Trabajo Fin de Grado
Grado en Ingenieria en Tecnologías Industriales

Análisis Computacional Cinemático de Mecanismos Planos: Código Fuente

Autor: Jesús Medrán Castro
Tutor: Carmen Madrigal Sánchez

Dep. Ingeniería Mecánica y Fabricación
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016

Trabajo Fin de Grado
Grado en Ingeniería en Tecnologías Industriales

Análisis Computacional Cinemático de Mecanismos Planos: Código Fuente

Autor:
Jesús Medrán Castro

Tutor:
Carmen Madrigal Sánchez
Profesor titular

Dep. de Ingeniería Mecánica y Fabricación
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2016

Proyecto Fin de Carrera: Análisis Computacional Cinemático de Mecanismos Planos: Código Fuente

Autor: Jesús Medrán Castro

Tutor: Carmen Madrigal Sánchez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2016

El Secretario del Tribunal

A mi familia

A mis maestros

Resumen

Este documento se presenta como un texto adicional al Trabajo de Fin de Grado: *Análisis Computacional Cinemático de Mecanismos Planos* escrito por el alumno Jesús Medrán Castro. En las siguientes páginas se recoge el código fuente del programa *CAC 2D* escrito integramente en *Matlab*. Se presentan todas las funciones utilizadas ordenadas alfabeticamente.

Índice

ACELERELA.M

```
function [Meca] = AceleRela(Meca)
%% Función que implementa el método de las aceleraciones relativas %%

   % Creación de Meca.Cuerpo(i).A
     Meca.Cuerpo(1).A=0;
      for i=[1:1:Meca.Nbody-1]
          Meca.Cuerpo(i+1).A=Meca.SolAcel(i);
      end
   %

   % Implementación de Meca.SolVeloc=A41;A81..... en Meca.Rela().Ax Ay
      APP=[];
      for i=[1:1:length(Meca.Grupo)]
          fin=Meca.Grupo(i).G(end);
          fin=Meca.Lazo(fin+1);
          if fin==1
          else
              APP=[APP,fin];
          end
      end
    if length(APP)==0
        APP=0;
    end
    APP;
%     % Restrinción en caso de bucle abierto [Pruebas]
%        if Meca.Bucle==1
%            APP(end)=[];
%        end
%     %
    Elim=1;
    k=1;
  if APP==0
  else
    for i=[Meca.Nbody:1:length(Meca.SolAcel)]
        if k>length(APP)
        else
            R=APP(k);
        if Elim==1
            Meca.Rela(R).Ax=Meca.SolAcel(i);
            Elim=Elim+1;
        else
            Meca.Rela(R).Ay=Meca.SolAcel(i);
            Elim=1;
            k=k+1;
        end
        end
    end
  end
   %

%%% Código nuevo inicio %%%
MAR=zeros(2,Meca.Nbody);
   % Caso de no existir pares prismáticos NPP=0
      if Meca.NPP==0
          % Calcular matriz MT
            MT=[];
```

```
MW=[];
MV=[];
for i=[1:1:length(Meca.Rela)]
   Rf=Meca.Lazo(i);
   Ri=Meca.Lazo(i+1);
   Posfin=[Meca.Rela(Rf).x;Meca.Rela(Rf).y];
   Posini=[Meca.Rela(Ri).x;Meca.Rela(Ri).y];
   T=Posfin-Posini;
   T=[-T(2);T(1)];
   MT=[MT,T];
end
for i=[1:1:length(MT(1,:))-1]
    W=Meca.Cuerpo(i+1).W;
    T=MT(:,i);
    T=[T(2);-T(1)];
    MW(:,i)=T.*(W.^2);
end
MW=[[0;0] MW];
for i=[1:1:length(MT(1,:))-1]
   A=Meca.Cuerpo(i+1).A;
   MT(:,i)=MT(:,i).*A;
end
MT(:,end)=[];
MT=[[0;0] MT];
k=1;
for i=[1:1:length(Meca.Rela)]
    if Meca.Rela(i).Codig==3
      Ri=Meca.Radios(k,1);
      Rj=Meca.Radios(k,2);
      landa=Meca.DirecPRSD(k,:);
      landa=(landa)./norm(landa)';
      wi=Meca.Cuerpo(Meca.Rela(i).Cuerpos(1)).W;
      wj=Meca.Cuerpo(Meca.Rela(i).Cuerpos(2)).W;
        if Meca.Radios(k,3)==1
           aij=((wi-wj).^2).*((Ri.*Rj)./(Ri-Rj)).*landa';
        end
        if Meca.Radios(k,3)==2
            aij=((wi-wj).^2).*((Ri.*Rj)./(Ri+Rj)).*landa';
        end
        if Meca.Radios(k,3)==3
            aij=((wi-wj).^2).*Ri.*landa';
        end
        if Meca.Radios(k,3)==4
            aij=((wi-wj).^2).*Rj.*landa';
        end
        V=[Meca.Rela(i).Vx;Meca.Rela(i).Vy];
        V=[-V(2);V(1)];
        MV(:,i)=aij+2*wi.*V;
        k=k+1;
    else
        MV(:,i)=[0;0];
    end
end
MAR=-MT-MW-MV;
%
% Igualación de los extremos a cero si es par de revolución
 if Meca.LazoPares(1)==1
     MAR(:,1)=0;
 end
 if Meca.LazoPares(end)==1
     MAR(:,end)=0;
 end
%
```

```
        % Proceso de suma
           for i=[2:1:length(MAR(1,:))]
               MAR(:,i)=MAR(:,i)+MAR(:,i-1);
           end
        %
        % Colocación MAR en Meca.Rela(i).Ax y Meca.Rela(i).Ay
               MAR(:,1)=0;
               MAR(:,end)=0;
           for i=[1:1:length(MAR(1,:))]
               Meca.Rela(i).Ax=MAR(1,i);
               Meca.Rela(i).Ay=MAR(2,i);
           end
        %
    end
%
% Caso de solo un par prismático NPP=1
    if Meca.NPP==1
        % Calcular matriz MT
          MT=[];
          MV=[];
          MW=[];
          for i=[1:1:length(Meca.Rela)]
             Ri=Meca.Lazo(i);
             Rf=Meca.Lazo(i+1);
             Posfin=[Meca.Rela(Rf).x;Meca.Rela(Rf).y];
             Posini=[Meca.Rela(Ri).x;Meca.Rela(Ri).y];
             T=Posfin-Posini;
             T=[-T(2);T(1)];
             MT=[MT,T];
          end
          for i=[1:1:length(MT(1,:))-1]
              W=Meca.Cuerpo(i+1).W;
              T=MT(:,i);
              T=[T(2);-T(1)];
              MW(:,i)=T.*(W.^2);
          end
          MW=[[0;0] MW];
          for i=[1:1:length(MT(1,:))-1]
             A=Meca.Cuerpo(i+1).A;
             MT(:,i)=MT(:,i).*A;
          end
          MT(:,end)=[];
          MT=[[0;0] MT];
          k=1;
          for i=[1:1:length(Meca.Rela)]
              if Meca.Rela(i).Codig==3
                Ri=Meca.Radios(k,1);
                Rj=Meca.Radios(k,2);
                landa=Meca.DirecPRSD(k,:);
                landa=(landa)./norm(landa)';
                wi=Meca.Cuerpo(Meca.Rela(i).Cuerpos(1)).W;
                wj=Meca.Cuerpo(Meca.Rela(i).Cuerpos(2)).W;
                  if Meca.Radios(k,3)==1
                     aij=((wi-wj).^2).*((Ri.*Rj)./(Ri-Rj)).*landa';
                  end
                  if Meca.Radios(k,3)==2
                      aij=((wi-wj).^2).*((Ri.*Rj)./(Ri+Rj)).*landa';
                  end
                  if Meca.Radios(k,3)==3
                      aij=((wi-wj).^2).*Ri.*landa';
                  end
                  if Meca.Radios(k,3)==4
```

```
                    aij=((wi-wj).^2).*Rj.*landa';
                end
                V=[Meca.Rela(i).Vx;Meca.Rela(i).Vy];
                V=[-V(2);V(1)];
                MV(:,i)=aij+2*wi.*V;
                k=k+1;
            else
                MV(:,i)=[0;0];
            end
        end
        MAR=MT-MW-MV;
    %
    % Igualación de los extremos a cero si es par de revolución
        if Meca.LazoPares(1)==1
            MAR(:,1)=0;
        end
        if Meca.LazoPares(end)==1
            MAR(:,end)=0;
        end
    %
    % Proceso de suma
        NVPP=[];
        for i=[1:1:length(Meca.LazoPares)]
            if Meca.LazoPares(i)==2
                NVPP=[NVPP,i];
            end
        end
        NVPP;
        % Suma-> Caso de NPP en primera posición
        if NVPP==1
            MAR=-MAR;
            for i=[length(MAR(1,:))-1:-1:1]
                MAR(:,i)=MAR(:,i)+MAR(:,i+1);
            end
        end
        %
        % Suma-> Caso de NPP en última posición
        if NVPP==Meca.Nrest
            for i=[2:1:length(MAR(1,:))]
                MAR(:,i)=MAR(:,i)+MAR(:,i-1);
            end
        end
        %
        % Suma-> Caso de NPP en cualquier posición
        if NVPP~=1 && NVPP~=Meca.Nrest
            for i=[2:1:NVPP]
                MAR(:,i)=MAR(:,i)+MAR(:,i-1);
            end
            for i=[length(MAR(1,:))-1:-1:NVPP+1]
                MAR(:,i)=-(MAR(:,i)+MAR(:,i+1));
            end
        end
        %
    %
    % Colocación MAR en Meca.Rela(i).Ax y Meca.Rela(i).Ay
            MAR(:,1)=0;
        for i=[1:1:length(MAR(1,:))]
            Meca.Rela(i).Ax=MAR(1,i);
            Meca.Rela(i).Ay=MAR(2,i);
        end
    %
end
%
```

```
% Caso de más de un par prismático NPP>=2
   if Meca.NPP>=2
       % Calcular matriz MT
         MT=[];
         for i=[1:1:length(Meca.Rela)]
            Ri=Meca.Lazo(i);
            Rf=Meca.Lazo(i+1);
            Posfin=[Meca.Rela(Rf).x;Meca.Rela(Rf).y];
            Posini=[Meca.Rela(Ri).x;Meca.Rela(Ri).y];
            T=Posfin-Posini;
            T=[-T(2);T(1)];
            MT=[MT,T];
         end
         MT(:,end)=[];
         MT=[[0;0] MT];
         for i=[1:1:length(MT(1,:))]
             W=Meca.Cuerpo(i).W;
             T=MT(:,i);
             T=[T(2);-T(1)];
             MW(:,i)=T.*(W.^2);
         end
         for i=[1:1:length(MT(1,:))-1]
            A=Meca.Cuerpo(i+1).A;
            MT(:,i)=MT(:,i).*A;
         end
         APP;
         for i=[1:1:length(APP)]
            MT(:,APP(i))=0;
         end
       %
       % Introducir las A calculadas de Meca.SolAce
          for i=[1:1:length(APP)]
             MAR(1,APP(i))=Meca.Rela(APP(i)).Ax;
             MAR(2,APP(i))=Meca.Rela(APP(i)).Ay;
          end
       %
         k=1;
         for i=[1:1:length(Meca.Rela)]
             if Meca.Rela(i).Codig==3
               Ri=Meca.Radios(k,1);
               Rj=Meca.Radios(k,2);
               landa=Meca.DirecPRSD(k,:);
               landa=(landa)./norm(landa)';
               wi=Meca.Cuerpo(Meca.Rela(i).Cuerpos(1)).W;
               wj=Meca.Cuerpo(Meca.Rela(i).Cuerpos(2)).W;
                 if Meca.Radios(k,3)==1
                    aij=((wi-wj).^2).*((Ri.*Rj)./(Ri-Rj)).*landa';
                 end
                 if Meca.Radios(k,3)==2
                     aij=((wi-wj).^2).*((Ri.*Rj)./(Ri+Rj)).*landa';
                 end
                 if Meca.Radios(k,3)==3
                     aij=((wi-wj).^2).*Ri.*landa';
                 end
                 if Meca.Radios(k,3)==4
                     aij=((wi-wj).^2).*Rj.*landa';
                 end
                 V=[Meca.Rela(i).Vx;Meca.Rela(i).Vy];
                 V=[-V(2);V(1)];
                 MV(:,i)=aij+2*wi.*V;
                 k=k+1;
             else
```

```
                    MV(:,i)=[0;0];
                end
            end
%             MT
%             MW
%             MV
%             MAR
            MV(:,end)=[];
            MV=[[0;0] MV];
          MAR=MT-MV-MW+MAR;
          % Igualación de los extremos a cero si es par de revolución
             if Meca.LazoPares(1)==1
                 MAR(:,1)=0;
             end
             if Meca.LazoPares(end)==1
                 MAR(:,end)=0;
             end
          %
          % Proceso de suma hacia la izquierda
             NVPP=[];
             for i=[1:1:length(Meca.LazoPares)]
                  if Meca.LazoPares(i)==2
                     NVPP=[NVPP,i];
                  end
             end
             NVPP(end)=[];
             k=0;
             for i=[length(MAR(1,:))-1:-1:1]
                 if length(NVPP)-k<=0
                     MAR(:,i)=MAR(:,i)+MAR(:,i+1);
                 else
                     if i==NVPP(end-k)
                         k=k+1;
                     else
                         MAR(:,i)=MAR(:,i)+MAR(:,i+1);
                     end
                 end
                 MAR;
             end
          %
          % Colocación MAR en Meca.Rela(i).Ax y Meca.Rela(i).Ay
             if Meca.LazoPares(1)==1
                 MAR(:,1)=0;
             end
             for i=[1:1:length(MAR(1,:))]
                 Meca.Rela(i).Ax=MAR(1,i);
                 Meca.Rela(i).Ay=MAR(2,i);
             end
          %
      end
   %
%%% Código nuevo final %%%

%% Código nuevo %%


%%%%%%%%%%%%%%%%%%


end
```

ACERCADE.M

```
function varargout = AcercaDe(varargin)
% ACERCADE MATLAB code for AcercaDe.fig
%      ACERCADE, by itself, creates a new ACERCADE or raises the existing
%      singleton*.
%
%      H = ACERCADE returns the handle to a new ACERCADE or the handle to
%      the existing singleton*.
%
%      ACERCADE('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ACERCADE.M with the given input arguments.
%
%      ACERCADE('Property','Value',...) creates a new ACERCADE or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before AcercaDe_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to AcercaDe_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help AcercaDe

% Last Modified by GUIDE v2.5 05-Jul-2016 18:28:56

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @AcercaDe_OpeningFcn, ...
                   'gui_OutputFcn',  @AcercaDe_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before AcercaDe is made visible.
function AcercaDe_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to AcercaDe (see VARARGIN)

% Choose default command line output for AcercaDe
```

```
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

Logo=imread('Logo1.png');
axes(handles.Logo);
imshow(Logo)
axis off

ETSI=imread('logoESI.jpg');
axes(handles.ETSI);
imshow(ETSI)
axis off

US=imread('logoUS.jpg');
axes(handles.US);
imshow(US)
axis off

% UIWAIT makes AcercaDe wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = AcercaDe_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

AYUDADIRECPRSD.M

```
function varargout = AyudaDirecPRSD(varargin)
% AYUDADIRECPRSD MATLAB code for AyudaDirecPRSD.fig
%      AYUDADIRECPRSD, by itself, creates a new AYUDADIRECPRSD or raises the
existing
%      singleton*.
%
%      H = AYUDADIRECPRSD returns the handle to a new AYUDADIRECPRSD or the
handle to
%      the existing singleton*.
%
%      AYUDADIRECPRSD('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in AYUDADIRECPRSD.M with the given input
arguments.
%
%      AYUDADIRECPRSD('Property','Value',...) creates a new AYUDADIRECPRSD or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before AyudaDirecPRSD_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to AyudaDirecPRSD_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help AyudaDirecPRSD

% Last Modified by GUIDE v2.5 05-Jul-2016 10:59:00

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @AyudaDirecPRSD_OpeningFcn, ...
                   'gui_OutputFcn',  @AyudaDirecPRSD_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before AyudaDirecPRSD is made visible.
function AyudaDirecPRSD_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to AyudaDirecPRSD (see VARARGIN)

% Choose default command line output for AyudaDirecPRSD
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes AyudaDirecPRSD wait for user response (see UIRESUME)
% uiwait(handles.figure1);
ConCov=imread('ConCov1.jpg');
axes(handles.ConcavoConvexo);
imshow(ConCov)
axis off

CovCon=imread('CovCon.jpg');
axes(handles.ConvexoConcavo);
imshow(CovCon)
axis off

RuedaPlano=imread('RuedaPlano.jpg');
axes(handles.RuedaPlano);
imshow(RuedaPlano)
axis off

PlanoRueda=imread('PlanoRueda.jpg');
axes(handles.PlanoRueda);
imshow(PlanoRueda)
axis off




% --- Outputs from this function are returned to the command line.
function varargout = AyudaDirecPRSD_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

AYUDARADIOS.M

```
function varargout = AyudaRadios(varargin)
% AYUDARADIOS MATLAB code for AyudaRadios.fig
%      AYUDARADIOS, by itself, creates a new AYUDARADIOS or raises the
existing
%      singleton*.
%
%      H = AYUDARADIOS returns the handle to a new AYUDARADIOS or the handle
to
%      the existing singleton*.
%
%      AYUDARADIOS('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in AYUDARADIOS.M with the given input
arguments.
%
%      AYUDARADIOS('Property','Value',...) creates a new AYUDARADIOS or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before AyudaRadios_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to AyudaRadios_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help AyudaRadios

% Last Modified by GUIDE v2.5 19-May-2016 20:46:07

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @AyudaRadios_OpeningFcn, ...
                   'gui_OutputFcn',  @AyudaRadios_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before AyudaRadios is made visible.
function AyudaRadios_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% varargin   command line arguments to AyudaRadios (see VARARGIN)

% Choose default command line output for AyudaRadios
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

Ayuda=imread('AyudaRadios1.jpg');
axes(handles.Grafica);
image(Ayuda)
axis off

% UIWAIT makes AyudaRadios wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = AyudaRadios_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes when figure1 is resized.
function figure1_ResizeFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on scroll wheel click while the figure is in focus.
function figure1_WindowScrollWheelFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  structure with the following fields (see FIGURE)
%   VerticalScrollCount: signed integer indicating direction and number of
clicks
%   VerticalScrollAmount: number of lines scrolled for each click
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

BUCLEAC.M

```
function [Meca] = BucleAC(Meca)
%% Función que modifica los datos de Meca en caso de Bucle abierto %%

   % Caso de bucle cerrado
     if Meca.Lazo(end)==1
         Meca.Bucle=0;
     end
   %

   % Caso de bucle abierto
      if Meca.Lazo(end)~=1
          Meca.Nrest=Meca.Nrest+1;
          Meca.NPP=Meca.NPP+1;
          Meca.Lazo=[Meca.Lazo,1];
          Meca.LazoPares=[Meca.LazoPares,2];
          Meca.Bucle=1;
      end
   %

end
```

CALCULADOR.M

```
function [Meca] = Calculador(Meca)
%% Función que resuelve de nuevo el problema de velocidad y aceleración para
su posterior simulación %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Intento de implementación de resolución del problema de velocidad V5 %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Cálculos %%
   % Determinación de bucle abierto o cerrado
      Meca=BucleAC(Meca);
   %
   % Creación de la subestructura Meca.Rela
      Meca=Relaciones(Meca);
   %
   % Creación de los grupos para la matriz TH - Meca.Grupo().G
      Meca=CreaGrupos(Meca);
   %
   % Creación de los tramos Meca.Tramo().T
      Meca=CreaTramos(Meca);
   %

   % Diferenciación del caso de NPP=0 y NPP~=0
      if Meca.NPP==0
          % Caso particular de NPP=0
             Meca=NOPP(Meca);
          %
      else
          % Creación de la matriz TH
             Meca=MatrizTH(Meca);
          %
          % Creación de la matriz HV
             Meca=MatrizHV(Meca);
          %
          % Unión de las matrices TH y HV. Creación de la Matriz Meca.Q
             Q=[Meca.TH,-Meca.HV];
             Meca.Q=Q;
          %
          % Aplicación de la condición de par prismático (Wi=Wj) a Meca.Q
             Meca=CondPrismatico(Meca);
          %
      end
          % Aplicación de los GDL y creación de las matrices Meca.Qind y
Meca.Qdep
             Meca=CondVGDL(Meca);
          %
          % Resolución del problema de velocidad
            Sol=Meca.Qdep\-Meca.Qind;
            Sol=Meca.VGDL+Sol;
            Meca.SolVeloc=Sol;
            if Meca.NPP~=0
               Meca=CondPrisSolV(Meca);
            end
          %
   %

   % Aplicación del Método de las velocidades relativas
      Meca=VelocRela(Meca);
   %
```

```
   % Resolución del problema de aceleraciones
      Meca=ProbAcel(Meca);
   %

   % Aplicación del Método de las aceleraciones relativas
      Meca=AceleRela(Meca);
   %



%%%%%%%%%%%%%%

%% Comprobaciones finales %%
   % Comprobador del resultado del problema de velocidad
         Meca=CompruebaVel(Meca);
         aux=sum(abs(Meca.CompruebaVel));
         ErrorV=log10(aux);
         if ErrorV==-Inf
             fprintf('\n--> Problema de velocidad correcto\n    con error
0\n')
         else
              if aux<10^-5
                 fprintf('\n--> Problema de velociad correcto\n    con un
errror del 10^%d\n',ErrorV)
              else
                 fprintf('\n--> Problema de velocidad\n    recuelto con algun
tipo de error\n')
              end
         end
   %

   % Comprobador del resultado del problema de aceleración
       Meca=CompruebaAce(Meca);
         aux=sum(abs(Meca.CompruebaAce));
         ErrorA=log10(aux);
         if ErrorA==-Inf
             fprintf('\n--> Problema de aceleración correcto\n    con error
0\n')
         else
              if aux<10^-5
                 fprintf('\n--> Problema de aceleración correcto\n    con un
errror del 10^%d\n',ErrorV)
              else
                 fprintf('\n--> Problema de aceleración\n    recuelto con
algun tipo de error\n')
              end
         end
   %


%%%%%%%%%%%%%%%%%%%%%%%%%%%

end
```

CARGA_ARCHIVOS.M

```
function [Meca] = Carga_Archivos(Directorio)
%% Función que carga un archivo guardado %%

   Directorio=char(Directorio);
   cd(Directorio);

   Nbody=fopen('Nbody.txt','r');
     Nbody=fscanf(Nbody,'%c',inf);
     Meca.Nbody=Nbody;

   Nrest=fopen('Nrest.txt','r');
     Nrest=fscanf(Nrest,'%c',inf);
     Meca.Nrest=Nrest;

   NPR=fopen('NPR.txt','r');
     NPR=fscanf(NPR,'%c',inf);
     Meca.NPR=NPR;

   NPP=fopen('NPP.txt','r');
     NPP=fscanf(NPP,'%c',inf);
     Meca.NPP=NPP;

   NPRSD=fopen('NPRSD.txt','r');
     NPRSD=fscanf(NPRSD,'%c',inf);
     Meca.NPRSD=NPRSD;

   Lazo=fopen('Lazo.txt','r');
      Lazo=fscanf(Lazo,'%c',inf);
      Meca.Lazo=Lazo;

   LazoPares=fopen('LazoPares.txt','r');
      LazoPares=fscanf(LazoPares,'%c',inf);
      Meca.LazoPares=LazoPares;

   VGDL=fopen('VGDL.txt','r');
      VGDL=fscanf(VGDL,'%c',inf);
      Meca.VGDL=VGDL;

   AGDL=fopen('AGDL.txt','r');
      AGDL=fscanf(AGDL,'%c',inf);
      Meca.AGDL=AGDL;

   Coord=fopen('Coord.txt','rt');
   M=[];
   while (~feof(Coord))
      x=fscanf(Coord,'%f %f',[1 2]);
      M=[M;x];
   end
   Coord=M;
      Meca.Coord=double(Coord);

   DirecPRSD=fopen('DirecPRSD.txt','r');
   M=[];
   while (~feof(DirecPRSD))
      x=fscanf(DirecPRSD,'%f %f %f',[1 3]);
      M=[M;x];
```

```
    end
    DirecPRSD=M;
       Meca.DirecPRSD=double(DirecPRSD);

    DirecPP=fopen('DirecPP.txt','r');
    M=[];
    while (~feof(DirecPP))
       x=fscanf(DirecPP,'%f %f',[1 2]);
       M=[M;x];
    end
    DirecPP=M;
       Meca.DirecPP=double(DirecPP);

    Radios=fopen('Radios.txt','r');
    M=[];
    while (~feof(Radios))
       x=fscanf(Radios,'%f %f',[1 2]);
       M=[M;x];
    end
    Radios=M;
       Meca.Radios=double(Radios);

end
```

COMPINI.M

```
function [Meca] = CompIni(Nbody,Nrest,NPR,NPP,NPRSD,Lazo,LazoPares,VGDL,AGDL)
%% Función realiza las comprobaciones iniciales %%

%% Comprobaciones iniciales %%
   % Comprobación 1
      if length(LazoPares)+1~=length(Lazo) || length(Lazo)~=Nrest+1
          fprintf('\n --> >>ERROR<<  Comprobación 1\n')
          Comp1=1;
      else
          Comp1=0;
      end
   %
   % Comprobación 2
      if Nrest~=NPR+NPP+NPRSD
         fprintf('\n --> >>ERROR<<  Comprobación 2\n')
         Comp2=1;
      else
          Comp2=0;
      end
   %
   % Comprobación 3
   npr=0;npp=0;nprsd=0;
      for i=[1:1:length(LazoPares)]
         if LazoPares(i)==1
             npr=npr+1;
         end
         if LazoPares(i)==2
             npp=npp+1;
         end
         if LazoPares(i)==3
             nprsd=nprsd+1;
         end
      end
     if npr~=NPR || npp~=NPP || nprsd~=NPRSD
         fprintf('\n --> >>ERROR<<  Comprobación 3\n')
         Comp3=1;
     else
         Comp3=0;
     end
   %
   % Comprobación 4
      Cont=0;
      for i=[1:1:length(LazoPares)]
          if LazoPares(i)==2
              if i==1
              else
                  Cont=Cont+1;
              end
          end
      end
      if NPP==1
          if length(VGDL)~=Nbody-1;
              fprintf('\n --> <<ERROR>>  Comprobación 4\n')
              Comp4=1;
          else
              if length(VGDL)~=length(AGDL)
                 fprintf('\n --> <<ERROR>>  Comprobación 4\n')
                 Comp4=1;
```

```
                else
                    Comp4=0;
                end
            end
        else
            if length(VGDL)~=Nbody-1+Cont*2
               fprintf('\n --> <<ERROR>>  Comprobación 4\n')
               Comp4=1;
            else
                if length(VGDL)~=length(AGDL)
                    fprintf('\n --> <<ERROR>>  Comprobación 4\n')
                    Comp4=1;
                else
                    Comp4=0;
                end
            end
        end
    %

        Meca.CompIni.Comp1=Comp1;
        Meca.CompIni.Comp2=Comp2;
        Meca.CompIni.Comp3=Comp3;
        Meca.CompIni.Comp4=Comp4;

end
```

COMPRUEBAACE.M

```
function [Meca] = CompruebaAce(Meca)
%% Función Comprobador para el problema de aceleración

% Cálculo del término de la izquierda

if Meca.NPP==0
  P=0;
    for i=[1:1:length(Meca.Grupo(1).G)]
        u=Meca.Grupo(1).G(i);
           if u==Meca.Nbody
               w=0;
               A=0;
           else
               w=Meca.Cuerpo(u+1).W;
               A=Meca.Cuerpo(u+1).A;
           end
         T=Meca.Tramo(u).T;
         Tp=[-T(2);T(1)];
         P=P+(Tp.*A)-(T.*((w).^2));
    end
   Izq=P;
else
      Izq=[];
      k=1;
      for i=[1:1:length(Meca.Grupo)]
          AT=0;
          WT=0;
          AWV=0;
          for j=[1:1:length(Meca.Grupo(i).G)]
              u=Meca.Grupo(i).G(j);
               if u==Meca.Nbody
                  w=0;
                  A=0;
               else
                   w=Meca.Cuerpo(u+1).W;
                  A=Meca.Cuerpo(u+1).A;
               end
              T=Meca.Tramo(u).T;
              T=[-T(2);T(1)];
              hp=Meca.DirecPP(i,:)';
              hp=[-hp(2);hp(1)];

              AT=AT+A*sum(T.*hp);
              WT=WT+(w^2)*sum([T(2);-T(1)].*hp);
              if Meca.Rela(u).Codig==3
                     Ri=Meca.Radios(k,1);
                     Rj=Meca.Radios(k,2);
                     landa=Meca.DirecPRSD(k,:);
                     landa=(landa)./norm(landa)';
                     wi=Meca.Cuerpo(Meca.Rela(u).Cuerpos(1)).W;
                     wj=Meca.Cuerpo(Meca.Rela(u).Cuerpos(2)).W;
                       if Meca.Radios(k,3)==1
                          aij=((wi-wj).^2).*((Ri.*Rj)./(Ri-Rj)).*landa';
                       end
                       if Meca.Radios(k,3)==2
                           aij=((wi-wj).^2).*((Ri.*Rj)./(Ri+Rj)).*landa';
                       end
                       if Meca.Radios(k,3)==3
```

```
                            aij=((wi-wj).^2).*Ri.*landa';
                        end
                        if Meca.Radios(k,3)==4
                            aij=((wi-wj).^2).*Rj.*landa';
                        end
                        V=[Meca.Rela(u).Vx;Meca.Rela(u).Vy];
                        V=[-V(2);V(1)];
                        aux=aij+2*wj.*V;
                        AWV=AWV+sum(aux.*hp);
                        k=k+1;
                end
            end
            Izq=[Izq;AT-WT-AWV];
        end
end
%

% Cálculo del término de la derecha
if Meca.NPP>=2
    Der=[];
    RCP=Meca.RCP;
    if Meca.Bucle==1
        RCP(end,:)=[];
    end

    Linea=[RCP(:,1)' 1];
    RCP;
    for i=[1:1:length(RCP(:,1))]
        hp=Meca.DirecPP(i,:)';
        hp=[-hp(2);hp(1)];
        fin=Linea(i+1);
        ini=Linea(i);
        Afin=[Meca.Rela(fin).Ax;Meca.Rela(fin).Ay];
        Aini=[Meca.Rela(ini).Ax;Meca.Rela(ini).Ay];
        P=sum((Afin-Aini).*hp);
        Der=[Der;P];
    end
else
    Der=0;
end

if Meca.NPP==0
     P=[0;0];
     cont=1;
      for i=[1:1:length(Meca.Grupo(1).G)]
          u=Meca.Grupo(1).G(i);
            if Meca.Rela(u).Codig==3
                wi=Meca.Cuerpo(u).W;
                  if u==Meca.Nbody
                      wj=0;
                  else
                      wj=Meca.Cuerpo(u+1).W;
                  end
                 Ri=Meca.Radios(cont,1);
                 Rj=Meca.Radios(cont,2);
                 landa=Meca.DirecPRSD(cont,:);
                 landa=(landa)./norm(landa);
                    if Meca.Radios(cont,3)==1
                         aij=((wi-wj).^2).*((Ri.*Rj)./(Ri-Rj)).*landa;
                    end
                    if Meca.Radios(cont,3)==2
```

```
                    aij=((wi-wj).^2).*((Ri.*Rj)./(Ri+Rj)).*landa;
                end
                if Meca.Radios(cont,3)==3
                    aij=((wi-wj).^2).*(Ri).*landa;
                end
                if Meca.Radios(cont,3)==4
                    aij=((wi-wj).^2).*(Rj).*landa;
                end
            P=P+aij';
            vi=[Meca.Rela(u).Vx;Meca.Rela(u).Vy];
            wv=2.*[-vi(2);vi(1)].*wj;
            P=P+(wv);
            cont=cont+1;
        else
            P=[0;0];
        end
    end
    Der=P;
end

%
    if Meca.Bucle==1
        Der=[Der;0];
    end

Meca.CompruebaAce=Izq-Der;
end
```

COMPRUEBAVEL.M

```
function [Meca] = CompruebaVel(Meca)
%% Función Comprobador para el problema de velocidad

% Cálculo del término de la izquierda
   Izq=[];
  if Meca.NPP==0
    P=0;
      for i=[1:1:length(Meca.Grupo(1).G)]
          u=Meca.Grupo(1).G(i);
             if u==Meca.Nbody
                 w=0;
             else
                 w=Meca.Cuerpo(u+1).W;
             end
          T=Meca.Tramo(u).T;
          T=[-T(2);T(1)];
          P=P+T.*w;
      end
      Izq=P;
  else
      for i=[1:1:length(Meca.Grupo)]
          P=0;
          for j=[1:1:length(Meca.Grupo(i).G)]
              u=Meca.Grupo(i).G(j);
              if u==Meca.Nbody
                  w=0;
              else
                  w=Meca.Cuerpo(u+1).W;
              end
              T=Meca.Tramo(u).T;
              T=[-T(2);T(1)];
              hp=Meca.DirecPP(i,:)';
              hp=[-hp(2);hp(1)];
              P=P+w*sum(T.*hp);
          end
          Izq=[Izq;P];
      end
  end
%

% Cálculo del término de la derecha
   Der=[];
   if Meca.NPP>=2
      RCP=Meca.RCP;
      if Meca.Bucle==1
          RCP(end,:)=[];
      end

      Linea=[RCP(:,1)' 1];

      for i=[1:1:length(RCP(:,1))]
          hp=Meca.DirecPP(i,:)';
          hp=[-hp(2);hp(1)];
          fin=Linea(i+1);
          ini=Linea(i);
          Vfin=[Meca.Rela(fin).Vx;Meca.Rela(fin).Vy];
```

```
            Vini=[Meca.Rela(ini).Vx;Meca.Rela(ini).Vy];
            P=sum((Vfin-Vini).*hp);
            Der=[Der;P];
        end
    else
        Der=0;
    end

 if Meca.NPP==0
     Der=[0;0];
 end

%
    if Meca.Bucle==1
        Der=[Der;0];
    end
Meca.CompruebaVel=Izq-Der;
end
```

CONDAGDL.M

```
function [Meca] = CondAGDL(Meca)
%% Función que aplica los GDL y crea las matrices Meca.Qdep y Meca.Qdep %%
   % Aceleraciones %
   Qdep=Meca.M;
   for i=[1:1:length(Meca.AGDL)]
       if Meca.AGDL(i)==0
       else
           Qdep(:,i)=0;
       end
   end
   Meca.Qdep=Qdep;

   Qind=Meca.M*Meca.AGDL;
   Meca.Qind=Qind;

end
```

CONDICIONES_DE_USO.M

```
function varargout = Condiciones_de_uso(varargin)
% CONDICIONES_DE_USO MATLAB code for Condiciones_de_uso.fig
%      CONDICIONES_DE_USO, by itself, creates a new CONDICIONES_DE_USO or
raises the existing
%      singleton*.
%
%      H = CONDICIONES_DE_USO returns the handle to a new CONDICIONES_DE_USO
or the handle to
%      the existing singleton*.
%
%      CONDICIONES_DE_USO('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in CONDICIONES_DE_USO.M with the given input
arguments.
%
%      CONDICIONES_DE_USO('Property','Value',...) creates a new
CONDICIONES_DE_USO or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Condiciones_de_uso_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Condiciones_de_uso_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Condiciones_de_uso

% Last Modified by GUIDE v2.5 20-Jul-2016 13:53:35

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Condiciones_de_uso_OpeningFcn, ...
                   'gui_OutputFcn',  @Condiciones_de_uso_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Condiciones_de_uso is made visible.
function Condiciones_de_uso_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Condiciones_de_uso (see VARARGIN)

% Choose default command line output for Condiciones_de_uso
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

Logo=imread('Logo1.png');
axes(handles.Logo);
imshow(Logo)
axis off

% UIWAIT makes Condiciones_de_uso wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Condiciones_de_uso_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on button press in Continuar.
function Continuar_Callback(hObject, eventdata, handles)
% hObject    handle to Continuar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

Prueba11()
close(Condiciones_de_uso)




% --- Executes on button press in NVM.
function checkbox2_Callback(hObject, eventdata, handles)
% hObject    handle to NVM (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of NVM
```

CONDPRISMATICO.M

```
function [Meca] = CondPrismatico(Meca)
%% Función que aplica la condición de par prismático (Wi=Wj) en Meca.Q %%

   RCP=[];
   for i=[1:1:length(Meca.Rela)]
       if Meca.Rela(i).Codig==2
           C1=Meca.Rela(i).Cuerpos(1);
           C2=Meca.Rela(i).Cuerpos(2);
           RCP=[RCP;C1,C2];
       end
   end
   Meca.RCP=RCP;
   % Paso la Matriz RCP a versión número de las columnas
      RCP=RCP-1;
   %
   % Procedimiento de suma de las columnas
      M=Meca.Q;

      % Restricción en caso de bucle abierto [Pruebas]
%         if Meca.Bucle==1
%             RCP(end,:)=[];
%         end
      %

      for Repeticiones=[1:1:Meca.NPP*5]
       % Casos en los que Wi=W1=0
         for i=[1:1:length(RCP(:,1))]
             a=RCP(i,1);
             b=RCP(i,2);
             if a==0
                 M(:,b)=0;
             end
             if b==0
                 M(:,a)=0;
             end
         end
       %
       for i=[1:1:length(RCP(:,1))]
           a=RCP(i,1);
           b=RCP(i,2);
           if a~=0 && b~=0
               M(:,a)=M(:,a)+M(:,b);
               M(:,b)=0;
           end
       end
   %
      end
   % Eliminación de columnas que sean Wi=Wj=Wk=W1=0
   if Meca.NPP>1
     LineaRCP=[];
      for i=[1:1:length(RCP(:,1))]
          LineaRCP=[LineaRCP,RCP(i,:)];
      end
      fin=0;
      i=1;
      LZ=[];
      RCP;
      while fin==0
```

```
        if RCP(i,2)==RCP(i+1,1)
            LZ=[LZ,RCP(i+1,1)];
            i=i+1;
               if i==length(RCP(:,1))
                   fin=1;
               end
        else
            fin=1;
        end

      end
      i=0;
      fin=0;
      while fin==0
        if RCP(end-i,1)==RCP(end-i-1,2)
            LZ=[LZ,RCP(end-i-1,:)];
            i=i+1;
               if i-1==0
                   fin=1;
               end
        else
            fin=1;
        end
      end
      LZ;
      for i=[1:1:length(LZ)]
          M(:,LZ(i))=0;
      end
   end
   %

   Meca.M=M;

end
```

CONDPRISSOLA.M

```
function [Meca] = CondPrisSolA(Meca)
%% Función que implementa las condiciones prismáticas Wi=Wj en la solución%%

   RCP=Meca.RCP-1;

   % Restrinción en caso de bucle abierto [Pruebas]
      if Meca.Bucle==1
          RCP(end,:)=[];
      end
   %

   Sol=Meca.SolAcel;
   for i=[1:1:length(RCP(:,1))]
        a=RCP(i,1);
        b=RCP(i,2);
          if a==0
              Sol(b)=0;
          end
          if b==0
             Sol(a)=0;
          end
   end
   RCP;
   for i=[1:1:length(RCP(:,1))]
       a=RCP(i,1);
       b=RCP(i,2);
          if a~=0 && b~=0
              S1=Sol(a);
              S2=Sol(b);
              if S1==0 && S2==0
              end
              if S1==0 && S2~=0
                  Sol(a)=S2;
              end
              if S1~=0 && S2==0
                 Sol(b)=S1;
              end
          end
   end

   Meca.SolAcel=Sol;

end
```

CONDPRISSOLV.M

```
function [Meca] = CondPrisSolV(Meca)
%% Función que implementa las condiciones prismáticas Wi=Wj en la solución%%

   RCP=Meca.RCP-1;

   % Restrinción en caso de bucle abierto [Pruebas]
      if Meca.Bucle==1
          RCP(end,:)=[];
      end
   %

   Sol=Meca.SolVeloc;
   for i=[1:1:length(RCP(:,1))]
        a=RCP(i,1);
        b=RCP(i,2);
          if a==0
              Sol(b)=0;
          end
          if b==0
             Sol(a)=0;
          end
   end
   RCP;
   for i=[1:1:length(RCP(:,1))]
       a=RCP(i,1);
       b=RCP(i,2);
          if a~=0 && b~=0
              S1=Sol(a);
              S2=Sol(b);
              if S1==0 && S2==0
              end
              if S1==0 && S2~=0
                  Sol(a)=S2;
              end
              if S1~=0 && S2==0
                 Sol(b)=S1;
              end
          end
   end

   Meca.SolVeloc=Sol;

end
```

CONDVGDL.M

```
function [Meca] = CondVGDL(Meca)
%% Función que aplica los GDL y crea las matrices Meca.Qdep y Meca.Qdep %%
   % Velocidades %
   Qdep=Meca.M;
   for i=[1:1:length(Meca.VGDL)]
       if Meca.VGDL(i)==0
       else
           Qdep(:,i)=0;
       end
   end
   Meca.Qdep=Qdep;

   Qind=Meca.M*Meca.VGDL;
   Meca.Qind=Qind;

end
```

CREACHAR.M

```
function [Meca] = CreaChar(Meca)
%% Función que crea dos caenas char con las velocidades iniciales a
introducir %%
% Meca.GDL.CharV para las velocidades
% Meca.GDL.CharA para las aceleraciones

V='[w2';
A='[a2';
 for i=[3:1:Meca.Nbody]
     f=num2str(i);
     V=[V ' w'];
     V=[V f];

     A=[A ' a'];
     A=[A f];
 end
  Meca.GDL.CharV.ang=V;
  Meca.GDL.CharA.ang=A;

  V=[];
  A=[];
 if Meca.NPP>=2
     V=[];
     A=[];
     Linea=Meca.RCP(:,1);
     for j=[1:1:length(Linea)]
         if Linea(j)==1
         else
             f=Linea(j)+1;
             f=num2str(f);
             V=[V ' Vx'];
             V=[V f];
             V=[V ' Vy'];
             V=[V f];

             A=[A ' Ax'];
             A=[A f];
             A=[A ' Ay'];
             A=[A f];
         end

     end
     V=[V ']'];
     A=[A ']'];

 else
    V=[V ']'];
    A=[A ']'];
 end

 Meca.GDL.CharV.lin=V;
 Meca.GDL.CharA.lin=A;
end
```

CREAGRUPOS.M

```
function [Meca] = CreaGrupos(Meca)
%% Función que genera los grupos para la matriz TH %%

   % Caso en el que no existan pares prismáticos o exista solo uno
     if Meca.NPP<=1
         Meca.Grupo(1).G=[1:1:Meca.Nrest];
     end
   %

%    % Caso en el que exisitan más de un par prismatico
%       if Meca.NPP>=2
%           G=[];
%           k=1;
%           for i=[1:1:length(Meca.LazoPares)]
%               G=[G,i];
%               if Meca.LazoPares(i)==2
%                   Meca.Grupo(k).G=G;
%                   k=k+1;
%                   G=[];
%               end
%           end
%       end
%    %

   % Caso en el que el mecanismo empiece o acabe con un par prismático
%        LazoModif=Meca.LazoPares;
%        if Meca.LazoPares(1)==2
%            LazoModif(1)=1;
%        end
%        LazoModif=[LazoModif,LazoModif(1)];
%        G=[];
%        k=1;
%           for i=[1:1:length(LazoModif)-1]
%              G=[G,i];
%               if LazoModif(i+1)==2
%                   Meca.Grupo(k).G=G;
%                   k=k+1;
%                   G=[];
%               end
%           end
   %

   if Meca.NPP>=2
       G=[];
       k=1;
       cont=0;

         for i=[1:1:length(Meca.LazoPares)-1]
            G=[G,i];
              if Meca.LazoPares(i)==2
                cont=cont+1;
              end
              if Meca.LazoPares(i+1)==2 && cont==1
                 Meca.Grupo(k).G=G;
                cont=0;
                k=k+1;
                G=[];
```

```
                end
                if cont==2
                    Meca.Grupo(k).G=G;
                    cont=0;
                    k=k+1;
                    G=[];
                end
                if i+1==length(Meca.LazoPares)
                    G=[G,i+1];
                    Meca.Grupo(k).G=G;
                    cont=0;
                    k=k+1;
                    G=[];
                end
            end

%   if Meca.Lazo(end)~=1
%       Meca.Grupo(end).G=[Meca.Grupo(end).G Meca.Lazo(end)];
%   end


    end

end
```

CREAINFORME.M

```
%% Informe del mecanismo
function [] = CreaInforme()

figure
Logo=imread('LogoInforme.jpg');
imshow(Logo)

 Fecha=date;


 Meca=evalin('base','Meca');
     if isempty(Meca.Nombre)
       Meca.Nombre='Mecanismo';
     end

 ListaPP=[];
 ListaPRSD=[];
  for i=[1:1:length(Meca.LazoPares)]
      if Meca.LazoPares(i)==2
          ListaPP=[ListaPP;i];
      end
      if Meca.LazoPares(i)==3
          ListaPRSD=[ListaPRSD;i];
      end
  end


%% Nombre del mecanismo
   fprintf('%s                          %s \n\n',Meca.Nombre,Fecha);
%% Datos del mecanismo
% Datos básicos
   fprintf('Número de eslabones: %d  \t;   Número de pares de restricción:
%d\n',Meca.Nbody,Meca.Nrest);
%%
% Datos de los pares
   fprintf('Pares de revolución: %d   \t;   Pares prismáticos:
%d',Meca.NPR,Meca.NPP);
%%
% Lazo del mecanismo
 fprintf('                {');
   for i=[1:1:length(Meca.Lazo)]
       fprintf(' %d',Meca.Lazo(i));
   end
 fprintf(' }');
%%
% Lazo de los pares del mecanismo
 fprintf('                {');
   for i=[1:1:length(Meca.LazoPares)]
       fprintf(' %d',Meca.LazoPares(i));
   end
 fprintf(' }');
fprintf('\n\n------------ Codificación de los pares ------------');

  fprintf('\n\n   Par de revolución: 1  \t;  Par prismático: 2');
  fprintf('\n      Par de rodadura sin deslizamiento: 3\n');

fprintf('\n--------------------------------------------------');
```

```
%% Geometría del mecanismo
% Coordenadas de los pares [X,Y]
 fprintf('\n')
   for i=[1:1:length(Meca.Coord(:,1))]
      fprintf('Par %d : [ %d ; %d ]\n',i,Meca.Coord(i,1),Meca.Coord(i,2));
   end
 fprintf('\n');
%%
% Direcciones de los pares prismáticos
if Meca.NPP==0
    fprintf('\n <<El mecanismo no posee pares prismáticos>>');
else
    fprintf('\n')
      for i=[1:1:length(Meca.DirecPP(:,1))]
         fprintf('Par prismático %d : \n',i);
         u=ListaPP(i);
         C1=Meca.Rela(u).Cuerpos(1);
         C2=Meca.Rela(u).Cuerpos(2);
         fprintf('   Eslabón %d  y  Eslabón %d\n',C1,C2);
         fprintf('Dirección del par: [ %d ; %d
]',Meca.DirecPP(i,1),Meca.DirecPP(i,2));
         if i==length(Meca.DirecPP(:,1))
         else
            fprintf('\n-----------------\n');
         end
      end
end
    fprintf('\n');
%%
% Direciones de los pares de rodadura sin deslizamiento
if Meca.NPRSD==0
    fprintf('\n <<El mecanismo no posee pares de rodadura sin
deslizamiento>>');
else
    fprintf('\n')
      for i=[1:1:length(Meca.DirecPRSD(:,1))]
         fprintf('Par de rodadura sin deslizamiento %d :\n',i);
         u=ListaPRSD(i);
         C1=Meca.Rela(u).Cuerpos(1);
         C2=Meca.Rela(u).Cuerpos(2);
         fprintf('   Eslabón %d  y  Eslabón %d\n',C1,C2);
         fprintf('Dirección del par: [ %d ; %d
]',Meca.DirecPRSD(i,1),Meca.DirecPRSD(i,2));
         if i==length(Meca.DirecPRSD)
         else
            fprintf('\n-----------------\n');
         end
      end
end
    fprintf('\n');
%%
% Radios de curvatura de los pares de rodadura sin deslizamiento
if Meca.NPRSD==0
    fprintf('\n <<El mecanismo no posee pares de rodadura sin
deslizamiento>>');
else
    fprintf('\n')
      for i=[1:1:length(Meca.Radios(:,1))]
         fprintf('Par de rodadura sin deslizamiento %d :\n',i);
         u=ListaPRSD(i);
```

```
         C1=Meca.Rela(u).Cuerpos(1);
         C2=Meca.Rela(u).Cuerpos(2);
         fprintf('    Radio de curvatura eslabón %d : %d     ;     Radio
curvatura eslabón %d : %d',C1,Meca.Radios(i,1),C2,Meca.Radios(i,2))
         if i==length(Meca.DirecPRSD)
         else
            fprintf('\n-----------------\n');
         end
      end
end
    fprintf('\n');

%%
% Tipología de los pares de rodadura sin deslizamiento
if Meca.NPRSD==0
    fprintf('\n <<El mecanismo no posee pares de rodadura sin
deslizamiento>>');
else
    fprintf('\n')
      for i=[1:1:length(Meca.Radios(:,1))]
         fprintf('Par de rodadura sin deslizamiento %d :\n',i);
         u=ListaPRSD(i);
         C1=Meca.Rela(u).Cuerpos(1);
         C2=Meca.Rela(u).Cuerpos(2);
         fprintf('   Eslabón %d  y  Eslabón %d\n',C1,C2);
         fprintf('   Tipología %d : ',Meca.Radios(i,3));
            if Meca.Radios(i,3)==1
                fprintf('Cóncavo - Convexo');
            end
            if Meca.Radios(i,3)==2
                fprintf('Convexo - Cóncavo');
            end
            if Meca.Radios(i,3)==3
                fprintf('Rueda - Plano');
            end
            if Meca.Radios(i,3)==4
                fprintf('Plano - Rueda');
            end
         if i==length(Meca.DirecPRSD)
         else
            fprintf('\n-----------------\n');
         end
      end
end
    fprintf('\n');

%% Velocidades y acelecaciones iniciales
% Velocidades iniciales
FV=[Meca.GDL.CharV.ang Meca.GDL.CharV.lin];
fprintf('\n %s : [',FV);
   for i=[1:1:length(Meca.VGDL)]
       fprintf(' %d',Meca.VGDL(i));
   end
 fprintf(' ] um/s \n');

%%
% Acelecaciones iniciales
FA=[Meca.GDL.CharA.ang Meca.GDL.CharA.lin];
fprintf('\n %s : [',FA);
   for i=[1:1:length(Meca.AGDL)]
       fprintf(' %d',Meca.AGDL(i));
   end
 fprintf(' ] um/s^2 \n');
```

```
%% Resultados
% Velocidades y aceleraciones lineales
if Meca.Prob==2
fprintf('\n');
   for i=[1:1:length(Meca.Rela)]
       fprintf('Relación %d : \n',i);
       fprintf('    Eslabón %d  y Eslabón
%d\n',Meca.Rela(i).Cuerpos(1),Meca.Rela(i).Cuerpos(2));
       fprintf('    Coordenadas : [ %d ; %d
]\n',Meca.Rela(i).x,Meca.Rela(i).y);
       fprintf('    Velocidad lineal : [ %d ; %d ]
um/s\n',Meca.Rela(i).Vx,Meca.Rela(i).Vy);
       fprintf('    Acelecarión lineal : [ %d ; %d ]
um/s^2\n',Meca.Rela(i).Ax,Meca.Rela(i).Ay);
          if i==length(Meca.Rela)
          else
              fprintf('\n-----------------\n');
          end
   end
fprintf('\n');
end
if Meca.Prob==1
fprintf('\n');
   for i=[1:1:length(Meca.Rela)]
       fprintf('Relación %d : \n',i);
       fprintf('    Eslabón %d  y Eslabón
%d\n',Meca.Rela(i).Cuerpos(1),Meca.Rela(i).Cuerpos(2));
       fprintf('    Coordenadas : [ %d ; %d
]\n',Meca.Rela(i).x,Meca.Rela(i).y);
       fprintf('    Velocidad lineal : [ %d ; %d ]
um/s\n',Meca.Rela(i).Vx,Meca.Rela(i).Vy);
          if i==length(Meca.Rela)
          else
              fprintf('\n-----------------\n');
          end
   end
fprintf('\n');
end

%%
% Velocidades y aceleraciones angulares
if Meca.Prob==2
fprintf('\n');
   for i=[1:1:length(Meca.Cuerpo)]
       fprintf('Eslabón %d : \n',i);
       fprintf('    Velocidad angular : %d rad/s\n',Meca.Cuerpo(i).W);
       fprintf('    Acelecarión angular : %d rad/s^2\n',Meca.Cuerpo(i).A);
          if i==length(Meca.Rela)
          else
              fprintf('\n-----------------\n');
          end
   end
fprintf('\n');
end
if Meca.Prob==1
fprintf('\n');
   for i=[1:1:length(Meca.Cuerpo)]
       fprintf('Eslabón %d : \n',i);
       fprintf('    Velocidad angular : %d rad/s\n',Meca.Cuerpo(i).W);
          if i==length(Meca.Rela)
```

```
          else
              fprintf('\n-----------------\n');
          end
   end
fprintf('\n');
end


%% Comprobaciones de los resultados
fprintf('\nCAC 2D comprueba los resultados obtenidos mediante \nel método de
cálculo con un error inferior a %d. \nSi se detectase algún problema en los
valores obtenidos \nse comunicará un posible fallo.',10^-5);
%%
% Comprobación del problema de velocidad
   aux=sum(abs(Meca.CompruebaVel));
   ErrorV=log10(aux);
      if ErrorV==-Inf
          fprintf('\n Problema de velocidad correcto\n    con error 0\n')
      else
           if aux<10^-5
              fprintf('\n Problema de velociad correcto\n    con un errror
del 10^%d\n',ErrorV)
           else
              fprintf('\n Problema de velocidad\n    recuelto con algun tipo
de error\n')
           end
      end

%%
if Meca.Prob==2
% Comprobación del problema de aceleración
   aux=sum(abs(Meca.CompruebaAce));
   ErrorA=log10(aux);
      if ErrorA==-Inf
          fprintf('\n Problema de aceleración correcto\n    con error 0\n')
      else
           if aux<10^-5
              fprintf('\n Problema de aceleración correcto\n    con un errror
del 10^%d\n',ErrorA)
           else
              fprintf('\n Problema de aceleración\n    recuelto con algun
tipo de error\n')
           end
      end
end
%% Esquema gráfico del mecanismo
figure
    axis auto
    title('Mecanismo en la posición dada')
    % Muestreo de los pares del mecanismo
       MP=MuestraPares(Meca);
       if MP==1
       else
           fprintf('\n --> >>ERROR<< Muestreo de los pares\n')
       end
    %
    % Muestreo de los cuerpos del mecanismo
       MC=MuestraCuerpos(Meca);
       if MC==1
       else
           fprintf('\n --> >>ERROR<< Muestreo de los cuerpos\n')
       end
    %
```

```
%% Distribución de velocidades y aceleraciones
    axis auto
    title('Mecanismo en la posición dada')
    % Muestreo de los pares del mecanismo
       MP=MuestraPares(Meca);
       if MP==1
       else
           fprintf('\n --> >>ERROR<< Muestreo de los pares\n')
       end
    %
    % Muestreo de los cuerpos del mecanismo
       MC=MuestraCuerpos(Meca);
       if MC==1
       else
           fprintf('\n --> >>ERROR<< Muestreo de los cuerpos\n')
       end
    %
    % Muestreo de las velocidades de cada Par
       MV=MuestraVelocidades(Meca);
       if MV==1
       else
           fprintf('\n --> >>ERROR<<  Muestreo de las velocidades\n')
       end
    %
    % Muestreo de las velocidades de cada Par
    if Meca.Prob==2
       MA=MuestraAceleraciones(Meca);
       if MA==1
       else
           fprintf('\n --> >>ERROR<<  Muestreo de las Aceleraciones\n')
       end
    end
    %

%%
fprintf('\n\n\n\n\n');

close
close
close
end
```

CREATRAMOS.M

```
function [Meca] = CreaTramos(Meca)
%% Función que genera los tramos a partir de las coordenadas - Meca.Tramo().T
%%

   for i=[1:1:Meca.Nrest-1]
       fin=[Meca.Rela(i+1).x;Meca.Rela(i+1).y];
       ini=[Meca.Rela(i).x;Meca.Rela(i).y];
       V=fin-ini;
       Meca.Tramo(i).T=V;
   end

   % Último tramo
          fin=[Meca.Rela(1).x;Meca.Rela(1).y];
          ini=[Meca.Rela(end).x;Meca.Rela(end).y];
          V=fin-ini;
          Meca.Tramo(Meca.Nrest).T=V;
   %
   % Si es un bucle abierto
%        if Meca.Lazo(end)~=1
%            u=Meca.Lazo(end);
%            Meca.Tramo(u).T=[0;0];
%        end
   %
end
```

ENTRADA.M

```
function varargout = Entrada(varargin)
% ENTRADA MATLAB code for Entrada.fig
%      ENTRADA, by itself, creates a new ENTRADA or raises the existing
%      singleton*.
%
%      H = ENTRADA returns the handle to a new ENTRADA or the handle to
%      the existing singleton*.
%
%      ENTRADA('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ENTRADA.M with the given input arguments.
%
%      ENTRADA('Property','Value',...) creates a new ENTRADA or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Entrada_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Entrada_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Entrada

% Last Modified by GUIDE v2.5 19-Jun-2016 19:29:15

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Entrada_OpeningFcn, ...
                   'gui_OutputFcn',  @Entrada_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Entrada is made visible.
function Entrada_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Entrada (see VARARGIN)

% Choose default command line output for Entrada
```

```
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

Logo=imread('Logo1.png');
axes(handles.Logo);
imshow(Logo)
axis off

LogoETSI=imread('LogoESI.jpg');
axes(handles.LogoETSI);
imshow(LogoETSI)
axis off

LogoUS=imread('LogoUS.jpg');
axes(handles.LogoUS);
imshow(LogoUS)
axis off

% UIWAIT makes Entrada wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Entrada_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in Entrar.
function Entrar_Callback(hObject, eventdata, handles)
% hObject    handle to Entrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over Salir.
Frase='Iniciando ...';
set(handles.Entrar,'String',Frase);
Condiciones_de_uso()
% pause(5)
close(Entrada)




% --- Executes on button press in Salir.
function Salir_Callback(hObject, eventdata, handles)
% hObject    handle to Salir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close
```

GUARDA_ARCHIVOSM

```
function [s] = Guarda_Archivos(Save)
%% Función que guarda los datos en la carpeta Archivos guardados %%

s=0;

W=what;
DirectorioA=W.path;
DirectorioN=[DirectorioA '\Archivos guardados'];
DirectorioN=[DirectorioN '\'];

cd(DirectorioN);
if isempty(Save.Nombre)
   mkdir([DirectorioN,'Mecanismo']);
   DirectorioN=[DirectorioN '\'];
   DirectorioN=[DirectorioN 'Mecanismo'];
else
   mkdir([DirectorioN,Save.Nombre]);
   DirectorioN=[DirectorioN '\'];
   DirectorioN=[DirectorioN Save.Nombre];
end

   cd(DirectorioN)

     ANbody=fopen('Nbody.txt','w');
     fprintf(ANbody,'%d',Save.Nbody);

     ANrest=fopen('Nrest.txt','w');
     fprintf(ANrest,'%d',Save.Nrest);

     ANPR=fopen('NPR.txt','w');
     fprintf(ANPR,'%d',Save.NPR);

     ANPP=fopen('NPP.txt','w');
     fprintf(ANPP,'%d',Save.NPP);

     ANPRSD=fopen('NPRSD.txt','w');
     fprintf(ANPRSD,'%d',Save.NPRSD);

     AVGDL=fopen('VGDL.txt','w');
       fprintf(AVGDL,'[');
       for i=[1:1:length(Save.VGDL)]
           if i==length(Save.VGDL)
              fprintf(AVGDL,'%d',Save.VGDL(i));
              fprintf(AVGDL,']');
           else
              fprintf(AVGDL,'%d',Save.VGDL(i));
              fprintf(AVGDL,' ');
           end
       end

     AAGDL=fopen('AGDL.txt','w');
       fprintf(AAGDL,'[');
       for i=[1:1:length(Save.AGDL)]
           if i==length(Save.AGDL)
              fprintf(AAGDL,'%d',Save.AGDL(i));
```

```
            fprintf(AAGDL,']');
        else
            fprintf(AAGDL,'%d',Save.AGDL(i));
            fprintf(AAGDL,' ');
        end
    end

    ALazo=fopen('Lazo.txt','w');
    fprintf(ALazo,'[');
    for i=[1:1:length(Save.Lazo)]
        if i==length(Save.Lazo)
            fprintf(ALazo,'%d',Save.Lazo(i));
            fprintf(ALazo,']');
        else
            fprintf(ALazo,'%d',Save.Lazo(i));
            fprintf(ALazo,' ');
        end
    end

    ALazoPares=fopen('LazoPares.txt','w');
    fprintf(ALazoPares,'[');
    for i=[1:1:length(Save.LazoPares)]
        if i==length(Save.LazoPares)
            fprintf(ALazoPares,'%d',Save.LazoPares(i));
            fprintf(ALazoPares,']');
        else
            fprintf(ALazoPares,'%d',Save.LazoPares(i));
            fprintf(ALazoPares,' ');
        end
    end

    ACoord=fopen('Coord.txt','wt');
    J=[];
    k=1;
%     for i=[1:1:100]
%         if isnan(Save.Coord(i,1)) || isnan(Save.Coord(i,2))
%             k=k+1;
%         end
%         M(i,1)=Save.Coord(i,1);
%         M(i,2)=Save.Coord(i,2);
%     end
%     if k==101
%         fprintf(ACoord,'');
%     else
%        for i=[1:1:100-k+1]
%            J(i,1)=double(M(i,1));
%            J(i,2)=double(M(i,2));
%        end
J=Save.Coord;
    if isempty(J)
        fprintf(ACoord,'');
    else
        for i=[1:1:length(J(:,1))]
            p=double(J(i,1));
            s=double(J(i,2));
            fprintf(ACoord,'%d ',p);
            fprintf(ACoord,'%d',s);
            fprintf(ACoord,'\n');
        end
    end

    ADirecPRSD=fopen('DirecPRSD.txt','wt');
```

```
      J=[];
       k=1;
%       for i=[1:1:100]
%           if isnan(Save.DirecPRSD(i,3))
%               k=k+1;
%           end
%           M(i,1)=Save.DirecPRSD(i,1);
%           M(i,2)=Save.DirecPRSD(i,2);
%           M(i,3)=Save.DirecPRSD(i,3);
%       end
%       if k==101
%           fprintf(ADirecPRSD,'');
%       else
%          for i=[1:1:100-k+1]
%              J(i,1)=double(M(i,1));
%              J(i,2)=double(M(i,2));
%              J(i,3)=double(M(i,3));
%          end
J=Save.DirecPRSD;
    if isempty(J)
        fprintf(ADirecPRSD,'');
    else
         for i=[1:1:length(J(:,1))]
             p=double(J(i,1));
             s=double(J(i,2));
             t=double(J(i,3));
             fprintf(ADirecPRSD,'%d ',p);
             fprintf(ADirecPRSD,'%d ',s);
             fprintf(ADirecPRSD,'%d',t);
             fprintf(ADirecPRSD,'\n');
         end
    end

      ARadios=fopen('Radios.txt','wt');
      J=[];
      k=1;
%       for i=[1:1:100]
%           if isnan(Save.Radios(i,1)) || isnan(Save.Radios(i,2))
%               k=k+1;
%           end
%           M(i,1)=Save.Radios(i,1);
%           M(i,2)=Save.Radios(i,2);
%       end
%       if k==101
%           fprintf(ARadios,'');
%       else
%          for i=[1:1:100-k+1]
%              J(i,1)=double(M(i,1));
%              J(i,2)=double(M(i,2));
%          end
J=Save.Radios;
    if isempty(J)
        fprintf(ARadios,'');
    else
         for i=[1:1:length(J(:,1))]
             p=double(J(i,1));
             s=double(J(i,2));
             fprintf(ARadios,'%d ',p);
             fprintf(ARadios,'%d',s);
             fprintf(ARadios,'\n');
         end
```

```
   end


     ADirecPP=fopen('DirecPP.txt','wt');
     J=[];
     k=1;
%      for i=[1:1:100]
%          if isnan(Save.DirecPP(i,1)) || isnan(Save.DirecPP(i,2))
%              k=k+1;
%          end
%          M(i,1)=Save.DirecPP(i,1);
%          M(i,2)=Save.DirecPP(i,2);
%      end
%      if k==101
%          fprintf(ADirecPP,'');
%      else
%         for i=[1:1:100-k+1]
%             J(i,1)=double(M(i,1));
%             J(i,2)=double(M(i,2));
%         end
J=Save.DirecPP;
   if isempty(J)
       fprintf(ADirecPP,'');
   else
        for i=[1:1:length(J(:,1))]
            p=double(J(i,1));
            s=double(J(i,2));
            fprintf(ADirecPP,'%d ',p);
            fprintf(ADirecPP,'%d',s);
            fprintf(ADirecPP,'\n');
        end
   end


 fclose all;


cd(DirectorioA);
 s=1;

end
```

HAZDIBUJO.M

```
function [] = HazDibujo(Meca)
%% Función que representa el dibujo del mecanismo %%

%     axis([-15 15 -15 15])
    axis auto
    title('Mecanismo en la posición dada')
    % Muestreo de los pares del mecanismo
       MP=MuestraPares(Meca);
       if MP==1
       else
           fprintf('\n --> >>ERROR<< Muestreo de los pares\n')
       end
    %
    % Muestreo de los cuerpos del mecanismo
       MC=MuestraCuerpos(Meca);
       if MC==1
       else
           fprintf('\n --> >>ERROR<< Muestreo de los cuerpos\n')
       end
    %
    % Muestreo de las velocidades de cada Par
       MV=MuestraVelocidades(Meca);
       if MV==1
       else
           fprintf('\n --> >>ERROR<<  Muestreo de las velocidades\n')
       end
    %
    % Muestreo de las velocidades de cada Par
    if Meca.Prob==2
       MA=MuestraAceleraciones(Meca);
       if MA==1
       else
           fprintf('\n --> >>ERROR<<  Muestreo de las Aceleraciones\n')
       end
    end
    %

end
```

HAZSIMULACION.M

```
function [] = HazSimulacion(Meca,Vista)
%% Función que representa la simulación del mecanismo %%

    axis([Vista.Xmin Vista.Xmax Vista.Ymin Vista.Xmax])

%     axis auto
  title('Simulación en movimiento del mecanismo')
    Meca.ResTiempo=[];
    Meca.Error=0;
    for i=[1:1:Vista.Npasos]

        Meca.Iteracion=i;
       pause(0.05);
       cla
          % Muestreo de los pares del mecanismo
            MP=MuestraPares(Meca);
            if MP==1
            else
                fprintf('\n --> >>ERROR<< Muestreo de los pares\n')
            end
          %
          % Muestreo de los cuerpos del mecanismo
             MC=MuestraCuerpos(Meca);
             if MC==1
             else
                fprintf('\n --> >>ERROR<< Muestreo de los cuerpos\n')
             end
         %
         % Muestreo de las velocidades de cada Par
            MV=MuestraVelocidades(Meca);
            if MV==1
            else
              fprintf('\n --> >>ERROR<<  Muestreo de las velocidades\n')
            end
         %
         % Muestreo de las Aceleraciones de cada Par
            MA=MuestraAceleraciones(Meca);
              if MA==1
              else
                fprintf('\n --> >>ERROR<<  Muestreo de las Aceleraciones\n')
              end
         %
            Coord=[];
            Meca=NuevasCoord(Meca);
            for j=[1:1:Meca.Nrest]
               Coord=[Coord;Meca.Rela(j).x Meca.Rela(j).y];
            end

            Meca.ResTiempo=[Meca.ResTiempo;i
[Meca.Rela(3).Ax+Meca.Rela(3).Ay]];
            Meca=NuevasGDL(Meca);
            Meca.Coord=Coord;
            Meca=Calculador(Meca);

    end
end
```

LISTADOCUERPOS.M

```
function [ListadoC] = ListadoCuerpos(Meca)
%% Función que crea el listado lso cuerpos existentes en Meca

ListadoC='';
for i=[1:1:Meca.Nbody]
    Frase='Eslabón ';
    n=num2str(i);
    Frase=[Frase n];
    ListadoC=[ListadoC;Frase];
end

end
```

LISTADORELACIONES.M

```
function [ListadoR] = ListadoRelaciones(Meca)
%% Función que crea el listado de las relaciones existentes en Meca

ListadoR='';
for i=[1:1:length(Meca.Rela)]
    Frase='Relación ';
    n=num2str(i);
    Frase=[Frase n];
    ListadoR=[ListadoR;Frase];
end

end
```

MATRIZAH.M

```
function [Meca] = MatrizAH(Meca)
%% Programa que calcula la matriz AH relacionada con los términos de
% aceleraciones relativa de los pares de rodadura sin deslizamiento aij

AH=zeros(length(Meca.Grupo),1);
if Meca.NPP==0
  cont=1;
  P=[0;0];
    for i=[1:1:length(Meca.Grupo(1).G)]
        u=Meca.Grupo(1).G(i);
           if Meca.Rela(u).Codig==3
               wi=Meca.Cuerpo(u).W;
                  if u==Meca.Nbody
                      wj=0;
                  else
                      wj=Meca.Cuerpo(u+1).W;
                  end
                Ri=Meca.Radios(cont,1);
                Rj=Meca.Radios(cont,2);
                landa=Meca.DirecPRSD(cont,:);
                landa=(landa)./norm(landa);
                   if Meca.Radios(cont,3)==1
                        aij=((wi-wj).^2).*((Ri.*Rj)./(Ri-Rj)).*landa;
                   end
                   if Meca.Radios(cont,3)==2
                        aij=((wi-wj).^2).*((Ri.*Rj)./(Ri+Rj)).*landa;
                   end
                   if Meca.Radios(cont,3)==3
                        aij=((wi-wj).^2).*(Ri).*landa;
                   end
                   if Meca.Radios(cont,3)==4
                        aij=((wi-wj).^2).*(Rj).*landa;
                   end

                P=P+aij';
                cont=cont+1;
           else
               P=[0;0];
           end
    end
  AH=P;

else
   cont=1;
      for i=[1:1:length(Meca.Grupo)]
          for j=[1:1:length(Meca.Grupo(i).G)]
              u=Meca.Grupo(i).G(j);
              if Meca.LazoPares(u)==3
                  wi=Meca.Cuerpo(u).W;
                  if u==Meca.Nbody
                      wj=0;
                  else
                      wj=Meca.Cuerpo(u+1).W;
                  end
                  Ri=Meca.Radios(cont,1);
                  Rj=Meca.Radios(cont,2);
```

```
                landa=Meca.DirecPRSD(cont,:);
                landa=(landa)./norm(landa)';
                if Meca.Radios(cont,3)==1
                    aij=((wi-wj).^2).*((Ri.*Rj)./(Ri-Rj)).*landa;
                end
                if Meca.Radios(cont,3)==2
                    aij=((wi-wj).^2).*((Ri.*Rj)./(Ri+Rj)).*landa;
                end
                if Meca.Radios(cont,3)==3
                    aij=((wi-wj).^2).*(Ri).*landa;
                end
                if Meca.Radios(cont,3)==4
                    aij=((wi-wj).^2).*(Rj).*landa;
                end
                hp=Meca.DirecPP(i,:)';
                hp=[-hp(2);hp(1)];
                AH(i,1)=sum(aij'.*hp);
                cont=cont+1;
            end
        end
    end

end
  Meca.AH=AH;


end
```

MATRIZHV.M

```
function [Meca] = MatrizHV(Meca)
%% Función que genera la matriz HV %%

   % Determinación de las Vi de cada grupo
      MV=[];
      for i=[1:1:length(Meca.Grupo)]
          ini=Meca.Grupo(i).G(1);
          fin=Meca.Grupo(i).G(end);
%           if Meca.Lazo(end)~=1 && i==length(Meca.Grupo)
%              MV=[MV;ini,Meca.Lazo(fin)];
%           else
            MV=[MV;ini,Meca.Lazo(fin+1)];
%           end
      end
      Meca.MV=MV;
   %

   % Creación de la matriz HV
      linea=[1:1:Meca.NPP,1];
      n=length(Meca.MV(:,1));
      k=1;
      Maux=[];
      for i=[1:1:length(linea)-1]
          Laux=[linea(i),linea(i+1)];
          Maux=[Maux;Laux];
      end
      hv=zeros(Meca.NPP);
      for i=[1:1:length(Maux(:,1))]
         u=Maux(i,1);
         v=Maux(i,2);
         hv(i,u)=-1;
         hv(i,v)=1;
      end
      % Determinación de la submatriz Hpx
        Z=zeros(length(hv(:,1)),1);
        Hpx=[];
        VHpx=[];
         for i=[1:1:length(hv(1,:))]
             Hpx=[Hpx,hv(:,i)];
             Hpx=[Hpx,Z];
         end
         for i=[1:1:Meca.NPP]
             h=Meca.DirecPP(i,:);
             hp=[-h(2);h(1)];
             VHpx=[VHpx;hp(1)];
         end
         for i=[1:1:length(VHpx)]
             for j=[1:1:length(Hpx(1,:))]
                 Hpx(i,j)=Hpx(i,j).*VHpx(i);
             end
         end
      %
      % Determinación de la submatriz Hpy
        Hpy=Z;
        VHpy=[];
         for i=[1:1:length(hv(1,:))]
```

```
            Hpy=[Hpy,hv(:,i)];
            Hpy=[Hpy,Z];
        end
        Hpy(:,end)=[];
        for i=[1:1:Meca.NPP]
            h=Meca.DirecPP(i,:);
            hp=[-h(2);h(1)];
            VHpy=[VHpy;hp(2)];
        end
        for i=[1:1:length(VHpy)]
            for j=[1:1:length(Hpy(1,:))]
                Hpy(i,j)=Hpy(i,j).*VHpy(i);
            end
        end
    %
    HV=Hpx+Hpy;

    % Eliminación de las columnas relacionadas con V1
        HV(:,2)=[];
        HV(:,1)=[];
    %
    Meca.HV=HV;

  %

end
```

MATRIZTH.M

```
function [Meca] = MatrizTH(Meca)
%% Función que genera la Matriz TH %%

TH=zeros(length(Meca.Grupo),Meca.Nbody);

   for i=[1:1:length(Meca.Grupo)]
       for j=[1:1:length(Meca.Grupo(i).G)]
           u=Meca.Grupo(i).G(j);
           T=Meca.Tramo(u).T;
           T=[-T(2);T(1)];
           hp=Meca.DirecPP(i,:)';
           hp=[-hp(2);hp(1)];
           TH(i,u)=sum(T.*hp);
       end
   end
   TH(:,Meca.Nbody)=[];
   Meca.TH=TH;

end
```

MATRIZWTH.M

```
function [Meca] = MatrizWTH(Meca)
%% Programa que calcula la matriz WTH

WTH=zeros(length(Meca.Grupo),Meca.Nbody);
if Meca.NPP==0
  P=0;
    for i=[1:1:length(Meca.Grupo(1).G)]
        u=Meca.Grupo(1).G(i);
          if u==Meca.Nbody
              w=0;
          else
              w=Meca.Cuerpo(u+1).W;
          end
        T=Meca.Tramo(u).T;
        P=P+T.*((w).^2);
    end
   WTH=P;
   Meca.WTH=WTH;
else
     for i=[1:1:length(Meca.Grupo)]
         for j=[1:1:length(Meca.Grupo(i).G)]
             u=Meca.Grupo(i).G(j);
             T=Meca.Tramo(u).T;
             T=[T(1);T(2)];
             hp=Meca.DirecPP(i,:)';
             hp=[-hp(2);hp(1)];
             if u==Meca.Nbody
                 w=0;
             else
                 w=Meca.Cuerpo(u+1).W;
             end
             WTH(i,u)=sum(T.*hp).*(w.^2);
         end
     end
     WTH(:,Meca.Nbody)=[];
     Meca.WTH=sum(WTH')';
end

end
```

MATRIZWVH.M

```
function [Meca] = MatrizWVH(Meca)
%% Función que calcula la matriz WVH relacionada con el par de rodadura sin
deslizamiento

WVH=zeros(length(Meca.Grupo),1);

if Meca.NPP==0
  P=[0;0];
    for i=[1:1:length(Meca.Grupo(1).G)]
        u=Meca.Grupo(1).G(i);
           if Meca.Rela(u).Codig==3
                if u==Meca.Nbody
                    wj=0;
                else
                    wj=Meca.Cuerpo(u+1).W;
                end
              vi=[Meca.Rela(u).Vx;Meca.Rela(u).Vy];
              wv=2.*[-vi(2);vi(1)].*wj;
              P=P+(wv);
           else
               P=[0;0];
           end
    end
    WVH=P;

else
      for i=[1:1:length(Meca.Grupo)]
          for j=[1:1:length(Meca.Grupo(i).G)]
              u=Meca.Grupo(i).G(j);
              if Meca.LazoPares(u)==3
                  if u==Meca.Nbody
                      wj=0;
                  else
                      wj=Meca.Cuerpo(u+1).W;
                  end
                  vi=[Meca.Rela(u).Vx;Meca.Rela(u).Vy];
                  wv=[-vi(2);vi(1)].*wj;
                  hp=Meca.DirecPP(i,:)';
                  hp=[-hp(2);hp(1)];
                  WVH(i,1)=2.*sum(wv.*hp);
              end
          end
      end
end
   Meca.WVH=WVH;


end
```

MUESTRAACELERACIONES.M

```
function [MA] = MuestraAceleraciones(Meca)
%% Función que muestra en un plot las velocidades de los pares del mecanismo
%%

MA=0;
hold on
for i=[1:1:length(Meca.Rela)]
    x=Meca.Rela(i).x;
    y=Meca.Rela(i).y;
    Ax=Meca.Rela(i).Ax;
    Ay=Meca.Rela(i).Ay;
    P=[x;y];
    A=[Ax;Ay];
    A=A./norm(A);
    Pfin=A+P;
    quiver3(P(1),P(2),0,Pfin(1)-P(1),Pfin(2)-
P(2),0,'g','linewidth',3,'MaxHeadSize',0.3)
end
% x0=0;
% y0=0;
% z0=0;
%
% x1=1;
% y1=2;
% z1=0;
% % axis([0 2 0 5 0 0 1 5]);
% hold on
% quiver3(x0,y0,z0,x1-x0,y1-y0,z1-z0,'r','linewidth',3,'MaxHeadSize',0.3)

MA=1;
end
```

MUESTRACUERPOS.M

```
function [MC] = MuestraCuerpos(Meca)
%% Función que muestra en un plot los cuerpos del mecanismo %%
   MC=0;
  for i=[1:1:length(Meca.Rela)-1]
      Pini=[Meca.Rela(i).x;Meca.Rela(i).y];
      Pfin=[Meca.Rela(i+1).x;Meca.Rela(i+1).y];
      line([Pini(1),Pfin(1)],[Pini(2),Pfin(2)],'Color','k','LineWidth',2)
      Cuerpo=Meca.Rela(i).Cuerpos(2);
      P=(Pini+Pfin)./2;
      P=P+[0.05;0.05];
      C=num2str(Cuerpo);
      text(P(1),P(2),C)
  end

  MC=1;
end
```

MUESTRAPARES.M

```
function [MP] = MuestraPares(Meca)
%% Función que muestra en un plot los pares del mecanismo %%

   MP=0;
   hold on
   g=[0.2;0];
   e=[0;0.1];
   k=1;
   for i=[1:1:length(Meca.Rela)]
       if Meca.Rela(i).Codig==1
           plot(Meca.Rela(i).x,Meca.Rela(i).y,'ob')
       end
       if Meca.Rela(i).Codig==2
           if i==length(Meca.Rela) && Meca.Bucle==1
           else
           x=Meca.Rela(i).x;
           y=Meca.Rela(i).y;
           P=[x;y];
           hp=Meca.Rela(i).hp;
           h=[-hp(2);hp(1)];
           hu=h./(norm(h));
           hpu=hp./norm(hp);
           P1=P-sum(g.*h).*hu+sum(e.*hp).*hpu;
           P2=P+sum(g.*h).*hu+sum(e.*hp).*hpu;
           P3=P+sum(g.*h).*hu-sum(e.*hp).*hpu;
           P4=P-sum(g.*h).*hu-sum(e.*hp).*hpu;
           line([P1(1),P2(1)],[P1(2),P2(2)],'LineWidth',2)
           line([P2(1),P3(1)],[P2(2),P3(2)],'LineWidth',2)
           line([P3(1),P4(1)],[P3(2),P4(2)],'LineWidth',2)
           line([P4(1),P1(1)],[P4(2),P1(2)],'LineWidth',2)
           end
       end
       if Meca.Rela(i).Codig==3
%            plot(Meca.Rela(i).x,Meca.Rela(i).y,'^b')
          Tipo=Meca.Radios(k,3);
          landa=Meca.DirecPRSD(k,:);
          landaT=([-landa(2) landa(1)])';
          landaN=((landa)./(norm(landa)))';

          %
            if Tipo==1
                R1=Meca.Radios(k,1);
                R2=Meca.Radios(k,2);

                P1=[Meca.Rela(i).x;Meca.Rela(i).y]+landaN.*R1;
                P2=[Meca.Rela(i).x;Meca.Rela(i).y]-landaN.*R2;
               t=linspace(0,2*pi,300);
               x1=R1*cos(t)+P1(1);
               y1=R1*sin(t)+P1(2);
               x2=R2*cos(t)+P2(1);
               y2=R2*sin(t)+P2(2);
               plot(x1,y1,'b','LineWidth',2)
               plot(x2,y2,'b','LineWidth',2)
            end
            if Tipo==2
                R1=Meca.Radios(k,1);
                R2=Meca.Radios(k,2);
```

```
                P1=[Meca.Rela(i).x;Meca.Rela(i).y]+landaN.*R1;
                P2=[Meca.Rela(i).x;Meca.Rela(i).y]+landaN.*R2;
               t=linspace(0,2*pi,300);
               x1=R1*cos(t)+P1(1);
               y1=R1*sin(t)+P1(2);
               x2=R2*cos(t)+P2(1);
               y2=R2*sin(t)+P2(2);
               plot(x1,y1,'b','LineWidth',2)
               plot(x2,y2,'b','LineWidth',2)
            end
            if Tipo==3
                R1=Meca.Radios(k,1);

                P1=[Meca.Rela(i).x;Meca.Rela(i).y]+landaN.*R1;
               t=linspace(0,2*pi,300);
               x1=R1*cos(t)+P1(1);
               y1=R1*sin(t)+P1(2);
               direcLanda=landaT./norm(landaT);
               Q1=[Meca.Rela(i).x;Meca.Rela(i).y]+direcLanda.*(R1/2);
               Q2=[Meca.Rela(i).x;Meca.Rela(i).y]-direcLanda.*(R1/2);
               plot(x1,y1,'b','LineWidth',2);
               line([Q1(1),Q2(1)],[Q1(2),Q2(2)],'LineWidth',2)
            end
            if Tipo==4
                R2=Meca.Radios(k,2);

                P1=[Meca.Rela(i).x;Meca.Rela(i).y]-landaN.*R2;
               t=linspace(0,2*pi,300);
               x1=R2*cos(t)+P1(1);
               y1=R2*sin(t)+P1(2);
               direcLanda=landaT./norm(landaT);
               Q1=[Meca.Rela(i).x;Meca.Rela(i).y]+direcLanda.*(R2/2);
               Q2=[Meca.Rela(i).x;Meca.Rela(i).y]-direcLanda.*(R2/2);
               plot(x1,y1,'b','LineWidth',2);
               line([Q1(1),Q2(1)],[Q1(2),Q2(2)],'LineWidth',2)
            end
          %

%           if Tipo==1 || Tipo==2
%            R1=Meca.Radios(k,1);
%            R2=Meca.Radios(k,2);
%            if i==1
%                x1=Meca.Rela(1).x;
%                y1=Meca.Rela(1).y;
%            else
%                x1=Meca.Rela(i-1).x;
%                y1=Meca.Rela(i-1).y;
%            end
%            if i==length(Meca.Rela)
%                x2=Meca.Rela(length(Meca.Rela)).x;
%                y2=Meca.Rela(length(Meca.Rela)).y;
%            else
%                x2=Meca.Rela(i+1).x;
%                y2=Meca.Rela(i+1).y;
%            end
%            t=linspace(0,2*pi,300);
%            x1=R1*cos(t)+x1;
%            y1=R1*sin(t)+y1;
%            x2=R2*cos(t)+x2;
%            y2=R2*sin(t)+y2;
%            plot(x1,y1,'b','LineWidth',2)
```

```
%             plot(x2,y2,'b','LineWidth',2)
%             end
%
%             if Tipo==3
%                 R1=Meca.Radios(k,1);
%                 t=linspace(0,2*pi,300);
%                 if i==1
%                     x1=Meca.Rela(1).x;
%                     y1=Meca.Rela(1).y;
%                 else
%                     x1=Meca.Rela(i-1).x;
%                     y1=Meca.Rela(i-1).y;
%                 end
%                 if i==length(Meca.Rela)
%                     x2=Meca.Rela(length(Meca.Rela)).x;
%                     y2=Meca.Rela(length(Meca.Rela)).y;
%                 else
%                     x2=Meca.Rela(i+1).x;
%                     y2=Meca.Rela(i+1).y;
%                 end
%
%                 x1=R1*cos(t)+x1;
%                 y1=R1*sin(t)+y1;
%                 direcLanda=landa./norm(landa);
%                 P1=[x2;y2]+direcLanda';
%                 P2=[x2;y2]-direcLanda';
%                 plot(x1,y1,'b','LineWidth',2);
%                 line([P1(1),P2(1)],[P1(2),P2(2)],'LineWidth',2)
%             end
%
%             if Tipo==4
%                 R2=Meca.Radios(k,2);
%                 t=linspace(0,2*pi,300);
%                 if i==1
%                     x1=Meca.Rela(1).x;
%                     y1=Meca.Rela(1).y;
%                 else
%                     x1=Meca.Rela(i-1).x;
%                     y1=Meca.Rela(i-1).y;
%                 end
%                 if i==length(Meca.Rela)
%                     x2=Meca.Rela(length(Meca.Rela)).x;
%                     y2=Meca.Rela(length(Meca.Rela)).y;
%                 else
%                     x2=Meca.Rela(i+1).x;
%                     y2=Meca.Rela(i+1).y;
%                 end
%
%                 x2=R2*cos(t)+x2;
%                 y2=R2*sin(t)+y2;
%                 direcLanda=landa./norm(landa);
%                 P1=[x1;y1]+direcLanda';
%                 P2=[x1;y1]-direcLanda';
%                 plot(x2,y2,'b','LineWidth',2);
%                 line([P1(1),P2(1)],[P1(2),P2(2)],'LineWidth',2)
%             end

            k=k+1;
        end
    end
    MP=1;

end
```

MUESTRARESULTADOS.M

```
function varargout = MuestraResultados(varargin)
% MUESTRARESULTADOS MATLAB code for MuestraResultados.fig
%      MUESTRARESULTADOS, by itself, creates a new MUESTRARESULTADOS or
raises the existing
%      singleton*.
%
%      H = MUESTRARESULTADOS returns the handle to a new MUESTRARESULTADOS or
the handle to
%      the existing singleton*.
%
%      MUESTRARESULTADOS('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in MUESTRARESULTADOS.M with the given input
arguments.
%
%      MUESTRARESULTADOS('Property','Value',...) creates a new
MUESTRARESULTADOS or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before MuestraResultados_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to MuestraResultados_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help MuestraResultados

% Last Modified by GUIDE v2.5 20-Jul-2016 02:48:50

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @MuestraResultados_OpeningFcn, ...
                   'gui_OutputFcn',  @MuestraResultados_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before MuestraResultados is made visible.
function MuestraResultados_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
```

```
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to MuestraResultados (see VARARGIN)

% Choose default command line output for MuestraResultados
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes MuestraResultados wait for user response (see UIRESUME)
% uiwait(handles.figure1);

Logo=imread('Logo1.png');
axes(handles.Logo);
image(Logo)
axis off
set(handles.CompA,'Visible','off');
set(handles.CompV,'Visible','off');

 Meca=evalin('base','Meca');
 ListaR=ListadoRelaciones(Meca);
 set(handles.ListaRela,'String',ListaR);
 Meca.ListaR=ListaR;
 assignin('base','Meca',Meca)

 ListaC=ListadoCuerpos(Meca);
 set(handles.ListaCuerpos,'String',ListaC);
 Meca.ListaC=ListaC;
 assignin('base','Meca',Meca)

if Meca.Prob==1 || Meca.Prob==2
   % Comprobador del resultado del problema de velocidad
         aux=sum(abs(Meca.CompruebaVel));
         ErrorV=log10(aux);
         if ErrorV==-Inf
             fprintf('\n--> Problema de velocidad correcto\n    con error
0\n')
               Verde=imread('Verde.png');
               axes(handles.CompV);
               image(Verde)
               axis off
         else
              if aux<10^-5
                 fprintf('\n--> Problema de velociad correcto\n    con un
errror del 10^%d\n',ErrorV)
                    Verde=imread('Verde.png');
                    axes(handles.CompV);
                    image(Verde)
                    axis off
              else
                 fprintf('\n--> Problema de velocidad\n    recuelto con algun
tipo de error\n')
                    Amarillo=imread('Amarillo.png');
                    axes(handles.CompV);
                    image(Amarillo)
                    axis off
              end
         end
         if Meca.Prob==1
             set(handles.CompA,'Visible','off');
```

```
        end
    %
end
if Meca.Prob==2
    % Comprobador del resultado del problema de aceleración
        aux=sum(abs(Meca.CompruebaAce));
        ErrorA=log10(aux);
        if ErrorA==-Inf
            fprintf('\n--> Problema de aceleración correcto\n    con error
0\n')
             Amarill=imread('Verde.png');
             axes(handles.CompA);
             image(Verde)
             axis off
        else
            if aux<10^-5
               fprintf('\n--> Problema de aceleración correcto\n    con un
errror del 10^%d\n',ErrorV)
             Verde=imread('Verde.png');
             axes(handles.CompA);
             image(Verde)
             axis off
            else
               fprintf('\n--> Problema de aceleración\n    recuelto con
algun tipo de error\n')
                  Amarillo=imread('Amarillo.png');
                  axes(handles.CompA);
                  image(Amarillo)
                  axis off
            end
        end
    %
end

if Meca.Dibujo==1
    axes(handles.Grafica);
    HazDibujo(Meca);
end

if Meca.Simulacion==1
    Vista=evalin('base','Vista');
    axes(handles.Grafica);
    HazSimulacion(Meca,Vista);
end


% --- Outputs from this function are returned to the command line.
function varargout = MuestraResultados_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in ListaRela.
function ListaRela_Callback(hObject, eventdata, handles)
% hObject    handle to ListaRela (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns ListaRela contents
as cell array
%        contents{get(hObject,'Value')} returns selected item from ListaRela
a=get(handles.ListaRela,'Value');
 Meca=evalin('base','Meca');
set(handles.ResC1,'String',Meca.Rela(a).Cuerpos(1));
set(handles.ResC2,'String',Meca.Rela(a).Cuerpos(2));
set(handles.ResX,'String',Meca.Rela(a).x);
set(handles.ResY,'String',Meca.Rela(a).y);
set(handles.ResVx,'String',Meca.Rela(a).Vx);
set(handles.ResVy,'String',Meca.Rela(a).Vy);
if Meca.Prob==2
   set(handles.ResAx,'String',Meca.Rela(a).Ax);
   set(handles.ResAy,'String',Meca.Rela(a).Ay);
end


% --- Executes during object creation, after setting all properties.
function ListaRela_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ListaRela (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in ListaCuerpos.
function ListaCuerpos_Callback(hObject, eventdata, handles)
% hObject    handle to ListaCuerpos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns ListaCuerpos
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
ListaCuerpos
b=get(handles.ListaCuerpos,'Value');
 Meca=evalin('base','Meca');
set(handles.ResW,'String',Meca.Cuerpo(b).W);
if Meca.Prob==2
   set(handles.ResA,'String',Meca.Cuerpo(b).A);
end

% --- Executes during object creation, after setting all properties.
function ListaCuerpos_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ListaCuerpos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in CreaInforme.
function CreaInforme_Callback(hObject, eventdata, handles)
% hObject    handle to CreaInforme (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

helpdlg('Si esta usanso MCR no se creará el informe. Por favor cierre el
programa y ejecute en Matlab 2012a o superior el archivo adjunto
CAC_2D_Matlab.m para poder crear el informe.','Advertencia del uso de MCR');

   Meca=evalin('base','Meca');
   if isempty(Meca.Nombre)
      Meca.Nombre='Mecanismo';
   end

  W=what;
DirectorioA=W.path;
DirectorioN=[DirectorioA '\Informes de archivos'];
DirectorioN=[DirectorioN '\'];

   opts.format='pdf';
   opts.showCode=false;
   opts.outputDir=DirectorioN;

   publish('CreaInforme',opts);

cd(DirectorioN)
Frase=['Informe mecanismo - ' Meca.Nombre '.pdf'];
movefile('CreaInforme.pdf',Frase);
open(Frase);

cd(DirectorioA)
```

MUESTRAVELOCIDADES.M

```
function [MV] = MuestraVelocidades(Meca)
%% Función que muestra en un plot las velocidades de los pares del mecanismo
%%

MV=0;
hold on
for i=[1:1:length(Meca.Rela)]
    x=Meca.Rela(i).x;
    y=Meca.Rela(i).y;
    Vx=Meca.Rela(i).Vx;
    Vy=Meca.Rela(i).Vy;
    P=[x;y];
    V=[Vx;Vy];
    V=V./norm(V);
    Pfin=V+P;
    quiver3(P(1),P(2),0,Pfin(1)-P(1),Pfin(2)-
P(2),0,'r','linewidth',3,'MaxHeadSize',0.3)
end
% x0=0;
% y0=0;
% z0=0;
%
% x1=1;
% y1=2;
% z1=0;
% % axis([0 2 0 5 0 0 1 5]);
% hold on
% quiver3(x0,y0,z0,x1-x0,y1-y0,z1-z0,'r','linewidth',3,'MaxHeadSize',0.3)

MV=1;
end
```

NOPP.M

```
function [Meca] = NOPP(Meca)
%% Función que resuelve el caso particular de NPP=0 %%

   M=[];
   for i=[1:1:length(Meca.Tramo)]
       Tx=Meca.Tramo(i).T(1);
       Ty=Meca.Tramo(i).T(2);
       V=[-Ty;Tx];
       M=[M,V];
   end

    M(:,end)=[];
   Meca.M=M;


end
```

NUEVASCOORD.M

```
function [Meca] = NuevasCoord(Meca)
%% Función que calcula las nuevas coordenadas del mecanismo %%

t=0.02;
Error=0;
for i=[1:1:Meca.Nrest]
    Xa=Meca.Rela(i).x;
    Ya=Meca.Rela(i).y;
    Va=[Meca.Rela(i).Vx;Meca.Rela(i).Vy];
    Aa=[Meca.Rela(i).Ax;Meca.Rela(i).Ay];

    if norm(Aa)==0
        if norm(Va)==0
            Pn=[Xa;Ya];
        else
           V=Va;
           direcV=V./norm(V);
           moduV=norm(V)*t;
           Xn=moduV.*direcV;
           Pn=[Xa;Ya]+Xn;
        end
    else
       direcA=Aa./norm(Aa);
       moduV=norm(Aa)*t/2;
%        Error=(norm(Aa)*t)/(24*Meca.Iteracion);
%        moduV=moduV+Error;
       Vn=moduV.*direcA;
       V=Va+Vn;

       direcV=V./norm(V);
       moduV=norm(V)*t;
       Xn=moduV.*direcV;

       Pn=[Xa;Ya]+Xn;
    end

       Meca.Rela(i).x=Pn(1);
       Meca.Rela(i).y=Pn(2);
       Meca.Error=Error;
end


end
```

NUEVASGDL.M

```
function [Meca] = NuevasGDL(Meca)
%% Función que calcula las nuevas velocidades iniciales de los GDL %%

t=0.02;
AAa=[];
VAa=[];
f=[];
k=0;
for i=[1:1:Meca.Nbody-1]
%     AAa=[AAa;Meca.Cuerpo(i+1).A];
    r=Meca.AGDL(i);
%     f=[f;r];
    AAa=[AAa;r];
    r=Meca.VGDL(i);
    VAa=[VAa;r];
end
% AAn=AAa;
% for i=[1:1:length(f)]
%     if f(i)==0
%         AAn(i)=0;
%     end
% end

VAn=VAa+AAa.*t;

VLn=[];
if Meca.NPP>=2
    Linea=Meca.RCP(:,1);
    for i=[1:1:length(Linea)]
        if Linea(i)==1
        else
           ALa=[Meca.Rela(Linea(i)).Ax;Meca.Rela(Linea(i)).Ay];
           VLa=[Meca.Rela(Linea(i)).Vx;Meca.Rela(Linea(i)).Vy];
            if norm(ALa)==0
                VLn=VLa;
            else
               direcA=ALa./norm(ALa);
               moduA=norm(ALa);

               velo=VLa+((moduA*t).*direcA);
               VLn=[VLn;velo];
            end
        end
    end
end

Meca.VGDL=[VAn;VLn];

end
```

PRESIMULACION.M

```
function varargout = PreSimulacion(varargin)
% PRESIMULACION MATLAB code for PreSimulacion.fig
%      PRESIMULACION, by itself, creates a new PRESIMULACION or raises the
existing
%      singleton*.
%
%      H = PRESIMULACION returns the handle to a new PRESIMULACION or the
handle to
%      the existing singleton*.
%
%      PRESIMULACION('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in PRESIMULACION.M with the given input
arguments.
%
%      PRESIMULACION('Property','Value',...) creates a new PRESIMULACION or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before PreSimulacion_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to PreSimulacion_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help PreSimulacion

% Last Modified by GUIDE v2.5 04-Jul-2016 17:40:44

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @PreSimulacion_OpeningFcn, ...
                   'gui_OutputFcn',  @PreSimulacion_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before PreSimulacion is made visible.
function PreSimulacion_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% varargin   command line arguments to PreSimulacion (see VARARGIN)
set(handles.Xmin,'String','-15');
set(handles.Ymin,'String','-15');
set(handles.Xmax,'String','15');
set(handles.Ymax,'String','15');
set(handles.Npasos,'String','50');
% Choose default command line output for PreSimulacion
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes PreSimulacion wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = PreSimulacion_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in Continuar.
function Continuar_Callback(hObject, eventdata, handles)
% hObject    handle to Continuar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Xmin=get(handles.Xmin,'String');
if isempty(Xmin)
    warndlg('Rellene el campo X mínima','Mensaje');
else
    Xmin=str2double(Xmin);
end
Ymin=get(handles.Ymin,'String');
if isempty(Xmin)
    warndlg('Rellene el campo Y mínima','Mensaje');
else
    Ymin=str2double(Ymin);
end
Xmax=get(handles.Xmax,'String');
if isempty(Xmax)
    warndlg('Rellene el campo X máxima','Mensaje');
else
    Xmax=str2double(Xmax);
end
Ymax=get(handles.Ymax,'String');
if isempty(Ymax)
    warndlg('Rellene el campo Y máxima','Mensaje');
else
    Ymax=str2double(Ymax);
end
Npasos=get(handles.Npasos,'String');
if isempty(Npasos)
    warndlg('Rellene el campo Número de pasos','Mensaje');
else
```

```
    Npasos=str2double(Npasos);
end
Vista.Xmin=Xmin;
Vista.Ymin=Ymin;
Vista.Xmax=Xmax;
Vista.Ymax=Ymax;
Vista.Npasos=Npasos;
assignin('base','Vista',Vista);
close
MuestraResultados();



function Npasos_Callback(hObject, eventdata, handles)
% hObject    handle to Npasos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Npasos as text
%        str2double(get(hObject,'String')) returns contents of Npasos as a
double


% --- Executes during object creation, after setting all properties.
function Npasos_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Npasos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Xmin_Callback(hObject, eventdata, handles)
% hObject    handle to Xmin (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Xmin as text
%        str2double(get(hObject,'String')) returns contents of Xmin as a
double


% --- Executes during object creation, after setting all properties.
function Xmin_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Xmin (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function Ymin_Callback(hObject, eventdata, handles)
% hObject    handle to Ymin (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Ymin as text
%        str2double(get(hObject,'String')) returns contents of Ymin as a
double


% --- Executes during object creation, after setting all properties.
function Ymin_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Ymin (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Ymax_Callback(hObject, eventdata, handles)
% hObject    handle to Ymax (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Ymax as text
%        str2double(get(hObject,'String')) returns contents of Ymax as a
double


% --- Executes during object creation, after setting all properties.
function Ymax_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Ymax (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Xmax_Callback(hObject, eventdata, handles)
% hObject    handle to Xmax (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of Xmax as text
%        str2double(get(hObject,'String')) returns contents of Xmax as a
double


% --- Executes during object creation, after setting all properties.
function Xmax_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Xmax (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

PROBACEL.M

```
function [Meca] = ProbAcel(Meca)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Programa para resolver el problema de aceleraciones %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Cálculo de la matriz WTH
   Meca=MatrizWTH(Meca);
%

% Cálculo de la matriz AH
   Meca=MatrizAH(Meca);
%

% Cálculo de la matriz WVH
   Meca=MatrizWVH(Meca);
%
% Aplicación de los GDL y creación de las matrices Meca.Qind y Meca.Qdep
   Meca=CondAGDL(Meca);
%

% Resolución del problema de aceleración
Meca.Qind
Meca.Qind=(Meca.Qind)-Meca.WTH-Meca.AH-Meca.WVH;
Meca.Qind
 Sol=Meca.Qdep\-Meca.Qind
 Sol=Meca.AGDL+Sol;
 Meca.SolAcel=Sol;
 if Meca.NPP~=0
     Meca=CondPrisSolA(Meca);
 end

%



end
```

PRUEBA11.M: CÓDIGO PRINCIPAL

```
function varargout = CAC_2D(varargin)
% PRUEBA11 MATLAB code for Prueba11.fig
%      PRUEBA11, by itself, creates a new PRUEBA11 or raises the existing
%      singleton*.
%
%      H = PRUEBA11 returns the handle to a new PRUEBA11 or the handle to
%      the existing singleton*.
%
%      PRUEBA11('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PRUEBA11.M with the given input arguments.
%
%      PRUEBA11('Property','Value',...) creates a new PRUEBA11 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Prueba11_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Prueba11_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Prueba11

% Last Modified by GUIDE v2.5 19-Jul-2016 04:45:04

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Prueba11_OpeningFcn, ...
                   'gui_OutputFcn',  @Prueba11_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Prueba11 is made visible.
function Prueba11_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Prueba11 (see VARARGIN)

% Choose default command line output for Prueba11
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);

% clc
Logo=imread('Logo1.png');
axes(handles.Logo);
imshow(Logo)
axis off

SPR=imread('PR_S1.jpg');
axes(handles.Simbolo_PR);
image(SPR)
axis off

SPP=imread('PP_S.jpg');
axes(handles.Simbolo_PP);
image(SPP)
axis off

SPRSD=imread('PRSD_S.jpg');
axes(handles.Simbolo_PRSD);
image(SPRSD)
axis off

   W=what;
   Directorio=W.path;
   Directorio=[Directorio '\'];
   mkdir([Directorio,'Archivos guardados'])

   W=what;
   Directorio=W.path;
   Directorio=[Directorio '\'];
   mkdir([Directorio,'Informes de archivos'])
% UIWAIT makes Prueba11 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Prueba11_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function NPR_Callback(hObject, eventdata, handles)
% hObject    handle to NPR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of NPR as text
%        str2double(get(hObject,'String')) returns contents of NPR as a
double
Nbody=get(handles.Nbody,'String');
Nrest=get(handles.Nrest,'String');
```

```
NPR=get(handles.NPR,'String');
NPP=get(handles.NPP,'String');
NPRSD=get(handles.NPRSD,'String');
Lazo=get(handles.Lazo,'String');
LazoPares=get(handles.LazoPares,'String');
NPRSD=str2double(NPRSD);
    if NPRSD>=1
       set(handles.Simulacion,'Enable','off');
       set(handles.Simulacion,'Value',0);
    end
    if NPRSD==0
      set(handles.Simulacion,'Enable','on');
    end

if isempty(Nbody) || isempty(Nrest) || isempty(NPR) || isempty(NPP) ||
isempty(NPRSD) || isempty(Lazo) || isempty(LazoPares)
    set(handles.VGDL,'Enable','off');
    set(handles.AGDL,'Enable','off');
    Frase='Faltan datos del mecanismo';
    set(handles.TextVang,'FontName','MS Sans Serif');
    set(handles.TextAang,'FontName','MS Sans Serif');
    set(handles.TextVang,'String',Frase);
    set(handles.TextAang,'String',Frase);
else
   set(handles.VGDL,'Enable','on');
   set(handles.AGDL,'Enable','on');
   Meca.Nbody=str2double(Nbody);
   Meca.Nrest=str2double(Nrest);
   Meca.NPR=str2double(NPR);
   Meca.NPP=str2double(NPP);
   Meca.NPRSD=str2double(NPRSD);
   Meca.Lazo=str2num(Lazo);
   Meca.LazoPares=str2num(LazoPares);
      RCP=[];
        Meca=BucleAC(Meca);
      for i=[1:1:length(Meca.LazoPares)]
         if Meca.LazoPares(i)==2
             C1=Meca.Lazo(i-1);
             C2=Meca.Lazo(i);
             RCP=[RCP;C1,C2];
         end
      end
      Meca.RCP=RCP;
    Meca=CreaChar(Meca);
    set(handles.TextVang,'FontName','Symbol');
    set(handles.TextAang,'FontName','Symbol');
    set(handles.TextVang,'String',Meca.GDL.CharV.ang);
    set(handles.TextAang,'String',Meca.GDL.CharA.ang);
    set(handles.TextVlin,'FontName','MS Sans Serif');
    set(handles.TextAling,'FontName','MS Sans Serif');
    set(handles.TextVlin,'String',Meca.GDL.CharV.lin);
    set(handles.TextAlin,'String',Meca.GDL.CharA.lin);

end


% --- Executes during object creation, after setting all properties.
function NPR_CreateFcn(hObject, eventdata, handles)
% hObject    handle to NPR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function NPP_Callback(hObject, eventdata, handles)
% hObject    handle to NPP (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of NPP as text
%        str2double(get(hObject,'String')) returns contents of NPP as a
double
Nbody=get(handles.Nbody,'String');
Nrest=get(handles.Nrest,'String');
NPR=get(handles.NPR,'String');
NPP=get(handles.NPP,'String');
NPRSD=get(handles.NPRSD,'String');
Lazo=get(handles.Lazo,'String');
LazoPares=get(handles.LazoPares,'String');

if isempty(Nbody) || isempty(Nrest) || isempty(NPR) || isempty(NPP) ||
isempty(NPRSD) || isempty(Lazo) || isempty(LazoPares)
    set(handles.VGDL,'Enable','off');
    set(handles.AGDL,'Enable','off');
    Frase='Faltan datos del mecanismo';
    set(handles.TextVang,'FontName','MS Sans Serif');
    set(handles.TextAang,'FontName','MS Sans Serif');
    set(handles.TextVang,'String',Frase);
    set(handles.TextAang,'String',Frase);
else
   set(handles.VGDL,'Enable','on');
   set(handles.AGDL,'Enable','on');
   Meca.Nbody=str2double(Nbody);
   Meca.Nrest=str2double(Nrest);
   Meca.NPR=str2double(NPR);
   Meca.NPP=str2double(NPP);
   Meca.NPRSD=str2double(NPRSD);
   Meca.Lazo=str2num(Lazo);
   Meca.LazoPares=str2num(LazoPares);
      RCP=[];
        Meca=BucleAC(Meca);
      for i=[1:1:length(Meca.LazoPares)]
         if Meca.LazoPares(i)==2
             C1=Meca.Lazo(i-1);
             C2=Meca.Lazo(i);
             RCP=[RCP;C1,C2];
         end
      end
      Meca.RCP=RCP;
      Meca=CreaChar(Meca);
    set(handles.TextVang,'FontName','Symbol');
    set(handles.TextAang,'FontName','Symbol');
    set(handles.TextVang,'String',Meca.GDL.CharV.ang);
    set(handles.TextAang,'String',Meca.GDL.CharA.ang);
    set(handles.TextVlin,'FontName','MS Sans Serif');
    set(handles.TextAlin,'FontName','MS Sans Serif');
```

```
    set(handles.TextVlin,'String',Meca.GDL.CharV.lin);
    set(handles.TextAlin,'String',Meca.GDL.CharA.lin);
end


% --- Executes during object creation, after setting all properties.
function NPP_CreateFcn(hObject, eventdata, handles)
% hObject    handle to NPP (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function NPRSD_Callback(hObject, eventdata, handles)
% hObject    handle to NPRSD (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of NPRSD as text
%        str2double(get(hObject,'String')) returns contents of NPRSD as a
double
Nbody=get(handles.Nbody,'String');
Nrest=get(handles.Nrest,'String');
NPR=get(handles.NPR,'String');
NPP=get(handles.NPP,'String');
NPRSD=get(handles.NPRSD,'String');
Lazo=get(handles.Lazo,'String');
LazoPares=get(handles.LazoPares,'String');
NPRSD=str2double(NPRSD);
    if NPRSD>=1
       set(handles.Simulacion,'Enable','off');
       set(handles.Simulacion,'Value',0);
    end
    if NPRSD==0
      set(handles.Simulacion,'Enable','on');
    end

if isempty(Nbody) || isempty(Nrest) || isempty(NPR) || isempty(NPP) ||
isempty(NPRSD) || isempty(Lazo) || isempty(LazoPares)
    set(handles.VGDL,'Enable','off');
    set(handles.AGDL,'Enable','off');
    Frase='Faltan datos del mecanismo';
    set(handles.TextVang,'FontName','MS Sans Serif');
    set(handles.TextAang,'FontName','MS Sans Serif');
    set(handles.TextVang,'String',Frase);
    set(handles.TextAang,'String',Frase);
else
   set(handles.VGDL,'Enable','on');
   set(handles.AGDL,'Enable','on');
   Meca.Nbody=str2double(Nbody);
   Meca.Nrest=str2double(Nrest);
   Meca.NPR=str2double(NPR);
   Meca.NPP=str2double(NPP);
   Meca.NPRSD=str2double(NPRSD);
   Meca.Lazo=str2num(Lazo);
```

```
   Meca.LazoPares=str2num(LazoPares);
      RCP=[];
        Meca=BucleAC(Meca);
      for i=[1:1:length(Meca.LazoPares)]
         if Meca.LazoPares(i)==2
             C1=Meca.Lazo(i-1);
             C2=Meca.Lazo(i);
             RCP=[RCP;C1,C2];
         end
      end
      Meca.RCP=RCP;
      Meca=CreaChar(Meca);
    set(handles.TextVang,'FontName','Symbol');
    set(handles.TextAang,'FontName','Symbol');
    set(handles.TextVang,'String',Meca.GDL.CharV.ang);
    set(handles.TextAang,'String',Meca.GDL.CharA.ang);
    set(handles.TextVlin,'FontName','MS Sans Serif');
    set(handles.TextAlin,'FontName','MS Sans Serif');
    set(handles.TextVlin,'String',Meca.GDL.CharV.lin);
    set(handles.TextAlin,'String',Meca.GDL.CharA.lin);
end

% --- Executes during object creation, after setting all properties.
function NPRSD_CreateFcn(hObject, eventdata, handles)
% hObject    handle to NPRSD (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in Resolver.
function Resolver_Callback(hObject, eventdata, handles)
% hObject    handle to Resolver (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

Nbody=get(handles.Nbody,'String');
if isempty(Nbody)
    warndlg('Rellene el campo Número de eslabones','Mensaje');
else
    Nbody=str2double(Nbody);
end
Nrest=get(handles.Nrest,'String');
if isempty(Nrest)
    warndlg('Rellene el campo Número de pares de restricción','Mensaje');
else
    Nrest=str2double(Nrest);
end
NPR=get(handles.NPR,'String');
if isempty(NPR)
    warndlg('Rellene el campo Pares de revolución','Mensaje');
else
    NPR=str2double(NPR);
end
NPP=get(handles.NPP,'String');
```

```
if isempty(NPP)
    warndlg('Rellene el campo Pares prismáticos','Mensaje');
else
    NPP=str2double(NPP);
end
NPRSD=get(handles.NPRSD,'String');
if isempty(NPRSD)
    warndlg('Rellene el campo Pares de rodadura sin
deslizamiento','Mensaje');
else
    NPRSD=str2double(NPRSD);
end
DirecPP=get(handles.DirecPP,'Data');
if isempty(DirecPP) && NPP>=1
    warndlg('Rellene el campo Direcciones de los pares
prismáticos','Mensaje');
else
%       DirecPP=str2double(DirecPP);
   assignin('base','M',DirecPP);
   if iscell(DirecPP)
      DirecPP=cell2mat(DirecPP);
   end
end
DirecPRSD=get(handles.DirecPRSD,'Data');
if isempty(DirecPRSD) && NPRSD>=1
    warndlg('Rellene el campo Direcciones de los pares de rodadura sin
deslizamiento','Mensaje');
else
%     DirecPRSD=str2double(DirecPRSD);
     if iscell(DirecPRSD)
        DirecPRSD=cell2mat(DirecPRSD);
     end
end
Radios=get(handles.Radios,'Data');
if isempty(Radios) && NPRSD>=1
    warndlg('Rellene el campo Radios de curvatura','Mensaje');
else
%     Radios=str2double(Radios);
    if iscell(Radios)
        Radios=cell2mat(Radios);
    end
end
Coord=get(handles.Coord,'Data');
if isempty(Coord)
    warndlg('Rellene el campo Coordenadas','Mensaje');
else
%     Coord=str2double(Coord)
   if iscell(Coord)
      Coord=cell2mat(Coord);
   end
end
Lazo=get(handles.Lazo,'String');
if isempty(Lazo)
    warndlg('Rellene el campo Lazo','Mensaje');
else
    Lazo=str2num(Lazo);
end
LazoPares=get(handles.LazoPares,'String');
if isempty(LazoPares)
    warndlg('Rellene el campo Lazo de los pares','Mensaje');
else
    LazoPares=str2num(LazoPares);
end
```

```
VGDL=get(handles.VGDL,'String');
if isempty(VGDL)
    warndlg('Rellene el campo Velocidades iniciales','Mensaje');
else
    VGDL=str2num(VGDL)';
end
AGDL=get(handles.AGDL,'String');
if isempty(AGDL)
    warndlg('Rellene el campo Aceleraciones iniciales','Mensaje');
else
    AGDL=str2num(AGDL)';
end

if NPRSD>=1
Radios=[Radios DirecPRSD(:,end)];
DirecPRSD(:,end)=[];
end

Velo=get(handles.Velo,'Value');
VeloyAce=get(handles.VeloyAce,'Value');
Simulacion=get(handles.Simulacion,'Value');
  if Velo==1
      Prob=1;
  end
  if VeloyAce==1
      Prob=2;
  end
       if NPRSD>=1 && Simulacion==1
        warndlg('No se podrá representar la simulación del mecanismo. Lea
condiciones de la simulación','Mensaje');
        Simulacion=0;
       end

  if Simulacion==1
       Meca.Simulacion=1;
       Prob=2;

Meca=Resuelve(Nbody,Nrest,NPR,NPP,NPRSD,Lazo,LazoPares,Coord,DirecPP,Radios,D
irecPRSD,VGDL,AGDL,Prob);
  Meca.Prob=Prob;
  Meca.Dibujo=1;
  Meca.Simulacion=Simulacion;
  Meca.Nombre=get(handles.Nombre,'String');
  assignin('base','Meca',Meca);
       PreSimulacion();
  else
      Meca.Simulacion=0;

Meca=Resuelve(Nbody,Nrest,NPR,NPP,NPRSD,Lazo,LazoPares,Coord,DirecPP,Radios,D
irecPRSD,VGDL,AGDL,Prob);
  Meca.Prob=Prob;
  Meca.Dibujo=1;
  Meca.Simulacion=Simulacion;
  Meca.Nombre=get(handles.Nombre,'String');
  assignin('base','Meca',Meca);
MuestraResultados();
  end


function VGDL_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to VGDL (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of VGDL as text
%        str2double(get(hObject,'String')) returns contents of VGDL as a
double


% --- Executes during object creation, after setting all properties.
function VGDL_CreateFcn(hObject, eventdata, handles)
% hObject    handle to VGDL (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function AGDL_Callback(hObject, eventdata, handles)
% hObject    handle to AGDL (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of AGDL as text
%        str2double(get(hObject,'String')) returns contents of AGDL as a
double


% --- Executes during object creation, after setting all properties.
function AGDL_CreateFcn(hObject, eventdata, handles)
% hObject    handle to AGDL (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Lazo_Callback(hObject, eventdata, handles)
% hObject    handle to Lazo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Lazo as text
%        str2double(get(hObject,'String')) returns contents of Lazo as a
double
Nbody=get(handles.Nbody,'String');
Nrest=get(handles.Nrest,'String');
NPR=get(handles.NPR,'String');
```

```
NPP=get(handles.NPP,'String');
NPRSD=get(handles.NPRSD,'String');
Lazo=get(handles.Lazo,'String');
LazoPares=get(handles.LazoPares,'String');

if isempty(Nbody) || isempty(Nrest) || isempty(NPR) || isempty(NPP) ||
isempty(NPRSD) || isempty(Lazo) || isempty(LazoPares)
    set(handles.VGDL,'Enable','off');
    set(handles.AGDL,'Enable','off');
    Frase='Faltan datos del mecanismo';
    set(handles.TextVang,'FontName','MS Sans Serif');
    set(handles.TextAang,'FontName','MS Sans Serif');
    set(handles.TextVang,'String',Frase);
    set(handles.TextAang,'String',Frase);
else
   set(handles.VGDL,'Enable','on');
   set(handles.AGDL,'Enable','on');
   Meca.Nbody=str2double(Nbody);
   Meca.Nrest=str2double(Nrest);
   Meca.NPR=str2double(NPR);
   Meca.NPP=str2double(NPP);
   Meca.NPRSD=str2double(NPRSD);
   Meca.Lazo=str2num(Lazo);
   Meca.LazoPares=str2num(LazoPares);
      RCP=[];
      Meca=BucleAC(Meca);
      for i=[1:1:length(Meca.LazoPares)]
         if Meca.LazoPares(i)==2
             C1=Meca.Lazo(i-1);
             C2=Meca.Lazo(i);
             RCP=[RCP;C1,C2];
         end
      end
      Meca.RCP=RCP;
      Meca=CreaChar(Meca);
    set(handles.TextVang,'FontName','Symbol');
    set(handles.TextAang,'FontName','Symbol');
    set(handles.TextVang,'String',Meca.GDL.CharV.ang);
    set(handles.TextAang,'String',Meca.GDL.CharA.ang);
    set(handles.TextVlin,'FontName','MS Sans Serif');
    set(handles.TextAlin,'FontName','MS Sans Serif');
    set(handles.TextVlin,'String',Meca.GDL.CharV.lin);
    set(handles.TextAlin,'String',Meca.GDL.CharA.lin);
end

% --- Executes during object creation, after setting all properties.
function Lazo_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Lazo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function LazoPares_Callback(hObject, eventdata, handles)
% hObject    handle to LazoPares (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of LazoPares as text
%        str2double(get(hObject,'String')) returns contents of LazoPares as a
double
Nbody=get(handles.Nbody,'String');
Nrest=get(handles.Nrest,'String');
NPR=get(handles.NPR,'String');
NPP=get(handles.NPP,'String');
NPRSD=get(handles.NPRSD,'String');
Lazo=get(handles.Lazo,'String');
LazoPares=get(handles.LazoPares,'String');

if isempty(Nbody) || isempty(Nrest) || isempty(NPR) || isempty(NPP) ||
isempty(NPRSD) || isempty(Lazo) || isempty(LazoPares)
    set(handles.VGDL,'Enable','off');
    set(handles.AGDL,'Enable','off');
    Frase='Faltan datos del mecanismo';
    set(handles.TextVang,'FontName','MS Sans Serif');
    set(handles.TextAang,'FontName','MS Sans Serif');
    set(handles.TextVang,'String',Frase);
    set(handles.TextAang,'String',Frase);
else
   set(handles.VGDL,'Enable','on');
   set(handles.AGDL,'Enable','on');
   Meca.Nbody=str2double(Nbody);
   Meca.Nrest=str2double(Nrest);
   Meca.NPR=str2double(NPR);
   Meca.NPP=str2double(NPP);
   Meca.NPRSD=str2double(NPRSD);
   Meca.Lazo=str2num(Lazo);
   Meca.LazoPares=str2num(LazoPares);
      RCP=[];
      Meca=BucleAC(Meca);
      for i=[1:1:length(Meca.LazoPares)]
         if Meca.LazoPares(i)==2
             C1=Meca.Lazo(i-1);
             C2=Meca.Lazo(i);
             RCP=[RCP;C1,C2];
         end
      end
      Meca.RCP=RCP;
      Meca=CreaChar(Meca);
    set(handles.TextVang,'FontName','Symbol');
    set(handles.TextAang,'FontName','Symbol');
    set(handles.TextVang,'String',Meca.GDL.CharV.ang);
    set(handles.TextAang,'String',Meca.GDL.CharA.ang);
    set(handles.TextVlin,'FontName','MS Sans Serif');
    set(handles.TextAlin,'FontName','MS Sans Serif');
    set(handles.TextVlin,'String',Meca.GDL.CharV.lin);
    set(handles.TextAlin,'String',Meca.GDL.CharA.lin);
end

% --- Executes during object creation, after setting all properties.
function LazoPares_CreateFcn(hObject, eventdata, handles)
% hObject    handle to LazoPares (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Nbody_Callback(hObject, eventdata, handles)
% hObject    handle to Nbody (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Nbody as text
%        str2double(get(hObject,'String')) returns contents of Nbody as a
double
Nbody=get(handles.Nbody,'String');
Nrest=get(handles.Nrest,'String');
NPR=get(handles.NPR,'String');
NPP=get(handles.NPP,'String');
NPRSD=get(handles.NPRSD,'String');
Lazo=get(handles.Lazo,'String');
LazoPares=get(handles.LazoPares,'String');
NPRSD=str2double(NPRSD)
    if NPRSD>=1
       set(handles.Simulacion,'Enable','off');
       set(handles.Simulacion,'Value',0);
    end
    if NPRSD==0
      set(handles.Simulacion,'Enable','on');
    end

if isempty(Nbody) || isempty(Nrest) || isempty(NPR) || isempty(NPP) ||
isempty(NPRSD) || isempty(Lazo) || isempty(LazoPares)
    set(handles.VGDL,'Enable','off');
    set(handles.AGDL,'Enable','off');
    Frase='Faltan datos del mecanismo';
    set(handles.TextVang,'FontName','MS Sans Serif');
    set(handles.TextAang,'FontName','MS Sans Serif');
    set(handles.TextVang,'String',Frase);
    set(handles.TextAang,'String',Frase);
else
   set(handles.VGDL,'Enable','on');
   set(handles.AGDL,'Enable','on');
   Meca.Nbody=str2double(Nbody);
   Meca.Nrest=str2double(Nrest);
   Meca.NPR=str2double(NPR);
   Meca.NPP=str2double(NPP);
   Meca.NPRSD=str2double(NPRSD);
   Meca.Lazo=str2num(Lazo);
   Meca.LazoPares=str2num(LazoPares);
      RCP=[];
       Meca=BucleAC(Meca);
      for i=[1:1:length(Meca.LazoPares)]
         if Meca.LazoPares(i)==2
             C1=Meca.Lazo(i-1);
             C2=Meca.Lazo(i);
             RCP=[RCP;C1,C2];
         end
      end
```

```
      Meca.RCP=RCP;
      Meca=CreaChar(Meca);
    set(handles.TextVang,'FontName','Symbol');
    set(handles.TextAang,'FontName','Symbol');
    set(handles.TextVang,'String',Meca.GDL.CharV.ang);
    set(handles.TextAang,'String',Meca.GDL.CharA.ang);
    set(handles.TextVlin,'FontName','MS Sans Serif');
    set(handles.TextAlin,'FontName','MS Sans Serif');
    set(handles.TextVlin,'String',Meca.GDL.CharV.lin);
    set(handles.TextAlin,'String',Meca.GDL.CharA.lin);
end

if isempty(Nbody) || isempty(Nrest)
    set(handles.NGDL,'Visible','off');
else
    set(handles.NGDL,'Visible','on');
    Meca.Nbody=str2double(Nbody);
    Meca.Nrest=str2double(Nrest);
    NGDL=3*(Meca.Nbody-1)-2*Meca.Nrest;
    set(handles.NGDL,'String',NGDL);
end

% --- Executes during object creation, after setting all properties.
function Nbody_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Nbody (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Nrest_Callback(hObject, eventdata, handles)
% hObject    handle to Nrest (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Nrest as text
%        str2double(get(hObject,'String')) returns contents of Nrest as a
double
Nbody=get(handles.Nbody,'String');
Nrest=get(handles.Nrest,'String');
NPR=get(handles.NPR,'String');
NPP=get(handles.NPP,'String');
NPRSD=get(handles.NPRSD,'String');
Lazo=get(handles.Lazo,'String');
LazoPares=get(handles.LazoPares,'String');
NPRSD=str2double(NPRSD);
    if NPRSD>=1
       set(handles.Simulacion,'Enable','off');
       set(handles.Simulacion,'Value',0);
    end
    if NPRSD==0
      set(handles.Simulacion,'Enable','on');
    end
```

```
if isempty(Nbody) || isempty(Nrest) || isempty(NPR) || isempty(NPP) ||
isempty(NPRSD) || isempty(Lazo) || isempty(LazoPares)
    set(handles.VGDL,'Enable','off');
    set(handles.AGDL,'Enable','off');
    Frase='Faltan datos del mecanismo';
    set(handles.TextVang,'FontName','MS Sans Serif');
    set(handles.TextAang,'FontName','MS Sans Serif');
    set(handles.TextVang,'String',Frase);
    set(handles.TextAang,'String',Frase);
else
   set(handles.VGDL,'Enable','on');
   set(handles.AGDL,'Enable','on');
   Meca.Nbody=str2double(Nbody);
   Meca.Nrest=str2double(Nrest);
   Meca.NPR=str2double(NPR);
   Meca.NPP=str2double(NPP);
   Meca.NPRSD=str2double(NPRSD);
   Meca.Lazo=str2num(Lazo);
   Meca.LazoPares=str2num(LazoPares);
      RCP=[];
       Meca=BucleAC(Meca);
      for i=[1:1:length(Meca.LazoPares)]
         if Meca.LazoPares(i)==2
             C1=Meca.Lazo(i-1);
             C2=Meca.Lazo(i);
             RCP=[RCP;C1,C2];
         end
      end
      Meca.RCP=RCP;
      Meca=CreaChar(Meca);
    set(handles.TextVang,'FontName','Symbol');
    set(handles.TextAang,'FontName','Symbol');
    set(handles.TextVang,'String',Meca.GDL.CharV.ang);
    set(handles.TextAang,'String',Meca.GDL.CharA.ang);
    set(handles.TextVlin,'FontName','MS Sans Serif');
    set(handles.TextAlin,'FontName','MS Sans Serif');
    set(handles.TextVlin,'String',Meca.GDL.CharV.lin);
    set(handles.TextAlin,'String',Meca.GDL.CharA.lin);
end

if isempty(Nbody) || isempty(Nrest)
    set(handles.NGDL,'Visible','off');
else
    set(handles.NGDL,'Visible','on');
    Meca.Nbody=str2double(Nbody);
    Meca.Nrest=str2double(Nrest);
    NGDL=3*(Meca.Nbody-1)-2*Meca.Nrest;
    set(handles.NGDL,'String',NGDL);
end

% --- Executes during object creation, after setting all properties.
function Nrest_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Nrest (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end

% --------------------------------------------------------------------
function Untitled_1_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_2_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function ManualUsuario_Callback(hObject, eventdata, handles)
% hObject    handle to ManualUsuario (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
open('Manual_de_usuario.pdf')


% --------------------------------------------------------------------
function Guardar_Callback(hObject, eventdata, handles)
% hObject    handle to Guardar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Nbody=get(handles.Nbody,'String');
if isempty(Nbody)
    warndlg('Rellene el campo Número de cuerpos','Mensaje');
else
    Nbody=str2double(Nbody);
end
Nrest=get(handles.Nrest,'String');
if isempty(Nrest)
    warndlg('Rellene el campo Número de pares de restricción','Mensaje');
else
    Nrest=str2double(Nrest);
end
NPR=get(handles.NPR,'String');
if isempty(NPR)
    warndlg('Rellene el campo Pares de revolución','Mensaje');
else
    NPR=str2double(NPR);
end
NPP=get(handles.NPP,'String');
if isempty(NPP)
    warndlg('Rellene el campo Pares prismáticos','Mensaje');
else
    NPP=str2double(NPP);
end
NPRSD=get(handles.NPRSD,'String');
if isempty(NPRSD)
    warndlg('Rellene el campo Pares de rodadura sin
deslizamiento','Mensaje');
else
    NPRSD=str2double(NPRSD);
end
DirecPP=get(handles.DirecPP,'Data');
if isempty(DirecPP) && NPP>=1
```

```
    warndlg('Rellene el campo Direcciones de los pares
prismáticos','Mensaje');
else
    if iscell(DirecPP)
       DirecPP=cell2mat(DirecPP);
    end
end
DirecPRSD=get(handles.DirecPRSD,'Data');
if isempty(DirecPRSD) && NPRSD>=1
    warndlg('Rellene el campo Direcciones de los pares de rodadura sin
deslizamiento','Mensaje');
else
    if iscell(DirecPRSD)
       DirecPRSD=cell2mat(DirecPRSD);
    end
end
Radios=get(handles.Radios,'Data');
if isempty(Radios) && NPRSD>=1
    warndlg('Rellene el campo Radios de curvatura','Mensaje');
else
    if iscell(Radios)
       Radios=cell2mat(Radios);
    end
end
Coord=get(handles.Coord,'Data');
if isempty(Coord)
    warndlg('Rellene el campo Coordenadas','Mensaje');
else
    if iscell(Coord)
       Coord=cell2mat(Coord);
    end
end
Lazo=get(handles.Lazo,'String');
if isempty(Lazo)
    warndlg('Rellene el campo Lazo','Mensaje');
else
    Lazo=str2num(Lazo);
end
LazoPares=get(handles.LazoPares,'String');
if isempty(LazoPares)
    warndlg('Rellene el campo Lazo de los pares','Mensaje');
else
    LazoPares=str2num(LazoPares);
end
VGDL=get(handles.VGDL,'String');
if isempty(VGDL)
    warndlg('Rellene el campo Velocidades iniciales','Mensaje');
else
    VGDL=str2num(VGDL)';
end
AGDL=get(handles.AGDL,'String');
if isempty(AGDL)
    warndlg('Rellene el campo Aceleraciones iniciales','Mensaje');
else
    AGDL=str2num(AGDL)';
end
Nombre=get(handles.Nombre,'String');


Save.Nombre=Nombre;
Save.Nbody=Nbody;
```

```
Save.Nrest=Nrest;
Save.NPR=NPR;
Save.NPP=NPP;
Save.NPRSD=NPRSD;
Save.VGDL=VGDL;
Save.AGDL=AGDL;
Save.Lazo=Lazo;
Save.LazoPares=LazoPares;
Save.Coord=Coord;
Save.DirecPP=DirecPP;
Save.DirecPRSD=DirecPRSD;
Save.Radios=Radios;

s=Guarda_Archivos(Save);
if s==0
    msgbox('Error al guardar archivo','Mensaje','error');
end
if s==1
    if isempty(Save.Nombre)
       Bien=imread('Bien.png');
       Frase='Archivo guardado con exito. Nombre:';
       Frase=[Frase ' Mecanismo'];
       msgbox(Frase,'Mensaje','custom',Bien);
    else
       Bien=imread('Bien.png');
       Frase='Archivo guardado con exito. Nombre:';
       Frase=[Frase Save.Nombre];
       msgbox(Frase,'Mensaje','custom',Bien);
    end
end



% --------------------------------------------------------------------
function Cargar_Callback(hObject, eventdata, handles)
% hObject    handle to Cargar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Nombre=get(handles.Nombre,'String');

if isempty(Nombre)
    warndlg('Introduzca el nombre del archivo a abrir en el campo: Nombre del
archivo','Mensaje');
else
    W=what;
    DirectorioA=W.path;
    DirectorioN=[DirectorioA '\Archivos guardados'];
    DirectorioN=[DirectorioN '\'];

    cd(DirectorioN);
    DirectorioF=[DirectorioN Nombre];
    DirectorioF=[DirectorioF '\'];
    E=exist(DirectorioF);
       if E==0
           Frase='No existe ningún archivo guardado con el nombre: ';
           Frase=[Frase Nombre];
           warndlg(Frase,'Mensaje');
           cd(DirectorioA);
       else
           cd(DirectorioA);
           DirectorioF=double(char(DirectorioF));
           Meca=Carga_Archivos(DirectorioF);
```

```
            cd(DirectorioA);
            Nbody=Meca.Nbody;
            set(handles.Nbody,'String',Nbody);
            Nrest=Meca.Nrest;
            set(handles.Nrest,'String',Nrest);
            NPR=Meca.NPR;
            set(handles.NPR,'String',NPR);
            NPP=Meca.NPP;
            set(handles.NPP,'String',NPP);
            NPRSD=Meca.NPRSD;
            set(handles.NPRSD,'String',NPRSD);
            Lazo=Meca.Lazo;
            set(handles.Lazo,'String',Lazo);
            LazoPares=Meca.LazoPares;
            set(handles.LazoPares,'String',LazoPares);
            VGDL=Meca.VGDL;
            set(handles.VGDL,'String',VGDL);
            AGDL=Meca.AGDL;
            set(handles.AGDL,'String',AGDL);
            Coord=Meca.Coord;
            assignin('base','Coord',Coord);
%             Coord=num2str(Coord);
            set(handles.Coord,'Data',Coord);
            DirecPP=Meca.DirecPP;
            assignin('base','DirecPP',Meca.DirecPP);
%             DirecPP=num2str(DirecPP);
            set(handles.DirecPP,'Data',DirecPP);
            DirecPRSD=Meca.DirecPRSD;
            assignin('base','DirecPRSD',Meca.DirecPRSD);
%             DirecPRSD=num2str(DirecPRSD);
            set(handles.DirecPRSD,'data',DirecPRSD);
            Radios=Meca.Radios;
            assignin('base','Radios',Meca.Radios);
%             Radios=num2str(Radios);
            set(handles.Radios,'data',Radios);

           Bien=imread('Bien.png');
           Frase='Archivo cargado con exito';
           msgbox(Frase,'Mensaje','custom',Bien);
           set(handles.VGDL,'Enable','on');
           set(handles.AGDL,'Enable','on');
       end
end

% --- Executes on selection change in Lista.
function Lista_Callback(hObject, eventdata, handles)
% hObject    handle to Lista (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns Lista contents as
cell array
%        contents{get(hObject,'Value')} returns selected item from Lista
Valor=get(handles.Lista,'Value');

if Valor==1
   set(handles.Nbody,'String','1');
   W=what;
   Directorio=W.path;
   Directorio=[Directorio '\'];
   mkdir([Directorio,'Guardar']);
```

```
end

if Valor==2
   set(handles.Nbody,'String','2');
end

% --- Executes during object creation, after setting all properties.
function Lista_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Lista (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --------------------------------------------------------------------
function Guardar_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to Guardar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

Nbody=get(handles.Nbody,'String');
if isempty(Nbody)
    warndlg('Rellene el campo Número de cuerpos','Mensaje');
else
    Nbody=str2double(Nbody);
end
Nrest=get(handles.Nrest,'String');
if isempty(Nrest)
    warndlg('Rellene el campo Número de pares de restricción','Mensaje');
else
    Nrest=str2double(Nrest);
end
NPR=get(handles.NPR,'String');
if isempty(NPR)
    warndlg('Rellene el campo Pares de revolución','Mensaje');
else
    NPR=str2double(NPR);
end
NPP=get(handles.NPP,'String');
if isempty(NPP)
    warndlg('Rellene el campo Pares prismáticos','Mensaje');
else
    NPP=str2double(NPP);
end
NPRSD=get(handles.NPRSD,'String');
if isempty(NPRSD)
    warndlg('Rellene el campo Pares de rodadura sin
deslizamiento','Mensaje');
else
    NPRSD=str2double(NPRSD);
end
DirecPP=get(handles.DirecPP,'Data');
if isempty(DirecPP) && NPP>=1
    warndlg('Rellene el campo Direcciones de los pares
prismáticos','Mensaje');
else
```

```
    if iscell(DirecPP)
       DirecPP=cell2mat(DirecPP);
    end
end
DirecPRSD=get(handles.DirecPRSD,'Data');
if isempty(DirecPRSD) && NPRSD>=1
    warndlg('Rellene el campo Direcciones de los pares de rodadura sin
deslizamiento','Mensaje');
else
    if iscell(DirecPRSD)
       DirecPRSD=cell2mat(DirecPRSD);
    end
end
Radios=get(handles.Radios,'Data');
if isempty(Radios) && NPRSD>=1
    warndlg('Rellene el campo Radios de curvatura','Mensaje');
else
    if iscell(Radios)
       Radios=cell2mat(Radios);
    end
end
Coord=get(handles.Coord,'Data');
if isempty(Coord)
    warndlg('Rellene el campo Coordenadas','Mensaje');
else
    if iscell(Coord)
       Coord=cell2mat(Coord);
    end
end
Lazo=get(handles.Lazo,'String');
if isempty(Lazo)
    warndlg('Rellene el campo Lazo','Mensaje');
else
    Lazo=str2num(Lazo);
end
LazoPares=get(handles.LazoPares,'String');
if isempty(LazoPares)
    warndlg('Rellene el campo Lazo de los pares','Mensaje');
else
    LazoPares=str2num(LazoPares);
end
VGDL=get(handles.VGDL,'String');
if isempty(VGDL)
    warndlg('Rellene el campo Velocidades iniciales','Mensaje');
else
    VGDL=str2num(VGDL)';
end
AGDL=get(handles.AGDL,'String');
if isempty(AGDL)
    warndlg('Rellene el campo Aceleraciones iniciales','Mensaje');
else
    AGDL=str2num(AGDL)';
end
Nombre=get(handles.Nombre,'String');


Save.Nombre=Nombre;
Save.Nbody=Nbody;
Save.Nrest=Nrest;
Save.NPR=NPR;
Save.NPP=NPP;
```

```
Save.NPRSD=NPRSD;
Save.VGDL=VGDL;
Save.AGDL=AGDL;
Save.Lazo=Lazo;
Save.LazoPares=LazoPares;
Save.Coord=Coord;
Save.DirecPP=DirecPP;
Save.DirecPRSD=DirecPRSD;
Save.Radios=Radios;

s=Guarda_Archivos(Save);
if s==0
    msgbox('Error al guardar archivo','Mensaje','error');
end
if s==1
    if isempty(Save.Nombre)
       Bien=imread('Bien.png');
       Frase='Archivo guardado con exito. Nombre:';
       Frase=[Frase ' Mecanismo'];
       msgbox(Frase,'Mensaje','custom',Bien);
    else
       Bien=imread('Bien.png');
       Frase='Archivo guardado con exito. Nombre:';
       Frase=[Frase Save.Nombre];
       msgbox(Frase,'Mensaje','custom',Bien);
    end
end



function Nombre_Callback(hObject, eventdata, handles)
% hObject    handle to Nombre (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Nombre as text
%        str2double(get(hObject,'String')) returns contents of Nombre as a
double


% --- Executes during object creation, after setting all properties.
function Nombre_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Nombre (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in ComprovacionesIniciales.
function ComprovacionesIniciales_Callback(hObject, eventdata, handles)
% hObject    handle to ComprovacionesIniciales (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

Error=0;
```

```
Nbody=get(handles.Nbody,'String');
if isempty(Nbody)
    warndlg('Rellene el campo Número de cuerpos','Mensaje');
    Error=1;
else
    Nbody=str2double(Nbody);
end
Nrest=get(handles.Nrest,'String');
if isempty(Nrest)
    warndlg('Rellene el campo Número de pares de restricción','Mensaje');
    Error=1;
else
    Nrest=str2double(Nrest);
end
NPR=get(handles.NPR,'String');
if isempty(NPR)
    warndlg('Rellene el campo Pares de revolución','Mensaje');
    Error=1;
else
    NPR=str2double(NPR);
end
NPP=get(handles.NPP,'String');
if isempty(NPP)
    warndlg('Rellene el campo Pares prismáticos','Mensaje');
    Error=1;
else
    NPP=str2double(NPP);
end
NPRSD=get(handles.NPRSD,'String');
if isempty(NPRSD)
    warndlg('Rellene el campo Pares de rodadura sin
deslizamiento','Mensaje');
    Error=1;
else
    NPRSD=str2double(NPRSD);
end
Lazo=get(handles.Lazo,'String');
if isempty(Lazo)
    warndlg('Rellene el campo Lazo','Mensaje');
    Error=1;
else
    Lazo=str2num(Lazo);
end
LazoPares=get(handles.LazoPares,'String');
if isempty(LazoPares)
    warndlg('Rellene el campo Lazo de los pares','Mensaje');
    Error=1;
else
    LazoPares=str2num(LazoPares);
end
VGDL=get(handles.VGDL,'String');
if isempty(VGDL)
    warndlg('Rellene el campo Velocidades iniciales','Mensaje');
    Error=1;
else
    VGDL=str2num(VGDL)';
end
AGDL=get(handles.AGDL,'String');
if isempty(AGDL)
    warndlg('Rellene el campo Aceleraciones iniciales','Mensaje');
    Error=1;
else
```

```
    AGDL=str2num(AGDL)';
end

if Error==0
  Meca=CompIni(Nbody,Nrest,NPR,NPP,NPRSD,Lazo,LazoPares,VGDL,AGDL);
  Bien=imread('Bien1.png');
  Mal=imread('Mal1.png');
   if Meca.CompIni.Comp1==1
      axes(handles.Comp1);
      image(Mal);
      axis off
   end
   if Meca.CompIni.Comp1==0
      axes(handles.Comp1);
      image(Bien);
      axis off
   end
   if Meca.CompIni.Comp2==1
      axes(handles.Comp2);
      image(Mal);
      axis off
   end
   if Meca.CompIni.Comp2==0
      axes(handles.Comp2);
      image(Bien);
      axis off
   end
   if Meca.CompIni.Comp3==1
      axes(handles.Comp3);
      image(Mal);
      axis off
   end
   if Meca.CompIni.Comp3==0
      axes(handles.Comp3);
      image(Bien);
      axis off
   end
   if Meca.CompIni.Comp4==1
      axes(handles.Comp4);
      image(Mal);
      axis off
   end
   if Meca.CompIni.Comp4==0
      axes(handles.Comp4);
      image(Bien);
      axis off
   end
end


% --------------------------------------------------------------------


% --- Executes when selected object is changed in uipanel5.
function uipanel5_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in uipanel5
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%   EventName: string 'SelectionChanged' (read only)
%   OldValue: handle of the previously selected object or empty if none was
selected
%   NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)
```

```
% --------------------------------------------------------------------
function Nuevo_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to Nuevo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

close all
Prueba11();
% Nada=imread('Nada.png');
% set(handles.Nombre,'String','');
% set(handles.Nbody,'String','');
% set(handles.Nrest,'String','');
% set(handles.NPR,'String','');
% set(handles.NPP,'String','');
% set(handles.NPRSD,'String','');
% set(handles.Lazo,'String','');
% set(handles.LazoPares,'String','');
% set(handles.VGDL,'String','');
% set(handles.AGDL,'String','');
% set(handles.TextVang,'String','');
% set(handles.TextVlin,'String','');
% set(handles.TextAang,'String','');
% set(handles.TextAlin,'String','');
% set(handles.Coord,'Data','');
% set(handles.DirecPP,'Data','');
% set(handles.DirecPRSD,'Data','');
% set(handles.Radios,'Data','');
%
%       axes(handles.Comp1);
%       image(Nada)
%       axis off
%       axes(handles.Comp2);
%       image(Nada)
%       axis off
%       axes(handles.Comp3);
%       image(Nada)
%       axis off
%       axes(handles.Comp4);
%       image(Nada)
%       axis off


% --------------------------------------------------------------------
function Cargar_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to Cargar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Nombre=get(handles.Nombre,'String');

if isempty(Nombre)
    warndlg('Introduzca el nombre del archivo a abrir en el campo: Nombre del
archivo','Mensaje');
else
    W=what;
    DirectorioA=W.path;
    DirectorioN=[DirectorioA '\Archivos guardados'];
```

```
    DirectorioN=[DirectorioN '\'];

    cd(DirectorioN);
    DirectorioF=[DirectorioN Nombre];
    DirectorioF=[DirectorioF '\'];
    E=exist(DirectorioF);
       if E==0
           Frase='No existe ningún archivo guardado con el nombre: ';
           Frase=[Frase Nombre];
           warndlg(Frase,'Mensaje');
           cd(DirectorioA);
       else
           cd(DirectorioA);
           DirectorioF=double(char(DirectorioF));
           Meca=Carga_Archivos(DirectorioF);
             cd(DirectorioA);
             Nbody=Meca.Nbody;
             set(handles.Nbody,'String',Nbody);
             Nrest=Meca.Nrest;
             set(handles.Nrest,'String',Nrest);
             NPR=Meca.NPR;
             set(handles.NPR,'String',NPR);
             NPP=Meca.NPP;
             set(handles.NPP,'String',NPP);
             NPRSD=Meca.NPRSD;
             set(handles.NPRSD,'String',NPRSD);
             Lazo=Meca.Lazo;
             set(handles.Lazo,'String',Lazo);
             LazoPares=Meca.LazoPares;
             set(handles.LazoPares,'String',LazoPares);
             VGDL=Meca.VGDL;
             set(handles.VGDL,'String',VGDL);
             AGDL=Meca.AGDL;
             set(handles.AGDL,'String',AGDL);
             Coord=Meca.Coord;
             assignin('base','Coord',Coord);
%              Coord=num2str(Coord);
             set(handles.Coord,'Data',Coord);
             DirecPP=Meca.DirecPP;
             assignin('base','DirecPP',Meca.DirecPP);
%              DirecPP=num2str(DirecPP);
             set(handles.DirecPP,'Data',DirecPP);
             DirecPRSD=Meca.DirecPRSD;
             assignin('base','DirecPRSD',Meca.DirecPRSD);
%              DirecPRSD=num2str(DirecPRSD);
             set(handles.DirecPRSD,'data',DirecPRSD);
             Radios=Meca.Radios;
             assignin('base','Radios',Meca.Radios);
%              Radios=num2str(Radios);
             set(handles.Radios,'data',Radios);

           Bien=imread('Bien.png');
           Frase='Archivo cargado con exito';
           msgbox(Frase,'Mensaje','custom',Bien);
           set(handles.VGDL,'Enable','on');
           set(handles.AGDL,'Enable','on');
       end
end

% --------------------------------------------------------------------
```

```
% --- Executes on button press in AyudaRC.
function AyudaRC_Callback(hObject, eventdata, handles)
% hObject    handle to AyudaRC (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of AyudaRC
AyudaRadios();

% --- Executes on button press in AyudaPRSD.
function AyudaPRSD_Callback(hObject, eventdata, handles)
% hObject    handle to AyudaPRSD (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of AyudaPRSD
AyudaDirecPRSD();


% --------------------------------------------------------------------
function Acercade_Callback(hObject, eventdata, handles)
% hObject    handle to Acercade (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
AcercaDe();


% --- Executes when figure1 is resized.
function figure1_ResizeFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function RefComp4_Callback(hObject, eventdata, handles)
% hObject    handle to RefComp4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
helpdlg('Esta comprobación repasa la correcta introducción de los vectores
Velocidades y Aceleraciones iniciales en cuanto a tamaño','Comprobación 1');


% --------------------------------------------------------------------
function RefComp1_Callback(hObject, eventdata, handles)
% hObject    handle to RefComp1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
helpdlg('Esta comprobación repasa la correcta introducción del Lazo de los
Pares y el número de restricciones del mecanismo','Comprobación 1');



% --------------------------------------------------------------------
function RefComp2_Callback(hObject, eventdata, handles)
% hObject    handle to RefComp2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
helpdlg('Esta comprobación repasa si coinciden el número de pares impuestos
con el número total de restricciones del mecanimo','Comprobación 1');


% --------------------------------------------------------------------
function RefComp3_Callback(hObject, eventdata, handles)
% hObject    handle to RefComp3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
helpdlg('Esta comprobación repasa la correcta introducción de los pares con
relación al Lazo de los Pares del mecanismo','Comprobación 1');
```

RELACIONES.M

```
function [Meca] = Relaciones(Meca)
%% Función que crea la subestructura Meca.Rela %%

   % Creación de Meca.Rela().Codig
      for i=[1:1:length(Meca.LazoPares)]
         Meca.Rela(i).Codig=Meca.LazoPares(i);
      end
   %
   % Creación de Meca.Rela().x y Meca.Rela().y
      for i=[1:1:Meca.Nrest]
          Meca.Rela(i).x=Meca.Coord(i,1);
          Meca.Rela(i).y=Meca.Coord(i,2);
      end
   %
   % Creación de Meca.Rela().h
      k=1;
      for i=[1:1:Meca.Nrest]
          if Meca.Rela(i).Codig==2
              h=Meca.DirecPP(k,:);
              hp=[-h(2);h(1)];
              Meca.Rela(i).hp=hp;
              k=k+1;
          end
      end
   %
   % Creación de Meca.Rela().Cuerpos
      for i=[1:1:Meca.Nrest]
          C1=Meca.Lazo(i);
          C2=Meca.Lazo(i+1);
          Meca.Rela(i).Cuerpos=[C1,C2];
      end
   %
   % Implantación de las condiciones del sólido 1
      Meca.Rela(1).Vx=0;
      Meca.Rela(1).Vy=0;
      Meca.Rela(Meca.Nrest).Vx=0;
      Meca.Rela(Meca.Nrest).Vy=0;
   %


end
```

RESUELVE.M

```
function [Meca] =
Resuelve(Nbody,Nrest,NPR,NPP,NPRSD,Lazo,LazoPares,Coord,DirecPP,Radios,DirecP
RSD,VGDL,AGDL,Prob)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Intento de implementación de resolución del problema de velocidad V6 %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Ventana de introducción de datos %%
Dibujo=0;
Simulacion=0;
NGDL=Nbody*3-Nrest*2-3;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Comprobaciones iniciales %%
   % Comprobación 1
      if length(LazoPares)+1~=length(Lazo) || length(Lazo)~=Nrest+1
          fprintf('\n --> >>ERROR<<  Comprobación 1\n')
          Comp1=1;
      else
          Comp1=0;
      end
   %
   % Comprobación 2
      if Nrest~=NPR+NPP+NPRSD
         fprintf('\n --> >>ERROR<<  Comprobación 2\n')
         Comp2=1;
      else
          Comp2=0;
      end
   %
   % Comprobación 3
   npr=0;npp=0;nprsd=0;
      for i=[1:1:length(LazoPares)]
         if LazoPares(i)==1
             npr=npr+1;
         end
         if LazoPares(i)==2
             npp=npp+1;
         end
         if LazoPares(i)==3
             nprsd=nprsd+1;
         end
      end
     if npr~=NPR || npp~=NPP || nprsd~=NPRSD
         fprintf('\n --> >>ERROR<<  Comprobación 3\n')
         Comp3=1;
     else
         Comp3=0;
     end
   %
   % Comprobación 4
      Cont=0;
      for i=[1:1:length(LazoPares)]
          if LazoPares(i)==2
              if i==1
              else
                  Cont=Cont+1;
              end
```

```
            end
        end
        if NPP==1
            if length(VGDL)~=Nbody-1;
                fprintf('\n --> <<ERROR>>  Comprobación 4\n')
                Comp4=1;
            else
                if length(VGDL)~=length(AGDL)
                   fprintf('\n --> <<ERROR>>  Comprobación 4\n')
                   Comp4=1;
                else
                    Comp4=0;
                end
            end
        else
            if length(VGDL)~=Nbody-1+Cont*2
               fprintf('\n --> <<ERROR>>  Comprobación 4\n')
               Comp4=1;
            else
                if length(VGDL)~=length(AGDL)
                    fprintf('\n --> <<ERROR>>  Comprobación 4\n')
                    Comp4=1;
                else
                    Comp4=0;
                end
            end
        end
    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Cálculos %%
    % Implantación de los datos en la matriz Meca
       Meca.Nbody=Nbody;
       Meca.Nrest=Nrest;
       Meca.NPR=NPR;
       Meca.NPP=NPP;
       Meca.NPRSD=NPRSD;
       Meca.NGDL=NGDL;
       Meca.VGDL=VGDL;
       Meca.AGDL=AGDL;
       Meca.Coord=Coord;
       Meca.DirecPP=DirecPP;
       Meca.Radios=Radios;
       Meca.DirecPRSD=DirecPRSD;
       Meca.Lazo=Lazo;
       Meca.LazoPares=LazoPares;
       Meca.CompIni.Comp1=Comp1;
       Meca.CompIni.Comp2=Comp2;
       Meca.CompIni.Comp3=Comp3;
       Meca.CompIni.Comp4=Comp4;
    %
    % Determinación de bucle abierto o cerrado
       Meca=BucleAC(Meca);
    %
    % Creación de la subestructura Meca.Rela
       Meca=Relaciones(Meca);
    %
    % Creación de los grupos para la matriz TH - Meca.Grupo().G
       Meca=CreaGrupos(Meca);
    %
    % Creación de los tramos Meca.Tramo().T
       Meca=CreaTramos(Meca);
    %
```

```
   % Diferenciación del caso de NPP=0 y NPP~=0
      if Meca.NPP==0
          % Caso particular de NPP=0
             Meca=NOPP(Meca);
          %
      else
          % Creación de la matriz TH
             Meca=MatrizTH(Meca);
          %
          % Creación de la matriz HV
             Meca=MatrizHV(Meca);
          %
          % Unión de las matrices TH y HV. Creación de la Matriz Meca.Q
             Q=[Meca.TH,-Meca.HV];
             Meca.Q=Q;
          %
          % Aplicación de la condición de par prismático (Wi=Wj) a Meca.Q
             Meca=CondPrismatico(Meca);
          %
      end
          % Creación de las lineas char para los GDL
              Meca=CreaChar(Meca);
          %
          % Aplicación de los GDL y creación de las matrices Meca.Qind y
Meca.Qdep
             Meca=CondVGDL(Meca);
          %
          % Resolución del problema de velocidad
            Sol=Meca.Qdep\-Meca.Qind;
            Sol=Meca.VGDL+Sol;
            Meca.SolVeloc=Sol;
            if Meca.NPP~=0
               Meca=CondPrisSolV(Meca);
            end
          %
   %

   % Aplicación del Método de las velocidades relativas
      Meca=VelocRela(Meca);
   %

 if Prob==2
   % Resolución del problema de aceleraciones
      Meca=ProbAcel(Meca);
   %

   % Aplicación del Método de las aceleraciones relativas
      Meca=AceleRela(Meca);
   %
 end

%%%%%%%%%%%%%%

%% Comprobaciones finales %%
   % Comprobador del resultado del problema de velocidad
         Meca=CompruebaVel(Meca);
         aux=sum(abs(Meca.CompruebaVel));
         ErrorV=log10(aux);
         if ErrorV==-Inf
             fprintf('\n--> Problema de velocidad correcto\n    con error
0\n')
         else
```

```
            if aux<10^-5
               fprintf('\n--> Problema de velociad correcto\n    con un
errror del 10^%d\n',ErrorV)
            else
               fprintf('\n--> Problema de velocidad\n    recuelto con algun
tipo de error\n')
            end
        end
   %
 if Prob==2
   % Comprobador del resultado del problema de aceleración
       Meca=CompruebaAce(Meca);
         aux=sum(abs(Meca.CompruebaAce));
         ErrorA=log10(aux);
         if ErrorA==-Inf
             fprintf('\n--> Problema de aceleración correcto\n    con error
0\n')
         else
            if aux<10^-5
               fprintf('\n--> Problema de aceleración correcto\n    con un
errror del 10^%d\n',ErrorA)
            else
               fprintf('\n--> Problema de aceleración\n    recuelto con
algun tipo de error\n')
            end
         end
   %
 end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Muestra de resultados %%
if Dibujo==1;
    axis([-15 15 -15 15])
    title('Mecanismo en la posición dada')
    % Muestreo de los pares del mecanismo
       MP=MuestraPares(Meca);
       if MP==1
       else
           fprintf('\n --> >>ERROR<< Muestreo de los pares\n')
       end
    %
    % Muestreo de los cuerpos del mecanismo
       MC=MuestraCuerpos(Meca);
       if MC==1
       else
           fprintf('\n --> >>ERROR<< Muestreo de los cuerpos\n')
       end
    %
    % Muestreo de las velocidades de cada Par
       MV=MuestraVelocidades(Meca);
       if MV==1
       else
           fprintf('\n --> >>ERROR<<  Muestreo de las velocidades\n')
       end
    %
    % Muestreo de las velocidades de cada Par
       MA=MuestraAceleraciones(Meca);
       if MA==1
       else
           fprintf('\n --> >>ERROR<<  Muestreo de las Aceleraciones\n')
```

```
        end
    %

end

%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Simulación %%
if Simulacion==1
    axis([-15 15 -15 15])
    title('Simulación en movimiento del mecanismo')
    Meca.ResTiempo=[];
    Meca.Error=0;
    for i=[1:1:9000]
        Meca.Iteracion=i;
       pause(0.05);
       cla
          % Muestreo de los pares del mecanismo
            MP=MuestraPares(Meca);
            if MP==1
            else
                fprintf('\n --> >>ERROR<< Muestreo de los pares\n')
            end
         %
         % Muestreo de los cuerpos del mecanismo
            MC=MuestraCuerpos(Meca);
            if MC==1
            else
               fprintf('\n --> >>ERROR<< Muestreo de los cuerpos\n')
            end
        %
        % Muestreo de las velocidades de cada Par
           MV=MuestraVelocidades(Meca);
           if MV==1
           else
             fprintf('\n --> >>ERROR<<  Muestreo de las velocidades\n')
           end
        %
        % Muestreo de las Aceleraciones de cada Par
           MA=MuestraAceleraciones(Meca);
             if MA==1
             else
               fprintf('\n --> >>ERROR<<  Muestreo de las Aceleraciones\n')
             end
        %
           Coord=[];
           Meca=NuevasCoord(Meca);
           for j=[1:1:Meca.Nrest]
              Coord=[Coord;Meca.Rela(j).x Meca.Rela(j).y];
           end

           Meca.ResTiempo=[Meca.ResTiempo;i
[Meca.Rela(3).Ax+Meca.Rela(3).Ay]];
           Meca=NuevasGDL(Meca);
           Meca.Coord=Coord;
           Meca=Calculador(Meca);
    end

end
%%%%%%%%%%%%%%%
end
```

TRANSLAZOPARES.M

```
function [W] = TransLazoPares(V)
%% Función que devuelve la transformación de lazo de los pares%%

   for i=[1:1:length(V)]
       W(i)=V(i);
       if V(i)==3
           W(i)=1;
       end
   end

end
```

VELOCRELA.M

```
function [Meca] = VelocRela(Meca)
%% Función que implementa el método de las velocidades relativas %%

   % Creación de Meca.Cuerpo(i).W
     Meca.Cuerpo(1).W=0;
      for i=[1:1:Meca.Nbody-1]
          Meca.Cuerpo(i+1).W=Meca.SolVeloc(i);
      end
   %

   % Implementación de Meca.SolVeloc=V41;V81..... en Meca.Rela().Vx Vy
      VPP=[];
      for i=[1:1:length(Meca.Grupo)]
          fin=Meca.Grupo(i).G(end);
          fin=Meca.Lazo(fin+1);
          if fin==1
          else
              VPP=[VPP,fin];
          end
      end
    if length(VPP)==0
        VPP=0;
    end
    VPP;
%     % Restrinción en caso de bucle abierto [Pruebas]
%        if Meca.Bucle==1
%            VPP(end)=[];
%        end
%     %
    Elim=1;
    k=1;
  if VPP==0
  else
    for i=[Meca.Nbody:1:length(Meca.SolVeloc)]
        if k>length(VPP)
        else
            R=VPP(k);
        if Elim==1
            Meca.Rela(R).Vx=Meca.SolVeloc(i);
            Elim=Elim+1;
        else
            Meca.Rela(R).Vy=Meca.SolVeloc(i);
            Elim=1;
            k=k+1;
        end
        end
    end
  end
   %

% %% Códgo nuevo V2 %%
% Meca.Rela(1).Vx=0;
% Meca.Rela(1).Vy=0;
% for i=[2:1:length(Meca.Rela)]
%     w=Meca.Cuerpo(i).W;
%     T=Meca.Tramo(i-1).T;
%     T=[-T(2);T(1)];
%     v=[Meca.Rela(i-1).Vx;Meca.Rela(i-1).Vy];
```

```
%     Vn=v+T.*w;
%     Meca.Rela(i).Vx=Vn(1);
%     Meca.Rela(i).Vy=Vn(2);
% end
% %% Código nuevo V2 final %%


%%% Código nuevo inicio %%%
MVR=zeros(2,Meca.Nbody);
   % Caso de no existir pares prismáticos NPP=0
      if Meca.NPP==0
          % Calcular matriz MT
            MT=[];
            for i=[1:1:length(Meca.Rela)]
               Rf=Meca.Lazo(i);
               Ri=Meca.Lazo(i+1);
               Posfin=[Meca.Rela(Rf).x;Meca.Rela(Rf).y];
               Posini=[Meca.Rela(Ri).x;Meca.Rela(Ri).y];
               T=Posfin-Posini;
               T=[-T(2);T(1)];
               MT=[MT,T];
            end
            for i=[1:1:length(MT(1,:))-1]
               W=Meca.Cuerpo(i+1).W;
               MT(:,i)=MT(:,i).*W;
            end
            MVR=-MT;
            MVR(:,end)=[];
            MVR=[[0;0],MVR];
          %
          % Igualación de los extremos a cero si es par de revolución
             if Meca.LazoPares(1)==1
                 MVR(:,1)=0;
             end
             if Meca.LazoPares(end)==1
                 MVR(:,end)=0;
             end
          %
          % Proceso de suma
             for i=[2:1:length(MVR(1,:))]
                 MVR(:,i)=MVR(:,i)+MVR(:,i-1);
             end
          %
          % Colocación MVR en Meca.Rela(i).Vx y Meca.Rela(i).Vy
                 MVR(:,1)=0;
                 MVR(:,end)=0;
             for i=[1:1:length(MVR(1,:))]
                 Meca.Rela(i).Vx=MVR(1,i);
                 Meca.Rela(i).Vy=MVR(2,i);
             end
          %
      end
   %
   % Caso de solo un par prismático NPP=1
      if Meca.NPP==1
          % Calcular matriz MT
            MT=[];
            for i=[1:1:length(Meca.Rela)]
               Rf=Meca.Lazo(i);
               Ri=Meca.Lazo(i+1);
               Posfin=[Meca.Rela(Rf).x;Meca.Rela(Rf).y];
```

```
            Posini=[Meca.Rela(Ri).x;Meca.Rela(Ri).y];
            T=Posfin-Posini;
            T=[-T(2);T(1)];
            MT=[MT,T];
         end
         for i=[1:1:length(MT(1,:))-1]
            W=Meca.Cuerpo(i+1).W;
            MT(:,i)=MT(:,i).*W;
         end
         MVR=-MT;
         MVR(:,end)=[];
         MVR=[[0;0],MVR];
      %
      % Igualación de los extremos a cero si es par de revolución
         if Meca.LazoPares(1)==1
             MVR(:,1)=0;
         end
         if Meca.LazoPares(end)==1
             MVR(:,end)=0;
         end
      %
      % Proceso de suma
         NVPP=[];
         for i=[1:1:length(Meca.LazoPares)]
              if Meca.LazoPares(i)==2
                 NVPP=[NVPP,i];
              end
         end
         % Suma-> Caso de NPP en primera posición
            if NVPP==1
                MVR=-MVR;
                for i=[length(MVR(1,:))-1:-1:1]
                    MVR(:,i)=MVR(:,i)+MVR(:,i+1);
                end
            end
         %
         % Suma-> Caso de NPP en última posición
            if NVPP==Meca.Nrest
                for i=[2:1:length(MVR(1,:))]
                    MVR(:,i)=MVR(:,i)+MVR(:,i-1);
                end
            end
         %
         MVR;
         % Suma-> Caso de NPP en cualquier posición
            if NVPP~=1 && NVPP~=Meca.Nrest
                for i=[2:1:NVPP]
                    MVR(:,i)=MVR(:,i)+MVR(:,i-1);
                end
                for i=[length(MVR(1,:))-1:-1:NVPP+1]
                    MVR(:,i)=-(MVR(:,i)+MVR(:,i+1));
                end
            end
         %
      %
      % Colocación MVR en Meca.Rela(i).Vx y Meca.Rela(i).Vy
             MVR(:,1)=0;
         for i=[1:1:length(MVR(1,:))]
             Meca.Rela(i).Vx=MVR(1,i);
             Meca.Rela(i).Vy=MVR(2,i);
         end
      %
end
```

```
%
% Caso de más de un par prismático NPP>=2
   if Meca.NPP>=2
       % Calcular matriz MT
         MT=[];
         for i=[1:1:length(Meca.Rela)]
            Rf=Meca.Lazo(i);
            Ri=Meca.Lazo(i+1);
            Posfin=[Meca.Rela(Rf).x;Meca.Rela(Rf).y];
            Posini=[Meca.Rela(Ri).x;Meca.Rela(Ri).y];
            T=Posfin-Posini;
            T=[-T(2);T(1)];
            MT=[MT,T];
         end
         for i=[1:1:length(MT(1,:))-1]
            W=Meca.Cuerpo(i+1).W;
            MT(:,i)=MT(:,i).*W;
         end
         for i=[1:1:length(VPP)]
            MT(:,VPP(i))=0;
         end
       %
       % Introducir las V calculadas de Meca.SolVeloc
          for i=[1:1:length(VPP)]
             MVR(1,VPP(i))=Meca.Rela(VPP(i)).Vx;
             MVR(2,VPP(i))=Meca.Rela(VPP(i)).Vy;
          end
       %
       MVR=MVR+MT;
       % Igualación de los extremos a cero si es par de revolución
          if Meca.LazoPares(1)==1
              MVR(:,1)=0;
          end
          if Meca.LazoPares(end)==1
              MVR(:,end)=0;
          end
       %
       % Proceso de suma hacia la izquierda
          NVPP=[];
          for i=[1:1:length(Meca.LazoPares)]
               if Meca.LazoPares(i)==2
                  NVPP=[NVPP,i];
               end
          end
          NVPP(end)=[];
          k=0;
          for i=[length(MVR(1,:))-1:-1:1]
              if length(NVPP)-k<=0
                  MVR(:,i)=MVR(:,i)+MVR(:,i+1);
              else
                  if i==NVPP(end-k)
                      k=k+1;
                  else
                      MVR(:,i)=MVR(:,i)+MVR(:,i+1);
                  end
              end
              MVR;
          end
       %
       % Colocación MVR en Meca.Rela(i).Vx y Meca.Rela(i).Vy
          if Meca.LazoPares(1)==1
              MVR(:,1)=0;
```

```
            end
            for i=[1:1:length(MVR(1,:))]
                Meca.Rela(i).Vx=MVR(1,i);
                Meca.Rela(i).Vy=MVR(2,i);
            end
          %
      end
   %
%%% Código nuevo final %%%


%%% Código antiguo inicio %%%
%
%
%    % Cálculo de Vx y Vy en cada Meca.Rela(i)
%        VPP;
%        MVR=zeros(2,Meca.Nrest);
%     if Meca.NPP>=2
%        for i=[1:1:length(VPP)]
%            R=VPP(i);
%            Vx=Meca.Rela(R).Vx;
%            Vy=Meca.Rela(R).Vy;
%            MVR(1,R)=Vx;
%            MVR(2,R)=Vy;
%        end
%        MVR;
%        MT=[];
%        for i=[1:1:length(Meca.Rela)]
%            Rf=Meca.Lazo(i);
%            Ri=Meca.Lazo(i+1);
%            Posfin=[Meca.Rela(Rf).x;Meca.Rela(Rf).y];
%            Posini=[Meca.Rela(Ri).x;Meca.Rela(Ri).y];
%            T=Posfin-Posini;
%            T=[-T(2);T(1)];
%            MT=[MT,T];
%        end
%        MT;
%        for i=[1:1:length(MT(1,:))-1]
%            W=Meca.Cuerpo(i+1).W;
%            MT(:,i)=MT(:,i).*W;
%        end
%        for i=[1:1:length(VPP)]
%            MT(:,VPP(i))=0;
%        end
%        MT;
%        % Copia hacia la izquierda
%        NVPP=[];
%        for i=[1:1:length(Meca.LazoPares)]
%            if Meca.LazoPares(i)==2
%                NVPP=[NVPP,i];
%            end
%        end
%
%           % Restrinción en caso de bucle abierto
%              if Meca.Bucle==1
%                  NVPP(end)=[];
%              end
%           %
%        NVPP;
%            k=0;
%            for i=[length(MVR(1,:))-1:-1:1]
%                if length(NVPP)-k<=0
%                        MVR(:,i)=MVR(:,i+1);
```

```
%                 else
%                     if i==NVPP(end-k)
%                         k=k+1;
%                     else
%                            MVR(:,i)=MVR(:,i+1);
%                     end
%                 end
%                 MVR;
%             end
%              MVR=MVR+MT;
%         %
%     end
%
%        % Caso en el que VPP=0
%         if VPP==0
%          MT=[];
%               for i=[1:1:length(Meca.Rela)]
%                   Rf=Meca.Lazo(i);
%                   Ri=Meca.Lazo(i+1);
%                   Posfin=[Meca.Rela(Rf).x;Meca.Rela(Rf).y];
%                   Posini=[Meca.Rela(Ri).x;Meca.Rela(Ri).y];
%                   T=Posfin-Posini;
%                   T=[-T(2);T(1)];
%                   MT=[MT,T];
%               end
%               for i=[1:1:length(MT(1,:))-1]
%                   W=Meca.Cuerpo(i+1).W;
%                   MT(:,i)=MT(:,i).*W;
%               end
%               MVR=MT;
%         end
%        %
%
% %    if Meca.LazoPares(1)==1
% %        MVR(:,1)=0;
% %    end
% % MVR(:,end)=[];
%    if Meca.LazoPares(end)==1
%        MVR(:,end)=0;
%    end
%
%         for i=[1:1:length(MVR(1,:))]
%             Vx=MVR(1,i);
%             Vy=MVR(2,i);
%             Meca.Rela(i).Vx=Vx;
%             Meca.Rela(i).Vy=Vy;
%         end
%    %

%%% Código antiguo final %%%


end
```