Proyecto Fin de Carrera

Universidad Carlos III de Madrid
Departamento de Ingeniería Mecánica

Manual de referencia de la aplicación "Análisis dinámico de un vehículo automóvil"

Javier Buhigas Pérez
Mayo de 2011

Resumen

El objeto de este documento es la descripción del programa “Análisis dinámico de un vehículo automóvil”.

Este programa permite realizar un análisis dinámico del comportamiento longitudinal (tracción y frenado) y lateral de un vehículo automóvil.

Este programa es propiedad de la Universidad Carlos III de Madrid y su fin es puramente didáctico.

Abstract

The object of this document is to describe the program “Análisis dinámico de un vehículo automóvil”.

This program allows to realize a dynamic analysis of the longitudinal (traction and braking) and lateral vehaviour of the automobile vehicle.

This program is property of the Universidad Carlos III de Madrid and its purpose is purely didactic.

Referencias

Ref. 1. BUHIGAS PÉREZ, Javier. *Manual de usuario de la aplicación "Análisis dinámico de un vehículo automóvil"*. Madrid: Universidad Carlos III de Madrid, 2011.

Ref. 2. BUHIGAS PÉREZ, Javier. *Proyecto Fin de Carrera: Desarrollo de una aplicación en interfaz gráfica de MatLab para la determinación del comportamiento dinámico de un vehículo automóvil*. Madrid: Universidad Carlos III de Madrid,

Ficha técnica

Programa: ANÁLISIS DINÁMICO DE UN VEHÍCULO AUTOMÓVIL.
Propósito: Didáctico. Uso docente en la Universidad Carlos III de Madrid.
Autor: Javier Buhigas Pérez.

Características del programa fuente: Desarrollado con la herramienta GUIDE de MatLab.
Lenguaje de programación: Lenguaje M.
Espacio en disco: 12.8 MB.
Número de módulos: 86 en total. 43 archivos *.m y 43 archivos *.fig.
Modos de ejecución: Interactivo, desde MatLab.
Fecha: Mayo de 2011.
Versión: 1.0

Programación de cada módulo

En este apartado se documenta independientemente cada GUI de la aplicación. Cada GUI está formada por un archivo *.m* que contiene el código del programa y un entorno gráfico *.fig* asociado a dicho código y que permite la interacción con los objetos. Se muestran y comentan por lo tanto sólo las líneas de código necesarias para comprender el funcionamiento de cada objeto y las relaciones entre GUIs.

Las funciones más utilizadas mostradas a continuación están explicadas en Ref. 2. Para conocer más sobre el uso de las funciones que aparecen en este capítulo, consultar la citada referencia.

La nomenclatura que sigue es consistente con el nombre de cada GUI del programa. Nótese que los nombres de cada GUI no incluyen signos de puntuación o espacios (estos se sustituyen por guiones bajos).

Para cada GUI se muestra una captura de pantalla de la misma, una tabla resumen de objetos, el código inicial de la GUI si procede[1], el código correspondiente a los botones de acción (*push buttons* y *edit texts*) y el código relativo a la introducción de las imágenes.

Las tablas de resumen tienen el valor de hacer referencia, principalmente, a los cuadros de texto para entrada y salida de datos. Así, si procede, se incluye una columna que indica la variable asociada a dicho objeto. Como es lógico, sólo los cuadros de texto antes mencionados están asociados a una variable, por lo que en las GUIs que no contengan este tipo de objetos, se omite dicha columna.

Para completar la comprensión de cada GUI, en Ref. 1 se proporciona información extra de los botones y los *edit texts*.

1. INICIO

Tabla 1. Resumen de objetos en "Inicio".

Objeto	Tipo	String	Tag
Comenzar	*Push button*	Comenzar	comenzar

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
```

Botones de acción:

- Comenzar

```
close Inicio
Menu_principal
```

Imágenes:

```
axes2 = imread('logo_uc3m.jpg');
axes(handles.axes2);
axis off;
imshow(axes2);
```

[1] Toda GUI tiene un código inicial que se ejecuta antes de mostrar en pantalla la figura. No se muestra todo este código, sino aquel que se haya incluido con algún objetivo, como la declaración de variables o cálculos entre las mismas.

2. MENU_PRINCIPAL

Tabla 2. Resumen de objetos en "Menu_principal".

Objeto	Tipo	String	Tag
Dinámica longitudinal: Tracción	*Push button*	Dinámica longitudinal: Tracción	traccion
Dinámica longitudinal: Frenado	*Push button*	Dinámica longitudinal: Frenado	frenado
Dinámica lateral	*Push button*	Dinámica lateral	lateral

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
```

Botones de acción:

- Dinámica Longitudinal: Tracción

```
close Menu_principal
Menu_traccion
```

- Dinámica Longitudinal: Frenado

```
close Menu_principal
Menu_frenado
```

- Dinámica Lateral

```
close Menu_principal
Menu_lateral
```

Imágenes:

```
axes_traccion = imread('traccion006.jpg');
axes(handles.axes_traccion);
axis off;
imshow(axes_traccion);

axes_frenado = imread('frenado008.jpg');
axes(handles.axes_frenado);
axis off;
imshow(axes_frenado);

axes_lateral = imread('derrape005.jpg');
axes(handles.axes_lateral);
axis off;
imshow(axes_lateral);
```

3. MENU_TRACCION

Tabla 3. Resumen de objetos en "Menu_traccion".

Objeto	Tipo	String	Tag
Reparto de cargas estáticas	*Push button*	Reparto de cargas estáticas	Reparto_de_cargas_estaticas
Esfuerzo tractor máximo	*Push button*	Esfuerzo tractor máximo	Esfuerzo_tractor_maximo
Rampa máxima	*Push button*	Rampa máxima	Rampa_maxima
Velocidad máxima y resistencias	*Push button*	Velocidad máxima y resistencias	Velocidad_maxima_y_resistencias
Aceleración	*Push button*	Aceleración	Aceleracion
Acuaplaneo	*Push button*	Acuaplaneo	Acuaplaneo
Volver al Menú Principal	*Push button*	Volver al Menú Principal	volver_ppal

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
```

Botones de acción:

- Reparto de cargas estáticas

```
close Menu_traccion
Reparto_de_cargas_estaticas
```

- Esfuerzo tractor máximo

```
close Menu_traccion
Esfuerzo_tractor_maximo
```

- Rampa máxima

```
close Menu_traccion
Rampa_maxima
```

- Velocidad máxima y resistencias

```
close Menu_traccion
Velocidad_maxima_y_resistencias_al_avance
```

- Aceleración

```
close Menu_traccion
Aceleracion
```

- Acuaplaneo

```
close Menu_traccion
Acuaplaneo
```

- Menú Principal

```
close Menu_traccion
Menu_principal
```

Imágenes:

```
axes1 = imread('traccion003.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

4. MENU_FRENADO

Tabla 4. Resumen de objetos en "Menu_frenado".

Objeto	Tipo	String	Tag
Par resistente del motor en una determinada marcha	*Push button*	Par resistente del motor en una determinada marcha	Par_resistente_del_motor
Reparto de cargas en la frenada	*Push button*	Reparto de cargas en la frenada	Reparto_de_cargas_en_la_frenada
Reparto óptimo en la frenada	*Push button*	Reparto óptimo en la frenada	Reparto_optimo_de_la_frenada
Bloqueo de las ruedas	*Push button*	Bloqueo de las ruedas	Bloqueo
Máxima deceleración sufrida antes de bloqueo	*Push button*	Máxima deceleración sufrida antes de bloqueo	Máxima_deceleración_antes_de_bloqueo

Rendimiento del frenado	*Push button*	Rendimiento del frenado	Rendimiento_del_frenado
Distancia y tiempo en una frenada de emergencia	*Push button*	Distancia y tiempo en una frenada de emergencia	Distancia_y_tiempo
Potencia disipada en el frenado	*Push button*	Potencia disipada en el frenado	Potencia_disipada_en_el_frenado
Volver al menú Principal	*Push button*	Volver al Menú Principal	volver_ppal

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
```

Botones de acción:

- Par resistente del motor en una determinada marcha

```
close Menu_frenado
Par_resistente_del_motor
```

- Reparto de cargas en la frenada

```
close Menu_frenado
Reparto_de_cargas_en_la_frenada
```

- Reparto óptimo en la frenada

```
close Menu_frenado
Reparto_optimo_de_la_frenada
```

- Bloqueo de las ruedas

```
close Menu_frenado
Bloqueo
```

- Máxima deceleración sufrida antes de bloqueo

```
close Menu_frenado
Maxima_deceleracion_antes_de_bloqueo
```

- Rendimiento del frenado

```
close Menu_frenado
Rendimiento_del_frenado
```

- Distancia y tiempo en una frenada de emergencia

```
close Menu_frenado
Distancia_y_tiempo
```

- Potencia disipada en el frenado

```
close Menu_frenado
Potencia_disipada_en_el_frenado
```

- Menú Principal

```
close Menu_frenado
Menu_principal
```

Imágenes:

```
axes1 = imread('frenado002.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

5. MENU_LATERAL

Tabla 5. Resumen de objetos en "Menu_lateral".

Objeto	Tipo	String	Tag
Suspensión rígida	Push button	Modelo de suspensión rígida	Suspension_rigida
Suspensión elástica	Push button	Modelo de suspensión elástica	Suspension_elastica
Suspensión de ballestas con juego libre	Push button	Suspensión de ballestas con juego libre	Suspension_ballestas
Indicando los centros de balanceo de suspensión de los neumáticos	Push button	Indicando los centros de balanceo de suspensión de los neumáticos	Centros_balanceo
Transferencia de carga lateral debida al balanceo	Push button	Transferencia de carga lateral debida al balanceo	Tranaferencia_de_carga_lateral

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
```

Botones de acción:

- Modelo de suspensión rígida

```
close Menu_lateral
Suspension_rigida
```

- Modelo de suspensión elástica

```
close Menu_lateral
Suspension_elastica
```

- Suspensión de ballestas con juego libre

```
close Menu_lateral
Suspension_ballestas
```

- Indicando los centros de balanceo de suspensión de los neumáticos

```
close Menu_lateral
Centros_balanceo
```

- Transferencia de carga lateral debida al balanceo

```
close Menu_lateral
Transferencia_de_carga_lateral
```

Imágenes:

```
axes1 = imread('vuelco006.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

6. REPARTO_DE_CARGAS_ESTATICAS

Tabla 6. Resumen de objetos en "Reparto_de_cargas_estaticas".

Objeto	Tipo	String	Tag	Variable asociada
Peso	*Edit text*		peso	Peso
	Static text	Peso	[2]	
Batalla	*Edit text*		batalla	Batalla
	Static text	Batalla		
Fzd	*Edit text*		fzd	Fzd
	Static text	Fzd		
Fzt	*Edit text*		fzt	Fzt
	Static text	Fzt		
Fzd(%)	*Edit text*		fzd_porcentaje	Fzd_porcentaje
	Static text	Fzt(%)		
Fzt(%)	*Edit text*		fzt_porcentaje	Fzt_porcentaje
	Static text	Fzt(%)		
Atrás	*Push button*	Atrás	atrás	
Volver al menú Principal	*Push button*	Volver al Menú Principal	volver_ppal	
Calcular	*Push button*	Calcular	calcular	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
```

Botones de acción:

- Atrás

```
close Reparto_de_cargas_estaticas
Menu_traccion
```

- Menú Principal

[2] Aunque todos los objetos tienen un *tag* asociado, al no ser relevante, se omite indicar el de los textos estáticos como este. En general tienen una etiqueta de la forma "*text***", donde ** es un número entero.

```
close Reparto_de_cargas_estaticas
Menu_Principal
```

- Calcular

```
global Peso
global Batalla
global Fzd
global Fzt
global Fzd_porcentaje
global Fzt_porcentaje
Peso=str2double(get(handles.peso,'String'));
Batalla=str2double(get(handles.batalla,'String'));
Fzd=str2double(get(handles.fzd,'String'));
Fzt=str2double(get(handles.fzt,'String'));
Fzd_porcentaje=str2double(get(handles.fzd_porcentaje,'Stri
ng'));
Fzt_porcentaje=str2double(get(handles.fzt_porcentaje,'Stri
ng'));

if isnan(Peso)
errordlg('El valor del peso debe ser numérico','ERROR')
set(handles.peso,'String',0);
Peso=0;
return
end

if Peso<0
errordlg('El valor del peso debe ser positivo','ERROR')
set(handles.peso,'String',0);
Peso=0;
return
end

if isnan(Batalla)
errordlg('El valor de la batalla debe ser
numérico','ERROR')
set(handles.batalla,'String',0);
Batalla=0;
return
end

if Batalla<0
errordlg('El valor de la batalla debe ser
positivo','ERROR')
set(handles.batalla,'String',0);
Batalla=0;
return
end

if isnan(Fzd)
errordlg('El valor de la fuerza en el eje delantero debe
ser numérico','ERROR')
set(handles.fzd,'String',0);
Fzd=0;
return
end
```

```
if Fzd<0
errordlg('El valor de la fuerza en el eje delantero debe
ser positivo','ERROR')
set(handles.fzd,'String',0);
Fzd=0;
return
end

if isnan(Fzt)
errordlg('El valor de la fuerza en el eje trasero debe ser
numérico','ERROR')
set(handles.fzt,'String',0);
Fzt=0;
return
end

if Fzt<0
errordlg('El valor de la fuerza en el eje trasero debe ser
positivo','ERROR')
set(handles.fzt,'String',0);
Fzt=0;
return
end

if isnan(Fzd_porcentaje)
errordlg('El valor del reparto en el eje delantero debe ser
numérico','ERROR')
set(handles.fzd_porcentaje,'String',0);
Fzd_porcentaje=0;
return
end

if Fzd_porcentaje<0
errordlg('El valor del reparto en el eje delantero debe ser
positivo','ERROR')
set(handles.fzd_porcentaje,'String',0);
Fzd_porcentaje=0;
return
end

if isnan(Fzt_porcentaje)
errordlg('El valor del reparto en el eje trasero debe ser
numérico','ERROR')
set(handles.fzt_porcentaje,'String',0);
Fzt_porcentaje=0;
return
end

if Fzt_porcentaje<0
errordlg('El valor del reparto en el eje trasero debe ser
positivo','ERROR')
set(handles.fzt_porcentaje,'String',0);
Fzt_porcentaje=0;
return
end

close Reparto_de_cargas_estaticas
```

```
Reparto_de_cargas_estaticas_resultados
```

Imágenes:

```
axes1 = imread('reparto.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

7. REPARTO_DE_CARGAS_ESTATICAS_RESULTADOS

Tabla 7. Resumen de objetos en "Reparto_de_cargas_estaticas_resultados".

Objeto	Tipo	String	Tag	Variable asociada
l1	*Edit text*		l1	L1
	Static text	l1		
l2	*Edit text*		l2	L2
	Static text	l2		
Atrás	*Push button*	Atrás	atras	
Volver al menú Principal	*Push button*	Volver al menú Principal	volver_ppal	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

global Peso
global Batalla
global Fzd
global Fzt
global Fzd_porcentaje
global Fzt_porcentaje
P=Peso*9.81;
if Fzd_porcentaje==0
    L1=Fzt*Batalla/P;
    L2=Fzd*Batalla/P;
elseif Fzd==0
    L1=0.01*Fzt_porcentaje*Batalla;
    L2=0.01*Fzd_porcentaje*Batalla;
end
set(handles.l1,'String',L1);
set(handles.l2,'String',L2);
```

Botones de acción:

- Atrás

```
close Reparto_de_cargas_estaticas_resultados
Reparto_de_cargas_estaticas
```

- Menú Principal

```
close Reparto_de_cargas_estaticas_resultados
Menu_Principal
```

Imágenes:

```
axes2 = imread('reparto001.jpg');
axes(handles.axes2);
axis off;
imshow(axes2);

axes1 = imread('reparto.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

8. ESFUERZO_TRACTOR_MAXIMO

Tabla 8. Resumen de objetos en "Esfuerzo_tractor_maximo".

Objeto	Tipo	String	Tag	Variable asociada
Peso	*Edit text*		peso	Peso
	Static text	Peso		
Coeficiente	*Edit text*		coeficiente	Coeficiente
	Static text	Coeficiente		
Adherencia	*Edit text*		adherencia	Adherencia
	Static text	Adherencia		
Batalla	*Edit text*		batalla	Batalla
	Static text	Batalla		
l1	*Edit text*		l1	L1
	Static text	l1		
l2	*Edit text*		l2	L2
	Static text	l2		
h	*Edit text*		h	H
	Static text	h		
Relación de transmisión	*Edit text*		relacion	Relacion
	Static text	Relación transmisión		
Par máximo	*Edit text*		par	Par
	Static text	Par máximo		

Rendimiento del motor	*Edit text*		rend_motor	Rend_motor
	Static text	Rendimiento motor		
Rendimiento de la transmisión	*Edit text*		rend_transmision	Rend_transmision
	Static text	Rendimiento transmisión		
Radio bajo carga	*Edit text*		radio	Radio
	Static text	Radio bajo carga		
Tipo de tracción	*Pop-up Menu*	Delantera Trasera Total	traccion_popupmenu	Tipo_traccion
	Static text	Tipo de tracción		
Atrás	*Push button*	Atrás	atras	
Volver al menú Principal	*Push button*	Volver al menú Principal	volver_ppal	
Calcular	*Push button*	Calcular	calcular	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
```

Botones de acción:

- Tipo de tracción

```
Aux=get(handles.traccion_popupmenu,'Value');
switch Aux
    case 1
    Tipo_traccion=1;
    case 2
    Tipo_traccion=2;
    case 3
    Tipo_traccion=3;
end
```

- Atrás

```
close Esfuerzo_tractor_maximo
Menu_traccion
```

- Menú Principal

```
close Esfuerzo_tractor_maximo
Menu_principal
```

- Calcular

```
global Peso
global Coeficiente
global Adherencia
global Batalla
global L1
global L2
global H
global Relacion
global Par
global Rend_motor
global Rend_transmision
global Radio
global Tipo_traccion
Peso=str2double(get(handles.peso,'String'));
Coeficiente=str2double(get(handles.coeficiente,'String'));
Adherencia=str2double(get(handles.adherencia,'String'));
Batalla=str2double(get(handles.batalla,'String'));
L1=str2double(get(handles.l1,'String'));
L2=str2double(get(handles.l2,'String'));
H=str2double(get(handles.h,'String'));
Relacion=str2double(get(handles.relacion,'String'));
Par=str2double(get(handles.par,'String'));
Rend_motor=str2double(get(handles.rend_motor,'String'));
Rend_transmision=str2double(get(handles.rend_transmision,'
String'));
Radio=str2double(get(handles.radio,'String'));

if isnan(Peso)
errordlg('El valor del peso debe ser numérico','ERROR')
set(handles.peso,'String',0);
Peso=0;
return
end

if Peso<0
errordlg('El valor del peso debe ser positivo','ERROR')
set(handles.peso,'String',0);
Peso=0;
return
end

if isnan(Coeficiente)
errordlg('El valor del coeficiente de resistencia a la
rodadura debe ser numérico','ERROR')
set(handles.coeficiente,'String',0);
Coeficiente=0;
return
end

if Coeficiente<0
errordlg('El valor del coeficiente de resistencia a la
rodadura debe ser positivo','ERROR')
set(handles.coeficiente,'String',0);
```

```
Coeficiente=0;
return
end

if isnan(Adherencia)
errordlg('El valor de la adherencia debe ser
numérico','ERROR')
set(handles.adherencia,'String',0);
Adherencia=0;
return
end

if Adherencia<0
errordlg('El valor de la adherencia debe ser
positivo','ERROR')
set(handles.adherencia,'String',0);
Adherencia=0;
return
end

if isnan(Batalla)
errordlg('El valor de la batalla debe ser
numérico','ERROR')
set(handles.batalla,'String',0);
Batalla=0;
return
end

if Batalla<0
errordlg('El valor de la batalla debe ser
positivo','ERROR')
set(handles.batalla,'String',0);
Batalla=0;
return
end

if isnan(L1)
errordlg('El valor de L1 debe ser numérico','ERROR')
set(handles.l1,'String',0);
L1=0;
return
end

if L1<=0
errordlg('El valor de L1 debe ser mayor que 0','ERROR')
set(handles.l1,'String',0);
L1=0;
return
end

if isnan(L2)
errordlg('El valor de L2 debe ser numérico','ERROR')
set(handles.l2,'String',0);
L2=0;
return
end

if L2<=0
```

```
errordlg('El valor de L2 debe ser mayor que 0','ERROR')
set(handles.l2,'String',0);
L2=0;
return
end

if isnan(H)
errordlg('El valor de la altura debe ser numérico','ERROR')
set(handles.h,'String',0);
H=0;
return
end

if H<=0
errordlg('El valor de la altura debe ser mayor que
0','ERROR')
set(handles.h,'String',0);
H=0;
return
end

if isnan(Relacion)
errordlg('El valor de la relación de transmisión debe ser
numérico','ERROR')
set(handles.relacion,'String',0);
Relacion=0;
return
end

if Relacion<=0
errordlg('El valor de la relación de transmisión debe ser
mayor que 0','ERROR')
set(handles.relacion,'String',0);
Relacion=0;
return
end

if isnan(Par)
errordlg('El valor del par debe ser numérico','ERROR')
set(handles.par,'String',0);
Par=0;
return
end

if Par<=0
errordlg('El valor del par debe ser mayor que 0','ERROR')
set(handles.par,'String',0);
Par=0;
return
end

if isnan(Rend_motor)
errordlg('El valor del rendimiento del motor debe ser
numérico','ERROR')
set(handles.rend_motor,'String',0);
Rend_motor=0;
return
end
```

```
if Rend_motor<=0
errordlg('El valor del rendimiento del motor debe ser mayor
que 0','ERROR')
set(handles.rend_motor,'String',0);
Rend_motor=0;
return
end

if isnan(Rend_transmision)
errordlg('El valor del rendimiento de la transmisión debe
ser numérico','ERROR')
set(handles.rend_transmision,'String',0);
Rend_transmision=0;
return
end

if Rend_transmision<=0
errordlg('El valor del rendimiento de la transmisión debe
ser mayor que 0','ERROR')
set(handles.rend_transmision,'String',0);
Rend_transmision=0;
return
end

if isnan(Radio)
errordlg('El valor del radio bajo carga debe ser
numérico','ERROR')
set(handles.radio,'String',0);
Radio=0;
return
end

if Radio<=0
errordlg('El valor del radio bajo carga debe ser mayor que
0','ERROR')
set(handles.radio,'String',0);
Radio=0;
return
end

close Esfuerzo_tractor_maximo
Esfuerzo_tractor_maximo_resultados
```

9. ESFUERZO_TRACTOR_MAXIMO_RESULTADOS

Tabla 9. Resumen de objetos en "Esfuerzo_tractor_maximo_resultados".

Objeto	Tipo	String	Tag	Variable asociada
Ft	*Edit text*		ft	L1
	Static text	Ft		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

global Peso
global Coeficiente
global Adherencia
global Batalla
global L1
global L2
global H
global Relacion
global Par
global Rend_motor
global Rend_transmision
global Radio
global Tipo_traccion

Peso=Peso*9.81;
Rend_motor=Rend_motor*0.01;
Rend_transmision=Rend_transmision*0.01;

Fmotriz=Relacion*Par*Rend_motor*Rend_transmision/Radio;

if Tipo_traccion==1
Fadh=Adherencia*Peso*(L2+H*Coeficiente)/(Batalla+Adherencia
*H);
else if Tipo_traccion==2
Fadh=Adherencia*Peso*(L1-H*Coeficiente)/(Batalla-
Adherencia*H);
    else
    Fadh=Adherencia*Peso;
    end
end

Fmotriz
```

```
Fadh
Ft=min(Fmotriz,Fadh);
set(handles.ft,'String',Ft);
```

Botones de acción:

- Atrás

```
close Esfuerzo_tractor_maximo_resultados
Esfuerzo_tractor_maximo
```

- Menú Principal

```
close Esfuerzo_tractor_maximo_resultados
Menu_principal
```

Imágenes:

```
axes1 = imread('traccion008.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);

axes2 = imread('traccion007.jpg');
axes(handles.axes2);
axis off;
imshow(axes2);
```

10. RAMPA_MAXIMA

Tabla 10. Resumen de objetos en "Rampa_maxima".

Objeto	Tipo	String	Tag	Variable asociada
Ft	*Edit text*		fuerza	Fuerza
	Static text	Ft		
P	*Edit text*		peso	Peso
	Static text	P		
fr	*Edit text*		coeficiente	Coeficiente
	Static text	fr		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	
Calcular	*Push button*	Calcular	Calcular	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
```

```
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
```

Botones de acción:

- Atrás

```
close Rampa_maxima
Menu_traccion
```

- Menú Principal

```
close Rampa_maxima
Menu_principal
```

- Calcular

```
global fuerza
global peso
global coeficiente
fuerza=str2double(get(handles.fuerza,'String'));
peso=str2double(get(handles.peso,'String'));
coeficiente=str2double(get(handles.coeficiente,'String'));
close Rampa_maxima
Rampa_maxima_resultados

if isnan(Peso)
errordlg('El valor del peso debe ser numérico','ERROR')
set(handles.peso,'String',0);
Peso=0;
return
end

if Peso<0
errordlg('El valor del peso debe ser positivo','ERROR')
set(handles.peso,'String',0);
Peso=0;
return
end

if isnan(Fuerza)
errordlg('El valor del esfuerzo tractor debe ser
numérico','ERROR')
set(handles.fuerza,'String',0);
Fuerza=0;
return
end

if Fuerza<0
errordlg('El valor del esfuerzo tractor debe ser
positivo','ERROR')
set(handles.fuerza,'String',0);
Fuerza=0;
return
```

```
end

if isnan(Coeficiente)
errordlg('El valor del coeficiente de resistencia a la
rodadura debe ser numérico','ERROR')
set(handles.coeficiente,'String',0);
Coeficiente=0;
return
end

if Coeficiente<0
errordlg('El valor del coeficiente de resistencia a la
rodadura debe ser positivo','ERROR')
set(handles.coeficiente,'String',0);
Coeficiente=0;
return
end

close Rampa_maxima
Rampa_maxima_resultados
```

Imágenes:

```
axes1 = imread('rampa001.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

11. RAMPA_MAXIMA_RESULTADOS

Tabla 11. Resumen de objetos en "Rampa_maxima_resultados".

Objeto	Tipo	String	Tag	Variable asociada
Rampa máxima	*Edit text*		rampa	Rampa
	Static text	Rampa máxima		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

scrsz = get(0, 'ScreenSize');
```

```
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

global Fuerza
global Peso
global Coeficiente

P=Peso*9.81;
Rampa=100*tan(asin((Fuerza-P*Coeficiente)/P));
set(handles.rampa,'String',Rampa);
```

Botones de acción:

- Atrás

```
close Rampa_maxima_resultados
Rampa_maxima
```

- Menú Principal

```
close Rampa_maxima_resultados
Menu_Principal
```

Imágenes:

```
axes1 = imread('rampa002.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);

axes2 = imread('rampa003.jpg');
axes(handles.axes2);
axis off;
imshow(axes2);
```

12. VELOCIDAD_MAXIMA_Y_RESISTENCIAS_AL_AVANCE

Tabla 12. Resumen de objetos en "Velocidad_maxima_y_resistencias_al_avance"

Objeto	Tipo	String	Tag	Variable asociada
Peso	*Edit text*		peso	Peso
	Static text	Peso		
Coeficiente	*Edit text*		coeficiente	Coeficiente
	Static text	Coeficiente		
Cx	*Edit text*		cx	Cx
	Static text	Cx		
Área frontal	*Edit text*		af	Af
	Static text	Área frontal		
Rampa	*Edit text*		rampa	Rampa
	Static text	Rampa		
Potencia máxima	*Edit text*		potencia	Potencia
	Static text	Potencia máxima		
Revoluciones máximas	*Edit text*		revoluciones	Revoluciones
	Static text	Revoluciones máximas		
Relación de transmisión	*Edit text*		relacion	Relacion
	Static text	Relación transmisión		
Deslizamiento	*Edit text*		i	I
	Static text	Deslizamiento		
Rendimiento del motor	*Edit text*		rend_motor	Rend_motor
	Static text	Rendimiento del motor		
Rendimiento de la transmisión	*Edit text*		rend_transmision	Rend_transmision
	Static text	Rend. de la transmisión		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	
Calcular	*Push button*	Calcular	Calcular	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
```

Botones de acción:

- Atrás

```
close Velocidad_maxima_y_resistencias_al_avance
Menu_traccion
```

- Menú Principal

```
close Velocidad_maxima_y_resistencias_al_avance
Menu_principal
```

- Calcular

```
global Peso
global Coeficiente
global Cx
global Af
global Rampa
global Potencia
global Revoluciones
global Relacion
global I
global Rend_motor
global Rend_transmision
Peso=str2double(get(handles.peso,'String'));
Coeficiente=str2double(get(handles.coeficiente,'String'));
Cx=str2double(get(handles.cx,'String'));
Af=str2double(get(handles.af,'String'));
Rampa=str2double(get(handles.rampa,'String'));
Potencia=str2double(get(handles.potencia,'String'));
Revoluciones=str2double(get(handles.revoluciones,'String'))
;
Relacion=str2double(get(handles.relacion,'String'));
I=str2double(get(handles.i,'String'));
Rend_motor=str2double(get(handles.rend_motor,'String'));
Rend_transmision=str2double(get(handles.rend_transmision,'S
tring'));

if isnan(Peso)
errordlg('El valor del peso debe ser numérico','ERROR')
set(handles.peso,'String',0);
Peso=0;
return
end
```

```
if Peso<0
errordlg('El valor del peso debe ser positivo','ERROR')
set(handles.peso,'String',0);
Peso=0;
return
end

if isnan(Coeficiente)
errordlg('El valor del coeficiente de resistencia a la
rodadura debe ser numérico','ERROR')
set(handles.coeficiente,'String',0);
Coeficiente=0;
return
end

if Coeficiente<0
errordlg('El valor del coeficiente de resistencia a la
rodadura debe ser positivo','ERROR')
set(handles.coeficiente,'String',0);
Coeficiente=0;
return
end

if isnan(Cx)
errordlg('El valor del coeficiente aerodinámico debe ser
numérico','ERROR')
set(handles.cx,'String',0);
Cx=0;
return
end

if Cx<0
errordlg('El valor del coeficiente aerodinámico debe ser
positivo','ERROR')
set(handles.cx,'String',0);
Cx=0;
return
end

if isnan(Af)
errordlg('El valor del área frontal debe ser
numérico','ERROR')
set(handles.af,'String',0);
Af=0;
return
end

if Af<0
errordlg('El valor del área frontal debe ser
positivo','ERROR')
set(handles.af,'String',0);
Af=0;
return
end

if isnan(Rampa)
errordlg('El valor de la rampa debe ser numérico','ERROR')
set(handles.rampa,'String',0);
```

```
Rampa=0;
return
end

if Rampa<0
errordlg('El valor de la rampa debe ser positivo','ERROR')
set(handles.rampa,'String',0);
Rampa=0;
return
end

if isnan(Potencia)
errordlg('El valor de la potencia debe ser
numérico','ERROR')
set(handles.potencia,'String',0);
Potencia=0;
return
end

if Potencia<0
errordlg('El valor de la potencia debe ser
positivo','ERROR')
set(handles.potencia,'String',0);
Potencia=0;
return
end

if isnan(Revoluciones)
errordlg('El valor de las revoluciones debe ser
numérico','ERROR')
set(handles.revoluciones,'String',0);
Revoluciones=0;
return
end

if Revoluciones<0
errordlg('El valor de las revoluciones debe ser
positivo','ERROR')
set(handles.revoluciones,'String',0);
Revoluciones=0;
return
end

if isnan(Relacion)
errordlg('El valor de la relación de transmisión debe ser
numérico','ERROR')
set(handles.relacion,'String',0);
Relacion=0;
return
end

if Relacion<0
errordlg('El valor de la relación de transmisión debe ser
positivo','ERROR')
set(handles.relacion,'String',0);
Relacion=0;
return
end
```

```
if isnan(I)
errordlg('El valor del deslizamiento debe ser
numérico','ERROR')
set(handles.i,'String',0);
I=0;
return
end

if I<0
errordlg('El valor del deslizamiento debe ser
positivo','ERROR')
set(handles.i,'String',0);
I=0;
return
end

if isnan(Rend_motor)
errordlg('El valor del rendimiento del motor debe ser
numérico','ERROR')
set(handles.rend_motor,'String',0);
Rend_motor=0;
return
end

if Rend_motor<0
errordlg('El valor del rendimiento del motor debe ser
positivo','ERROR')
set(handles.rend_motor,'String',0);
Rend_motor=0;
return
end

if Rend_motor>100
errordlg('El valor del rendimiento del motor debe ser menor
que 100%','ERROR')
set(handles.rend_motor,'String',0);
Rend_motor=100;
return
end

if isnan(Rend_transmision)
errordlg('El valor del rendimiento de la transmisión debe
ser numérico','ERROR')
set(handles.rend_transmision,'String',0);
Rend_transmision=0;
return
end

if Rend_transmision<0
errordlg('El valor del rendimiento de la transmision debe
ser positivo','ERROR')
set(handles.rend_transmision,'String',0);
Rend_transmision=0;
return
end

if Rend_transmision>100
```

```
errordlg('El valor del rendimiento de la transmision debe
ser menor que 100%','ERROR')
set(handles.rend_transmision,'String',0);
Rend_transmision=100;
return
end

close Velocidad_maxima_y_resistencias_al_avance
Velocidad_maxima_y_resistencias_al_avance_resultados
```

Imágenes:

```
axes1 = imread('velocidad003.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

13. VELOCIDAD_MAXIMA_Y_RESISTENCIAS_AL_AVANCE_RESULTADOS

Tabla 13. Resumen de objetos en "Velocidad_maxima_y_resistencias_al_avance_resultados"

Objeto	Tipo	String	Tag	Variable asociada
Resistencia aerodinámica	*Edit text*		ra	Ra
	Static text	Resistencia aerodinámica		
Resistencia a la rodadura	*Edit text*		rr	Rr
	Static text	Resistencia a la rodadura		
Resistencia gravitatoria	*Edit text*		rg	Rg
	Static text	Resistencia gravitatoria		
Velocidad máxima	*Edit text*		vmax	Vmax
	Static text	Velocidad máxima		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
```

```
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

global Peso
global Coeficiente
global Cx
global Af
global Rampa
global Potencia
global Revoluciones
global Relacion
global I
global Rend_motor
global Rend_transmision

Rend_transmision=Rend_transmision*0.01;
Rend_motor=Rend_motor*0.01;
P=9.81*Peso;
Pot=Potencia*735.5*Rend_motor*Rend_transmision;
Vm2=pi*Revoluciones*(1-I)/(30*Relacion);
A=0.5*1.225*Cx*Af;
B=0;
C=(Coeficiente+0.01*Rampa)*P;
D=-Pot;
E=[A B C D];
F=roots(E);
if isreal(F(1))==1
    Vm1=F(1);
else if isreal(F(2))==1
        Vm1=F(2);
    else
        Vm1=F(3);
    end
end
V=min(Vm1,Vm2);
Vmax=3.6*V;
Ra=A*V^2;
Rr=Coeficiente*P;
Rg=0.01*Rampa*P;

set(handles.vmax,'String',Vmax);
set(handles.ra,'String',Ra);
set(handles.rr,'String',Rr);
set(handles.rg,'String',Rg);
```

Botones de acción:

- Atrás

```
close Velocidad_maxima_y_resistencias_al_avance_resultados
Velocidad_maxima_y_resistencias_al_avance
```

- Menú Principal

```
close Velocidad_maxima_y_resistencias_al_avance_resultados
Menu_principal
```

Imágenes:

```
axes_v = imread('velocidad001.jpg');
axes(handles.axes_v);
axis off;
imshow(axes_v);

axes_ra = imread('resistencias003.jpg');
axes(handles.axes_ra);
axis off;
imshow(axes_ra);

axes_rr = imread('resistencias008.jpg');
axes(handles.axes_rr);
axis off;
imshow(axes_rr);

axes_rg = imread('resistencias006.jpg');
axes(handles.axes_rg);
axis off;
imshow(axes_rg);
```

14. ACELERACION

Tabla 14. Resumen de objetos en "Aceleración"

Objeto	Tipo	String	Tag	Variable asociada
Peso	*Edit text*		peso	Peso
	Static text	Peso		
Ft	*Edit text*		ft	Ft
	Static text	Ft		
Resistencia	*Edit text*		resistencia	Resistencia
	Static text	Resistencia		
Masas rotativas	*Edit text*		gamma	Gamma
	Static text	Masas rotativas		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	
Calcular	*Push button*	Calcular	Calcular	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
```

Botones de acción:

- Atrás

```
close Aceleracion
Menu_traccion
```

- Menú Principal

```
close Aceleracion
Menu_principal
```

- Calcular

```
global Peso
global Ft
global Resistencia
global Gamma
Peso=str2double(get(handles.peso,'String'));
Ft=str2double(get(handles.ft,'String'));
Resistencia=str2double(get(handles.resistencia,'String'));
Gamma=str2double(get(handles.gamma,'String'));

if isnan(Peso)
errordlg('El valor del peso debe ser numérico','ERROR')
set(handles.peso,'String',0);
Peso=0;
return
end

if Peso<0
errordlg('El valor del peso debe ser positivo','ERROR')
set(handles.peso,'String',0);
Peso=0;
return
end

if isnan(Ft)
errordlg('El valor del esfuerzo tractor debe ser
numérico','ERROR')
set(handles.ft,'String',0);
Ft=0;
return
end

if Ft<0
errordlg('El valor del esfuerzo tractor debe ser
positivo','ERROR')
set(handles.ft,'String',0);
Ft=0;
return
end

if isnan(Resistencia)
errordlg('El valor de la resistencia debe ser
numérico','ERROR')
```

```
set(handles.resistencia,'String',0);
Resistencia=0;
return
end

if Resistencia<0
errordlg('El valor de la resistencia debe ser
positivo','ERROR')
set(handles.resistencia,'String',0);
Resistencia=0;
return
end

if isnan(Gamma)
errordlg('El valor del coeficiente de masas rotatorias debe
ser numérico','ERROR')
set(handles.gamma,'String',0);
Gamma=0;
return
end

if Gamma<=0
errordlg('El valor del coeficiente de masas rotatorias debe
ser mayor que 0','ERROR')
set(handles.gamma,'String',0);
Gamma=0;
return
end

if Ft<=Resistencia
errordlg('El esfuerzo tractor debe ser mayor que las
resistencias al avance','ERROR')
return
end

close Aceleracion
Aceleracion_resultados
```

Imágenes:

```
axes1 = imread('aceleracion001.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

15. ACELERACION_RESULTADOS

Tabla 15. Resumen de objetos en "Aceleración_resultados"

Objeto	Tipo	String	Tag	Variable asociada
Aceleración	*Edit text*		aceleración	Aceleracion
	Static text	Aceleración		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

global Peso
global Ft
global Resistencia
global Gamma
Aceleracion=(Ft-Resistencia)/(Gamma*Peso);
set(handles.aceleracion,'String',Aceleracion);
```

Botones de acción:

- Atrás

```
close Aceleracion_resultados
Aceleracion
```

- Menú Principal

```
close Aceleracion_resultados
Menu_principal
```

Imágenes:

```
axes1 = imread('aceleracion002.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);

axes2 = imread('aceleracion003.jpg');
axes(handles.axes2);
axis off;
imshow(axes2);
```

16. ACUAPLANEO

Tabla 16. Resumen de objetos en "Acuaplaneo"

Objeto	Tipo	String	Tag	Variable asociada
Presión	*Edit text*		presion	Presion
	Static text	Presión		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	
Calcular	*Push button*	Calcular	Calcular	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
```

Botones de acción:

- Atrás

```
close Acuaplaneo
Menu_traccion
```

- Menú Principal

```
close Acuaplaneo
Menu_Principal
```

- Calcular

```
global Presion
Presion=str2double(get(handles.presion,'String'));

if isnan(Presion)
errordlg('El valor de la presión debe ser
numérico','ERROR')
set(handles.presion,'String',0);
Presion=0;
return
end

if Presion<0
errordlg('El valor de la presion debe ser
positivo','ERROR')
set(handles.presion,'String',0);
```

```
Presion=0;
return
end

close Acuaplaneo
Acuaplaneo_resultados
```

Imágenes:

```
axes1 = imread('aquaplaning003.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

17. ACUAPLANEO_RESULTADOS

Tabla 17. Resumen de objetos en "Acuaplaneo_resultados".

Objeto	Tipo	String	Tag	Variable asociada
Velocidad de transición	*Edit text*		va1	Va1
	Static text	Velocidad de transición		
Velocidad de hidroplaneo	*Edit text*		va2	Va2
	Static text	Velocidad de hidroplaneo		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

global presion
P=100*presion;
VA1=3.94*realsqrt(P);
VA2=6.34*realsqrt(P);
set(handles.va1,'String',VA1);
set(handles.va2,'String',VA2);
```

Botones de acción:

- Atrás

```
close Acuaplaneo_resultados
Acuaplaneo
```

- Menú Principal

```
close Acuaplaneo_resultados
Menu_Principal
```

Imágenes:

```
axes1 = imread('aquaplaning001.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

18. PAR_RESISTENTE_DEL_MOTOR

Tabla 18. Resumen de objetos en "Par_resistente_del_motor".

Objeto	Tipo	String	Tag	Variable asociada
Mc	*Edit text*		par	Par
	Static text	Mc		
Relación	*Edit text*		relacion	Relacion
	Static text	Relación		
Rendimiento	*Edit text*		rendimiento	Rendimiento
	Static text	Rendimiento		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	
Calcular	*Push button*	Calcular	Calcular	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
```

Botones de acción:

- Atrás

```
close Par_resistente_del_motor
```

```
Menu_frenado
```

- Menú Principal

```
close Par_resistente_del_motor
Menu_Principal
```

- Calcular

```
global Par
global Relacion
global Rendimiento
Par=str2double(get(handles.par,'String'));
Relacion=str2double(get(handles.relacion,'String'));
Rendimiento=str2double(get(handles.rendimiento,'String'));

if isnan(Relacion)
errordlg('El valor de la relación de transmisión debe ser
numérico','ERROR')
set(handles.relacion,'String',0);
Relacion=0;
return
end

if Relacion<=0
errordlg('El valor de la relación de transmisión debe ser
mayor que 0','ERROR')
set(handles.relacion,'String',0);
Relacion=0;
return
end

if isnan(Par)
errordlg('El valor del par debe ser numérico','ERROR')
set(handles.par,'String',0);
Par=0;
return
end

if Par<=0
errordlg('El valor del par debe ser mayor que 0','ERROR')
set(handles.par,'String',0);
Par=0;
return
end

if isnan(Rendimiento)
errordlg('El valor del rendimiento debe ser
numérico','ERROR')
set(handles.rendimiento,'String',0);
Rendimiento=0;
return
end

if Rendimiento<=0
```

```
errordlg('El valor del rendimiento debe ser mayor que
0','ERROR')
set(handles.rendimiento,'String',0);
Rendimientor=0;
return
end

close Par_resistente_del_motor
Par_resistente_del_motor_resultados
```

Imágenes:

```
axes1 = imread('par_resistente001.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);

axes2 = imread('par_resistente004.jpg');
axes(handles.axes2);
axis off;
imshow(axes2);
```

19. PAR_RESISTENTE_DEL_MOTOR_RESULTADOS

Tabla 19. Resumen de objetos en "Par_resistente_del_motor_resultados".

Objeto	Tipo	String	Tag	Variable asociada
Par resistente del motor	*Edit text*		par_resultado	Res
	Static text	Par resistente del motor		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	menu_ppal	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

global Par
global Relacion
global Rendimiento
Res=Par*Relacion/Rendimiento;
set(handles.par_resultado,'String',Res);
```

Botones de acción:

- Atrás

```
close Par_resistente_del_motor_resultados
Par_resistente_del_motor
```

- Menú Principal

```
close Par_resistente_del_motor_resultados
Menu_Principal
```

Imágenes:

```
axes1 = imread('motor002.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);

axes2 = imread('par_resistente002.jpg');
axes(handles.axes2);
axis off;
imshow(axes2);
```

20. REPARTO_DE_CARGAS_EN_LA_FRENADA

Tabla 20. Resumen de objetos en "Reparto_de_cargas_en_la_frenada".

Objeto	Tipo	String	Tag	Variable asociada
Peso	*Edit text*		peso	Peso
	Static text	Peso		
Coeficiente	*Edit text*		coeficiente	Coeficiente
	Static text	Coeficiente		
Batalla	*Edit text*		batalla	Batalla
	Static text	Batalla		
Adherencia	*Edit text*		adherencia	Adherencia
	Static text	Adherencia		
l1	*Edit text*		l1	L1
	Static text	l1		
l2	*Edit text*		l2	L2
	Static text	l2		
h	*Edit text*		h	H
	Static text	h		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	menu_ppal	
Calcular	*Push button*	Calcular	Calcular	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
```

Botones de acción:

- Atrás

```
close Reparto_de_cargas_en_la_frenada
Menu_frenado
```

- Menú Principal

```
close Reparto_de_cargas_en_la_frenada
Menu_Principal
```

- Calcular

```
global Peso
global Coeficiente
global Batalla
global Adherencia
global L1
global L2
global H
Peso=str2double(get(handles.peso,'String'));
Coeficiente=str2double(get(handles.coeficiente,'String'));
Batalla=str2double(get(handles.batalla,'String'));
Adherencia=str2double(get(handles.adherencia,'String'));
L1=str2double(get(handles.l1,'String'));
L2=str2double(get(handles.l2,'String'));
H=str2double(get(handles.h,'String'));

if isnan(Peso)
errordlg('El valor del peso debe ser numérico','ERROR')
set(handles.peso,'String',0);
Peso=0;
return
end

if Peso<0
errordlg('El valor del peso debe ser positivo','ERROR')
set(handles.peso,'String',0);
Peso=0;
return
end

if isnan(Coeficiente)
errordlg('El valor del coeficiente de resistencia a la
rodadura debe ser numérico','ERROR')
```

```
set(handles.coeficiente,'String',0);
Coeficiente=0;
return
end

if Coeficiente<0
errordlg('El valor del coeficiente de resistencia a la
rodadura debe ser positivo','ERROR')
set(handles.coeficiente,'String',0);
Coeficiente=0;
return
end

if isnan(Batalla)
errordlg('El valor de la batalla debe ser
numérico','ERROR')
set(handles.batalla,'String',0);
Batalla=0;
return
end

if Batalla<0
errordlg('El valor de la batalla debe ser
positivo','ERROR')
set(handles.batalla,'String',0);
Batalla=0;
return
end

if isnan(Adherencia)
errordlg('El valor de la adherencia debe ser
numérico','ERROR')
set(handles.adherencia,'String',0);
Adherencia=0;
return
end

if Adherencia<0
errordlg('El valor de la adherencia debe ser
positivo','ERROR')
set(handles.adherencia,'String',0);
Adherencia=0;
return
end

if isnan(L1)
errordlg('El valor de L1 debe ser numérico','ERROR')
set(handles.l1,'String',0);
L1=0;
return
end

if L1<0
errordlg('El valor de L1 debe ser positivo','ERROR')
set(handles.l1,'String',0);
L1=0;
return
end
```

```
if isnan(L2)
errordlg('El valor de L2 debe ser numérico','ERROR')
set(handles.l2,'String',0);
L2=0;
return
end

if L2<0
errordlg('El valor de L2 debe ser positivo','ERROR')
set(handles.l2,'String',0);
L2=0;
return
end

if isnan(H)
errordlg('El valor de la altura debe ser numérico','ERROR')
set(handles.h,'String',0);
H=0;
return
end

if H<0
errordlg('El valor de la altura debe ser positivo','ERROR')
set(handles.h,'String',0);
H=0;
return
end

close Reparto_de_cargas_en_la_frenada
Reparto_de_cargas_en_la_frenada_resultados
```

Imágenes:

```
axes1 = imread('frenado006.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

21. REPARTO_DE_CARGAS_EN_LA_FRENADA_RESULTADOS

Tabla 21. Resumen de objetos en "Reparto_de_cargas_en_la_frenada_resultados".

Objeto	Tipo	String	Tag	Variable asociada
Carga en el eje delantero	*Edit text*		fuerza_del	Fd
	Edit text		porcentaje_del	Del
	Static text	Carga en el eje delantero		
Carga en el eje trasero	*Edit text*		fuerza_tra	Ft
	Edit text		porcentaje_tra	Tra
	Static text	Carga en el eje trasero		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

global Peso
global Coeficiente
global Batalla
global Adherencia
global L1
global L2
global H

P=Peso/9.81;
Fd=(P*L2+H*P*(Adherencia+Coeficiente))/Batalla;
Ft=(P*L1-H*P*(Adherencia+Coeficiente))/Batalla;
Del=100*Fd/(Fd+Ft);
Tra=100*Ft/(Fd+Ft);
set(handles.fuerza_del,'String',Fd);
set(handles.fuerza_tra,'String',Ft);
set(handles.porcentaje_del,'String',Del);
set(handles.porcentaje_tra,'String',Tra);
```

Botones de acción:

- Atrás

```
close Reparto_de_cargas_en_la_frenada_resultados
Reparto_de_cargas_en_la_frenada
```

- Menú Principal

```
close Reparto_de_cargas_en_la_frenada_resultados
Menu_Principal
```

Imágenes:

```
axes1 = imread('repartofrenada001.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

22. REPARTO_OPTIMO_DE_LA_FRENADA

Tabla 22. Resumen de objetos en "Reparto_optimo_de_la_frenada".

Objeto	Tipo	String	Tag	Variable asociada
Coeficiente	*Edit text*		coeficiente	Coeficiente
	Static text	Coeficiente		
Adherencia	*Edit text*		adherencia	Adherencia
	Static text	Adherencia		
l1	*Edit text*		l1	L1
	Static text	l1		
l2	*Edit text*		l2	L2
	Static text	l2		
h	*Edit text*		h	H
	Static text	h		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	
Calcular	*Push button*	Calcular	Calcular	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
```

Botones de acción:

- Atrás

```
close Reparto_optimo_de_la_frenada
Menu_frenado
```

- Menú Principal

```
close Reparto_optimo_de_la_frenada
Menu_principal
```

- Calcular

```
global Coeficiente
global Adherencia
global L1
global L2
global H
Coeficiente=str2double(get(handles.coeficiente,'String'));
Adherencia=str2double(get(handles.adherencia,'String'));
L1=str2double(get(handles.l1,'String'));
L2=str2double(get(handles.l2,'String'));
H=str2double(get(handles.h,'String'));

if isnan(Coeficiente)
errordlg('El valor del coeficiente de resistencia a la
rodadura debe ser numérico','ERROR')
set(handles.coeficiente,'String',0);
Coeficiente=0;
return
end

if Coeficiente<0
errordlg('El valor del coeficiente de resistencia a la
rodadura debe ser positivo','ERROR')
set(handles.coeficiente,'String',0);
Coeficiente=0;
return
end


if isnan(Adherencia)
errordlg('El valor de la adherencia debe ser
numérico','ERROR')
set(handles.adherencia,'String',0);
Adherencia=0;
return
end

if Adherencia<0
errordlg('El valor de la adherencia debe ser
positivo','ERROR')
set(handles.adherencia,'String',0);
Adherencia=0;
return
end

if isnan(L1)
errordlg('El valor de L1 debe ser numérico','ERROR')
```

```
set(handles.l1,'String',0);
L1=0;
return
end

if L1<0
errordlg('El valor de L1 debe ser positivo','ERROR')
set(handles.l1,'String',0);
L1=0;
return
end

if isnan(L2)
errordlg('El valor de L2 debe ser numérico','ERROR')
set(handles.l2,'String',0);
L2=0;
return
end

if L2<0
errordlg('El valor de L2 debe ser positivo','ERROR')
set(handles.l2,'String',0);
L2=0;
return
end

if isnan(H)
errordlg('El valor de la altura debe ser numérico','ERROR')
set(handles.h,'String',0);
H=0;
return
end

if H<0
errordlg('El valor de la altura debe ser positivo','ERROR')
set(handles.h,'String',0);
H=0;
return
end

close Reparto_optimo_de_la_frenada
Reparto_optimo_de_la_frenada_resultados
```

Imágenes:

```
axes1 = imread('frenado001.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

23. REPARTO_OPTIMO_DE_LA_FRENADA_RESULTADOS

Tabla 23. Resumen de objetos en "Reparto_optimo_de_la_frenada_resultados".

Objeto	Tipo	String	Tag	Variable asociada
En las ruedas delanteras	*Edit text*		res_del	Kdel
	Static text	En las ruedas delanteras		
En las ruedas delanteras	*Edit text*		res_tra	Ktra
	Static text	En las ruedas delanteras		
Atrás	Push button	Atrás	atras	
Menú Principal	Push button	Menú Principal	Menu_ppal	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

global Coeficiente
global Adherencia
global L1
global L2
global H
C=(L2+H*(Coeficiente+Adherencia))/(L1-
H*(Coeficiente+Adherencia));
Ktra=100/(1+C);
Kdel=100-Ktra;
set(handles.res_del,'String',Kdel);
set(handles.res_tra,'String',Ktra);
```

Botones de acción:

- Atrás

```
close Reparto_optimo_de_la_frenada_resultados
Reparto_optimo_de_la_frenada
```

- Menú Principal

```
close Reparto_optimo_de_la_frenada_resultados
Menu_principal
```

Imágenes:

```
axes1 = imread('frenado017.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

24. BLOQUEO

Tabla 24. Resumen de objetos en "Bloqueo".

Objeto	Tipo	String	Tag	Variable asociada
Fuerza de frenado impuesta por el limitador de presión	*Edit text*		flimd	Flimd
	Static text	Flim,d		
	Edit text		flimt	Flimt
	Static text	Flim,t		
Reparto de cargas en la frenada	*Edit text*		fzd	Fzd
	Static text	Fzd		
	Edit text		fzt	Fzt
	Static text	Fzt		
Adherencia	*Edit text*		adherencia	Adherencia
	Static text	Adherencia		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	
Calcular	*Push button*	Calcular	Calcular	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
```

Botones de acción:

- Atrás

```
close Bloqueo
Menu_frenado
```

- Menú Principal

```
close Bloqueo
```

```
Menu_principal
```

- Calcular

```
global Flimd
global Flimt
global Fzd
global Fzt
global Adherencia
Flimd=str2double(get(handles.flimd,'String'));
Flimt=str2double(get(handles.flimt,'String'));
Fzd=str2double(get(handles.fzd,'String'));
Fzt=str2double(get(handles.fzt,'String'));
Adherencia=str2double(get(handles.adherencia,'String'));

if isnan(Flimd)
errordlg('El valor de la fuerza límite en el eje delantero
debe ser numérico','ERROR')
set(handles.flimd,'String',0);
Flimd=0;
return
end

if Flimd<0
errordlg('El valor de la fuerza límite en el eje delantero
debe ser positivo','ERROR')
set(handles.flimd,'String',0);
Flimd=0;
return
end

if isnan(Flimt)
errordlg('El valor de la fuerza límite en el eje trasero
debe ser numérico','ERROR')
set(handles.flimt,'String',0);
Flimt=0;
return
end

if Flimt<0
errordlg('El valor de la fuerza límite en el eje trasero
debe ser positivo','ERROR')
set(handles.flimt,'String',0);
Flimt=0;
return
end

if isnan(Fzd)
errordlg('El valor de la fuerza en el eje delantero debe
ser numérico','ERROR')
set(handles.fzd,'String',0);
Fzd=0;
return
end

if Fzd<0
```

```
errordlg('El valor de la fuerza en el eje delantero debe
ser positivo','ERROR')
set(handles.fzd,'String',0);
Fzd=0;
return
end

if isnan(Fzt)
errordlg('El valor de la fuerza en el eje trasero debe ser
numérico','ERROR')
set(handles.fzt,'String',0);
Fzt=0;
return
end

if Fzt<0
errordlg('El valor de la fuerza en el eje trasero debe ser
positivo','ERROR')
set(handles.fzt,'String',0);
Fzt=0;
return
end


if isnan(Adherencia)
errordlg('El valor de la adherencia debe ser
numérico','ERROR')
set(handles.adherencia,'String',0);
Adherencia=0;
return
end

if Adherencia<0
errordlg('El valor de la adherencia debe ser
positivo','ERROR')
set(handles.adherencia,'String',0);
Adherencia=0;
return
end

if Flimd<Flimt
errordlg('El valor de la fuerza a la que actúa el limitador
delantero debe ser mayor que el del trasero','ERROR')
return
end

close Bloqueo
Bloqueo_resultados
```

Imágenes:

```
axes1 = imread('bloqueo001.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

25. BLOQUEO_RESULTADOS

Tabla 25. Resumen de objetos en "Bloqueo_resultados"

Objeto	Tipo	String	Tag	Variable asociada
	Static text	Bloquean las ruedas delanteras No bloquean las ruedas delanteras	delantero	
	Static text	Bloquean las ruedas traseras No bloquean las ruedas traseras	trasero	
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

global Flimd
global Flimt
global Fzd
global Fzt
global Adherencia
Fadhd=Adherencia*Fzd;
Fadht=Adherencia*Fzt;
if Fadhd<Flimd
    set(handles.delantero,'String','Bloquean las ruedas
delanteras');
elseif Fadhd>=Flimd
    set(handles.delantero,'String','No bloquean las ruedas
delanteras');
end

if Fadht<Flimt
    set(handles.trasero,'String','Bloquean las ruedas
traseras');
elseif Fadht>=Flimt
    set(handles.trasero,'String','No bloquean las ruedas
traseras');
end
```

Botones de acción:

- Atrás

```
close Bloqueo_resultados
Bloqueo
```

- Menú Principal

```
close Bloqueo_resultados
Menu_principal
```

Imágenes:

```
axes1 = imread('bloqueo002.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);

axes2 = imread('frenado011.jpg');
axes(handles.axes2);
axis off;
imshow(axes2);
```

26. MAXIMA_DECELERACION_ANTES_DE_BLOQUEO

Tabla 26. Resumen de objetos en "Maxima_deceleracion_antes_de_bloqueo"

Objeto	Tipo	String	Tag	Variable asociada
Coeficiente	*Edit text*		coeficiente	Coeficiente
	Static text	Coeficiente		
Adherencia	*Edit text*		adherencia	Adherencia
	Static text	Adherencia		
Batalla	*Edit text*		batalla	Batalla
	Static text	Batalla		
l1	*Edit text*		l1	L1
	Static text	l1		
l2	*Edit text*		l2	L2
	Static text	l2		
h	*Edit text*		h	H
	Static text	h		
Kfd	*Edit text*		kfd	Kfd
	Static text	Kfd		
Kft	*Edit text*		kft	Kft
	Static text	Kft		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	
Calcular	*Push button*	Calcular	Calcular	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
```

Botones de acción:

- Atrás

```
close Maxima_deceleracion_antes_de_bloqueo
Menu_frenado
```

- Menú Principal

```
close Maxima_deceleracion_antes_de_bloqueo
Menu_principal
```

- Calcular

```
global Coeficiente
global Adherencia
global Batalla
global L1
global L2
global H
global Kfd
global Kft
Coeficiente=str2double(get(handles.coeficiente,'String'));
Adherencia=str2double(get(handles.adherencia,'String'));
Batalla=str2double(get(handles.batalla,'String'));
L1=str2double(get(handles.l1,'String'));
L2=str2double(get(handles.l2,'String'));
H=str2double(get(handles.h,'String'));
Kfd=str2double(get(handles.kfd,'String'));
Kft=str2double(get(handles.kft,'String'));

if isnan(Coeficiente)
errordlg('El valor del coeficiente de resistencia a la
rodadura debe ser numérico','ERROR')
set(handles.coeficiente,'String',0);
Coeficiente=0;
return
end

if Coeficiente<0
errordlg('El valor del coeficiente de resistencia a la
rodadura debe ser positivo','ERROR')
set(handles.coeficiente,'String',0);
Coeficiente=0;
return
end
```

```
if isnan(Adherencia)
errordlg('El valor de la adherencia debe ser
numérico','ERROR')
set(handles.adherencia,'String',0);
Adherencia=0;
return
end

if Adherencia<0
errordlg('El valor de la adherencia debe ser
positivo','ERROR')
set(handles.adherencia,'String',0);
Adherencia=0;
return
end

if isnan(Batalla)
errordlg('El valor de la batalla debe ser
numérico','ERROR')
set(handles.batalla,'String',0);
Batalla=0;
return
end

if Batalla<0
errordlg('El valor de la batalla debe ser
positivo','ERROR')
set(handles.batalla,'String',0);
Batalla=0;
return
end

if isnan(L1)
errordlg('El valor de L1 debe ser numérico','ERROR')
set(handles.l1,'String',0);
L1=0;
return
end

if L1<0
errordlg('El valor de L1 debe ser positivo','ERROR')
set(handles.l1,'String',0);
L1=0;
return
end

if isnan(L2)
errordlg('El valor de L2 debe ser numérico','ERROR')
set(handles.l2,'String',0);
L2=0;
return
end

if L2<0
errordlg('El valor de L2 debe ser positivo','ERROR')
set(handles.l2,'String',0);
L2=0;
```

```
return
end

if isnan(H)
errordlg('El valor de la altura debe ser numérico','ERROR')
set(handles.h,'String',0);
H=0;
return
end

if H<0
errordlg('El valor de la altura debe ser positivo','ERROR')
set(handles.h,'String',0);
H=0;
return
end

if isnan(Kfd)
errordlg('El valor del reparto en el eje delantero debe ser
numérico','ERROR')
set(handles.kfd,'String',0);
Kfd=0;
return
end

if Kfd<0
errordlg('El valor del reparto en el eje delantero debe ser
positivo','ERROR')
set(handles.Kzd,'String',0);
Kzd=0;
return
end

if isnan(Kft)
errordlg('El valor del reparto en el eje trasero debe ser
numérico','ERROR')
set(handles.Kft,'String',0);
Kft=0;
return
end

if Kft<0
errordlg('El valor del reparto en el eje trasero debe ser
positivo','ERROR')
set(handles.kft,'String',0);
Kft=0;
return
end

close Maxima_deceleracion_antes_de_bloqueo
Maxima_deceleracion_antes_de_bloqueo_resultados
```

Imágenes:

```
axes1 = imread('frenado007.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

27. MAXIMA_DECELERACION_ANTES_DE_BLOQUEO_RESULTADOS

Tabla 27. Resumen de objetos en "Maxima_deceleracion_antes_de_bloqueo_resultados"

Objeto	Tipo	String	Tag	Variable asociada
En las ruedas delanteras	*Edit text*		aceld	Aceld
	Static text	En las ruedas delanteras		
En las ruedas delanteras	*Edit text*		acelt	Acelt
	Static text	En las ruedas delanteras		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

global Coeficiente
global Adherencia
global Batalla
global L1
global L2
global H
global Kfd
global Kft
Kfd_uni=0.01*Kfd;
Kft_uni=0.01*Kft;
Aceld=(Adherencia*L2/Batalla+Coeficiente*Kfd_uni)/(Kfd_uni-
Adherencia*H/Batalla);
Acelt=(Adherencia*L1/Batalla+Coeficiente*Kft_uni)/(Kfd_uni+
Adherencia*H/Batalla);
set(handles.aceld,'String',Aceld);
set(handles.acelt,'String',Acelt);
```

Botones de acción:

- Atrás

```
close Maxima_deceleracion_antes_de_bloqueo_resultados
Maxima_deceleracion_antes_de_bloqueo
```

- Menú Principal

```
close Maxima_deceleracion_antes_de_bloqueo_resultados
Menu_principal
```

Imágenes:

```
axes1 = imread('frenado003.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

28. RENDIMIENTO_DEL_FRENADO

Tabla 28. Resumen de objetos en "Rendimiento_del_frenado".

Objeto	Tipo	String	Tag	Variable asociada
Adherencia	*Edit text*		adherencia	Adherencia
	Static text	Adherencia		
Aceleración máxima	*Edit text*		amax	Amax
	Static text	a maxima		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	
Calcular	*Push button*	Calcular	Calcular	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
```

Botones de acción:

- Atrás

```
close Rendimiento_del_frenado
Menu_frenado
```

- Menú Principal

```
close Rendimiento_del_frenado
Menu_principal
```

- Calcular

```
global Adherencia
global Amax
Adherencia=str2double(get(handles.adherencia,'String'));
Amax=str2double(get(handles.amax,'String'));

if isnan(Adherencia)
errordlg('El valor de la adherencia debe ser
numérico','ERROR')
set(handles.adherencia,'String',0);
Adherencia=0;
return
end

if Adherencia<0
errordlg('El valor de la adherencia debe ser
positivo','ERROR')
set(handles.adherencia,'String',0);
Adherencia=0;
return
end

if isnan(Amax)
errordlg('El valor de la aceleración debe ser
numérico','ERROR')
set(handles.amax,'String',0);
Amaxn=0;
return
end

if Amax<0
errordlg('El valor de la aceleración debe ser
positivo','ERROR')
set(handles.amax,'String',0);
Amax=0;
return
end

if Amax>0.3
errordlg('Tenga en cuenta el confort de los pasajeros o el
desplazamiento de la carga','ERROR')
set(handles.amax,'String',0);
Amax=0;
return
end

close Rendimiento_del_frenado
Rendimiento_del_frenado_resultados
```

Imágenes:

```
axes1 = imread('frenado010.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);

axes2 = imread('frenado012.jpg');
axes(handles.axes2);
axis off;
imshow(axes2);
```

29. RENDIMIENTO_DEL_FRENADO_RESULTADOS

Tabla 29. Resumen de objetos en "Rendimiento_del_frenado_resultados".

Objeto	Tipo	String	Tag	Variable asociada
Rendimiento	*Edit text*		rendimiento	Rendimiento
	Static text	Rendimiento		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

global Adherencia
global Amax
Rendimiento=100*Amax/(Adherencia);
set(handles.rendimiento,'String',Rendimiento);
```

Botones de acción:

- Atrás

```
close Rendimiento_del_frenado_resultados
Rendimiento_del_frenado
```

- Menú Principal

```
close Rendimiento_del_frenado_resultados
Menu_principal
```

Imágenes:

```
axes1 = imread('frenado015.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);

axes2 = imread('frenado013.jpg');
axes(handles.axes2);
axis off;
imshow(axes2);
```

30. DISTANCIA_Y_TIEMPO

Tabla 30. Resumen de objetos en "Distancia_y_tiempo".

Objeto	Tipo	String	Tag	Variable asociada
Peso	*Edit text*		peso	Peso
	Static text	Peso		
Coeficiente	*Edit text*		coeficiente	Coeficiente
	Static text	Coeficiente		
Adherencia	*Edit text*		adherencia	Adherencia
	Static text	Adherencia		
Cx	*Edit text*		cx	Cx
	Static text	Cx		
Área frontal	*Edit text*		af	Af
	Static text	Área frontal		
Velocidad	*Edit text*		v	V
	Static text	Velocidad		
Rampa	*Edit text*		rampa	Rampa
	Static text	Rampa		
Rendimiento del frenado	*Edit text*		rendimiento	Rendimiento
	Static text	Rendimiento frenado		
Coeficiente de masas rotativas	*Edit text*		gamma	Gamma
	Static text	Masas rotativas		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	
Calcular	*Push button*	Calcular	Calcular	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
```

Botones de acción:

- Atrás

```
close Distancia_y_tiempo
Menu_frenado
```

- Menú Principal

```
close Distancia_y_tiempo
Menu_principal
```

- Calcular

```
global Peso
global Coeficiente
global Adherencia
global Cx
global Af
global V
global Rampa
global Rendimiento
global Gamma
Peso=str2double(get(handles.peso,'String'));
Coeficiente=str2double(get(handles.coeficiente,'String'));
Adherencia=str2double(get(handles.adherencia,'String'));
Cx=str2double(get(handles.cx,'String'));
Af=str2double(get(handles.af,'String'));
V=str2double(get(handles.v,'String'));
Rampa=str2double(get(handles.rampa,'String'));
Rendimiento=str2double(get(handles.rendimiento,'String'));
Gamma=str2double(get(handles.gamma,'String'));

if isnan(Peso)
errordlg('El valor del peso debe ser numérico','ERROR')
set(handles.peso,'String',0);
Peso=0;
return
end

if Peso<0
errordlg('El valor del peso debe ser positivo','ERROR')
set(handles.peso,'String',0);
Peso=0;
return
end

if isnan(Coeficiente)
errordlg('El valor del coeficiente de resistencia a la
rodadura debe ser numérico','ERROR')
set(handles.coeficiente,'String',0);
Coeficiente=0;
return
end

if Coeficiente<0
errordlg('El valor del coeficiente de resistencia a la
rodadura debe ser positivo','ERROR')
set(handles.coeficiente,'String',0);
Coeficiente=0;
return
end

if isnan(Adherencia)
errordlg('El valor de la adherencia debe ser
numérico','ERROR')
```

```
set(handles.adherencia,'String',0);
Adherencia=0;
return
end

if Adherencia<0
errordlg('El valor de la adherencia debe ser
positivo','ERROR')
set(handles.adherencia,'String',0);
Adherencia=0;
return
end

if isnan(Cx)
errordlg('El valor del coeficiente aerodinámico debe ser
numérico','ERROR')
set(handles.cx,'String',0);
Cx=0;
return
end

if Cx<0
errordlg('El valor del coeficiente aerodinámico debe ser
positivo','ERROR')
set(handles.cx,'String',0);
Cx=0;
return
end

if isnan(Af)
errordlg('El valor del área frontal debe ser
numérico','ERROR')
set(handles.af,'String',0);
Af=0;
return
end

if Af<0
errordlg('El valor del área frontal debe ser
positivo','ERROR')
set(handles.af,'String',0);
Af=0;
return
end

if isnan(V)
errordlg('El valor de la velocidad debe ser
numérico','ERROR')
set(handles.v,'String',0);
V=0;
return
end

if V<0
errordlg('El valor de la velocidad debe ser
positivo','ERROR')
set(handles.v,'String',0);
V=0;
```

```
return
end

if isnan(Rampa)
errordlg('El valor de la rampa debe ser numérico','ERROR')
set(handles.rampa,'String',0);
Rampa=0;
return
end

if Rampa<0
errordlg('El valor de la rampa debe ser positivo','ERROR')
set(handles.rampa,'String',0);
Rampa=0;
return
end

if isnan(Rendimiento)
errordlg('El valor del rendimiento debe ser
numérico','ERROR')
set(handles.rendimiento,'String',0);
Rendimiento=0;
return
end

if Rendimiento<0
errordlg('El valor del rendimiento debe ser
positivo','ERROR')
set(handles.rendimiento,'String',0);
Rendimiento=0;
return
end

if isnan(Gamma)
errordlg('El valor del coeficiente de masas rotatorias debe
ser numérico','ERROR')
set(handles.gamma,'String',0);
Gamma=0;
return
end

if Gamma<=0
errordlg('El valor del coeficiente de masas rotatorias debe
ser mayor que 0','ERROR')
set(handles.gamma,'String',0);
Gamma=0;
return
end

close Distancia_y_tiempo
Distancia_y_tiempo_resultados
```

Imágenes:

```
axes1 = imread('distancia002.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

31. DISTANCIA_Y_TIEMPO_RESULTADOS

Tabla 31. Resumen de objetos en "Distancia_y_tiempo_resultados".

Objeto	Tipo	String	Tag	Variable asociada
Distancia recorrida	*Edit text*		spt	Spt
	Static text	Distancia recorrida		
Tiempo hasta parada	*Edit text*		tpt	Tpt
	Static text	Tiempo hasta parada		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

global Peso
global Coeficiente
global Adherencia
global Cx
global Af
global V
global Rampa
global Rendimiento
global Gamma
P=Peso*9.81;
Alfa=asin(Rampa/100);
Rendimiento=0.01*Rendimiento;
V=V/3.6;
Sp=(Peso*Gamma/(9.81*Cx*Af))*log(1+(0.5*Cx*Af*V^2)/(Rendimi
ento*Adherencia*P+P*sin(Alfa)+P*Coeficiente));
Spt=Sp+V*1.3;
Tp=(Peso*Gamma/9.81)*V/(Rendimiento*Adherencia*P+P*sin(Alfa
)+P*Coeficiente);
Tpt=Tp+1.3;
set(handles.spt,'String',Spt);
set(handles.tpt,'String',Tpt);
```

Botones de acción:

- Atrás

```
close Distancia_y_tiempo_resultados
```

```
Distancia_y_tiempo
```

- Menú Principal

```
close Distancia_y_tiempo_resultados
Menu_principal
```

Imágenes:

```
axes1 = imread('distancia004.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

32. POTENCIA_DISIPADA_EN_EL_FRENADO

Tabla 32. Resumen de objetos en "Potencia_disipada_en_el_frenado".

Objeto	Tipo	String	Tag	Variable asociada
Peso	*Edit text*		peso	Peso
	Static text	Peso		
Velocidad	*Edit text*		v	V
	Static text	Velocidad		
Tiempo hasta parada	*Edit text*		t	T
	Static text	Tiempo hasta parada		
Coeficiente de masas rotativas	*Edit text*		gamma	Gamma
	Static text	Masas rotativas		
Rampa	*Edit text*		rampa	Rampa
	Static text	Rampa		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	
Calcular	*Push button*	Calcular	Calcular	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
```

Botones de acción:

- Atrás

```
close Potencia_disipada_en_el_frenado
Menu_frenado
```

- Menú Principal

```
close Potencia_disipada_en_el_frenado
Menu_principal
```

- Calcular

```
global Peso
global V
global T
global Rampa
global Gamma
Peso=str2double(get(handles.peso,'String'));
V=str2double(get(handles.v,'String'));
T=str2double(get(handles.t,'String'));
Rampa=str2double(get(handles.rampa,'String'));
Gamma=str2double(get(handles.gamma,'String'));

if isnan(Peso)
errordlg('El valor del peso debe ser numérico','ERROR')
set(handles.peso,'String',0);
Peso=0;
return
end

if Peso<0
errordlg('El valor del peso debe ser positivo','ERROR')
set(handles.peso,'String',0);
Peso=0;
return
end

if isnan(V)
errordlg('El valor de la velocidad debe ser
numérico','ERROR')
set(handles.v,'String',0);
V=0;
return
end

if V<0
errordlg('El valor de la velocidad debe ser
positivo','ERROR')
set(handles.v,'String',0);
V=0;
return
end
```

```
if isnan(T)
errordlg('El valor del tiempo debe ser numérico','ERROR')
set(handles.t,'String',0);
T=0;
return
end

if T<0
errordlg('El valor del tiempo debe ser positivo','ERROR')
set(handles.t,'String',0);
T=0;
return
end

if isnan(Rampa)
errordlg('El valor de la rampa debe ser numérico','ERROR')
set(handles.rampa,'String',0);
Rampa=0;
return
end

if Rampa<0
errordlg('El valor de la rampa debe ser positivo','ERROR')
set(handles.rampa,'String',0);
Rampa=0;
return
end

if isnan(Gamma)
errordlg('El valor del coeficiente de masas rotatorias debe
ser numérico','ERROR')
set(handles.gamma,'String',0);
Gamma=0;
return
end

if Gamma<=0
errordlg('El valor del coeficiente de masas rotatorias debe
ser mayor que 0','ERROR')
set(handles.gamma,'String',0);
Gamma=0;
return
end

close Potencia_disipada_en_el_frenado
Potencia_disipada_en_el_frenado_resultados
```

Imágenes:

```
axes1 = imread('frenado005.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

33. POTENCIA_DISIPADA_EN_EL_FRENADO_RESULTADOS

Tabla 33. Resumen de objetos en "Potencia_disipada_en_el_frenado_resultados".

Objeto	Tipo	String	Tag	Variable asociada
Frenado brusco	*Edit text*		hm1	Hm1
	Static text	Frenado brusco:		
Descenso prolongado	*Edit text*		hm2	Hm2
	Static text	Descenso prolongado:		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

global Peso
global V
global T
global Rampa
global Gamma
Alfa=asin(Rampa/100);
Vel=V/3.6;
Hm1=0.001*0.5*Peso*Gamma*Vel^2/(9.81*T);
Hm2=0.001*V*Peso*sin(Alfa);
set(handles.hm1,'String',Hm1);
set(handles.hm2,'String',Hm2);
```

Botones de acción:

- Atrás

```
close Potencia_disipada_en_el_frenado_resultados
Potencia_disipada_en_el_frenado
```

- Menú Principal

```
close Potencia_disipada_en_el_frenado_resultados
Menu_principal
```

Imágenes:

```
axes1 = imread('frenado009.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

34. SUSPENSION_RIGIDA

Tabla 34. Resumen de objetos en "Suspensión_rigida".

Objeto	Tipo	String	Tag	Variable asociada
Peso	*Edit text*		peso	Peso
	Static text	Peso		
Giro	*Edit text*		giro	Giro
	Static text	Giro		
Vía	*Edit text*		via	Via
	Static text	Vía		
h	*Edit text*		h	H
	Static text	h		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	
Calcular	*Push button*	Calcular	Calcular	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
```

Botones de acción:

- Atrás

```
close Suspension_rigida
Menu_lateral
```

- Menú Principal

```
close Suspension_rigida
Menu_principal
```

- Calcular

```
global Peso
global Giro
global Via
global H
Peso=str2double(get(handles.peso,'String'));
Giro=str2double(get(handles.giro,'String'));
Via=str2double(get(handles.via,'String'));
H=str2double(get(handles.h,'String'));

if isnan(Via)
errordlg('El valor de la vía debe ser numérico','ERROR')
set(handles.via,'String',0);
Via=0;
return
end

if Via<0
errordlg('El valor de la vía debe ser positivo','ERROR')
set(handles.via,'String',0);
Via=0;
return
end

if isnan(Giro)
errordlg('El valor del giro debe ser numérico','ERROR')
set(handles.giro,'String',0);
Giro=0;
return
end

if isnan(Peso)
errordlg('El valor del peso debe ser numérico','ERROR')
set(handles.peso,'String',0);
Peso=0;
return
end

if Peso<0
errordlg('El valor del peso debe ser positivo','ERROR')
set(handles.peso,'String',0);
Peso=0;
return
end

if isnan(H)
errordlg('El valor de la altura debe ser numérico','ERROR')
set(handles.h,'String',0);
H=0;
return
end

if H<=0
errordlg('El valor de la altura debe ser mayor que
0','ERROR')
set(handles.h,'String',0);
H=0;
return
end
```

```
close Suspension_rigida
Suspension_rigida_resultados
```

Imágenes:

```
axes1 = imread('suspension_rigida.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

35. SUSPENSION_RIGIDA_RESULTADOS

Tabla 35. Resumen de objetos en "Suspensión_rigida_resultados".

Objeto	Tipo	String	Tag	Variable asociada
Aceleración lateral máxima	*Edit text*		a_ymax	A_ymax
	Static text	Aceleración lateral máxima		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

global Peso
global Giro
global Via
global H
P=Peso*9.81;
A_ymax=Via*9.81/(2*H);

A_y=(0:0.1:(A_ymax+2));
M_yv=(P*H/9.81)*A_y;
M_yR=0.5*P*Via;
M_yD=P*H*Giro;
W=waitbar(0.5,'Calculando...');
pause(1);
close(W);
set(handles.a_ymax,'String',A_ymax);
```

```
axes(handles.axes1)

axis on
plot(-A_y,M_yv);
hold on;
plot(-
A_y,P*Via*0.5*ones(length(A_y)),'r',xlabel('a_y'),ylabel('M
_y'))
hold on;
A=[-A_ymax -A_ymax];
I=[0 M_yR];
plot(A,I,'k-.')
hold off;
grid on;
xlim([-A_ymax-2 0]);
ylim([0 P*(A_ymax+2)*H/9.81]);

axes(handles.axes2)

phi=(0:0.001:Giro);
M_yRN=M_yR-P*H*phi;
plot(phi,M_yRN,'b');
hold on;
plot(phi,0.5*P*Via*ones(length(phi)),'r-
.',xlabel('O'),ylabel(' '));
hold off;
grid on;
xlim([0 Giro]);
ylim([0 P*(A_ymax+2)*H/9.81]);
```

Botones de acción:

- Atrás

```
close Suspension_rigida_resultados
Suspension_rigida
```

- Menú Principal

```
close Suspension_rigida_resultados
Menu_principal
```

Imágenes:

```
axes3 = imread('vuelco003.jpg');
axes(handles.axes3);
axis off;
imshow(axes3);
```

36. SUSPENSION_ELASTICA

Tabla 36. Resumen de objetos en "Suspensión_elastica".

Objeto	Tipo	String	Tag	Variable asociada
Peso	*Edit text*		peso	Peso
	Static text	Peso		
Giro	*Edit text*		giro	Giro
	Static text	Giro		
Vía	*Edit text*		via	Via
	Static text	Vía		
h	*Edit text*		h	H
	Static text	h		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	
Calcular	*Push button*	Calcular	Calcular	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
```

Botones de acción:

- Atrás

```
close Suspension_elastica
Menu_lateral
```

- Menú Principal

```
close Suspension_elastica
Menu_principal
```

- Calcular

```
global Peso
global Giro
global Via
global H
Peso=str2double(get(handles.peso,'String'));
Giro=str2double(get(handles.giro,'String'));
```

```
Via=str2double(get(handles.via,'String'));
H=str2double(get(handles.h,'String'));

if isnan(Via)
errordlg('El valor de la vía debe ser numérico','ERROR')
set(handles.via,'String',0);
Via=0;
return
end

if Via<0
errordlg('El valor de la vía debe ser positivo','ERROR')
set(handles.via,'String',0);
Via=0;
return
end

if isnan(Giro)
errordlg('El valor del giro debe ser numérico','ERROR')
set(handles.giro,'String',0);
Giro=0;
return
end

if isnan(Peso)
errordlg('El valor del peso debe ser numérico','ERROR')
set(handles.peso,'String',0);
Peso=0;
return
end

if Peso<0
errordlg('El valor del peso debe ser positivo','ERROR')
set(handles.peso,'String',0);
Peso=0;
return
end

if isnan(H)
errordlg('El valor de la altura debe ser numérico','ERROR')
set(handles.h,'String',0);
H=0;
return
end

if H<=0
errordlg('El valor de la altura debe ser mayor que
0','ERROR')
set(handles.h,'String',0);
H=0;
return
end

close Suspension_elastica
Suspension_elastica_resultados
```

Imágenes:

```
axes1 = imread('suspension_rigida.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

37. SUSPENSION_ELASTICA_RESULTADOS

Tabla 37. Resumen de objetos en "Suspensión_elastica_resultados".

Objeto	Tipo	String	Tag	Variable asociada
Aceleración lateral máxima	*Edit text*		a_ymax	A_ymax
	Static text	Aceleración lateral máxima		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

global Peso
global Giro
global Via
global H
P=Peso*9.81;
A_ymax=(Via/(2*H)-Giro)*9.81;

set(handles.a_ymax,'String',A_ymax);

A_y=(0:0.1:(A_ymax+2));
M_yv=(P*H/9.81)*A_y;
M_yR1=0.5*P*Via;
M_yR=P*H*A_ymax/9.81;
M_yD=P*H*Giro;
W=waitbar(0.5,'Calculando...');
pause(1);
close(W);

axes(handles.axes1)

axis on
plot(-A_y,M_yv);
```

```
hold on;
plot(-
A_y,P*Via*0.5*ones(length(A_y)),'r',xlabel('a_y'),ylabel('M
_y'))
hold on;
plot(-A_y,M_yR*ones(length(A_y)),'r')
hold on;
A=[-A_ymax -A_ymax];
I=[0 M_yR];
plot(A,I,'k-.')
hold off;
grid on;
xlim([-A_ymax-2 0]);
ylim([0 0.5*P*Via+100]);

axes(handles.axes2)

phi=(0:0.001:Giro*(1+0.5));
M_yRN=M_yR-P*H*phi;
plot(phi,M_yRN,'b');
hold on;
plot(phi,0.5*P*Via*ones(length(phi)),'r-
.',xlabel('O'),ylabel(' '));
hold on;
OX=(0:0.01:Giro);
plot(OX,M_yR*ones(length(OX),1),'r')
hold off;
grid on;
xlim([0 Giro*(1+0.5)]);
ylim([0 0.5*P*Via+100]);
```

Botones de acción:

- Atrás

```
close Suspension_elastica_resultados
Suspension_elastica
```

- Menú Principal

```
close Suspension_elastica_resultados
Menu_principal
```

Imágenes:

```
axes3 = imread('vuelco001.jpg');
axes(handles.axes3);
axis off;
imshow(axes3);
```

38. SUSPENSION_BALLESTAS

Tabla 38. Resumen de objetos en "Suspensión_ballestas".

Objeto	Tipo	String	Tag	Variable asociada
Peso	*Edit text*		peso	Peso
	Static text	Peso		
Giro	*Edit text*		giro	Giro
	Static text	Giro		
Vía	*Edit text*		via	Via
	Static text	Vía		
h	*Edit text*		h	H
	Static text	h		
Juego libre	*Edit text*		j	J
	Static text	Juego libre		
Distancia entre suspensiones	*Edit text*		c	C
	Static text	Distancia suspensiones		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	
Calcular	*Push button*	Calcular	Calcular	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
```

Botones de acción:

- Atrás

```
close Suspension_ballestas
Menu_lateral
```

- Menú Principal

```
close Suspension_ballestas
Menu_principal
```

- Calcular

```
global Peso
global Giro
global Via
global H
global J
global C
Peso=str2double(get(handles.peso,'String'));
Giro=str2double(get(handles.giro,'String'));
Via=str2double(get(handles.via,'String'));
H=str2double(get(handles.h,'String'));
J=str2double(get(handles.j,'String'));
C=str2double(get(handles.c,'String'));

if isnan(Via)
errordlg('El valor de la vía debe ser numérico','ERROR')
set(handles.via,'String',0);
Via=0;
return
end

if Via<0
errordlg('El valor de la vía debe ser positivo','ERROR')
set(handles.via,'String',0);
Via=0;
return
end

if isnan(Giro)
errordlg('El valor del giro debe ser numérico','ERROR')
set(handles.giro,'String',0);
Giro=0;
return
end

if isnan(Peso)
errordlg('El valor del peso debe ser numérico','ERROR')
set(handles.peso,'String',0);
Peso=0;
return
end

if Peso<0
errordlg('El valor del peso debe ser positivo','ERROR')
set(handles.peso,'String',0);
Peso=0;
return
end

if isnan(H)
errordlg('El valor de la altura debe ser numérico','ERROR')
set(handles.h,'String',0);
H=0;
return
end

if H<=0
```

```
errordlg('El valor de la altura debe ser mayor que
0','ERROR')
set(handles.h,'String',0);
H=0;
return
end

if isnan(J)
errordlg('El valor del juego debe ser numérico','ERROR')
set(handles.j,'String',0);
J=0;
return
end

if J<0
errordlg('El valor del juego debe ser positivo','ERROR')
set(handles.j,'String',0);
J=0;
return
end

if isnan(C)
errordlg('El valor de la distancia entre suspensiones debe
ser numérico','ERROR')
set(handles.c,'String',0);
C=0;
return
end

if C<0
errordlg('El valor de la distancia entre suspensiones debe
ser positivo','ERROR')
set(handles.c,'String',0);
C=0;
return
end

close Suspension_ballestas
Suspension_ballestas_resultados
```

Imágenes:

```
axes1 = imread('suspension_ballestas.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

39. SUSPENSION_BALLESTAS_RESULTADOS

Tabla 39. Resumen de objetos en "Suspensión_ballestas_resultados".

Objeto	Tipo	String	Tag	Variable asociada
Aceleración lateral máxima	*Edit text*		a_ymax	A_ymax
	Static text	Aceleración lateral máxima		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

global Peso
global Giro
global Via
global H
global J
global C
P=Peso*9.81;
Giro=Giro+J/C;
A_ymax=(Via/(2*H)-Giro)*9.81;

A_y=(0:0.1:(A_ymax+2));
M_yv=(P*H/9.81)*A_y;
M_yR1=0.5*P*Via;
M_yR=P*H*A_ymax/9.81;
M_yD=P*H*Giro;
W=waitbar(0.5,'Calculando...');
pause(1);
close(W);
set(handles.a_ymax,'String',A_ymax);

axes(handles.axes1)

axis on
plot(-A_y,M_yv);
hold on;
plot(-
A_y,P*Via*0.5*ones(length(A_y)),'g',xlabel('a_y'),ylabel('M
_y'))
hold on;
plot(-A_y,M_yR*ones(length(A_y)),'r')
```

```
hold on;
A=[-A_ymax -A_ymax];
I=[0 M_yR];
plot(A,I,'k-.')
hold off;
grid on;
xlim([-A_ymax-2 0]);
ylim([0 0.5*P*Via+100]);

axes(handles.axes2)

phi=(0:0.001:Giro*(1+0.5));
M_yRN=M_yR-P*H*phi;
plot(phi,M_yRN,'b');
hold on;
plot(phi,0.5*P*Via*ones(length(phi)),'g-
.',xlabel('O'),ylabel(' '));
hold on;
OX=(0:0.0001:Giro);
plot(OX,M_yR*ones(length(OX),1),'r')
hold on;
A=[Giro Giro];
I=[0 M_yR];
plot(A,I,'k-.')
hold off;
grid on;
xlim([0 Giro*(1+0.5)]);
ylim([0 0.5*P*Via+100]);
```

Botones de acción:

- Atrás

```
close Suspension_ballestas_resultados
Suspension_ballestas
```

- Menú Principal

```
close Suspension_ballestas_resultados
Menu_principal
```

Imágenes:

```
axes3 = imread('vuelco005.jpg');
axes(handles.axes3);
axis off;
imshow(axes3);
```

40. CENTROS_BALANCEO

Tabla 40. Resumen de objetos en "Centros_balanceo".

Objeto	Tipo	String	Tag	Variable asociada
Peso	*Edit text*		peso	Peso
	Static text	Peso		
Giro1	*Edit text*		giro1	Giro1
	Static text	Giro1		
Giro2	*Edit text*		giro2	Giro2
	Static text	Giro2		
Vía	*Edit text*		via	Via
	Static text	Vía		
h1	*Edit text*		h1	H1
	Static text	h1		
h2	*Edit text*		h2	H2
	Static text	h2		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	
Calcular	*Push button*	Calcular	Calcular	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
```

Botones de acción:

- Atrás

```
close Centros_balanceo
Menu_lateral
```

- Menú Principal

```
close Centros_balanceo
Menu_principal
```

Imágenes:

```
axes1 = imread('suspension_centros_balanceo.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

41. CENTROS_BALANCEO_RESULTADOS

Tabla 41. Resumen de objetos en "Centros_balanceo_resultados".

Objeto	Tipo	String	Tag	Variable asociada
Límite de vuelco A	*Edit text*		a_ymaxA	A_ymaxA
	Static text	Límite de vuelco A		
Límite de vuelco B	*Edit text*		a_ymaxB	A_ymaxB
	Static text	Límite de vuelco B		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

global Peso
global Giro1
global Giro2
global Via
global H1
global H2
P=Peso*9.81;
H=H1+H2;

A_ymaxA=(Via/(2*H)-Giro2)*9.81;

A_y=(0:0.1:(A_ymaxA+2));
M_yv=(P*H/9.81)*A_y;
M_yR1=0.5*P*Via;
M_yR=P*H*A_ymaxA/9.81;
M_yD=P*H*Giro2;

W=waitbar(0.5,'Calculando...');
pause(1);
close(W);
set(handles.a_ymaxA,'String',A_ymaxA);
```

```
axes(handles.axes1)

axis on
plot(-A_y,M_yv);
hold on;
plot(-
A_y,P*Via*0.5*ones(length(A_y)),'r',xlabel('a_y'),ylabel('M
_y'))
hold on;
A=[0 -A_ymaxA];
I=[M_yR M_yR];
plot(A,I,'b')
hold on;
A=[-A_ymaxA -A_ymaxA];
I=[0 M_yR];
plot(A,I,'k-.')
hold on;
grid on;

axes(handles.axes2)

phi=(0:0.001:Giro2*(1+0.5));
M_yRN=M_yR1-P*H*phi;
plot(phi,M_yRN,'k');
hold on;
plot(phi,0.5*P*Via*ones(length(phi)),'k-
.',xlabel('O'),ylabel(' '));
hold on;
OX=(0:0.01:Giro2);
plot(OX,M_yR*ones(length(OX),1),'b')
hold on;
A=[Giro2 Giro2];
I=[0 M_yR];
plot(A,I,'k-.')
grid on;
xlim([0 Giro2*(1+0.5)]);
ylim([0 0.5*P*Via+100]);

A_ymaxB=(Via/(2*H))*9.81;

set(handles.a_ymaxB,'String',A_ymaxB);

A_y=(0:0.1:(A_ymaxB+2));
M_yv=(P*H/9.81)*A_y;
M_yR=0.5*P*Via;
M_yD=P*H*Giro1;

axes(handles.axes1)

axis on
plot(-A_y,M_yv,'k');
hold on;
plot(-
A_y,P*Via*0.5*ones(length(A_y)),xlabel('a_y'),ylabel('M_y')
)
hold on;
A=[0 -A_ymaxB];
```

```
I=[M_yR M_yR];
plot(A,I,'b')
hold on;
A=[-A_ymaxB -A_ymaxB];
I=[0 M_yR];
plot(A,I,'k-.')
hold on;
XLIM([-A_ymaxB-2 0])
YLIM([0 0.5*P*Via+100])
grid on;
```

Botones de acción:

- Atrás

```
close Centros_balanceo_resultados
Centros_balanceo
```

- Menú Principal

```
close Centros_balanceo_resultados
Menu_principal
```

Imágenes:

```
axes3 = imread('vuelco002.jpg');
axes(handles.axes3);
axis off;
imshow(axes3);
```

42. TRANSFERENCIA_DE_CARGA_LATERAL

Tabla 42. Resumen de objetos en "Transferencia_de_carga_lateral".

Objeto	Tipo	String	Tag	Variable asociada
ms	*Edit text*		ms	Ms
	Static text	ms		
msd	*Edit text*		msd	Msd
	Static text	msd		
mst	*Edit text*		mst	Mst
	Static text	mst		
mnsd	*Edit text*		mnsd	Mnsd
	Static text	mnsd		
mnst	*Edit text*		mnst	Mnst
	Static text	mnst		
r1	*Edit text*		r1	R1
	Static text	r1		
hd	*Edit text*		hd	Hd
	Static text	hd		
ht	*Edit text*		ht	Ht
	Static text	ht		
h1	*Edit text*		h1	H1
	Static text	h1		
h2	*Edit text*		h2	H2
	Static text	h2		
Bd	*Edit text*		bd	Bd
	Static text	Bd		
Bt	*Edit text*		bt	Bt
	Static text	Bt		
Aceleración lateral	*Edit text*		ay	Ay
	Static text	Aceleración lateral		
Kd	*Edit text*		kd	Kd
	Static text	Kd		
Kt	*Edit text*		kt	Kt
	Static text	Kt		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	
Calcular	*Push button*	Calcular	Calcular	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
```

```
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
```

Botones de acción:

- Atrás

```
close Transferencia_de_carga_lateral
Menu_lateral
```

- Menú Principal

```
close Transferencia_de_carga_lateral
Menu_principal
```

- Calcular

```
global Ms
global Msd
global Mst
global Mnsd
global Mnst
global R1
global Hd
global Ht
global H1
global H2
global Bd
global Bt
global Ay
global Kd
global Kt
Ms=str2double(get(handles.ms,'String'));
Msd=str2double(get(handles.msd,'String'));
Mst=str2double(get(handles.mst,'String'));
Mnsd=str2double(get(handles.mnsd,'String'));
Mnst=str2double(get(handles.mnst,'String'));
R1=str2double(get(handles.r1,'String'));
Hd=str2double(get(handles.hd,'String'));
Ht=str2double(get(handles.ht,'String'));
H1=str2double(get(handles.h1,'String'));
H2=str2double(get(handles.h2,'String'));
Bd=str2double(get(handles.bd,'String'));
Bt=str2double(get(handles.bt,'String'));
Ay=str2double(get(handles.ay,'String'));
Kd=str2double(get(handles.kd,'String'));
Kt=str2double(get(handles.kt,'String'));

if isnan(Ms)
errordlg('El valor de la masa suspendida debe ser
numérico','ERROR')
set(handles.ms,'String',0);
Ms=0;
return
```

```
end

if Ms<=0
errordlg('El valor de la masa suspendida debe ser mayor que
0','ERROR')
set(handles.ms,'String',0);
Ms=0;
return
end

if isnan(Msd)
errordlg('El valor de la masa suspendida delantera debe ser
numérico','ERROR')
set(handles.msd,'String',0);
Msd=0;
return
end

if Msd<=0
errordlg('El valor de la masa suspendida delantera debe ser
mayor que 0','ERROR')
set(handles.msd,'String',0);
Msd=0;
return
end

if isnan(Mst)
errordlg('El valor de la masa suspendida trasera debe ser
numérico','ERROR')
set(handles.mst,'String',0);
Mst=0;
return
end

if Mst<=0
errordlg('El valor de la masa suspendida trasera debe ser
mayor que 0','ERROR')
set(handles.mst,'String',0);
Mst=0;
return
end

if isnan(Mnsd)
errordlg('El valor de la masa no suspendida delantera debe
ser numérico','ERROR')
set(handles.mnsd,'String',0);
Mnsd=0;
return
end

if Mnsd<=0
errordlg('El valor de la masa no suspendida delantera debe
ser mayor que 0','ERROR')
set(handles.mnsd,'String',0);
Mnsd=0;
return
end
```

```
if isnan(Mnst)
errordlg('El valor de la masa no suspendida trasera debe
ser numérico','ERROR')
set(handles.mnst,'String',0);
Mnst=0;
return
end

if Mst<=0
errordlg('El valor de la masa suspendida trasera debe ser
mayor que 0','ERROR')
set(handles.mst,'String',0);
Mst=0;
return
end

close Transferencia_de_carga_lateral
Transferencia_de_carga_lateral_resultados
```

Imágenes:

```
axes1 = imread('transferencia001.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);
```

43. TRANSFERENCIA_DE_CARGA_LATERAL_RESULTADOS

Tabla 43. Resumen de objetos en "Transferencia_de_carga_lateral_resultados".

Objeto	Tipo	String	Tag	Variable asociada
Transferencia de carga delantera	*Edit text*		trans_del	Trans_del
	Static text	Transferencia de carga delantera		
Transferencia de carga trasera	*Edit text*		trans_tra	Trans_tra
	Static text	Transferencia de carga trasera		
Atrás	*Push button*	Atrás	atras	
Menú Principal	*Push button*	Menú Principal	Menu_ppal	

Código inicial:

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
```

```
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

global Ms
global Msd
global Mst
global Mnsd
global Mnst
global R1
global Hd
global Ht
global H1
global H2
global Bd
global Bt
global Ay
global Kd
global Kt

Fd=(Ay/Bd)*(Kd*Ms*R1/(Kd+Kt-Ms*9.81*R1)+Msd*Hd+Mnsd+H1);
Ft=(Ay/Bt)*(Kt*Ms*R1/(Kd+Kt-Ms*9.81*R1)+Mst*Ht+Mnst+H2);
%Para expresarlo en kg
Fd=Fd/9.81;
Ft=Ft/9.81;
set(handles.trans_del,'String',Fd);
set(handles.trans_tra,'String',Ft);
```

Botones de acción:

- Atrás

```
close Transferencia_de_carga_lateral_resultados
Transferencia_de_carga_lateral
```

- Menú Principal

```
close Transferencia_de_carga_lateral_resultados
Menu_principal
```

Imágenes:

```
axes1 = imread('vuelco007.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);

axes4 = imread('transferencia002.jpg');
axes(handles.axes4);
axis off;
imshow(axes4);
```

Consideraciones sobre la programación

Para modificar la aplicación "Análisis dinámico de un vehículo automóvil", ténganse en cuenta las siguientes consideraciones.

- Todas las GUIs han sido diseñadas con el mismo tamaño: [520, -45, 1152, 845]. Al crear una nueva GUI, establecer estos mismos parámetros.
- Los *tags* de los objetos se guardan en el *handles* de cada GUI, por lo tanto pueden repetirse en diferentes GUIs, pero no en una misma. Así, objetos comunes en todas las GUIs, como los botones de "Menú principal" o "Atrás", pueden compartir el mismo *tag* aunque su *callback* sea distinto.
- Para recuperar el valor de una variable establecida en una GUI y usarlo en una segunda, en ambas GUIs se deben declarar las variables como globales, pues la memoria del *handles* se borra al cerrar cada GUI.
- Para conseguir máxima compatibilidad, establecer en las opciones de preferencias de cada GUI, compatibilidad con MatLab 5. Ver Figura 1.

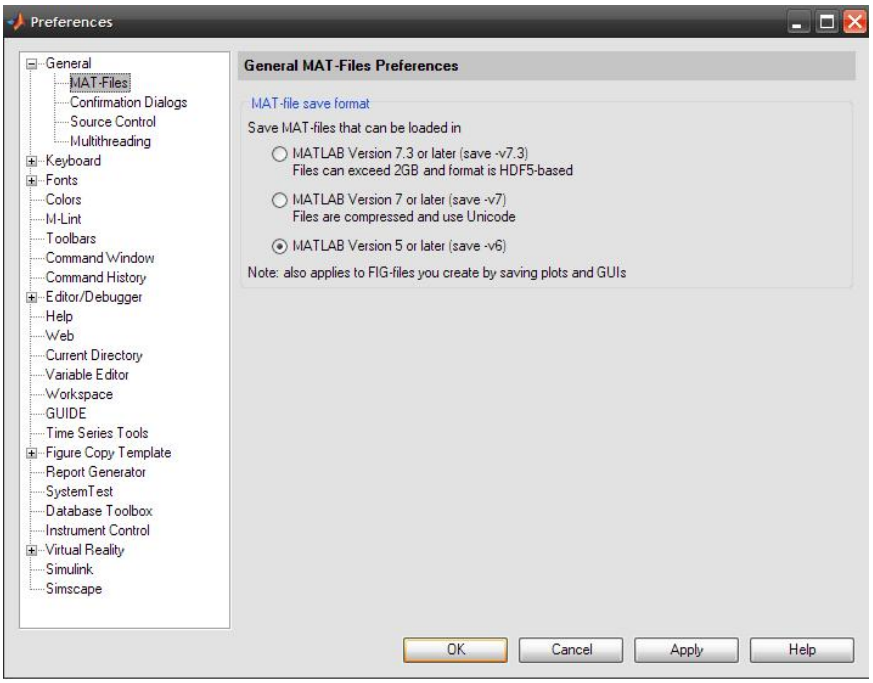

Figura 1. Captura de pantalla del menú de preferencias de GUIDE. Detalle de las preferencias del formato de los archivos

- Todas las imágenes, *figures* y *m-files*, deben estar en la misma carpeta (no se pueden incluir subcarpetas), que a su vez debe ser la carpeta de trabajo especificada a través del "*Current directory browser*".
- MatLab puede confundir fácilmente los nombres de los *tags* de los de las variables, ya que diferencia entre mayúsculas y minúsculas. Se recomienda, para seguir con el mismo criterio, nombrar a todos los *tags* sólo en minúsculas y las variables correspondientes comenzando en mayúscula.
- Para seguir con el mismo criterio, siempre que se vaya a programar una prestación nueva, crear una GUI donde se introducen los datos de entrada y una segunda GUI donde se muestren los resultados. En esta segunda GUI es donde se realizan las operaciones necesarias.
- Todas las GUIs creadas tienen fondo blanco. El fondo de una GUI se modifica haciendo doble clic sobre el mismo al editar la figura, abriéndose el *editor inspector* del objeto que es el fondo.
- Tratar de mantener una coherencia estética. Por ejemplo, se han diferenciado los tres análisis de tracción, frenado y dinámica lateral, por colores. Si se incluye un análisis de dinámica vertical, establecer otro color. De igual modo, tratar de mantener el criterio de cuadros de títulos establecido.