7
Referências bibliográficas

ABDULMASSIH, D. S.. **Modelos de veículos rígidos para Análise e Simulação de Colisões e Reconstituição de Acidentes**. Dissertação de mestrado, PUC-Rio, 2003.

ABE, M. et al. **Three-dimensional behavior Simulation of Vehicle collision by dynamic model**: JSAE, junho de 1998.

Accident Reconstruction Network. **Disponível em :**
<www.accidentreconstruction.com/orgs/index.asp>

BAKER, S. J.; FRICKE, L.B. **The traffic accident investigation manual**. 9ª ed. Illinois: Northwestern University Traffic Institute, 1986.

BONNETT, G. M. **Understanding delta V from damage**. 1996. Disponível em: www.rec-tec.com/deltaV.html

BREWER, J. C. **Effects of angles and offsets in crash simulations of automobiles with light trucks.** EUA: Volpe National Transportation Systems Center, Paper No 308, 2001.

BROWN, D. R. et al. **Practical application of vehicle speed determination from crush measurements.** SAE-870798, 1987.

BUKHARD, B. M. **ΔV, BEV and coefficient of restitution relationships as applied to the interpretation of vehicle crash test data.** SAE-2001-01-499, 2001.

BURKHARD, P. M. **Determination of b_1 coefficients from lower and higher speed impacts using peak force.** SAE-2001-01-0501, 2001.

CARPENTER, N. J.; WELCHER, J. B. **Stiffness and crush energy for vehicle collision and its relationship to barrier equivalent velocity (BEV).** SAE-2001-01-0500, 2001.

CARVALHO, F. V. et al. **Modelos de veículos flexíveis para o tratamento de colisões planas**. SAE Brasil 2003-01-3617, 2003.

CLIFF, W. E.; ANDREAS, M. **Reconstruction of twenty staged collisions with PC-Crash's optimizer**. SAE-2001-01-0507, 2001.

DA ROCHA, R. S. et al. **Um modelo para simulação computacional da colisão de veículos terrestres com estrutura deformável**. COBEM 99, 1999.

DAY, T. D. **An Overview of the EDSMAC4 collision simulation model.** SAE-1999-01-0102, Accident Reconstruction: Technology and Animation IX (SP-1407), 1999.

Day, T. D.; HARGENS, R. L. **An overview of the way EDCRASH computes delta-V.** SAE-870045, 1987.

Day, T. D.; HARGENS, R. L. **An overview of the way EDSMAC computes delta-V.** SAE-880069, 1988.

Day, T. D.; HARGENS, R. L. **Application and misapplication of computer programs for accident reconstruction.** SAE-890738, 1989.

DAY, T. D.; SIDALL, D. E. **Three dimensional Reconstruction and simulation of motor vehicle accidents.** SAE-960890, 1996.

DAY, T. D.; SIDALL, D. E., **Validation of several reconstruction and simulation models in the HVE scientific visualization environment**. SAE-960891, Accident Reconstruction: Technology and Animation VI (SP-1150), 1996.

DAY, T. D.; YORK, A. R. **Validation of DyMesh for vehicle vs barrier collisions.** SAE-2000-01-0844, Accident Reconstruction: Technology and Visualization (SP-1491), 2000.

DAY, T.D. **The scientific visualization of motor accidents.** SAE-940922, 1994.

DUFF, J. **Co-axial impact theory lecture, special problems.** Y2Km Come Crash with us, IPTM Lecture, Maio de 2000.

FAY, R. et al. **PC-Crash and HVE, an overview of similarities and differences**. SAE – 2001-01-0505, 2001.

FÉLEZ,J.; VERA, C.; MARTINEZ, M. L., **Traffic Accident Analysis Using Virtual Reality and Bond Graph Techniques.** INSIA Universidad Politécnica de Madrid, 1998

GENTA, G. **Motor vehicle dynamics - modeling and simulation**, Worlds Scientific, 1997.

HUANG, M. **Vehicle crash mechanics**. CRC Press, 2002.

HUANG, M.; TYAN, T; FARUQUE, O. Revista Automotive Engineering International, pag 81, **Target-vehicle modeling in crash analysis**, junho de 2001.

HUPPER, P., WECH, L.; SCHÜLER, F. **Crash tests with electronically guided vehicles**. Imech, 1992.

Keva Engineering. Disponível em: <www.kevaeng.com/Publications.shtml>

LOZANO, J. A.; VERA C.; FÉLEZ, J. **A computational dynamical model for traffic accident reconstruction**. International Journal of Vehicle Design, vol. 19, nº 2, pág 213, 1998.

MACMILLAN, R.H. **Dynamics of vehicle collisions.** St. Helier, 1983.

Matlab – versão do estudante, guia do usuário, versão 4. The Math Works inc.,, 1997

MC HENRY, B. G.; MC HENRY, R. R. **Effects of restitution in the application of crush coefficients.** SAE-970960, 1997.

MC HENRY, B. G.; MC HENRY, R. R., **SMAC-97 refinement of the collision algorithm.** SAE970947, 1997.

MC HENRY, B. G. **SMAC computer program.** Presented at the SAE Accident Reconstruction state-of-the-art TOPTEC, dezembro de 1999. Disponível em: <http://www.mchenrysoftware.com/>.

MC HENRY, B.G. **The algorithms of CRASH.** Disponível em: <www.mchenrysoftware.com>, 2001.

MOTOZAWA, Y.; KAMEI, T. **Controlando a desaceleração durante uma colisão**. Honda R&d Co. Revista Engenharia Automotiva e Aeroespacial, novembro/ dezembro de 2000.

National Highway Traffic Safety Administration. Disponível em: <www.nhtsa.dot.gov>

National Roads and Motorists' Association. Disponível em: <www.nrma.com.au>

NEADES, J. **A Comparison between Edcrash and AI Damage**. Disponível em <http://www.aitsuk.com>

NYSTROM, G. A. **Stiffness parameters for vehicle collision analysis, an update.** SAE-2001-01-0502, 2001.

OLSON B. D.; SMITH, C. C. **A modular dynamic simulation approach to the reconstruction of automobile accidents**, Journal of Dynamic Systems Measurement and Control, Dezembro de 1993.

PRASSAD, A. K. **Energy absorbing properties of vehicle structures and their use in estimating impact severity in automobile collisions**", ImechE 925209, p61, 1992.

RIVERS, R. W. **Traffic accident investigators' and reconstructionists' book of formulae and tables**, Charles C Thomas Pub Ltd; November 1999.

SaGA Contents. Disponível em:
<http://puddle.mit.edu/~glenn/kirill/SAGAHTML/Contents.html>

SIDALL, D. E.; DAY, T. D., **Updating the vehicle class categories**, SAE-960897, Accident Reconstruction: Technology and Animation VI (SP-1150), 1996.

SÖDEBERG, U.; TIDBORG, F. **Evaluation of methods for calculation of impact severity in frontal impacts**, M.Sc. Thesis, Department of Machine and Vehicle Design - Chalmers University of Technology, 1999.

SPERANZA NETO et al., **The dynamic behavior of elastic-plastic materials modeled by bond graphs.** Diname, 1997.

SPERANZA NETO, M.; DA ROCHA, R. S. **Um modelo para a colisão de veículos terrestres com estrutura deformável**. COBEM 99, 1999.

SPERANZA NETO, M.; LAGE, J. D. **Simulação de colisão de veículos terrestres. Um tratamento através da dinâmica de sistemas**. SIMEA 97, 1997

VAN KIRK, D. J. **Vehicular accident investigation and reconstruction,** CRC Press, 2001.

VERA, C., APARAICIO. F.; SAN ROMÁN, J. L. **Theoretic model for the computer analysis of vehicle collisions.** Safety Science n19, 1995.

WOOLLEY, R. L. **Non-linear damage analysis in accident reconstruction.** SAE-2001-01-0504, 2001.

YAMAGUCHI, J. et al. **Crashworthiness**, Automotive Engineering International, pag 74, Junho de 1999.

YAO CHAN, C. **Studies of vehicle collisions – a documentation of the simulation codes: SMAC (Simulation Model of Automobile Collisions)**, Institute of transportation Studies, University of California, Berkeley, 1997.

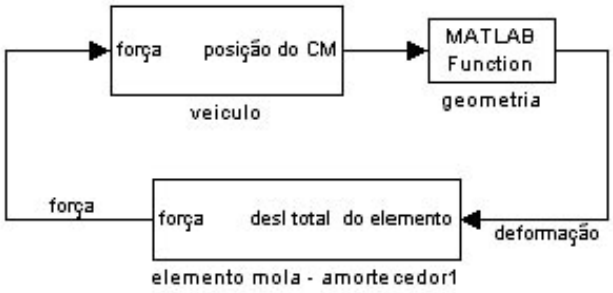
YORK, A. R.; DAY, T. D. **The DyMesh method for three-dimensional multi-vehicle collision simulation.** SAE-1999-01-0104, Accident Reconstruction: Technology and Animation IX (SP-1407), 1999.

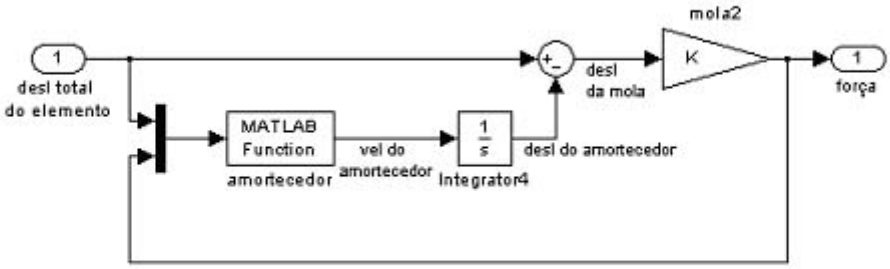
ZAOUK, A. K. et al. **Development and Evaluation of a C-1500 Truck Model for a Roadside Hardware Impact Simulation.** FHWA/NHTSA National Crash Analysis Center, The George Washington University, 1996.

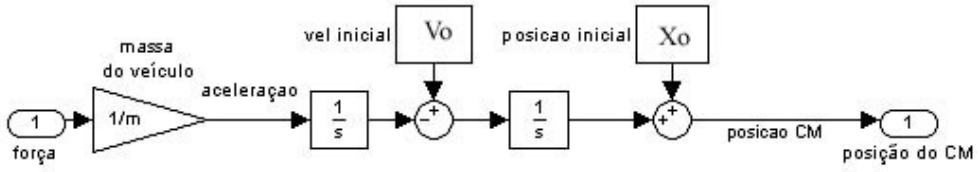
Apêndice A
Código em Matlab dos modelos criados

A.1
Choque central frontal veículo deformável – barreira rígida

A.1.1
barreirarígida.mdl







A.1.2
dados.m

```
%fornece dados iniciais
%dados inicialmente testado para veiculo medio
clear all;
global  ld lt barreira K Ro m
m=1893; %massa do veiculo
ld=1; %cm a dianteira
barreira=ld; % barreira
%Ro=31324.65;
%K=2872521.6;
Ro=5.946e4; %constante para calculo do amortecimento
K=81.7*Ro; %rigidez frontal do veiculo
barreirarigida
```

A.1.3
geometria.m

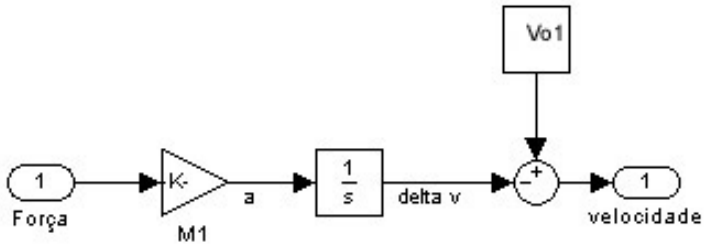
```
function def=geometria(Xcm)
global ld barreira
    naodef=Xcm+ld;
    %coordenadas do veiculo deformado
if Xcm+ld>barreira
  A=barreira;
else
  A=naodef;
end
%Deformacoes totais em cada regiao
def=norm(naodef-A);
```

A.1.4
amortecedor.m

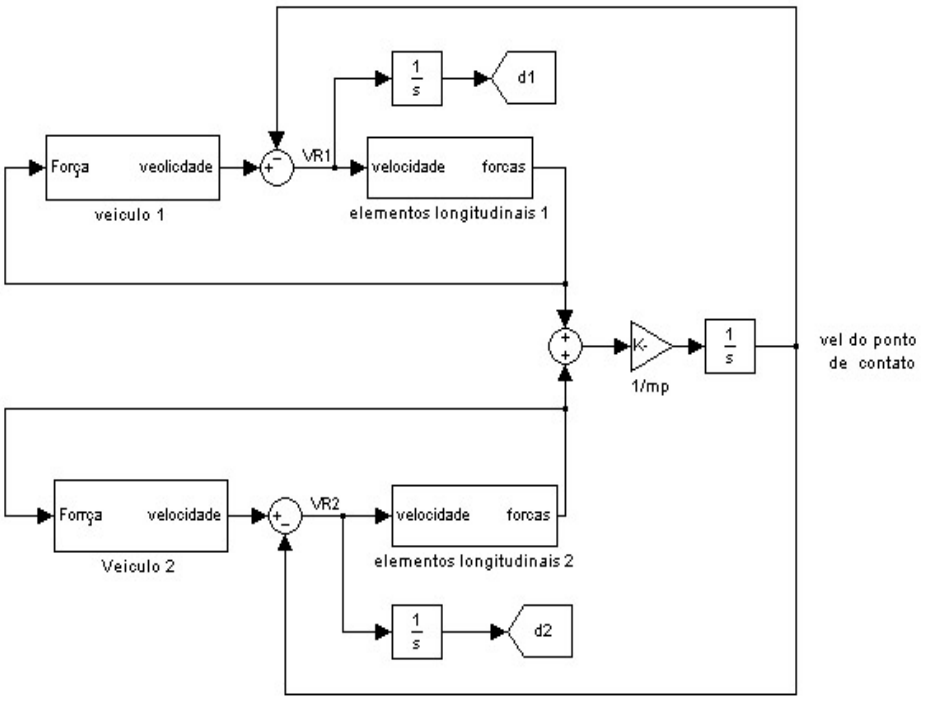
```
function V=amortecedor(X,F) %calcula o coeficiente de
amortecimento e devolve velocidade de deslocamento do amortecedor
global Ro
if F<3000
    R=10e6;
else
    R=1.2*Ro-(Ro*X^2)/2;
end
V=F/R;
```

A.2
Choque frontal central entre dois veículos

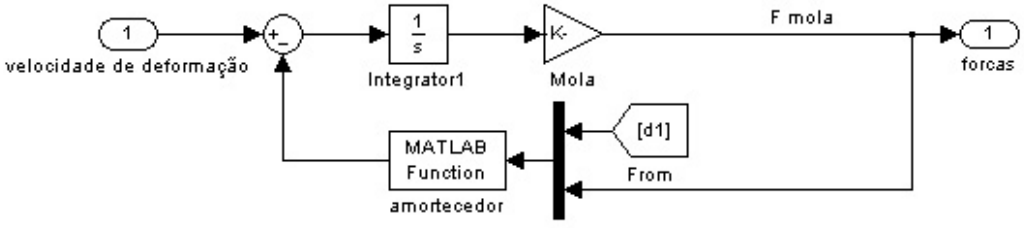
A.2.1
choque.mdl



Representação da dinâmica do veículo 1.



Modelo unidimensional para colisão central entre dois veículos.



Representação dos elementos flexíveis.

A.2.2
amortecedor1.m

```
function v=amortecedor1(u)
F=u(2); d=u(1);Co=5.946e4;%52536;
if abs(F)>=3000
    v=F/(1.2*Co-(Co*d^2)/2);
else
    v=0;
end
```

A.2.3
amortecedor2.m

```
function v=amortecedor2(u)
F=u(2); d=u(1);Co=3.580e4;
if abs(F)>=3000
    v=F/(1.2*Co-(Co*d^2)/2);
else
    v=0;
end
```

A.3
Colisões bidimensionais entre dois veículos

A.3.1
colisao.m

```
% arquivo colisao.m
% arquivo pricipal para programa de colisoes veiculares

clear all;
global P1 P2 %contornos dos veiculos em coordenadas globais,
contornos originais
global CM   %posiçoes dos centros de massa
global impacto S  %tipo de impacto e areas atingidas
global nd nl  %numero de pontos de discretizaçao dos veiculos
global plocal1 plocal2 plocal1o plocal2o %contornos dos veiculos
em coordenadas locais
global b ld lt m I K1 K2 Co1 Co2  %dados dos veiculos
global teste1 teste2   %auxiliares
echo on
entre com o tipo de impacto (default impacto=1):
1-central;
2-offset;
3-obliquo;
keyboard;
entre com a area atingida do veiculo 1(default S(1)=1):
1-frente;
2-traseira;
3-lateral;
4-frente-lateral;
5-traseira-lateral;
keyboard;
entre com a area atingida do veiculo 2(default S(2)=1):
1-frente;
```

```
2-traseira;
3-lateral;
4-frente-lateral;
5-traseira-lateral;
keyboard;
echo off;
veiculos;
posicionamento_inicial;
divisao_inicial (1);
divisao_inicial (2);
P1=divisao (1,CM(1:3));
plocal1=plocal1o;
P2=divisao(2,CM(4:6));
plocal2=plocal2o;
teste1=zeros(1,length(P1));  %inicializaçao de teste1
teste2=zeros(1,length(P2));  %inicializaçao de teste 2
rigidez;

 %criar figura 1
figure(1);
scrsz=get(0,'ScreenSize');
set(1,'Position',[scrsz(1) scrsz(1) 2*scrsz(3)/5 2*scrsz(4)/5]);
%limites dos eixos
title('veiculos deformados');

 %criar figura 2
figure(2);
set(2,'Position',[3*scrsz(3)/5 scrsz(1) 2*scrsz(3)/5
2*scrsz(4)/5]);   %limites dos eixos
figure(2);
title('veiculos nao deformados');
choque;
```

A.3.2
veiculos.m

```
% veiculos.m
%fornece dados dos veiculos
global b ld lt m I kf kl kt Cof Col Cot

%veiculo 1
m(1)=900; %massa do veiculo
b(1)=1.5;    % bitola do veiculo
lt(1)=3; %distancia do cm a traseira
ld(1)=1; %distancia do cm a dianteira
I(1)=2207%momento de inercia
Cof(1)=5.946e4; %constante para calculo do amortecimento frontal
Col(1)=3.525e4; %constante para calculo do amortecimento lateral
Cot(1)=4.178e3; %constante para calculo do amortecimento  traseiro
kf(1)=81.7*Cof(1); %rigidez frontal do veiculo
kl(1)=81.7*Cof(1); %rigidez lateral do veiculo
kt(1)=81.7*Cof(1); %rigidez traseira do veiculo

%veiculo 2
m(2)=1338; %massa do veiculo
b(2)=0.4;    % bitola do veiculo
lt(2)=0.25; %distancia do cm a traseira
ld(2)=0.25; %distancia do cm a dianteira
I(2)=2207%momento de inercia
Cof(2)=10e7; %constante para calculo do amortecimento frontal
```

```
Col(2)=10e7; %constante para calculo do amortecimento lateral
Cot(2)=3.756e4; %constante para calculo do amortecimento  traseiro
kf(2)=81.7*Cof(2); %rigidez frontal do veiculo
kl(2)=81.7*Cof(2); %rigidez lateral do veiculo
kt(2)=81.7*Cof(2); %rigidez traseira do veiculo
```

A.3.3
posicionamento_inicial.m

```
%posicionamento_inicial.m
%posiciona os veiculos no plano
global v CM impacto
Xcm(1)=0;Ycm(1)=0;fi(1)=0;Ycm(2)=Ycm(1); %valores default
echo on
% entre com a pocicao inicial
%veiculo1 (localizado em (0,0)):
keyboard
%veiculo 2:
echo off
switch impacto
case 1 % impacto central
    echo on;
    %entre com a coordenada Xcm2(Xcm(2)=? m)
    keyboard
    echo off;
case 2 %impacto com offset
    echo on
    %entre com as coordenadas Xcm e Ycm(Xcm(2)=?;Ycm(2)=?)
    keyboard
    echo off;
case 3 %impacto obliquo
    echo on;
    %entre com as coordenadas Xcm e Ycm(Xcm(2)=?;Ycm(2)=?,fi(2)=?)
    keyboard;
    echo off;
end
CM2=[Xcm(2) Ycm(2) fi(2)];
CM1=[Xcm(1) Ycm(1) fi(1)];
CM=[CM1 CM2]; %coordenadas dos centros de massa dos veiculos
```

A.3.4
divisao_inicial.m

```
function divisao_inicial (veiculo)
%permite entrar com o numero de pontos para discretizaçao dos
veiculos
global  S nd nl
echo off
if S(veiculo)<=3 %uma so area atingida
    echo on;
    disp('numero de pontos para a area atingida no veiculo
(nd(veiculo)=10)')
    keyboard
else
    if S(veiculo)==4  %duas areas atingidas (frontal e lateral)
        echo on
        disp ('numero de pontos para as areas frontal e lateral do
veiculo (dianteira nd(veiculo)=5; lateral: nl(veiculo) =5)')
        keyboard;
```

```
    else %duas areas atingidas (traseira e lateral)
         echo on
        disp('numero de pontos para as areas traseira e lateral do
veiculo (traseira nd(veiculo)=5;lateral nl(veiculo)=5)')
        keyboard;
    end
end
echo off;
```

A.3.5
divisao.m

```
function P=divisao (veiculo,CM)
%discretiza as areas envolvidas

global  S ld lt b nd nl P1or P2or plocal1o plocal2o

%vertices do retangulo
A=[ld(veiculo);-b(veiculo)/2];
B=[ld(veiculo);b(veiculo)/2];
C=[-lt(veiculo);b(veiculo)/2];
D=[-lt(veiculo);-b(veiculo)/2];

switch S(veiculo) %tipo de impacto
case 1 %impacto frontal
    for i=1:nd(veiculo)
        P(1:2,i)=A+[0;((i-1)*b(veiculo)/(nd(veiculo)-1))];
        %divide a regiao frontal
    end
    P=[P C D]; %matriz de pontos de contorno
case 2 %impacto traseiro
    for i=1:nd(veiculo)
        P(1:2,i)=C-[0;(i-1)*(b(veiculo)/(nd(veiculo)-1))];
        %divide a regiao traseira
    end
    P=[A B P]; %matriz de pontos de contorno
case 3 %impacto lateral
    for i=1:nd(veiculo)
        P(1:2,i)=B-[(i-1)*(ld(veiculo)+lt(veiculo))/(nd(veiculo)-
1);0];
        %divide a lateral esquerda
        P(1:2,nd(veiculo)+i)=D+[(i-
1)*(ld(veiculo)+lt(veiculo))/(nd(veiculo)-1);0];
        %divide a lateral direita
    end
case 4 %frontal-lateral
    for i=0:nd(veiculo)-1
        P(1:2,i+1)=A+[0;(i)*(b(veiculo)/(nd(veiculo)-1))];
        %divide a regiao frontal
    end
    for i=1:nl(veiculo)-1
        P(1:2,nd(veiculo)+i)=B-
[i*(ld(veiculo)+lt(veiculo))/(nl(veiculo)-1);0];
        %divide a lateral esquerda
        P(1:2,nd(veiculo)+nl(veiculo)+i-1)=D+[(i-
1)*(ld(veiculo)+lt(veiculo))/(nl(veiculo)-1);0];
        % divide a lateral direita
    end
case 5 %traseira-lateral
    for i=1:nl(veiculo)
```

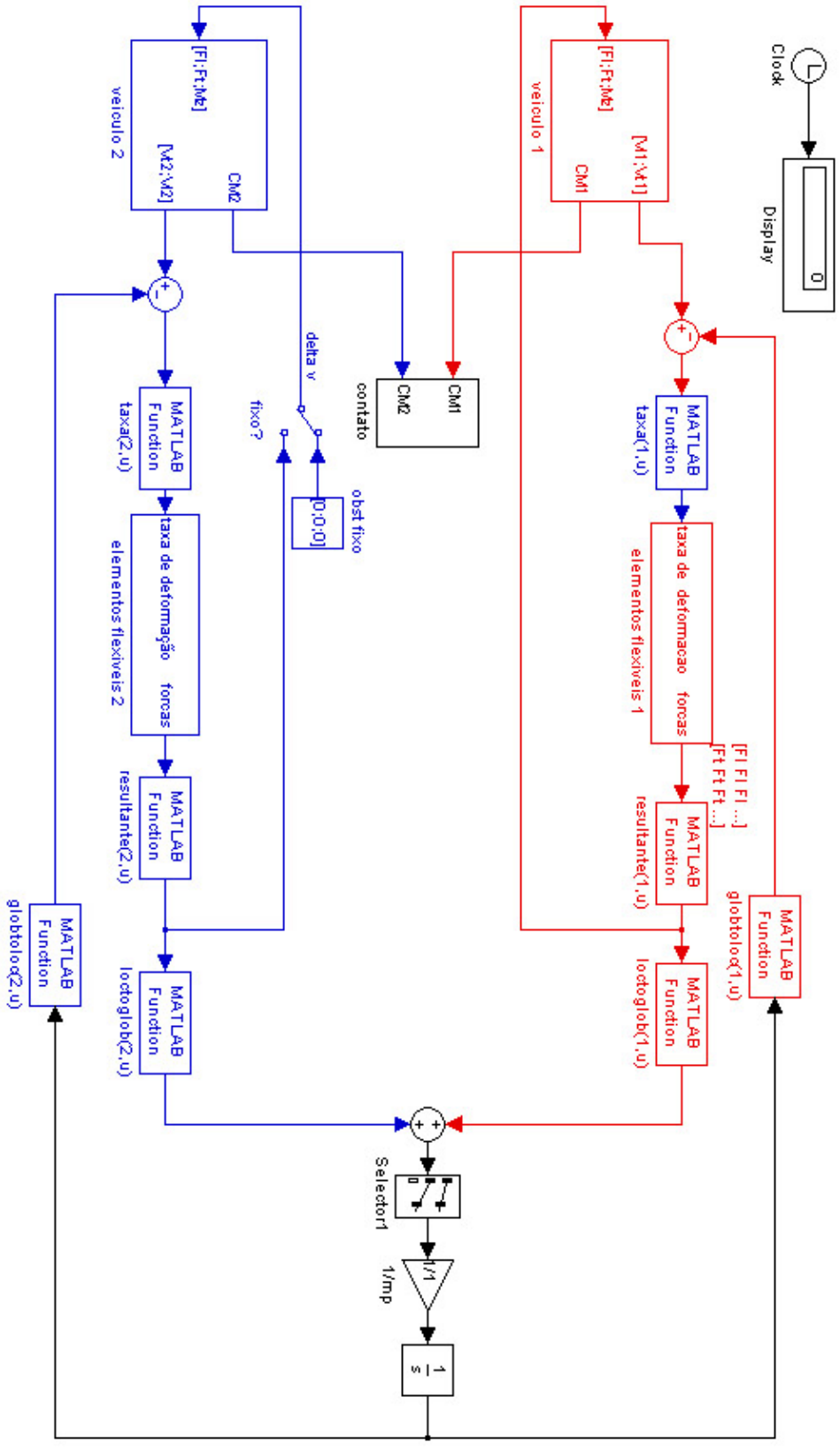
```
        P(1:2,i)=B-[(i-1)*(ld(veiculo)+lt(veiculo))/(nd(veiculo)-
1);0];
        %divide a lateral esquerda
        P(1:2,nl(veiculo)+nd(veiculo)-2+i)=D+[(i-
1)*(ld(veiculo)+lt(veiculo))/(nd(veiculo)-1);0];
        %divide a lateral direita
    end
    for i=1:nd(veiculo)-2
        P(1:2,nd(veiculo)+i)=C-[0;i*(b(veiculo)/(nd(veiculo)-1))];
% divide a traseira
    end
end

%iniciaçao da matriz com coordenadas locais iniciais
if veiculo==1
    plocal1o=P;
else
    plocal2o=P;
end

%transformaçao para coordenadas globais
G=[cos(CM(3)) -sin(CM(3));
    sin(CM(3)) cos(CM(3))];       %matriz de transformaçao
P=G*P;
P(1,:)=P(1,:)+CM(1);P(2,:)=P(2,:)+CM(2);
```

A.3.6
rigidez.m

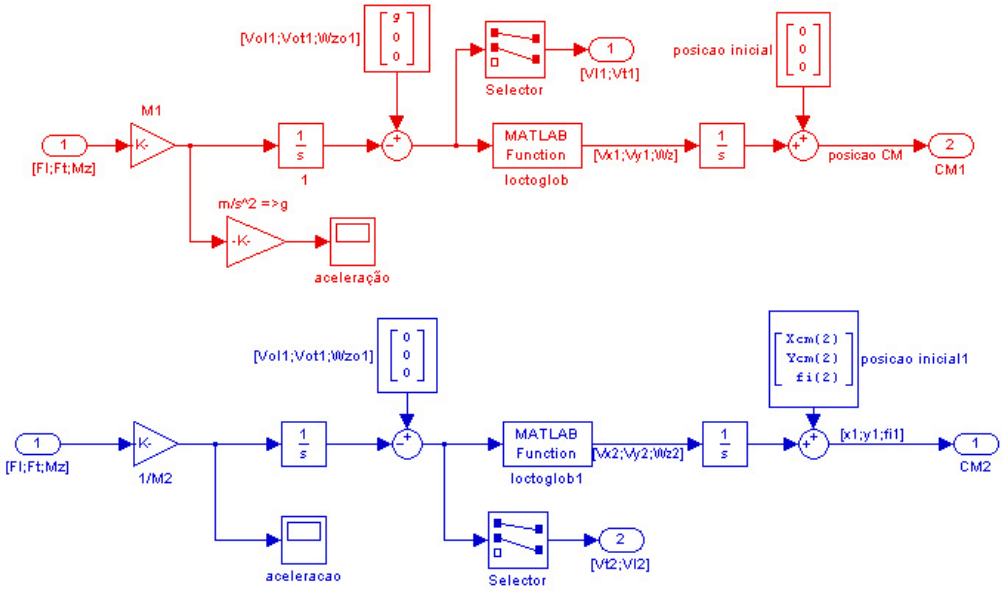
```
%discretizaçao dos parametros de rigidez
function rigidez
global  S nd nl kf kl kt Cof Col Cot K1 K2 Co1 Co2

%matriz com os coeficientes de rigidez e de amortecimento
discretizados para veiculo 1:
KC=rig(1);
%matriz com os coeficentes de rigidez dos elementos discretizados
para veiculo 1:
K1=[KC(1:length(KC)/2);KC(1:length(KC)/2)];
%matriz com os coeficentes de amortecimento discretizados para
veiculo 1:
Co1=[KC(length(KC)/2+1:length(KC));KC(length(KC)/2+1:length(KC))];

%matriz com os coeficientes de rigidez e de amortecimento
discretizados para veiculo 2:
KC=rig(2);
%matriz com os coeficentes de rigidez dos elementos discretizados
para veiculo 2:
K2=[KC(1:length(KC)/2);KC(1:length(KC)/2)];
%matriz com os coeficentes de amortecimento discretizados para
veiculo 2:
Co2=[KC(length(KC)/2+1:length(KC));KC(length(KC)/2+1:length(KC))];

%calcula matriz linha com coeficientes de rigidez e de
amortecimento
function KC=rig(veiculo)
global  S nd nl kf kl kt Cof Col Cot
switch S(veiculo)
case 1 %impacto frontal
    n=nd(veiculo);
```

```
    K(1:n)=kf(veiculo)/n;
    C(1:n)=Cof(veiculo)/n;
    K=[K 0 0];
    C=[C 0 0];
case 2 %impacto traseiro
    n=nd(veiculo);
    K(1:n)=kt(veiculo)/n;
    C(1:n)=Cot(veiculo)/n;
    K=[0 0 K];
case 3 %impacto lateral
    n=nd(veiculo)
    K(1:2*nd)=kl(veiculo)/n;
    C(1:2*nd)=Col(veiculo)/n;
case 4 %frontal-lateral
    n1=nd(veiculo);n2=nl(veiculo);
    K(1:n1)=kf(veiculo)/n1;
    C(1:n1)=Cof(veiculo)/n1;
    K(n1+1:n1+2*n2-2)=kl(veiculo)/(n2-1);
    C(n1+1:n1+2*n2-2)=Col(veiculo)/(n2-1);
case 5 %traseira-lateral
    n1=nd(veiculo);
    n2=nl(veiculo);
    K(1:n2-1)=kl(veiuclo)/n1;
    C(1:n2-1)=Col(veiuclo)/n1;
    K(n2:n2+n1)=kt(veiuclo)/n1;
    C(n2:n2+n1)=Cot(veiuclo)/n1;
    K(n2+n1+1:n1+2*n2-2)=kl(veiuclo)/n1;
    C(n2+n1+1:n1+2*n2-2)=Col(veiuclo)/n1;
end
KC=[K C];
```

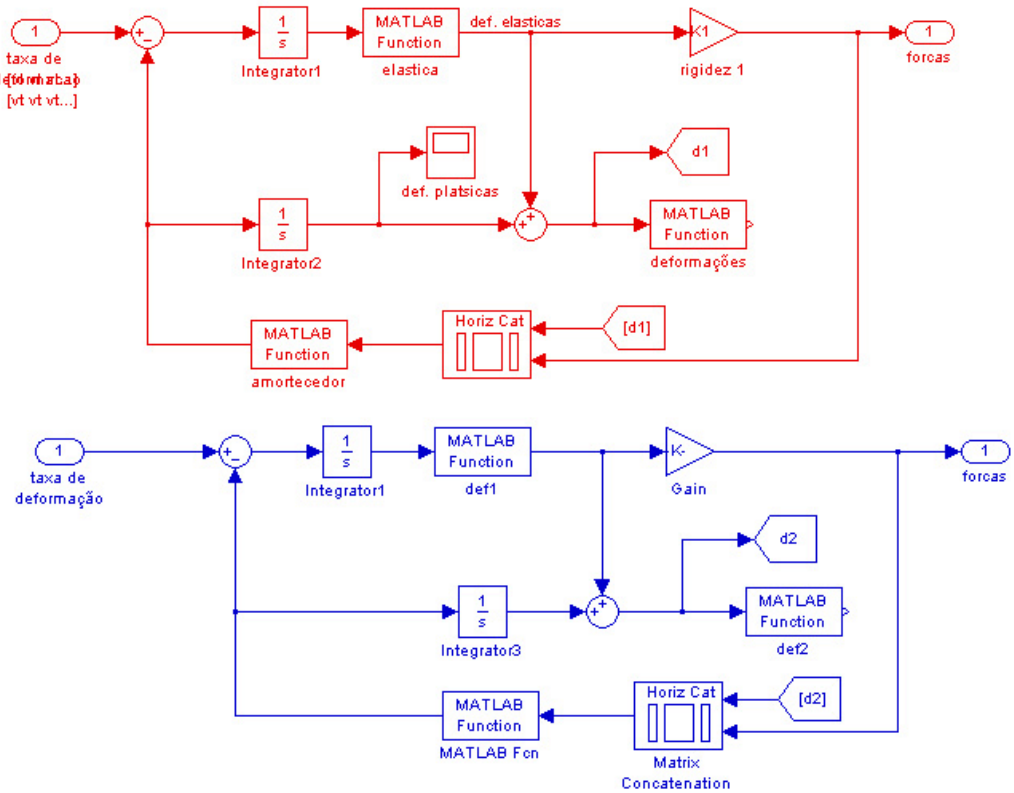
A.3.7
choque.mdl



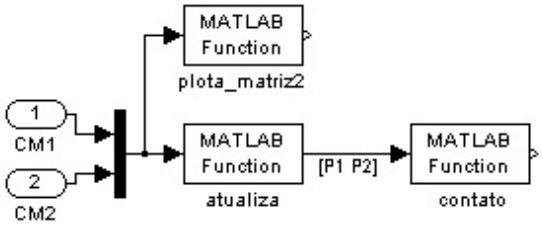
A.3.7.1
veiculo1 e veiculo2



A.3.7.2
elementos flexíveis



A.3.7.3
contato



A.3.8
loctoglob.m

```
%transforma coordenadas locais em globais
function[vet]=loctoglob(veiculo,V)

global CM %coordenadas dos dois veiculos
if veiculo==1
   fi=CM(3); %azimute do veiculo 1
else
    fi=CM(6);%azimute do veiculo 2
end
u=V(1);v=V(2);
vet(1,1)=u*cos(fi)-v*sin(fi);
vet(2,1)=u*sin(fi)+v*cos(fi);
if length(V)==3
    vet(3,1)=V(3);
end
```

A.3.9
globtoloc.m

```
%transforma coordenadas globais em locais
function [vet]=globtoloc(veiculo,V)

global CM
u=V(1);v=V(2);
if veiculo==1
   fi=CM(3);
else
    fi=CM(6);
end
vet(1)=u*cos(fi)-v*sin(fi);
vet(2)=u*sin(fi)+v*cos(fi);
vet=vet(:);
```

A.3.10
taxa.m

```
%multiplica a velocidade relativa pelas matrizes teste1 ou teste2
function vr=taxa(veiculo,u)
global teste1 teste2
if veiculo==1
```

```
    vr=u*teste1;
else
    vr=u*teste2;
end
```

A.3.11
resultante.m

```
function F=resultante(veiculo,f)
%retorna  somatorio de Fx Fy e o momento em z em ambos os veiculos

global plocal1 plocal2 CM teste1 teste2;
%aplicaçao da força nos pontos de contato
if veiculo==1
    f=f.*[teste1;teste1];
else
    f=f.*[teste2;teste2];
end
%forças resultantes (somatorio):
F=sum(f,2);

%calculo dos momentos resultantes (força x distancia ao CM):
F(3)=0;
if veiculo==1
    for i=1:length(f)
        F(3)=F(3)+(plocal1(1,i)*f(2,i)-plocal1(2,i)*f(1,i));
    end
else
    for i=1:length(f)
        F(3)=F(3)+(plocal2(1,i)*f(2,i)-plocal2(2,i)*f(1,i));
    end
end
```

A.3.12
elastica.m

```
function vr=elastica(veiculo,u)
%anula as deformaçoes elastica dos pontos sem contato

vr=u.*(u>0);
```

A.3.13
isinploly.m

```
function  isin = isinpoly(x,y,xp,yp)
 % fecha o contorno do poligono
xp = [xp(:); xp(1)];
yp = [yp(:); yp(1)];
sz = size(x);
x = x(:); y = y(:);

lp = length(xp); l = length(x);
ep = ones(1,lp);
e = ones(1,l);

 % calcula a soma dos angulos entre os vetores do ponto aos
vertices do poligono
```

```
A = diff(atan2(yp(:,e)-y(:,ep)',xp(:,e)-x(:,ep)'))/pi; %colunas
contem as diferenças entre os angulos consecutivos
A = A+2*((A<-1)-(A>1)); %ajusta angulos obtusos
isin = any(A==1)-any(A==-1); %verifica colunas que possuem angulo
pi ou -pi
isin = (abs(sum(A))-isin)/2; %transforma 2pi em 1

 % verificacao de pontos pertencentes ao limite
 A = (yp(:,e)==y(:,ep)')&(xp(:,e)==x(:,ep)');
fnd = find(any(A));
isin(fnd) = .5*ones(size(fnd));
isin = round(isin*2)/2;

 % redimensionamento do output
isin = reshape(isin,sz(1),sz(2));
```

A.3.14
amortecedores.m

```
%calcula o coeficiente de amortecimento e devolve velocidade de
deslocamento do amortecedor
function V=amortecedores(veiculo,u)
global Co1 Co2

%determinaçao da matriz de amortecimento de acordo com o veiculo
if veiculo==1
    Co=Co1;
else
    Co=Co2;
end

%desmembramento de u em deformaçoes e forças:
deformacoes=u(1:2,1:length(Co));
forcas=u(1:2,length(Co)+1:length(u));

%divisao das forças pelas deformaçoes
 for j=1:length(forcas)
     for i=1:2
         if abs(forcas(i,j))>=1000 & Co(i,j)~=0
             V(i,j)=forcas(i,j)/(1.2*Co(i,j)-
(Co(i,j)*deformacoes(i,j)^2)/2);
         else
             V(i,j)=0;
         end
     end
 end
```

A.3.15
deformaçoes

```
function v=deformacoes(veiculo,u)
%efetua atualizaçao da matriz de coordenadas locais de acordo com
o veiculo
global plocal1 plocal2 plocal1o plocal2o
if veiculo==1
    plocal1=plocal1o-u;
else
    plocal2=plocal2o-u;
end
```

A.3.16
atualiza(u)

```
%teste de contato veirificando distancia do ponto ao lado atingido
function P=atualiza(u)
global lt ld b P1 P2 plocal1 plocal2 CM
CM=u;
G=[cos(CM(3)) -sin(CM(3));
    sin(CM(3)) cos(CM(3))];
P1=G*plocal1;
P1(1,:)=P1(1,:)+CM(1);P1(2,:)=P1(2,:)+CM(2); %atualiza P1

G=[cos(CM(6)) -sin(CM(6));
    sin(CM(6)) cos(CM(6))];
P2=G*plocal2;
P2(1,:)=P2(1,:)+CM(4);P2(2,:)=P2(2,:)+CM(5); %atualiza P2
figure(1)
P=[P1 P2];
plota_matriz(P);
```

A.3.17
contato(u)

```
function contato(u)
global teste1 teste2 P1
teste1=isinpoly(u(1,1:length(P1)),u(2,1:length(P1)),u(1,length(P1)
+1:length(u)),u(2,length(P1)+1:length(u)));
teste2=isinpoly(u(1,length(P1)+1:length(u)),u(2,length(P1)+1:lengt
h(u)),u(1,1:length(P1)),u(2,1:length(P1)));
```

A.3.18
plota_matriz(u)

```
%plota uma matriz controlada
function plota_matriz(P)
global P1
figure(1);
clf;
axis([-3 10 -4 4]);
line(P(1,1:length(P1)),P(2,1:length(P1)),'color','r');
line([P(1,length(P1)) P(1,1)],[P(2,length(P1))
P(2,1)],'color','r');
line(P(1,length(P1)+1:length(P)),P(2,length(P1)+1:length(P)),'colo
r','b');
line([P(1,length(P)) P(1,length(P1)+1)],[P(2,length(P))
P(2,length(P1)+1)],'color','b');
```

A.3.19
plota_matriz2(u)

```
%plota uma matriz controlada
function plota_matriz2(CM)
global lt ld b
A1=loctoglob(1,[ld(1);-b(1)/2])+CM(1:2);
B1=loctoglob(1,[ld(1);b(1)/2])+CM(1:2);
C1=loctoglob(1,[-lt(1);b(1)/2])+CM(1:2);
```

```
D1=loctoglob(1,[-lt(1);-b(1)/2])+CM(1:2);
A2=loctoglob(2,[ld(2);-b(2)/2])+CM(4:5);
B2=loctoglob(2,[ld(2);b(2)/2])+CM(4:5);
C2=loctoglob(2,[-lt(2);b(2)/2])+CM(4:5);
D2=loctoglob(2,[-lt(2);-b(2)/2])+CM(4:5);
figure(2);
clf;
axis([-3 10 -4 4]);
line([A1(1) B1(1) C1(1) D1(1) A1(1)],[A1(2) B1(2) C1(2) D1(2)
A1(2)],'color','r');
line([A2(1) B2(1) C2(1) D2(1) A2(1)],[A2(2) B2(2) C2(2) D2(2)
A2(2)],'color','b');
```