L2*cos(alfa)*W_alfa;  %velocidade em y do pto 2
V1_t=D_x1*cos(alfa)+D_y1*sin(alfa); %velocidade tangencial a carroceria do
%veiculo no pto 1
V2_t=D_x2*cos(alfa)+D_y2*sin(alfa); %velocidade tangencial a carroceria do
%veiculo no pto 2

%velocidade em x da roda 1
Vc1_x=(V1_t+(h1+eta_1)*W_alfa)/(cos(alfa)+tan(gama1)*sin(alfa));
%velocidade em x da roda 2
Vc2_x=(V2_t+(h2+eta_2)*W_alfa)/(cos(alfa)+tan(gama2)*sin(alfa));

%taxa de deformacao em 1
D_eta_1=-(D_x1-Vc1_x)*sin(alfa)+(D_y1-Vc1_x*tan(gama1))*cos(alfa);
%taxa de deformacao em 2
D_eta_2=-(D_x2-Vc2_x)*sin(alfa)+(D_y2-Vc2_x*tan(gama2))*cos(alfa);
%***** calculo dos parametros das Fats Desejadas**********************
%calculo dos parametros das fat's otimas
a1_x=cos(alfa)/(m*cos(gama1-alfa));
a2_x=cos(alfa)/(m*cos(gama2-alfa));
a_x=((K*eta_1+c*D_eta_1)*(sin(alfa)+tan(gama1-
alfa)*cos(alfa))+(K*eta_2+c*D_eta_2)*(sin(alfa)+tan(gama2-alfa)*cos(alfa)))/m;
a1_y=sin(alfa)/(m*cos(gama1-alfa));
a2_y=sin(alfa)/(m*cos(gama2-alfa));
a_y=((K*eta_1+c*D_eta_1)*(-cos(alfa)+tan(gama1-
alfa)*sin(alfa))+(K*eta_2+c*D_eta_2)*(-cos(alfa)+tan(gama2-alfa)*sin(alfa))-
P)/m;
a1_alfa=(h1+eta_1)/(I*cos(gama1-alfa));
a2_alfa=(h2+eta_2)/(I*cos(gama2-alfa));
a_alfa=((K*eta_1+c*D_eta_1)*(L1+tan(gama1-
alfa)*(h1+eta_1))+(K*eta_2+c*D_eta_2)*(-L2+tan(gama2-alfa)*(h2+eta_2)))/I;
```

```
%Parametros dos intervalos de possiveis Fat's
e1=(sin(alfa)/(cos(gama1)*cos(gama1-alfa))-tan(gama1));
S1=(K*eta_1+c*D_eta_1)*(tan(gama1-alfa)*sin(alfa)-cos(alfa))/cos(gama1);
e2=sin(alfa)/(cos(gama2)*cos(gama2-alfa))-tan(gama2);
S2=(K*eta_2+c*D_eta_2)*(tan(gama2-alfa)*sin(alfa)-cos(alfa))/cos(gama2);
%retorno
return
```

%**posiciona_centros**

```
%funcao que posiciona os centros das rodas sobre a curva de centros
%ci=posiciona_centro(xc,yc,alfa,opt)
%sendo opt- parametro que indica o centro de que roda sera ajustado, se da
%roda 1 ou da roda 2
function ci=posiciona_centro(xc,yc,alfa,opt)
global h1 h2 L1 L2
%var. de controle
 max_iter=100;          %const. com o maximo de iteracoes
 tol=1e-8;             %const. com o valor do max. erro permitido em metros
 erro=1e16;              %erro
 iter=1;               %var. que controla numero de iteracoes
 %posicionamento inicial do centro ci
 if opt==1
    yci=yc-L1*sin(alfa)-h1*cos(alfa);
    xci=xc-L1*cos(alfa)+h1*sin(alfa);
    L=-L1;
else
    yci=yc+L2*sin(alfa)-h2*cos(alfa);
    xci=xc+L2*cos(alfa)+h2*sin(alfa);
    L=L2;
end
%caso o valor absoluto do angulo de inclinacao (alfa) seja maior que 1e-9 entao
%entra-se no loop para se ajustar o centro da roda a curva de centro.
%Caso contrario o terreno e plano e o ajuste fica definido em um unico calculo
```

```
if abs(alfa)>1e-9
   %loop
   while (abs(erro)>tol)&(iter<max_iter)
       erro=yci-valor_gx(xci);
        yci=valor_gx(xci);
        xci=xci+erro*tan(alfa);
         iter=iter+1;
   end %while
else
   xci=xc+L*cos(alfa);
   yci=valor_gx(xci);
end
%parametros de verificacao do ajuste
if opt==1
   T=-(yci-yc+L1*sin(alfa))/cos(alfa);
   h=h1;
else
   T=-(yci-yc-L2*sin(alfa))/cos(alfa);
   h=h2;
end
%verificacao se o centro ficou acima do chassi do veiculo, que e impossivel
if ((isempty(T))|((T-h)>0))|((T<0)&((alfa>pi/2)|(alfa<-pi/2)))
   yci=yc+L*sin(alfa)-h*cos(alfa);
   xci=xc+L*cos(alfa)+h*sin(alfa);
elseif T<0
   yci=yc+L*sin(alfa);
   xci=xc+L*cos(alfa);
end
ci=[xci;yci];
 return
```

```
%contact_angle
%[ang,xc,yc]=contact_angle(xp)
%programa que calcula o angulo(ang) e o ponto de contato(xc,yc)
%entre a roda e o solo, dado o valor xp pertencente a curva de centros
%com coord. x=xp
function [ang,xc,yc]=contact_angle(xp)
global Tab_terreno
dx=Tab_terreno(2,1)-Tab_terreno(1,1);    %espaçamento entre os valores de x ao
%longo do perfil do terreno
id1=fix((xp-Tab_terreno(1,1))/dx)+1;       %chute inicial para o indice do valor
%procurado
if  (xp<Tab_terreno(1,2))+(xp>Tab_terreno(end,2))        %se esta condicao for
%verdade xp nao pertence a curva de centro
   ang=[];xc=[];yc=[];        %retorna vazio para todas as saidas
   return
end
%verifica se o indice esta dentro dos limites da dimensao da tabela Tab_terreno
if id1<1
  id1=1;
elseif id1>size(Tab_terreno,1)
   id1=size(Tab_terreno,1)-1;
end
%algoritmo de procura
 while (id1<size(Tab_terreno,1))*(id1>0)
   if Tab_terreno(id1,2)<=xp
       if Tab_terreno(id1+1,2)>=xp
          break
       else
         id1=id1+1;
       end
   else
      id1=id1-1;
   end
end %while
```

```
%calculo das saídas
%peso da interpolaçao
  p=(xp-Tab_terreno(id1,2))/(Tab_terreno(id1+1,2)-Tab_terreno(id1,2));
%calculo de xc
 xc=Tab_terreno(id1,1)+p*(Tab_terreno(id1+1,1)-Tab_terreno(id1,1));
%calculo de yc
 yc=Tab_terreno(id1,3)+p*(Tab_terreno(id1+1,3)-Tab_terreno(id1,3));
%calculo de ang
 ang=atan((Tab_terreno(id1+1,3)-Tab_terreno(id1,3))/(Tab_terreno(id1+1,1)-
Tab_terreno(id1,1)));
 return
```

```
%valor_gx
%v_y=valor_gx(xi)
%funcao que procura valor de y(v_y) em x=xi na curva de centros[g(x)]
%xi- ponto que se deseja calcular o valor de g(x)
function v_y=valor_gx(xi)
global Tab_terreno
dx=Tab_terreno(2,1)-Tab_terreno(1,1);   %passo em x
if (xi<Tab_terreno(1,2))|(xi>Tab_terreno(end,2))     %se condicao for verdade xi
nao pertence a curva de centros
   v_y=[]; %retorna vazio
   return
end

%chute inicial para indice de procura
if (fix((xi-Tab_terreno(1,2))/dx)+1)<1
   id1=1;
else
   id1=fix((xi-Tab_terreno(1,2))/dx)+1;
end
```

```
%procura os indices dos valore imediatamente menor que xi(id1) e
%imediatamente maior que xi(id2)
while (Tab_terreno(id1,2)>xi)&(id1>1)
   id1=id1-1;
end
id2=id1+1;
while (Tab_terreno(id2,2)<xi)&(id2<size(Tab_terreno,1))
    id1=id1+1;
    id2=id1+1;
end
%devolve o valor interpolado de v_y
if abs(Tab_terreno(id1,4)-Tab_terreno(id2,4))>1e-6
   v_y=Tab_terreno(id1,4)+...
     ((xi-Tab_terreno(id1,2))/(Tab_terreno(id2,2)-
Tab_terreno(id1,2)))*(Tab_terreno(id2,4)-Tab_terreno(id1,4));
else
   v_y=(Tab_terreno(id1,4)+Tab_terreno(id2,4))/2;
end
```

%control_pot_ODE

```
%funcao que controla a aceleracao desejada para o veiculo no centro de massa e
minimiza a potencia requerida
function [F_x,F_y,T_in]=control_pot_ODE(xc,yc,alfa,V_xc,V_yc,W_alfa)
global D_eta_1 D_eta_2 h1 h2 Fat K c P mi L1 L2 m I r Tab_n Tab_fat Tab_pot
Vd Tsat
%centro das rodas
c1=posiciona_centro(xc,yc,alfa,1);
c2=posiciona_centro(xc,yc,alfa,2);
%calcula angulos de contato
gama1=contact_angle(c1(1));
gama2=contact_angle(c2(1));
if isempty(gama1)|isempty(gama2)
   F_x=[];
```

```
   disp('Erro angulo de contato')
   return
end
if abs(gama1-gama2)<1e-6    %verifica se gama1 e gama2 sao iguais
   gama1=gama2;
end
%calculo dos parametros
[a1_x,a2_x,a_x,a1_y,a2_y,a_y,a1_alfa,a2_alfa,a_alfa,e1,S1,e2,S2,eta_1,eta_2]=pa
ram_control_ODE(xc,yc,alfa,V_xc,V_yc,W_alfa);
%% Geracao dos intervalos das Fat's  %
% geracao dos intervalos possiveis para Fat1 e Fat2
%intervalo da Fat1 I1=[fat1_min,fat1_max]
if abs(e1)<1e-6  %N1 e Fat1 sao desacopladas
   fat1_min=max(-Tsat/r,-mi*S1);
   fat1_max=min(Tsat/r,mi*S1);
else
   if e1>0
      if (1-mi*e1)>0
         fat1_min=max([-Tsat/r,(0.01*P-S1)/e1,-mi*S1/(1+mi*e1)]);
         fat1_max=min(Tsat/r,mi*S1/(1-mi*e1));
      else
         fat1_min=max([-Tsat/r,(0.01*P-S1)/e1,-mi*S1/(1+mi*e1),mi*S1/(1-
mi*e1)]);
         fat1_max=Tsat/r;
      end
   else       %e1<0
      if (1+mi*e1)>0
         fat1_min=max([-Tsat/r,-mi*S1/(1+mi*e1)]);
         fat1_max=min([(0.01*P-S1)/e1,Tsat/r,mi*S1/(1-mi*e1)]);
      else
         fat1_min=-Tsat/r;
         fat1_max=min([(0.01*P-S1)/e1,-mi*S1/(1+mi*e1),mi*S1/(1-
mi*e1),Tsat/r]);
      end
```

```
   end%(if interno e1>0)
end%(if maior)
%intervalo da Fat2 I2=[fat2_min,fat2_max]
if abs(e2)<1e-6  %N2 e Fat2 sao desacopladas
   fat2_min=max(-Tsat/r,-mi*S2);
   fat2_max=min(Tsat/r,mi*S2);
else
   if e2>0
      if (1-mi*e2)>0
         fat2_min=max([-Tsat/r,(0.01*P-S2)/e2,-mi*S2/(1+mi*e2)]);
         fat2_max=min(Tsat/r,mi*S2/(1-mi*e2));
      else
         fat2_min=max([-Tsat/r,(0.01*P-S2)/e2,-mi*S2/(1+mi*e2),mi*S2/(1-
mi*e2)]);
         fat2_max=Tsat/r;
      end
   else
      if (1+mi*e2)>0
         fat2_min=max([-Tsat/r,-mi*S2/(1+mi*e2)]);
         fat2_max=min([(0.01*P-S2)/e2,Tsat/r,mi*S2/(1-mi*e2)]);
      else
         fat2_min=-Tsat/r;
         fat2_max=min([(0.01*P-S2)/e2,-mi*S2/(1+mi*e2),mi*S2/(1-
mi*e2),Tsat/r]);
      end
   end%(if interno e2>0)
end%(if maior)
```

```
%verifica se existe intrvalo com normais positivas e sem deslizamento
if (fat1_min>fat1_max)|(fat2_min>fat2_max)
   disp('Nao e possivel gerar configuracao desejada')
   [fat1_min,fat1_max],
   [fat2_min,fat2_max],
   F_x=[];
   return
end

%Velocidade desejada no centro de massa e atualizacao dos intervalos das Fat's
%calculo pos pi's
pi_1=a1_x*cos(alfa)+a1_y*sin(alfa);
pi_2=a2_x*cos(alfa)+a2_y*sin(alfa);
a_0=a_x*cos(alfa)+a_y*sin(alfa);
%Velocidade inicial
V0_t=V_xc*cos(alfa)+V_yc*sin(alfa);
%Erro da velocidade
DV1_t=Vd-V0_t;
%constante
Kp=mi*9.8*sin(pi/6)/abs(Vd);
%atualizacao dos intervalos conforme a velocidade desejada Vd Fat's
if (-pi_1/pi_2)<0 %geracao do novo intervalo para Fat2 restringido-a para a
%velocidade desejada
  fat2_min_new=-(pi_1/pi_2)*fat1_max+(Kp*DV1_t-a_0)/pi_2;
  fat2_max_new=-(pi_1/pi_2)*fat1_min+(Kp*DV1_t-a_0)/pi_2;
else
   fat2_min_new=-(pi_1/pi_2)*fat1_min+(Kp*DV1_t-a_0)/pi_2;
   fat2_max_new=-(pi_1/pi_2)*fat1_max+(Kp*DV1_t-a_0)/pi_2;
end
```

```
%verifica situacao do novo intervalo de Fat2 em relacao ao antigo e calcula as
Fat's desejadas
if (fat2_min_new>fat2_max)|(fat2_max_new<fat2_min)  %os 2 sao disjuntos
   if sign(pi_1)==sign(pi_2)
      [min_v,id_min]=min([abs(pi_1*fat1_max+pi_2*fat2_max+a_0-
Kp*DV1_t),abs(pi_1*fat1_min+pi_2*fat2_min+a_0-Kp*DV1_t) ]);
         %Fat's desejadas
      Fat1_d=rem(id_min,2)*fat1_max+fix(id_min/2)*fat1_min;
      Fat2_d=rem(id_min,2)*fat2_max+fix(id_min/2)*fat2_min;
      clear min_v id_min
   else
      [min_v,id_min]=min([abs(pi_1*fat1_max+pi_2*fat2_min+a_0-
Kp*DV1_t),abs(pi_1*fat1_min+pi_2*fat2_max+a_0-Kp*DV1_t)]);
      %Fat's desejadas
          Fat1_d=rem(id_min,2)*fat1_max+fix(id_min/2)*fat1_min;
         Fat2_d=rem(id_min,2)*fat2_min+fix(id_min/2)*fat2_max;
      clear min_v id_min
   end%(if interno dos Fat_d's)

else %interseccao dos 2 diferente de vazio (minimiza potencia)
      %geracao dos novos intervalos para Fat1 e Fat2
   fat2_min=max(fat2_min_new,fat2_min); %ajusta o intervalo de Fat2 para a
%velocidade Vd
   fat2_max=min(fat2_max_new,fat2_max);
      if (-pi_2/pi_1)<0 %ajusta o intervalo de Fat1 em relacao ao intervalo de Fat2
       fat1_min=-(pi_2/pi_1)*fat2_max+(Kp*DV1_t-a_0)/pi_1;
       fat1_max=-(pi_2/pi_1)*fat2_min+(Kp*DV1_t-a_0)/pi_1;
   else
       fat1_min=-(pi_2/pi_1)*fat2_min+(Kp*DV1_t-a_0)/pi_1;
      fat1_max=-(pi_2/pi_1)*fat2_max+(Kp*DV1_t-a_0)/pi_1;
  end
    %calculo dos Fat's otimos
  delta_1=abs((V0_t+(h1+eta_1)*W_alfa)/cos(gama1-alfa));        %veocidade na
%roda 1 no instante atual
```

```
  delta_2=abs((V0_t+(h2+eta_2)*W_alfa)/cos(gama2-alfa));       %velocidade  na
%roda 2 no instante atual

  if gama1==gama2
   Fat1_d=(fat1_min+fat1_max)/2;   %Fat's desejadas
   Fat2_d=(fat2_min+fat2_max)/2;
  Else
 [min1,id1]=min([delta_1*abs((Kp*DV1_t-a_0)/pi_1),delta_2*abs((Kp*DV1_t-
a_0)/pi_2)]);
       Fat1_d=rem(id1,2)*(Kp*DV1_t-a_0)/pi_1;
       Fat2_d=fix(id1/2)*(Kp*DV1_t-a_0)/pi_2;
      clear min1 id1
 if
((Fat1_d>fat1_max)|(Fat1_d<fat1_min))|((Fat2_d>fat2_max)|(Fat2_d<fat2_min))
         if sign(pi_1)==sign(pi_2)
          [min2,id2]=min([delta_1*abs(fat1_min)+delta_2*abs(fat2_max),
                   delta_1*abs(fat1_max)+delta_2*abs(fat2_min)]);
            Fat1_d=rem(id2,2)*fat1_min+fix(id2/2)*fat1_max;
             Fat2_d=rem(id2,2)*fat2_max+fix(id2/2)*fat2_min;

          else
          [min2,id2]=min([delta_1*abs(fat1_min)+delta_2*abs(fat2_min),delta_1*a
bs(fat1_max)+delta_2*abs(fat2_max)]);
              Fat1_d=rem(id2,2)*fat1_min+fix(id2/2)*fat1_max;
              Fat2_d=rem(id2,2)*fat2_min+fix(id2/2)*fat2_max;
           end
          clear min2 id2
       end
   end

end
```

%**control_PI**

```
function [F_x,F_y,T_in]=control_PI(xc,yc,alfa,V_xc,V_yc,W_alfa)
global m I h1 h2 r Fat K c P mi L1 L2 Vd Tab_n Tab_fat Tab_pot Tempo Integral

%calculo dos parametros do estado
%centro das rodas
c1=posiciona_centro(xc,yc,alfa,1);
c2=posiciona_centro(xc,yc,alfa,2);

%calcula angulos de contato
gama1=contact_angle(c1(1));
gama2=contact_angle(c2(1));
if isempty(gama1)|isempty(gama2)
   F_x=[];
   return
end
%calculo dos parametros
[a1_x,a2_x,a_x,a1_y,a2_y,a_y,a1_alfa,a2_alfa,a_alfa,e1,S1,e2,S2,eta_1,eta_2]=pa
ram_control_ODE(xc,yc,alfa,V_xc,V_yc,W_alfa);
% %calculo das velocidades de deformacoes D_eta's
x1=xc-L1*cos(alfa);
y1=yc-L1*sin(alfa);
x2=xc+L2*cos(alfa);
y2=yc+L2*sin(alfa);

D_x1=V_xc+L1*sin(alfa)*W_alfa;
D_y1=V_yc-L1*cos(alfa)*W_alfa;
D_x2=V_xc-L2*sin(alfa)*W_alfa;
D_y2=V_yc+L2*cos(alfa)*W_alfa;

V1_t=D_x1*cos(alfa)+D_y1*sin(alfa);
V2_t=D_x2*cos(alfa)+D_y2*sin(alfa);

Vc1_x=(V1_t+(h1+eta_1)*W_alfa)/(cos(alfa)+tan(gama1)*sin(alfa));
```

```
Vc2_x=(V2_t+(h2+eta_2)*W_alfa)/(cos(alfa)+tan(gama2)*sin(alfa));

D_eta_1=-(D_x1-Vc1_x)*sin(alfa)+(D_y1-Vc1_x*tan(gama1))*cos(alfa);
D_eta_2=-(D_x2-Vc2_x)*sin(alfa)+(D_y2-Vc2_x*tan(gama2))*cos(alfa);
%----------- Geracao das Fat's ----------------
Fsat=2*48/r;                    %forca de saturacao dos motores
V0_t=V_xc*cos(alfa)+V_yc*sin(alfa);  %velocidade do veiculo ao longo do eixo
%de seu chassi
%somatorio da integral
if size(Tempo,1)==1
   Integral=0;
else
   Integral=Integral+(Tempo(end)-Tempo(end-1))*(Vd-V0_t);
end

%Atribuicao dos ganhos proporcional e integral
Kp=180*mi*9.8*sin(pi/6)/abs(Vd);  %ganho proporcional
Ki=150*mi*9.8*sin(pi/6)/abs(Vd);    %ganho integral

%definicao das forcas de atrito que devem ser obtidas
%o controle deseja obter forcas iguais nas duas rodas
Fat(1)=Kp*(Vd-V0_t)/2+Ki*Integral/2;
Fat(2)=Kp*(Vd-V0_t)/2+Ki*Integral/2;
%ajuste devido saturaçao do motor
if abs(Fat(1))>Fsat
   Fat(1)=Fsat*sign(Fat(1));
   Fat(2)=Fsat*sign(Fat(2));
end

%calculo das Normais
%---------- Na roda 1 ------------
if eta_1<-1e-9      %caso seja verdade o veiculo esta em contato com o solo
   %cacula a normal  para o valor das forcas de atrito
   % desejadas (Fat1 e Fat2), definida anteriormente
```

```
   N(1)=e1*Fat(1)+S1;
   F1=Fat(1);
   %verifica se normal e negativa
   if N(1)<=0
      N(1)=0;
      F1=0;
   end
   %correcao devido deslizamento
   if (abs(Fat(1))>mi*N(1))&(N(1)~=0)
      A1=[cos(gama1)+mi*sign(Fat(1))*sin(gama1),-sin(alfa);sin(gama1)-
mi*sign(Fat(1))*cos(gama1),cos(alfa)];
      U1=[-(K*eta_1+c*D_eta_1)*cos(alfa);(K*eta_1+c*D_eta_1)*sin(alfa)];
      u1=inv(A1)*U1;
      N(1)=u1(1);
      F1=sign(Fat(1))*mi*N(1);
   end%if deslizamento

else    %veiculo esta descolado do solo
            N(1)=0;
           F1=0;
end %if N1
clear A1 u1 U1

%------------ Na roda 2 ------------------
%idem a roda 1
if eta_2<-1e-9
            N(2)=e2*Fat(2)+S2;
           F2=Fat(2);

          if N(2)<=0
            N(2)=0;
          F2=0;
   end
```

```
        %correcao devido deslizamento
    if (abs(Fat(2))>mi*N(2))&(N(2)~=0)
        A2=[cos(gama2)+mi*sign(Fat(2))*sin(gama2),-sin(alfa);sin(gama2)-
mi*sign(Fat(2))*cos(gama2),cos(alfa)];
        U2=[-(K*eta_2+c*D_eta_2)*cos(alfa);(K*eta_2+c*D_eta_2)*sin(alfa)];
        u2=inv(A2)*U2;
        N(2)=u2(1);
        Fr=sign(Fat(2))*mi*N(2);
    end%if deslizamento

else
    N(2)=0;
    F2=0;
end %if N1
clear A2 u2 U2

%-------- Armazenamento dos dados ----------
%Das normais
Tab_n=[Tab_n;N];
%Das forcas que devem atuar na periferia da roda para um dado torque do motor
Tab_fat=[Tab_fat;Fat'];
%da potencia requerida pelo sistema
delta_1=abs((V0_t+(h1+eta_1)*W_alfa)/cos(gama1-alfa)); %velocidade na roda 1
%no instante atual
delta_2=abs((V0_t+(h2+eta_2)*W_alfa)/cos(gama2-alfa)); %velocidade na roda 2
%no instante atual
Tab_pot=[Tab_pot;delta_1*abs(Fat(1))+delta_2*abs(Fat(2))];
%calculo das forcas agindo na carroceria do veiculo
F_x=m*(a1_x*Fat(1)+a2_x*Fat(2)+a_x);
F_y=m*(a1_y*Fat(1)+a2_y*Fat(2)+a_y);
%calculo do torque
T_in=I*(a1_alfa*Fat(1)+a2_alfa*Fat(2)+a_alfa);
return
```

```
%view_sim_ODE
%arquivo que mostra a animaçao da simulacao
for i=1:size(e_t,1)
   xc=e_t(i,1);
   yc=e_t(i,2);
   alfa=e_t(i,3);
   c1=posiciona_centro(xc,yc,alfa,1);
   c2=posiciona_centro(xc,yc,alfa,2);
   if rem(i,15)==0 %desenha, a cada 15 passos, o estado do sistema
      clf
      print_vehicle;
      pause(t(i)-t(i-1));
   end
end
```

```
%salva_dados
%arquivo que salva dados em arquivo texto (.txt)
echo off
%obtencao dos dados para serem salvos
%ajusta os valores das tabelas para os pontos utilizados na resolucao do
%problema pela ODE45
%novas tabelas que armazenarao os dados reais da simulaçao
Tab_N=[];
Tab_Fat=[];
Tab_Pot=[];

%encontra os valores reais utilizados na simulacao
[v,ind_t]=intersect(Tempo,t);
[v,ind_x]=intersect(t,Tempo);

%ajusta as tabelas
Tab_N=Tab_n(ind_t,:);
Tab_Fat=Tab_fat(ind_t,:);
```

```
Tab_Pot=Tab_pot(ind_t,:);

%vetor de tempo e coordenada x
tx=Tempo(ind_t);
coord_x=e_t(ind_x,1);

%velocidade do CM em cada estado do sistema
Vc_t=e_t(:,4).*cos(e_t(:,3))+e_t(:,5).*sin(e_t(:,3));

%definicao da tabela de dados Tab_coord e Tab_veloc
Tab_coord=[coord_x,Tab_N,Tab_Fat,Tab_Pot];
Tab_veloc=[t,Vc_t];
clear Vc_t
plot_prop       %plota os graficos dos dados obtidos na simulaçao

%pergunta se deseja salvar os dados
resp=input('Deseja salvar esses dados ([1]-sim,[0]-nao):');
if resp==1
   %entrada do nome do arquivo pelo usuario
   nome=input('Digite o nome do perfil do terreno e o tempo de simulaçao com
dois digitos [terreno_tempo]:','s');
   %confirmacao do nome do arquivo
   conf=0;
   nome
   conf=input('Confirme nome ([1]-sim,[0]-nao):');
   nome_arq=strcat('..\dados_simula\sim_',nome);

   if conf==1
      nome_arq1=strcat(nome_arq,'.txt');
      nome_arq2=strcat(nome_arq,'_veloc.txt');
      save(nome_arq1,'Tab_coord','-ASCII','-TABS');
      save(nome_arq2,'Tab_veloc','-ASCII','-TABS');
      disp('Arquivos salvo como:')
      nome_arq1,nome_arq2
```

```
   else
      disp('Erro na execuçao do programa. Finalizando...')
      pause(1)
       beep
   end
else
   disp('Finalizando Programa...')
end
```

```
%plot_prop
%arquivo que plota as propriedades do sistema
%carrega os valores a partir das tabelas: Tab_coord e Tab_veloc
coord_x=Tab_coord(:,1);
Tab_N=Tab_coord(:,2:3);
Tab_Fat=Tab_coord(:,4:5);
Tab_Pot=Tab_coord(:,6);
t=Tab_veloc(:,1);
Vc_t=Tab_veloc(:,2);
%limites do eixo dos x
if coord_x(1)<coord_x(end)
   x_inf=coord_x(1);
   x_sup=coord_x(end);
else
   x_inf=coord_x(end);
   x_sup=coord_x(1);
end
% a)Plotagem das normais
figure(2)
plot(coord_x,Tab_N(:,1),coord_x,Tab_N(:,2)),
legend('\fontsize{12}\bf N_1','\fontsize{12}\bf N_2'),
ylabel('\fontsize{12}\bf [N]'),xlabel('\fontsize{12}\bf [m]'),
title('\fontsize{16}\bf Gráfico das Forças Normais');
```

```
xlim([x_inf x_sup]);

% b)Plotagem das Fat's
figure(3)
plot(coord_x,Tab_Fat(:,1),coord_x,Tab_Fat(:,2)),
legend('\fontsize{12}\bf F_1','\fontsize{12}\bf F_2'),
ylabel('\fontsize{12}\bf [N]'),xlabel('\fontsize{12}\bf [m]'),
title('\fontsize{16}\bf Gráfico das Forças de Traçao');
xlim([x_inf x_sup]);

% c)Plotagem da razao |Fat|/N
figure(4)
plot(coord_x,coord_x*0+mi,'r',coord_x,abs(Tab_Fat(:,1))./Tab_N(:,1),coord
_x,abs(Tab_Fat(:,2))./Tab_N(:,2)),
legend('\fontsize{12}\bf  \mu','\fontsize{12}\bf  |F_1|/N_1','\fontsize{12}\bf
|F_2|/N_2'),
xlabel('\fontsize{12}\bf [m]'),
title('\fontsize{16}\bf Gráfico |F_i|/N_i');
min1=min(abs(Tab_Fat(:,1))./Tab_N(:,1));
min2=min(abs(Tab_Fat(:,2))./Tab_N(:,2));
min_y=min(min1,min2);
ylim([0.9*min_y 1.2]);
xlim([x_inf x_sup]);

% d)Plotagem da velocidade no centro de massa
figure(5)
plot(t,Vc_t,t,t*0+Vd,'r'),
legend('\fontsize{12}\bf  Velocidade  Real','\fontsize{12}\bf  Velocidade
desejada'),
xlabel('\fontsize{12}\bf Tempo em [s]'),ylabel('\fontsize{12}\bf [m/s]'),
title('\fontsize{16}\bf Gráfico das Velocidades');

% e)Plotagem da Potencia dissipada
figure(6)
```

```
plot(coord_x,Tab_Pot),
ylabel('\fontsize{12}\bf [W]'),xlabel('\fontsize{12}\bf [m]'),
title('\fontsize{16}\bf Gráfico da Potência dissipada em Watts');
xlim([x_inf x_sup]);
clear Tab_N Tab_Fat Tab_Pot t Vc_t
```

```
%print_vehicle
%programa que plota o carro e o perfil
global contador

%plotaegm do perfil do terreno e geracao dos circulos das rodas
npts=500;
contador=1;
if contador==1
   xperfil=Tab_terreno(fix(linspace(1,size(Tab_terreno,1),npts)),1);
   yperfil=Tab_terreno(fix(linspace(1,size(Tab_terreno,1),npts)),3);
   gama=[0:pi/10:2*pi];
   xcirc=r*cos(gama);
   ycirc=r*sin(gama);
   contador=2;
   clear gama;
end

%plota perfil do terreno
plot(xperfil,yperfil);
hold on;

%geracao da carcaca
p1=[xc;yc]-[L1*cos(alfa);L1*sin(alfa)];
p2=[xc;yc]+[L2*cos(alfa);L2*sin(alfa)];
xm=0.5*(p1(1)+p2(1));
ym=0.5*(p1(2)+p2(2));
gama=[alfa:pi/20:(alfa+pi)];
```

```
raio=(L1+L2)/2;
xcarc=raio*cos(gama)+xm;
ycarc=raio*sin(gama)+ym;
clear gama xm ym raio


%rodas
%Geracao das rodas 1 e 2
xr1=xcirc+c1(1);
yr1=ycirc+c1(2);
xr2=xcirc+c2(1);
yr2=ycirc+c2(2);

%plota o carro
plot(xcarc,ycarc,'b',xr1,yr1,'k',xr2,yr2,'k',[c1(1),p1(1)],[c1(2),p1(2)],'g',[c2(1
),p2(1)],[c2(2),p2(2)],'g');axis equal

%[c1(1),c2(1)],[c1(2),c2(2)],'b',
%pinta o carro e as rodas
fill([xcarc,xcarc(1)],[ycarc,ycarc(1)],'c')
fill(xr1,yr1,'k')
fill(xr2,yr2,'k')

return
if sum(N>0)==1
   text(7,1.5,'Normal descolou!');
elseif sum(N==0)==2
   text(7,1.5,'Decolou voo!');
end
%configuracao do eixo
axis([(min(x_i)-r) (max(x)+r) (min(fx_i)-1) max(fx)+h+1]);
%limpa var.
clear xt yt gama xr1 xr2 yr1 yr2;
---------------------------------------------------------------------------------------
```

```
%Xp
%funcao que devolve vetor para calculo do estado em ode45
function dxdt=Xp(t,st_T)
global I m xc yc alfa Tab_n Tab_fat Tab_pot Tempo Vd

%e_t - vetor de estado no tempo t
%e_t=[xc yc alfa d_xc d_yc d_alfa];
xc=st_T(1);
yc=st_T(2);
alfa=st_T(3);
Tempo=[Tempo;t];
t
%controle PI da velocidade do veiculo
%[F_x,F_y,T_in]=control_PI(xc,yc,alfa,st_T(4),st_T(5),st_T(6));

%controle minimizando a potencia
[F_x,F_y,T_in]=control_pot_ODE(xc,yc,alfa,st_T(4),st_T(5),st_T(6));

%retorno
   dxdt=[st_T(4);st_T(5);st_T(6);F_x/m;F_y/m;T_in/I];
return
```

Em seguida segue os arquivos utilizados na simulação do modelo rígido.

```
% sim_ccontrol
%programa da pirncipal da simulacao do sistema robotico modelado como corpo
rigido
%fecha todas as figuras e limpa o workspace
close all
clear all
```

```
%******* Entrada dos dados do sistema ************************
dados_iniciais; %dados do veiculo,parametros da simulçao, tabelas de
%rmazenamento e estado inicial do sistema

%define velocidade desejada e constante de proporcionalidade Kp
Vd=0.5;                    %velocidade desejada em m/s
Kp=mi*9.8*sin(pi/6)/abs(Vd);  %constante para o controle proposto


%******************* Simulaçao *********************
%---------------loop principal---------------
while t<=tmax

   calc_param          %calcula os parametros dinamicos do sistema

   control_Vd_Pot      %calcula as Fat's que devem ser aplicadas as rodas do robo

   if isempty(Fat)     %caso o vetor das Fat seja vazio, a simulaçao e finalizada
      return
   end

   %armazenamento dos dados da simulaçao
   Tab_x=[Tab_x;X(1)];
   Tab_N=[Tab_N;N'];
   Tab_Fat=[Tab_Fat;Fat'];
   Tab_Vd_pot=[Tab_Vd_pot;V0,(abs(Fat(1))/abs(cos(gama1-
alfa))+abs(Fat(2))/abs(cos(gama2-alfa)))*abs(V0)];

   %calcula o novo estado
   novo_estado     %calcula aceleracoes, X, dX e os centros das rodas

   %ajusta veiculo ao solo
    [erro,alfa]=busca_alfa(X(1),X(2),X(3));       %funçao que devolve o erro em
%y(erro) e o ang. alfa(ja ajustado)
```

```
    X(2)=X(2)+erro;                    %corrige a coord. y do CM
    X(3)=alfa;                         %corrige o ang. alfa
    yc=X(2);                           %atualiza yc
    T_alfa=[cos(alfa),-sin(alfa);sin(alfa),cos(alfa)];
    c1=[xc;yc]+T_alfa*[-L/2;-h];       %calcula o centro da roda 1
    c2=[xc;yc]+T_alfa*[L/2;-h];        %calcula o centro da roda 2

    %atualiza a var. t
    t=t+dt;

  %visualizacao da simulacao
  if (rem(cont,10)==0) %imprime a cada 10 passos
      plota_figura;   %plota o veiculo e o terreno
      pause(.1);clf;  %aguarda .1s e limpa a figura
   end
   cont=cont+1;       %atualiza var. de controle

end
 %----------  fim do loop    --------------------

view_control_2D        %plota os graficos referentes aos dados da simulacao
salva_dados_rigid      %salva os dados da simulacao

clear all              %limpa o workspace
-------------------------------------------------------------------------------------------

% calc_param
%procedimento que calcula os parametros do passo k para a realizacao dos
%calculos dinamicos
%---------------------------------------------
%parametros geometricos do robo e do terreno
     %obtencao dos ang. e dos pontos de contato
   [gama1,x1,y1]=contact_angle(c1(1));%angulo de contato, coord. x e y da roda1
```

```
[gama2,x2,y2]=contact_angle(c2(1));%angulo de contato, coord. x e y da roda2

if isempty(gama1)|isempty(gama2)%se algum angulo de contato e vazio,
%ocorreu erro
    disp('Erro angulo de contato');
    Fat=[];                 %devolve o vetor das Fat's vazio
    return
end

%parametros do perfil do terreno
[ang,x1_dx,y1_dx]=contact_angle(c1(1)+dx);     %(x1_dx,y1_dx)-coord.     do
%perfil do terreno para x=xc1+dx
[ang,x2_dx,y2_dx]=contact_angle(c2(1)+dx);     %(x2_dx,y2_dx)-coord.     do
%perfil do terreno para x=xc2+dx

[ang,x1_dx_,y1_dx_]=contact_angle(c1(1)-dx);  %(x1_dx_,y1_dx_)-coord.  do
%perfil do terreno para x=xc1-dx
[ang,x2_dx_,y2_dx_]=contact_angle(c2(1)-dx);  %(x2_dx_,y2_dx_)-coord.  do
%perfil do terreno para x=xc2-dx

%primeira derivada espacial
df1=(y1_dx-y1)/dx;    %no ponto de contato 1
df2=(y2_dx-y2)/dx;    %no ponto de contato 2

%segunda derivada espacial
 d2f1=(y1_dx-2*y1+y1_dx_)/dx^2; %no pt de contato 1
 d2f2=(y2_dx-2*y2+y2_dx_)/dx^2; %no pt de contato 2

%parametros geometricos e de contato
 q1=(yc-y1)*sin(gama1)+(xc-x1)*cos(gama1);       %braço de alavanca de N1
%em relaçao ao CM
 q2=(yc-y2)*sin(gama2)+(xc-x2)*cos(gama2);       %braço de alavanca de N2
%em relaçao ao CM
```

```
    n1=(yc-y1)*cos(gama1)-(xc-x1)*sin(gama1);       %braço de alavanca de Fat1
%em relaçao ao CM
    n2=(yc-y2)*cos(gama2)-(xc-x2)*sin(gama2);       %braço de alavanca de Fat2
%em relaçao ao CM
   clear x1 x1_dx x1_dx_ x2 x2_dx x2_dx_ y1 y1_dx y1_dx_ y2 y2_dx y2_dx_
  %----------parametros das equaçoes de restricoes------
 E1=T*(df1*sin(alfa+teta)+cos(alfa+teta));      %coef. da aceleraçao angular na eq.
%de restriçao da roda 1
 E2=T*(df2*cos(alfa+delta)-sin(alfa+delta));      %coef. da aceleraçao angular na
%eq. de restriçao da roda 2
 F1=-d2f1*dX(1)^2+T*(sin(alfa+teta)-df1*cos(alfa+teta)-
d2f1*T*sin(alfa+teta)^2)*dX(3)^2-...
2*d2f1*T*sin(alfa+teta)*dX(1)*dX(3);%termo independente da eq. de restriçao
%da roda 1

 F2=-d2f2*dX(1)^2+T*(cos(alfa+delta)+df2*sin(alfa+delta)-
d2f2*T*cos(alfa+delta)^2)*dX(3)^2-...
  -2*d2f2*T*cos(alfa+delta)*dX(1)*dX(3);  %termo independente da eq. de
%restriçao da roda 1
```

```
% control_Vd_Pot
%arquivo que implementa o controle desenvolvido para o modelo de corpo rigido
%e que devolve as Fat's que devem ser obtidas pelo sistema
%verifica se ocorreu algum erro na simulaçao
if isempty(Fat)
   return
end

%******   Calculo do dominio D das Fat's  ***********
clear A u G B U
%declaracao das matrizes para as equaçoes dinamicas do sistema
M_1=diag([1/m,1/m,1/I]);                          %matriz de inercia invertida
```

```
A=[sin(gama1),sin(gama2);-cos(gama1),-cos(gama2);q1,q2];     %matriz dos coef.
%das normais
B=[cos(gama1),cos(gama2);sin(gama1),sin(gama2);n1,n2];       %matriz dos coef.
%das Fat's
u_=[0;-P;0];                                    %vetor dos termos independentes

%matrizes das equaçoes de restriçao
 if abs(gama1-gama2)>1e-6 %angulos de contato tido como diferentes
    C=[tan(gama1),-1,E1;tan(gama2),-1,E2];      %matriz de coef. das aceleraçoes
%generalizadas
    F_=[F1;F2];                          %vetor dos termos independentes

 else  %angulos de contato tidos como iguais
    C=[tan(gama1),-1,E1;0,0,1];
    F_=[F1;0];
 end

 %calculo das matrizes que relacionam as normais com as Fat's
 %N=G*Fat+U_
 inv_CM_1A=inv(C*M_1*A);
 U_=inv_CM_1A*(C*M_1*u_-F_);
 G=inv_CM_1A*(C*M_1*B);

 %geracao dos parametros da velocidade
 Delta=M_1*(B-A*G);
 S_=M_1*(u_-A*U_);

  pi_1=Delta(1,1)*cos(alfa)+Delta(2,1)*sin(alfa);  %coef. de Fat1

  pi_2=Delta(1,2)*cos(alfa)+Delta(2,2)*sin(alfa);  %coef. de Fat2

  a_0=S_(1)*cos(alfa)+S_(2)*sin(alfa);                %termo que contem forças de
%campo, superficiais, etc agindo no sistema
```

```
V0=dX(1)*cos(alfa)+dX(2)*sin(alfa);    %veloc. do robo no instante analisado


%********** Geracao dos vetores para encontrar D'  **********
%coeficientes da normal 1
neta_1=G(1,1);      %coeficiente da fat1
neta_2=G(1,2);      %coeficiente da fat2
neta=U_(1);         %termo independente

%coeficientes da normal 2
delta_1=G(2,1);      %coeficiente da fat1
delta_2=G(2,2);      %coeficiente da fat2
delta=U_(2);         %termo independente

%a Fat2 sera funcao de Fat1 como segue:
%Fat2=n*Fat1+b
%com: n e b definidos abaixo
n=-pi_1/pi_2;
b=(Kp*(Vd-V0)-a_0)/pi_2;

%definicao do intervalo de Fat1
%U_maior- vetor dos coefic. de Fat1 na relacao de maior ou igual ao vetor
%W_maior
U_maior=[1;n;(1+mi*(neta_1+neta_2*n));(n+mi*(delta_1+delta_2*n));
      (neta_1+neta_2*n);(delta_1+delta_2*n)];

W_maior=[-Fsat;(-Fsat-b);-mi*(neta+neta_2*b);-(b+mi*(delta+delta_2*b));
   (0.01*P-neta-neta_2*b);(0.01*P-delta-delta_2*b)];

%U_menor- vetor dos coefic. de Fat1 na relacao de menor ou igual ao vetor
%W_menor
U_menor=[1;n;(1/mi-neta_1-neta_2*n);(n*(1/mi-delta_2)-delta_1)];

W_menor=[Fsat;(Fsat-b);(neta_2*b+neta);(delta+delta_2*b-b/mi)];
```

```
%Calculo dos vetores
%u_maior=termos maiores ou igual
%u_menor=termos menores ou igual
u_menor=[];
u_maior=[];

for i=1:size(U_maior,1)
   if U_maior(i)>0
      u_maior=[u_maior;W_maior(i)/U_maior(i)];
   elseif U_maior(i)<0
      u_menor=[u_menor;W_maior(i)/U_maior(i)];
   end
end

for i=1:size(U_menor,1)
   if U_menor(i)>0
      u_menor=[u_menor;W_menor(i)/U_menor(i)];
   elseif U_menor(i)<0
      u_maior=[u_maior;W_menor(i)/U_menor(i)];
   end
end

%********** Geracao dos intervalos I1 e I2 **********
%I1 - intervalo das possiveis força de atrito da roda 1(Fat1)
%I2 - intervalo das possiveis força de atrito da roda 2(Fat2)
%os intervalos I1 e I2 sao calculados tendo como hipotese que Vd pode ser
%obtida pelo controle
I1=[max(u_maior);min(u_menor)];     %configura I1
%calcula I2 correspondente
if n>0
     I2=[n*I1(1)+b;n*I1(2)+b];
else
     I2=[n*I1(2)+b;n*I1(1)+b];
end %fim ajuste I2
```

```
clear U_maior U_menor W_maior W_menor u_maior u_menor b

% Verificacao de I1 e I2
%Se I1 e I2 sao validos o controle pode controlar Vd e tambem pode minimizar o
%consumo de potencia,
%em caso negativo segue o calculo das Fat's para maximizar a tracao do veiculo
%de forma a obter uma velocidade o  proxima possivel de Vd
if (I1(2)<I1(1))|(I2(2)<I2(1))  %intervalos nao validos
  I1=[];I2=[];          %configura I1 e I2 como vazios
  flag=flag+1;          %flag que sinaliza a geraçao ou nao das malhas das Fat's

  calc_Fat_veloc        %calcula as Fat's que maximizam a traçao do robo

  if ~isempty(Fat)   %verifica Fat e valida
     %em caso afirmativo calcula N e retorna para sim_ccontrol
     N=[neta_1*Fat(1)+neta_2*Fat(2)+neta;
          delta_1*Fat(1)+delta_2*Fat(2)+delta];
     return
  else
     %em caso negativo retorna para sim_ccontrol sem calcular N
      return
  end   %fim  verificacao do vetor Fat
else
    %I1 e I2 sao validos e o programa buscara as Fat's que minimizam a potencia

    %Geraçao do Fat's otimas
     nabla_1=1/abs(cos(gama1-alfa));       %coef. de Fat1 na equaçao da potencia
     nabla_2=1/abs(cos(gama2-alfa));       %coef. de Fat2 na equaçao da potencia

     %toma os pontos de fronteira como candidatos a ponto de otimo
     if n>0
          Fats_otimas=[I1(1),I2(1);I1(2),I2(2)];    %estrutura que armazena os pts
%candidatos a pt otimo
```

```
  else
        Fats_otimas=[I1(1),I2(2);I1(2),I2(1)];
    end

    %armazena possiveis ptos de minima potencia
    %para atuaçao somente da roda 1
    if ((Kp*(Vd-V0)-a_0)/pi_1>=I1(1))&((Kp*(Vd-V0)-a_0)/pi_1<=I1(2))
        Fats_otimas=[Fats_otimas;(Kp*(Vd-V0)-a_0)/pi_1,0];
     end

    %para atuaçao somente da roda 2
 if                    ((Kp*(Vd-V0)-a_0)/pi_2>=I2(1))&((Kp*(Vd-V0)-a_0)/pi_2
<=I2(2))%(I1(1)<=0)&(I1(2)>=0)
         Fats_otimas=[Fats_otimas;0,(Kp*(Vd-V0)-a_0)/pi_2];
    end
    clear n

   %calcula os valores otimos das Fat's e suas respectivas normais
   if abs(gama1-gama2)>1e-6 %caso os angulos de contato sejam diferentes
     %procura-se em Fats_otimas o par que minimiza a potencia
      P_min=nabla_1*abs(Fats_otimas(1,1))+nabla_2*abs(Fats_otimas(1,2));
      idx_otimo=1;
      for i=2:size(Fats_otimas,1)
          if
((nabla_1*abs(Fats_otimas(i,1))+nabla_2*abs(Fats_otimas(i,2))))<=P_min
             P_min=nabla_1*abs(Fats_otimas(i,1))+nabla_2*abs(Fats_otimas(i,2));
             idx_otimo=i;
          end  %fim verificacao de potencia minima
      end %fim do loop para se encontrar P_min

      %filtragem dos dados
      %verifica se ha mais de um pt em Fats_otimas que minimiza a potencia
%dentro de uma tolerancia=1.5e-6
```

```
id_ajust=find(abs(nabla_1*abs(Fats_otimas(:,1))+nabla_2*abs(Fats_otimas(:,2))-
P_min)<1.5e-6);
       if isempty(id_ajust)|size(id_ajust)==1
            Fat=[Fats_otimas(idx_otimo,1);Fats_otimas(idx_otimo,2)];
        else
         [v,id]=min(sqrt((Fats_otimas(id_ajust,1)-
Fat(1)).^2+(Fats_otimas(id_ajust,2)-Fat(2)).^2));
         Fat=[Fats_otimas(id_ajust(id),1);Fats_otimas(id_ajust(id),2)];
         clear v id
       end %fim do if
       id_ajust=[];

     else %angulo tidos como iguais
        %nesse caso todos os ptos do segmento que une os pontos extremos
        %de D' minimizam a potencia, assim o controle toma o pto medio
        %desse segmento para atuar o sistema
         Fat=[mean(I1);mean(I2)];
     end %fim busca da Fat otima

    %calcula as normais
     N=[neta_1*Fat(1)+neta_2*Fat(2)+neta;
       delta_1*Fat(1)+delta_2*Fat(2)+delta];

    %verifica se as normais sao validas
     if sum(N<0)>0
      Fat=[];
     end
     return %retorna para sim_ccontrol
 end %fim do if para o calculo de Fat e N
 ----------------------------------------------------------------------------------------------
```

```
% calc_Fat_veloc
if flag<=1
   intervalo_Fats=linspace(-Fsat,Fsat,100);
   [Fat1,Fat2]=meshgrid(intervalo_Fats,intervalo_Fats);
end

%calculo das normais
N1=neta_1*Fat1+neta_2*Fat2+neta;
N2=delta_1*Fat1+delta_2*Fat2+delta;

%******* Verificacao das normais positivas  ***********
id1_pos=find(N1>0);
id2_pos=find(N2>0);

id_pos=intersect(id1_pos,id2_pos);
if isempty(id_pos)
   disp('Nenhuma Fat encontrada.Normal negativa.');
   Fat=[];
   return
end

Fat1_pos=Fat1(id_pos);
Fat2_pos=Fat2(id_pos);
N1_pos=N1(id_pos);
N2_pos=N2(id_pos);
%********** verificacao de nao deslizamento **********
id1_slip=find(abs(Fat1_pos)./N1_pos<=mi);
id2_slip=find(abs(Fat2_pos)./N2_pos<=mi);
id_slip=intersect(id1_slip,id2_slip);
if isempty(id_slip)
   disp('Nenhuma Fat encontrada.Deslizamento');
   Fat=[];
   return
end
```

```
Fat1_slip=Fat1_pos(id_slip);
Fat2_slip=Fat2_pos(id_slip);
N1_slip=N1_pos(id_slip);
N2_slip=N2_pos(id_slip);
clear Fat1_pos Fat2_pos N1_pos N2_pos
%********** Geraçao do Fat's otimas  **************
%funcao objetivo
F_obj=abs(Vd-V0-(pi_1*Fat1_slip+pi_2*Fat2_slip+a_0)./Kp); %funcao objetivo,
%a qual deseja-se otimizar
[v,id_result]=min(F_obj);

Fat=[Fat1_slip(id_result);Fat2_slip(id_result)];

clear v id_result Fat1_slip Fat2_slip N1_slip N2_slip
-----------------------------------------------------------------------------------------------

% novo_estado
%arquivo que calcula as aceleracoes para atualizar o estado do sistema
%atraves do calculo de X e dX do passo K+1

%geraçao das matrizes utilizadas no calculo das aceleraçoes generalizadas
u=[Fat(1)*cos(gama1)+Fat(2)*cos(gama2);Fat(1)*sin(gama1)+Fat(2)*sin(gama2)
-P;n1*Fat(1)+n2*Fat(2)];
D=[-sin(gama1),-sin(gama2);cos(gama1),cos(gama2);-q1,-q2];
M_1=[1/m,0,0;0,1/m,0;0,0,1/I];
%aceleracao d2X
d2X=M_1*(u+D*N);
%********** ajusta estado ******************
%calculo da posicao do centro de massa no passo k+1 e das velocidades
%generalizadas
 xc=X(1)+dX(1)*dt+d2X(1)*(dt^2)/2;
yc=X(2)+dX(2)*dt+d2X(2)*(dt^2)/2;
```

```
 if abs(gama1-gama2)>1e-7
    alfa=X(3)+dX(3)*dt+d2X(3)*(dt^2)/2;
    dX=dX+d2X*dt;
else
    alfa=X(3);
    dX=[dX(1)+d2X(1)*dt;dX(2)+d2X(2)*dt;dX(3)];
end
 X=[xc;yc;alfa];

%calculo dos centros das rodas
 T_alfa=[cos(alfa),-sin(alfa);sin(alfa),cos(alfa)]; %matriz de rotaçao
 c1=[xc;yc]+T_alfa*[-L/2;-h];
 c2=[xc;yc]+T_alfa*[L/2;-h];
```

```
%busca_alfa
%[erro,alfa_min]=busca_alfa(xc,yc,init)
%programa que devolve o angulo(alfa_min) da inclinacao do veiculo que
%minimiza o erro(erro) entre o centro das rodas e a curva de centros, dados
%as coord. do centro de massa do veiculo [xc,yc] e um chute inicial para
%alfa [init]
function [erro,alfa_min]=busca_alfa(xc,yc,init)
%programa faz a varredura nos dois sentidos
%alfa1 procura a esquerda de init e alfa2 a direita
    alfa1=init; %inicia alfa1
    alfa2=init; %inicia alfa2

    [e1,erro]=erro_h(xc,yc,init);    %funcao que devolve a diferenca entre os erros
%dos centros das rodas e a curva de centro(e1) e o erro medio(erro)

    if (e1~=0)
        di=pi/180;      %tamanho do passo de busca
        alfa_ini=init;
        alfa_fim=init;
```

```
fim_procura=0;
%inicia busca
while ((alfa1>-0.99*pi/2)|(alfa2<0.99*pi/2))&(fim_procura==0)
   alfa1=alfa1-di;
   alfa2=alfa2+di;
   if (alfa1<-0.99*pi/2)
      alfa1=-0.99*pi/2;
   end
   if (alfa2>0.99*pi/2)
      alfa2=0.99*pi/2;
   end
   if sign(e1)~=sign(erro_h(xc,yc,alfa1))
      alfa_ini = alfa1+di;
      alfa_fim = alfa1;
      sinal=-1;
      fim_procura=1;
   elseif sign(e1)~=sign(erro_h(xc,yc,alfa2))
      alfa_ini = alfa2-di;
      alfa_fim = alfa2;
      sinal=1;
      fim_procura=1;
   end
end
%refina a busca
if (fim_procura==1)
   for di=[pi/1800 pi/18000] %inclua pi/180000, etc para aumentar a precisão
      [e1,erro]=erro_h(xc,yc,alfa_ini);
      for alfa1=[alfa_ini+di*sinal:di*sinal:alfa_fim]
         if (sign(e1)~=sign(erro_h(xc,yc,alfa1)))
            alfa_ini = alfa1-di*sinal;
            alfa_fim = alfa1;
            break
         end
      end
```

```
        end
        [e1,erro]=erro_h(xc,yc,alfa_ini);
        [e2,erro]=erro_h(xc,yc,alfa_fim);
        if (e1~=e2)
           alfa_min = alfa_ini + (alfa_fim-alfa_ini)*abs(e1/(e1-e2));
        else
           alfa_min = alfa_ini;
        end
        [e1,erro]=erro_h(xc,yc,alfa_min);
     else
        alfa_min=init;
     end
  else
     alfa_min=init;
  end
```

```
%erro_h
%[d_e,err]=erro_h(xc,yc,alfa)
%funcao que devolve a diferenca entre os erros da roda1
%e da roda2[d_e], e tambem o erro medio[err] dados:
%[xc,yc]-coord. cartesianas do centro de massa do veiculo
%alfa- angulo de inclinaçao do veiculo em relaçao ao eixo dos x
function [d_e,err]=erro_h(xc,yc,alfa)
global L1 L2 h1 h2
cm=[xc;yc];         %ccord. do centro de massa
T_alfa=[cos(alfa),-sin(alfa);sin(alfa),cos(alfa)];  %matriz de rotaçao de um angulo
%alfa em relacao ao eixo Z
P1=cm+T_alfa*[-L1;-h1];     %coord. do centro da roda 1
P2=cm+T_alfa*[L2;-h2];      %coord. do centro da roda 2
f1=valor_gx(P1(1));    %valor de y na curva de centro para o x do centro da roda 1
f2=valor_gx(P2(1));    %valor de y na curva de centro para o x do centro da roda 2
e_1=f1-P1(2);              %erro em y do centro da roda 1 para a curva de centros
e_2=f2-P2(2);              %erro em y do centro da roda 2 para a curva de centros
```

```
d_e=e_2-e_1;       %diferenca entre o erros da roda 1 e da roda 2, respectivamente
err=0.5*(e_1+e_2);          %erro medio
return
```

%**plota_figura**

```
%programa que plota o carro e o perfil
global contador

%plotagem das figuras
%perfil do terreno e geracao dos circulos das rodas
npts=500;       %numero de ptos que serao utilizados para plotar o perfil do terreno
contador=1;     %var. que controla impressao do perfil do terreno

if contador==1
    %coord. do terreno que serao plotadas
    xperfil=x(fix(linspace(1,size(x,1),npts)));
    yperfil=fx(fix(linspace(1,size(fx,1),npts)));
     %circunferencia das rodas
    gama=[0:pi/10:2*pi];
    xcirc=r*cos(gama);
    ycirc=r*sin(gama);
     %ajusta var.
    contador=2;
     clear gama;
end

%plota perfil do terreno
plot(xperfil,yperfil);
hold on;
```

```
%geracao da carcaca
xm=0.5*(c1(1)+c2(1));
ym=0.5*(c1(2)+c2(2));
gama=[alfa:pi/20:(alfa+pi)];
xcarc=(L/2)*cos(gama)+xm;
ycarc=(L/2)*sin(gama)+ym;
clear gama xm ym

%rodas
%Geracao das rodas 1 e 2
xr1=xcirc+c1(1);
yr1=ycirc+c1(2);
xr2=xcirc+c2(1);
yr2=ycirc+c2(2);

%plota o carro
plot(xcarc,ycarc,'b',[c1(1),c2(1)],[c1(2),c2(2)],'b',xr1,yr1,'k',xr2,yr2,'k');

%pinta o carro e as rodas
fill([xcarc,c1(1),c2(1)],[ycarc,c1(2),c2(2)],'c')
fill(xr1,yr1,'k')
fill(xr2,yr2,'k')

%configuracao do eixo
axis([(min(x_i)-r) (max(x)+r) (min(fx_i)-1) max(fx)+h+1]);

%limpa var.
clear xt yt gama xr1 xr2 yr1 yr2;
```

```
%view_control_2D
%arquivo que plota os graficos do controle 2D do veiculo como corpo rigido
%vetor tempo
t=linspace(0,t,size(Tab_N,1));

%plota as normais
figure(2);
plot(t,Tab_N(:,1),t,Tab_N(:,2)),legend('N_1','N_2'),xlabel('Tempo[s]'),
ylabel('[N]');

%plota as Fat's
figure(3);
plot(t,Tab_Fat(:,1),t,Tab_Fat(:,2)),legend('F_1','F_2'),xlabel('Tempo[s]'),
ylabel('[N]');

%plota as razoes Fat/N
figure(4);
plot(t,0*t+mi,'r',t,abs(Tab_Fat(:,1))./Tab_N(:,1),t,abs(Tab_Fat(:,2))./Tab_N(:,2)),
legend('mi','Fat1/N1','Fat2/N2'),xlabel('Tempo[s]'),

%plota as velocidades: real e desejada
figure(5)
plot(t,Tab_Vd_pot(:,1),t,t*0+Vd),title('Grafico das Velocidades do CM');
legend('velocidade real','velocidade desejada');xlabel('Tempo[s]');ylabel('[m/s]')

%plota a potencia
figure(6);
plot(t,Tab_Vd_pot(:,2)),title('Grafico da Potencia consumida'),
xlabel('Tempo[s]'),ylabel('[W]');
```

```
% salva_dados
%arquivo que salva dados em arquivo texto (.txt)
echo off
%obtencao dos dados para serem salvos
t=0:dt:(tmax); %geracao do vetor de tempo
%definicao da tabela de dados Tab_coord e Tab_veloc
Tab_coord=[Tab_x,Tab_N,Tab_Fat,Tab_Vd_pot(:,2)];
Tab_veloc=[t',Tab_Vd_pot(:,1)];
%pergunta se deseja salvar os dados
resp=input('Deseja salvar esses dados ([1]-sim,[0]-nao):');
if resp==1
   %entrada do nome do arquivo pelo usuario
   nome=input('Digite o nome do perfil do terreno e o tempo de simulaçao com
dois digitos [terreno_tempo]:','s');
   %confirmacao do nome do arquivo
   conf=0;
   nome
   conf=input('Confirme nome ([1]-sim,[0]-nao):');
   nome_arq=strcat('..\dados_simula\sim_',nome);
   if conf==1
      nome_arq1=strcat(nome_arq,'.txt');
      nome_arq2=strcat(nome_arq,'_veloc.txt');
      save(nome_arq1,'Tab_coord','-ASCII','-TABS');
      save(nome_arq2,'Tab_veloc','-ASCII','-TABS');
      disp('Arquivos salvo como:'), nome_arq1,nome_arq2
   else
      disp('Erro na execuçao do programa. Finalizando...')
      pause(1)
       beep
   end
else
   disp('Finalizando Programa...')
end
```
