6
Referências Bibliográficas

1. Speranza Neto, M., Barreto, A.R., da Silva, F.R., Pedroza, B.C., “A Reconstituição de Acidentes Através de uma Abordagem Adequada”, SAE Paper No. 2003-1103-2282, Congresso SAE BRASIL, 2003.
2. Abdulmassih, D.S., “Modelos de Veículos Rígidos para Análise de Colisões e Reconstituição de Acidentes”, Dissertação de Mestrado, DEM/PUC-Rio, 2003.
3. Macmillan, R.H., “Dynamics of Vehicle Collisions”, Inderscience Enterprises Ltd., 1983.
4. Fox, R.L, “Optimization Methods for Engineering Design”, Addison-Wesley Publishing Company, 1973.
5. Genta, G., “Motor Vehicle Dynamics Modeling and Simulation”, World Scientific, 1997.
6. Kost, G. and Werner, S.M., “Use of Monte Carlo Simulation Techniques in Accident Reconstruction”, SAE Paper No. 940719, Society of Automotive Engineers, 1994.
7. Moser, A. and Steffan, H., “Automatic Optimization of Pre-Impact Parameters Using Post Impact Trajectories and Rest Positions”, SAE Paper No. 9803373, Society of Automotive Engineers, 1998.
8. Pohlheim, H. and Hunt, K.J., “Control of Lateral Vehicle Dynamics and Dynamic Optimization using Genetic Algorithm Toolbox”, http://www.pohlheim.com/Papers/vehicle_gal95/gal1_1.html
9. Day, T. D. e Metz, L. D. - “The Simulation for Driver Inputs Using a Vehicle Driver Model” - Artigo SAE No. 2000-01-1313 - SAE 2000 World Congress – Detroit – Michigan – 2000.
10. Allen, R. W.; Rosenthal T. J. e Hogue J . R. – “Modeling and Simulation of Driver/Vehicle Interaction” – Artigo SAE No. 960177 – 1996.
11. Genetic Algorithm and Direct Search Toolbox User’s Guide, Version 1 The MathWorks, Inc., 2005.
12. Notas de Aula da disciplina “Computação Evolucionária” ministrada na PUC-Rio em 2005.
13. Winston, W. L. Operations Research: Applications and Algorithms, Duxbury Press, Boston, 1987.
14. Speranza Neto, M. e Spínola, A. L., “Análise do Comportamento Dinâmico de um Veículo em uma Trajetória Pré-Definida através de um Modelo Cinemático em Malha Fechada”, Artigo SAE Brasil 2005.

7
Apêndices – Programas Matlab

Encontram-se nestes anexos os códigos fontes utilizados para:

- o(s) modelo(s) do(s) veículo(s)
- a matriz de transformação das condições pré-choque para as condições pós-choque
- a simulação da trajetória pós-choque
- a otimização.

7.1. Uma Aplicação Trivial

```
function z = my_fun(x);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% as massas foram assumidas iguais a 1
% o choque foi assumido perfeitamente elástico
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% índice i indica veículo i
% índice a indica instante imediatamente anterior ao choque
% índice p indica instante imediatamente posterior ao choque
% índice t indica instante posterior ao choque quando as vel. são nulas
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% V1a^2 + V2a^2 = V1p^2 + V2p^2 (conservação de energia cinética)
% V1a + V2a = V1p + V2p        (conservação de quantidade de movimento)
% V1p = V2a
% V2p = V1a
% Sit = Sip + Vip*T - Ai*T^2/2
% Vit = 0 = Vip - Ai*T --> T = Vip/Ai
% Sit = Sip + Vip*(Vip/Ai) - Ai*(Vip/Ai)^2/2 = Sip + Vip^2/(2*Ai)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Via --> V1xa = -V2xa = V1ya = 10; V2ya = 0  (esperado)
% Sip --> Xip = Yip = 10
% Ai  --> A1x= A1y = -5; A2x = A2y = 10
% Sit --> X1t = 0; X2t = 15; Y1t = 10; Y2t = 15
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% x(1)=V1xa (esperado = 10)
% x(2)=V2xa (esperado = -10)
% x(3)=V1ya (esperado = 10)
% x(4)=V2ya (esperado = 0)
% a =V1xp
% b =V2xp
% c =V1yp
% d =V2yp
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% V1p = V2a:
a = x(2);
c = x(4);
% V2p = V1a:
b = x(1);
d = x(3);
% Sit = Sip + Vip^2/(2*Ai):
e = 0 - 10 - a^2/(-10);
f = 15 - 10 - b^2/20;
g = 10 - 10 - c^2/(-10);
h = 15 - 10 - d^2/20;
% Função a ser minimizada:
z = (e^2 + f^2 + g^2 + h^2)^0.5;
```

Modelo de Colisão e de Trajetória

```
options = gaoptimset ('PopInitRange', [5 -15 5 -5 ; 15 -5 15 5] , 'PopulationSize', 100 ,
        'EliteCount' , 2 , 'Generations' , 50 , 'StallGenLimit' , 5000 , 'StallTimeLimit' ,
        600000 , 'CrossoverFcn', @crossoverintermediate , 'PlotFcns' ,
        [@gaplotbestindiv , @gaplotbestf] ) ;
[x fval reason] = ga(@my_fun, 4, options);
```

Parâmetros e Configurações da Função GA

7.2.
Pós-Colisão para Um Veículo

A seguir tem-se o diagrama de blocos *Simulink/Matlab* para a simulação do modelo pós-choque desenvolvido.

Nas páginas seguintes encontram-se listadas as funções:

- *log2glob.m*, que realiza a transformação de coordenadas do referencial local para o global;
- *aceleracoes.m*, que calcula as acelerações instantâneas do veículo;
- *unitarios.m*, que determina as direções das velocidades instantâneas de cada pneu, nas quais as forças de atrito estão aplicadas, calcula os termos $u_{lJ,i}$ que as ponderam em cada eixo local e realiza seus somatórios.

Modelos e funções equivalentes a estes serão utilizados no caso de pós-colisão para dois veículos e no caso completo e encontram-se explícitas nos respectivos anexos.

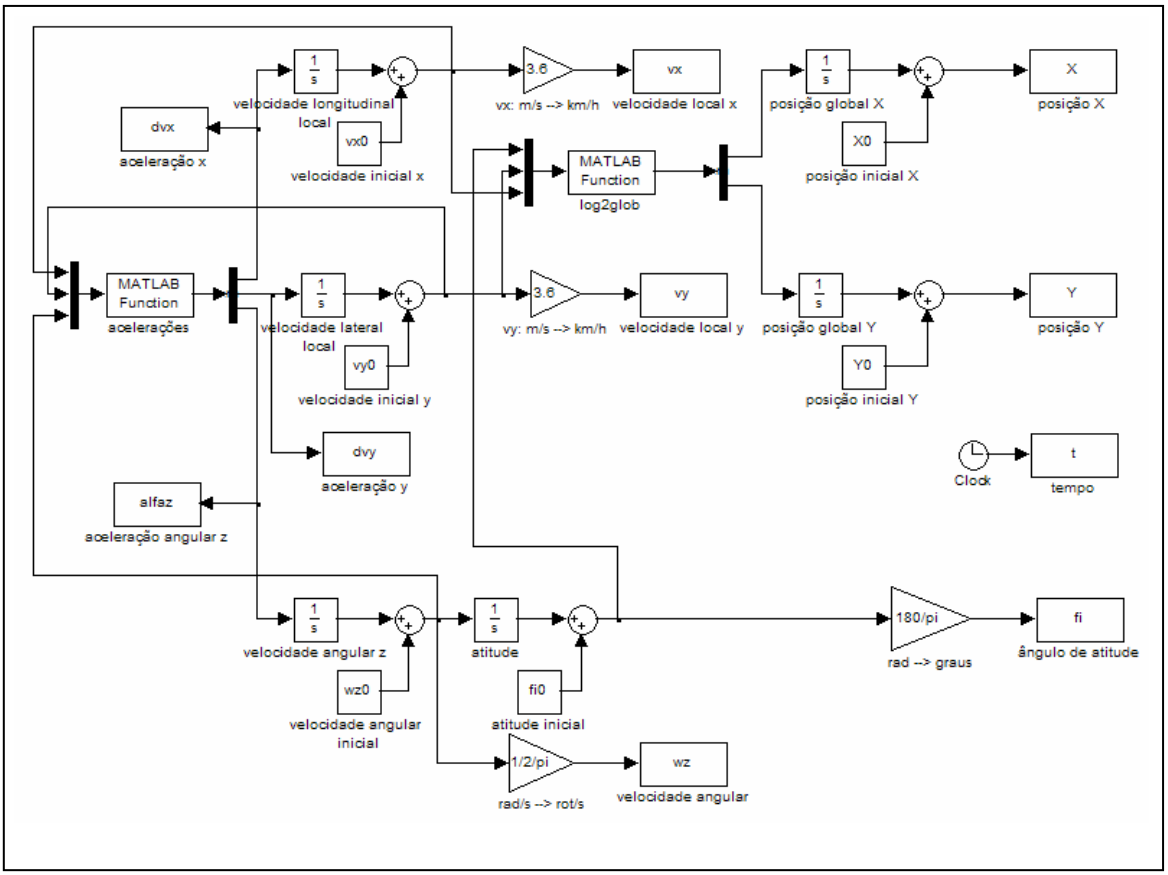

Diagrama de Blocos do Modelo Pós-Choque

```
function[V]=loc2glob(fi,vy,ux)

V(1,1)=ux*cos(fi)-vy*sin(fi);
V(2,1)=ux*sin(fi)+vy*cos(fi);
```

log2glob.m - Coordenadas Locais para Globais

```
function[V]=aceleracoes(vx,vy,wz)

global adx ady adz bax bay baz

[sux,suy,suz] = unitarios(vx,vy,wz);

err = 1e-2;

if abs(vx) >= err
   ax=-adx*sux-bax*vx;
else
   ax=0;
end

if abs(vy) >= err
  ay=-ady*suy-bay*vy;
else
  ay=0;
end

if abs(wz) >= err
  alfaz=-adz*suz-baz*wz;
else
  alfaz=0;
end

V(1,1)=ax+wz*vy;
V(2,1)=ay-wz*vx;
V(3,1)=alfaz;
```

aceleracoes.m - Acelerações Instantâneas

```
function [sx,sy,sz] = unitarios(u,v,w);

global bd bt ld lt

% uDD
vddx=u+w*bd/2;
vddy=v+w*ld;
vdd=sqrt(vddx^2+vddy^2);
uddx = vddx/vdd;
uddy = vddy/vdd;

% uDE
vdex=u-w*bd/2;
vdey=v+w*ld;
vde=sqrt(vdex^2+vdey^2);
udex = vdex/vde;
udey = vdey/vde;

% uTD
vtdx=u+w*bt/2;
vtdy=v-w*lt;
vtd=sqrt(vtdx^2+vtdy^2);
utdx = vtdx/vtd;
utdy = vtdy/vtd;

% uTE
vtex=u-w*bt/2;
vtey=v-w*lt;
vte=sqrt(vtex^2+vtey^2);
utex = vtex/vte;
utey = vtey/vte;

% SOMATÓRIO DAS COMPONENTES DOS UNITÄRIOS

% Ponderação das forças em X
sx = uddx + udex + utdx + utex;

% Ponderaçao das forças em Y
sy = uddy + udey + utdy + utey;

% Ponderaçao dos momentos em Z
sz =(uddx - udex)*bd/2 + (utdx - utex)*bt/2 + (uddy + udey)*ld - (utdy + utey)*lt;
```

unitarios.m - Direções das Velocidades Instantâneas de cada Pneu

```
global adx ady adz bax bay baz
global bd bt ld lt
global Xfc Yfc psifc
global ro Cx Cy Cmz S Jz m
global vx0 X0 vy0 Y0 wz0 psi0
global dt tf

Xfc=30;
Yfc=2;
psifc=335;

pista='pista';

g = 9.81;

m = 1000;
Jz = 2000;
rz2 = Jz/m;

ld = 1;
lt = 1.5;
bt = 1.5;
bd = 1.5;

mu = 0.7;

adx = mu*g/4;
ady = mu*g/4;
adz = mu*g/4/rz2;

ro = 1.2;
Cx = 0.30;
Cy = 0.80;
Cmz = 0.2;
S = 2.0;

tf = 3;

dt = 0.05;
```

Parâmetros do Veículo, da Interação e da Simulação

```
function [d]= fun_ga(var)

global adx ady adz bax bay baz
global bd bt ld lt
global Xfc Yfc psifc
global ro Cx Cy Cmz S Jz m
global vx0 X0 vy0 Y0 wz0 psi0
global dt tf

vx0= var(1);
X0 = var(2);

vy0= var(3);
Y0 = var(4);

wz0 = var(5);
psi0 = var(6)*pi/180;

bax = ro*Cx*S*abs(vx0)/2/m;
bay = ro*Cy*S*abs(vy0)/2/m;
baz = ro*Cmz*S*(ld+lt)*abs(wz0)/2/Jz;

sim pos_colisao

Xf=X(length(X));
Yf=Y(length(Y));
psif=psig(length(psig));

Xdd=Xf+ld*cos(psif*pi/180)+(bd/2)*sin(psif*pi/180);
Ydd=Yf+ld*sin(psif*pi/180)-(bd/2)*cos(psif*pi/180);
Xte=Xf-lt*cos(psif*pi/180)-(bt/2)*sin(psif*pi/180);
Yte=Yf-lt*sin(psif*pi/180)+(bt/2)*cos(psif*pi/180);

Xddc=Xfc+ld*cos(psifc*pi/180)+(bd/2)*sin(psifc*pi/180);
Yddc=Yfc+ld*sin(psifc*pi/180)-(bd/2)*cos(psifc*pi/180);
Xtec=Xfc-lt*cos(psifc*pi/180)-(bt/2)*sin(psifc*pi/180);
Ytec=Yfc-lt*sin(psifc*pi/180)+(bt/2)*cos(psifc*pi/180);

d=1000*(sqrt((Xdd-Xddc)^2+(Ydd-Yddc)^2+(Xte-Xtec)^2+(Yte-Ytec)^2));
```

Função de Avaliação do GA

```
dados_ga
options = gaoptimset ('PopInitRange', [5 15 -10 0 0 5 ; 15 25 0 10 10 15] ,
        'PopulationSize', 100 ,  'EliteCount' , 5 ,'CrossoverFraction' , 0.95 ,
        'Generations' , 40 , 'FitnessLimit' , 10.0 , 'StallGenLimit' , 10 ,
        'StallTimeLimit' , 1000 , 'CrossoverFcn', @crossoverintermediate,
        'MutationFcn' , {@mutationgaussian  [3]  [0.6500]} , 'PlotFcns' ,
        [@gaplotbestindiv , @gaplotbestf])
[x fval reason finalscores] = ga(@fun2_ga, 6, options)
```

Parâmetros e Configurações da Função GA

7.3.
Pós-Colisão para Dois Veículos

A seguir tem-se o diagrama de blocos *Simulink/Matlab* para a simulação do modelo desenvolvido. Na página seguinte, encontra-se o diagrama de blocos *Simulink/Matlab* referente bloco "VEÍCULO A". O bloco para o veículo B é inteiramente equivalente. As funções citadas nos dois diagramas, bem como todas as demais funções utilizadas neste caso, encontram-se após a apresentação dos mesmos.

Além de um segundo veículo no diagrama de blocos, também houve a inclusão da função *teste_de_parada_2.m*. Esta função foi utilizada para melhor determinar em que momento a simulação deve cessar. Através desta função evita-se a perda de tempo de computação para os casos onde o tempo de parada é menor que o estipulado (em geral, 3 segundos), evitando também a parada equivocada para os poucos casos que venham a exceder este mesmo tempo. As funções utilizadas para gerar a animação sempre utilizarão os dados referentes ao melhor indivíduo gerado pelo GA depois que este termina sua otimização.

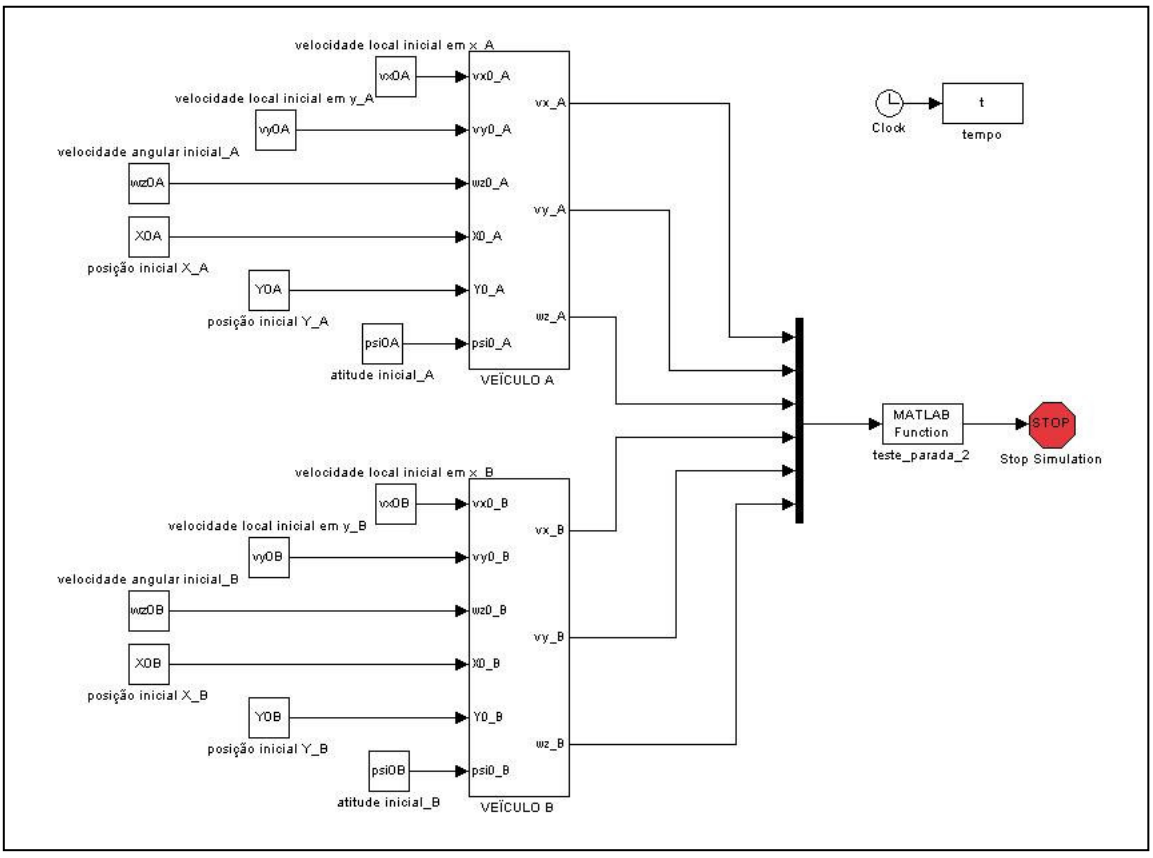

Diagrama de Blocos de Simulação dos Veículos na Condição Pós-Choque

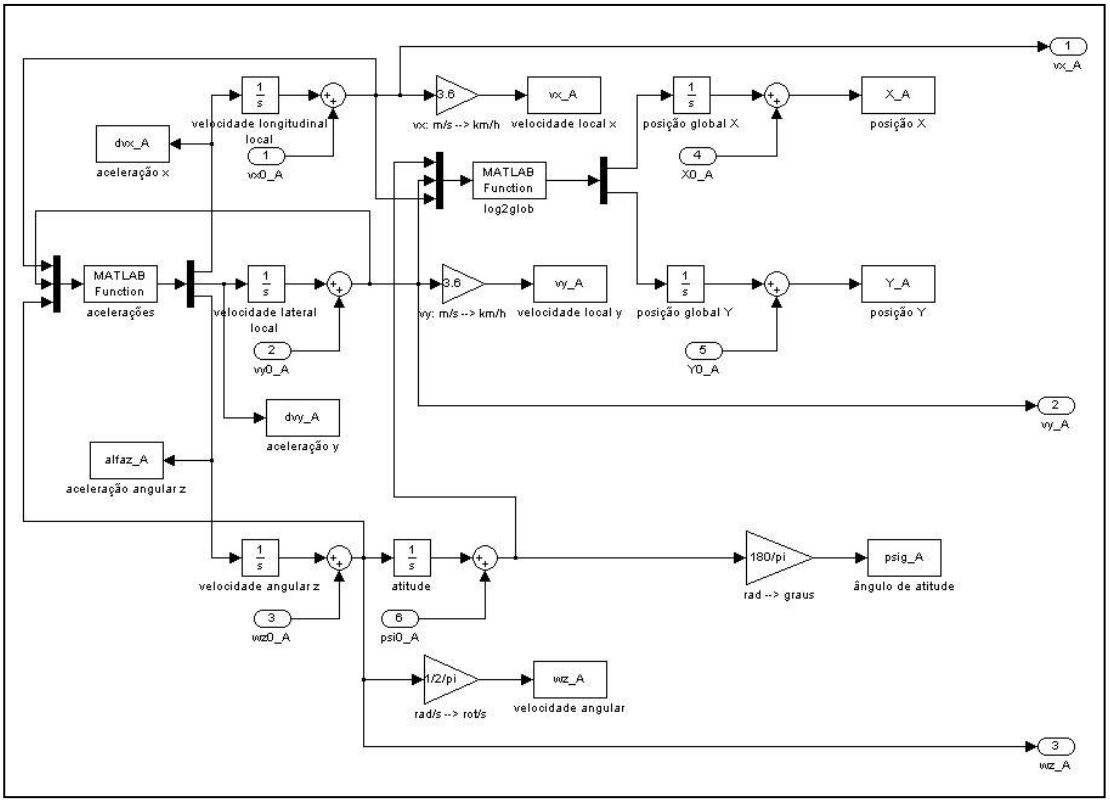

Diagrama de Blocos do Veículo A

```
% GLOBAL DO CARRO A %
global adxA adyA adzA baxA bayA bazA
global bdA btA ldA ltA
global X0cA Y0cA psi0cA XfcA YfcA psifcA
global CxA CyA CmzA SA JzA mA
global vx0A X0A vy0A Y0A wz0A psi0A

% GLOBAL DO CARRO B %
global adxB adyB adzB baxB bayB bazB
global bdB btB ldB ltB
global X0cB Y0cB psi0cB XfcB YfcB psifcB
global CxB CyB CmzB SB JzB mB
global vx0B X0B vy0B Y0B wz0B psi0B

% GLOBAL DE AMBOS %
global ro dt tf



pista='pista';
g = 9.81;
ro = 1.2;
dt = 0.05;
tf = 5;

% CARRO A%

% - GEOMETRIA INICIAL (IRÁ VARIAR ATRAVÉS DO GA)%
X0cA = 38.9211;
Y0cA = 1.0579;
psi0cA = 0.2099;

% - GEOMETRIA FINAL %
XfcA = 53.7802;
YfcA = -3.4036;
psifcA = 437.5954;

% - CARACTERÍSTICAS %
mA = 1000;
JzA = 2000;
rz2A = JzA/mA;

ldA = 1;
ltA = 1.5;
btA = 1.5;
bdA = 1.5;
```

Parâmetros dos Veículos, da Interação e da Simulação

```
muA = 0.7;

adxA = muA*g/4;
adyA = muA*g/4;
adzA = muA*g/4/rz2A;

CxA = 0.30;
CyA = 0.80;
CmzA = 0.2;
SA = 2.0;

% CARRO B%

% - GEOMETRIA INICIAL (IRÁ VARIAR ATRAVÉS DO GA)%
X0cB = 41.2513;
Y0cB = 2.8040;
psi0cB = 3.1416;

% - GEOMETRIA FINAL %
XfcB = 36.0249;
YfcB = 7.6644;
psifcB = 212.2860;

% - CARACTERÍSTICAS %
mB = 1000;
JzB = 2000;
rz2B = JzB/mB;

ldB = 1;
ltB = 1.5;
btB = 1.5;
bdB = 1.5;

muB = 0.7;

adxB = muB*g/4;
adyB = muB*g/4;
adzB = muB*g/4/rz2B;

CxB = 0.30;
CyB = 0.80;
CmzB = 0.2;
SB = 2.0;
```

Parâmetros dos Veículos, da Interação e da Simulação (continuação)

```
function [d]= fun_ga(var)

% GLOBAL DO CARRO A %
global adxA adyA adzA baxA bayA bazA
global bdA btA ldA ltA
global X0cA Y0cA psi0cA XfcA YfcA psifcA
global CxA CyA CmzA SA JzA mA
global vx0A X0A vy0A Y0A wz0A psi0A

% GLOBAL DO CARRO B %
global adxB adyB adzB baxB bayB bazB
global bdB btB ldB ltB
global X0cB Y0cB psi0cB XfcB YfcB psifcB
global CxB CyB CmzB SB JzB mB
global vx0B X0B vy0B Y0B wz0B psi0B

% GLOBAL DE AMBOS %
global ro dt tf

% VARIÁVEIS GLOBAIS %
X0 = var(1);
Y0 = var(2);

% VARIÁVEIS DO CARRO A %
vx0A = var(3);
X0A = X0cA + var(1);

vy0A = var(4);
Y0A = Y0cA + var(2);

wz0A = var(5);
psi0A = psi0cA;

baxA = ro*CxA*SA*abs(vx0A)/2/mA;
bayA = ro*CyA*SA*abs(vy0A)/2/mA;
bazA = ro*CmzA*SA*(ldA+ltA)*abs(wz0A)/2/JzA;

% VARIÁVEIS DO CARRO B %
vx0B = var(6);
X0B = X0cB + var(1);

vy0B = var(7);
Y0B = Y0cB + var(2);

wz0B = var(8);
psi0B = psi0cB;
```

Função de Avaliação

```
baxB = ro*CxB*SB*abs(vx0B)/2/mB;
bayB = ro*CyB*SB*abs(vy0B)/2/mB;
bazB = ro*CmzB*SB*(ldB+ltB)*abs(wz0B)/2/JzB;

% SIMULA AMBOS OS CARROS %
sim pos_colisao_2

% POSIÇÃO DO CARRO A %
XfA=X_A(length(X_A));
YfA=Y_A(length(Y_A));
psifA=psig_A(length(psig_A));

XddA=XfA+ldA*cos(psifA*pi/180)+(bdA/2)*sin(psifA*pi/180);
YddA=YfA+ldA*sin(psifA*pi/180)-(bdA/2)*cos(psifA*pi/180);
XteA=XfA-ltA*cos(psifA*pi/180)-(btA/2)*sin(psifA*pi/180);
YteA=YfA-ltA*sin(psifA*pi/180)+(btA/2)*cos(psifA*pi/180);

XddcA=XfcA+ldA*cos(psifcA*pi/180)+(bdA/2)*sin(psifcA*pi/180);
YddcA=YfcA+ldA*sin(psifcA*pi/180)-(bdA/2)*cos(psifcA*pi/180);
XtecA=XfcA-ltA*cos(psifcA*pi/180)-(btA/2)*sin(psifcA*pi/180);
YtecA=YfcA-ltA*sin(psifcA*pi/180)+(btA/2)*cos(psifcA*pi/180);

% POSIÇÃO DO CARRO B %
XfB=X_B(length(X_B));
YfB=Y_B(length(Y_B));
psifB=psig_B(length(psig_B));

XddB=XfB+ldB*cos(psifB*pi/180)+(bdB/2)*sin(psifB*pi/180);
YddB=YfB+ldB*sin(psifB*pi/180)-(bdB/2)*cos(psifB*pi/180);
XteB=XfB-ltB*cos(psifB*pi/180)-(btB/2)*sin(psifB*pi/180);
YteB=YfB-ltB*sin(psifB*pi/180)+(btB/2)*cos(psifB*pi/180);

XddcB=XfcB+ldB*cos(psifcB*pi/180)+(bdB/2)*sin(psifcB*pi/180);
YddcB=YfcB+ldB*sin(psifcB*pi/180)-(bdB/2)*cos(psifcB*pi/180);
XtecB=XfcB-ltB*cos(psifcB*pi/180)-(btB/2)*sin(psifcB*pi/180);
YtecB=YfcB-ltB*sin(psifcB*pi/180)+(btB/2)*cos(psifcB*pi/180);

% FUNÇÃO DE AVALIAÇÃO %
d = 1000 * sqrt ((XddA - XddcA)^2 + (YddA - YddcA)^2 +
                 (XteA - XtecA)^2 + (YteA - YtecA)^2 +
                 (XddB - XddcB)^2 + (YddB - YddcB)^2 +
                 (XteB - XtecB)^2 + (YteB - YtecB)^2);
```

Função de Avaliação (continuação)

```
function[V]=loc2glob(fi,vy,ux)

V(1,1)=ux*cos(fi)-vy*sin(fi);
V(2,1)=ux*sin(fi)+vy*cos(fi);
```

log2glob.m - Coordenadas Locais para Globais

```
% ACELERACOES NO REF. LOCAL

function[V]=aceleracoes_A(vx,vy,wz)

global adxA adyA adzA baxA bayA bazA

[sux,suy,suz] = unitarios_A(vx,vy,wz);

err = 1e-2;

if abs(vx) >= err
   ax=-adxA*sux-baxA*vx;
else
   ax=0;
end

if abs(vy) >= err
  ay=-adyA*suy-bayA*vy;
else
  ay=0;
end

if abs(wz) >= err
  alfaz=-adzA*suz-bazA*wz;
else
  alfaz=0;
end

V(1,1)=ax+wz*vy;
V(2,1)=ay-wz*vx;
V(3,1)=alfaz;
```

aceleracoes.m - Acelerações Instantâneas do Veículo A

```
% ACELERACOES NO REF. LOCAL

function[V]=aceleracoes_B(vx,vy,wz)

global adxB adyB adzB baxB bayB bazB

[sux,suy,suz] = unitarios_B(vx,vy,wz);

err = 1e-2;

if abs(vx) >= err
   ax=-adxB*sux-baxB*vx;
else
   ax=0;
end

if abs(vy) >= err
  ay=-adyB*suy-bayB*vy;
else
  ay=0;
end

if abs(wz) >= err
  alfaz=-adzB*suz-bazB*wz;
else
  alfaz=0;
end

V(1,1)=ax+wz*vy;
V(2,1)=ay-wz*vx;
V(3,1)=alfaz;
```

aceleracoes.m - Acelerações Instantâneas do Veículo B

```
% UNITÁRIOS DAS DIRECOES DAS VELOCIDADES DOS PNEUS

function [sx,sy,sz] = unitarios_A(u,v,w);

global bdA btA ldA ltA

% uDD
vddx=u+w*bdA/2;
vddy=v+w*ldA;
vdd=sqrt(vddx^2+vddy^2);

uddx = vddx/vdd;
uddy = vddy/vdd;

% uDE
vdex=u-w*bdA/2;
vdey=v+w*ldA;
vde=sqrt(vdex^2+vdey^2);

udex = vdex/vde;
udey = vdey/vde;

% uTD
vtdx=u+w*btA/2;
vtdy=v-w*ltA;
vtd=sqrt(vtdx^2+vtdy^2);

utdx = vtdx/vtd;
utdy = vtdy/vtd;

% uTE
vtex=u-w*btA/2;
vtey=v-w*ltA;
vte=sqrt(vtex^2+vtey^2);

utex = vtex/vte;
utey = vtey/vte;

% SOMATÓRIO DAS COMPONENTES DOS UNITÄRIOS

% Ponderação das forças em X
sx = uddx + udex + utdx + utex;

% Ponderaçao das forças em Y
sy = uddy + udey + utdy + utey;

% Ponderaçao dos momentos em Z
sz =(uddx - udex)*bdA/2 + (utdx - utex)*btA/2 + (uddy + udey)*ldA - (utdy + utey)*ltA;
```

unitarios.m - Direções das Velocidades Instantâneas de cada Pneu do Veículo A

```
% UNITÁRIOS DAS DIRECOES DAS VELOCIDADES DOS PNEUS

function [sx,sy,sz] = unitarios_B(u,v,w);

global bdB btB ldB ltB

% uDD
vddx=u+w*bdB/2;
vddy=v+w*ldB;
vdd=sqrt(vddx^2+vddy^2);

uddx = vddx/vdd;
uddy = vddy/vdd;

% uDE
vdex=u-w*bdB/2;
vdey=v+w*ldB;
vde=sqrt(vdex^2+vdey^2);

udex = vdex/vde;
udey = vdey/vde;

% uTD
vtdx=u+w*btB/2;
vtdy=v-w*ltB;
vtd=sqrt(vtdx^2+vtdy^2);

utdx = vtdx/vtd;
utdy = vtdy/vtd;

% uTE
vtex=u-w*btB/2;
vtey=v-w*ltB;
vte=sqrt(vtex^2+vtey^2);

utex = vtex/vte;
utey = vtey/vte;

% SOMATÓRIO DAS COMPONENTES DOS UNITÄRIOS

% Ponderação das forças em X
sx = uddx + udex + utdx + utex;

% Ponderaçao das forças em Y
sy = uddy + udey + utdy + utey;

% Ponderaçao dos momentos em Z
sz =(uddx - udex)*bdB/2 + (utdx - utex)*btB/2 + (uddy + udey)*ldB - (utdy + utey)*ltB;
```

unitarios.m - Direções das Velocidades Instantâneas de cada Pneu do Veículo B

```
% teste de parada da simulacao

function[T]=teste_parada_2(vxA,vyA,wzA,vxB,vyB,wzB)

erro=.05;
T = 0;

para=abs(vxA)+abs(vyA)+abs(wzA)+abs(vxB)+abs(vyB)+abs(wzB);

if para <= erro
   T = 1;
end
```

teste_de_parada_2.m - Função de Parada da Simulação

```
close all
clear
clc
dados_ga_2

options = gaoptimset ('PopInitRange', [-4.0 -2.5 8.0 -9.5 2.5 3.5 -10.0 -1.0 ; 4.0 2.5 15.5 -3.5 8.5 10.5 -4.0 3.0] ,
        'PopulationSize', 100 , 'EliteCount' , 5 , 'CrossoverFraction' , 0.95 , 'Generations' , 40 ,
        'FitnessLimit' , 100.0 , 'StallGenLimit' , 10 , 'StallTimeLimit' , 1000 ,
        'CrossoverFcn', @crossoverintermediate, 'MutationFcn' , {@mutationgaussian [3] [0.6500]} ,
        'PlotFcns' , [@gaplotbestindiv , @gaplotbestf])

[x fval reason finalscores] = ga(@fun_ga_2, 8, options)
pause
close all
anima_ga
```

Parâmetros e Configurações da Função GA

```
global adxA adyA adzA baxA bayA bazA
global bdA btA ldA ltA

global adxB adyB adzB baxB bayB bazB
global bdB btB ldB ltB

pista='pista';

g = 9.81;
ro = 1.2;

% ---- VEICULO A -----

x1A=x(3);x2A=X0cA+x(1);x3A=x(4);x4A=Y0cA+x(2);x5A=x(5);x6A=psi0cA*180/pi;

vx0A= x1A;
X0A = x2A;

vy0A= x3A;
Y0A = x4A;

wz0A = x5A;
psi0A = x6A*pi/180;

mA = 1000;
JzA = 2000;
rz2A = JzA/mA;
ldA = 1;
ltA = 1.5;
btA = 1.5;
bdA = 1.5;

muA = 0.7;

adxA = muA*g/4;
adyA = muA*g/4;
adzA = muA*g/4/rz2A;

CxA = 0.30;
CyA = 0.80;
CmzA = 0.2;
SA = 2.0;

baxA = ro*CxA*SA*abs(vx0A)/2/mA;
bayA = ro*CyA*SA*abs(vy0A)/2/mA;
bazA = ro*CmzA*SA*(ldA+ltA)*abs(wz0A)/2/JzA;
```

anima_ga.m

```
% ---- VEICULO B -----

x1B=x(6);x2B=X0cB+x(1);x3B=x(7);x4B=Y0cB+x(2);x5B=x(8);x6B=psi0cB*180/pi;

vx0B= x1B;
X0B = x2B;

vy0B= x3B;
Y0B = x4B;

wz0B = x5B;
psi0B = x6B*pi/180;

mB = 1000;
JzB = 2000;
rz2B = JzB/mB;
ldB = 1;
ltB = 1.5;
btB = 1.5;
bdB = 1.5;

muB = 0.7;

adxB = muB*g/4;
adyB = muB*g/4;
adzB = muB*g/4/rz2B;

CxB = 0.30;
CyB = 0.80;
CmzB = 0.2;
SB = 2.0;

baxB = ro*CxB*SB*abs(vx0B)/2/mB;
bayB = ro*CyB*SB*abs(vy0B)/2/mB;
bazB = ro*CmzB*SB*(ldB+ltB)*abs(wz0B)/2/JzB;

% ---- SIMULACAO

tf = 5;
dt = 0.05;

sim pos_colisao_2

V_A=sqrt(vx_A.^2+vy_A.^2);
V_B=sqrt(vx_B.^2+vy_B.^2);
anima_trajetoria_t_2(XfcA,YfcA,psifcA,X_A,Y_A,psig_A,V_A,wz_A,
                     XfcB,YfcB,psifcB,X_B,Y_B,psig_B,V_B,wz_B,t,pista);
pause
```

anima_ga.m (continuação)

```
function
        anima_trajetoria_t(XfcA,YfcA,psifcA,XcgA,YcgA,psiA,VA,wA,XfcB,YfcB,psifcB,XcgB,YcgB,psiB,VB,
        wB,t,pista)
%
% Funcao que recebe os vetores de deslocamento em X, Y e o vetor contendo o angulo de yaw, ao longo
% de uma trajetoria
%
% Implementada por A. Spinola - maio/2004
% Modificada por M.Speranza Neto - maio/2005
% Modificada por G. N. Martins - outubro/2005

global np nd

np = 100;  % numero de pontos para representacao dos carros
nd = 25;   % numero de pontos para representacao da dianteira dos carros

% Carro A %
car_A(1) = 0.5;   % largura do carro
car_A(2) = 2.5;   % comprimento do carro
car_A(3) = 1.5;   % distancia entre eixos
car_A(4) = 0.9;   % distancia do eixo dianteiro ao CG
car_A(5) = 0.6;   % distancia do eixo traseiro ao CG
car_A(6) = 1.2;   % bitola dianteira
car_A(7) = 1.0;   % bitola traseira
car_A(8) = 0.7;   % diametro do pneu
car_A(9) = 0.2;   % banda de rodagem do pneu

cor_A ='r';       % cor do carro
corf_A = 'm';     % cor da posição esperada
nome_A='Vermelho';   % nome do carro

% Carro B %
car_B(1) = 0.5;   % largura do carro
car_B(2) = 2.5;   % comprimento do carro
car_B(3) = 1.5;   % distancia entre eixos
car_B(4) = 0.9;   % distancia do eixo dianteiro ao CG
car_B(5) = 0.6;   % distancia do eixo traseiro ao CG
car_B(6) = 1.2;   % bitola dianteira
car_B(7) = 1.0;   % bitola traseira
car_B(8) = 0.7;   % diametro do pneu
car_B(9) = 0.2;   % banda de rodagem do pneu

cor_B ='b';       % cor do carro
corf_B = 'c';     % cor da posição esperada
nome_B='Azul';   % nome do carro

frame=1;
dim = length(t);
```

anima_trajetoria_2.m

```
for j = 1:1:dim

   plotar_pista(pista);

   anima_carro_2(XcgA(j),YcgA(j),psiA(j)*pi/180,cor_A,XcgB(j),YcgB(j),psiB(j)*pi/180,cor_B);
   anima_carro_2(XfcA,YfcA,psifcA*pi/180,corf_A,XfcB,YfcB,psifcB*pi/180,corf_B);

   st=num2str(t(j));
   title(strcat('Animacao de movimento  - Instante de tempo :',st,' s'));

   % Carro A %
   svA=num2str(round(VA(j)*100)/100);
   swA=num2str(round(wA(j)*100)/100);
   spsiA=num2str(round(psiA(j)*100)/100);
   sXA=num2str(round((XcgA(j)-XcgA(1))*100)/100);
   sYA=num2str(round((YcgA(j)-YcgA(1))*100)/100);

   % Carro B %
   svB=num2str(round(VB(j)*100)/100);
   swB=num2str(round(wB(j)*100)/100);
   spsiB=num2str(round(psiB(j)*100)/100);
   sXB=num2str(round((XcgB(j)-XcgB(1))*100)/100);
   sYB=num2str(round((YcgB(j)-YcgB(1))*100)/100);

   xm=axis;
   text(xm(2)-18,xm(4)-1.0,strcat('Tempo :',st,' s'));

   % Carro A %
   text(xm(2)-29,xm(4)-2.0,strcat('Veículo :',nome_A));
   text(xm(2)-29,xm(4)-3.0,strcat('Vel. linear :',svA,'km/h'));
   text(xm(2)-29,xm(4)-4.0,strcat('Vel. angular :',swA,'rot/s'));
   text(xm(2)-29,xm(4)-5.0,strcat('Desloc. X :',sXA,'m'));
   text(xm(2)-29,xm(4)-6.0,strcat('Desloc. Y :',sYA,'m'));
   text(xm(2)-29,xm(4)-7.0,strcat('Ângulo :',spsiA,'graus'));

   % Carro B %
   text(xm(2)-9,xm(4)-2.0,strcat('Veículo :',nome_B));
   text(xm(2)-9,xm(4)-3.0,strcat('Vel. linear :',svB,'km/h'));
   text(xm(2)-9,xm(4)-4.0,strcat('Vel. angular :',swB,'rot/s'));
   text(xm(2)-9,xm(4)-5.0,strcat('Desloc. X :',sXB,'m'));
   text(xm(2)-9,xm(4)-6.0,strcat('Desloc. Y :',sYB,'m'));
   text(xm(2)-9,xm(4)-7.0,strcat('Ângulo :',spsiB,'graus'));
   pause(0.15)
   M(frame) = getframe;
   frame=frame+1;

end
```

anima_trajetoria_2.m (continuação)

```
function carro_2(XcgA,YcgA,psiA,cor_A,XcgB,YcgB,psiB,cor_B)
% Função para construção de uma representação de dois veículos.
% Possui como entradas as coordenadas dos CGs, os ângulos de yaw, e as cores dos mesmos.
% Implementada por Guilherme Martins - outubro/2005

global np nd

% Carro A %
largA = 2;
compA = 5;
corA = 'r';

xldA = [(XcgA-cos(psiA)*compA/2+sin(psiA)*largA/2) (XcgA+cos(psiA)*compA/2+sin(psiA)*largA/2)];
xleA = [(XcgA+cos(psiA)*compA/2-sin(psiA)*largA/2) (XcgA-cos(psiA)*compA/2-sin(psiA)*largA/2)];
xdA = [(XcgA+cos(psiA)*compA/2+sin(psiA)*largA/2) (XcgA+cos(psiA)*compA/2-sin(psiA)*largA/2)];
xtA = [(XcgA-cos(psiA)*compA/2-sin(psiA)*largA/2) (XcgA-cos(psiA)*compA/2+sin(psiA)*largA/2)];

xA= [xdA xleA xtA xldA XcgA xleA];

yldA = [(YcgA-sin(psiA)*compA/2-cos(psiA)*largA/2) (YcgA+sin(psiA)*compA/2-cos(psiA)*largA/2)];
yleA = [(YcgA+sin(psiA)*compA/2+cos(psiA)*largA/2) (YcgA-sin(psiA)*compA/2+cos(psiA)*largA/2)];
ydA = [(YcgA+sin(psiA)*compA/2-cos(psiA)*largA/2) (YcgA+sin(psiA)*compA/2+cos(psiA)*largA/2)];
ytA = [(YcgA-sin(psiA)*compA/2+cos(psiA)*largA/2) (YcgA-sin(psiA)*compA/2-cos(psiA)*largA/2)];

yA= [ydA yleA ytA yldA YcgA yleA];

% Carro B %
largB = 2;
compB = 5;
corB = 'b';

xldB = [(XcgB-cos(psiB)*compB/2+sin(psiB)*largB/2) (XcgB+cos(psiB)*compB/2+sin(psiB)*largB/2)];
xleB = [(XcgB+cos(psiB)*compB/2-sin(psiB)*largB/2) (XcgB-cos(psiB)*compB/2-sin(psiB)*largB/2)];
xdB = [(XcgB+cos(psiB)*compB/2+sin(psiB)*largB/2) (XcgB+cos(psiB)*compB/2-sin(psiB)*largB/2)];
xtB = [(XcgB-cos(psiB)*compB/2-sin(psiB)*largB/2) (XcgB-cos(psiB)*compB/2+sin(psiB)*largB/2)];

xB= [xdB xleB xtB xldB XcgB xleB];

yldB = [(YcgB-sin(psiB)*compB/2-cos(psiB)*largB/2) (YcgB+sin(psiB)*compB/2-cos(psiB)*largB/2)];
yleB = [(YcgB+sin(psiB)*compB/2+cos(psiB)*largB/2) (YcgB-sin(psiB)*compB/2+cos(psiB)*largB/2)];
ydB = [(YcgB+sin(psiB)*compB/2-cos(psiB)*largB/2) (YcgB+sin(psiB)*compB/2+cos(psiB)*largB/2)];
ytB = [(YcgB-sin(psiB)*compB/2+cos(psiB)*largB/2) (YcgB-sin(psiB)*compB/2-cos(psiB)*largB/2)];

yB= [ydB yleB ytB yldB YcgB yleB];

plot(xA,yA,cor_A,xB,yB,cor_B);
axis([(XcgA+XcgB)/2-1.5*(compA+compB)  (XcgA+XcgB)/2+1.5*(compA+compB)  (YcgA+YcgB)/2-
      (compA+compB) (YcgA+YcgB)/2+2*(compA+compB)]);
```

anima_carro_2.m

```
function [] = plotar_pista(arquivo);
% Representar a pista no referencial global

load(arquivo);

figure(11)
close (11)
figure(11)
hold

lacos=3;
% largura do acostamento

plotar(xcp,ycp,xdp,ydp,xep,yep);

plotar(xcp,ycp,xdp,ydp-lacos,xep,yep+lacos);
% Acostamento

plotar_i(xc,yc,xd,yd,xe,ye);

plotar_i(xc,yc,xd,yd-lacos,xe,ye+lacos);
% Acostamento
```

plotar_pista.m

```
function [] = plotar(xc,yc,xd,yd,xe,ye)
% Representar os trechos da pista no referencial global

quadro(xc,xd,xe,yc,yd,ye);

plot(xd,yd,'-k');
plot(xc,yc,'-y');
plot(xe,ye,'-k');
```

plotar.m

```
function [] = plotar_i(xc,yc,xd,yd,xe,ye)

% Representar o trecho inicial da pista

plot(xd,yd,'-k');
plot(xc,yc,'-y');
plot(xe,ye,'-k');
```

plotar_i.m

7.4.
Caso Completo

Os códigos-fonte apresentados a seguir são somente aqueles que diferem dos que constam no Apêndice item 8.3.

Cabe esclarecer que os arquivos dados_ga.m e limites_ga.m devem ser alterados para a análise de cada evento de colisão. Os arquivos dados_ga.m e limites_ga.m aqui apresentados restringem-se respectivamente aos casos de colisão lateral oblíqua e colisão traseira oblíqua, trabalhados no Capítulo 5.

```
global ma Ja
global mb Jb
global cr lambda
global Xcga Ycga Xcgb Ycgb fiac fibc
global LI1 LS1 LI2 LS2 LI3 LS3 LI4 LS4 LI5 LS5 LI6 LS6 LI7 LS7 LI8 LS8
global wa2 wb2 Vax2 Vay2 Vbx2 Vby2
global var

 % Coordenadas dos CG dos veículos em relação ao referencial do impacto (m)

  xa=Xcga;
  ya=Ycga;
  xb=Xcgb;
  yb=Ycgb;

  Vax1=(LS3+LI3+var(3)*(LS3-LI3))/2;
  Vay1=(LS4+LI4+var(4)*(LS4-LI4))/2;
  Vbx1=(LS6+LI6+var(6)*(LS6-LI6))/2;
  Vby1=(LS7+LI7+var(7)*(LS7-LI7))/2;

  wa1 = (LS5+LI5+var(5)*(LS5-LI5))/2;
  wb1 = (LS8+LI8+var(8)*(LS8-LI8))/2;

  [vax1,vay1]=ref2ref(0,0,fiac,Vax1,Vay1);
  [vbx1,vby1]=ref2ref(0,0,fibc,Vbx1,Vby1);

  %
  % ANALISE DE COLISOES PLANAS DE VEICULOS RIGIDOS
  %
  % SOLUCAO MATRICIAL
  %
  A1 = [     ma          0       mb          0     0     0;
        0           ma       0          mb     0     0;
   ma*(ya-lambda*xa)  0       0          0     -Ja    0;
        0           0  mb*(yb-lambda*xb)  0     0     -Jb;
       cr           0      -cr          0    cr*ya  -cr*yb;
      lambda        -1       0          0     0      0];

  A2 = [     ma          0       mb          0     0     0;
        0           ma       0          mb     0     0;
   ma*(ya-lambda*xa)  0       0          0     -Ja    0;
        0           0   mb*(yb-lambda*xb)  0     0     -Jb;
       -1           0        1          0     -ya    yb;
      lambda        -1       0          0     0      0];
```

colisao_ga.m

```
v_1=[vax1;vay1;vbx1;vby1;wa1;wb1];

   v_2=inv(A2)*A1*v_1;

   vax2=v_2(1);
   vay2=v_2(2);
   vbx2=v_2(3);
   vby2=v_2(4);
   wa2= v_2(5);
   wb2= v_2(6);

   %
   % Velocidades depois da colisao no referencial local dos veiculos
   %
   [Vax2,Vay2]=ref2ref(0,0,-fiac,vax2,vay2);
   [Vbx2,Vby2]=ref2ref(0,0,-fibc,vbx2,vby2);
```

colisao_ga.m (continuação)


```
global LI1 LS1 LI2 LS2 LI3 LS3 LI4 LS4 LI5 LS5 LI6 LS6 LI7 LS7 LI8 LS8
global LI9 LS9 LI10 LS10 LI11 LS11 LI12 LS12 LI13 LS13 LI14 LS14 LI15 LS15 LI16 LS16 LI17
       LS17
global ba lda lta
global bb ldb ltb
global fia_g fib_g
global Xcga Ycga Xcgb Ycgb fiac fibc
global X0 Y0
global Xcgag Ycgag Xcgbg Ycgbg
global var



% VARIÁVEIS GLOBAIS %
X0 = (LS1+LI1+var(1)*(LS1-LI1))/2;
Y0 = (LS2+LI2+var(2)*(LS2-LI2))/2;

% Veiculo 1
xia = (LS9+LI9+var(9)*(LS9-LI9))/2;
yia = (LS10+LI10+var(10)*(LS10-LI10))/2;
xfa = (LS11+LI11+var(11)*(LS11-LI11))/2;
yfa = (LS12+LI12+var(12)*(LS12-LI12))/2;

%

raio2 = (xia-xfa)^2 + (yia-yfa)^2;
```

geometria_ga.m

```
% Veiculo 2
xib = (LS13+LI13+var(13)*(LS13-LI13))/2;
yib = (LS14+LI14+var(14)*(LS14-LI14))/2;


if (LI13+LS13)/2 == ldb                 %se ponto inicial na dianteira
   if yib > 0                           %se ponto inicial na dianteira esquerda
      if (LI16+LS16)/2 < yib            %se ponto final à direita
         if sqrt(raio2) < (bb/2 + yib)          %se ponto final na dianteira
            xfb = ldb;
            yfb = yib - sqrt(raio2);
         else                           %se ponto final na lateral direita
            xfb = min (roots([1   -2*xib   (xib^2+((bb/2)+yib)^2-raio2)]));
            yfb = -bb/2;
         end
      else                              %se ponto final à esquerda
         if sqrt(raio2) < (bb/2 - yib)          %se ponto final na dianteira
            xfb = ldb;
            yfb = yib + sqrt(raio2);
         else                           %se ponto final na lateral esquerda
            xfb = min (roots([1   -2*xib   (xib^2+((bb/2)-yib)^2-raio2)]));
            yfb = bb/2;
         end
      end
   else                                 %se ponto inicial na dianteira direita
      if (LI16+LS16)/2 > yib            %se ponto final à esquerda
         if sqrt(raio2) < (bb/2 - yib)          %se ponto final na dianteira
            xfb = ldb;
            yfb = yib + sqrt(raio2);
         else                           %se ponto final na lateral esquerda
            xfb = min (roots([1   -2*xib   (xib^2+((bb/2)-yib)^2-raio2)]));
            yfb = bb/2;
         end
      else                              %se ponto final à direita
         if sqrt(raio2) < (bb/2 + yib)          %se ponto final na dianteira
            xfb = ldb;
            yfb = yib - sqrt(raio2);
         else                           %se ponto final na lateral direita
            xfb = min (roots([1   -2*xib   (xib^2+((bb/2)+yib)^2-raio2)]));
            yfb = -bb/2;
         end
      end
   end
else
```

geometria_ga.m (continuação)

```
if (LI13+LS13)/2 == -ltb
    if yib > 0
      if (LI16+LS16)/2 < yib
        if sqrt(raio2) < (bb/2 + yib)
          xfb = -ltb;
          yfb = yib - sqrt(raio2);
        else
          xfb = max (roots([1   -2*xib   (xib^2+((bb/2)+yib)^2-raio2)]));
          yfb = -bb/2;
        end
      else
        if sqrt(raio2) < (bb/2 - yib)
          xfb = -ltb;
          yfb = yib + sqrt(raio2);
        else
          xfb = max (roots([1   -2*xib   (xib^2+((bb/2)-yib)^2-raio2)]));
          yfb = bb/2;
        end
      end
    else
      if (LI16+LS16)/2 > yib
        if sqrt(raio2) < (bb/2 - yib)
          xfb = -ltb;
          yfb = yib + sqrt(raio2);
        else
          xfb = max (roots([1   -2*xib   (xib^2+((bb/2)-yib)^2-raio2)]));
          yfb = bb/2;
        end
      else
        if sqrt(raio2) < (bb/2 + yib)
          xfb = -ltb;
          yfb = yib - sqrt(raio2);
        else
          xfb = max (roots([1   -2*xib   (xib^2+((bb/2)+yib)^2-raio2)]));
          yfb = -bb/2;
        end
      end
    end
  else
```

geometria_ga.m (continuação)

```
if (LI14+LS14)/2 == bb/2
   if xib > 0
      if (LI15+LS15)/2 < xib
         if sqrt(raio2) < (ltb + xib)
            xfb = xib - sqrt(raio2);
            yfb = bb/2;
         else
            xfb = -ltb;
            yfb = min (roots ([1   -2*yib   (yib^2+(ltb+xib)^2-raio2)]));
         end
      else
         if sqrt(raio2) < (ldb - xib)
            xfb = xib + sqrt(raio2);
            yfb = bb/2;
         else
            xfb = ldb;
            yfb = min (roots ([1   -2*yib   (yib^2+(ldb-xib)^2-raio2)]));
         end
      end
   else
      if (LI15+LS15)/2 > xib
         if sqrt(raio2) < (ldb - xib)
            xfb = xib + sqrt(raio2);
            yfb = bb/2;
         else
            xfb = ldb;
            yfb = min (roots ([1   -2*yib   (yib^2+(ldb-xib)^2-raio2)]));
         end
      else
         if sqrt(raio2) < (ltb + xib)
            xfb = xib - sqrt(raio2);
            yfb = bb/2;
         else
            xfb = -ltb;
            yfb = min (roots ([1   -2*yib   (yib^2+(ltb+xib)^2-raio2)]));
         end
      end
   end
```

geometria_ga.m (continuação)

```
else
        if xib > 0
           if (LI15+LS15)/2 < xib
              if sqrt(raio2) < (ltb + xib)
                 xfb = xib - sqrt(raio2);
                 yfb = -bb/2;
              else
                 xfb = -ltb;
                 yfb = max (roots ([1   -2*yib   (yib^2+(ltb+xib)^2-raio2)]));
              end
           else
              if sqrt(raio2) < (ldb - xib)
                 xfb = xib + sqrt(raio2);
                 yfb = -bb/2;
              else
                 xfb = ldb;
                 yfb = max (roots ([1   -2*yib   (yib^2+(ldb-xib)^2-raio2)]));
              end
           end
        else
           if (LI15+LS15)/2 > xib
              if sqrt(raio2) < (ldb - xib)
                 xfb = xib + sqrt(raio2);
                 yfb = -bb/2;
              else
                 xfb = ldb;
                 yfb = max (roots ([1   -2*yib   (yib^2+(ldb-xib)^2-raio2)]));
              end
           else
              if sqrt(raio2) < (ltb + xib)
                 xfb = xib - sqrt(raio2);
                 yfb = -bb/2;
              else
                 xfb = -ltb;
                 yfb = max (roots ([1   -2*yib   (yib^2+(ltb+xib)^2-raio2)]));
              end
           end
        end
     end
  end
end
```

geometria_ga.m (continuação)

```
% Ângulos (em graus) iniciais dos veiculos em relacao ao referencial global
% Veiculo 1
%
ag=0;
%
% Veiculo 2
%
bg=180;
%
fiag = ag*pi/180;
fibg = bg*pi/180;
%
% Angulo do veiculo 1 em relacao ao referencial da colisao
%
aa = atan2((yfa-yia),(xfa-xia));
fiac=(fiag+((pi/2)-aa));
fia=fiag;

% Angulo do veiculo 2 em relacao ao referencial da colisao
%
ab = atan2((yfb-yib),(xfb-xib));
fibc = (fibg+((pi/2)-ab));
fib = fibg;

% Determinacao do plano de colisao
   %
   if xfa > xia
         Xma = xia+(xfa-xia)/2;
   else
         Xma = xfa+(xia-xfa)/2;
   end
   if yfa > yfa
        Yma = yia+(yfa-yia)/2;
   else
        Yma = yfa+(yia-yfa)/2;
   end

 if xfb > xib
         Xmb = xib+(xfb-xib)/2;
   else
         Xmb = xfb+(xib-xfb)/2;
   end
   if yfb > yfb
        Ymb = yib+(yfb-yib)/2;
   else
        Ymb = yfb+(yib-yfb)/2;
   end
```

geometria_ga.m (continuação)

```
% Posicao dos veiculos no instante da colisao
%

[Xcga,Ycga] = ref2ref(0,0,fiac,-Xma,-Yma);

[Xcgb,Ycgb] = ref2ref(0,0,fibc,-Xmb,-Ymb);

% Posicao do choque
%
px = X0;
py = Y0;
%
% Entre com a orientacao (em graus) do referencial da colisao (eixo x) em relacao ao
%referencial global (eixo X)
%
% Obs:
%     1) Para alinhar o Veiculo 1 com o eixo X: axg = ag-(fiac*180/pi)
%     2) Para alinhar o Veiculo 2 com o eixo X: axg = bg-(fibc*180/pi)
%
%
 axg = ag-(fiac*180/pi)-(LS17+LI17+var(15)*(LS17-LI17))/2;
 ax  = axg*pi/180;
%
% Angulos dos veículos com relação ao eixo X do referencial global (graus)
%
fia_g=(fiac+ax)*180/pi;
fib_g=(fibc+ax)*180/pi;

[Xcgag,Ycgag] = ref2ref(px,py,ax,Xcga,Ycga);     % Posicao inicial no referencial global
[Xcgbg,Ycgbg] = ref2ref(px,py,ax,Xcgb,Ycgb);
```

geometria_ga.m (continuação)

```
% Transformacao de coordenadas
function[X,Y] = ref2ref(X0,Y0,fi,x,y)

X=X0+x*cos(fi)-y*sin(fi);
Y=Y0+x*sin(fi)+y*cos(fi);
```

ref2ref.m

```
% local para global
function[V]=loc2glob(fi,vy,ux)

V(1,1)=ux*cos(fi)-vy*sin(fi);
V(2,1)=ux*sin(fi)+vy*cos(fi);
```

loc2glob.m

```
% GLOBAL DO CARRO A %
global adxA adyA adzA baxA bayA bazA
global bdA btA ldA ltA
global X0cA Y0cA psi0cA XfcA YfcA psifcA
global CxA CyA CmzA SA JzA mA
global vx0A X0A vy0A Y0A wz0A psi0A

% GLOBAL DO CARRO B %
global adxB adyB adzB baxB bayB bazB
global bdB btB ldB ltB
global X0cB Y0cB psi0cB XfcB YfcB psifcB
global CxB CyB CmzB SB JzB mB
global vx0B X0B vy0B Y0B wz0B psi0B

% GLOBAL DE AMBOS %
global ro dt tf
global d
global wa2 wb2 Vax2 Vay2 Vbx2 Vby2
global fia_g fib_g
global Xcga Ycga Xcgb Ycgb fiac fibc
global X0 Y0
global Xcgag Ycgag Xcgbg Ycgbg

% VARIÁVEIS DO CARRO A %
vx0A = Vax2;
X0A = Xcgag;
vy0A = Vay2;
Y0A = Ycgag;
wz0A = wa2;
psi0A = fia_g*pi/180;

baxA = ro*CxA*SA*abs(vx0A)/2/mA;
bayA = ro*CyA*SA*abs(vy0A)/2/mA;
bazA = ro*CmzA*SA*(ldA+ltA)*abs(wz0A)/2/JzA;

% VARIÁVEIS DO CARRO B %
vx0B = Vbx2;
X0B = Xcgbg;
vy0B = Vby2;
Y0B = Ycgbg;
wz0B = wb2;
psi0B = fib_g*pi/180;

baxB = ro*CxB*SB*abs(vx0B)/2/mB;
bayB = ro*CyB*SB*abs(vy0B)/2/mB;
bazB = ro*CmzB*SB*(ldB+ltB)*abs(wz0B)/2/JzB;
```

pos_colisao_ga.m

```
% SIMULA AMBOS OS CARROS %
sim pos_colisao_2

% POSIÇÃO FINAL DO CARRO A %
XfA=X_A(length(X_A));
YfA=Y_A(length(Y_A));
psifA=psig_A(length(psig_A));

XddA=XfA+ldA*cos(psifA*pi/180)+(bdA/2)*sin(psifA*pi/180);
YddA=YfA+ldA*sin(psifA*pi/180)-(bdA/2)*cos(psifA*pi/180);
XteA=XfA-ltA*cos(psifA*pi/180)-(btA/2)*sin(psifA*pi/180);
YteA=YfA-ltA*sin(psifA*pi/180)+(btA/2)*cos(psifA*pi/180);

XddcA=XfcA+ldA*cos(psifcA*pi/180)+(bdA/2)*sin(psifcA*pi/180);
YddcA=YfcA+ldA*sin(psifcA*pi/180)-(bdA/2)*cos(psifcA*pi/180);
XtecA=XfcA-ltA*cos(psifcA*pi/180)-(btA/2)*sin(psifcA*pi/180);
YtecA=YfcA-ltA*sin(psifcA*pi/180)+(btA/2)*cos(psifcA*pi/180);

% POSIÇÃO FINAL DO CARRO B %
XfB=X_B(length(X_B));
YfB=Y_B(length(Y_B));
psifB=psig_B(length(psig_B));

XddB=XfB+ldB*cos(psifB*pi/180)+(bdB/2)*sin(psifB*pi/180);
YddB=YfB+ldB*sin(psifB*pi/180)-(bdB/2)*cos(psifB*pi/180);
XteB=XfB-ltB*cos(psifB*pi/180)-(btB/2)*sin(psifB*pi/180);
YteB=YfB-ltB*sin(psifB*pi/180)+(btB/2)*cos(psifB*pi/180);

XddcB=XfcB+ldB*cos(psifcB*pi/180)+(bdB/2)*sin(psifcB*pi/180);
YddcB=YfcB+ldB*sin(psifcB*pi/180)-(bdB/2)*cos(psifcB*pi/180);
XtecB=XfcB-ltB*cos(psifcB*pi/180)-(btB/2)*sin(psifcB*pi/180);
YtecB=YfcB-ltB*sin(psifcB*pi/180)+(btB/2)*cos(psifcB*pi/180);

% FUNÇÃO DE AVALIAÇÃO %
d = 1000*sqrt((XddA-XddcA)^2+(YddA-YddcA)^2+(XteA-XtecA)^2+(YteA-YtecA)^2+(XddB-
    XddcB)^2+(YddB-YddcB)^2+(XteB-XtecB)^2+(YteB-YtecB)^2);
```

pos_colisao_ga.m

```
global d
global var
function [d]= fun_ga_2_lim(vet)

var = vet;
geometria_ga
colisao_ga
pos_colisao_ga
```

fun_ga_2_lim.m

```
close all
clear
clc

limites_ga

dados_ga_2

options = gaoptimset ('PopInitRange', [-1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0;
        1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0] , 'PopulationSize', 100 , 'EliteCount' , 5 ,
        'CrossoverFraction' , 0.95 , 'MigrationInterval', 50 , 'Generations' , 20 , 'FitnessLimit' , 100.0 ,
        'StallGenLimit' , 3 , 'StallTimeLimit' , 1000 , 'CrossoverFcn', @crossoverintermediate,
        'MutationFcn' , {@mutationgaussian [1] [0.7500]} , 'OutputFcns', @gaoutputgen ,
        'PlotFcns' , [@gaplotbestindiv , @gaplotbestf , @gaplotdistance , @gaplotscorediversity])

[x fval reason finalscores] = ga(@fun_ga_2_lim, 15, options)

pause

close all

anima_ga_lim
```

run_fun_2_lim.m

```
%Colisão Frontal%
% GLOBAL DO CARRO A %
global adxA adyA adzA baxA bayA bazA
global bdA btA ldA ltA
global ba lda lta
global X0cA Y0cA psi0cA XfcA YfcA psifcA
global CxA CyA CmzA SA JzA mA
global vx0A X0A vy0A Y0A wz0A psi0A
global ma Ja

% GLOBAL DO CARRO B %
global adxB adyB adzB baxB bayB bazB
global bdB btB ldB ltB
global bb ldb ltb
global X0cB Y0cB psi0cB XfcB YfcB psifcB
global CxB CyB CmzB SB JzB mB
global vx0B X0B vy0B Y0B wz0B psi0B
global mb Jb

% GLOBAL DE AMBOS %
global ro dt tf
global cr lambda



pista='pista';
g = 9.81;
ro = 1.2;
dt = 0.05;
tf = 5;

% CARRO A%

% - GEOMETRIA FINAL %
XfcA = 81.5412;
YfcA = -4.8245;
psifcA = -7.5545;

% - CARACTERÍSTICAS %
mA = ma;
JzA = Ja;
rz2A = JzA/mA;

ldA = lda;
ltA = lta;
btA = ba;
bdA = ba;
```

dados_ga.m

```
muA = 0.7;

adxA = muA*g/4;
adyA = muA*g/4;
adzA = muA*g/4/rz2A;

CxA = 0.30;
CyA = 0.80;
CmzA = 0.2;
SA = 2.0;

baxA = ro*CxA*SA*abs(vx0A)/2/mA;
bayA = ro*CyA*SA*abs(vy0A)/2/mA;
bazA = ro*CmzA*SA*(ldA+ltA)*abs(wz0A)/2/JzA;

% CARRO B%

% - GEOMETRIA FINAL %
XfcB = 27.0112;
YfcB = 7.7570;
psifcB = 155.1137;

% - CARACTERÍSTICAS %
mB = mb;
JzB = Jb;
rz2B = JzB/mB;

ldB = ldb;
ltB = ltb;
btB = bb;
bdB = bb;

muB = 0.7;

adxB = muB*g/4;
adyB = muB*g/4;
adzB = muB*g/4/rz2B;

CxB = 0.30;
CyB = 0.80;
CmzB = 0.2;
SB = 2.0;

baxB = ro*CxB*SB*abs(vx0B)/2/mB;
bayB = ro*CyB*SB*abs(vy0B)/2/mB;
bazB = ro*CmzB*SB*(ldB+ltB)*abs(wz0B)/2/JzB;
```

dados_ga.m (continuação)

```
%
   % Coeficiente de Restituição
   %
   % Choque perfeitamente elástico cr = 1
   % Choque inelástico 0 < cr < 1
   % Choque perfeitamente plástico cr = 0
   %
   %
   cr=0.5;

   %
   % Coeficiente de Atrito Transversal
   %
   %
   % sinal(VR1t/VR1n) > 0 lambda > 0
   % sinal(VR1t/VR1n) < 0 lambda < 0
   %
   % ATENÇÃO:
   % 1) Ordem de grandeza de lambda: aproximadamente 0,5
   % 2) O coeficiente de atrito (lambda) pode ser maior que aquele associado ao
        deslizamento das superfícies dos veículos ... !
   % 3) Normalmente para choque nao obliquos lambda = 0.
   % 4) Para choques laterais lambda >< 0
   %
   %
   lambda=-1.8000;
```

dados_ga.m (continuação)

```
%Colisão Traseira%

global LI1 LS1 LI2 LS2 LI3 LS3 LI4 LS4 LI5 LS5 LI6 LS6 LI7 LS7 LI8 LS8
global LI9 LS9 LI10 LS10 LI11 LS11 LI12 LS12 LI13 LS13 LI14 LS14 LI15 LS15 LI16 LS16 LI17 LS17
global ba lda lta
global ma Ja
global bb ldb ltb
global mb Jb

% DIMENSOES E PROPRIEDADES DOS VEICULOS

% VEICULO 1

ma=1040;
Ja=1.2694e+003;
ba = 1.5480;
lda = 1.6148;
lta = 2.4438;

% VEICULO 2

mb=1668;
Jb=2.0974e+003;
bb = 1.6080;
ldb = 1.6224;
ltb = 2.4336;

% LIMITES INFERIORES E SUPERIORES

% POSICAO E VELOCIDADES DOS VEICULOS

LI1 = 39;       %Limite inferior da variavel x do local de colisao
LS1 = 41;       %Limite superior da variavel x do local de colisao
LI2 = -1.5;     %Limite inferior da variavel y do local de colisao
LS2 = -0.5;     %Limite superior da variavel y do local de colisao
LI3 = 10;       %Limite inferior da velocidade x de pre-colisao do veiculo 1
LS3 = 14;       %Limite superior da velocidade x de pre-colisao do veiculo 1
LI4 = -2;       %Limite inferior da velocidade y de pre-colisao do veiculo 1
LS4 = 0;        %Limite superior da velocidade y de pre-colisao do veiculo 1
LI5 = 1.0;      %Limite inferior da velocidade angular de pre-colisao do veiculo 1
LS5 = 2.0;      %Limite superior da velocidade angular de pre-colisao do veiculo 1
LI6 = 8;        %Limite inferior da velocidade x de pre-colisao do veiculo 2
LS6 = 12;       %Limite superior da velocidade x de pre-colisao do veiculo 2
LI7 = 0.5;      %Limite inferior da velocidade y de pre-colisao do veiculo 2
LS7 = 1.5;      %Limite superior da velocidade y de pre-colisao do veiculo 2
LI8 = -0.8;     %Limite inferior da velocidade angular de pre-colisao do veiculo 2
LS8 = -0.2;     %Limite superior da velocidade angular de pre-colisao do veiculo 2
```

limites_ga.m

```
% PARTES COLIDIDAS DO VEICULO 1

LI9 = lda;              %Limite inferior de xia
LS9 = lda;             %Limite superior de xia
LI10 =-0.55*ba/2;  %Limite inferior de yia
LS10 =-0.45*ba/2; %Limite superior de yia
LI11 = 0.89*lda;      %Limite inferior de xfa
LS11 = 0.91*lda;     %Limite superior de xfa
LI12 = ba/2;           %Limite inferior de yfa
LS12 = ba/2;;         %Limite superior de yfa


% PARTES COLIDIDAS DO VEICULO 2

LI13 =-0.91*ltb;     %Limite inferior de xib
LS13 =-0.89*ltb;   %Limite superior de xib
LI14 = bb/2;          %Limite inferior de yib
LS14 = bb/2;         %Limite superior de yib
LI15 = -ltb;           %Limite inferior de xfb
LS15 = -ltb;          %Limite superior de xfb
LI16 = bb/2; ;        %Limite inferior de yfb
LS16 = bb/2;         %Limite superior de yfb

%
% LIMITES DO ANGULO (EM GRAUS) DO VEICULO 1 NO REFERENCIAL GLOBAL
%
LI17 = 2;           %Limite inferior do angulo (em graus) do veículo 1 em relação ao referencial global
LS17 = 4;          %Limite superior do angulo (em graus) do veículo 1 em relação ao referencial global
```

limites_ga.m (continuação)