UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

MODELADO Y CARACTERIZACIÓN DE SISTEMAS DE SUSPENSIÓN EN VEHÍCULOS AUTOMÓVILES

GRADO EN INGENIERÍA MECÁNICA

TRABAJO FIN DE GRADO

DEPARTAMENTO DE INGENIERÍA MECÁNICA

Autor:

Bruno Cebolla Bono

Tutor:

D. Juan Francisco Dols Ruiz

Septiembre 2017

ÍNDICE

1. INTRODUCCIÓN.

El propósito del presente trabajo es desarrollar una aplicación informática que sirva como herramienta para el estudio del comportamiento vertical de un vehículo sometido a las fuerzas excitadoras introducidas por la carretera. Dicha aplicación se utilizará en la asignatura de automóviles que se imparte como optativa en el grado de ingeniería mecánica de la Universitat Politècnica de València. El objetivo es proporcionar a los alumnos una herramienta que sirva como primera aproximación al estudio de las vibraciones en un vehículo automóvil.

La aplicación constará de una interfaz gráfica que permita elegir entre diferentes modelos del vehículo, que van desde un cuarto de vehículo (modelos de 1 g.d.l y 2 g.d.l), medio vehículo (modelos de 4 g.d.l) y del vehículo completo (modelos de 7 g.d.l). Además, en los modelos de cuatro y siete grados de libertad, se permitirá elegir entre dos tipos de suspensión como son la suspensión independiente y la suspensión de eje rígido. En el caso del modelo de siete grados de libertad la suspensión de eje rígido se limita al tren posterior. Además, la interfaz permite modificar el valor de cualquier parámetro de la suspensión (rigidez, amortiguamiento, etc.) y representar la respuesta del vehículo en el dominio de la frecuencia.

Para llevar a cabo dicha aplicación se ha elegido la herramienta GUIDE de MATLAB, la cual permite crear una interfaz gráfica de usuario con diferentes elementos que permiten la entrada de datos numéricos, así como otros que permiten la visualización de resultados ya sean numéricos o gráficos. La interfaz gráfica, en adelante GUI (Graphical User Interface) constara de diferentes ventanas que se dividen en, ventanas de entradas de datos y ventanas de visualización de resultados. En las ventanas de entrada de datos se muestra al usuario todos los parámetros que hará falta introducir, y por tanto deberán ser conocidos de antemano, para poder realizar los cálculos y llevar a cabo el estudio de dicho modelo. Además, esta ventana proporcionará una breve descripción de cada parámetro, así como una imagen que ubica y relaciona cada parámetro con su posición en el vehículo. Una vez introducidos los datos se podrá pasar a la ventada de resultados donde se mostrarán los resultados fruto de un análisis en frecuencia que caracterizará el comportamiento vertical del vehículo. Dichos resultados se compondrán tanto de resultados numéricos como gráficos.

Para programar la GUI se ha utilizado el lenguaje de programación de MATLAB, el cual permite un tratamiento más o menos sencillo de matrices y cálculos numéricos, así como la obtención de gráficos.

2. VIBRACIONES EN EL VEHÍCULO.

Los Automóviles están sometidos a un amplio espectro de vibraciones, que se transmiten a los pasajeros de forma táctil, visual o audible. El término **vibraciones** se usa normalmente en referencia a vibraciones táctiles y visuales, mientras que las vibraciones audibles se definen como **ruido**. El espectro de vibraciones se puede dividir de acuerdo con la frecuencia y clasificarlas como vibraciones (0-25 Hz) y ruido (25-25000 Hz). El punto límite de 25 Hz es aproximadamente la frecuencia inferior del umbral de audición, mientras que el límite superior de frecuencia es la vibración simple común a todos los vehículos de motor. Para entender el entorno vibracional del vehículo hay que analizar las fuentes de excitación de vibraciones, la respuesta del vehículo y la percepción humana y tolerancia a las vibraciones. Si nos limitamos únicamente a las frecuencias verticales, la gama existente puede ser dividida en tres tipos:

- 1-3 Hz: Corresponden a las frecuencias naturales de la carrocería.
- 5-40 Hz: Frecuencias de oscilación de las masas no suspendidas (generalmente entre 10 y 20 Hz).
- 40-250 Hz: Oscilaciones producidas en las masas no suspendidas, debidas a las vibraciones naturales en los neumáticos.

De este espectro, los elementos comunes que afectan al confort de marcha del pasajero están referidos a las vibraciones táctiles y visuales, mientras que las vibraciones auditivas quedan reconocidas dentro del campo de los "ruidos".

Las fuentes de excitación por las que se originan vibraciones en el vehículo se pueden dividir en dos grandes grupos:

- Ajenas al vehículo o **indirectas**: son las que se transmiten a la masa suspendida a través de las masas no suspendidas y cuyo principal exponente es el estado del terreno (irregularidades de la carretera) por donde debe circular el mismo.
- Propias del vehículo o **directas**: son las ejercidas sobre la masa suspendida por elementos contenidos o apoyados en ella, es decir, son fuentes de excitación de vibraciones que están incorporadas al propio vehículo y que surgen principalmente de componentes giratorios o rotativos del mismo, como son conjuntos llantas/neumáticos, el sistema de tracción/transmisión, el motor y acciones aerodinámicas.

2.1. Irregularidades del terreno.

La rugosidad de la carretera incluye desde los baches resultantes de fallos localizados en el pavimento, hasta las inevitables desviaciones aleatorias procedentes de los límites prácticos de precisión con los que es posible llevar a cabo la construcción y el mantenimiento de la superficie de la carretera. La rugosidad o irregularidad se define en función de la modificación del perfil, a lo largo del ancho de vía, sobre el que el vehículo pasa. Los perfiles de carretera pueden incluirse en la categoría general de *señales aleatorias de banda ancha*, y por ello pueden escribirse tanto por el propio perfil como por sus propiedades estadísticas. Una de las representaciones más utilizadas es la función denominada **Densidad espectral de Potencia** (*Power Spectral Density*, *PSD*), o simplemente densidad espectral.

Las ruedas del vehículo están excitadas verticalmente por las irregularidades de la carretera, que pueden considerarse fenómenos aleatorios y, por tanto, representarse mediante funciones aleatorias.

A la hora de estudiar las irregularidades de la carretera, se supone que el perfil superficial de la carretera es una función aleatoria ergódica, por lo que, estudiando un tramo de la carretera suficientemente representativo, se puede caracterizar toda la carretera. Este estudio pasa por la determinación de parámetros estadísticos adecuados, caracterizándose la curva de densidad espectral (S_z) de sus irregularidades verticales. Las irregularidades de las carreteras se pueden representar como la suma de un número casi infinito de irregularidades armónicas cuyas amplitudes disminuyen a medida que aumenta la frecuencia.

Para irregularidades de carretera, la relación aproximada entre la densidad espectral y la frecuencia espacial es:

$$S_z(\omega) = C_{sp} \cdot \omega^{-N} \qquad (1)$$

donde C_{sp} y N son coeficientes que dependen del tipo de carretera. En la siguiente tabla se dan valores de estos coeficientes para algunos tipos de carreteras.

Tabla 1 - .Coeficientes en función de la carretera.

Tipo de superficie	N	C_{sp}
Autopista uniforme	3,8	$4{,}3 \cdot 10^{-11}$
Autopista rugosa	2,1	$8{,}1 \cdot 10^{-6}$
Carretera uniforme	2,1	$4{,}8 \cdot 10^{-7}$
Carretera con grava	2,1	$4{,}4 \cdot 10^{-6}$
Campo arado	1,6	$66{,}5 \cdot 10^{-4}$

Si se tiene en cuenta que, cuando un vehículo circula por una carretera a una velocidad determinada, sus ruedas recorren las irregularidades de la carretera a una velocidad que depende de la de movimiento del vehículo, las irregularidades espaciales $z(x)$ se convierten en excitaciones temporales en el vehículo $z(t)$.

Las frecuencias espacial y temporal se pueden relacionar mediante la expresión:

$$frec_{temporal}\left[\frac{ciclos}{seg}\right] = frec_{espacial}\left[\frac{ciclos}{m}\right] \cdot velocidad\left[\frac{m}{seg}\right] \qquad (2)$$

Entonces, las curvas de densidad espectral de la carretera se pueden transformar en curvas de densidad espectral de excitación vertical sobre el vehículo (sobre las ruedas):

$$S_{excitac\_Rueda}\left[\frac{m^2}{\frac{ciclos}{s}}\right] = \frac{S_{Carretera}\left[\frac{m^2}{\frac{ciclo}{m}}\right]}{velocidad\left[\frac{m}{s}\right]} \qquad (3)$$

Conociendo la densidad espectral de excitación en una rueda y utilizando la función de transferencia, se puede obtener la densidad espectral de respuesta en cualquier grado de libertad. Si $H(\omega)$ es la función de transferencia entre la excitación vertical en la rueda y su respuesta en un grado de libertad concreto, y $S_r(\omega)$ es la densidad espectral de la respuesta, entonces:

$$S_r(\omega) = |H(\omega)|^2 \cdot S_z(\omega) \quad (4)$$

La elevación del perfil medido sobre una longitud de la carretera se puede descomponer mediante la transformada de Fourier en una serie de ondas senoidales de fase y amplitud variables. La densidad espectral de potencia o PSD es una representación de las amplitudes en función de las frecuencias espaciales. La forma más útil y sencilla de medir las vibraciones es la aceleración producida. Para comprender la dinámica del vehículo, las irregularidades se deben ver como aceleraciones en las ruedas, y por ello el perfil de elevación se transforma en desplazamientos que son función del tiempo.

Los puntos del perfil de la carretera en las ruedas izquierda y derecha se promedian generalmente antes de obtener el PSD, aunque el PSD para cada una de las ruedas suele ser generalmente muy parecido al promedio. La diferencia de elevación entre los puntos de los perfiles izquierdo y derecho representa una excitación de balanceo del vehículo.

A bajas frecuencias el balanceo que proviene de la carretera es mucho menor que la excitación vertical. No obstante, el balanceo crece con la frecuencia espacial, porque la respuesta a compresión de la suspensión en las ruedas disminuye con el incremento de la excitación vertical. Para la mayoría de los vehículos la resonancia en balanceo aparece a frecuencias menores (entre 0,5 y 1 Hz) que la resonancia en vaivén vertical, con lo que la respuesta dominante es el vaivén. A elevadas frecuencias, donde las excitaciones de vaivén y balanceo son aproximadamente de la misma magnitud, los vehículos son menos sensibles al balanceo.

En general se acepta que ondulaciones con amplitudes cuyos valores superan los 2 cm aproximadamente son molestas a velocidades normales de marcha. Amplitudes menores de 1,3 cm son características de firmes de calidad media, y valores inferiores a 5 mm son indicativos de firmes con una gran calidad superficial. Otra forma de modelizar las irregularidades del firme es expresar éstas como suma de las amplitudes medidas por unidad de longitud rastreada. Por lo general, los valores más adecuados oscilan entre 1,18 y 1,34 m/km, aunque los más habituales se encuentran entre 1,58 y 3,95 m/km.

2.2. Fuentes de vibración propias del vehículo.

Las vibraciones producidas por el propio vehículo provienen esencialmente de las ruedas, del grupo motor-caja de cambios y de la transmisión. Idealmente el conjunto llanta/neumático, es suave y flexible para absorber parte de las perturbaciones producidas por los baches e irregularidades de la calzada, y no contribuye a la excitación del vehículo. En la práctica, las imperfecciones y defectos en la fabricación de las llantas, neumáticos, uniones, mangueta, frenos y elementos giratorios suelen dar lugar a irregularidades que facilitan la transmisión de vibraciones y que pueden ser agrupadas en desequilibrio de masas, variaciones dimensionales y variaciones de rigidez. Estas irregularidades combinadas en el sistema llanta/neumático ocasionan variaciones en las fuerzas y momentos que se transmiten al eje del vehículo y actúan como fuente de vibraciones.

El sistema de transmisión está formado por el embrague, la caja de cambios, el diferencial, el árbol de transmisión y los semiejes o palieres que se unen a las ruedas. De estos componentes, el árbol de transmisión es una de las principales fuentes de vibraciones. Esto es debido al desequilibrio de las masas de los elementos sobre la cadena de transmisión y a los pares secundarios, o momentos, sobre la cadena de transmisión.

La propia naturaleza mecánica del motor, y el hecho de que gire y proporcione par al sistema de transmisión, hacen que sea una fuente de excitación de vibraciones del vehículo. Por otra parte, la masa del motor, en combinación con la masa de los elementos que forman la caja de cambios,

es una parte substancial del chasis que, si se diseña de forma adecuada, puede actuar como elemento de absorción de vibraciones.

2.3. Vibraciones aerodinámicas.

El flujo de aire alrededor del vehículo causa vibraciones de origen aerodinámico. Ciertos elementos de la carrocería como son los espejos retrovisores, antenas de radio y otros salientes o protuberancias, originan separaciones periódicas de flujo. La frecuencia de estas alteraciones se sitúa a menudo dentro del campo audible, aunque las de baja frecuencia pueden ser causa de vibraciones de las que se debe hacer cargo el sistema de suspensión.

2.4. Percepción y tolerancia humanas a las vibraciones.

El diseño de los vehículos automóviles debe tener presente el comportamiento del cuerpo humano desde el punto de vista mecánico, así como la capacidad para soportar vibraciones. En los automóviles actuales, los desarrollos se centran en la línea de la comodidad y la seguridad, tratando de disminuir el cansancio producido y la pérdida de atención del conductor. A nivel biomecánico, el cuerpo es un mecanismo complejo que se deberá analizar como una estructura a la que afectan ruidos y vibraciones.

De forma general, el cuerpo humano, constituido por miembros con masa y elasticidad, se comporta como un sistema vibratorio. Por otra parte, al estar formado también por elementos viscoelásticos, amortigua las vibraciones generales originadas por acciones exteriores.

El sistema muscular reacciona continuamente a las solicitaciones generadas por las vibraciones, adaptándose a ellas y compensando muscularmente las cargas, lo que acaba produciendo cansancio físico. Por otro lado, los desplazamientos que se producen en el viajero dentro del vehículo pueden, en general, dar lugar a un determinado grado de incomodidad y, en el caso del conductor, provocar la falta de atención o limitar su capacidad de manejo de los sistemas de control del vehículo.

La disminución del confort debido a las vibraciones mecánicas se denomina Incomodidad Cinética vibratoria (ICV), cuyos límites son difíciles de establecer por depender del nivel de sensibilidad de cada persona, por tanto, el nivel vibracional o confort de marcha del pasajero es una percepción subjetiva. Básicamente, puede decirse que no existe un estándar absoluto de confort o incomodidad humana expresada en términos físicos tales como amplitudes o aceleración a una determinada frecuencia. No obstante, sí hay una concordancia suficiente entre los datos obtenidos de los experimentos realizados por diferentes investigadores como para poder definir una zona por encima de la cual la vibración es intolerable y por debajo de la cual es insignificante.

El cabeceo produce sensación de náuseas y alteraciones en el laberinto auditivo que modifican el sentido del equilibrio. Si el aparato vestibular y el líquido coclear del oído interno es sometido de forma continua a aceleraciones lineales y/o angulares de frecuencias entre 0,5 y 0,75 Hz, se produce vértigo y mareo. La sensibilidad de algunos individuos o determinadas circunstancias pueden extender este valor hasta 1 Hz. La frecuencia para la sensibilidad máxima, así como el tiempo necesario para que comiencen a aparecer los síntomas dependen también del individuo. Las frecuencias de 5-6 Hz causan fatiga general, debida a la resonancia de los músculos. Los objetos de la región visceral se ven afectados por frecuencias entre 5 y 7 Hz. La frecuencia depende del individuo y de la amplitud de la vibración; en general mayores amplitudes aumentan la frecuencia a la que el individuo se hace sensitivo. Otra indicación del efecto de la sensibilidad de la región visceral es que, para un nivel de tolerancia dado, puede permitir a 6 Hz sólo el 0,7 de la aceleración a 15 Hz, dando la misma incomodidad. Aun así, la amplitud de las vibraciones

senoidales dada para esta relación de la aceleración es, a 6 Hz, 4,36 veces la que hay a 15 Hz. Del mismo modo, la entrada en resonancia del diafragma (4-8 Hz) o la cara frontal del tórax (10-50 Hz) produce la consiguiente aparición de dificultades respiratorias. La cabeza y el cuello son muy sensibles a vibraciones que varían entre los 18 y 20 Hz, y las frecuencias del orden de 20 Hz son perjudiciales para las vértebras cervicales. Otras frecuencias de resonancia para otras partes del cuerpo son:

- Pierna flexionada (sentado): 2 Hz
- Pierna rígida: 20 Hz
- Torso superior (Hombro): 4-5 Hz
- Antebrazo: 5-10 Hz
- Columna vertebral (axial): 10-12 Hz
- Brazo: 16-30 Hz
- Mano: 30-50 Hz globo ocular: 20-90 Hz

La vibración transmitida al globo ocular produce una pérdida de agudeza en la visión. Así, una oscilación en la cabeza de 0,1 mm a una frecuencia de 60 Hz produce una pérdida de agudeza visual de 20 minutos de arco, que supone unos 60 cm de indefinición visual a 100 m.

En general, parece que las frecuencias verticales que resultan más incómodas para las personas se encuentran entre 20 y 200 Hz (aunque la fatiga aparece más rápidamente cuando las vibraciones están entre 4 y 8 Hz) o por debajo de 0,75 Hz, y en ellas pueden aparecer vértigo y mareo. Las frecuencias laterales o longitudinales en el mismo rango son también molestas porque alteran el mecanismo de equilibrio del oído interno. Las frecuencias naturales del sistema de suspensión se diseñan normalmente para que estén en el medio rango de frecuencias aceptables, aunque el análisis total de las vibraciones del vehículo contiene normalmente elementos de frecuencias superiores, pero de menor amplitud, derivados de la resonancia de los neumáticos (50-100 Hz) y de la elasticidad de elementos elásticos (15-50 Hz).

Finalmente se concluye que el campo en el cual las vibraciones son aceptables está restringido a frecuencias comprendidas entre 1 y 2 Hz, jugando el asiento un papel fundamental en la sensación de confort, cuyos muelles deben tener sus frecuencias naturales alejadas de las frecuencias de las oscilaciones transmitidas a la carrocería por las suspensiones, para evitar los fenómenos de resonancia.

Como nota final, se podría pensar que la meta de los ingenieros de automoción, en lo que respecta al confort, debería ser generalmente eliminar todas las vibraciones en el vehículo. Aunque esto no será nunca posible en un vehículo de motor, es una importante dirección de desarrollo. En este punto hay dos fenómenos contradictorios que se deben tener en cuenta. Por una parte, la eliminación de una vibración siempre producirá otra molestia menos importante. Por otra parte, en el límite, la eliminación de toda la vibración es también indeseable, ya que las vibraciones son un modo de percibir el terreno y por ello un factor muy importante a considerar por el conductor de un vehículo. Las vibraciones tanto permanentes (las inducidas por el motor, desequilibrios, etc.), como transitorias (frenadas, baches, etc.) inducen sobre el ocupante del vehículo oscilaciones que pueden mantenerse en límites aceptables con una adecuada suspensión y amortiguamiento en los dispositivos en los que se sienta el viajero.

3. EL SISTEMA DE SUSPENSIÓN.

3.1. Concepto y necesidad de la suspensión.

La suspensión de un vehículo es el conjunto de órganos y piezas que se encargan de absorber y amortiguar las irregularidades que se producen al circular, lo que evita que las oscilaciones que se originan en las ruedas debido a las distintas condiciones de marcha se transmitan a los ocupantes del vehículo.

La suspensión intenta mantener en todo momento el contacto de las ruedas con el suelo para cualquier condición de marcha. De este modo se logra mejorar l guiado y la adherencia de las ruedas, lo que mejora el confort y la estabilidad del vehículo.

La suspensión está formada por los tres grupos de piezas siguientes:

- Componentes elásticos.
- Elementos de amortiguación y estabilidad.
- Componentes de fijación y guiado.

3.1.1. **Concepto de masas no suspendidas.**

La masa del automóvil se divide en dos desde el enfoque de la suspensión:

- **Masa no suspendida:** está compuesta por las ruedas y todos sus elementos anexos, es decir, manguetas, bujes, elementos de suspensión, frenos, etc. y en general, todo lo que oscila con las ruedas.
- **Masa suspendida:** hace referencia al resto del vehículo, es decir, todos los elementos no incluidos en la masa suspendida (motor, carrocería, etc.). Se cumple que cuanto menor sea el peso de las masas no suspendidas, mejor será el funcionamiento del sistema de suspensión.

3.1.2. **Funciones de la suspensión.**

El sistema de suspensión en un vehículo tiene las siguientes funciones:

- Función de mejora del confort.

  La primera misión que cumple el sistema de suspensión es la de evitar, en la medida de lo posible, que las irregularidades del terreno por el que el vehículo circula se transmitan al mismo, mejorando así la comodidad de los ocupantes, al mismo tiempo que se optimiza la marcha del propio vehículo. También ayudan a mejorar el confort los neumáticos y los asientos.

- Función de protección del vehículo.

  Al impedir que se transmitan íntegramente las irregularidades del terreno, las cuales acabarían resintiendo la carrocería o chasis, así como a otros elementos del vehículo. De este modo se puede diseñar una carrocería menos resistente, y por tanto más liviana. Aquí también cumplen

un papel trascendental los neumáticos, al absorber las irregularidades más pequeñas, simplificando el trabajo de la suspensión.

- Función estabilizadora.

La cual se lleva a cabo al absorber gran parte de la fuerza centrífuga que se genera durante los cambios de dirección, evitando, mediante la deformación de los elementos elásticos, que dicha fuerza se transmita íntegramente a las ruedas. Ello permite descargar de trabajo a los neumáticos, elevando el límite de adherencia y evitando los balanceos. En esta labor la suspensión se ve respaldada por un elemento auxiliar, no siempre presente, denominado barra estabilizadora. Esta función se contrapone con la de mejora del confort, ya que ambas dependen del grado de dureza o tarado dispuesto en los elementos elásticos. Así, cuanto más blandos sean estos más confortable será el vehículo, en detrimento de su estabilidad. Por el contrario, una suspensión con elementos elásticos de gran dureza proporcionará una gran estabilidad en curvas, al poseer un límite muy alto de absorción de fuerzas, mientras que el confort se verá resentido al no filtrarse las irregularidades de pequeño y mediano tamaño.

- Asegurar el contacto permanente de las ruedas con el suelo.

El hecho de que la carrocería y las ruedas puedan aproximarse o alejarse entre sí, permite que éstas se adapten a las diferencias de relieve del pavimento, de tal manera que siempre estén en contacto con el mismo. Si el vehículo no dispusiese de suspensión, a poco que el terreno estuviese desnivelado, alguna rueda dejaría de estar en contacto con el suelo.

En el momento en que una rueda deja de estar en contacto con el suelo, sus funciones de transmisión de esfuerzos de frenado y tracción, así como las de direccionabilidad, quedan totalmente anuladas.

- Mejorar la direccionabilidad.

Esto se cumple al asegurarse el contacto de las ruedas con el suelo, permitiendo mantener su función directriz.

3.2. Componentes del sistema de suspensión.

3.2.1. **Elementos elásticos.**

Es el componente, con cuya deformación se permiten los movimientos relativos entre la rueda y la carrocería. Absorbe también la energía producida por los impactos generados por las irregularidades del terreno, así como por la fuerza centrífuga que surge al abordar el vehículo una curva. Dicha energía es devuelta posteriormente en el movimiento de extensión, al recuperar las ruedas su posición inicial.

Se considera que un elemento elástico es de **tarado blando**, cuando la fuerza a ejercer para deformarlo no es muy alta, sobre todo si se compara con otro de **tarado duro**, el cual precisa de un mayor esfuerzo para ser deformado.

3.2.1.1. Muelle helicoidal.

Es el elemento elástico utilizado con mayor frecuencia en los sistemas de suspensión. este componente enlaza la carrocería con la rueda adoptando diferentes formas de montaje.

El muelle helicoidal, está formado por un hilo de acero enrollado en espiral en torno a un eje imaginario y teórico, trabajando a torsión. En los extremos se dispone de forma aplanada, para facilitar su asentamiento en sus respectivos alojamientos. Las características del muelle, de las cuales la dureza (resistencia a ser deformado) es la más importante, dependen de los siguientes factores:

- Longitud.

  Un muelle es más blando cuanto más largo es, siempre y cuando el resto de las variables sean las mismas.

- Diámetro de las espiras.

  El cual no hay que confundir con el del hilo. También puede definirse como el diámetro exterior del muelle, siempre y cuando no sea cónico. La dureza disminuye a medida que se incrementa el diámetro de las espiras, siempre que no se varíe el diámetro del hilo.

- Coeficiente elástico del acero empleado en su fabricación.

  El cual determina la fuerza necesaria para deformarle en la unidad de longitud. Cuanto más alto sea, mayor será la dureza, comparado con otro cuyas restantes variables sean similares.

- Gradiente.

  Es la distancia entre dos espiras consecutivas. Su influencia en la dureza se establece en proporción inversa, de tal forma que cuanto mayor sea dicha distancia, menor será el esfuerzo a efectuar para deformarlo. También determina la inclinación de las espiras, ya que a mayor inclinación (respecto a la perpendicular a su eje imaginario), mayor separación entre espiras, y por tanto mayor gradiente.

- Diámetro del hilo.

  Su influencia en la dureza se estable en proporción directa, de tal forma que la fuerza a ejercer para deformarlo es mayor a medida que aumenta su sección.

Figura 1. Muelles helicoidales (Esteban José Domínguez, Julián Ferrer: "Mecánica del vehículo" (2011)).

– Comportamiento en el vehículo.

El tarado de los muelles influye sobre la amplitud y frecuencia de las oscilaciones. Así pues, un muelle blando realiza oscilaciones de mayor recorrido pero muy espaciadas en el tiempo, es decir, de poca frecuencia. Por el contrario, un muelle duro poseerá una mayor frecuencia, ya que los ciclos de oscilaciones se sucederán más rápidamente, pero éstas poseerán menor recorrido. Cuando el vehículo circula muy cargado, el muelle se comporta igual que uno de tarado más blando, por tanto, la frecuencia es menor, pero las oscilaciones son de mayor recorrido. Por el

contrario, cuando el vehículo circula vacío, las oscilaciones son más frecuentes, pero de menor recorrido.

– Muelles de progresividad variable.

Combinan el confort de los muelles blandos y la estabilidad de los duros siendo, por tanto, más adecuados para las exigencias dinámicas del vehículo que los de paso fijo. Existen dos variantes:

- Muelles de paso variable.

  El gradiente o paso varía a lo largo de su longitud. La zona en la que las espiras están más separadas trabaja en la primera parte del recorrido, absorbiendo los impactos más pequeños. Por otra parte, la zona en la que el gradiente es menor es la encargada de absorber los impactos o balanceos más acusados después de haberse comprimido la parte más blanda.

- Muelles cónicos.

  Su disposición permite conseguir resultados similares a los conseguidos con los de paso variable. Así, las espiras de mayor diámetro son las encargadas de absorber los impactos menores, mientras que las de menor diámetro absorben las cargas más acusadas, disponiéndose por tanto de un muelle de dureza variable.

– Aptitudes de los muelles como elementos elásticos.

- Gran facilidad de ubicación, ya que pueden ir montados en posición horizontal, vertical e inclinada, lo cual los hace muy útiles para su empleo en sistemas independientes.
- Idoneidad para su empleo en el tren delantero, al no condicionar el giro de las ruedas directrices en su orientación.
- Poseen una gran sencillez constructiva, lo cual repercute en su ligereza y efectividad.
- Por el contrario, carecen de falta de rigidez vertical, lo cual obliga a disponer elementos de guiado a la rueda en sus movimientos oscilatorios.

3.2.1.2. Esferas y balonas neumáticas.

Las esferas se montan en las suspensiones hidroneumáticas y disponen de una cámara de gas nitrógeno a presión separada por una membrana del líquido del circuito.

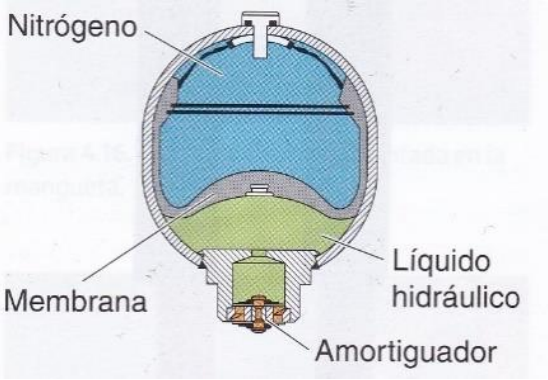


Figura 2. Esfera y amortiguador. (Esteban José Domínguez, Julián Ferrer: "Mecánica del vehículo" (2011)).

Las balonas son recipientes estancos fabricados en caucho que permiten dilatarse con la presión del aire y deformarse cuando soportan una carga.

Figura 3. . Balona. (Esteban José Domínguez, Julián Ferrer: "Mecánica del vehículo" (2011)).

Los elementos neumáticos presentan múltiples ventajas:

- Facilidad para variar su grado de dureza, al modificarse la presión del aire contenido en su interior, de tal forma que a mayor presión, más dureza.
- Curva de flexibilidad variable y progresiva, ya que el aire, cuanto más comprimido, más esfuerzo requiere para seguir comprimiéndolo, aumentando por tanto su tarado en la parte final del recorrido. Su eficacia en este aspecto, está muy por encima de la de cualquier muelle de progresividad variable.

3.2.1.3. Ballestas.

Están formadas por una serie de hojas o láminas de acero (y ocasionalmente fibra) con propiedades elásticas, dispuestas con longitud decreciente a partir de la hoja principal, denominada **hoja maestra**. Las ballestas van sujetas a la carrocería o chasis por medio de la hoja maestra, para lo cual ésta posee sus dos extremos curvados formando un orificio u ojo donde se disponen unos bulones montados sobre casquillos de bronce o silentblocks para su fijación al chasis, y además permiten la pivotación del conjunto, al flexionar la hoja como consecuencia de la absorción de alguna carga o irregularidad. Para compensar las variaciones de longitud que experimenta la hoja maestra al deformarse, en uno de los extremos se dispone una fijación articulada denominada **gemela**.

La fijación de las hojas se realiza mediante dos abrazaderas, las cuales mantienen unidas todas las hojas permitiendo el deslizamiento entre las mismas al deformarse. Estas abrazaderas son conocidas como **abarcones**. A su vez, por su parte central, van unidas por un perno pasante que une todas las hojas en sentido perpendicular a las mismas, el cual recibe el nombre de **capuchino**.

Las ballestas se pueden clasificar en:

- **Semielípticas:** Su forma posee cierta curvatura en forma semielíptica, y están formadas por hojas planas de espesor constante. Las hojas están en contacto unas con otras, lo que provoca que aparezca fricción entre ellas, frenando sus oscilaciones. Esta fricción provoca una acción de soldadura entre las hojas, y por tanto es necesaria su constante lubricación. Para ello se dispone una capa de grasa blanda entre las hojas, que impide los rozamientos excesivos y, principalmente la oxidación de las hojas.
- **Parabólicas:** Las hojas se disponen en menor número (dos o tres) que en la semielípticas, estando unidas entre sí en los extremos, así como en su parte central (mediante el capuchino

y los abarcones), no teniendo contacto en las zonas comprendidas entre ambas partes, para lo cual disponen de una forma especial con doble curvatura que permite dicha disposición. Son más sencillas y ligeras. Además, presentan mayor grosor en la parte central, siendo más delgadas en los extremos, consiguiéndose así una curvatura más uniforme en toda su longitud.

- **Asimétricas**: Es un montaje especial con el capuchino descentrado.

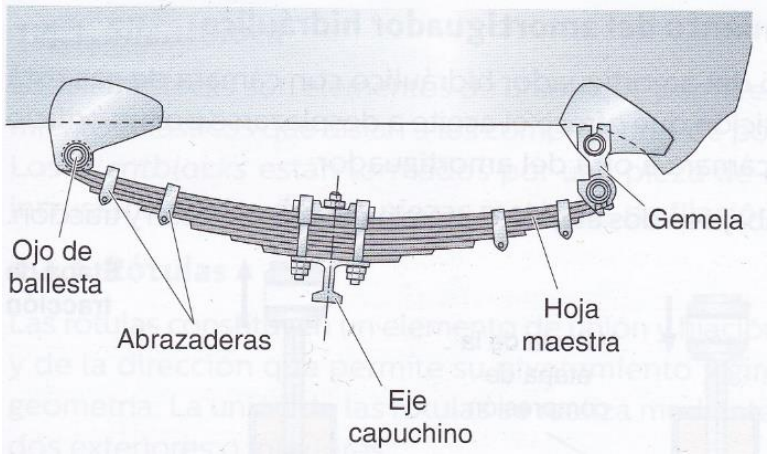


Figura 4. Ballesta y fijación al chasis. (Esteban José Domínguez, Julián Ferrer: "Mecánica del vehículo" (2011)).

Las ballestas presentan las siguientes ventajas como elementos elásticos:

- Sólo se deforman en sentido vertical, lo cual permite prescindir de elementos de guiado, siendo las propias ballestas las que cumplen la citada función de guiado y sujeción del eje.
- Son económicas, fáciles de fabricar y de reparar, al admitir la posibilidad de sustituir la hoja u hojas dañadas, así como volver a dar curvatura a las mismas.
- Se puede modificar su tarado, añadiendo o quitando hojas.
- El rozamiento entre las hojas dota al conjunto de un cierto efecto amortiguante.

Por el contrario, presentan los inconvenientes que a continuación se presentan:

- Son pesadas.
- Precisan de un mantenimiento periódico, ya que es necesario engrasar las hojas para evitar su oxidación y agarrotamiento.
- Por su disposición constructiva, no permiten grandes recorridos de suspensión.

En un vehículo las ballestas se pueden disponer en sentido longitudinal o transversal:

- **Montaje longitudinal**: es el más utilizado en la mayoría de los vehículos industriales. Se realiza montando la ballesta con un punto "fijo" en la parte delantera de la misma (según el desplazamiento del vehículo) y otro "móvil", para permitir los movimientos oscilantes de la misma cuando se deforma con la reacción del bastidor. Plantea el inconveniente de limitar la capacidad de giro de las ruedas directrices en su orientación.

- **Montaje transversal**: son habituales en ciertos vehículos todo terreno, y se utiliza una sola ballesta por eje. En este caso se anula la ventaja que poseen las longitudinales en cuanto a capacidad de guiado, siendo necesario disponer de brazos de suspensión, a los cuales por otro lado va unida la hoja maestra en sus extremos.

La firmeza de las ballestas viene dada sobre todo por el grosor y número de las hojas en proporción directa, así como por la elasticidad del acero empleado en su fabricación. Por el contrario, su longitud influye sobre la firmeza en proporción inversa, ya que aumenta el brazo de palanca, generándose un par o momento de fuerza superior, para una misma carga a soportar. El número de hojas de una ballesta viene determinado por el peso que ha de soportar, y también de la flexibilidad que se desee obtener

La suspensión por ballestas suele utilizarse en vehículos dotados de puente trasero rígido y eje delantero de la misma naturaleza.

3.2.1.4. Barras de torsión.

En este sistema de suspensión, la elasticidad se consigue mediante la torsión o revirado de una barra de sección circular, sujeta en uno de sus extremos al chasis o carrocería, y solidaria en el otro extremo a un elemento móvil como un brazo de suspensión. Por su propia elasticidad, cuando cesa la fuerza que la mantiene revirada, la barra recupera su posición original.

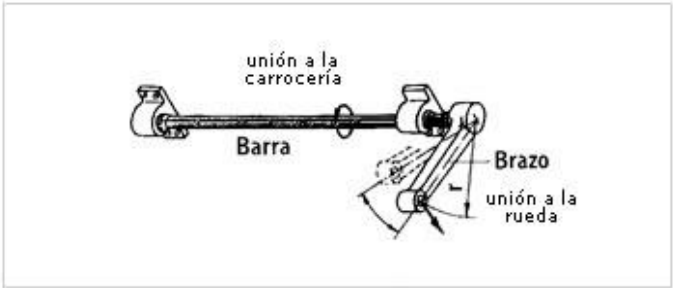


Figura 5. Barra de torsión. (www.aficionadosalamecanoca.net)

3.2.1.5. Barras estabilizadoras.

La barra estabilizadora funciona como una barra de torsión, cuya función es estabilizar la caja del vehículo frente a acciones que produzcan un movimiento de balanceo, por cuya causa se la llama también barra antibalanceo. Se constituye como una barra flexible de acero, doblada por sus extremos en ángulo recto aproximadamente. En el montaje se une por sus extremos a cada uno de los brazos inferiores del sistema de suspensión, mientras que en el tramo recto central se fija a la carrocería por medio de casquillos de caucho y abrazaderas.

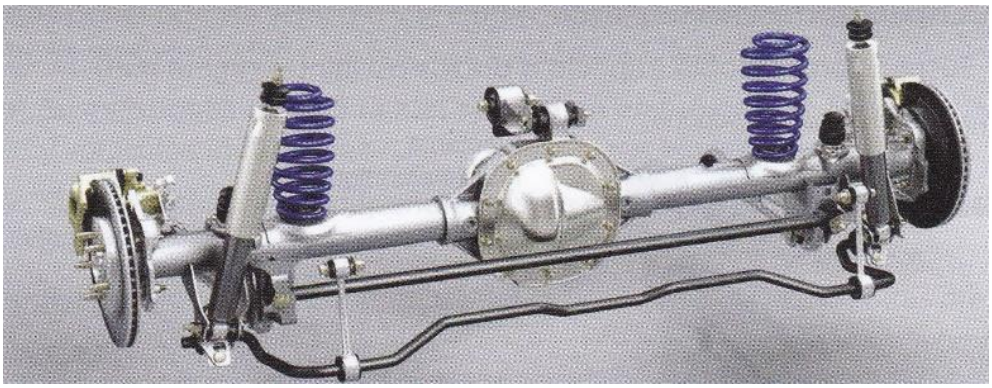


Figura 6. Montaje de la barra estabilizadora en un puente trasero rígido. (Esteban José Domínguez, Julián Ferrer: "Mecánica del vehículo" (2011)).

3.2.2. **Amortiguadores.**

La misión de los amortiguadores es neutralizar las oscilaciones de la masa suspendida originadas por el elemento flexible (muelles, ballestas, barras de torsión), al adaptarse a las irregularidades del terreno, convirtiendo en calor la energía generada por dichas oscilaciones.

Estos movimientos oscilantes, de no ser eliminados, provocarían en el vehículo movimientos desestabilizadores que repercutirían gravemente en la seguridad activa del mismo y de sus ocupantes. Las ruedas, en sus movimientos oscilantes, perderían el contacto con el pavimento, lo cual traería consigo las siguientes desventajas:

- Falta de direccionalidad.
- Incremento de las distancias de frenada.
- Disminución del confort.

Estas desventajas se resuelven con el uso de amortiguadores.

Los amortiguadores empleados en automoción son los telescópicos, de tipo hidráulico, que basan su funcionamiento en la resistencia que ofrece todo líquido viscoso al paso por un orificio. Dicho líquido resulta ser un aceite de tipo mineral (aunque también se utiliza el sintético en modelos de rango superior) con aditivos antiespumantes.

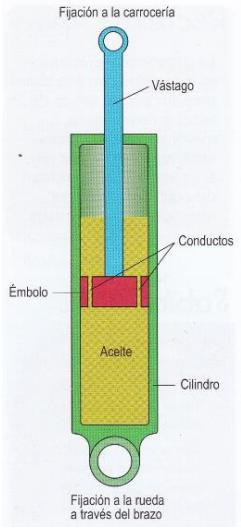


Figura 7. Amortiguador telescópico monotubo.(Miguel Ángel Pérez Belló "circuitos de Fluidos. Suspensión y Dirección (2011)).

Los amortiguadores hidráulicos telescópicos pueden ser de doble tubo o monotubo, de simple o doble efecto, siendo el amortiguador bitubo de doble efecto el más común.

- **Amortiguadores bitubo**: su estructura es similar a los monotubo, salvo que en los bitubo se añade una tercera cámara o cámara exterior, alrededor de la inferior, concéntrica con la misma, rodeándola, y comunicándose ambas mediante unas válvulas unidireccionales ubicadas en la parte inferior de la cámara para solventar el inconveniente de las diferentes variaciones de volumen. Así, durante la compresión, al ser mayor el volumen desalojado por la cámara inferior, por no contar con la presencia del vástago que limita su sección, el aceite que no puede pasar a la cámara superior, por ser menor el volumen desalojado,

pasa a la cámara exterior. Durante la extensión, el proceso se invierte, llenándose la cámara inferior con el aceite procedente de las otras dos cámaras.

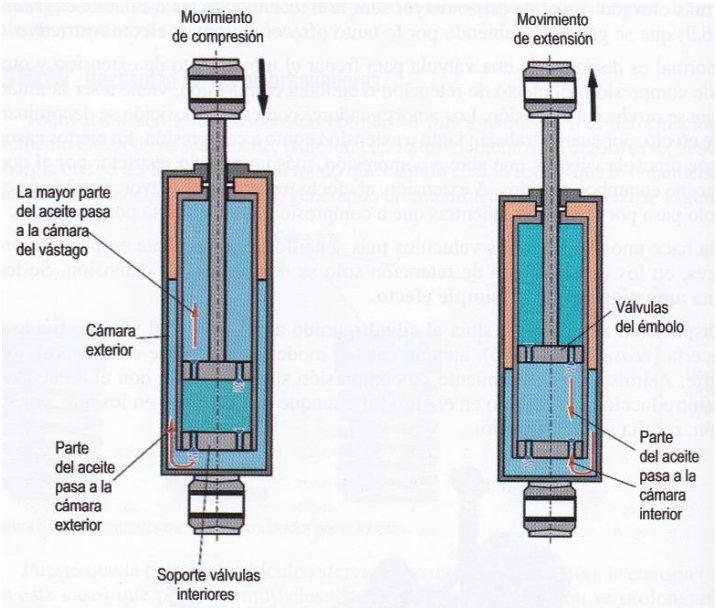


Figura 8. Amortiguador bitubo. (Miguel Ángel Pérez Belló "circuitos de Fluidos. Suspensión y Dirección (2011)).

- **Amortiguadores monotubo**: en ellos se dispone un **cilindro** lleno de aceite, por cuyo interior se desliza un **émbolo** solidario a un **vástago**, que divide el cilindro en dos cámaras. El conjunto formado por el cilindro, el émbolo y su vástago, constituyen la unidad telescópica. Dicho émbolo dispone de unos **orificios**, controlados por **válvulas** que regulan el paso del aceite de una cámara a otra. En los amortiguadores telescópico se forman dos cámaras a ambos lados del émbolo, pero con la salvedad de que las variaciones de volumen no se producen por igual a ambos lados. En efecto, la cámara dispuesta en el lado del vástago, presenta menores variaciones de volumen, ya que el espacio ocupado por el mismo hace que disminuya su sección útil. La otra cámara, al no disponer del vástago, presenta mayor sección útil, por lo que para un mismo recorrido del émbolo, la variación de volumen es mayor.

- **Amortiguadores de gas**: Su disposición y principio de funcionamiento es similar a los amortiguadores hidráulicos de tipo telescópico monotubo. La diferencia radica en que, en los amortiguadores de gas, se dispone un émbolo flotante en la cámara inferior, que divide ésta en dos. En la parte inferior del émbolo flotante, se dispone un depósito de gas, generalmente nitrógeno, mientras que, en la parte superior, se dispone el aceite. Al disponerse el émbolo de forma flotante, la presión del depósito de gas se transmite al aceite, lo cual aporta las siguientes ventajas:

  - Absorbe las variaciones de volumen en la cámara inferior, resultantes de los distintos volúmenes desalojados por el émbolo.

  - Al encontrarse el aceite sometido a presión, se evita la formación de burbujas, las cuales traen consigo comportamientos irregulares. Ello redunda en un comportamiento mucho más estable del amortiguador. A largo plazo, la ausencia de burbujas o espuma alarga la vida del aceite, al no existir aire que lo oxide.

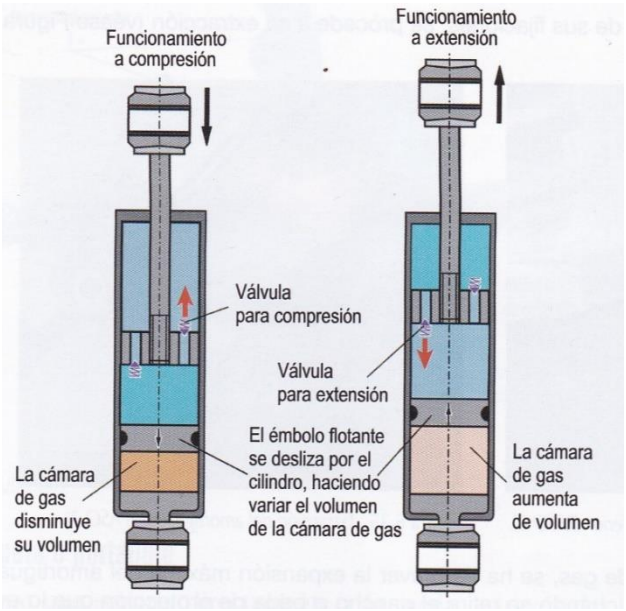


Figura 9. Amortiguador de gas. (Miguel Ángel Pérez Belló "circuitos de Fluidos. Suspensión y Dirección (2011)).

- **Amortiguadores de doble efecto**: suelen ser los más empleados, y frenan el muelle tanto a extensión como a compresión. Disponen de válvulas unidireccionales, de tal forma que el aceita pasa por distintos orificios en la compresión y en la extensión. El empleo de válvulas unidireccionales permite disponer orificios de distinto diámetro para extensión y compresión, lo cual tiene la ventaja de poder independizar el tarado o efecto de retención en ambas circunstancias, siendo siempre más destacado el efecto de retención a extensión que a compresión.
- **Amortiguadores de simple efecto**: sólo disponen de válvulas de retención a extensión, siendo éstas de grandes dimensiones que apenas ralentiza el paso del aceite a compresión. Son muy poco utilizados hoy en día. Cuando el muelle se dispara, después de la compresión sufrida por el paso de la rueda por un obstáculo, el amortiguador frena ese disparo sobre todo y, posteriormente, las oscilaciones seguidas por el muelle.

3.2.3. **Elementos estructurales.**

Son aquellos que sirven de sujeción y guiado a la rueda en su desplazamiento, posibilitando el funcionamiento de la suspensión y la amortiguación, e interrelacionándolos con los restantes dispositivos de tracción (o propulsión) y dirección. Sirven también de soporte y fijación a componentes de los sistemas citados, así como entre ellos mismos.

3.2.3.1. Trapecios.

Sirven de soporte y fijación a la mangueta, a la cual van unidos a través de la rótula, además sirven de guía en sus recorridos oscilantes al conjunto mangueta-rueda. Según el sistema de suspensión empleado se pueden disponer uno o dos por rueda. la fijación y pivotamiento de los trapecios a la carrocería se efectúa a través de dos articulaciones, bien directamente, o bien a través del subchasis, realizándose siempre, en cualquier caso, a través de silentblocks. Algún modelo deportivo emplea rótulas uniball.

Cuando poseen una forma triangular también son conocidos como triángulos de suspensión. debido a su doble punto de anclaje al bastidor, no sólo cumple funciones de guiado de la rueda en sus oscilaciones de suspensión, sino que también proporciona sujeción al conjunto, evitando disponer tirantes de reacción que impidan las oscilaciones longitudinales del mismo. Actúa también por tanto como un elemento resistente y no sólo de guiado.

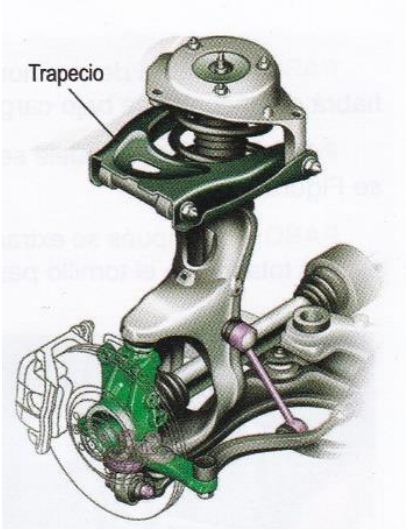


Figura 10. Trapecio. (Miguel Ángel Pérez Belló "circuitos de Fluidos. Suspensión y Dirección (2011)).

3.2.3.2. Brazos.

Cumplen una función parecida a la de los trapecios, diferenciándose de los mismos en su arquitectura, ya que sólo poseen un punto de anclaje a la carrocería. En el eje delantero sólo sirven de guiado a la rueda en sus desplazamientos propios de la suspensión, no impidiendo las oscilaciones longitudinales, que han de ser contrarrestadas con la disposición de un tirante de reacción. En el eje trasero además de cumplir con funciones de guiado, también cumplen funciones de sujeción del conjunto de la suspensión.

3.2.3.3. Tirantes.

Son un complemento a los brazos de suspensión, con los que se consigue una sujeción y un guiado efectivo de la rueda. con su disposición se evitan alteraciones en la geometría de la misma y sus ángulos. En concreto, se montan conjuntamente con los brazos cuando éstos se disponen transversalmente, sobre todo en eje delantero. También sirven de complemento a los sistemas multibrazo, sobre todo cuando se disponen en el eje trasero y sus ruedas son motrices. Suelen tener forma tubular, fabricándose en acero.

3.2.3.4. Mangueta y buje.

La mangueta de la suspensión es una pieza fabricada con acero o aleaciones que une el buje de la rueda y la rueda a los elementos de la suspensión, tirantes, trapecios, amortiguador, etc. La

mangueta se diseña teniendo en cuenta las características geométricas del vehículo. En el interior del buje se montan los rodamientos o cojinetes que garantizan el giro de la rueda.

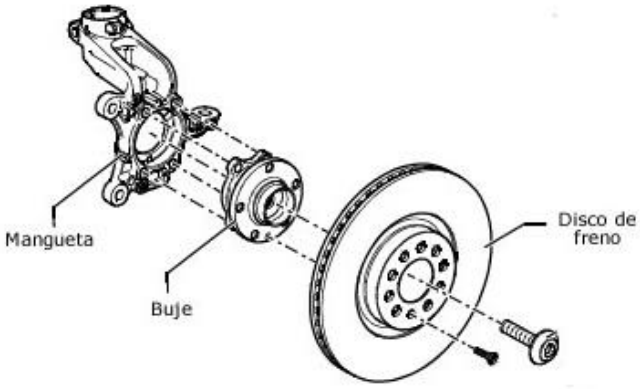


Figura 11. Mangueta y buje (www.aficionadosalamecanica.net).

3.2.3.5. Rótulas, articulaciones y silentblocks.

Sirven como elemento de unión, y forman parte de las articulaciones que transmiten el movimiento de los sistemas de dirección y suspensión

- **Rótula**: constituyen un elemento de unión y fijación de la suspensión y de la dirección que permiten su pivotamiento y giro manteniendo la geometría de las ruedas.
- **Silentblocks:** son elementos de unión que permiten pequeños movimientos elásticos que aíslan a los componentes de posibles vibraciones. Están formados por una pieza de plástico incrustada dentro de dos piezas metálicas de fijación.

3.3. Sistemas de suspensión.

Generalmente, las suspensiones se clasifican en dos grandes grupos: suspensiones de **eje rígido** o dependiente y suspensiones **independientes**. Existe una tercera tipología denominada **semiindependiente**, por ser funcionalmente intermedia entre las dos anteriores.

3.3.1. Suspensiones de eje rígido.

Una suspensión de eje rígido es aquella en la que las ruedas se unen a los extremos de una barra o elemento rígido, pudiendo girar independientemente, aunque cualquier otro movimiento de una de las ruedas se transmite a la rueda opuesta, haciendo que ambas señalen en la misma dirección y tengan la misma caída. La suspensión mediante eje rígido motriz se utiliza en el eje trasero de muchos vehículos turismos antiguos, vehículos turismos americanos y en la mayoría de los camiones, mientras que en el eje delantero se utiliza en vehículos con tracción a las cuatro ruedas y camiones pesados, sin que se encuentre en turismos. Este sistema está cada vez más en desuso.

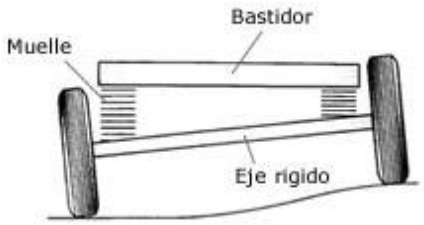


Figura 12. Suspensión Rígida (www.aficionadosalameca-nica.net

Básicamente está formado por un eje transversal, construido en chapa de acero si el eje en que se monta no es motriz, o por un tubo hueco, en cuyo interior giran los palieres, en caso de que sí lo sea. En este último caso, también contiene al mecanismo grupo-diferencial, elevándose ostensiblemente el peso del conjunto.

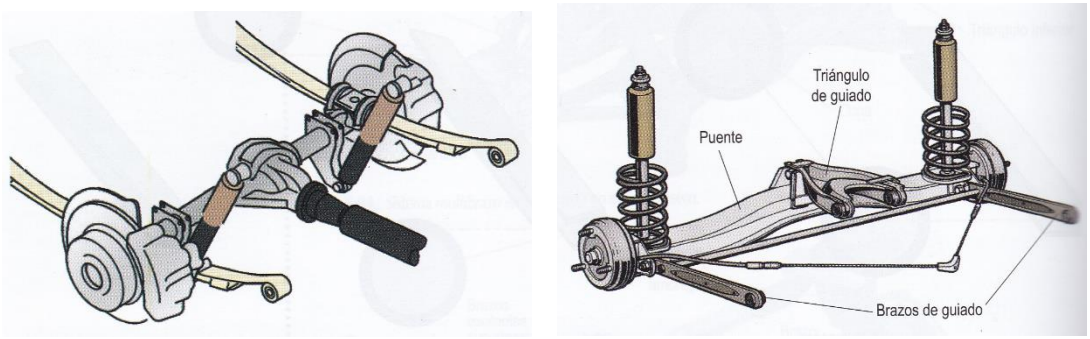


Figura 13. Suspensión rígida. (Miguel ángel Pérez Belló "Circuitos de Fluidos. Suspensión y Dirección (2011)).

Si emplea ballestas como elemento elástico de la suspensión, no suele disponer elementos de guiado tales como tirantes o brazos, ya que las mismas bastan para realizar dicha función, si bien son de tipo parabólico, dada su menor resistencia transversal, sí suelen disponer de alguno. En caso de emplear muelles helicoidales, sí se hace necesario disponer de unos brazos de guiado y pivotamiento. En vehículos pesados, cada vez se utilizan más los elementos elásticos de tipo neumático.

En turismos, su empleo quedó apartado del eje delantero hace ya muchas décadas, empleándose tan sólo, y cada vez menos, en el eje trasero.

Algunas ventajas del eje rígido sobre otros sistemas son:

- Capacidad para mantener siempre las ruedas en contacto con el suelo.
- Cuando se emplea en el eje motriz, ofrece un perfecto guiado de los palieres, pudiendo prescindir del empleo de juntas de articulación en los mismos.
- Robustez y sencillez.

En cambio, presenta los siguientes inconvenientes:

- Posee un elevado peso del conjunto, lo cual condiciona en gran medida el trabajo del sistema de suspensión.
- Desaprovechamiento del espacio destinado al compartimento de carga, quedando la altura del maletero muy menguada.

- Menor grado de confortabilidad.
- Menor estabilidad y adherencia del vehículo en curvas, siendo el comportamiento especialmente crítico cuando el pavimento está mojado.
- Cuando el eje es motriz, es muy susceptible de sufrir vibraciones en la dirección.
- Transmiten los movimientos y vibraciones de una de las ruedas a la otra.

3.3.2. **Suspensiones independientes.**

En lo que respecta a las suspensiones independientes, éstas permiten oscilar verticalmente las ruedas de cada eje por separado, sin que el movimiento de una afecte a la otra, adaptándose a las diferentes condiciones del pavimento. Casi todos los turismos y los camiones pequeños utilizan algún tipo de suspensión delantera independiente. Se aplican tanto a ejes anteriores como posteriores.

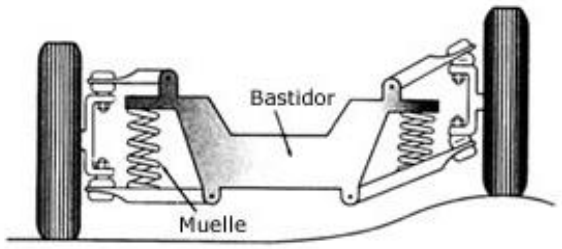


Figura 14. Suspensión independiente (www.aficionadosalamecanica.net)

Las suspensiones independientes presentan las siguientes ventajas respecto a alas de eje rígido:

- Ocupan menos espacio.
- Mejor resistencia a vibraciones de la dirección.
- Menor masa suspendida.
- Mayor rigidez al balanceo para una misma elasticidad del conjunto de la suspensión.

A continuación, se comentan las configuraciones aplicables más comúnmente utilizadas tanto para ejes anteriores como posteriores.

- **Semiejes oscilantes.**

  Esta configuración destaca por su gran sencillez, ya que la rueda está unida por un único brazo transversal. La suspensión de semiejes oscilantes apareció con la idea de conseguir una suspensión trasera independiente lo más sencilla posible, que permitiera que el cubo de salida de la transmisión y el brazo portante de la rueda permanecieran coaxiales ante las alteraciones de la carretera.

  La longitud de los semiejes era generalmente algo menor que la mitad del ancho de vía del vehículo, y el centro de balanceo solía estar por encima del eje de giro de los brazos, por lo que en curva la transferencia de carga era muy elevada. Si bien esta transferencia de carga no era necesariamente un inconveniente, los cambios en la caída de las ruedas en curva suponían que la rueda más cargada adquiría un ángulo de caída positivo, lo que reducía significativamente la potencia disponible en curva, en especial con neumáticos anchos. Uniendo esto a los efectos negativos que producían los pares giroscópicos generados, este

tipo de suspensión fue abandonado por sistemas de suspensión en los que la rueda más cargada pudiera adoptar ángulos negativos de caída.

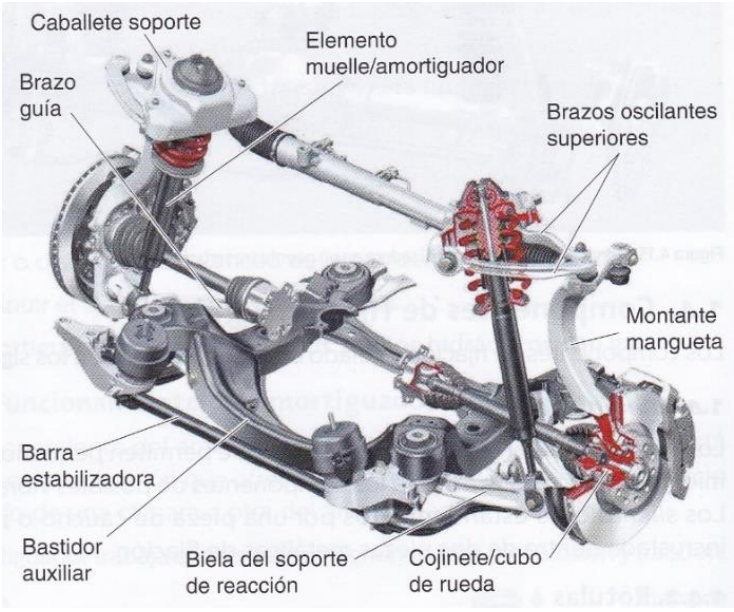


Figura 15. Suspensión de semiejes oscilantes. (Esteban José Domínguez, Julián Ferrer: "Mecánica del vehículo" (2011)).

- **Mcpherson.**

Es el sistema más empleado en turismos en el eje delantero. Está formado por una columna telescópica, en la que se integran conjuntamente el muelle y el amortiguador, situándose este último dentro del primero.

Los conjuntos Mcpherson ofrecen las siguientes ventajas:

- Compacidad y facilidad de desmontaje y montaje como conjunto.
- Simplicidad del sistema de suspensión al eliminarse componentes, ya que la función resistente de la columna hace que sólo sea necesario disponer un trapecio, o en su defecto, un brazo y un tirante.
- Idoneidad para su utilización en el eje delantero, dada la posibilidad de giro del conjunto, a la par que la rueda directriz.
- La geometría de la rueda apenas sufre variaciones durante el recorrido de suspensión.

La principal desventaja de las suspensiones Mcpherson es que las fuerzas de rozamiento, que se producen como consecuencia del deslizamiento de los elementos que hacen de guía en los amortiguadores, son elevadas, al ser estos elementos portantes.

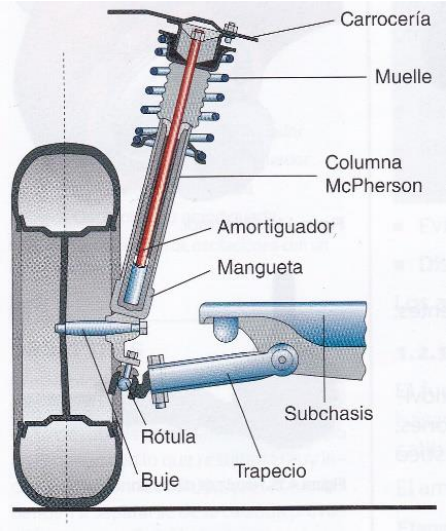


Figura 16. Suspensión Mcpherson. (Esteban José Domínguez, Julián Ferrer: "Mecánica del vehículo" (2011)).

❖ **Paralelogramo deformable.**

Sistema utilizado en el eje delantero casi exclusivamente, con el que se consigue un excelente guiado de la rueda durante su movimiento oscilatorio, sin apenas sufrir variaciones los ángulos que componen la geometría de la rueda, en especial, el más afectado en estos casos es el ángulo de caída.

Está formada por dos trapecios superpuestos paralelos entre sí, formando una estructura de paralelogramo deformable, siendo ligeramente más largo el inferior. Los dos van unidos a la mangueta a través de sendas rótulas, estableciéndose como eje de pivote el que une ambas

rótulas. El elemento elástico suele ser un muelle helicoidal, ubicado sobre cualquiera de los dos trapecios. El amortiguador puede ir separado o concéntrico con el muelle. La suspensión moderna de paralelogramo está formada por brazos desiguales y no paralelos.

Esta disposición es más sofisticada y cara de elaborar que la Mcpherson, si bien propicia un mejor comportamiento del vehículo, dado el superior guiado de la rueda.

Figura 17. Suspensión de paralelogramo deformable. (Miguel Ángel Pérez Belló "circuitos de Fluidos. Suspensión y Dirección (2011)).

❖ **Brazos tirados o arrastrados.**

Es un sistema casi exclusivo del eje trasero, empleado tan sólo en vehículos de tracción delantera. Cumple sólo funciones de sustentación, empleando unos brazos longitudinales de gran sección y robustez, con su eje de articulación dispuesto en sentido transversal. Permite prescindir de cualquier otro elemento de guiado como un tirante o un brazo.

Proporciona un buen guiado de las ruedas en sus movimientos oscilatorios en línea recta. Sin embargo, en curvas de gran apoyo, las ruedas exteriores adoptan un excesivo ángulo de caída. Esto hace que disminuya la huella del neumático, al tiempo que la desplaza de la zona central de la banda de rodadura, lo cual trae consigo una disminución de la adherencia. Como elemento elástico emplea los muelles helicoidales o las barras de torsión, siendo especialmente indicado para estas últimas.

Las principales desventajas de este sistema son:

- Los cambios en la caída y en la dirección que se producen cuando el vehículo toma una curva.
- Dificultad de regular de forma eficiente la dirección.
- Una cierta inestabilidad debida a que la combinación de la caída y la dirección sobre la rueda interior actúan contra la dirección de giro, imponiendo un comportamiento subvirador del eje trasero, aunque la aparición de una fuerza lateral contribuye a que se comporte como sobrevirador si no se controla.
- Dificultad de alcanzar el compromiso entre los efectos perturbadores traseros de hundimiento en aceleración y elevación en frenada.

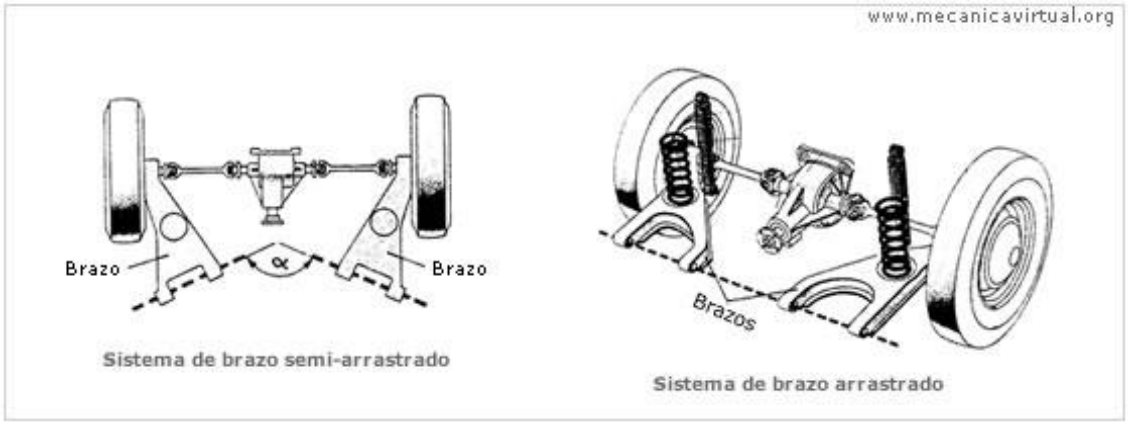


Figura 18. Suspensión de brazos arrastrados. (www.aficionadosalamecanica.com)

❖ **Suspensión Multibrazo o Multilink.**

Se utiliza en el eje trasero de los turismos de propulsión. Proporciona un eficaz guiado de las ruedas en sus movimientos oscilatorios, variando la geometría de las mismas haciéndolas adoptar los ángulos más idóneos para que estas cumplan sus funciones, al tiempo que absorbe todas las reacciones dinámicas que se generan en las mismas.

Disponen de hasta cinco elementos de guiado y sujeción, entre trapecios, brazos y tirantes, formando una estructura compleja, pero muy eficaz, sin la cual se presentarían serios problemas de motricidad y estabilidad, así como de confortabilidad.

Como elementos elásticos, suelen emplear en su mayoría los muelles helicoidales, si bien cada vez está más extendido el empleo de los elementos neumáticos, al disponerse éstos conjuntamente con las suspensiones de tarado variable pilotadas electrónicamente. Además, destaca por poseer una masa no suspendida relativamente baja. Este tipo de suspensiones tiene

el inconveniente de ser sistemas bastante complejos, así como de tener un coste de fabricación muy elevado.

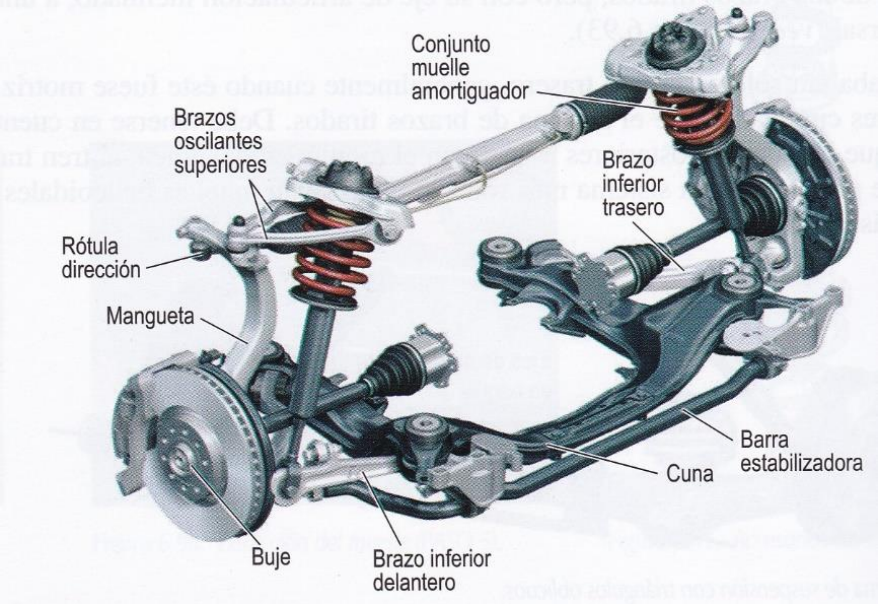


Figura 19. Suspensión multilink. (Miguel Ángel Pérez Belló "circuitos de Fluidos. Suspensión y Dirección (2011)).

3.3.3. **Suspensiones semiindependientes.**

Es otra variante del eje rígido empleada en los ejes traseros no motrices, cuya principal ventaja estriba en su sencillez y polivalencia. Su estructura se asemeja a la de un sistema de brazos tirados, en la que los mismos van unidos por un travesaño, en una zona muy próxima al eje de articulación. Dicho travesaño, dada su sección en "U", presenta la particularidad de poseer cierta flexibilidad, de tal forma que permite movimientos relativos entre ambas ruedas, a diferencia del eje rígido, en el que ambas oscilan a la vez.

Dada la ausencia de motricidad, presenta las ventajas del eje rígido en cuanto a guiado de las ruedas. Además, presenta la ventaja de su sencillez constructiva y de disposición en el vehículo. Por todo ello, su uso está muy extendido en vehículos de los segmentos inferiores y medios.

- **Eje de Dion.**

La suspensión *De Dion* se puede considerar como un tipo intermedio entre la suspensión trasera de eje rígido motriz y la suspensión independiente. Es una variante perfeccionada del eje rígido cuya principal ventaja sobre el mismo estriba en la disminución de las masas no suspendidas, al independizarse del conjunto el mecanismo grupo-diferencial. Para ello, el eje rígido dispone de unos brazos de guiado, empleándose muelles como elementos elásticos. Por tanto, el conjunto grupo-diferencial va fijado a la carrocería, transmitiéndose el movimiento a los conjuntos buje-rueda a través de los palieres articulados y telescópicos, que permiten variaciones de ancho de vía, formados por un tubo hueco rígido ligero generalmente curvado para no interferir con el alojamiento del diferencial y el final de la transmisión. A diferencia del eje rígido tubular, precisa de juntas cardan u homocinéticas.

Si bien en un primer momento se pensó que la suspensión *De Dion* resultaba muy eficaz y permitía mantener las ruedas con la caída y derivas adecuadas, posteriormente se observó que, en cuanto en una de las ruedas se sufría una compresión, en la parte trasera del vehículo se producía un cambio de dirección y por ello una pérdida de control direccional y estabilidad.

Otro de los mayores inconvenientes de este tipo de suspensión es la necesidad de llevar un tubo deslizante o semiejes desplazables transversalmente, lo que provoca que, en conjunto, el rozamiento del sistema sea elevado.

❖ **Eje torsional.**

El eje torsional es un tipo de suspensión semirrígida (semi-independiente), utilizada en las suspensiones traseras, en vehículos que tienen tracción delantera. La traviesa o tubo que une las dos ruedas tiene forma de "U", por lo que es capaz de deformarse un cierto ángulo cuando una de las ruedas encuentra un obstáculo, para después una vez pasado el obstáculo volver a la posición inicial.

Las ruedas están unidas rígidamente a dos brazos longitudinales unidos por un travesaño que los une y que se tuerce durante las sacudidas no simétricas, dando estabilidad al vehículo. Debido al bajo peso, al bajo coste y al poco espacio que ocupan, esta configuración es ideal para instalarla junto con otros componentes debajo del piso (depósito de combustible, escape, etc.). Esta configuración ha convertido a este tipo de suspensiones en una de las más empleadas en vehículos de gama media-baja.

Figura 20. Suspensión de eje torsional. (www.senseikoche.com)

3.4. Modelos de suspensión especiales.

3.4.1. **Amortiguador-compensador de carga.**

Los sistemas de regulación automática del nivel de la carrocería tienen por misión paliar los efectos de la variación de carga. Dichos sistemas disponen de dos amortiguadores en el eje trasero que permiten modificar su longitud sin más que introducir aire en unas cámaras que disponen a tal efecto. El aire a presión lo genera un compresor, movido por el motor de explosión, y almacenado en un calderín de reserva. La regulación es efectuada por medio de una válvula que está conectada, a través de un varillaje especial, con la suspensión trasera.

Cuando el vehículo se carga, se produce un descenso de la parte trasera y la válvula se abre para permitir el paso de aire a las cámaras de los amortiguadores, hasta que la carrocería se nivela. Cuando se descarga, la parte trasera se eleva abriendo la válvula que deja salir el aire del amortiguador, hasta que se vuelve a nivelar.

En los primeros modelos el sistema de regulación era totalmente mecánico, introduciéndose posteriormente los sistemas de control electrónico. Estos se diferencian en que controlan la posición por medio de unos sensores de desplazamiento y de una electroválvula.

3.4.2. **Suspensiones de amortiguación controlada.**

Estos sistemas actúan sobre la capacidad de absorción de los amortiguadores, adecuando su tarado más blando o más duro de acuerdo con el tipo de conducción y el recorrido realizo por el vehículo. A este efecto, el amortiguador está provisto de electroválvulas que permiten modificar los pasos calibrados de aceite entre las cámaras, de manera que su acción de frenado sobre las oscilaciones del muelle de suspensión se adaptada de la manera más conveniente, pilotándose las electroválvulas por medio de un calculador electrónico, que a su vez recibe diferentes señales de las condiciones de rodaje del vehículo.

El dispositivo de suspensión controlada permite adaptar la dureza de la amortiguación y, consecuentemente, de la suspensión, a tres estados de funcionamiento diferentes:

- **Suspensión deportiva:** se regulan los amortiguadores de manera que resulten duros.
- **Suspensión media o normal:** los amortiguadores son regulados a una dureza media, buscando un compromiso entre el confort y la estabilidad.
- **Suspensión confortable:** se regulan los amortiguadores de manera que resulten blandos, dando preferencia al confort del vehículo para circular por carreteras mal pavimentadas sin que se transmitan excesivas vibraciones a los pasajeros.

Los sistemas de amortiguación controlada permiten al conductor elegir entre dos modos de utilización: uno automático, donde la amortiguación es controlada electrónicamente en función de la utilización del vehículo; y otro impuesto, donde la amortiguación permanece constantemente en fase deportiva.

3.4.3. **Suspensión neumática.**

La suspensión neumática basa su funcionamiento en las propiedades que ofrece el aire sometido a presión. En esta suspensión, se sustituye el resorte mecánico (muelle, ballesta o barra de torsión) por un fuelle o cojín de aire que varía su rigidez.

La diferencia fundamental entre una suspensión basada en resortes mecánicos y otra neumática se centra en que las suspensiones mecánicas tienen rigidez constante, mientras que en las neumáticas esta rigidez es variable, debido a que a medida que se comprime el aire, más difícil resulta seguir comprimiéndolo. Como consecuencia de ello, la suspensión resulta más dura y a la inversa.

La principal característica que define a la suspensión neumática como tal es el hecho de utilizar el aire como sistema de suspensión.

Una suspensión neumática consiste básicamente en un cilindro unido al bastidor, dentro del cual hay un pistón unido al brazo oscilante de la rueda por medio de un vástago. El interior del cilindro está lleno de aire a una determinada presión, que ejerce la función de muelle,

Los sensores de altura controlan en todo momento la posición de la carrocería, enviando la señal a un control electrónico que gobierna una electroválvula. Cuando se producen variaciones de altura, el control actúa introduciendo o dejando salir aire de los muelles, manteniendo siempre una altura respecto al suelo constante. El aire a presión es generado por un compresor, movido por el motor de explosión del automóvil, y normalmente almacenado en un calderín.

En un turismo la suspensión neumática presenta ciertas ventajas respecto a la convencional. La principal es que mantiene la altura constante o puede regularla a gusto del conductor. Proporciona una circulación más confortable, tiene una frecuencia propia prácticamente independiente de la carga y su funcionamiento es más suave y ausente de ruidos por no existir partes metálicas en contacto.

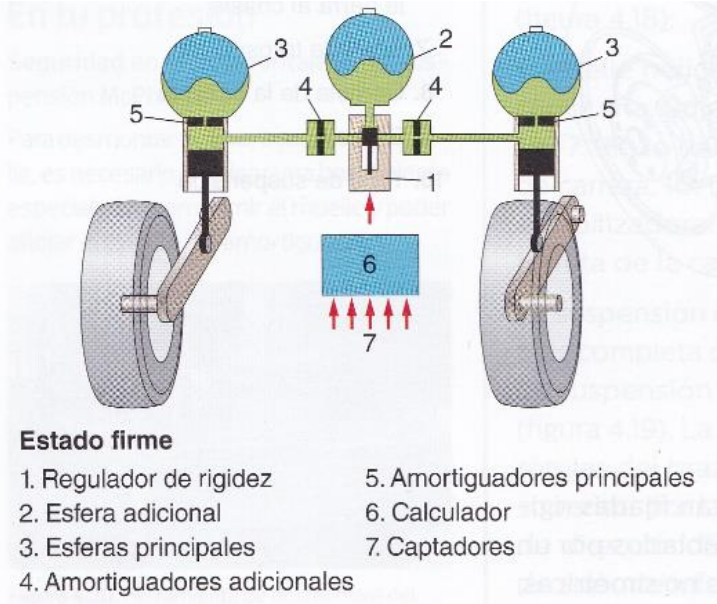


Figura 21. Componentes suspensión hidroneumática. . (Esteban José Domínguez, Julián Ferrer: "Mecánica del vehículo" (2011)).

3.4.4. **Suspensión hidroneumática.**

Esta suspensión combina un sistema mixto de elementos hidráulicos y neumáticos que garantiza una suspensión suave y elástica, facilitando, además, el reglaje y nivelación de la carrocería de forma automática. Esta suspensión proporciona la confortable sensación de "flotar", una gran estabilidad, que hace que apenas se noten las desigualdades del terreno y también un notable agarre de las ruedas al mismo.

Este tipo de suspensión tiene como principio la utilización de unas esferas metálicas divididas en dos cámaras interiores por medio de una membrana de goma deformable. En la cámara superior, la esfera contiene herméticamente encerrado un gas a presión (nitrógeno) que es compresible y, por tanto, realiza la función del muelle. Dicho gas es comprimido por la acción de un fluido hidráulico mineral situado en la cámara inferior de la esfera. En la base de la esfera se encuentran unos pasos calibrados a través de los cuales el fluido pasa hacia un cilindro que es prolongación de la esfera y, en cuyo interior se encuentra un pistón móvil que se desplaza por el interior de éste. Dicho pistón se encuentra sujeto al brazo de la suspensión por medio de un vástago. El aceite hidráulico supone la transmisión entre el pistón y la masa de gas, en lo que constituye el principio de funcionamiento de la suspensión hidroneumática.

La nivelación de la carrocería se consigue haciendo entrar aceite a presión en el cilindro cuando aumenta la carga, o haciéndole salir cuando ésta disminuye, por medio de una válvula de corredera (válvula niveladora).

El sistema permite dar tres niveles de altura al vehículo: la más baja para circular por autopistas a gran velocidad, pues al bajar la carrocería lo hace el centro de gravedad del vehículo, aumentando su estabilidad; la intermedia, para circulación normal; y la más alta, para circular por terreno accidentado.

Este modelo de suspensión funciona con un rozamiento mínimo debido a las propiedades lubricantes del líquido, y resulta sumamente confortable para los pasajeros, proporcionando además gran estabilidad al vehículo. Su diseño sencillo agrupa en un solo conjunto el muelle y el amortiguador.

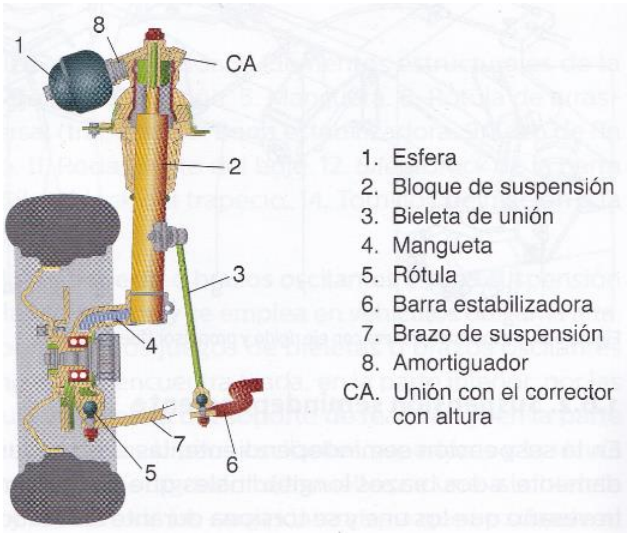


Figura 22. Suspensión delantera hidroneumática. . (Esteban José Domínguez, Julián Ferrer: "Mecánica del vehículo" (2011)).

3.4.5. **Suspensión hidractiva.**

La suspensión hidractiva es un modelo de suspensión pilotada. Este tipo de suspensión parte de una suspensión hidroneumática convencional, a la que se añaden otros elementos hidráulicos controlados electrónicamente.

Las suspensiones de tipo hidroneumático controladas por ordenador permiten variar la flexibilidad y amortiguación de las mismas, adaptándola a las condiciones de marcha del vehículo y tipo de conducción, controlando la inclinación de la carrocería a medida que surgen los obstáculos, cambia la velocidad, se gira el volante o se actúa en los frenos. El sistema permite nivelar uno o ambos ejes cuando se carga el vehículo y reducir la altura de la carrocería para rodaje a alta velocidad, o incrementarla a baja velocidad cuando se transita por carreteras en mal estado.

La gestión del sistema hidráulico y las condiciones de su conmutación son confiadas a un calculador electrónico, que compara las informaciones recibidas de los captadores con leyes previamente integradas, para controlar el correcto funcionamiento del sistema.

Por otra parte, el conductor puede elegir el modo de suspensión entre confort y deportivo accionando un interruptor en el tablero de instrumentos. Además, el conductor puede imponer la altura de la carrocería mediante un mando adecuado de tres posiciones de marcha (normal de carretera, alta para circulación por terreno accidentado, y baja para cargar el vehículo o enganchar un remolque) más una cuarta para la función de gato.

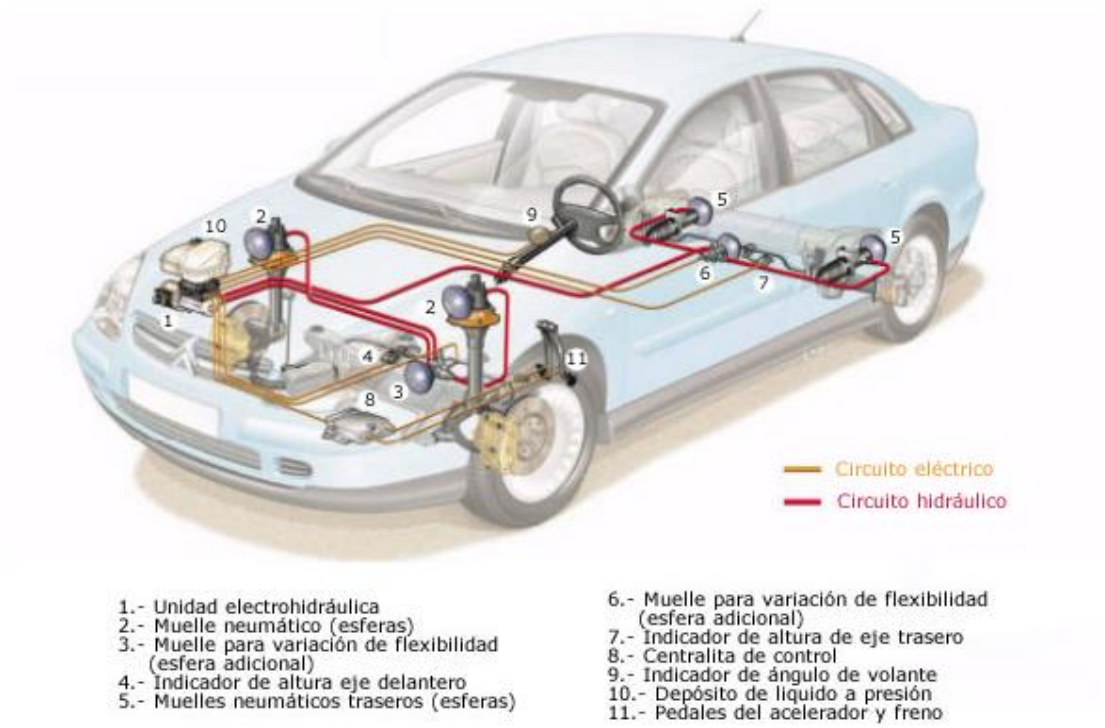


Figura 23. Esquema de una suspensión hidractiva (www.aficionadosalamecanica.net).

3.4.6. **Sistema activo antibalanceo.**

El balanceo que sufre la carrocería de un vehículo cuando recorre una curva, es atenuado por las barras de torsión dispuestas en cada uno de los trenes de rodaje, de manera que a mayor grosor de estas barras corresponde una mayor rigidez y, por tanto, un mayor efecto frente al balanceo; pero en contrapartida, disminuye en la misma proporción la flexibilidad de la suspensión, que se hace más rígida, no solo en curva, sino también circulando en recta, lo que implica una pérdida de confort de los pasajeros.

Una configuración ideal sería aquella en la que se dispusieran barras estabilizadoras de poca rigidez en ambos trenes, manteniendo un buen nivel de confort, pero suficiente en todo caso para mantener la estabilidad en recta, y pudiera aumentarse dicha rigidez en curva, cuando se produce el balanceo de la carrocería. Esto supone la funcionalidad de una barra estabilizadora de flexibilidad variable.

La rigidez de una barra estabilizadora puede ser aumentada con la ayuda de un dispositivo hidráulico enlazado a ella, consistente en que uno de los extremos de las barras estabilizadoras de cada tren de rodaje, se une al émbolo de un cilindro hidráulico, fijado al elemento portador de la suspensión de una de las ruedas de cada eje.

Para la activación del sistema, la unidad de control recibe información de dos sensores: uno de velocidad del vehículo y el otro de giro del volante de la dirección, en función de las cuales acciona la electroválvula reguladora que comanda el sistema hidráulico.

El sistema antibalanceo combinado con la suspensión hidractiva, actuando en conjunto, proporcionan al vehículo un confort de marcha excelente, cualquiera que sea su modo de utilización. Dado que con este sistema se varía no sólo el tarado de la amortiguación, sino la flexibilidad de la suspensión, el vehículo equiparado así tiene mejor comportamiento en cualquier

condición de marcha, por cuya causa a este modelo de suspensión se le llama también **suspensión inteligente**.

En los sistemas neumáticos de suspensión se logra una l imitación del balanceo de la carrocería, actuando sobre cada una de las unidades neumáticas de la suspensión de manera individual, aumentando la presión en las del lado exterior en la curva para aumentar la rigidez y compensar la inclinación que sufre la carrocería en estas condiciones de marcha. En otros sistemas, la barra estabilizadora está partida en dos mitades, enlazadas por un motor hidráulico, que al ser puesto en funcionamiento refuerza la rigidez dela barra.

4. EL ENTORNO DE PROGRAMACIÓN MATLAB.

MATLAB es un entorno de computación técnica que posibilita la ejecución del cálculo numérico y simbólico de forma rápida y precisa, acompañado de características gráficas y de visualización avanzadas aptas para el trabajo científico y la ingeniería. MATLAB es un entorno interactivo para el análisis y el modelado que implementa más de 500 funciones para el trabajo en distintos campos de la ciencia.

Por otra parte, MATLAB presenta un lenguaje de programación de muy alto nivel basado en vectores, arrays y matrices.

Además, el entorno básico de MATLAB se complementa con una amplia colección de toolboxes que contienen funciones específicas para determinadas aplicaciones en diferentes ramas de las ciencias e ingeniería.

La arquitectura de MATLAB es abierta y ampliamente extensible, permitiendo la relación con Excel, C, Fortran y otras aplicaciones externas muy utilizadas e importantes. Entre otras cosas, el código escrito en lenguaje de MATLAB puede ser traducido a C de forma inmediata.

MATLAB también permite la operatividad entre plataformas posibilitando trabajar con distintos sistemas operativos y relacionar el trabajo realizado en las distintas plataformas.

MATLAB es un software en continuo crecimiento y muy adaptable a los avances científicos y al trabajo en laboratorios I+D, que resuelve los problemas que presenta la ingeniería en el desarrollo de productos innovadores.

En el campo de la automoción, MATLAB posibilita aplicaciones para trabajar en la ingeniería de control, sistemas de suspensión, sistemas ABS y diseño de bloques de embrague.

MATLAB es una de las muchas y sofisticadas herramientas de computación disponibles en el comercio para resolver problemas de matemáticas, tales como Maple, Mathematica y Mathcad. A pesar de lo que afirman sus defensores, ninguna de ellas es “la mejor”. Todas tienen fortalezas y debilidades. Cada una permitirá efectuar cálculos matemáticos básicos, pero difieren en el modo como manejan los cálculos simbólicos y procesos matemáticos más complicados, como la manipulación de matrices. Por ejemplo, MATLAB es superior en los cálculos que involucran matrices, mientras que Maple lo supera en los cálculos simbólicos. El nombre mismo de MATLAB es una abreviatura de *Matrix Laboratory*, laboratorio matricial.

Dado que MATLAB es tan fácil de usar, muchas tareas de programación se llevan a cabo con él. Sin embargo, MATLAB no siempre es la mejor herramienta para usar en una tarea de programación. El programa destaca en cálculos numéricos, especialmente en los relacionados con matrices y gráficas, pero no en el procesamiento de palabras. C++ y FORTRAN son programas de propósito general y serían los programas de elección para aplicaciones grandes como los sistemas operativos o el software de diseño. Por lo general, los programas de alto nivel no ofrecen

acceso fácil a la graficación, que es una aplicación en la que destaca MATLAB. El área principal de interferencia entre MATLAB y los programas de alato nivel es el "procesamiento de números": programas que requieren cálculos repetitivos o el procesamiento de grandes cantidades de datos. Tanto MATLAB como los programas de alto nivel son buenos en el procesamiento de números. Por lo general, es más fácil escribir un programa que "procese números" en MATLAB, pero usualmente se ejecutará más rápido en C++ o FORTRAN. La única excepción a esta regla son los cálculos que involucran matrices: puesto que MATLAB es óptimo para matrices, si un problema se puede formular con una solución matricial, MATLAB lo ejecuta sustancialmente más rápido que un programa similar en un lenguaje de alto nivel.

5. FUNDAMENTOS MATEMÁTICOS.

5.1. Método de la función de transferencia.

El método de la función de transferencia, basado en transformadas de Laplace, se utiliza comúnmente para formular y resolver problemas dinámicos en la literatura de controles. También se puede utilizar para resolver problemas de vibración forzada. La función de transferencia relaciona la salida de un sistema con su entrada. Esta función permite separar la entrada, el sistema y la salida en tres partes separadas y distintas (a diferencia de la ecuación diferencial, en la cual los tres aspectos no son fáciles de separar).

La función de transferencia de una ecuación diferencial lineal invariable con el tiempo se define como la relación de la transformada de Laplace de la salida o función de respuesta, con la transformada de Laplace de la entrada o función forzada, suponiendo condiciones iniciales cero.

El procedimiento general utilizado para determinar la función de transferencia de una ecuación diferencial lineal, implica tomar las transformadas de Laplace de ambos lados, suponiendo condiciones iniciales cero, y resolviendo para la relación de la transformada de Laplace de la salida y la transformada de Laplace de la entrada. Como la ecuación diferencial lineal se compone de la variable y sus derivadas, la transformada de Laplace la transforma en una ecuación polinomial en la variable de Laplace *s*.

5.1.1. **Transformada de Laplace de la derivada.**

Sea $f(t)$ una función derivable hasta orden n, la transformada de Laplace de la primera derivada de $f(t)$ se define como

$$\mathcal{L}\left[\frac{df(t)}{dt}\right] = \int_0^\infty e^{-st}\frac{df(t)}{dt}dt \qquad (5)$$

Si utilizamos integración por partes, la ecuación (1) se expresa como

$$\mathcal{L}\left[\frac{df(t)}{dt}\right] = e^{-st}f(t)|_0^\infty - \int_0^\infty (-se^{-st})f(t)dt = -f(0) + sF(s) \qquad (6)$$

Donde $F(s)$ es la transformada de Laplace de $f(t)$ y, $f(0)$ es el valor inicial de $f(t)$, es decir, el valor de $f(t=0)$. Utilizando un método parecido, la transformada de Laplace de la segunda derivada de $f(t)$ se define como

$$\mathcal{L}\left[\frac{d^2f(t)}{dt^2}\right] = \int_0^\infty e^{-st}\frac{d^2f(t)}{dt^2}dt \qquad (7)$$

La ecuación (3) se puede simplificar para obtener

$$\mathcal{L}\left[\frac{d^2f(t)}{dt^2}\right] = -\dot{f}(0) - sf(0) + s^2F(s) \qquad (8)$$

Donde $\dot{f}(0)$ es el valor de $\frac{df}{dt}$ en el instante $t=0$. Utilizando un método semejante, la transformada de Laplace de la derivada enésima de $f(t)$ se puede hallar como

$$\mathcal{L}\left[\frac{d^nf(t)}{dt^n}\right] = \mathcal{L}\left[f^{(n)}\right] = s^nF(s) - s^{n-1}f(0) - s^{n-2}f^{(1)}(0) - \cdots - sf^{(n-1)}(0) \qquad (9)$$

Donde $f(n)$ se utiliza para denotar la derivada enésima de f, $\frac{d^nf(t)}{dt^n}$.

5.1.2. **Función de transferencia de una ecuación diferencial.**

La siguiente ecuación diferencial lineal invariable con el tiempo de orden enésimo, rige el comportamiento de un sistema dinámico:

$$\begin{aligned} a_n\frac{d^nx(t)}{dt^n} + a_{n-1}\frac{d^{n-1}x(t)}{dt^{n-1}} + \cdots + a_0x(t) \\ = b_m\frac{d^mf(t)}{dt^m} + b_{m-1}\frac{d^{m-1}f(t)}{dt^{m-1}} + \cdots + b_0f(t) \end{aligned} \qquad (10)$$

Donde $x(t)$ es la salida, $f(t)$ es la entrada, t es el tiempo y las a_i y b_i son constantes.

Tomando la transformada de Laplace de ambos lados de la ecuación (10), obtenemos

$$\begin{aligned} a_ns^nX(s) + a_{n-1}s^{n-1}X(s) + \cdots + a_0X(s) \\ + condiciones\ iniciales\ que\ conllevan\ x(t) \\ = b_ms^mF(s) + b_{m-1}s^{m-1}F(s) + \cdots + b_0F(s) \\ + condiciones\ iniciales\ que\ conllevan\ f(t) \end{aligned} \qquad (11)$$

Se ve que la ecuación (11) es una expresión puramente algebraica. Si se supone que todas las condiciones iniciales son cero, la ecuación (11) se reduce a la forma siguiente:

$$\{a_ns^n + a_{n-1}s^{n-1} + \cdots + a_0\}X(s) = \{b_ms^m + b_{m-1}s^{m-1} + \cdots + b_0\}F(s) \qquad (12)$$

Resolviendo la ecuación (12), la función de transferencia del sistema evaluado en condiciones iniciales cero, $T(s)$, se determina como la relación de la transformada de la salida, $X(s)$, y la transformada de la entrada, $F(s)$:

$$T(s) = \frac{X(s)}{F(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \cdots + b_0}{a_n s^n + a_{n-1} s^{n-1} + \cdots + a_0} \quad (13)$$

Se ve que la función de transferencia identifica la entrada, $F(s)$, la salida, $X(s)$, y el sistema (definido por la expresión del lado derecho de la ecuación (13)) como entidades aparte.

A modo de ejemplo calcularemos la función de transferencia de un sistema de un solo grado de libertad viscosamente amortiguado sometido a una fuerza externa $f(t)$ como muestra la siguiente figura.

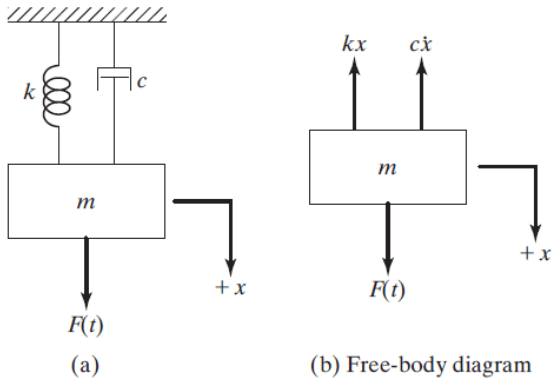


Figura 24. Sistema Resorte-Masa-Amortiguador (Singiresu S. Rao, 2011)

La ecuación de movimiento del sistema está dada por

$$m\ddot{x} + c\dot{x} + kx = f(t) \quad (14)$$

Si tomamos las transformadas de Laplace de ambos lados de la ecuación (14), obtenemos

$$m\mathcal{L}[\ddot{x}(t)] + c\mathcal{L}[\dot{x}(t)] + k\mathcal{L}[x] = \mathcal{L}[f(t)] \quad (15)$$

o

$$m\{s^2X(s) - sx(0) - \dot{x}(0)\} + c\{sX(s) - x(0)\} + kX(s) = F(s) \quad (16)$$

La ecuación (16) se puede reescribir como

$$\{ms^2 + cs + k\}X(s) - \{msx(0) + m\dot{x}(0) + sx(0)\} = F(s) \quad (17)$$

Donde $X(s) = \mathcal{L}[x(t)]$ y $F(s) = \mathcal{L}[f(t)]$. La función de transferencia del sistema se obtiene a partir de la ecuación (17), fijando $x(0) = \dot{x}(0) = 0$, como

$$T(s) = \frac{\mathcal{L}[salida]}{\mathcal{L}[entrada]}\Big|_{condiciones\ iniciales\ cero} = \frac{X(s)}{F(s)} = \frac{1}{ms^2 + cs + k} \qquad (18)$$

Notas:

1. La función de transferencia es una propiedad del sistema y no se relaciona con la entrada o la función forzada.

2. La función de transferencia no proporciona ninguna información sobre la estructura física del sistema. De hecho, las funciones de transferencia de muchos sistemas físicamente diferentes pueden ser idénticas.

3. La representación de un sistema dinámico mediante la función de transferencia es muy útil en la teoría de control, así como en pruebas de vibración para medir la respuesta dinámica y para identificar sistemas. Por ejemplo, en el caso de un sistema cuyos parámetros como masa (m), constante de amortiguamiento (c) y rigidez de resorte (k) no son conocidos, la función de transferencia se determina experimentalmente midiendo la respuesta o salida ante una entrada conocida. Una vez determinada la función de transferencia, describe por completo las características dinámicas del sistema.

En pruebas de vibración, la respuesta de vibración medida (por una entrada o función forzada conocida) podría ser el desplazamiento, la velocidad o, más comúnmente, la aceleración.

4. Si se conoce la función de transferencia de un sistema, la salida o respuesta del sistema se puede determinar para cualquier tipo de entrada.

$$X(s) = T(s)F(s) \qquad (19)$$

5. La variable *s* en la transformada de Laplace es un número complejo y por consiguiente la función de transferencia es una cantidad compleja. Se indica la variable *s* en la transformada de Laplace en forma compleja como

$$s = \sigma + i\omega_d \qquad (20)$$

donde σ y ω_d representan las partes real e imaginaria respectivamente de *s*.

6. La transformada de Laplace convierte una ecuación diferencial lineal en una expresión algebraica. Transforma las funciones definida en función de la variable independiente (como el tiempo), en funciones en términos de la cantidad compleja *s* como la variable independiente. Para utilizar la transformada de Laplace, primero tenemos que determinar la función de transferencia del sistema.

5.1.3. **Función de transferencia de frecuencia.**

La función de transferencia de frecuencia, $T(i\omega)$, se puede obtener sustituyendo $s = i\omega$ en la función de transferencia general $T(s)$.

Para el sistema resorte-masa-amortiguador considerado en el apartado anterior, la función de transferencia general está dada por

$$T(s) = \frac{1}{ms^2 + cs + k} \qquad (21)$$

Utilizando $s = i\omega$, la función de transferencia de frecuencia del sistema es

$$T(i\omega) = \frac{1}{k - m\omega^2 + ic\omega} \qquad (22)$$

La amplitud o magnitud de $T(i\omega)$ está dada por

$$M_s(s) = |T(i\omega)| = \frac{1}{[(k - m\omega^2)^2 + (ic\omega)^2]^{1/2}} \qquad (23)$$

y el ángulo de fase por

$$\phi_s = \tan^{-1}\left(\frac{\omega c}{m\omega^2 - k}\right) \qquad (24)$$

5.1.4. **Representación de las características de respuesta en frecuencia.**

La respuesta de frecuencia de un sistema de segundo grado, como el sistema de resorte-masa-amortiguador, indica la respuesta de estado estable del sistema a una entrada senoidal para posibles frecuencias diferentes a la de la entrada senoidal. Para algunos sistemas, la frecuencia ω variará dentro de un rango considerablemente grande. En esos casos es conveniente utilizar escalas logarítmicas para acomodar el rango completo de ω en gráficas trazadas en papel de tamaño estándar.

Diagramas de Bode.

Un diagrama de Bode se compone de dos gráficas, una gráfica del logaritmo de la magnitud de la función de transferencia de frecuencia (M) contra el logaritmo de la frecuencia (ω) y una gráfica del ángulo de fase (ϕ) contra el logaritmo de la frecuencia (ω).

Como representación estándar de la magnitud logarítmica de $T(i\omega)$, se utiliza una unidad logarítmica conocida como *decibel*, abreviada dB. La relación de magnitud en decibeles, *m*, se define como

$$m = 10\log_{10}(M^2) = 20\log_{10} M \ \ dB \qquad (25)$$

De la ecuación (25) se ve que para cualquier número N, su valor en decibeles es $20\log_{10} N$.

5.2. Superposición modal.

El modelo correspondiente a un sistema de N g.d.l. con amortiguamiento viscoso es:

$$[M]\{\ddot{U}\} + [C]\{\dot{U}\} + [K]\{U\} = \{F(t)\} \qquad (26)$$

en primer lugar, estudiaremos el problema de vibraciones libres, que corresponderá a un problema de valores y vectores propios. En segundo lugar, analizaremos el problema de respuesta en régimen permanente, centrándonos en el caso de fuerzas excitadoras armónicas y, por lo tanto, en la respuesta en frecuencia.

5.2.1. **Vibraciones libres.**

Para el caso de vibraciones libres las fuerzas exteriores son nulas, y en consecuencia:

$$[M]\{\ddot{U}\} + [C]\{\dot{U}\} + [K]\{U\} = \{0\} \qquad (27)$$

Consideremos que la solución es de la forma:

$$\{U\} = \{\overline{U}\}e^{\lambda t} \qquad (28)$$

Sustituyendo en la ecuación diferencial obtenemos:

$$(\lambda^2[M] + \lambda[C] + [K])\{\overline{U}\} = \{0\} \qquad (29)$$

Para obtener una solución distinta de la trivial, el determinante de la matriz de coeficientes debe ser nulo. Desarrollando este determinante se obtiene el polinomio característico de grado 2N, del que se pueden obtener sus correspondientes raíces (λ_r). para cada una de las raíces, valores propios, se obtendrá en correspondiente vector propio o modo de vibración. Considerando las características de las matrices que definen el problema (simétricas, de coeficientes constantes), las raíces del polinomio característico aparecen por parejas conjugadas y los vectores propios son complejos apareciendo igualmente por parejas conjugadas:

$$\left.\begin{matrix}\lambda_r, \lambda_r^* \\ \{\Psi\}_r, \{\Psi^*\}_r\end{matrix}\right\} r = 1, \dots, N \qquad (30)$$

El hecho de que los vectores propios sean complejos implica que cuando el sistema vibra a una frecuencia natural, los movimientos de los diferentes g.d.l están desfasados entre sí ángulos arbitrarios. Las raíces del polinomio característico se pueden expresar como:

$$\lambda_r = \omega_r \left(-\zeta_r + i\sqrt{1-\zeta_r^2} \right) \quad (31)$$

Definamos un nuevo vector $\{Y\}$, que incluye desplazamientos y velocidades, como:

$$\{Y\} = \begin{Bmatrix} U \\ \dot{U} \end{Bmatrix} \quad (32)$$

La ecuación diferencial de movimiento puede escribirse entonces como:

$$[C:M]\{\dot{Y}\} + [K:0]\{Y\} = \{0\} \quad (33)$$

Con objeto de que las matrices del problema sean cuadradas y simétricas, redefinimos el problema insertando nuevas ecuaciones, resultando finalmente:

$$\begin{bmatrix} C & M \\ M & 0 \end{bmatrix}\{\dot{Y}\} + \begin{bmatrix} K & 0 \\ 0 & -M \end{bmatrix}\{Y\} = \{0\} \leftrightarrow [A]\{\dot{Y}\} + [B]\{Y\} = \{0\} \quad (34)$$

Así, la formulación obtenida corresponde a un problema estándar de valores y vectores propios, aunque el tamaño de las matrices es el doble del original. Suponiendo la solución de la forma:

$$\{Y\} = \{\bar{Y}\}e^{\lambda t} \quad (35)$$

Y sustituyendo en la ecuación diferencial se obtiene:

$$(\lambda_r [A] + [B])\{\sigma\}_r = \{0\} \quad (36)$$

La solución correspondiente es un conjunto de valores propios (por parejas conjugadas) y vectores propios (por parejas conjugadas debido a las características de las matrices [A] y [B]):

$$\begin{cases} \lambda_r \\ \{\sigma\}_r \end{cases} \quad r = 1, \ldots, 2N \ (parejas\ conjugadas) \quad (37)$$

Los vectores propios son en este caso complejos. Definiendo la matriz $[\sigma]$ como aquella que sus columnas son los vectores propios, se obtienen como propiedades de ortogonalidad:

$$\begin{matrix} [\sigma]^T [A][\sigma] = [a_r] \\ [\sigma]^T [B][\sigma] = [b_r] \end{matrix} \quad con: \lambda_r = -\frac{b_r}{a_r} \quad r = 1, \ldots, 2N \quad (38)$$

Siendo las matrices $[a_r]$ y $[b_r]$ diagonales. Es posible escalar los vectores propios de forma que, por ejemplo, la matriz $[A]$ sea la identidad. Esto se consigue considerando los vectores propios:

$$\{\theta\}_r = \frac{\{\sigma\}_r}{\left(\{\sigma\}_r^T [A]\{\sigma\}_r\right)^{1/2}} \quad (39)$$

De esta forma, al realizar la transformación, se obtiene:

$$\begin{aligned}[\theta]^T[A][\theta]&=[I]\\ [\theta]^T[B][\theta]&=[-\lambda_r]\end{aligned} \qquad (40)$$

o, lo que es lo mismo, el conjunto de ecuaciones diferenciales:

$$\begin{aligned}\dot{q}_r-\lambda_r q_r&=Q_r(t)=\{\theta\}_r^T\{F(t)\}\\ \dot{q}_r-\lambda_r^* q_r&=Q_r(t)=\{\theta^*\}_r^T\{F(t)\}\end{aligned}\quad r=1,\dots,N \qquad (41)$$

con

$$\{U(t)\}=[\theta]\{q(t)\}$$

5.2.2. **Funciones de respuesta en frecuencia.**

Analicemos ahora la respuesta en régimen permanente del sistema de N g.d.l. con amortiguamiento viscoso general para el caso de fuerzas excitadoras armónicas. En este caso, las fuerzas exteriores serán:

$$\{F(t)\}=\{\overline{F}\}e^{i\omega t} \qquad (42)$$

La solución, el sistema desacoplado, será de la forma:

$$q_r(t)=\bar{q}_r e^{i\omega t} \qquad (43)$$

y, sustituyendo en la ecuación diferencial:

$$\left(i\omega\bar{q}_r-\lambda_r\bar{q}_r\right)=\{\theta\}_r^T\{\overline{F}\}\rightarrow\bar{q}_r=\frac{\{\theta\}_r^T\{\overline{F}\}}{i\omega-\lambda_r} \qquad (44)$$

La respuesta en la coordenada {U} será, por tanto:

$$\{U(t)\}=\{\overline{U}\}e^{i\omega t}=\sum_{r=1}^{N}\left(\{\theta\}_r\frac{\{\theta\}_r^T\{\overline{F}\}}{i\omega-\lambda_r}+\{\theta^*\}_r\frac{\{\theta^*\}_r^T\{\overline{F}\}}{i\omega-\lambda_r^*}\right)e^{/\omega t} \qquad (45)$$

Si suponemos que solo está excitado el g.d.l. ‘k’, es decir, que la única componente no nula del vector $\{\overline{F}\}$ es la k-ésima, y analizamos la respuesta en el g.d.l. ‘j’, obtenemos la correspondiente función de respuesta en frecuencia:

$$H_{jk}(\omega)=\frac{\overline{U}_j}{\overline{F}_k}=\sum_{r=1}^{N}\left(\frac{\{\theta_j\}_r\{\theta_k\}_r}{i\omega-\lambda_r}+\frac{\{\theta_j^*\}_r\{\theta_k^*\}_r}{i\omega-\lambda_r^*}\right) \qquad (46)$$

5.3. Análisis de espacio de estado.

5.3.1. Formulación del espacio de estado.

Para explicar la formulación del espacio de estado, nos basaremos en el sistema de tres grados grados de libertad amortiguado que se muestra en la Figura 25.

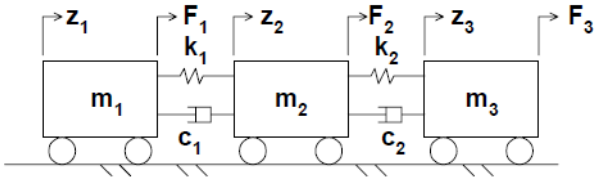


Figura 25. Sistema de tres grados de libertad amortiguado. (Michael R. Hatch, 2001)

Aplicanto la segunda ley de Newton, las ecuaciones de movimiento en forma matricial son:

$$\begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_2 \end{bmatrix}\begin{bmatrix} \ddot{z}_1 \\ \ddot{z}_2 \\ \ddot{z}_3 \end{bmatrix}+\begin{bmatrix} c_1 & -c_1 & 0 \\ -c_1 & (c_1+c_2) & -c_2 \\ 0 & -c_2 & c_2 \end{bmatrix}\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \end{bmatrix} + \begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & (k_1+k_2) & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix}\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}=\begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix} \qquad (47)$$

Expandiendo las ecuaciones:

$$\begin{gathered} m_1\ddot{z}_1+c_1\dot{z}_1-c_1\dot{z}_2+k_1z_1-k_1z_2=F_1 \\ m_2\ddot{z}_2-c_1\dot{z}_1+(c_1+c_2)\dot{z}_2-c_2\dot{z}_3-k_1z_1+(k_1+k_2)z_2-k_2z_3=F_2 \\ m_3\ddot{z}_3-c_2\dot{z}_2+c_2\dot{z}_3-k_2z_2+k_2z_3=F_3 \end{gathered} \qquad (48a,b,c)$$

Las tres ecuaciones anteriores son ecuaciones diferenciales de segundo orden las cuales requieren conocer las condiciones iniciales de posición y velocidad para los tres grados de libertad para resolver la respuesta transitoria.

En la formulación del espacio de estado, las tres ecuaciones diferenciales de segundo orden, se convierten en seis ecuaciones diferenciales de primer orden. Siguiendo la notación típica del espacio de estado, nos referiremos a los estados como “x” y a las salidas como “y”.

Empezamos resolviendo las tres ecuaciones de (48) para las derivadas más altas, en este caso tres segundas derivadas, $\ddot{z}_1, \ddot{z}_2, \ddot{z}_3$:

$$\begin{aligned} \ddot{z}_1 &= \left(-c_1\dot{z}_1 + c_1\dot{z}_2 - k_1 z_1 + k_1 z_2 + F_1\right)/m_1 \\ \ddot{z}_2 &= \left(c_1\dot{z}_1 - \left(c_1 + c_2\right)\dot{z}_2 + c_2\dot{z}_3 + k_1 z_1 - \left(k_1 + k_2\right)z_2 + k_2 z_3 + F_2\right)/m_2 \\ \ddot{z}_3 &= \left(c_2\dot{z}_2 - c_2\dot{z}_3 + k_2 z_2 - k_2 z_3 + F_3\right)/m_3 \end{aligned} \qquad (49a,b,c)$$

Ahora cambiamos la notación, utilizando "x" para definir los 6 estados; tres posiciones y tres velocidades:

$$x_1 = z_1 \text{ Posición de la Masa 1} \qquad (50)$$

$$x_2 = \dot{z}_1 \text{ Velocidad de la Masa 1} \qquad (51)$$

$$x_3 = z_2 \text{ Posición de la Masa 2} \qquad (52)$$

$$x_4 = \dot{z}_2 \text{ Velocidad de la Masa 2} \qquad (53)$$

$$x_5 = z_3 \text{ Posición de la Masa 3} \qquad (54)$$

$$x_6 = \dot{z}_3 \text{ Velocidad de la Masa 3} \qquad (55)$$

Utilizando esta notación observamos la relación entre el estado y sus primeras derivadas:

$$\dot{z}_1 = x_2 = \dot{x}_1 \qquad (56)$$

$$\dot{z}_2 = x_4 = \dot{x}_3 \qquad (57)$$

$$\dot{z}_3 = x_6 = \dot{x}_5 \qquad (58)$$

También entre las primeras y segundas derivadas:

$$\ddot{z}_1 = \dot{x}_2 \qquad (59)$$

$$\ddot{z}_2 = \dot{x}_4 \qquad (60)$$

$$\ddot{z}_3 = \dot{x}_6 \qquad (61)$$

Reescribiendo las ecuaciones para $\ddot{z}_1, \ddot{z}_2$ y $\ddot{z}_3$ en términos para los ocho estados desde x_1 a x_6 y añadiendo las dos ecuaciones que definen la relación entre posición y velocidad:

$$
\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= \left(-c_1 x_2 + c_1 x_4 - k_1 x_1 + k_1 x_3 + F_1\right)/m_1 \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= \left(c_1 x_2 - \left(c_1 + c_2\right)x_4 + c_2 x_6 + k_1 x_1 - \left(k_1 + k_2\right)x_3 + k_2 x_5 + F_2\right)/m_2 \\
\dot{x}_5 &= x_6 \\
\dot{x}_6 &= \left(c_2 x_4 - c_2 x_6 + k_2 x_3 - k_2 x_5 + F_3\right)/m_3
\end{aligned}
\qquad \textit{(62a-f)}
$$

Reescribiendo las ecuaciones (62a-f) en forma matricial se obtiene: $[\dot{x}] = [A][x] + [B]u$

$$
\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{Bmatrix} =
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 \\
\frac{-k_1}{m_1} & \frac{-c_1}{m_1} & \frac{k_1}{m_1} & \frac{c_1}{m_1} & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
\frac{k_1}{m_2} & \frac{c_1}{m_2} & \frac{-\left(k_1 + k_2\right)}{m_2} & \frac{-\left(c_1 + c_2\right)}{m_2} & \frac{k_2}{m_2} & \frac{c_2}{m_2} \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & \frac{k_2}{m_3} & \frac{c_2}{m_3} & \frac{-k_2}{m_3} & \frac{-c_2}{m_3}
\end{bmatrix}
\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{Bmatrix} +
\begin{Bmatrix} 0 \\ \frac{F_1}{m_1} \\ 0 \\ \frac{F_2}{m_2} \\ 0 \\ \frac{F_3}{m_3} \end{Bmatrix} \cdot (1)
\qquad \textit{(63)}
$$

5.3.2. **Definición de las ecuaciones de movimiento del espacio de estado.**

La forma de las ecuaciones de movimiento en el espacio de estado es:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathrm{u} \qquad \textit{(64)}$$

Donde las matrices **A** y **B** se muestran en (63). la matriz **A** se conoce como matriz del sistema, la matriz **B** es la matriz de entrada, y el escalar u es la entrada. El vector columna **x** es el estado del sistema.

5.3.3. **Formas de la matriz de entrada.**

Debido a que "u" es un escalar, la naturaleza de la matriz de entrada **B** cambia dependiendo de qué entrada se utiliza. Si el sistema es un sistema de entrada única (Single Input, SI) con una fuerza en la masa 1, 2 o 3 la matriz **B** cambia de la siguiente manera:

$$F_1: \quad \mathbf{B} = \begin{bmatrix} 0 \\ \frac{F_1}{m_1} \\ 0 \\ 0 \\ 0 \end{bmatrix}, \qquad F_2: \quad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{F_2}{m_2} \\ 0 \\ 0 \end{bmatrix}, \qquad F_3: \quad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{F_3}{m_3} \end{bmatrix} \qquad \textit{(65a,b,c)}$$

Si la misma función forzada u (por ejemplo, una función escalón o una función seno) se aplica a varios grados de libertad simultáneamente (por ejemplo, una fuerza de magnitud F_1 a la masa 1 y una fuerza de magnitud F_3 a la masa 3) la matriz de entrada se convertiría en:

$$\mathbf{B} = \begin{bmatrix} 0 \\ \frac{F_1}{m_1} \\ 0 \\ 0 \\ 0 \\ \frac{F_3}{m_3} \end{bmatrix} \qquad \textit{(66)}$$

Para un sistema de entra múltiple (MI, Multi Input), donde las fuerzas se aplican independientemente entre sí a las masas separadas, es necesaria una matriz de entrada de múltiples columnas. Por ejemplo, para entradas diferentes en la masa 1 y la masa 2, ninguna en la masa 3, la matriz de entrada se convertiría en:

$$\mathbf{B} = \begin{bmatrix} 0 & 0 \\ \frac{F_1}{m_1} & 0 \\ 0 & 0 \\ 0 & \frac{F_2}{m_2} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad \textit{(67)}$$

5.3.4. **Formas de la matriz de salida.**

Para explicar el caso en el que la salida deseada no es sólo los estados, sino que es una combinación lineal de los estados, se define una matriz de salida **C** para relacionar las salidas con los estados. Además, una matriz **D**, conocida como matriz de transmisión directa, se multiplica por la entrada "u" para tener en cuenta salidas que están relacionadas con las entradas pero que pasan por alto los estados.

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}u \qquad \textit{(68)}$$

La matriz de salida **C** tiene tantas filas como salidas requeridas y tantas columnas como estados. La matriz de transmisión directa **D** tiene el mismo número de columnas que la matriz de entrada **B** y tantas filas como la matriz de salida **C**.

En nuestro ejemplo estamos interesados en los 6 estados, desplazamientos y velocidades, por lo que la ecuación matricial de salida resulta, donde **C** es la matriz identidad y **D** se asume que es cero:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (69)$$

Expandiendo, las ecuaciones matriciales se convierten en:

$$y_1 = x_1 \qquad (= z_1) \quad (70)$$

$$y_2 = x_2 \qquad (= \dot{z}_1) \quad (71)$$

$$y_3 = x_3 \qquad (= z_2) \quad (72)$$

$$y_4 = x_4 \qquad (= \dot{z}_2) \quad (73)$$

$$y_5 = x_5 \qquad (= z_3) \quad (74)$$

$$y_6 = x_6 \qquad (= \dot{z}_3) \quad (75)$$

Si nosotros sólo estamos interesados en los tres desplazamientos y no en las tres velocidades, la ecuación de salida sería, suponiendo que **D** es cero:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} + (0)(1) \quad (76)$$

Expandiendo:

$$y_1 = x_1 \qquad (= z_1) \quad (77)$$

$$y_2 = x_3 \qquad (= z_2) \quad (78)$$

$$y_3 = x_5 \qquad (= z_3) \quad (79)$$

Por otro lado, si las salidas son combinaciones lineales de los estados, como en un problema de sistemas de control, la ecuación de salida podría ser similar (donde a, b y c son escalares), suponiendo que **D** es cero:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & a & 0 & b & 0 \\ c & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} + (0)(1) \qquad (80)$$

Expandiendo:

$$y_1 = ax_3 + bx_5 \qquad (= az_2 + bz_3) \qquad (81)$$

$$y_2 = cx_1 + x_3 \qquad (= cz_1 + z_2) \qquad (82)$$

$$y_3 = x_1 \qquad (= z_1) \qquad (83)$$

$$y_4 = x_4 \qquad (= \dot{z}_2) \qquad (84)$$

Si se aplica una sola fuerza y se desea una sola salida (SISO), por ejemplo, una fuerza aplicada a la masa 1 y el desplazamiento de salida en la masa 3, suponiendo que **D** es cero:

$$y = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} + (0)(1) \qquad (85)$$

Con todas las posibles variaciones de la ecuación de salida, la ecuación de estado nunca cambia; es siempre:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u \qquad (86)$$

5.3.5. **Valores y vectores propios complejos – Forma del espacio de estado.**

El análisis más básico que uno puede realizar en un sistema dinámico es resolver sus valores propios (frecuencias naturales) y vectores propios (modos de vibración).

Empezamos por proponer que hay un conjunto de condiciones iniciales tales que si el sistema es liberado con dicho conjunto, el sistema responderá en uno de sus modos de vibración naturales. Para ello, ponemos la función forzada a cero y escribimos la ecuación de movimiento homogénea del espacio de estado:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} \tag{87}$$

Definimos el movimiento en un modo principal como:

$$\mathbf{x_i} = \mathbf{x_{mi}} e^{\lambda_i t} \tag{88}$$

Donde:

λ_i es el i-ésimo valor propio, la frecuencia natural del i-ésimo modo de vibración
x_i es el vector de estados de la i-ésima frecuencia.
x_{mi} es el i-ésimo vector propio, el modo natural para el i-ésimo modo.

Para nuestro tdof (z_1 a z_3), sistema de seis estados (x_1 a x_6), para el i-ésimo valor propio y vector propio, la ecuación aparecería como:

$$\begin{bmatrix} z_{1i} \\ \dot{z}_{1i} \\ z_{2i} \\ \dot{z}_{2i} \\ z_{3i} \\ \dot{z}_{3i} \end{bmatrix} = \begin{bmatrix} x_{1i} \\ x_{2i} \\ x_{3i} \\ x_{4i} \\ x_{5i} \\ x_{6i} \end{bmatrix} = \mathbf{x_{mi}} e^{\lambda_i t} = \begin{bmatrix} x_{m1i} \\ x_{m2i} \\ x_{m3i} \\ x_{m4i} \\ x_{m5i} \\ x_{m6i} \end{bmatrix} e^{\lambda_i t} \tag{89}$$

Derivando la ecuación modal de desplazamiento anterior se obtiene la ecuación modal de velocidad:

$$\dot{\mathbf{x}}_{\mathbf{mi}} = \frac{\mathrm{d}}{\mathrm{d}t}\left[\mathbf{x_{mi}} e^{\lambda t}\right] = \lambda \mathbf{x_{mi}} e^{\lambda t} \tag{90}$$

Sustituyendo en la ecuación de estado y cancelando el término exponencial conduce a:

$$\begin{gathered} \dot{\mathbf{x}} = \mathbf{A}\mathbf{x} \\ \lambda \mathbf{x_{mi}} e^{\lambda t} = \mathbf{A}\mathbf{x_{mi}} e^{\lambda t} \\ \lambda \mathbf{x_{mi}} = \mathbf{A}\mathbf{x_{mi}} \\ (\lambda \mathbf{I} - \mathbf{A})\mathbf{x_{mi}} = \mathbf{0} \end{gathered} \tag{91a-d}$$

La ecuación (91c) se conoce como "problema de valor propio" clásico. Si x_{mi} es distinto de cero en (91d), la solución existe solo si el siguiente determinante es cero:

$$|(\lambda \mathbf{I} - \mathbf{A})| = 0 \tag{92}$$

Cogiendo la matriz del sistema **A** de la ecuación (63) e introduciéndola en (91):

$$(\lambda\mathbf{I}-\mathbf{A})=\lambda\mathbf{I}-\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{-k_1}{m_1} & \frac{-c_1}{m_1} & \frac{k_1}{m_1} & \frac{c_1}{m_1} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{k_1}{m_2} & \frac{c_1}{m_2} & \frac{-(k_1+k_2)}{m_2} & \frac{-(c_1+c_2)}{m_2} & \frac{k_2}{m_2} & \frac{c_2}{m_2} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{k_2}{m_3} & \frac{c_2}{m_3} & \frac{-k_2}{m_3} & \frac{-c_2}{m_3} \end{bmatrix} \quad (93)$$

En el caso más general, los valores y vectores propios suelen ser complejos y suelen aparecer por parejas conjugadas. El término λn1 representa el primer valor propio complejo de cualquiera de los tres conjuntos de valores propios de nuestro ejemplo. El término λn2 se utiliza para describir el segundo valor propio del conjunto, y su conjugado se representa como: $\lambda_{n2}=\lambda_{n1}^{*}$, donde "*" indica un complejo conjugado. Las partes real e imaginaria vienen definidas por σ_{nx} y ω_{nx}, respectivamente:

$$\lambda_{n1}=\sigma_{n1}+j\omega_{n1}$$
$$\lambda_{n2}=\lambda_{n1}^{*}=\sigma_{n1}-j\omega_{n1} \quad (94)$$

La Figura 26 describe gráficamente las componentes de un valor propio complejo. La figura muestra dos valores propios complejos conjugados (polos) en la mitad izquierda del plano con el símbolo "x". las partes reales de los dos valores propios son las mismas y vienen dadas por el símbolo σ, ambas partes imaginarias tienen una distancia al origen de ω, denominada frecuencia natural amortiguada. La distancia radial del origen a los polos viene dada por ω_n, y se denomina frecuencia natural no amortiguada. El ángulo entre el eje imaginario y la línea del origen al polo se utiliza para definir la cantidad de amortiguamiento del modo, denominado como relación de amortiguación o porcentaje de amortiguación crítica, ζ. Si σ = 0, θ = 0 y no hay amortiguamiento, por tanto $\omega = \omega_n$.

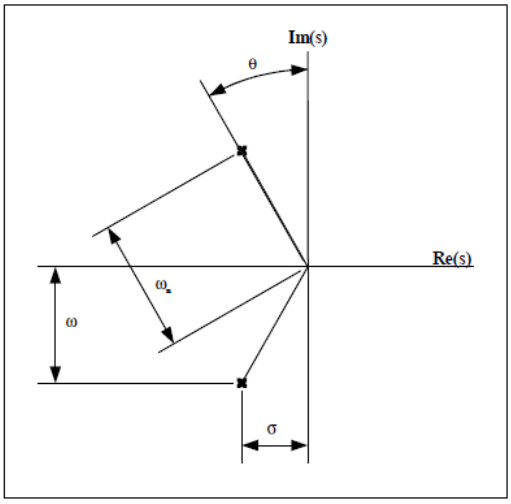


Figura 26. Nomenclatura del valor propio complejo (polo) en el plano complejo. (Michael R. Hatch, 2001)

Haciendo referencia a la Figura 26 para la definición de θ, la ecuación para calcular ζ para un modo a partir de las componentes real e imaginaria del valor propio es:

$$\zeta = \sin\theta = \sin\left(\tan^{-1}\left(\frac{\mathrm{Re}(\lambda)}{\mathrm{Im}(\lambda)}\right)\right) = \sin\left(\tan^{-1}\left(\frac{\sigma}{\omega}\right)\right) \qquad (95)$$

6. MODELADO DEL SISTEMA DE SUSPENSIÓN.

6.1. Modelo de un grado de libertad.

La Figura 27 muestra el modelo más simple para el análisis de las vibraciones verticales de un automóvil. Este modelo se conoce también como modelo de 1/8 de vehículo. El modelo de 1 grado de libertad que muestra la figura 27 podría asemejarse a la suspensión independiente de un cuarto de vehículo. La masa m_s representa una cuarta parte del cuerpo del automóvil, la cuál está montada sobre una suspensión compuesta por un resorte helicoidal lineal k_s en paralelo con un amortiguador hidráulico lineal c_s. Cuando m_s vibra en una posición tal como en la Figura 27, su diagrama de cuerpo libre es como muestra la Figura 27(c).

El sistema de suspensión de un automóvil es excitado armónicamente por la superficie de la carretera a través de la suspensión. Sea *y* el desplazamiento de la base, y x_s el desplazamiento de la masa suspendida con respecto a su posición de equilibrio estático en el tiempo, entonces el alargamiento neto del resorte es $x_s - y$, y la velocidad relativa entre los dos extremos del amortiguador es $\dot{x}_s - \dot{y}$.

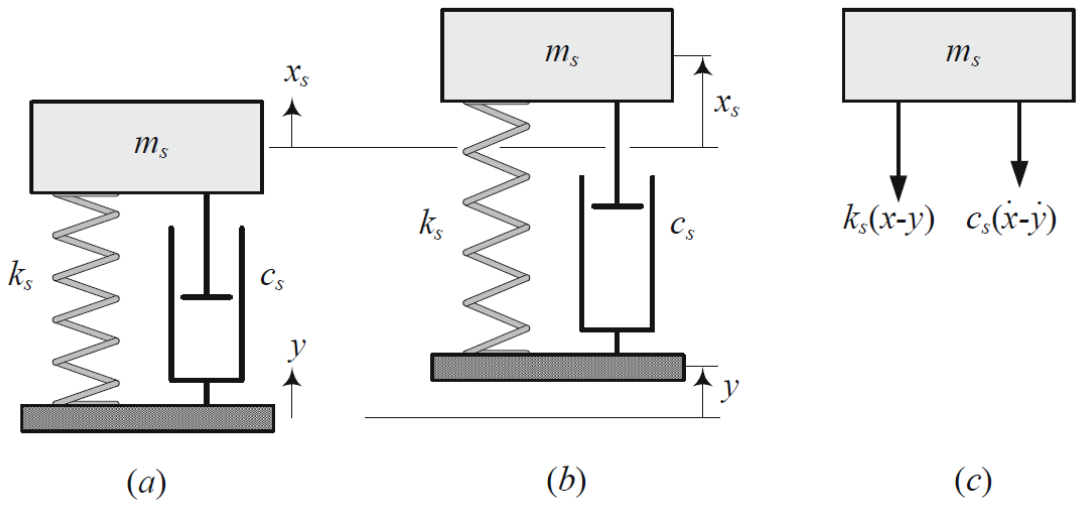


Figura 27. Modelo de 1 grado de libertad o de 1/8 de vehículo (a-b) con su diagrama de cuerpo libre (c). (Reza N. Jazar, 2014)

Del diagrama de cuerpo libre que se muestra en la Figura 27(c), y aplicando la segunda ley de Newton, se obtiene la ecuación de movimiento, la cual será una ecuación diferencial de la forma:

$$m_s\ddot{x}_s = -k_s(x_s - y) - c_s(\dot{x}_s - \dot{y}) \qquad (96)$$

La cual puede ser simplificada a la siguiente ecuación, donde se separan las variables de entrada "*y*" y la de salida "*x*"

$$m_s\ddot{x}_s + c_s\dot{x}_s + k_s x_s = k_s y + c_s\dot{y} \qquad (97)$$

Donde el término de la derecha de la igualdad representar la función de entrada o fuerza excitadora, por tanto

$$m_s\ddot{x}_s + c_s\dot{x}_s + k_s x_s = F \qquad (98)$$

6.2. Modelo de dos grados de libertad.

La Figura 28 (a)-(c) muestra el equilibrio, movimiento y diagrama de cuerpo libre de un sistema de dos grados de libertad. El diagrama de cuerpo libre está representado en base a la suposición

$$x_s > x_u > y \qquad (99)$$

Cuando en el modelo simple de un cuarto de vehículo se considera, además de la masa suspendida "m_s", la masa no suspendida "m_u", y la rigidez del neumático "k_u", el modelo se convierte en uno de 2 grados de libertad, tal y como muestra la Figura 28(a)-(c).

Básicamente, este modelo de 2 g.d.l. consiste en una masa suspendida "m_s" que se apoya sobre el sistema primario de suspensión, quedando así unida a la masa no suspendida del eje de la rueda, "m_u". La suspensión tendrá determinados unos parámetros de rigidez "k_s" y de amortiguación "c_s". El neumático se representa mediante un simple muelle de rigidez "k_u", aunque a menudo éste suele ir acompañado de un pequeño amortiguador que representa el aporte amortiguante intrínseco a la naturaleza visco-elástica de la goma del neumático "c_u". Debido a que el efecto amortiguante del neumático es mucho más pequeño que el de la suspensión, éste se suele despreciar.

Aplicando la segunda ley de Newton, del diagrama de cuerpo libre se obtiene que la ecuación de movimiento es un sistema de dos ecuaciones diferenciales de segundo orden de la forma:

$$\begin{aligned} m_s\ddot{x}_s &= -c_s(\dot{x}_s - \dot{x}_u) - k_s(x_s - x_u) = 0 \\ m_u\ddot{x}_u &= c_s(\dot{x}_s - \dot{x}_u) + k_s(x_s - x_u) - c_u(\dot{x}_u - \dot{y}) - k_u(x_u - y) = 0 \end{aligned} \qquad (100a,b)$$

Normalmente las ecuaciones de movimiento para un sistema lineal se reorganizan en forma matricial para sacar provecho del cálculo matricial.

$$[\mathbf{M}]\{\ddot{\mathbf{x}}\} + [\mathbf{C}]\{\dot{\mathbf{x}}\} + [\mathbf{K}]\{\mathbf{x}\} = \{\mathbf{F}\} \qquad (101)$$

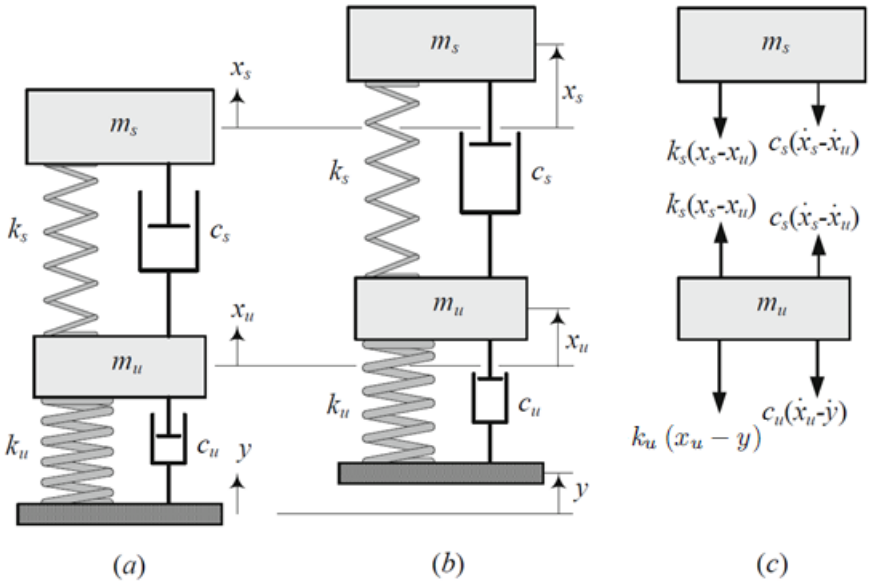


Figura 28. Modelo de dos grados de libertad o modelo de ¼ de vehículo (a-b) y su diagrama de cuerpo libre. (Reza N. Jazar, 2014)

Reordenando las ecuaciones 100(a)-(b) se obtiene el siguiente conjunto de ecuaciones:

$$\begin{bmatrix} m_s & 0 \\ 0 & m_u \end{bmatrix} \cdot \begin{Bmatrix} \ddot{x}_s \\ \ddot{x}_u \end{Bmatrix} + \begin{bmatrix} c_s & -c_s \\ -c_s & c_s + c_u \end{bmatrix} \cdot \begin{Bmatrix} \dot{x}_s \\ \dot{x}_u \end{Bmatrix} + \begin{bmatrix} k_s & -k_s \\ -k_s & k_s + k_u \end{bmatrix} \cdot \begin{Bmatrix} x_s \\ x_u \end{Bmatrix} = \begin{Bmatrix} 0 \\ k_u \cdot y + c_u \dot{y} \end{Bmatrix} = \begin{Bmatrix} 0 \\ F \end{Bmatrix} \quad (102)$$

6.3. Modelo de tres grados de libertad.

La Figura 29 muestra un modelo de cuarto de vehículo al cual se le ha añadido el conductor. El conductor se ha modelado mediante una masa m_d sobre un cojín lineal encima de la masa suspendida m_s. la rigidez k_d y el amortiguamiento c_d representan el asiento del conductor.

Siguiendo el mismo método de antes, se llega a que las ecuaciones de movimiento son de la forma:

$$\begin{bmatrix} m_u & 0 & 0 \\ 0 & m_s & 0 \\ 0 & 0 & m_d \end{bmatrix} \begin{Bmatrix} \ddot{x}_u \\ \ddot{x}_s \\ \ddot{x}_d \end{Bmatrix} + \begin{bmatrix} c_s + c_u & -c_s & 0 \\ -c_s & c_d + c_s & -c_d \\ 0 & -c_d & c_d \end{bmatrix} \begin{Bmatrix} \dot{x}_u \\ \dot{x}_s \\ \dot{x}_d \end{Bmatrix} + \begin{bmatrix} k_s + k_u & -k_s & 0 \\ -k_s & k_d + k_s & -k_d \\ 0 & -k_d & k_d \end{bmatrix} \begin{Bmatrix} x_u \\ x_s \\ x_d \end{Bmatrix} = \begin{Bmatrix} c_u \cdot \dot{y} + k_u \cdot y \\ 0 \\ 0 \end{Bmatrix} = \begin{Bmatrix} F \\ 0 \\ 0 \end{Bmatrix} \quad (103)$$

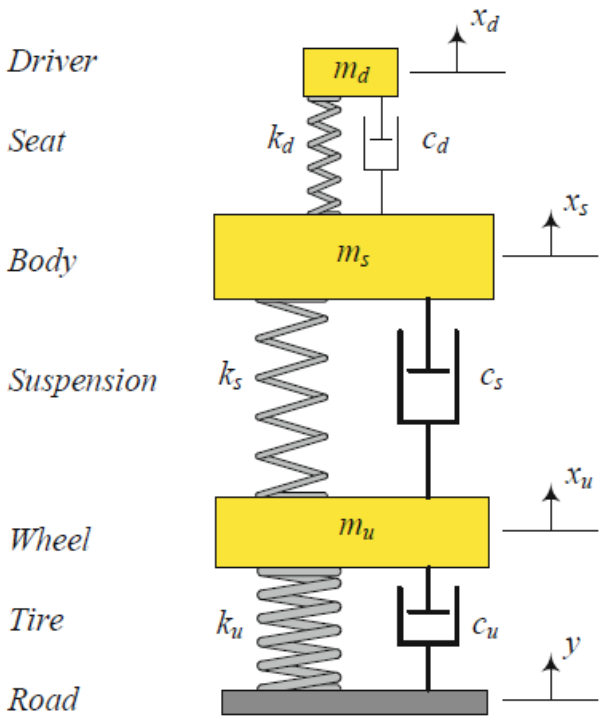


Figura 29. Modelo de ¼ de vehículo con conductor (Reza N. Jazar, 2014)

6.4. Modelo de cuatro grados de libertad.

Cuando en el modelo simple de un cuarto de vehículo se tiene en cuenta la tipología del sistema de suspensión, diferenciando entre suspensión independiente y suspensión de eje rígido, se puede abordar el estudio de las funciones de transferencia con modelos de mayor número de grados de libertad. Este aumento de grados de libertad permite, a su vez, considerar el análisis del movimiento vertical-transversal o vertical-longitudinal. En el presente apartado abordaremos el estudio de las ecuaciones del movimiento para cada uno de los modelos de 4 g.d.l., en función del tipo de suspensión.

6.4.1. **Modelo de 4 g.d.l. con suspensión independiente.**

6.4.1.1. Medio automóvil y modo de balanceo.

Para examinar y optimizar la vibración de balanceo, podemos utilizar un modelo vibratorio de medio vehículo. La Figura 30 muestra un modelo de medio vehículo con cuatro grados de libertad y suspensión independiente. Este modelo incluye el movimiento de bailoteo x, el movimiento de balanceo φ, el desplazamiento vertical de las ruedas x_1 y x_2 y las excitaciones independientes de la carretera y_1 y y_2.

Los grados de libertad del modelo serán:

- Desplazamiento vertical de la rueda izquierda, x_1.
- Desplazamiento vertical de la rueda derecha, x_2.
- Desplazamiento vertical de la carrocería, x.

– Ángulo de balanceo de la carrocería, φ.

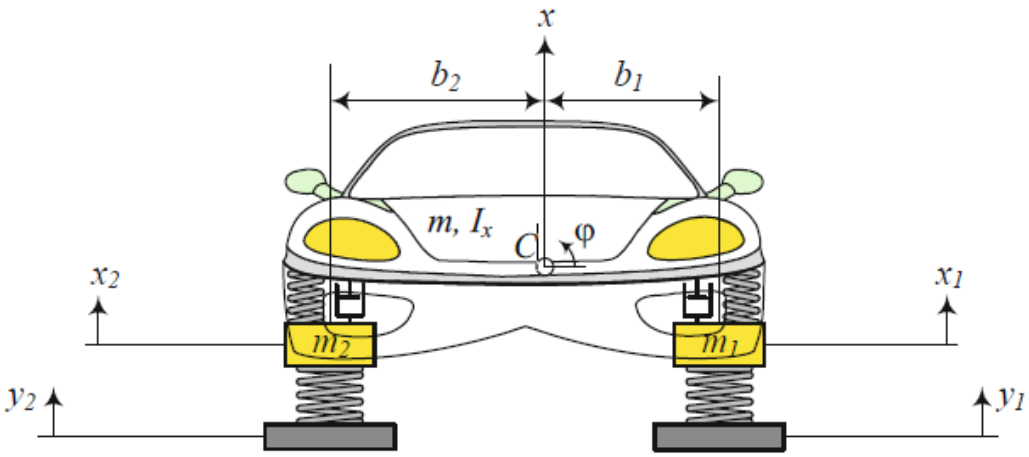


Figura 30. Modelo vibratorio de medio automóvil. (Reza N. Jazar, 2014)

Las ecuaciones de movimiento se encuentran aplicando la segunda Ley de Newton. Para el modelo vibratorio de medio automóvil de un vehículo son:

$$m\ddot{x} + c(\dot{x} - \dot{x}_1 + b_1\dot{\varphi}) + c(\dot{x} - \dot{x}_2 - b_2\dot{\varphi}) + k(x - x_1 + b_1\varphi) + k(x - x_2 - b_2\varphi) = 0 \qquad (104)$$

$$I_x\ddot{\varphi} + b_1c(\dot{x} - \dot{x}_1 + b_1\dot{\varphi}) - b_2c(\dot{x} - \dot{x}_2 - b_2\dot{\varphi}) + b_1k(x - x_1 + b_1\varphi) - b_2k(x - x_2 - b_2\varphi) + k_R\varphi = 0 \qquad (105)$$

$$m_1\ddot{x}_1 - c(\dot{x} - \dot{x}_1 + b_1\dot{\varphi}) - k(x - x_1 + b_1\varphi) + k_t(x_1 - y_1) = 0 \qquad (106)$$

$$m_2\ddot{x}_2 - c(\dot{x} - \dot{x}_2 - b_2\dot{\varphi}) - k(x - x_2 - b_2\varphi) + k_t(x_2 - y_2) = 0 \qquad (107)$$

El modelo de medio automóvil puede ser diferente para la mitad delantera y la mitad trasera debido al uso de diferentes suspensiones y distribución de masas. Además, en las mitades delantera y trasera se pueden utilizar diferentes barras antivuelco y diferente rigidez torsional.

La Figura 31 muestra un modelo vibratorio mejor del sistema. El cuerpo del vehículo se supone que es una barra rígida. Esta barra tiene una masa *m*, la cual es la mitad delantera o trasera de la masa total del cuerpo, y un momento de inercia longitudinal I_x, el cual es la mitad del momento de inercia total del cuerpo. Las ruedas izquierda y derecha tienen una masa m_1 y m_2 respectivamente, aunque normalmente suelen ser iguales. La rigidez de los neumáticos se indica mediante k_t. Debido a que el amortiguamiento de los neumáticos es mucho más pequeño que el amortiguamiento de los amortiguadores, éste se suele despreciar para una mayor simplicidad en los cálculos. La suspensión del vehículo tiene rigidez *k* y amortiguamiento *c* para las ruedas izquierda y derecha. Es común hacer que la suspensión de la izquierda y de la derecha sean iguales. Por eso, su rigidez y amortiguamiento son iguales. Sin embargo, el modelo de medio automóvil tiene diferente *k*, *c* y k_t para la parte delantera o trasera.

El vehículo puede tener también una barra antivuelco con una rigidez torsional k_R en la parte delantera o trasera. Utilizando un modelo simple, la barra antivuelco proporciona un par de torsión M_R proporcional al ángulo de balanceo φ.

$$M_R = -k_R \varphi \tag{108}$$

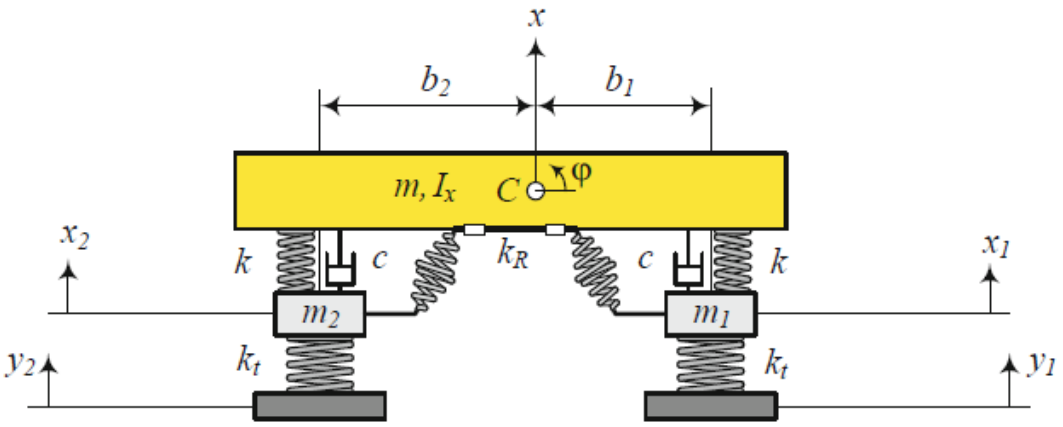


Figura 31. Modelo de medio vehículo con barra de torsión. (Reza N. Jazar, 2014)

Sin embargo, un modelo mejor del efecto de la barra antivuelco es

$$M_R = -k_R \left(\varphi - \frac{x_1 - x_2}{\omega} \right) \tag{109}$$

El conjunto de ecuaciones de movimiento puede ser reordenado en forma matricial de la forma

$$[\mathbf{M}]\{\ddot{\mathbf{x}}\} + [\mathbf{C}]\{\dot{\mathbf{x}}\} + [\mathbf{K}]\{\mathbf{x}\} = \{\mathbf{F}\} \tag{110}$$

donde,

$$\{x\} = \begin{bmatrix} x \\ \varphi \\ x_1 \\ x_2 \end{bmatrix} \tag{111}$$

$$[M] = \begin{bmatrix} m & 0 & 0 & 0 \\ 0 & I_x & 0 & 0 \\ 0 & 0 & m_1 & 0 \\ 0 & 0 & 0 & m_2 \end{bmatrix} \tag{112}$$

$$[C] = \begin{bmatrix} 2c & cb_1 - cb_2 & -c & -c \\ cb_1 - cb_2 & cb_1^2 + cb_2^2 & -cb_1 & cb_2 \\ -c & -cb_1 & c & 0 \\ -c & cb_2 & 0 & c \end{bmatrix} \tag{113}$$

$$[K]=\begin{bmatrix} 2k & kb_1-kb_2 & -k & -k \\ kb_1-kb_2 & kb_1^2+kb_2^2+k_R & -kb_1 & kb_2 \\ -k & -kb_1 & k+kt & 0 \\ -k & kb_2 & 0 & k+kt \end{bmatrix} \quad (114)$$

$$\begin{bmatrix} 0 \\ 0 \\ y_1k_t \\ y_2k_t \end{bmatrix}=\begin{bmatrix} 0 \\ 0 \\ F_1 \\ F_2 \end{bmatrix} \quad (115)$$

En la Tabla 2 se muestra a modo de resumen todos los parámetros que intervienen en el modelo de medio vehículo y modo de balanceo, y que además aparecen en la Figura 31.

Tabla 2 – Parámetros del modelo vibratorio de medio vehículo y modelo de balanceo.

Parámetro	*Definición*
m	Masa suspendida (masa total de la carrocería)
I_x	Momento de Inercia de la carrocería alrededor de su eje longitudinal
φ	Ángulo de balanceo de la carrocería
m_1	Masa no suspendida izquierda (masa de la rueda izquierda)
m_2	Masa no suspendida derecha (masa de la rueda derecha)
c	Coeficiente de amortiguación de la suspensión
k	Coeficiente de rigidez de la suspensión
kt	Coeficiente de rigidez de los neumáticos
k_R	Coeficiente de rigidez de la barra antivuelco
y_1	Amplitud de la excitación de la rueda izquierda
y_2	Amplitud de la excitación de la rueda derecha
b_1	Distancia del centro de gravedad a la suspensión izquierda
b_2	Distancia del centro de gravedad a la suspensión derecha

Este mismo modelo puede ser también utilizado para realizar el análisis del comportamiento vertical-longitudinal como se verá en el siguiente apartado. En este caso las variables k_{t1}, k_1, c_1, y m_1 tienen el mismo significado, pero referido al eje delantero, mientras que k_{t2}, k_2, c_2 y m_2, harán referencia al eje trasero. En cuanto al momento de inercia I_x, ahora se referirá al momento de inercia respecto al eje transversal del vehículo I_y, y las distancias b_1 y b_2, representarán las distancias del centro de gravedad al eje delantero y eje trasero respectivamente.

6.4.1.2. Modelo de bicicleta y modo de cabeceo.

El modelo de un cuarto de vehículo es excelente para examinar y optimizar el movimiento de bailoteo del cuerpo debido a las vibraciones. Sin embargo, podemos ampliar el modelo vibratorio de un vehículo para introducir el cabeceo y otros modos de vibración. La Figura 32 muestra el modelo vibratorio de bicicleta de un vehículo. Este modelo incluye el movimiento de bailoteo x, el ángulo de cabeceo θ, el movimiento vertical de las ruedas x_1 y x_2, y las excitaciones independientes de la carretera y_1 y y_2.

Los grados de libertad del modelo serán:

- Desplazamiento vertical de la rueda delantera, x_1.
- Desplazamiento vertical de la rueda trasera, x_2.
- Desplazamiento vertical de la carrocería, x.
- Ángulo de cabeceo de la carrocería, θ.

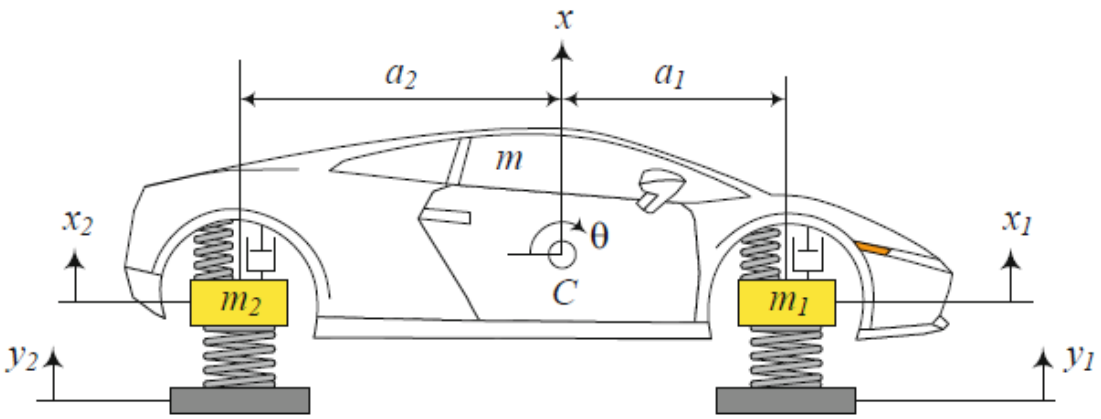


Figura 32. Modelo vibratorio de bicicleta de un vehículo. (Reza N. Jazar, 2014)

Las ecuaciones de movimiento para el modelo de bicicleta de un vehículo son:

$$m\ddot{x} + c_1\left(\dot{x} - \dot{x}_1 - a_1\dot{\theta}\right) + c_2\left(\dot{x} - \dot{x}_2 + a_2\dot{\theta}\right) + k_1\left(x - x_1 - a_1\theta\right) + k_2\left(x - x_2 + a_2\theta\right) = 0 \qquad (116)$$

$$\begin{aligned} &I_y\ddot{\theta} - a_1c_1\left(\dot{x} - \dot{x}_1 - a_1\dot{\theta}\right) + a_2c_2\left(\dot{x} - \dot{x}_2 + a_2\dot{\theta}\right) \\ &- a_1k_1\left(x - x_1 - a_1\theta\right) + a_2k_2\left(x - x_2 + a_2\theta\right) = 0 \end{aligned} \qquad (117)$$

$$m_1\ddot{x}_1 - c_1\left(\dot{x} - \dot{x}_1 - a_1\dot{\theta}\right) - k_1\left(x - x_1 - a_1\theta\right) + k_{t1}\left(x_1 - y_1\right) = 0 \qquad (118)$$

$$m_2\ddot{x}_2 - c_2\left(\dot{x} - \dot{x}_2 + a_2\dot{\theta}\right) - k_2\left(x - x_2 + a_2\theta\right) + k_{t2}\left(x_2 - y_2\right) = 0 \qquad (119)$$

La Figura 33 muestra el modelo vibratorio del sistema. El cuerpo del vehículo se supone que es una barra rígida. Esta barra tiene una masa *m*, la cual es la mitad de la masa total de la carrocería, y un momento de inercia transversal I_y, el cual es la mitad del momento de inercia transversal total de la masa de la carrocería. Las ruedas delantera y trasera tienen una masa m_1 y m_2 respectivamente. La rigidez de los neumáticos está indica por los parámetros kt_1 y kt_2. Esta diferencia se debe a que los neumáticos traseros normalmente son más rígidos que los delanteros, aunque en un modelo más simple podemos asumir que $kt_1 = kt_2$. El amortiguamiento de los neumáticos es mucho más pequeño que la amortiguación de los amortiguadores, por tanto, podemos ignorar el amortiguamiento de los neumáticos para unos cálculos más simples.

Como recordatorio, la definición de los parámetros y variables empleados se indican en la Tabla 3.

Tabla 3 – Parámetros del modelo vibratorio de bicicleta de un vehículo.

Parámetro	Significado
m	Masa suspendida, mitad de la masa de la carrocería
m_1	Masa no suspendida delantera, masa de la rueda delantera
m_2	Masa no suspendida trasera, masa de la rueda trasera
x	Desplazamiento vertical de la carrocería
x_1	Desplazamiento vertical de la rueda delantera
x_2	Desplazamiento vertical de la rueda trasera
θ	Ángulo de balanceo de la carrocería
y_1	Amplitud de la excitación de la rueda delantera
y_2	Amplitud de la excitación de la rueda trasera
I_y	Mitad del momento de inercia de la carrocería alrededor de su eje transversal
a_1	Distancia del centro de gravedad al eje delantero
a_2	Distancia del centro de gravedad al eje trasero

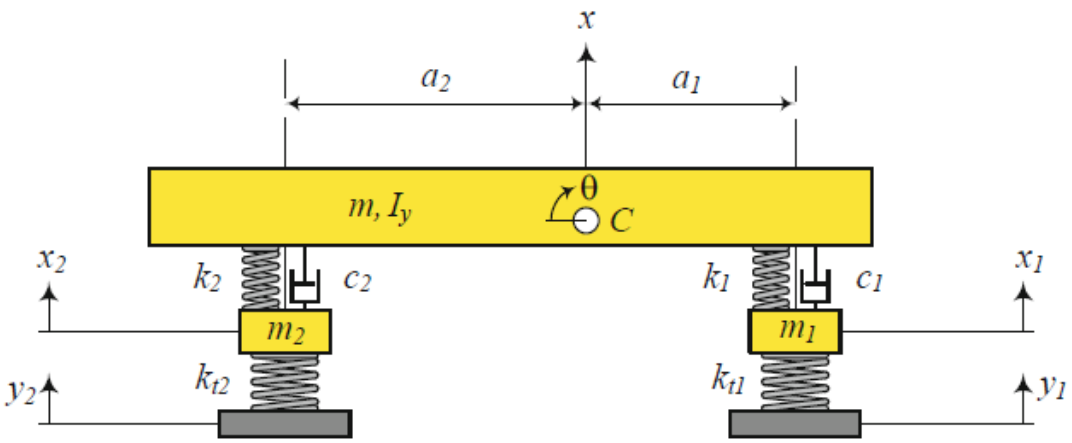


Figura 33. Modelo de bicicleta para las vibraciones de un vehículo. (Reza N. Jazar, 2014)

El conjunto de ecuaciones de movimiento (104)-(107) pueden ser reordenadas en forma matricial de forma

$$[\mathbf{M}]\{\ddot{\mathbf{x}}\} + [\mathbf{C}]\{\dot{\mathbf{x}}\} + [\mathbf{K}]\{\mathbf{x}\} = \{\mathbf{F}\} \qquad (120)$$

donde,

$$\{\boldsymbol{x}\} = \begin{bmatrix} x \\ \theta \\ x_1 \\ x_2 \end{bmatrix} \qquad (121)$$

$$[\boldsymbol{M}] = \begin{bmatrix} m & 0 & 0 & 0 \\ 0 & I_y & 0 & 0 \\ 0 & 0 & m_1 & 0 \\ 0 & 0 & 0 & m_2 \end{bmatrix} \quad (122)$$

$$[\boldsymbol{C}] = \begin{bmatrix} c_1 + c_2 & c_2 a_2 - c_1 a_1 & -c_1 & -c_2 \\ c_2 a_2 - c_1 a_1 & c_1 a_1^2 + c_2 a_2^2 & c_1 a_1 & -c_2 a_2 \\ -c_1 & c_1 a_1 & c_1 & 0 \\ -c_2 & -c_2 a_2 & 0 & c_2 \end{bmatrix} \quad (123)$$

$$[\boldsymbol{K}] = \begin{bmatrix} k_1 + k_2 & k_2 a_2 - k_1 a_1 & -k_1 & -k_2 \\ k_2 a_2 - k_1 a_1 & k_1 a_1^2 + k_2 a_2^2 & k_1 a_1 & -k_2 a_2 \\ -k_1 & k_1 a_1 & k_1 + kt_1 & 0 \\ -k_2 & -k_2 a_2 & 0 & k_2 + kt_2 \end{bmatrix} \quad (124)$$

$$\{\boldsymbol{F}\} = \begin{bmatrix} 0 \\ 0 \\ y_1 kt_1 \\ y_2 kt_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ F_1 \\ F_2 \end{bmatrix} \quad (125)$$

6.4.2. **Modelo de 4 g.d.l. con suspensión de eje rígido.**

Este modelo se utiliza para analizar el comportamiento lateral de los vehículos que incorporan ejes rígidos. En este caso los grados de libertad del modelo serán:

- Desplazamiento vertical del eje, z1.
- Ángulo de balanceo del eje, θ1.
- Desplazamiento vertical de la carrocería, z2.
- Ángulo de balanceo de la carrocería, θ2.

En este modelo de eje rígido se puede analizar, a diferencia del de suspensión independiente, la influencia de la diferencia existente entre la vía del eje (a1 + a2) y la distancia entre los puntos de anclaje de las suspensiones (b1 y b2).

Aplicando la segunda Ley de Newton, las ecuaciones de movimiento son:

$$\begin{aligned} & m_1 \ddot{z}_1 - c(\dot{z}_2 - \dot{z}_1 - a_1\dot{\varphi}_1 + b_1\dot{\varphi}_2) - c(\dot{z}_2 - \dot{z}_1 + a_2\dot{\varphi}_1 - b_2\dot{\varphi}_2) \\ & - k(z_2 - z_1 - a_1\varphi_1 + b_1\varphi_2) - k(z_2 - z_1 + a_2\varphi_1 - b_2\varphi_2) \\ & + kt(z_1 - y_1 + a_1\varphi_1) + kt(z_1 - y_2 - a_2\varphi_1) = 0 \end{aligned} \quad (126)$$

$$\begin{aligned} & I_{x1}\ddot{\varphi}_1 - a_1 c(\dot{z}_2 - \dot{z}_1 - a_1\dot{\varphi}_1 + b_1\dot{\varphi}_2) + a_2 c(\dot{z}_2 - \dot{z}_1 + a_2\dot{\varphi}_1 - b_2\dot{\varphi}_2) \\ & - a_1 k(z_2 - z_1 - a_1\varphi_1 + b_1\varphi_2) + a_2 k(z_2 - z_1 + a_2\varphi_1 - b_2\varphi_2) \\ & + a_1 kt(z_1 - y_1 + a_1\varphi_1) - a_2 kt(z_1 - y_2 - a_2\varphi_1) = 0 \end{aligned} \quad (127)$$

$$m_2\ddot{z}_2 + c(\dot{z}_2 - \dot{z}_1 - a_1\dot{\varphi}_1 + b_1\dot{\varphi}_2) + c(\dot{z}_2 - \dot{z}_1 + a_2\dot{\varphi}_1 - b_2\dot{\varphi}_2) + k(z_2 - z_1 - a_1\varphi_1 + b_1\varphi_2) + k(z_2 - z_1 + a_2\varphi_1 - b_2\varphi_2) = 0 \quad (128)$$

$$I_{x2}\ddot{\varphi}_2 + b_1c(\dot{z}_2 - \dot{z}_1 - a_1\dot{\varphi}_1 + b_1\dot{\varphi}_2) - b_2c(\dot{z}_2 - \dot{z}_1 + a_2\dot{\varphi}_1 - b_2\dot{\varphi}_2) + b_1k(z_2 - z_1 - a_1\varphi_1 + b_1\varphi_2) - b_2k(z_2 - z_1 + a_2\varphi_1 - b_2\varphi_2) = 0 \quad (129)$$

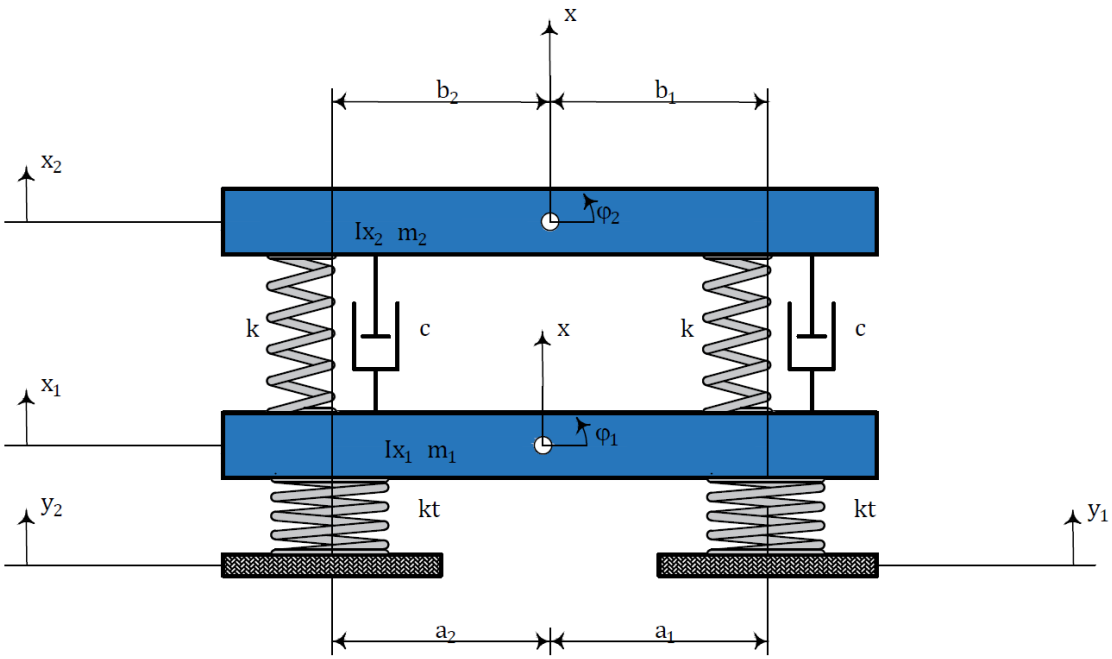


Figura 34. Modelo de medio vehículo con suspensión rígida.

El conjunto de ecuaciones de movimiento (126)-(129) pueden ser reordenadas en forma matricial de forma

$$[\mathbf{M}]\{\ddot{\mathbf{x}}\} + [\mathbf{C}]\{\dot{\mathbf{x}}\} + [\mathbf{K}]\{\mathbf{x}\} = \{\mathbf{F}\} \quad (130)$$

donde,

$$\{x\} = \begin{Bmatrix} x_1 \\ \varphi_1 \\ x_2 \\ \varphi_2 \end{Bmatrix} \quad (131)$$

$$[M] = \begin{bmatrix} m_1 & 0 & 0 & 0 \\ 0 & I_{x1} & 0 & 0 \\ 0 & 0 & m_2 & 0 \\ 0 & 0 & 0 & I_{x2} \end{bmatrix} \quad (132)$$

$$[C]=\begin{bmatrix} 2c & a_1c-a_2c & -2c & b_2c-b_1c \\ a_1c-a_2c & a_1^2c+a_2^2c & a_2c-a_1c & -a_1b_1c-a_2b_2c \\ -2c & a_2c-a_1c & 2c & b_1c-b_2c \\ b_2c-b_1c & -a_1b_1c-a_2b_2c & b_1c-b_2c & b_1^2c+b_2^2c \end{bmatrix} \quad (133)$$

$$[K]=\begin{bmatrix} 2k+2kt & kta_1-kta_2+ka_1-ka_2 & -2k & kb_2-kb_1 \\ kta_1-kta_2+ka_1-ka_2 & kta_1^2+kta_2^2+ka_1^2+ka_2^2 & ka_2-ka_1 & -ka_1b_1-ka_2b_2 \\ -2k & ka_2-ka_1 & 2k & kb_1-kb_2 \\ kb_2-kb_1 & -ka_1b_1-ka_2b_2 & kb_1-kb_2 & kb_1^2+kb_2^2 \end{bmatrix} \quad (134)$$

$$\{F\}=\begin{Bmatrix} kty_1+kty_2 \\ -kta_1y_1+kta_2y_2 \\ 0 \\ 0 \end{Bmatrix}=\begin{Bmatrix} F_1+F_2 \\ -F_1a_1+F_2a_2 \\ 0 \\ 0 \end{Bmatrix} \quad (135)$$

Como recordatorio, la definición de los parámetros y variables empleados se indican en la Tabla 4.

Tabla 4 – Parámetros y variables empleados en el modelo de medio vehículo con suspensión de eje rígido

Parámetro	*Significado*
m_1	Masa del eje rígido (masa no suspendida)
m_2	Masa total de la carrocería (masa no suspendida)
φ_1	Ángulo de balanceo del eje
φ_2	Ángulo de balanceo de la carrocería
Ix_1	Momento de Inercia del eje rígido alrededor de su eje longitudinal
Ix_2	Momento de Inercia de la carrocería alrededor de su eje longitudinal
c	Coeficiente de amortiguación de la suspensión
k	Coeficiente de rigidez de la suspensión
kt	Coeficiente de rigidez del neumático
y_1	Amplitud de la excitación de la rueda izquierda
y_2	Amplitud de la excitación de la rueda derecha
a_1	Distancia del centro de gravedad a la rueda izquierda
a_2	Distancia del centro de gravedad a la rueda derecha
b_1	Distancia del centro de gravedad al punto de anclaje de la suspensión izquierda
b_2	Distancia del centro de gravedad al punto de anclaje de la suspensión derecha

6.5. Modelo de siete grados de libertad.

Como descripción final de la modelización del sistema de suspensión de un vehículo desde el punto de vista del análisis de la respuesta del mismo ante las diferentes fuentes de excitación, abordaremos aquí el caso más general del modelo de 7 grados de libertad. El modelo que se representa permite la distinción entre suspensión con eje rígido trasero y suspensiones independientes a las cuatro ruedas. El análisis se basa en el estudio del movimiento vertical-transversal y vertical-longitudinal de la masa suspendida (carrocería).

6.5.1. **Modelo de 7 g.d.l. con suspensión independiente.**

El modelo vibratorio general de un vehículo se llama modelo de *automóvil completo*. Este modelo se muestra en la Figura 35, e incluye el movimiento de bailoteo o desplazamiento vertical x, el movimiento de balanceo φ, el movimiento de cabeceo θ, el desplazamiento vertical de las ruedas x_1, x_2, x_3 y x_4 y las excitaciones independientes de la carretera y_1, y_2, y_3 y y_4.

La Figura 35 representa un modelo de 7 grados de libertad con suspensión independiente a las cuatro ruedas. Los grados de libertad del modelo serán los siguientes:

- Desplazamiento vertical de la rueda delantera izquierda, x_1.
- Desplazamiento vertical de la rueda delantera derecha, x_2.
- Desplazamiento vertical de la rueda trasera izquierda, x_4.
- Desplazamiento vertical de la rueda trasera derecha, x_3.
- Desplazamiento vertical de la masa suspendida (carrocería), x.
- Ángulo de balanceo de la carrocería, θ.
- Ángulo de cabeceo de la carrocería, φ.

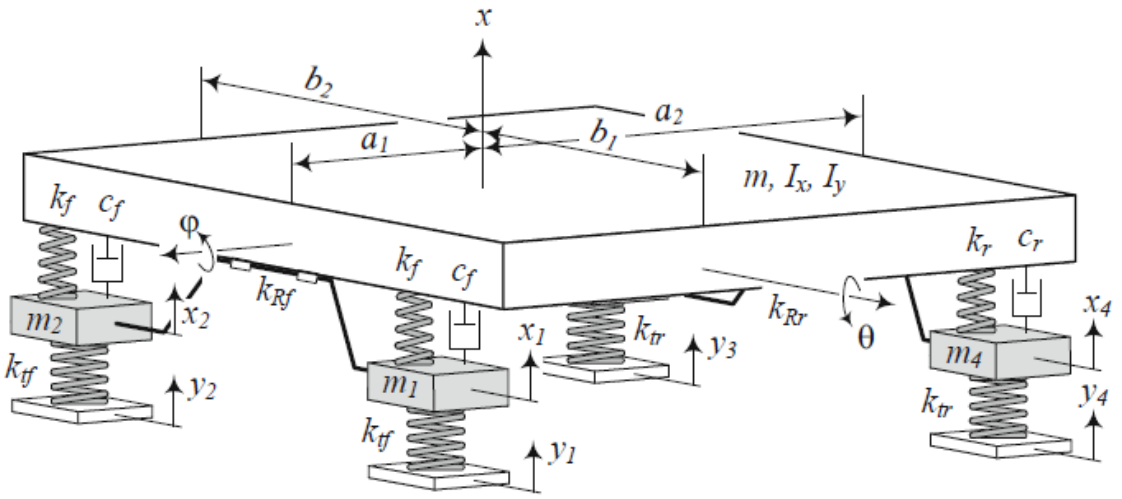


Figura 35. Modelo de vehículo completo. (Reza N. Jazar, 2014)

El modelo vibratorio de vehículo completo tiene siete grados de libertad con las siguientes ecuaciones de movimiento.

$$
\begin{aligned}
&m\ddot{x} + c_f\left(\dot{x} - \dot{x}_1 + b_1\dot{\varphi} - a_1\dot{\theta}\right) + c_f\left(\dot{x} - \dot{x}_2 - b_2\dot{\varphi} - a_1\dot{\theta}\right) \\
&+ c_r\left(\dot{x} - \dot{x}_3 - b_2\dot{\varphi} + a_2\dot{\theta}\right) + c_r\left(\dot{x} - \dot{x}_4 + b_1\dot{\varphi} + a_2\dot{\theta}\right) \\
&+ k_f\left(x - x_1 + b_1\varphi - a_1\theta\right) + k_f\left(x - x_2 - b_2\varphi - a_1\theta\right) \\
&+ k_r\left(x - x_3 - b_2\varphi + a_2\theta\right) + k_r\left(x - x_4 + b_1\varphi + a_2\theta\right) = 0
\end{aligned}
\qquad (136)
$$

$$
\begin{aligned}
&I_x\ddot{\varphi} + b_1c_f\left(\dot{x} - \dot{x}_1 + b_1\dot{\varphi} - a_1\dot{\theta}\right) - b_2c_f\left(\dot{x} - \dot{x}_2 - b_2\dot{\varphi} - a_1\dot{\theta}\right) \\
&- b_2c_r\left(\dot{x} - \dot{x}_3 - b_2\dot{\varphi} + a_2\dot{\theta}\right) + b_1c_r\left(\dot{x} - \dot{x}_4 + b_1\dot{\varphi} + a_2\dot{\theta}\right) \\
&+ b_1k_f\left(x - x_1 + b_1\varphi - a_1\theta\right) - b_2k_f\left(x - x_2 - b_2\varphi - a_1\theta\right) \\
&- b_2k_r\left(x - x_3 - b_2\varphi + a_2\theta\right) + b_1k_r\left(x - x_4 + b_1\varphi + a_2\theta\right) \\
&+ k_R\left(\varphi - \frac{x_1 - x_2}{\omega}\right) = 0
\end{aligned}
\qquad (137)
$$

$$
\begin{aligned}
&I_y\ddot{\theta} - a_1c_f\left(\dot{x} - \dot{x}_1 + b_1\dot{\varphi} - a_1\dot{\theta}\right) - a_1c_f\left(\dot{x} - \dot{x}_2 - b_2\dot{\varphi} - a_1\dot{\theta}\right) \\
&+ a_2c_r\left(\dot{x} - \dot{x}_3 - b_2\dot{\varphi} + a_2\dot{\theta}\right) + a_2c_r\left(\dot{x} - \dot{x}_4 + b_1\dot{\varphi} + a_2\dot{\theta}\right) \\
&- a_1k_f\left(x - x_1 + b_1\varphi - a_1\theta\right) - a_1k_f\left(x - x_2 - b_2\varphi - a_1\theta\right) \\
&+ a_2k_r\left(x - x_3 - b_2\varphi + a_2\theta\right) + a_2k_r\left(x - x_4 + b_1\varphi + a_2\theta\right) = 0
\end{aligned}
\qquad (138)
$$

$$
\begin{aligned}
&m_f\ddot{x}_1 - c_f\left(\dot{x} - \dot{x}_1 + b_1\dot{\varphi} - a_1\dot{\theta}\right) - k_f\left(x - x_1 + b_1\varphi - a_1\theta\right) \\
&- k_R\frac{1}{\omega}\left(\varphi - \frac{x_1 - x_2}{\omega}\right) + kt_f\left(x_1 - y_1\right) = 0
\end{aligned}
\qquad (139)
$$

$$
\begin{aligned}
&m_f\ddot{x}_2 - c_f\left(\dot{x} - \dot{x}_2 - b_2\dot{\varphi} - a_1\dot{\theta}\right) - k_f\left(x - x_2 - b_2\varphi - a_1\theta\right) \\
&+ k_R\frac{1}{\omega}\left(\varphi - \frac{x_1 - x_2}{\omega}\right) + kt_f\left(x_2 - y_2\right) = 0
\end{aligned}
\qquad (140)
$$

$$
\begin{aligned}
&m_r\ddot{x}_3 - c_r\left(\dot{x} - \dot{x}_3 - b_2\dot{\varphi} + a_2\dot{\theta}\right) - k_r\left(x - x_3 - b_2\varphi + a_2\theta\right) \\
&+ kt_r\left(x_3 - y_3\right) = 0
\end{aligned}
\qquad (141)
$$

$$
\begin{aligned}
&m_r\ddot{x}_4 - c_r\left(\dot{x} - \dot{x}_4 + b_1\dot{\varphi} + a_2\dot{\theta}\right) - k_r\left(x - x_4 + b_1\varphi + a_2\theta\right) \\
&+ kt_r\left(x_4 - y_4\right) = 0
\end{aligned}
\qquad (142)
$$

La Figura 35 representa el modelo vibratorio del sistema. Se supone que el cuerpo del vehículo es una losa rígida. Esta losa tiene una masa *m*, la cual es la masa total del cuerpo, un momento de inercia longitudinal I_x, y un momento de inercia lateral I_y. Los momentos de inercia son sólo los momentos de inercia del cuerpo y no los momentos de inercia de todo el vehículo. Las ruedas tienen una masa *m*1, *m*2, *m*3 y *m*4 respectivamente. Sin embargo, es común hacer

$$
m_1 = m_2 = m_f \qquad (143)
$$

$$m_3 = m_4 = m_r \tag{144}$$

La rigidez de los neumáticos delanteros y traseros se indica por kt_f y kt_r respectivamente. Podemos despreciar el amortiguamiento de los neumáticos, debido a que es mucho más pequeño que el amortiguamiento de los amortiguadores.

La suspensión del vehículo tiene rigidez k_f y amortiguamiento c_f en la parte delantera y rigidez k_r y amortiguamiento c_r en la parte trasera. Es común hacer las suspensiones de la izquierda y de la derecha iguales. Por eso, sus rigideces y amortiguamientos son iguales. El vehículo puede tener también una barra antivuelco delante y detrás, con una rigidez torsional k_R. Utilizando un modelo simple, la barra antivuelco proporciona un par de torsión M_R proporcional al ángulo de balanceo.

$$M_R = -\left(k_{R_f} + k_{R_r}\right)\varphi = -k_R\varphi \tag{145}$$

Sin embargo, un mejor modelo de la reacción de la barra antivuelco es

$$M_R = -k_{R_f}\left(\varphi - \frac{x_1 - x_2}{\omega_f}\right) - k_{R_f}\left(\varphi - \frac{x_4 - x_3}{\omega_r}\right) \tag{146}$$

La mayoría de los coches solo tiene barra antivuelco en la parte delantera. Para estos automóviles, el momento de la barra antivuelco se simplifica a

$$M_R = -k_R\left(\varphi - \frac{x_1 - x_2}{\omega}\right) \tag{147}$$

donde,

$$\omega_f \equiv \omega = b_1 + b_2 \tag{148}$$

$$k_{R_f} \equiv k_R \tag{149}$$

El conjunto de ecuaciones de movimiento puede ser reordenado en forma matricial

$$[\mathbf{M}]\{\ddot{\mathbf{x}}\} + [\mathbf{C}]\{\dot{\mathbf{x}}\} + [\mathbf{K}]\{\mathbf{x}\} = \{\mathbf{F}\} \tag{150}$$

donde,

$$\{x\} = \begin{Bmatrix} x & \varphi & \theta & x_1 & x_2 & x_3 & x_4 \end{Bmatrix}^T \tag{151}$$

$$[M] = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & I_x & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & I_y & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_f & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & m_f & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & m_r & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & m_r \end{bmatrix} \tag{152}$$

$$[C]=\begin{bmatrix} c_{11} & c_{12} & c_{13} & -c_f & -c_f & -c_r & -c_r \\ c_{21} & c_{22} & c_{23} & -b_1c_f & b_2c_f & b_2c_r & -b_1c_r \\ c_{31} & c_{32} & c_{33} & a_1c_f & a_1c_f & -a_2c_r & -a_2c_r \\ -c_f & -b_1c_f & a_1c_f & c_f & 0 & 0 & 0 \\ -c_f & b_2c_f & a_1c_f & 0 & c_f & 0 & 0 \\ -c_r & b_2c_r & -a_2c_r & 0 & 0 & c_r & 0 \\ -c_r & -b_1c_r & -a_2c_r & 0 & 0 & 0 & c_r \end{bmatrix} \qquad (153)$$

$$\begin{aligned} c_{11} &= 2c_f + 2c_r \\ c_{21} &= c_{12} = b_1c_f - b_2c_f + b_1c_r - b_2c_r \\ c_{31} &= c_{13} = 2a_2c_r - 2a_1c_f \\ c_{22} &= b_1^2c_f + b_2^2c_f + b_1^2c_r + b_2^2c_r \\ c_{32} &= c_{23} = a_1b_2c_f - a_1b_1c_f + a_2b_1c_r - a_2b_2c_r \\ c_{33} &= 2c_fa_1^2 + 2c_ra_2^2 \end{aligned} \qquad (154)$$

$$[K]=\begin{bmatrix} k_{11} & k_{12} & k_{13} & -k_f & -k_f & -k_r & -k_r \\ k_{21} & k_{22} & k_{23} & k_{24} & k_{25} & b_1k_r & -b_2k_r \\ k_{31} & k_{32} & k_{33} & a_1k_f & a_1k_f & -a_2k_r & -a_2k_r \\ -k_f & k_{42} & a_1k_f & k_{44} & -\frac{k_R}{\omega^2} & 0 & 0 \\ -k_f & k_{52} & a_1k_f & -\frac{k_R}{\omega^2} & k_{55} & 0 & 0 \\ -k_r & b_2k_r & -a_2k_r & 0 & 0 & k_r + kt_r & 0 \\ -k_r & -b_2k_r & -a_2k_r & 0 & 0 & 0 & k_r + kt_r \end{bmatrix} \qquad (155)$$

$$
\begin{aligned}
k_{11} &= 2k_f + 2k_r \\
k_{21} &= k_{12} = b_1 k_f - b_2 k_f + b_1 k_r - b_2 k_r \\
k_{31} &= k_{13} = 2a_2 k_r - 2a_1 k_f \\
k_{22} &= k_R + b_1^2 k_f + b_2^2 k_f + b_1^2 k_r + b_2^2 k_r \\
k_{32} &= k_{23} = a_1 b_2 k_f - a_1 b_1 k_f + a_2 b_1 k_r - a_2 b_2 k_r \\
k_{42} &= k_{24} = -b_1 k_f - \frac{1}{\omega} k_R \\
k_{52} &= k_{25} = b_2 k_f + \frac{1}{\omega} k_R \\
k_{33} &= 2k_f a_1^2 + 2k_r a_2^2 \\
k_{44} &= k_{55} = k_f + kt_f + \frac{1}{\omega^2} k_R
\end{aligned}
\qquad (156)
$$

$$
\{F\} = \begin{Bmatrix} 0 & 0 & 0 & y_1 kt_f & y_2 kt_f & y_3 kt_r & y_4 kt_r \end{Bmatrix}^T \qquad (157)
$$

En la tabla 5 se muestra la nomenclatura empleada para definir todos elementos empleados en el modelo de siete grados de libertad y con suspensión independiente y que se muestran en la Figura 35.

Tabla 5 – Parámetros y variables del modelo de vehículo completo.

Parámetro	*Significado*
m	masa total de la carrocería (masa suspendida)
m_1	masa de la rueda delantera izquierda
m_2	masa de la rueda delantera derecha
m_3	masa de la rueda trasera derecha
m_4	masa de la rueda trasera izquierda
Ix	momento de inercia de la carrocería alrededor de su eje longitudinal
Iy	momento de inercia de la carrocería alrededor de su eje transversal
cf	Coeficiente de amortiguamiento de la suspensión delantera
cr	Coeficiente de amortiguamiento de la suspensión trasera
kf	Coeficiente de rigidez de la suspensión delantera
kr	Coeficiente de rigidez de la suspensión trasera
kt_f	Coeficiente de rigidez de los neumáticos delanteros
kt_r	Coeficiente de rigidez de los neumáticos traseros
kR_f	Coeficiente de rigidez de la barra antivuelco delantera
kR_r	Coeficiente de rigidez de la barra antivuelco trasera
y_1	Amplitud de la excitación de la rueda delantera izquierda
y_2	Amplitud de la excitación de la rueda delantera derecha
y_3	Amplitud de la excitación de la rueda trasera derecha
y_4	Amplitud de la excitación de la rueda trasera izquierda
a_1	Distancia longitudinal del centro de gravedad a la suspensión delantera
a_2	Distancia longitudinal del centro de gravedad a la suspensión trasera

b_1	Distancia transversal del centro de gravedad a la suspensión del lado izquierdo
b_2	Distancia transversal del centro de gravedad a la suspensión del lado derecho
φ	Ángulo de balanceo de la carrocería
θ	Ángulo de cabeceo de la carrocería

7. RESULTADOS Y CONCLUSIONES.

7.1. Resultados.

La falta de antecedentes en este tipo de análisis hace más interesante este trabajo ya que será de gran ayuda a la hora de estudiar el comportamiento del vehículo frente a las vibraciones producidas por las irregularidades de la carretera y poder mejorar el confort, la estabilidad, y en definitiva todas las características que influyen en el comportamiento del vehículo.

Además, los diferentes artículos encontrados referentes al modelado de la suspensión de vehículos se limitan a modelar sólo medio vehículo, por lo que el alcance de este trabajo por lo que al número de grados de libertad se refiere, va más allá de modelos de cuatro grados de libertad, puesto que en él se tienen en cuenta modelos de vehículo completo o de siete grados de libertad. Además, estos artículos sólo plantean la posibilidad de estudiar el comportamiento de suspensiones independientes, por lo que en este aspecto este trabajo también es más completo ya que también incorpora el caso de modelos con suspensión de eje rígido.

Para verificar los resultdos, los valores de los diferentes parámetros de la suspensión, se han obtenido del libro "Vehicle dynamics. Theory and application" de Reza N. Jazar.

La falta de antecedentes en este tipo de análisis ha hecho difícil su verificación. Para comprobar que los cálculos están bien hechos y son correctos se ha recurrido a programar cada modelo, según los métodos explicados en el apartado 4, y a comparar los diferentes resultados obtenidos en cada método, pudiendo comprobar que para los cuatro métodos se obtiene los mismos resultados. Además, se ha utilizado un cuarto método, basado en el cálculo simbólico de la función de transferencia según el método de la transformada de Laplace. Por tanto, tenemos que para verificar el programa se han utilizado dos métodos basados en la transformada de Laplace, uno basado en el método de superposición modal y el otro basado en el método del espacio de estado.

Los dos métodos de la transformada de Laplace son prácticamente iguales, sólo difieren en el método de representación. En ambos métodos se calcula de forma simbólica la función de transferencia, pero en uno de ellos se utiliza la función "bode" de MATLAB para representar los gráficos. Esta función genera dos gráficos automáticamente, uno para la magnitud y otro para la fase. El problema de este método es que hay que obtener la expresión simbólica previamente e introducirla en la función *bode*, lo que resulta difícil de manipular para los modelos de cuatro grados de libertad dando lugar a errores, provocando un funcionamiento lento en MATLAB llegando incluso a detenerse durante minutos. En el caso del modelo de siete grados de libertad es aún peor, ya que MATLAB no puede representar de forma simbólica la función de transferencia de este modelo debido a que solo puede representar 35.000 caracteres. El otro método basado en la transformada de Laplace es más ágil, aunque hay que calcular la magnitud y la fase de la función de transferencia aparte (en el otro método lo hacia la función *bode*) y luego utilizar las funciones *loglog* y *semilogx* para representar la magnitud y la fase respectivamente. El problema de este método es que implica que Matlab realice cálculos simbólicos, y esta es una de las limitaciones de los ejecutables de MATLAB, que no pueden realizar cálculos simbólicos. Quedan pues descartados como métodos de programación los dos basados en la transformada de Laplace. Nos quedan pues los métodos de superposición modal y de espacio de estado. En el análisis modal hay muchos datos basura. Al ampliar las matrices, las funciones de respuesta en frecuencia están repetidas. Al final para llevar a cabo este trabajo se ha elegido el método del espacio de estado, por su sencillez y su menor coste computacional, dando lugar a resultados más rápidos y un manejo de los datos más sencillo y ágil.

7.2. Conclusiones.

Este trabajo tenía como objetivo desarrollar una aplicación informática que modelara la suspensión de un vehículo y mostrara por pantalla los resultados de realizar un análisis en frecuencia. Dicha aplicación debía servir como herramienta a los alumnos de la asignatura de automóviles, para ayudarles en la comprensión del estudio de las vibraciones en los vehículos.

Para alcanzar el objetivo del presente trabajo, ha sido necesario aprender y familiarizarse con una herramienta de software matemático y su lenguaje de programación como MATLAB. Además, también ha sido necesario familiarizarse con la herramienta GUIDE que incorpora el software de MATLAB, una herramienta que permite la creación y edición de interfaces de usuario (GUI). Dicho software ha permitido resolver las ecuaciones matemáticas que describen el comportamiento de cada modelo, así como desarrollar la aplicación informática.

Podemos decir, que se ha logrado el objetivo de este trabajo, pues se ha logrado crear una aplicación informática que permite el análisis de varios modelos con diferentes grados de libertad y tipo de suspensión de un vehículo. Dicha aplicación está planteada de un modo muy sencillo para ayudar al usuario con su uso. La aplicación consta de una ventana de inicio donde se da la posibilidad de elegir los grados de libertad a analizar. Esta ventana da paso a otra ventana que consta, de una imagen del modelo para ayudar a su comprensión, un panel de entrada de datos donde se especifican todas las variables que hacen falta para el análisis, otro panel que muestra las frecuencias naturales del sistema y un panel para la obtención de los gráficos de la función de transferencia. El panel de gráficos abre otra ventana donde se visualizan sólo las gráficas de la magnitud y la fase de la función de transferencia.

Podemos concluir que este trabajo será capaz de cumplir con su objetivo, y proporcionar a los alumnos de la asignatura de automóviles, una herramienta que sirva como primera aproximación al estudio del efecto que producen las vibraciones en el vehículo, ya sea sobre la carrocería, sobre las masas no suspendidas o los ocupantes del vehículo.

7.3. Trabajos futuros.

En el presente trabajo, se ha modelado sólo la tipología de suspensión pasiva de un automóvil, que permite un estudio sencillo de las vibraciones en un automóvil. Esto está bien como introducción al estudio de las vibraciones en automóviles y así sentar una base para poder entender conceptos más complejos como son las suspensiones activas y semiactivas. Actualmente, casi cualquier modelo de automóvil lleva incorporados sistemas de control que intervienen sobre el sistema de suspensión y tracción, dando lugar a ello a suspensiones activas y semiactivas. Por tanto, un posible proyecto futuro seria modelizar esta clase de suspensiones y realizar una comparación entre las respuestas que producen cada una. Para ello sería necesario familiarizarse con la herramienta Simulink de MATLAB, una herramienta que permite la simulación multidominio y el diseño basado en modelos un entorno de diagramas de bloque.

Por otro lado, la aplicación desarrollada sólo muestra el comportamiento del sistema para una rigidez y amortiguamiento dados introducidos por el usuario, sin saber si estos valores son adecuados para el sistema. Sería interesante desarrollar una aplicación para la optimización de la suspensión, de modo que proporcionara los valores de rigidez y amortiguamiento óptimos.

Además, se podría extender el análisis al dominio del tiempo y obtener el desplazamiento de cada grado de libertad. Al realizar un análisis en el dominio del tiempo, se puede estudiar la respuesta del sistema frente a diferentes funciones excitadoras. En este trabajo nos hemos limitado al estudio de los modelos excitados por la base, dejando los otros tipos de excitación harmónica para desarrollos futuros, excepto el caso de vibración forzada que no tiene ningún interés en el estudio de dinámica de vehículos. En este trabajo se ha supuesto que el sistema está excitado por las irregularidades de la carretera, lo que supone un análisis de un modelo donde la función forzada actúa sobre la base. Como propuesta para desarrollos futuros se propone el estudio de los dos casos adicionales más que consisten, uno en un modelo excitado por una fuerza excéntrica sobre la masa suspendida, y el otro en un modelo excitado en la base por una excitación excéntrica. El modelo de excitación excéntrica corresponde a un modelo para cualquier tipo de motor rotatorio en suspensión, tal como un motor de combustión interna de un automóvil sobre sus soportes. El modelo de excitación excéntrica en la base es un modelo para cualquier equipo montado sobre un motor.

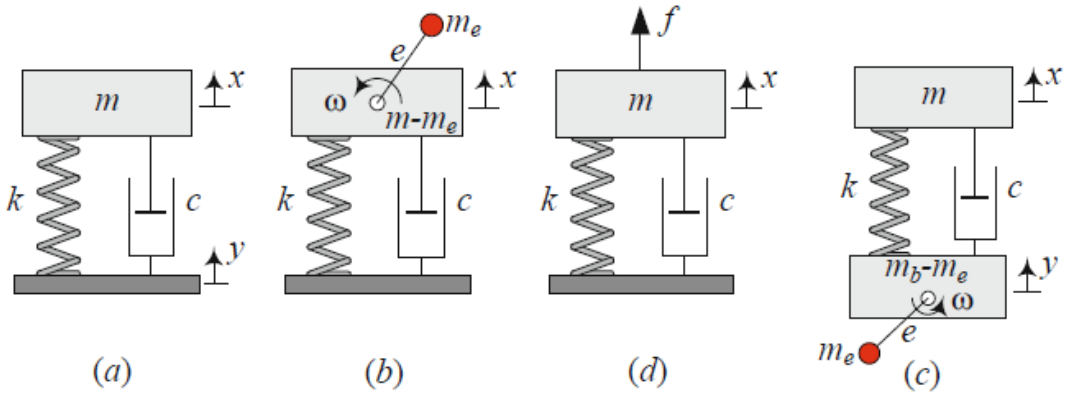


Figura 36. Los cuatro tipos de un sistema de un grado de libertad harmónicamente excitado: a. excitación de la base, b. excitación excéntrica, c. excitación excéntrica de la base, d. excitación forzada. (Reza N. Jazar, 2014)

Por último, en los diferentes modelos utilizados, no se ha tenido en cuenta el efecto amortiguante de los neumáticos, como ya se dijo, éste es despreciable si lo comparamos con el amortiguamiento de la suspensión, por los que en futuros análisis se podría añadir un pequeño amortiguador a los

diferentes modelos de más de un grado de libertad, que represente el efecto amortiguante introducido por los neumáticos.

8. PRESUPUESTO.

1. Autor:
Bruno Cebolla Bono
2. Departamento:
Ingeniería Mecánica y de Materiales
3. Descripción del Proyecto:
- Título: Modelado de la suspensión de un vehículo.
- Duración: 6 meses
- Costes indirectos: 20%
4. Presupuesto total del Proyecto:
30.942,00 €

5. Desglose presupuestario

PERSONAL

Apellidos y nombre	Tarea	Categoría	Dedicación (h)	Coste (€/h)	Coste (€)
Cebolla Bono Bruno	Modelado y simulación	Ingeniero Técnico Industrial	660	30	19.800,00
	Redacción		132	30	3.960,00
				Total	**23.760,00**

EQUIPOS

Descripción	Coste (€)	% Uso dedicado al proyecto	Dedicación (meses)	Periodo depreciación	Coste imputable
Ordenador personal DELL Inspiron 5559	850,00	100	6	12	425,00
Licencia Sofware	6.000,00	80	4	12	1.600,00
				Total	**2.025,00**

6. Resumen de costes (€)	
Personal	23.760,00
Amortización	2.025,00
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes Indirectos	5.157,00
Total	**30.942,00**

El presupuesto total de este proyecto asciende a la cantidad de treinta mil novecientos cuarenta y dos euros.

València, a 1 de Septiembre de 2017

El ingeniero proyectista Fdo. Bruno Cebolla Bono.

9. BIBLIOGRAFIA.

Arias-Paz, Manuel (2006): *"Manual de automóviles"*. Madrid, Dossat 2000.

Ríos, Orlando (1998): *"La suspensión. Automóviles de competición"*. Barcelona, CEAC.

Luque, P.; Álvarez, D.; Vera, C. (2004): *"Ingeniería del Automóvil. Sistemas y Comportamiento Dinámico"*. Madrid, Paraninfo.

Alonso, J. M. (2008): *"Técnicas del Automóvil. Chasis"*. Madrid, Paraninfo.

Daniel J. Inman (2014): *"Engineering vibration. Edinburg Gate"*. Pearson Education.

Singiresu, S. Rao (2012): *"Vibraciones Mecánicas. México"*. Pearson Educación de México.

Michael R Hatch (2001): *"Vibration Simulation Using MATLAB and ANSYS".* Chapman & Hall/CRC

Pérez Bello, M. Ángel (2004): *"Tecnología de la suspensión, dirección y ruedas. Circuitos hidráulicos y neumáticos"*. Madrid, Dossat 2000.

Font Mezquita, José; Dols Ruiz, Juan Fco. (2006): *"Tratado sobre automóviles. Tomo IV. La Dinámica del automóvil"*. València, SPUPV-2006.875. ETSII.

Font Mezquita, José; Dols Ruiz, Juan Fco. (2004): *"Tratado sobre automóviles. Tomo II. Tecnología del automóvil"*. València, SPUPV-2006.875. ETSII.

Orovio Astudillo, Manuel (2010): *"Tecnología del automóvil"*. Madrid, Ediciones Paraninfo, SA.

Domínguez, Esteban José; Ferrer, Julián (2014): *"Mecánica del vehículo"*. Madrid, Editorial Editex, SA.

Pérez Bello, M. Ángel (2011): *"Circuitos de fluidos. Suspensión y dirección"*. Madrid, Ediciones Paraninfo, SA.

Daniels, Jeff (2005): *"Tecnología del coche moderno"*. Barcelona, Ediciones CEAC.

Nuno M. M. Maia; Júlio M. M. Silva (1997): *"Theoretical and experimental modal analysis"*. Baldock, Hertfordshire, England, Research Studies Press LTD.

Roy R. Craig, Jr. (1981): *"Structural Dynamics: An introduction to computer methods"*. John Wiley & Sons.

Pérez López, César (2002): *"MATLAB y sus aplicaciones en las ciencias y la ingeniería"*. Madrid, Pearson Educación S.A.

Moore, Holly (2012): *"MATLAB for engineers"*. Pearson Educación S.A.

Gómez Pérez, Ana (2011): "Análisis a fatiga del chasis de la motocicleta MotoStudent debido a las fuerzas del motor".

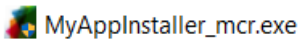
www.aficionadosalamecanica.net

ANEXOS

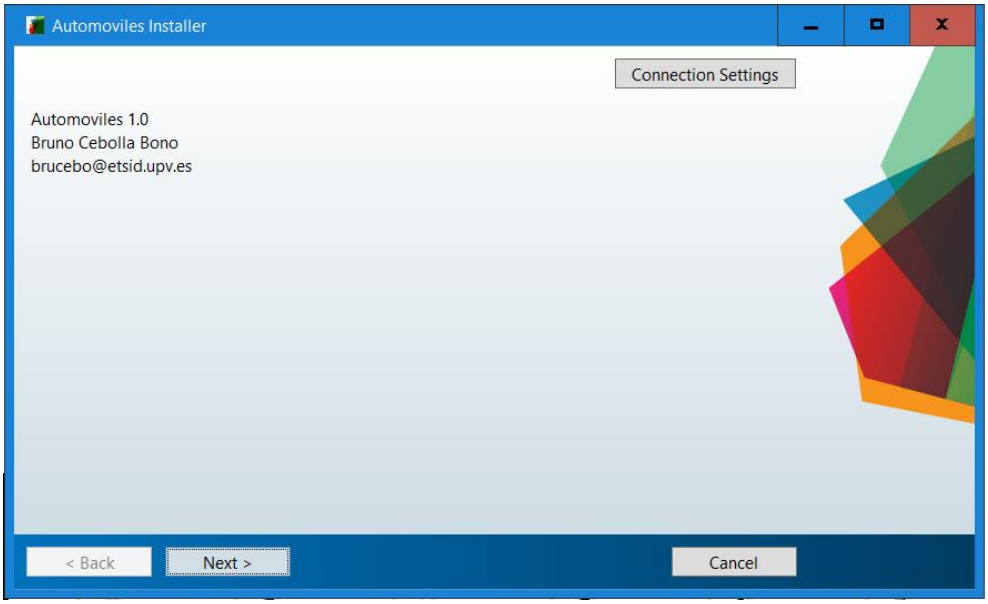
ANEXO A. MANUAL DE USUARIO

A.1. Manual de instalación.

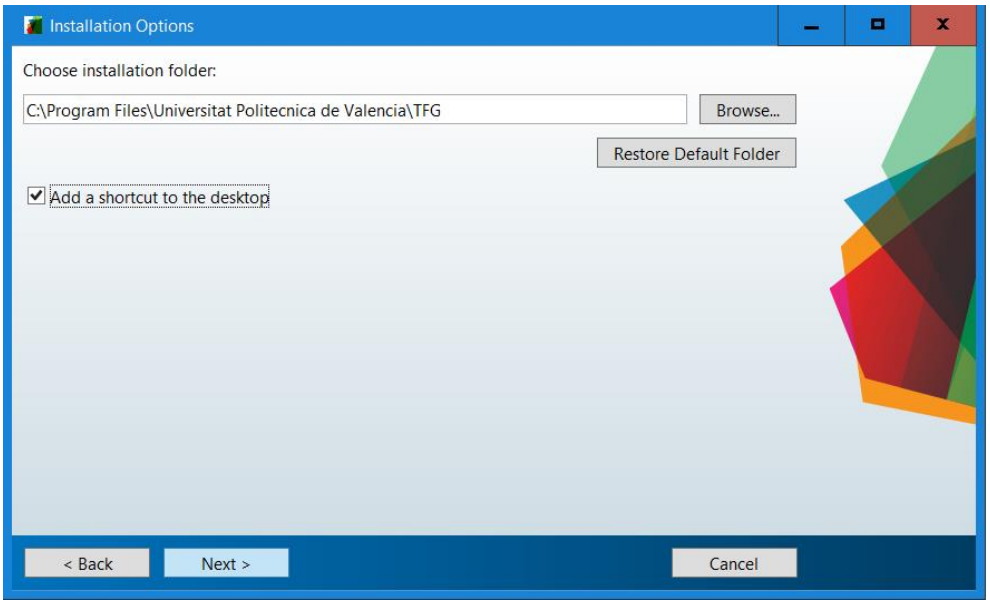
1. Abrir la carpeta *Automóviles*.
2. abrir la carpeta *for_redistribution*.
3. Hacer click con el botón derecho del ratón sobre el archivo .exe que aparece y seleccionar ejecutar como administrador.



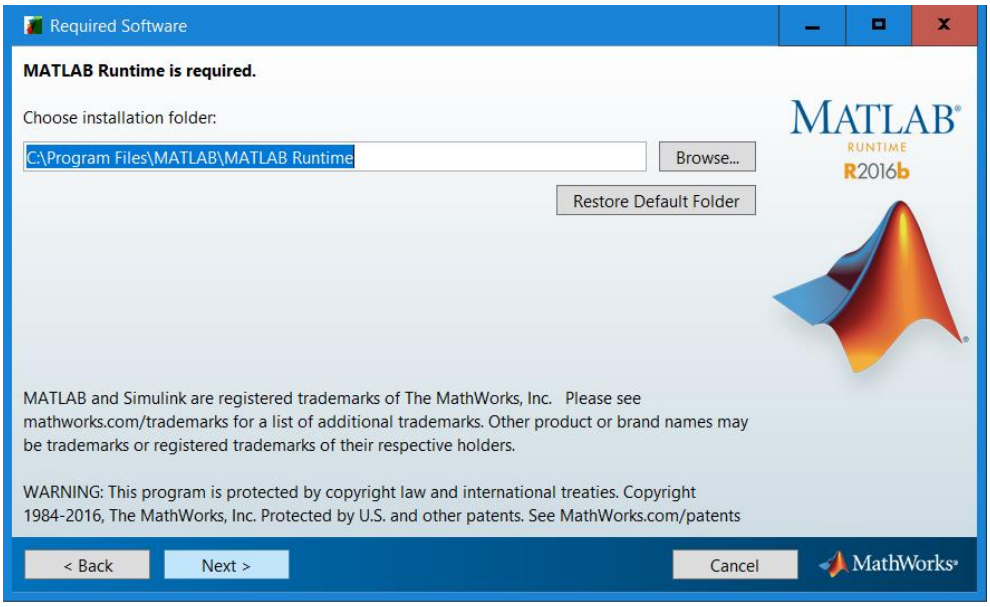
4. Aparecerá la siguiente ventada, donde pulsaremos *Next*.



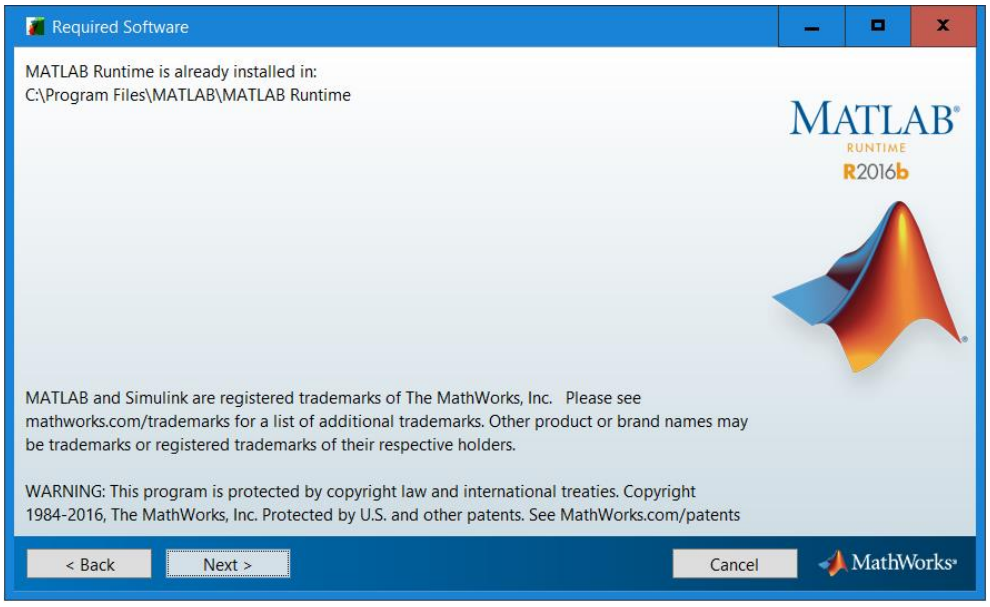
5. En la siguiente ventana seleccionaremos la carpeta donde se instalará el programa. A continuación seleccionaremos la casilla "*Add a shortcut to the desktop*" para crear un acceso directo en el escritorio. Antes de pulsar en *Next* modificaremos el nombre del directorio de destino quitando las tíldes.



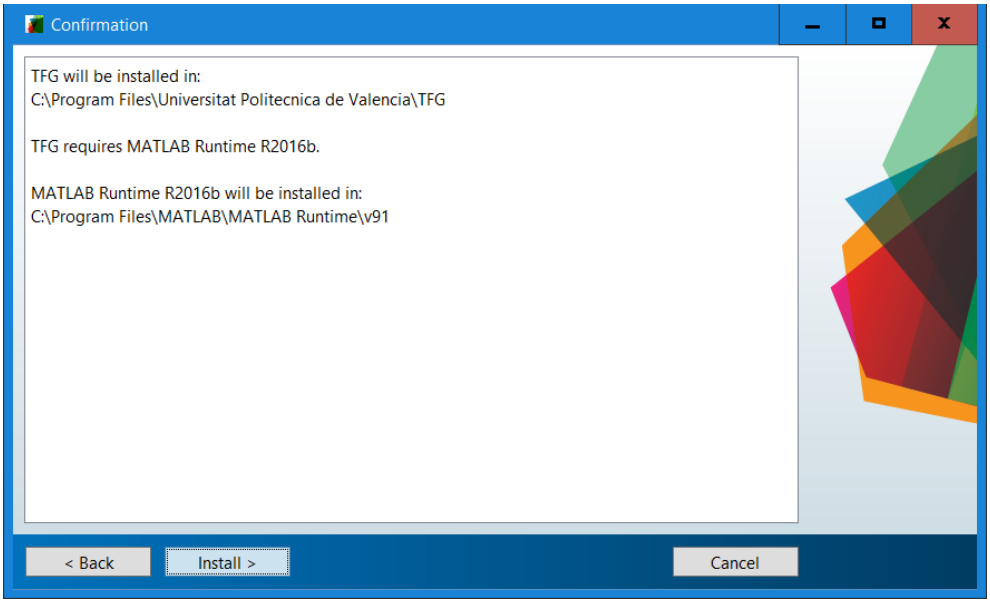
6. La siguiente ventana hace referencia a MATLAB Runtime. MATLAB Runtime es un complemento necesario para que las aplicaciones creadas con MATLAB funciones en cualquier ordenador que no tenga instalado MATLAB. Si no tenemos instalado MATLAB, aparecerá la siguiente ventana, en este caso podemos instalar MATLAB Runtime de forma gratuita en el siguiente enlace (https://es.mathworks.com/products/compiler/mcr.html)



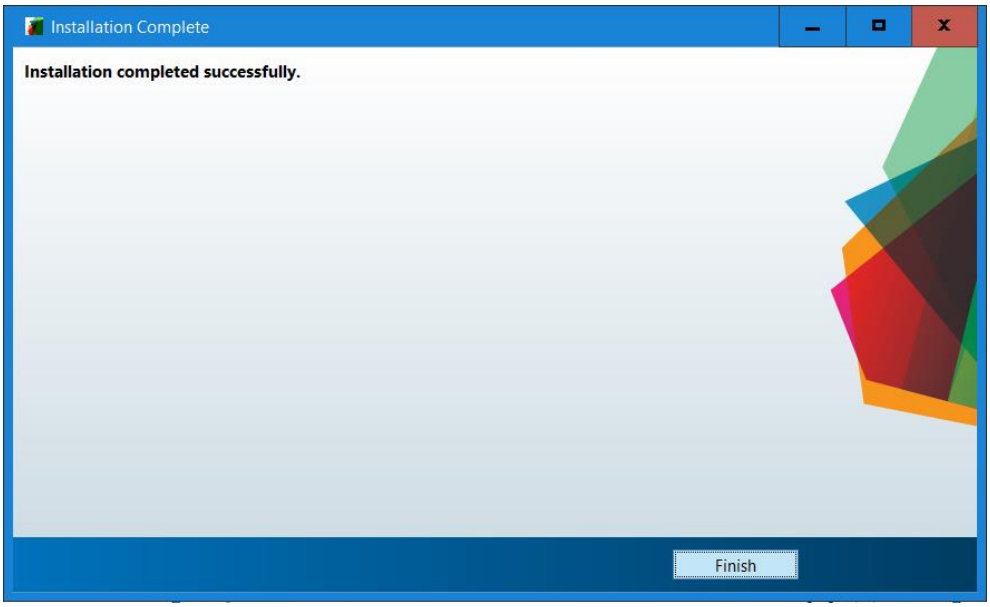
Si ya tenemos instalado MATLAB Runtime, aparecerá la siguiente ventana donde sólo hay que pulsar sobre *Next*.



7. Por último, hacemos click sobre el botón "*Install*", y esperamos a que el programa se instale.

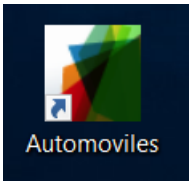


8. Una vez el proceso ha terminado, hacemos click sobre *Finish*.


Installation Complete
Installation completed successfully.
Finish

A.2. Manual de funcionamiento.

Una vez instalado el programa, vamos al escritorio y hacemos doble click sobre el ícono



y esperamos mientras se inicia el programa.

Una vez se ha iniciado el programa aparecerá la siguiente ventana

Figura 37. Ventana de inicio del programa.

En esta ventana aparecen siete botones situados en el borde izquierdo. Cada botón abre una ventana correspondiente a un modelo concreto de la suspensión del vehiculo. En la Tabla 6 se define cada botón.

Las ventanas que se habren al hacer click sobre los botones son similares, las pequeñas diferencias que existent entre cada una de ellas se deben a cuestiones de espacio. La organización de dichas ventanes es muy sencilla. Cada ventada consta de una imagen que representa el modelo a analizar, un panel de datos de entrada donde se especifican todos los parámetros necesarios que el usuario deberá introducir para realizar el análisis, un panel donde se calculán las frecuencias naturales de cada sistema, y un panel de gráficos. El funcionamiento de cada ventana es el mismo, una vez introducidos los datos se pulsa sobre el botón calcular para obtener las frecuencias naturales, se elige la función de transferencia deseada del menú desplegable, y se hace click sobre el botón REPRESENTAR. Debido a que todas las ventanas funcionan del mismo modo sólo se explicará el funcionamiento de la ventana del modelo de un grado de libertad para no hacer este manual redundante. Del resto de ventanas se explicaran los parámetros necesarios para realizar el análisis.

Tabla 6 – Definición de los botones de la pantalla de inicio.

Botón	***Significado***
1 GDL	Abre una ventana para el anàlisis del modelo de un grado de libertad o modelo de 1/8 de vehículo de la suspensión de un vehiculo.
2 GDL	Abre una ventana para el anàlisis del modelo de dos grados de libertad, o modelo de ¼ de vehículo de la suspensión de un vehiculo.
3 GDL	Abre una ventana para el anàlisis del modelo de tres grados de libertad de la suspensión de un vehiculo.
4 GDL susp. ind. pitch	Abre una ventana para el anàlisis del modelo de cuatro grados de libertad, o modelo de medio vehiculo con suspensión independiente y movimiento de cabeceo de la suspensión de un vehiculo.
4 GDL susp. ind. roll	Abre una ventana para el anàlisis del modelo de cuatro grados de libertad, o modelo de medio vehiculo con suspensión independiente y movimiento de balanceo de la suspensión de un vehiculo.
4 GDL susp. Eje rígido	Abre una ventana para el anàlisis del modelo de cuatro grados de libertad, o modelo de medio vehiculo con suspensión de eje rígido y movimiento de balanceo de la suspensión de un vehiculo.
7 GDL	Abre una ventana para el anàlisis del modelo de siete grados de libertad, o modelo de vehiculo completo, con suspensión independiente delante y detrás de la suspensión de un vehiculo.

Ventana del modelo de un grado de libertad.

Esta ventana se abre como resultado de haber hecho click sobre el botón *1 GDL*, y su apariencia es la siguiente:

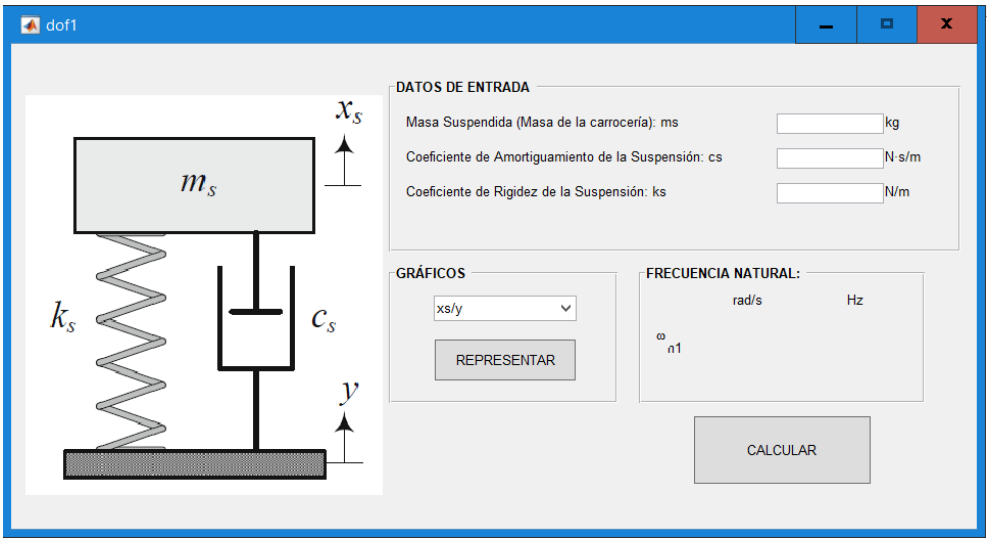


Figura 38. Ventana del modelo de un grado de libertad (1 GDL)

En la venta se pueden observar los elementos descritos anteriormente, a saber, una imagen representativa del modelo a analizar, un panel de datos de entrada, un panel de gráficos y un panel de frecuencias naturales.

PANEL DE DATOS DE ENTRADA.

En el panel de datos de entrada se especifican los parámetros a introducir para realizar el análisis. Estos son:

Parámetro	*Definición*
m_s	Masa suspendida del vehículo, que representa ¼ de la masa total de la carrocería. La cuál deberá expresarse en quilogramos (kg)
c_s	Coeficiente de amortiguamiento de la suspensión, cuyas unidades deberán ser N·s/m
k_s	Coeficiente de rigidez de la suspensión del vehículo, cuyas unidades deberán ser N/m

Los datos sólo podrán ser números, es decir, no pueden contener letras ni otros caracteres. Si alguno de los datos de entrada contiene decimales, éstos deberán expresarse mediante un punto ".".

PANEL DE FRECUENCIAS NATURALES.

En el panel de frecuencias naturales, se muestran las frecuencias naturales del sistema, tanto en radianes por segundo (rad/s) en la primera columna, como en Hercios (Hz) en la segunda columna. Para ver los resultados en este panel, hay que hacer click sobre el botón CALCULAR.

PANEL DE GRÁFICOS.

En el panel de gráficos, se pueden distinguir dos elementos. Por un lado tenemos un menú desplegable que nos da la opción de elegir que función de transferencia que queremos visualizar, y por otro lado, nos encontramos un botón llamado REPRESENTAR que al hacer click sobre él, nos abrirá otra ventana donde se mostraran dos gráficos, en el gráfico superior se representará la magnitud de la función de transferencia en decibelios (db), y bajo éste, se representará la fase de la misma función de transferecnia en grados (deg). Para obtener una nueva gráfica, se ciera la ventana de gráficos, se selecciona otra función y se vuelve a pulsar el botón REPRESENTAR.

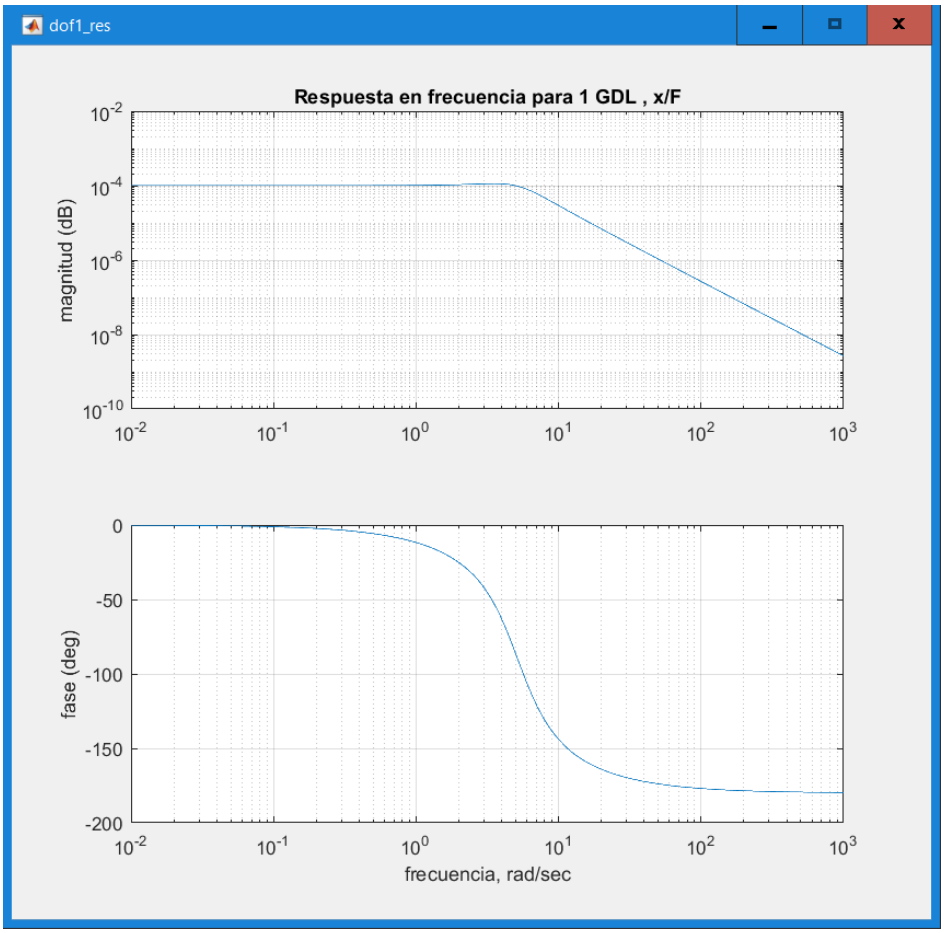
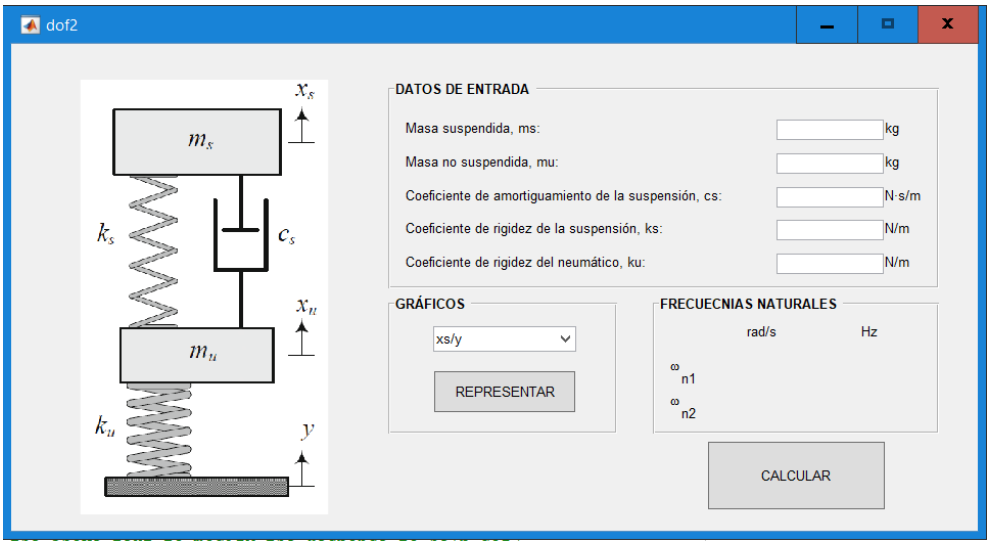


Figura 39. Ventana de gráficos.

Ventana del modelo de dos grados de libertad.

La apariencia de la ventana del modelo de dos grados de libertad se muestra en la siguiente figura.

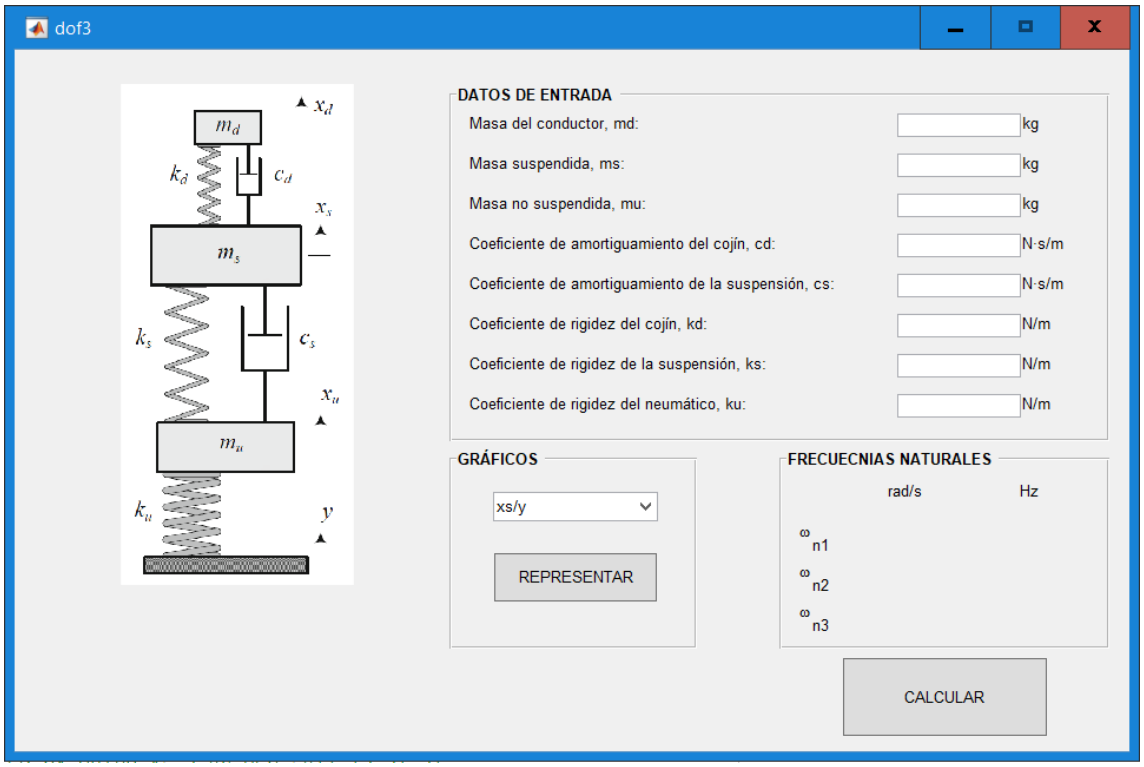


Como se puede observar, la estructura de esta ventana es la misma que la anterior.

Los parámetros necesarios para realizar el análisis son:

Parámetro	*Definición*
m_s	Masa suspendida del vehículo, que representa ¼ de la masa total de la carrocería. La cuál deberá expresarse en quilogramos (kg)
m_u	Masa no suspendida del vehículo, representa una rueda del vehículo. La cuál deberá expresarse en quilogramos (kg)
c_s	Coeficiente de amortiguamiento de la suspensión, cuyas unidades deberán ser N·s/m
k_s	Coeficiente de rigidez de la suspensión del vehículo, cuyas unidades deberán ser N/m
k_u	Coeficiente de rigidez del neumático, cuyas unidades deberán ser N/m

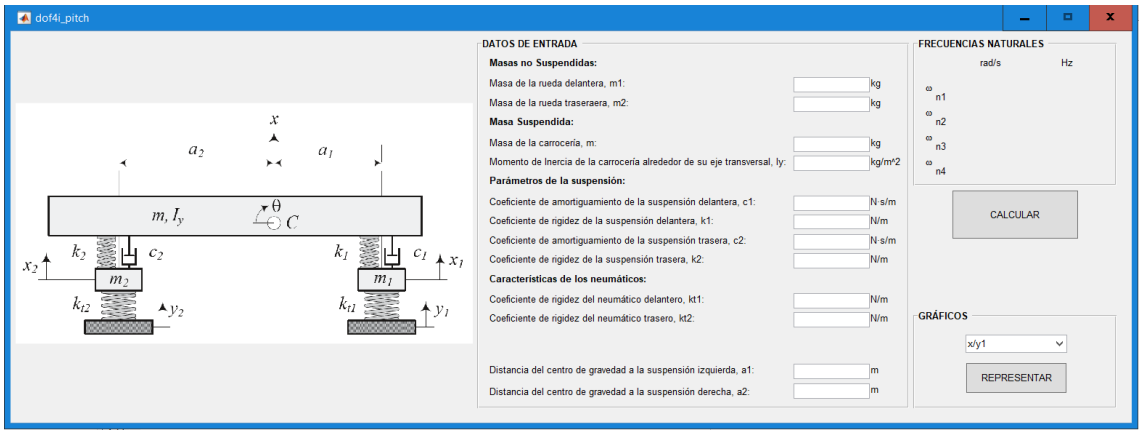
Ventana del modelo de tres grados de libertad.



Los parámetros necesarios para realizar el análisis son:

Parámetro	*Definición*
m_s	Masa suspendida del vehículo, que representa ¼ de la masa total de la carrocería. La cuál deberá expresarse en quilogramos (kg)
m_u	Masa no suspendida del vehículo, representa una rueda del vehículo. La cuál deberá expresarse en quilogramos (kg)
m_d	Masa del conductor. La cuál deberá expresarse en quilogramos (kg)
c_s	Coeficiente de amortiguamiento de la suspensión, cuyas unidades deberán ser N·s/m
c_d	Coeficiente de amortiguamiento del asiento del conductor, cuyas unidades deberán ser N·s/m
k_s	Coeficiente de rigidez de la suspensión del vehículo, cuyas unidades deberán ser N/m
k_u	Coeficiente de rigidez del neumático, cuyas unidades deberán ser N/m
k_d	Coeficiente de rigidez del asiento del conductor, cuyas unidades deberán ser N/m

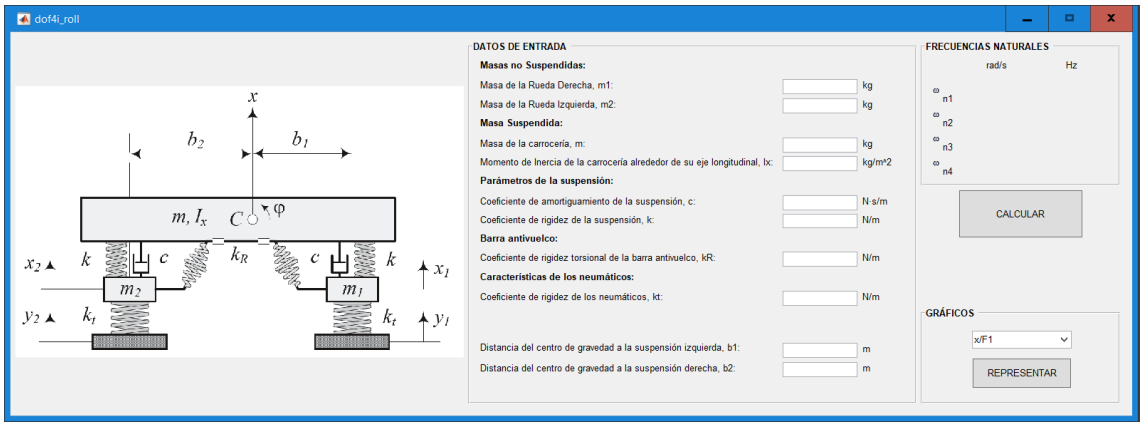
Ventana del modelo de cuatro grados de libertad con suspensión independiente y movimiento de cabeceo.



Los parámetros necesarios para realizar el análisis son:

Parámetro	Significado
m	Masa suspendida, representa 1/2 de la masa total de la carrocería. La cuál deberá expresarse en quilogramos (kg)
m_1	Masa no suspendida delantera, masa de la rueda delantera. La cuál deberá expresarse en quilogramos (kg)
m_2	Masa no suspendida trasera, masa de la rueda trasera. La cuál deberá expresarse en quilogramos (kg)
I_y	1/2 del momento de inercia de la carrocería alrededor de su eje transversal, expresado en kg/m^2
c_1	Coeficiente de amortiguamiento de la suspensión delantera, cuyas unidades deberán ser N·s/m
c_2	Coeficiente de amortiguamiento de la suspensión trasera, cuyas unidades deberán ser N·s/m
k_1	Coeficiente de rigidez de la suspensión delantera, cuyas unidades deberán ser N/m
k_2	Coeficiente de rigidez de la suspensión trasera, cuyas unidades deberán ser N/m
k_{t1}	Coeficiente de rigidez del neumático delantero, cuyas unidades deberán ser N/m
k_{t2}	Coeficiente de rigidez del neumático trasero, cuyas unidades deberán ser N/m
a_1	Distancia del centro de gravedad al eje delantero expresada en metros
a_2	Distancia del centro de gravedad al eje trasero expresada en metros

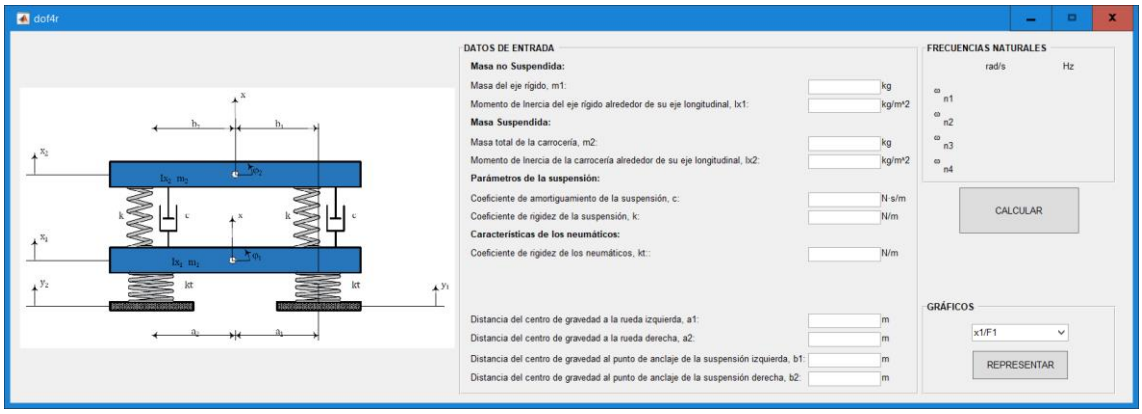
Ventana del modelo de cuatro grados de libertad con suspensión independiente y movimiento de balanceo.



Los parámetros necesarios para realizar el análisis son:

Parámetro	Significado
m	Masa suspendida, representa 1/2 de la masa total de la carrocería. La cuál deberá expresarse en quilogramos (kg)
m_1	Masa no suspendida izquierda, masa de la rueda izquierda. La cuál deberá expresarse en quilogramos (kg)
m_2	Masa no suspendida derecha, masa de la rueda derecha. La cuál deberá expresarse en quilogramos (kg)
I_x	1/2 del momento de inercia de la carrocería alrededor de su eje longitudinal, expresado en kg/m^2
c	Coeficiente de amortiguamiento de la suspensión, cuyas unidades deberán ser N·s/m
k	Coeficiente de rigidez de la suspensión, cuyas unidades deberán ser N/m
k_t	Coeficiente de rigidez de los neumáticos, cuyas unidades deberán ser N/m
k_R	Coeficiente de rigidez torsional de la barra antivuelco, cuyas unidades deberán ser N·m/rad
b_1	Distancia del centro de gravedad a la suspensión izquierda expresada en metros (m)
b_2	Distancia del centro de gravedad a la suspensión derecha expresada en metros (m)

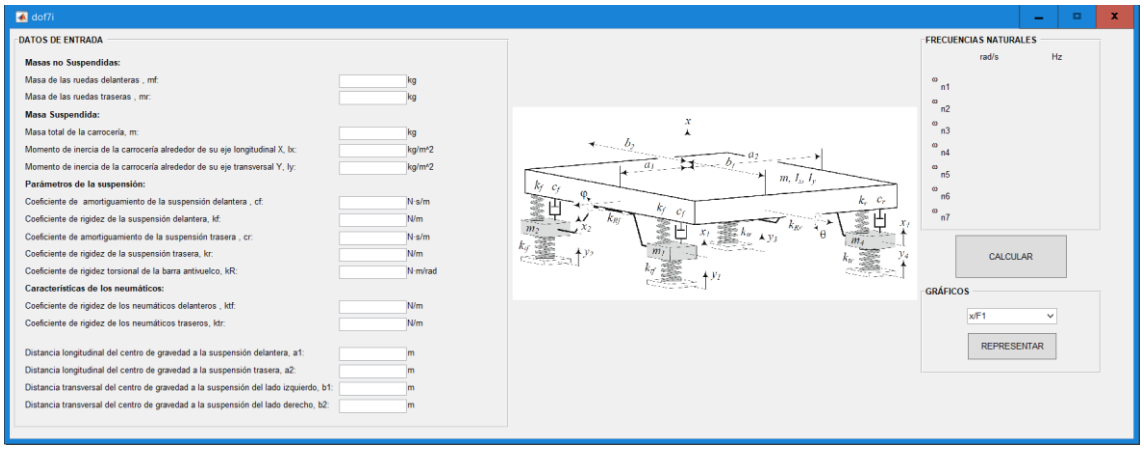
Ventana del modelo de cuatro grados de libertad con suspensión de eje rígido.



Los parámetros necesarios para realizar el análisis son:

Parámetro	Significado
m_1	Masa no suspendida, masa del eje rígido de la suspensión. La cuál deberá expresarse en quilogramos (kg)
m_2	Masa suspendida, 1/2 de la masa de la carrocería. La cuál deberá expresarse en quilogramos (kg)
I_{x1}	1/2 del momento de inercia del eje rígido de la suspensión alrededor de su eje longitudinal, expresado en kg/m^2
I_{x2}	1/2 del momento de inercia de la carrocería alrededor de su eje longitudinal, expresado en kg/m^2
c	Coeficiente de amortiguamiento de la suspensión, cuyas unidades deberán ser N·s/m
k	Coeficiente de rigidez de la suspensión, cuyas unidades deberán ser N/m
k_t	Coeficiente de rigidez de los neumáticos, cuyas unidades deberán ser N/m
a_1	Distancia del centro de gravedad a la rueda izquierda expresada en metros (m)
a_2	Distancia del centro de gravedad a la rueda derecha expresada en metros (m)
b_1	Distancia del centro de gravedad al punto de anclaje de la suspensión izquierda expresada en metros (m)
b_2	Distancia del centro de gravedad al punto de anclaje de la suspensión derecha expresada en metros (m)

Ventana del modelo de siete grados de libertad.



Los parámetros necesarios para realizar el análisis son:

Parámetro	*Significado*
m	Masa suspendida, masa total de la carrocería en kilogramos (kg)
m_1	Masa no suspendida, masa de la rueda delantera izquierda en kilogramos (kg)
m_2	Masa no suspendida, masa de la rueda delantera derecha en kilogramos (kg)
m_3	Masa no suspendida, masa de la rueda trasera derecha en kilogramos (kg)
m_4	Masa no suspendida, masa de la rueda trasera izquierda en kilogramos (kg)
Ix	Momento de inercia de la carrocería alrededor de su eje longitudinal expresado en kg/m^2
Iy	Momento de inercia de la carrocería alrededor de su eje transversal expresado en kg/m^2
cf	Coeficiente de amortiguamiento de la suspensión delantera expresado en N·s/m
cr	Coeficiente de amortiguamiento de la suspensión trasera expresado en N·s/m
kf	Coeficiente de rigidez de la suspensión delantera expresado en N/m
kr	Coeficiente de rigidez de la suspensión trasera expresado en N/m
kt_f	Coeficiente de rigidez de los neumáticos delanteros expresado en N/m
kt_r	Coeficiente de rigidez de los neumáticos traseros expresado en N/m
kR_f	Coeficiente de rigidez torsional de la barra antivuelco delantera expresado en N·m/rad
a_1	Distancia longitudinal del centro de gravedad a la suspensión delantera expresada en metros (m)
a_2	Distancia longitudinal del centro de gravedad a la suspensión trasera expresada en metros (m)
b_1	Distancia transversal del centro de gravedad a la suspensión del lado izquierdo expresada en metros (m)
b_2	Distancia transversal del centro de gravedad a la suspensión del lado derecho expresada en metros (m)

ANEXO B. CÓDIGO DE PROGRAMACIÓN

Ventana de inicio.

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

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

% Last Modified by GUIDE v2.5 01-Sep-2017 10:29:38

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

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
```

```
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Inicio is made visible.
function Inicio_OpeningFcn(hObject, eventdata, handles,
varargin)

% Fondo de pantalla

a=imread('presentacion.jpg');
image(a)
axis off

% Centrar ventana

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)]);

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

% Update handles structure
guidata(hObject, handles);

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


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

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

function gdl1_Callback(hObject, eventdata, handles)
dof1;

function gdl2_Callback(hObject, eventdata, handles)
dof2;
```

```
function gdl3_Callback(hObject, eventdata, handles)
dof3;

function gdl4pitch_Callback(hObject, eventdata, handles)
dof4i_pitch;

function gdl4roll_Callback(hObject, eventdata, handles)
dof4i_roll;

function gdl4rigido_Callback(hObject, eventdata, handles)
dof4r;

function gdl7_Callback(hObject, eventdata, handles)
dof7i;

function figure1_CloseRequestFcn(hObject, eventdata, handles)

opc=questdlg('¿Desea salir del programa?','SA-
LIR','Si','No','No');
if strcmp(opc,'No')
    return;
end

% Hint: delete(hObject) closes the figure

delete(hObject);

function axes1_CreateFcn(hObject, eventdata, handles)
```

Ventana de introducción de datos del modelo de un grado de libertad.

```
function varargout = dof1(varargin)
% DOF1 MATLAB code for dof1.fig
%      DOF1, by itself, creates a new DOF1 or raises the exist-
ing
%      singleton*.
%
%      H = DOF1 returns the handle to a new DOF1 or the handle
to
%      the existing singleton*.
%
%      DOF1('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in DOF1.M with the given input
arguments.
%
%      DOF1('Property','Value',...) creates a new DOF1 or raises
the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before dof1_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes prop-
erty application
%      stop.  All inputs are passed to dof1_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI al-
lows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help dof1

% Last Modified by GUIDE v2.5 06-Sep-2017 15:24:09

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @dof1_OpeningFcn, ...
                   'gui_OutputFcn',  @dof1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT

function dof1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to dof1 (see VARARGIN)

% Centrar ventana

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)]);

% Choose default command line output for dof1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes dof1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

function varargout = dof1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

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

function calcular_Callback(hObject, eventdata, handles)

global ms1 ks1

% Frecuencia natural del Sistema

wn1 = sqrt(ks1/ms1);

set(handles.wn1r,'String',wn1);

% Frecuencia natural del Sistema en Hz

wn1z =(1/(2*pi))*sqrt(ks1/ms1);

set(handles.wn1h,'String',wn1z);
```

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

% Hints: contents = cellstr(get(hObject,'String')) returns FT
contents as cell array
%        contents{get(hObject,'Value')} returns selected item
from FT

function FT_CreateFcn(hObject, eventdata, handles)
% hObject    handle to FT (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function representar_Callback(hObject, eventdata, handles)
global valor1
valor1 = get(handles.FT, 'Value');
dof1_res;

function ms_Callback(hObject, eventdata, handles)
global ms1
ms1 = str2double(get(hObject,'String'));

function ms_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ms (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function cs_Callback(hObject, eventdata, handles)
global cs1
cs1 = str2double(get(hObject,'String'));

function cs_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cs (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function ks_Callback(hObject, eventdata, handles)
global ks1
ks1 = str2double(get(hObject,'String'));

function ks_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ks (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function axes1_CreateFcn(hObject, eventdata, handles)
imshow(imread('1gdl.PNG'));


% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)

opc=questdlg('¿Seguro que desea cerrar esta ventana?','SA-
LIR','Si','No','No');
if strcmp(opc,'No')
    return;
end

% Hint: delete(hObject) closes the figure
delete(hObject);
```

Ventada de gráficos del modelo de un grado de libertad.

```
function varargout = dof1_res(varargin)
% DOF1_RES MATLAB code for dof1_res.fig
%      DOF1_RES, by itself, creates a new DOF1_RES or raises the
existing
%      singleton*.
%
%      H = DOF1_RES returns the handle to a new DOF1_RES or the
handle to
%      the existing singleton*.
%
%      DOF1_RES('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in DOF1_RES.M with the given in-
put arguments.
%
%      DOF1_RES('Property','Value',...) creates a new DOF1_RES
or raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before dof1_res_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes prop-
erty application
%      stop.  All inputs are passed to dof1_res_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI al-
lows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help dof1_res

% Last Modified by GUIDE v2.5 27-Aug-2017 18:50:03

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @dof1_res_OpeningFcn, ...
                   'gui_OutputFcn',  @dof1_res_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT


% --- Executes just before dof1_res is made visible.
function dof1_res_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to dof1_res (see VARARGIN)

% Centrar ventana

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)]);

% Choose default command line output for dof1_res
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes dof1_res wait for user response (see UIRESUME)
% uiwait(handles.figure1);

function varargout = dof1_res_OutputFcn(hObject, eventdata, han-
dles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

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

function figure1_CloseRequestFcn(hObject, eventdata, handles)
opc=questdlg('¿Seguro que desea cerrar esta ventana?','SA-
LIR','Si','No','No');
if strcmp(opc,'No')
    return;
end

% Hint: delete(hObject) closes the figure
delete(hObject);

function axes1_CreateFcn(hObject, eventdata, handles)
```

```
global ms1 cs1 ks1 valor1

% Definición de un vector de frecuencias para utilizar rad/s

w = 0:0.01:10e2;

% precalcular la conversión de radianes a grados

rad2deg = 180/pi;

% definición de s como la multiplicación del operador imaginario
por el
% vector de frecuencias en radianes

s = j*w;

% definición de la función de respuesta en frecuencia

xfer = 1 ./ (ms1*s.^2 + cs1*s + ks1);

% cálculo de la magnitud y la fase para la respuesta en frecuen-
cia

mag = abs(xfer);
phs = angle(xfer)*rad2deg;

switch valor1
    case 1
        subplot(2,1,1)
        loglog(w,mag)
        title('Respuesta en frecuencia para 1 GDL , x/F')
        ylabel('magnitud (dB)')
        grid on
        subplot(2,1,2)
        semilogx(w,phs)
        xlabel('frecuencia, rad/sec')
        ylabel('fase (deg)')
        grid on
end
```

Ventana de introducción de datos del modelo de dos grados de libertad.

```
function varargout = dof2(varargin)
%DOF2 MATLAB code file for dof2.fig
%      DOF2, by itself, creates a new DOF2 or raises the exist-
ing
%      singleton*.
%
%      H = DOF2 returns the handle to a new DOF2 or the handle
to
%      the existing singleton*.
%
%      DOF2('Property','Value',...) creates a new DOF2 using the
%      given property value pairs. Unrecognized properties are
passed via
%      varargin to dof2_OpeningFcn.  This calling syntax pro-
duces a
%      warning when there is an existing singleton*.
%
%      DOF2('CALLBACK') and DOF2('CALLBACK',hObject,...) call
the
%      local function named CALLBACK in DOF2.M with the given
input
%      arguments.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI al-
lows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help dof2

% Last Modified by GUIDE v2.5 06-Sep-2017 15:25:41

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @dof2_OpeningFcn, ...
                   'gui_OutputFcn',  @dof2_OutputFcn, ...
                   'gui_LayoutFcn',  [], ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

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

function dof2_OpeningFcn(hObject, eventdata, handles, varargin)
```

```
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   unrecognized PropertyName/PropertyValue pairs from
the
%            command line (see VARARGIN)

% Centrar ventana

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)]);

% Choose default command line output for dof2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes dof2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

function varargout = dof2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

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

function axes1_CreateFcn(hObject, eventdata, handles)
imshow(imread('2gdl.PNG'));

function representar_Callback(hObject, eventdata, handles)

global valor2

valor2 = get(handles.FT, 'Value');

dof2_res;

function ms_Callback(hObject, eventdata, handles)
global ms2
ms2 = str2double(get(hObject,'String'));

% --- Executes during object creation, after setting all proper-
ties.
```

```
function ms_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ms (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function mu_Callback(hObject, eventdata, handles)
global mu2
mu2 = str2double(get(hObject,'String'));

function mu_CreateFcn(hObject, eventdata, handles)
% hObject    handle to mu (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function cs_Callback(hObject, eventdata, handles)
global cs2
cs2 = str2double(get(hObject,'String'));

function cs_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cs (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function ks_Callback(hObject, eventdata, handles)
global ks2
ks2 = str2double(get(hObject,'String'));
```

```
function ks_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ks (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function ku_Callback(hObject, eventdata, handles)
global ku2
ku2 = str2double(get(hObject,'String'));

function ku_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ku (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function figure1_CloseRequestFcn(hObject, eventdata, ~)

opc=questdlg('¿Seguro que desea cerrar esta ventana?','SA-
LIR','Si','No','No');
if strcmp(opc,'No')
    return;
end
% Hint: delete(hObject) closes the figure
delete(hObject);

function calcular_Callback(hObject, eventdata, handles)

global ms2 mu2 cs2 ks2 ku2

% Matriz de masa

M = [ms2 0;0 mu2];

% Matriz de amortiguamiento

C = [cs2 -cs2;-cs2 cs2];
```

```
% Matriz de rigidez

K = [ks2 -ks2;-ks2 ks2+ku2];

Mr = sqrtm(M);

Kt = inv(Mr)*K*inv(Mr);

[V,D] = eig(Kt); % vectores (V) y valores propios (D)

W = sqrt(diag(D)); % frecuencias naturales en rad/s
wn1 = W(1,1);
wn2 = W(2,1);

Wz = (1/(2*pi))*W; % frecuencias naturales en Hz
wn1z = Wz(1,1);
wn2z = Wz(2,1);

set(handles.wn1r,'String',wn1);
set(handles.wn2r,'String',wn2);
set(handles.wn1h,'String',wn1z);
set(handles.wn2h,'String',wn2z);

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

% Hints: contents = cellstr(get(hObject,'String')) returns FT
contents as cell array
%        contents{get(hObject,'Value')} returns selected item
from FT

function FT_CreateFcn(hObject, eventdata, handles)
% hObject    handle to FT (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

Ventada de gráficos del modelo de dos grados de libertad.

```
function varargout = dof2_res(varargin)
% DOF2_RES MATLAB code for dof2_res.fig
%      DOF2_RES, by itself, creates a new DOF2_RES or raises the
existing
%      singleton*.
%
%      H = DOF2_RES returns the handle to a new DOF2_RES or the
handle to
%      the existing singleton*.
%
%      DOF2_RES('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in DOF2_RES.M with the given in-
put arguments.
%
%      DOF2_RES('Property','Value',...) creates a new DOF2_RES
or raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before dof2_res_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes prop-
erty application
%      stop.  All inputs are passed to dof2_res_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI al-
lows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help dof2_res

% Last Modified by GUIDE v2.5 06-Sep-2017 15:27:59

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @dof2_res_OpeningFcn, ...
                   'gui_OutputFcn',  @dof2_res_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT

function dof2_res_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to dof2_res (see VARARGIN)

% Centrar ventana

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)]);

% Choose default command line output for dof2_res
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes dof2_res wait for user response (see UIRESUME)
% uiwait(handles.figure1);

function varargout = dof2_res_OutputFcn(hObject, eventdata, han-
dles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

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

function figure1_CloseRequestFcn(hObject, eventdata, handles)
opc=questdlg('¿Seguro que desea cerrar esta ventana?','SA-
LIR','Si','No','No');
if strcmp(opc,'No')
    return;
end

% Hint: delete(hObject) closes the figure
delete(hObject);

function axes1_CreateFcn(hObject, eventdata, handles)

global ms2 mu2 cs2 ks2 ku2 valor2
```

```
F = 1;

% matriz del sistema, a

a = [ 0 1 0 0
-ks2/ms2 -cs2/ms2 ks2/ms2 cs2/ms2
0 0 0 1
ks2/mu2 cs2/mu2 -(ks2+ku2)/mu2 -cs2/mu2];

% matriz de entrada, b

b = [ 0; 0; 0; F/mu2 ];

% matriz de salida, c

c = eye(4,4);

% matriz de transmisión directa, d

d = 0;

% resolver los valores propios de la matriz del sistema

[xm,omega] = eig(a);

% Definición de un vector de frecuencias para utilizar, radia-
nes/seg.

w = 0:0.1:10e2;

% utilizar la función "ss" para definir el sistema de espacio de
estado

sssys = ss(a,b,c,d);

% cálculo de la magnitud y la fase mediante el comando bode.
% primer índice 1-4: z1 z1dot z2 z2dot
% segundo índice 1: F1 (pues solo hay una fuerza)
% tercer índice 1-200: extrae todos los puntos de la frecuencia
utilizando ":"

[mag,phs] = bode(sssys,w);

z12mag = mag(1,1,:);
z12phs = phs(1,1,:);

z22mag = mag(3,1,:);
z22phs = phs(3,1,:);


% calculo de la magnitud en decibelios, db

z12magdb = 20*log10(z12mag);

z22magdb = 20*log10(z22mag);
```

```
% representación de las funciones de transferencia

switch valor2
    case 1
        subplot(2,1,1)
        semilogx(w,z12magdb(1,:))
        title('Función de transferencia para 2 GDL, x_s/F')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,z12phs(1,:))
        ylabel('fase, deg')
        xlabel('frecuencia (rad/s)')
        grid
    case 2
        subplot(2,1,1)
        semilogx(w,z22magdb(1,:))
        title('Función de transferencia para 2 GDL, x_u/F')
        ylabel('magnitude, db')
        grid
        subplot(2,1,2)
        semilogx(w,z22phs(1,:))
        ylabel('phase, deg')
        xlabel('frecuencia (rad/s)')
        grid
end
```

Ventana de introducción de datos del modelo de tres grados de libertad.

```
function varargout = dof3(varargin)
%DOF3 MATLAB code file for dof3.fig
%      DOF3, by itself, creates a new DOF3 or raises the exist-
ing
%      singleton*.
%
%      H = DOF3 returns the handle to a new DOF3 or the handle
to
%      the existing singleton*.
%
%      DOF3('Property','Value',...) creates a new DOF3 using the
%      given property value pairs. Unrecognized properties are
passed via
%      varargin to dof3_OpeningFcn.  This calling syntax pro-
duces a
%      warning when there is an existing singleton*.
%
%      DOF3('CALLBACK') and DOF3('CALLBACK',hObject,...) call
the
%      local function named CALLBACK in DOF3.M with the given
input
%      arguments.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI al-
lows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help dof3

% Last Modified by GUIDE v2.5 06-Sep-2017 15:31:37

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @dof3_OpeningFcn, ...
                   'gui_OutputFcn',  @dof3_OutputFcn, ...
                   'gui_LayoutFcn',  [], ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

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

function dof3_OpeningFcn(hObject, eventdata, handles, varargin)
```

```
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   unrecognized PropertyName/PropertyValue pairs from
the
%            command line (see VARARGIN)

% Centrar ventana

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)]);

% Choose default command line output for dof3
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes dof3 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

function varargout = dof3_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

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

function axes1_CreateFcn(hObject, eventdata, handles)
imshow(imread('3gdl.PNG'));

function representar_Callback(hObject, eventdata, handles)

global valor3

valor3 = get(handles.FT, 'Value');

dof3_res;

function ms_Callback(hObject, eventdata, handles)
global ms3
ms3 = str2double(get(hObject,'String'));

function ms_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ms (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function mu_Callback(hObject, eventdata, handles)
global mu3
mu3 = str2double(get(hObject,'String'));

function mu_CreateFcn(hObject, eventdata, handles)
% hObject    handle to mu (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function cs_Callback(hObject, eventdata, handles)
global cs3
cs3 = str2double(get(hObject,'String'));

function cs_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cs (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function ks_Callback(hObject, eventdata, handles)
global ks3
ks3 = str2double(get(hObject,'String'));

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

```
% hObject    handle to ks (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function ku_Callback(hObject, eventdata, handles)
global ku3
ku3 = str2double(get(hObject,'String'));

function ku_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ku (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function figure1_CloseRequestFcn(hObject, eventdata, ~)

opc=questdlg('¿Seguro que desea cerrar esta ventana?','SA-
LIR','Si','No','No');
if strcmp(opc,'No')
    return;
end

% Hint: delete(hObject) closes the figure
delete(hObject);

function kd_Callback(hObject, eventdata, handles)
global kd3
kd3 = str2double(get(hObject,'String'));

function kd_CreateFcn(hObject, eventdata, handles)
% hObject    handle to kd (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called
```

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

function cd_Callback(hObject, eventdata, handles)
global cd3
cd3 = str2double(get(hObject,'String'));

function cd_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cd (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function md_Callback(hObject, eventdata, handles)
global md3
md3 = str2double(get(hObject,'String'));

function md_CreateFcn(hObject, eventdata, handles)
% hObject    handle to md (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function calcular_Callback(hObject, eventdata, handles)

global ms3 mu3 md3 cs3 cd3 ks3 kd3 ku3

% Matriz de masa

M=[mu3 0 0
    0 ms3 0
    0 0 md3];
```

```
% Matriz de amortiguamiento

c11=cs3;
c12=-cs3;
c13=0;

c21=-cs3;
c22=cd3+cs3;
c23=-cd3;

c31=0;
c32=-cd3;
c33=cd3;

C=[c11 c12 c13
    c21 c22 c23
    c31 c32 c33];

% Matriz de rigidez

k11=ks3+ku3;
k12=-ks3;
k13=0;

k21=-ks3;
k22=kd3+ks3;
k23=-kd3;

k31=0;
k32=-kd3;
k33=kd3;

K = [ks3+ku3 -ks3 0
    -ks3 kd3+ks3 -kd3
    0 -kd3 kd3];

Mr = sqrtm(M);

Kt = inv(Mr)*K*inv(Mr);

[V,D] = eig(Kt); % vectores (V) y valores propios (D)

W = sqrt(diag(D)); % frecuencias naturales en rad/s
wn1 = W(1,1);
wn2 = W(2,1);
wn3 = W(3,1);

Wz = (1/(2*pi))*W; % frecuencias naturales en Hz
wn1z = Wz(1,1);
wn2z = Wz(2,1);
wn3z = Wz(3,1);

set(handles.wn1r,'String',wn1);
set(handles.wn2r,'String',wn2);
set(handles.wn3r,'String',wn3);
set(handles.wn1h,'String',wn1z);
```

```
set(handles.wn2h,'String',wn2z);
set(handles.wn3h,'String',wn3z);

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

% Hints: contents = cellstr(get(hObject,'String')) returns FT
contents as cell array
%        contents{get(hObject,'Value')} returns selected item
from FT

function FT_CreateFcn(hObject, eventdata, handles)
% hObject    handle to FT (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

Ventada de gráficos del modelo de tres grados de libertad.

```
function varargout = dof3_res(varargin)
% DOF3_RES MATLAB code for dof3_res.fig
%      DOF3_RES, by itself, creates a new DOF3_RES or raises the
existing
%      singleton*.
%
%      H = DOF3_RES returns the handle to a new DOF3_RES or the
handle to
%      the existing singleton*.
%
%      DOF3_RES('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in DOF3_RES.M with the given in-
put arguments.
%
%      DOF3_RES('Property','Value',...) creates a new DOF3_RES
or raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before dof3_res_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes prop-
erty application
%      stop.  All inputs are passed to dof3_res_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI al-
lows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help dof3_res

% Last Modified by GUIDE v2.5 06-Sep-2017 15:32:30

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @dof3_res_OpeningFcn, ...
                   'gui_OutputFcn',  @dof3_res_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT

function dof3_res_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to dof3_res (see VARARGIN)

% Centrar ventana

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)]);

% Choose default command line output for dof3_res
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes dof3_res wait for user response (see UIRESUME)
% uiwait(handles.figure1);

function varargout = dof3_res_OutputFcn(hObject, eventdata, han-
dles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

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

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

% Hints: contents = cellstr(get(hObject,'String')) returns FT
contents as cell array
%        contents{get(hObject,'Value')} returns selected item
from FT

function FT_CreateFcn(hObject, eventdata, handles)
% hObject    handle to FT (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function figure1_CloseRequestFcn(hObject, eventdata, handles)

opc=questdlg('¿Seguro que desea cerrar esta ventana?','SA-
LIR','Si','No','No');
if strcmp(opc,'No')
    return;
end

% Hint: delete(hObject) closes the figure
delete(hObject);

function axes1_CreateFcn(hObject, eventdata, handles)

global ms3 mu3 md3 cs3 cd3 ks3 kd3 ku3 valor3

F1 = 1;

% matriz del sistema, a

a = [ 0 1 0 0 0 0
-(ks3+ku3)/mu3 -cs3/mu3 ks3/mu3 cs3/mu3 0 0
0 0 0 1 0 0
ks3/ms3 cs3/ms3 -(kd3+ks3)/ms3 -(cd3+cs3)/ms3 kd3/ms3 cd3/ms3
0 0 0 0 0 1
0 0 kd3/md3 cd3/md3 -kd3/md3 -cd3/md3];

% matriz de entrada, b

b = [ 0 ; F1/mu3; 0; 0; 0; 0];

% matriz de salida, c

c = eye(6,6);

% matriz de transmision directa, d

d = 0;

% valores y vectores propios del sistema

[xm,omega] = eig(a);
```

```
% Definición de un vector de frecuencias para utilizar, radia-
nes/seg.

w = 0:0.01:10e2;

% utilizar la función "ss" para definir el sistema de espacio de
estado

sssys = ss(a,b,c,d);

% cálculo de la magnitud y la fase mediante el comando bode.
% primer índice 1-6: z1 z1dot z2 z2dot z3 z3dot
% segundo índice 1: F1 (pues solo hay una fuerza)
% tercer índice 1-200: extrae todos los puntos de la frecuencia
utilizando ":"

[mag,phs] = bode(sssys,w);

z11mag = mag(1,1,:);
z11phs = phs(1,1,:);

z21mag = mag(3,1,:);
z21phs = phs(3,1,:);

z31mag = mag(5,1,:);
z31phs = phs(5,1,:);

% calculo de la magnitud en decibelios, db

z11magdb = 20*log10(z11mag);

z21magdb = 20*log10(z21mag);

z31magdb = 20*log10(z31mag);

% representación de las funciones de transferencia

switch valor3
    case 1
        subplot(2,1,1)
        semilogx(w,z11magdb(1,:))
        title('Función de transferencia para 3 GDL, x_u/F')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,z11phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('fase, deg')
        grid
    case 2
        subplot(2,1,1)
        semilogx(w,z21magdb(1,:))
        title('Función de transferencia para 3 GDL, x_s/F')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
```

```
        semilogx(w,z21phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('fase, deg')
        grid
    case 3
        subplot(2,1,1)
        semilogx(w,z31magdb(1,:))
        title('Función de transferencia para 3 GDL, x_d/F')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,z31phs(1,:))
        ylabel('fase, deg')
        xlabel('Frecuencia, rad/s')
        grid
end
```

Ventana de introducción de datos del modelo de cuatro grados de libertad con suspensión independiente y movimiento de cabeceo.

```
function varargout = dof4i_pitch(varargin)
% DOF4I_PITCH MATLAB code for dof4i_pitch.fig
%      DOF4I_PITCH, by itself, creates a1 new DOF4I_PITCH or
raises the existing
%      singleton*.
%
%      H = DOF4I_PITCH returns the handle to a1 new DOF4I_PITCH
or the handle to
%      the existing singleton*.
%
%      DOF4I_PITCH('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in DOF4I_PITCH.m with the given
input arguments.
%
%      DOF4I_PITCH('Property','Value',...) creates a1 new
DOF4I_PITCH or raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before dof4i_pitch_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes prop-
erty application
%      stop.  All inputs are passed to dof4i_pitch_OpeningFcn
via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI al-
lows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help dof4i_pitch

% Last Modified by GUIDE v2.5 28-Aug-2017 14:01:38

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @dof4i_pitch_OpeningFcn,
...
                   'gui_OutputFcn',  @dof4i_pitch_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
```

```
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before dof4i_pitch is made visible.
function dof4i_pitch_OpeningFcn(hObject, eventdata, handles,
varargin)

% Centrar ventana

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)]);

% Choose default command line output for dof4i_pitch
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes dof4i_pitch wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command
line.
function varargout = dof4i_pitch_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a1 future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

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


% --- Executes on button press in calcular.
function calcular_Callback(hObject, eventdata, handles)

global m4p m14p m24p Iy4p c14p c24p k14p k24p kt14p kt24p a14p
a24p

% Matriz de masa

M=[m4p 0 0 0; 0 Iy4p 0 0; 0 0 m14p 0; 0 0 0 m24p];

% Matriz de amortiguamiento
```

```
c11 = c14p+c24p;
c12 = c24p*a24p-c14p*a14p;
c13 = -c14p;
c14 = -c24p;

c21 = c24p*a24p-c14p*a14p;
c22 = c14p*a14p^2+c24p*a24p^2;
c23 = c14p*a14p;
c24 = -c24p*a24p;

c31 = -c14p;
c32 = c14p*a14p;
c33 = c14p;
c34 = 0;

c41 = -c24p;
c42 = -c24p*a24p;
c43 = 0;
c44 = c24p;

C=[c11 c12 c13 c14; c21 c22 c23 c24; c31 c32 c33 c34; c41 c42
c43 c44];

% Matriz de rigidez

k11 = k14p+k24p;
k12 = k24p*a24p-k14p*a14p;
k13 = -k14p;
k14 = -k24p;

k21 = k24p*a24p-k14p*a14p;
k22 = k14p*a14p^2+k24p*a24p^2;
k23 = k14p*a14p;
k24 = -k24p*a24p;

k31 = -k14p;
k32 = k14p*a14p;
k33 = k14p+kt14p;
k34 = 0;

k41 = -k24p;
k42 = -k24p*a24p;
k43 = 0;
k44 = k24p+kt24p;

K=[k11 k12 k13 k14
    k21 k22 k23 k24
    k31 k32 k33 k34
    k41 k42 k43 k44];

Mr = sqrtm(M);
Kt = inv(Mr)*K*inv(Mr);
[V,D] = eig(Kt);

W = sqrt(diag(D));% frecuencias naturales en rad/s
wn1 = W(1,1);
```

```
wn2 = W(2,1);
wn3 = W(3,1);
wn4 = W(4,1);

Wz = (1/(2*pi))*W; % frecuencias naturales en Hz
wn1z = Wz(1,1);
wn2z = Wz(2,1);
wn3z = Wz(3,1);
wn4z = Wz(4,1);

set(handles.wn1r,'String',wn1);
set(handles.wn2r,'String',wn2);
set(handles.wn3r,'String',wn3);
set(handles.wn4r,'String',wn4);

set(handles.wn1h,'String',wn1z);
set(handles.wn2h,'String',wn2z);
set(handles.wn3h,'String',wn3z);
set(handles.wn4h,'String',wn4z);


function a1_Callback(hObject, eventdata, handles)
global a14p
a14p = str2double(get(hObject,'String'));


% --- Executes during object creation, after setting all proper-
ties.
function a1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to a1 (see GCBO)
% eventdata  reserved - to be defined in a1 future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: edit controls usually have a1 white background on Win-
dows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function a2_Callback(hObject, eventdata, handles)
global a24p
a24p = str2double(get(hObject,'String'));


% --- Executes during object creation, after setting all proper-
ties.
function a2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to a2 (see GCBO)
% eventdata  reserved - to be defined in a1 future version of
MATLAB
```

```
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: edit controls usually have a1 white background on Win-
dows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function m_Callback(hObject, eventdata, handles)
global m4p
m4p = str2double(get(hObject,'String'));


% --- Executes during object creation, after setting all proper-
ties.
function m_CreateFcn(hObject, eventdata, handles)
% hObject    handle to m (see GCBO)
% eventdata  reserved - to be defined in a1 future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: edit controls usually have a1 white background on Win-
dows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function Iy_Callback(hObject, eventdata, handles)
global Iy4p
Iy4p = str2double(get(hObject,'String'));


% --- Executes during object creation, after setting all proper-
ties.
function Iy_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Iy (see GCBO)
% eventdata  reserved - to be defined in a1 future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: edit controls usually have a1 white background on Win-
dows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function k1_Callback(hObject, eventdata, handles)
global k14p
k14p = str2double(get(hObject,'String'));


% --- Executes during object creation, after setting all proper-
ties.
function k1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to k1 (see GCBO)
% eventdata  reserved - to be defined in a1 future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: edit controls usually have a1 white background on Win-
dows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function c1_Callback(hObject, eventdata, handles)
global c14p
c14p = str2double(get(hObject,'String'));


% --- Executes during object creation, after setting all proper-
ties.
function c1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to c1 (see GCBO)
% eventdata  reserved - to be defined in a1 future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: edit controls usually have a1 white background on Win-
dows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function m1_Callback(hObject, eventdata, handles)
global m14p
m14p = str2double(get(hObject,'String'));


% --- Executes during object creation, after setting all proper-
ties.
function m1_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to m1 (see GCBO)
% eventdata  reserved - to be defined in a1 future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: edit controls usually have a1 white background on Win-
dows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function k2_Callback(hObject, eventdata, handles)
global k24p
k24p = str2double(get(hObject,'String'));


% --- Executes during object creation, after setting all proper-
ties.
function k2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to k2 (see GCBO)
% eventdata  reserved - to be defined in a1 future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: edit controls usually have a1 white background on Win-
dows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function m2_Callback(hObject, eventdata, handles)
global m24p
m24p = str2double(get(hObject,'String'));

% --- Executes during object creation, after setting all proper-
ties.
function m2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to m2 (see GCBO)
% eventdata  reserved - to be defined in a1 future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: edit controls usually have a1 white background on Win-
dows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
```

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


% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)

opc=questdlg('¿Seguro que desea cerrar esta ventana?','SA-
LIR','Si','No','No');
if strcmp(opc,'No')
    return;
end

% Hint: delete(hObject) closes the figure
delete(hObject);


% --- Executes during object creation, after setting all proper-
ties.
function axes1_CreateFcn(hObject, eventdata, handles)
imshow(imread('4gdl_pitch.PNG'));


% --- Executes on selection change in FT.
function FT_Callback(hObject, eventdata, handles)
% hObject    handle to FT (see GCBO)
% eventdata  reserved - to be defined in a1 future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns FT
contents as cell array
%        contents{get(hObject,'Value')} returns selected item
from FT


% --- Executes during object creation, after setting all proper-
ties.
function FT_CreateFcn(hObject, eventdata, handles)
% hObject    handle to FT (see GCBO)
% eventdata  reserved - to be defined in a1 future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: popupmenu controls usually have a1 white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in representar.
function representar_Callback(hObject, eventdata, handles)
```

```
global valor4p

valor4p = get(handles.FT,'Value');

dof4_pitch_res;


function kt1_Callback(hObject, eventdata, handles)
global kt14p
kt14p = str2double(get(hObject,'String'));

% --- Executes during object creation, after setting all proper-
ties.
function kt1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to kt1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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


function kt2_Callback(hObject, eventdata, handles)
global kt24p
kt24p = str2double(get(hObject,'String'));


% --- Executes during object creation, after setting all proper-
ties.
function kt2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to kt2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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


function c2_Callback(hObject, eventdata, handles)
global c24p
c24p = str2double(get(hObject,'String'));
```

```
% --- Executes during object creation, after setting all proper-
ties.
function c2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to c2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

Ventana de gráficos del modelo de cuatro grados de libertad con suspensión independiente y movimiento de cabeceo.

```
function varargout = dof4_pitch_res(varargin)
% DOF4_PITCH_RES MATLAB code for dof4_pitch_res.fig
%      DOF4_PITCH_RES, by itself, creates a new DOF4_PITCH_RES
or raises the existing
%      singleton*.
%
%      H = DOF4_PITCH_RES returns the handle to a new
DOF4_PITCH_RES or the handle to
%      the existing singleton*.
%
%      DOF4_PITCH_RES('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in DOF4_PITCH_RES.M with the
given input arguments.
%
%      DOF4_PITCH_RES('Property','Value',...) creates a new
DOF4_PITCH_RES or raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before dof4_pitch_res_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes prop-
erty application
%      stop.  All inputs are passed to dof4_pitch_res_OpeningFcn
via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI al-
lows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
dof4_pitch_res

% Last Modified by GUIDE v2.5 28-Aug-2017 10:38:46

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @dof4_pitch_res_OpeningFcn,
...
                   'gui_OutputFcn',  @dof4_pitch_res_OutputFcn,
...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
```

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

function dof4_pitch_res_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to dof4_pitch_res (see
VARARGIN)

% Centrar ventana

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)]);

% Choose default command line output for dof4_pitch_res
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes dof4_pitch_res wait for user response (see
UIRESUME)
% uiwait(handles.figure1);

function varargout = dof4_pitch_res_OutputFcn(hObject, event-
data, handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

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

function figure1_CloseRequestFcn(hObject, eventdata, handles)
opc=questdlg('¿Seguro que desea cerrar esta ventana?','SA-
LIR','Si','No','No');
if strcmp(opc,'No')
    return;
end
```

```
% Hint: delete(hObject) closes the figure
delete(hObject);

function axes1_CreateFcn(hObject, eventdata, handles)

global m4p m14p m24p Iy4p c14p c24p k14p k24p kt14p kt24p a14p
a24p valor4p

F14 = 1;
F24 = 1;

% matriz del sistema, a

a = [ 0 1 0 0 0 0 0 0
-(k14p+k24p)/m4p -(c14p+c24p)/m4p -(k24p*a24p-k14p*a14p)/m4p -
(c24p*a24p-c14p*a14p)/m4p k14p/m14p c14p/m14p k24p/m14p
c24p/m14p
0 0 0 1 0 0 0 0
-(k24p*a24p-k14p*a14p)/Iy4p -(c24p*a24p-c14p*a14p)/Iy4p -
(k14p*a14p^2+k24p*a24p^2)/Iy4p -(c14p*a14p^2+c24p*a24p^2)/Iy4p -
k14p*a14p/Iy4p -c14p*a14p/Iy4p k24p*a24p/Iy4p c24p*a24p/Iy4p
0 0 0 0 0 1 0 0
k14p/m14p c14p/m14p -k14p*a14p/m4p -c14p*a14p/m14p -
(k14p+kt14p)/m14p -c14p/m4p 0 0
0 0 0 0 0 0 0 1
k24p/m24p c24p/m24p k24p*a24p/m24p c24p*a24p/m24p 0 0 -
(k24p+kt24p)/m24p -c24p/m24p];

% matriz de entrada, b

b = [ 0 0; 0 0; 0 0; 0 0;0 0; F14/m14p 0; 0 0; 0 F24/m24p];

% matriz de salida, c

c = eye(8,8);

% matriz de transmision directa, d

d = 0;

% resolver los valores propios de la matriz del sistema

[xm,omega] = eig(a);

% Definición de un vector de frecuencias para utilizar, radia-
nes/seg.

w = 0:0.01:10e2;

% utilizar la función "ss" para definir el sistema de espacio de
estado

sssys = ss(a,b,c,d);

% cálculo de la magnitud y la fase mediante el comando bode.
% primer índice 1-8: z1 z1dot z2 z2dot z3 z3dot z4 z4dot
```

```
% segundo índice 1-2: F1 F2
% tercer índice 1-200: extrae todos los puntos de la frecuencia
utilizando ":"

[mag,phs] = bode(sssys,w);

z1mag = mag(1,1,:);
z1phs = phs(1,1,:);

t1mag = mag(3,1,:);
t1phs = phs(3,1,:);

z11mag = mag(5,1,:);
z11phs = phs(5,1,:);

z21mag = mag(7,1,:);
z21phs = phs(7,1,:);

z2mag = mag(1,2,:);
z2phs = phs(1,2,:);

t2mag = mag(3,2,:);
t2phs = phs(3,2,:);

z12mag = mag(5,2,:);
z12phs = phs(5,2,:);

z22mag = mag(7,2,:);
z22phs = phs(7,2,:);

% calculo de la magnitud en decibelios, db

z1magdb = 20*log10(z1mag);

t1magdb = 20*log10(t1mag);

z11magdb = 20*log10(z11mag);

z21magdb = 20*log10(z21mag);

z2magdb = 20*log10(z2mag);

t2magdb = 20*log10(t2mag);

z12magdb = 20*log10(z12mag);

z22magdb = 20*log10(z22mag);

% representación de las funciones de transferencia

switch valor4p
    case 1
        subplot(2,1,1)
        semilogx(w,z1magdb(1,:))
        title('Función de transferencia para 4 GDL modo cabeceo,
x/F1')
```

```
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,z1phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('fase, deg')
        grid
    case 2
        subplot(2,1,1)
        semilogx(w,t1magdb(1,:))
        title('Función de transferencia para 4 GDL modo cabeceo,
\theta/F_1')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,t1phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('fase, deg')
        grid
    case 3
        subplot(2,1,1)
        semilogx(w,z11magdb(1,:))
        title('Función de transferencia para 4 GDL modo cabeceo,
x_1/F_1')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,z11phs(1,:))
        ylabel('fase, deg')
        xlabel('Frecuencia, rad/s')
        grid
    case 4
        subplot(2,1,1)
        semilogx(w,z21magdb(1,:))
        title('Función de transferencia para 4 GDL modo cabeceo,
x_2/F_1')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,z21phs(1,:))
        ylabel('fase, deg')
        xlabel('Frecuencia, rad/s')
        grid
    case 5
        subplot(2,1,1)
        semilogx(w,z2magdb(1,:))
        title('Función de transferencia para 4 GDL modo cabeceo,
x/F_2')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,z2phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('phase, deg')
        grid
    case 6
```

```
        subplot(2,1,1)
        semilogx(w,t2magdb(1,:))
        title('Función de transferencia para 4 GDL modo cabeceo,
\theta/F_2')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,t2phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('fase, deg')
        grid
    case 7
        subplot(2,1,1)
        semilogx(w,z12magdb(1,:))
        title('Función de transferencia para 4 GDL modo cabeceo,
x_1/F_2')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,z12phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('fase, deg')
        grid
    case 8
        subplot(2,1,1)
        semilogx(w,z22magdb(1,:))
        title('Función de transferencia para 4 GDL modo cabeceo,
x_2/F_2')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,z22phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('fase, deg')
        grid
end
```

Ventada de introducción de datos del modelo de cuatro grados de libertad con suspensión independiente y movimiento de balanceo.

```
function varargout = dof4i_roll(varargin)
% DOF4I_ROLL MATLAB code for dof4i_roll.fig
%      DOF4I_ROLL, by itself, creates b1 new DOF4I_ROLL or
raises the existing
%      singleton*.
%
%      H = DOF4I_ROLL returns the handle to b1 new DOF4I_ROLL or
the handle to
%      the existing singleton*.
%
%      DOF4I_ROLL('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in DOF4I_ROLL.m with the given
input arguments.
%
%      DOF4I_ROLL('Property','Value',...) creates b1 new
DOF4I_ROLL or raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before dof4i_roll_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes prop-
erty application
%      stop.  All inputs are passed to dof4i_roll_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI al-
lows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help dof4i_roll

% Last Modified by GUIDE v2.5 28-Aug-2017 11:28:37

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @dof4i_roll_OpeningFcn, ...
                   'gui_OutputFcn',  @dof4i_roll_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
```

```
end
% End initialization code - DO NOT EDIT

function dof4i_roll_OpeningFcn(hObject, eventdata, handles,
varargin)

% Centrar ventana

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)]);

% Choose default command line output for dof4i_roll
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes dof4i_roll wait for user response (see UIRESUME)
% uiwait(handles.figure1);

function varargout = dof4i_roll_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in b1 future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

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

function calcular_Callback(hObject, eventdata, handles)

global m4i m14i m24i Ix4i c4i k4i kt4i kR4i b14i b24i

% Matriz de masa

M=[m4i 0 0 0
    0 Ix4i 0 0
    0 0 m14i 0
    0 0 0 m24i];

% Matriz de amortiguamiento

c11 = 2*c4i;
c12 = c4i*b14i-c4i*b24i;
c13 = -c4i;
c14 = -c4i;

c21 = c4i*b14i-c4i*b24i;
```

```
c22 = c4i*b14i^2+c4i*b24i^2;
c23 = -c4i*b14i;
c24 = c4i*b24i;

c31 = -c4i;
c32 = -c4i*b14i;
c33 = c4i;
c34 = 0;

c41 = -c4i;
c42 = c4i*b24i;
c43 = 0;
c44 = c4i;

C=[c11 c12 c13 c14;c21 c22 c23 c24;c31 c32 c33 c34;c41 c42 c43
c44];

% Matriz de rigidez

k11 = 2*k4i;
k12 = k4i*b14i-k4i*b24i;
k13 = -k4i;
k14 = -k4i;

k21 = k4i*b14i-k4i*b24i;
k22 = k4i*b14i^2+k4i*b24i^2+kR4i;
k23 = -k4i*b14i;
k24 = k4i*b24i;

k31 = -k4i;
k32 = -k4i*b14i;
k33 = k4i+kt4i;
k34 = 0;

k41 = -k4i;
k42 = k4i*b24i;
k43 = 0;
k44 = k4i+kt4i;

K=[k11 k12 k13 k14
    k21 k22 k23 k24
    k31 k32 k33 k34
    k41 k42 k43 k44];

Mr = sqrtm(M);
Kt = inv(Mr)*K*inv(Mr);
[V,D] = eig(Kt);

W = sqrt(diag(D));% frecuencias naturales en rad/s
wn1 = W(1,1);
wn2 = W(2,1);
wn3 = W(3,1);
wn4 = W(4,1);

Wz = (1/(2*pi))*W; % frecuencias naturales en Hz
wn1z = Wz(1,1);
```

```
wn2z = Wz(2,1);
wn3z = Wz(3,1);
wn4z = Wz(4,1);

set(handles.wn1r,'String',wn1);
set(handles.wn2r,'String',wn2);
set(handles.wn3r,'String',wn3);
set(handles.wn4r,'String',wn4);

set(handles.wn1h,'String',wn1z);
set(handles.wn2h,'String',wn2z);
set(handles.wn3h,'String',wn3z);
set(handles.wn4h,'String',wn4z);

function b1_Callback(hObject, eventdata, handles)
global b14i
b14i = str2double(get(hObject,'String'));

function b1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to b1 (see GCBO)
% eventdata  reserved - to be defined in b1 future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: edit controls usually have b1 white background on Win-
dows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function b2_Callback(hObject, eventdata, handles)
global b24i
b24i = str2double(get(hObject,'String'));

function b2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to b2 (see GCBO)
% eventdata  reserved - to be defined in b1 future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: edit controls usually have b1 white background on Win-
dows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function m_Callback(hObject, eventdata, handles)
global m4i
m4i = str2double(get(hObject,'String'));
```

```
function m_CreateFcn(hObject, eventdata, handles)
% hObject    handle to m (see GCBO)
% eventdata  reserved - to be defined in b1 future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: edit controls usually have b1 white background on Win-
dows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function Ix_Callback(hObject, eventdata, handles)
global Ix4i
Ix4i = str2double(get(hObject,'String'));

function Ix_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Ix (see GCBO)
% eventdata  reserved - to be defined in b1 future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: edit controls usually have b1 white background on Win-
dows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function k_Callback(hObject, eventdata, handles)
global k4i
k4i = str2double(get(hObject,'String'));

function k_CreateFcn(hObject, eventdata, handles)
% hObject    handle to k (see GCBO)
% eventdata  reserved - to be defined in b1 future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: edit controls usually have b1 white background on Win-
dows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function c_Callback(hObject, eventdata, handles)
global c4i
c4i = str2double(get(hObject,'String'));
```

```
function c_CreateFcn(hObject, eventdata, handles)
% hObject    handle to c (see GCBO)
% eventdata  reserved - to be defined in b1 future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: edit controls usually have b1 white background on Win-
dows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function m1_Callback(hObject, eventdata, handles)
global m14i
m14i = str2double(get(hObject,'String'));

function m1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to m1 (see GCBO)
% eventdata  reserved - to be defined in b1 future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: edit controls usually have b1 white background on Win-
dows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function kR_Callback(hObject, eventdata, handles)
global kR4i
kR4i = str2double(get(hObject,'String'));

function kR_CreateFcn(hObject, eventdata, handles)
% hObject    handle to kR (see GCBO)
% eventdata  reserved - to be defined in b1 future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: edit controls usually have b1 white background on Win-
dows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function m2_Callback(hObject, eventdata, handles)
global m24i
```

```
m24i = str2double(get(hObject,'String'));

function m2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to m2 (see GCBO)
% eventdata  reserved - to be defined in b1 future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: edit controls usually have b1 white background on Win-
dows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function figure1_CloseRequestFcn(hObject, eventdata, handles)

opc=questdlg('¿Seguro que desea cerrar esta ventana?','SA-
LIR','Si','No','No');
if strcmp(opc,'No')
    return;
end

% Hint: delete(hObject) closes the figure
delete(hObject);

function axes1_CreateFcn(hObject, eventdata, handles)
imshow(imread('4gdl_roll.PNG'));

function FT_Callback(hObject, eventdata, handles)
% hObject    handle to FT (see GCBO)
% eventdata  reserved - to be defined in b1 future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns FT
contents as cell array
%        contents{get(hObject,'Value')} returns selected item
from FT

function FT_CreateFcn(hObject, eventdata, handles)
% hObject    handle to FT (see GCBO)
% eventdata  reserved - to be defined in b1 future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: popupmenu controls usually have b1 white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

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

global valor4i

valor4i = get(handles.FT,'Value');

dof4_roll_res;

function kt_Callback(hObject, eventdata, handles)
global kt4i
kt4i = str2double(get(hObject,'String'));

function kt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to kt (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

Ventada de gráficos del modelo de cuatro grados de libertad con suspensión independiente y movimiento de balanceo.

```
function varargout = dof4_roll_res(varargin)
% DOF4_ROLL_RES MATLAB code for dof4_roll_res.fig
%      DOF4_ROLL_RES, by itself, creates a new DOF4_ROLL_RES or
raises the existing
%      singleton*.
%
%      H = DOF4_ROLL_RES returns the handle to a new
DOF4_ROLL_RES or the handle to
%      the existing singleton*.
%
%      DOF4_ROLL_RES('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in DOF4_ROLL_RES.M with the given
input arguments.
%
%      DOF4_ROLL_RES('Property','Value',...) creates a new
DOF4_ROLL_RES or raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before dof4_roll_res_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes prop-
erty application
%      stop.  All inputs are passed to dof4_roll_res_OpeningFcn
via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI al-
lows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
dof4_roll_res

% Last Modified by GUIDE v2.5 28-Aug-2017 10:56:11

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @dof4_roll_res_OpeningFcn,
...
                   'gui_OutputFcn',  @dof4_roll_res_OutputFcn,
...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
```

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


% --- Executes just before dof4_roll_res is made visible.
function dof4_roll_res_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to dof4_roll_res (see
VARARGIN)

% Centrar ventana

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)]);

% Choose default command line output for dof4_roll_res
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes dof4_roll_res wait for user response (see
UIRESUME)
% uiwait(handles.figure1);

function varargout = dof4_roll_res_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

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

function figure1_CloseRequestFcn(hObject, eventdata, handles)
opc=questdlg('¿Seguro que desea cerrar esta ventana?','SA-
LIR','Si','No','No');
if strcmp(opc,'No')
    return;
```

```
end

% Hint: delete(hObject) closes the figure
delete(hObject);

function axes1_CreateFcn(hObject, eventdata, handles)

global m4i m14i m24i Ix4i c4i k4i kt4i kR4i b14i b24i valor4i

F1 = 1;
F2 = 1;

% matriz del sistema, a

a = [ 0 1 0 0 0 0 0 0
k4i/m4i c4i/m4i k4i/m4i c4i/m4i -2*k4i/m4i -2*c4i/m4i -
(k4i*b14i-k4i*b24i)/m4i -(c4i*b14i-c4i*b24i)/m4i
0 0 0 1 0 0 0 0
(k4i*b14i)/Ix4i (c4i*b14i)/Ix4i (-k4i*b24i)/Ix4i (-
c4i*b24i)/Ix4i -(k4i*b14i-k4i*b24i)/Ix4i -(c4i*b14i-
c4i*b24i)/Ix4i -(k4i*b14i^2+k4i*b24i^2+kR4i)/Ix4i -
(c4i*b14i^2+c4i*b24i^2)/Ix4i
0 0 0 0 0 1 0 0
-(k4i+kt4i)/m14i -c4i/m14i 0 0 k4i/m14i c4i/m14i k4i*b14i/m14i
c4i*b14i/m14i
0 0 0 0 0 0 0 1
0 0 -(k4i+kt4i)/m24i -c4i/m24i k4i/m24i c4i/m24i -k4i*b24i/m24i
-c4i*b24i/m24i];

% matriz de entrada, b

b = [ 0 0; F1/m14i 0; 0 0; 0 F2/m24i; 0 0; 0 0; 0 0; 0 0];

% matriz de salida, c

c = eye(8,8);

% matriz de transmision directa, d

d = 0;

% resolver los valores propios de la matriz del sistema

[xm,omega] = eig(a);

% Definición de un vector de frecuencias para utilizar, radia-
nes/seg.

w = 10e-1:0.01:10e2;

% utilizar la función "ss" para definir el sistema de espacio de
estado

sssys = ss(a,b,c,d);

% cálculo de la magnitud y la fase mediante el comando bode.
```

```
% primer índice 1-8: z1 z1dot z2 z2dot z3 z3dot z4 z4dot
% segundo índice 1-2: F1 F2
% tercer índice 1-200: extrae todos los puntos de la frecuencia
utilizando ":"

[mag,phs] = bode(sssys,w);

z1mag = mag(1,1,:);
z1phs = phs(1,1,:);

p1mag = mag(3,1,:);
p1phs = phs(3,1,:);

z11mag = mag(5,1,:);
z11phs = phs(5,1,:);

z21mag = mag(7,1,:);
z21phs = phs(7,1,:);

z2mag = mag(1,2,:);
z2phs = phs(1,2,:);

p2mag = mag(3,2,:);
p2phs = phs(3,2,:);

z12mag = mag(5,2,:);
z12phs = phs(5,2,:);

z22mag = mag(7,2,:);
z22phs = phs(7,2,:);

% calculate the magnitude in decibels, db

z1magdb = 20*log10(z1mag);

p1magdb = 20*log10(p1mag);

z11magdb = 20*log10(z11mag);

z21magdb = 20*log10(z21mag);

z2magdb = 20*log10(z2mag);

p2magdb = 20*log10(p2mag);

z12magdb = 20*log10(z12mag);

z22magdb = 20*log10(z22mag);

% representación de las funciones de transferencia

switch valor4i
    case 1
        subplot(2,1,1)
        semilogx(w,z1magdb(1,:))
```

```
        title('Función de transferencia para 4 GDL modo balan-
ceo, x/F1')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,z1phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('fase, deg')
        grid
    case 2
        subplot(2,1,1)
        semilogx(w,p1magdb(1,:))
        title('Función de transferencia para 4 GDL modo balan-
ceo, \phi/F_1')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,p1phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('fase, deg')
        grid
    case 3
        subplot(2,1,1)
        semilogx(w,z11magdb(1,:))
        title('Función de transferencia para 4 GDL modo balan-
ceo, x_1/F_1')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,z11phs(1,:))
        ylabel('fase, deg')
        xlabel('Frecuencia, rad/s')
        grid
    case 4
        subplot(2,1,1)
        semilogx(w,z21magdb(1,:))
        title('Función de transferencia para 4 GDL modo balan-
ceo, x_2/F_1')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,z21phs(1,:))
        ylabel('fase, deg')
        xlabel('Frecuencia, rad/s')
        grid
    case 5
        subplot(2,1,1)
        semilogx(w,z2magdb(1,:))
        title('Función de transferencia para 4 GDL modo balan-
ceo, x/F_2')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,z2phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('phase, deg')
```

```
        grid
    case 6
        subplot(2,1,1)
        semilogx(w,p2magdb(1,:))
        title('Función de transferencia para 4 GDL modo balan-
ceo, \phi/F_2')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,p2phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('fase, deg')
        grid
    case 7
        subplot(2,1,1)
        semilogx(w,z12magdb(1,:))
        title('Función de transferencia para 4 GDL modo balan-
ceo, x_1/F_2')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,z12phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('fase, deg')
        grid
    case 8
        subplot(2,1,1)
        semilogx(w,z22magdb(1,:))
        title('Función de transferencia para 4 GDL modo balan-
ceo, x_2/F_2')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,z22phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('fase, deg')
        grid
end
```

Ventana de introducción de datos del modelo de cuatro grados de libertad y eje rígido.

```
function varargout = dof4r(varargin)
% DOF4R MATLAB code for dof4r.fig
%      DOF4R, by itself, creates a new DOF4R or raises the ex-
isting
%      singleton*.
%
%      H = DOF4R returns the handle to a new DOF4R or the handle
to
%      the existing singleton*.
%
%      DOF4R('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in DOF4R.M with the given input
arguments.
%
%      DOF4R('Property','Value',...) creates a new DOF4R or
raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before dof4r_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes prop-
erty application
%      stop.  All inputs are passed to dof4r_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI al-
lows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help dof4r

% Last Modified by GUIDE v2.5 28-Aug-2017 20:23:25

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @dof4r_OpeningFcn, ...
                   'gui_OutputFcn',  @dof4r_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT

function dof4r_OpeningFcn(hObject, eventdata, handles, varargin)

% Centrar ventana

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)]);

% Choose default command line output for dof4r
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes dof4r wait for user response (see UIRESUME)
% uiwait(handles.figure1);

function varargout = dof4r_OutputFcn(hObject, eventdata, han-
dles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

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

function figure1_CloseRequestFcn(hObject, eventdata, handles)

opc=questdlg('¿Seguro que desea cerrar esta ventana?','SA-
LIR','Si','No','No');
if strcmp(opc,'No')
    return;
end

% Hint: delete(hObject) closes the figure
delete(hObject);

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

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

```
function kt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to kt (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

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

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

function k_CreateFcn(hObject, eventdata, handles)
% hObject    handle to k (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

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

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

function c_CreateFcn(hObject, eventdata, handles)
% hObject    handle to c (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called
```

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

function m1_Callback(hObject, eventdata, handles)
% hObject    handle to m1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of m1 as text
%        str2double(get(hObject,'String')) returns contents of
m1 as a double

function m1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to m1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function Ix1_Callback(hObject, eventdata, handles)
% hObject    handle to Ix1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Ix1 as text
%        str2double(get(hObject,'String')) returns contents of
Ix1 as a double

function Ix1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Ix1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: edit controls usually have a white background on Win-
dows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
```

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

function m2_Callback(hObject, eventdata, handles)
% hObject    handle to m2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of m2 as text
%        str2double(get(hObject,'String')) returns contents of
m2 as a double

function m2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to m2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function Ix2_Callback(hObject, eventdata, handles)
% hObject    handle to Ix2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Ix2 as text
%        str2double(get(hObject,'String')) returns contents of
Ix2 as a double

function Ix2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Ix2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function a1_Callback(hObject, eventdata, handles)
% hObject    handle to a1 (see GCBO)
```

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

% Hints: get(hObject,'String') returns contents of a1 as text
%        str2double(get(hObject,'String')) returns contents of
a1 as a double

function a1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to a1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function a2_Callback(hObject, eventdata, handles)
% hObject    handle to a2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of a2 as text
%        str2double(get(hObject,'String')) returns contents of
a2 as a double

function a2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to a2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function b1_Callback(hObject, eventdata, handles)
% hObject    handle to b1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of b1 as text
```

```
%        str2double(get(hObject,'String')) returns contents of
b1 as a double

function b1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to b1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function b2_Callback(hObject, eventdata, handles)
% hObject    handle to b2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of b2 as text
%        str2double(get(hObject,'String')) returns contents of
b2 as a double


function b2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to b2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function axes1_CreateFcn(hObject, eventdata, handles)
imshow(imread('4gdl_rigido.PNG'));

function calcular_Callback(hObject, eventdata, handles)

global m14r m24r Ix14r Ix24r c4r k4r kt4r a14r a24r b14r b24r

m14r = str2double(get(handles.m1,'String'));
m24r = str2double(get(handles.m2,'String'));

Ix14r = str2double(get(handles.Ix1,'String'));
Ix24r = str2double(get(handles.Ix2,'String'));
```

```
c4r = str2double(get(handles.c,'String'));

k4r = str2double(get(handles.k,'String'));

kt4r = str2double(get(handles.kt,'String'));

a14r = str2double(get(handles.a1,'String'));
a24r = str2double(get(handles.a2,'String'));
b14r = str2double(get(handles.b1,'String'));
b24r = str2double(get(handles.b2,'String'));

% Matriz de masa

M = [m14r 0 0 0
    0 Ix14r 0 0
    0 0 m24r 0
    0 0 0 Ix24r];

% Matriz de amortiguamiento

c11 = 2*c4r;
c12 = c4r*a14r-c4r*a24r;
c13 = -2*c4r;
c14 = c4r*b24r-c4r*b14r;

c21 = c4r*a14r-c4r*a24r;
c22 = c4r*a14r^2+c4r*a24r^2;
c23 = c4r*a24r-c4r*a14r;
c24 = -c4r*a14r*b14r-c4r*a24r*b24r;

c31 = -2*c4r;
c32 = c4r*a24r-c4r*a14r;
c33 = 2*c4r;
c34 = c4r*b14r-c4r*b24r;

c41 = c4r*b24r-c4r*b14r;
c42 = -c4r*a14r*b14r-c4r*a24r*b24r;
c43 = c4r*b14r-c4r*b24r;
c44 = c4r*b14r^2+c4r*b24r^2;

C = [c11 c12 c13 c14
    c21 c22 c23 c24
    c31 c32 c33 c34
    c41 c42 c43 c44];

% Matriz de rigidez

k11 = 2*kt4r+2*k4r;
k12 = kt4r*a14r-kt4r*a24r+k4r*a14r-k4r*a24r;
k13 = -2*k4r;
k14 = k4r*b24r-k4r*b14r;

k21 = kt4r*a14r-kt4r*a24r+k4r*a14r-k4r*a24r;
k22 = kt4r*a14r^2+kt4r*a24r^2+k4r*a14r^2+k4r*a24r^2;
k23 = k4r*a24r-k4r*a14r;
```

```
k24 = -k4r*a14r*b14r-k4r*a24r*b24r;

k31 = -2*k4r;
k32 = k4r*a24r-k4r*a14r;
k33 = 2*k4r;
k34 = k4r*b14r-k4r*b24r;

k41 = k4r*b24r-k4r*b14r;
k42 = -k4r*a14r*b14r-k4r*a24r*b24r;
k43 = k4r*b14r-k4r*b24r;
k44 = k4r*b14r^2+k4r*b24r^2;

K=[k11 k12 k13 k14
    k21 k22 k23 k24
    k31 k32 k33 k34
    k41 k42 k43 k44];

Mr = sqrtm(M);
Kt = inv(Mr)*K*inv(Mr);
[V,D] = eig(Kt);

W = sqrt(diag(D));% frecuencias naturales en rad/s
wn1 = W(1,1);
wn2 = W(2,1);
wn3 = W(3,1);
wn4 = W(4,1);

Wz = (1/(2*pi))*W; % frecuencias naturales en Hz
wn1z = Wz(1,1);
wn2z = Wz(2,1);
wn3z = Wz(3,1);
wn4z = Wz(4,1);

set(handles.wn1r,'String',wn1);
set(handles.wn2r,'String',wn2);
set(handles.wn3r,'String',wn3);
set(handles.wn4r,'String',wn4);

set(handles.wn1h,'String',wn1z);
set(handles.wn2h,'String',wn2z);
set(handles.wn3h,'String',wn3z);
set(handles.wn4h,'String',wn4z);

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

% Hints: contents = cellstr(get(hObject,'String')) returns FT
contents as cell array
%        contents{get(hObject,'Value')} returns selected item
from FT

function FT_CreateFcn(hObject, eventdata, handles)
% hObject    handle to FT (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function representar_Callback(hObject, eventdata, handles)
global valor4r

valor4r = get(handles.FT,'Value');

dof4r_res;
```

Ventada de gráficos del modelo de cuatro grados de libertad y suspensión de eje rígido.

```
function varargout = dof4r_res(varargin)
% DOF4R_RES MATLAB code for dof4r_res.fig
%      DOF4R_RES, by itself, creates a new DOF4R_RES or raises
the existing
%      singleton*.
%
%      H = DOF4R_RES returns the handle to a new DOF4R_RES or
the handle to
%      the existing singleton*.
%
%      DOF4R_RES('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in DOF4R_RES.M with the given in-
put arguments.
%
%      DOF4R_RES('Property','Value',...) creates a new DOF4R_RES
or raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before dof4r_res_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes prop-
erty application
%      stop.  All inputs are passed to dof4r_res_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI al-
lows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help dof4r_res

% Last Modified by GUIDE v2.5 28-Aug-2017 20:46:29

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @dof4r_res_OpeningFcn, ...
                   'gui_OutputFcn',  @dof4r_res_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT

function dof4r_res_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to dof4r_res (see VARARGIN)

% Centrar ventana

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)]);

% Choose default command line output for dof4r_res
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes dof4r_res wait for user response (see UIRESUME)
% uiwait(handles.figure1);

function varargout = dof4r_res_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

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

function figure1_CloseRequestFcn(hObject, eventdata, handles)
opc=questdlg('¿Seguro que desea cerrar esta ventana?','SA-
LIR','Si','No','No');
if strcmp(opc,'No')
    return;
end

% Hint: delete(hObject) closes the figure
delete(hObject);

function axes1_CreateFcn(hObject, eventdata, handles)

global m14r m24r Ix14r Ix24r c4r k4r kt4r a14r a24r b14r b24r
valor4r
```

```
F1 = 1;
F2 = 1;

% matriz del sistema, a

a = [ 0 1 0 0 0 0 0 0
(-2*k4r-2*kt4r)/m14r -2*c4r/m14r (-k4r*a14r+k4r*a24r-
kt4r*a14r+kt4r*a24r)/m14r (c4r*a24r-c4r*a14r)/m14r 2*k4r/m14r
2*c4r/m14r (k4r*b14r-k4r*b24r)/m14r (c4r*b14r-c4r*b24r)/m14r
0 0 0 1 0 0 0 0
(-k4r*a14r+k4r*a24r-kt4r*a14r+kt4r*a24r)/Ix14r (c4r*a24r-
c4r*a14r)/Ix14r (-k4r*a14r^2-k4r*a24r^2-kt4r*a14r^2-
kt4r*a24r^2)/Ix14r (-c4r*a14r^2-c4r*a24r^2)/Ix14r (k4r*a14r-
k4r*a24r)/Ix14r (c4r*a14r-c4r*a24r)/Ix14r
(k4r*a14r*b14r+k4r*a24r*b24r)/Ix14r
(c4r*a14r*b14r+c4r*a24r*b24r)/Ix14r
0 0 0 0 0 1 0 0
2*k4r/m24r 2*c4r/m24r (k4r*a14r-k4r*a24r)/m24r (c4r*a14r-
c4r*a24r)/m24r -2*k4r/m24r -2*c4r/m24r (k4r*b24r-k4r*b14r)/m24r
(c4r*b24r-c4r*b14r)/m24r
0 0 0 0 0 0 0 1
(k4r*b14r-k4r*b24r)/Ix24r (c4r*b14r-c4r*b24r)/Ix24r
(k4r*a14r*b14r+k4r*a24r*b24r)/Ix24r
(c4r*a14r*b14r+c4r*a24r*b24r)/Ix24r (k4r*b24r-k4r*b14r)/Ix24r
(c4r*b24r-c4r*b14r)/Ix24r (-k4r*b14r^2-k4r*b24r^2)/Ix24r (-
c4r*b14r^2-c4r*b24r^2)/Ix24r];

% matriz de entrada, b

b = [ 0 0; F1/m14r F2/m14r; 0 0; -F1*a14r/Ix14r F2*a24r/Ix14r; 0
0; 0 0; 0 0; 0 0];

% matriz de salida, c

c = eye(8,8);

% matriz de transmision directa, d

d = 0;

% resolver los valores propios de la matriz del sistema

[xm,omega] = eig(a);

% Definición de un vector de frecuencias para utilizar, radia-
nes/seg.

w = 0:0.01:10e2;

% utilizar la función "ss" para definir el sistema de espacio de
estado

sssys = ss(a,b,c,d);

% cálculo de la magnitud y la fase mediante el comando bode.
```

```
% primer índice 1-8: z1 z1dot z2 z2dot z3 z3dot z4 z4dot
% segundo índice 1-2: F1 F2
% tercer índice 1-200: extrae todos los puntos de la frecuencia
utilizando ":"

[mag,phs] = bode(sssys,w);

z11mag = mag(1,1,:);
z11phs = phs(1,1,:);

p11mag = mag(3,1,:);
p11phs = phs(3,1,:);

z21mag = mag(5,1,:);
z21phs = phs(5,1,:);

p21mag = mag(7,1,:);
p21phs = phs(7,1,:);

z12mag = mag(1,2,:);
z12phs = phs(1,2,:);

p12mag = mag(3,2,:);
p12phs = phs(3,2,:);

z22mag = mag(5,2,:);
z22phs = phs(5,2,:);

p22mag = mag(7,2,:);
p22phs = phs(7,2,:);

% calculo de la magnitud en decibelios, db

z11magdb = 20*log10(z11mag);

p11magdb = 20*log10(p11mag);

z21magdb = 20*log10(z21mag);

p21magdb = 20*log10(p21mag);

z12magdb = 20*log10(z12mag);

p12magdb = 20*log10(p12mag);

z22magdb = 20*log10(z22mag);

p22magdb = 20*log10(p22mag);

% representación de las funciones de transferencia

switch valor4r
    case 1
        subplot(2,1,1)
        semilogx(w,z11magdb(1,:))
```

```
        title('Función de transferencia para 4 GDL eje rígido,
x_1/F1')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,z11phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('fase, deg')
        grid
    case 2
        subplot(2,1,1)
        semilogx(w,p11magdb(1,:))
        title('Función de transferencia para 4 GDL eje rígido,
\phi_1/F_1')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,p11phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('fase, deg')
        grid
    case 3
        subplot(2,1,1)
        semilogx(w,z21magdb(1,:))
        title('Función de transferencia para 4 GDL eje rígido,
x_2/F_1')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,z21phs(1,:))
        ylabel('fase, deg')
        xlabel('Frecuencia, rad/s')
        grid
    case 4
        subplot(2,1,1)
        semilogx(w,p21magdb(1,:))
        title('Función de transferencia para 4 GDL GDL eje rí-
gido, \phi_2/F_1')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,p21phs(1,:))
        ylabel('fase, deg')
        xlabel('Frecuencia, rad/s')
        grid
    case 5
        subplot(2,1,1)
        semilogx(w,z12magdb(1,:))
        title('Función de transferencia para 4 GDL GDL eje rí-
gido, x_1/F_2')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,z12phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('phase, deg')
```

```
        grid
    case 6
        subplot(2,1,1)
        semilogx(w,p12magdb(1,:))
        title('Función de transferencia para 4 GDL GDL eje rí-
gido, \phi_1/F_2')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,p12phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('fase, deg')
        grid
    case 7
        subplot(2,1,1)
        semilogx(w,z22magdb(1,:))
        title('Función de transferencia para 4 GDL GDL eje rí-
gido, x_2/F_2')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,z22phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('fase, deg')
        grid
    case 8
        subplot(2,1,1)
        semilogx(w,p22magdb(1,:))
        title('Función de transferencia para 4 GDL GDL eje rí-
gido, \phi_2/F_2')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,p22phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('fase, deg')
        grid
end
```

Ventana de introducción de datos del modelo de siete grados de libertad.

```
function varargout = dof7i(varargin)
% DOF7I MATLAB code for dof7i.fig
%      DOF7I, by itself, creates a new DOF7I or raises the ex-
isting
%      singleton*.
%
%      H = DOF7I returns the handle to a new DOF7I or the handle
to
%      the existing singleton*.
%
%      DOF7I('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in DOF7I.M with the given input
arguments.
%
%      DOF7I('Property','Value',...) creates a new DOF7I or
raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before dof7i_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes prop-
erty application
%      stop.  All inputs are passed to dof7i_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI al-
lows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help dof7i

% Last Modified by GUIDE v2.5 29-Aug-2017 15:29:47

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @dof7i_OpeningFcn, ...
                   'gui_OutputFcn',  @dof7i_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT

function dof7i_OpeningFcn(hObject, eventdata, handles, varargin)

% Centrar ventana

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)]);

% Choose default command line output for dof7i
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes dof7i wait for user response (see UIRESUME)
% uiwait(handles.figure1);

function varargout = dof7i_OutputFcn(hObject, eventdata, han-
dles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

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

function a1_Callback(hObject, eventdata, handles)
% hObject    handle to a2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of a2 as text
%        str2double(get(hObject,'String')) returns contents of
a2 as a double

function a1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to a2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: edit controls usually have a white background on Win-
dows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function a2_Callback(hObject, eventdata, handles)
% hObject    handle to a1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of a1 as text
%        str2double(get(hObject,'String')) returns contents of
a1 as a double

function a2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to a1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function b1_Callback(hObject, eventdata, handles)
% hObject    handle to a1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of a1 as text
%        str2double(get(hObject,'String')) returns contents of
a1 as a double

function b1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to a1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function b2_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to a2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of a2 as text
%        str2double(get(hObject,'String')) returns contents of
a2 as a double

function b2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to a2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function ktr_Callback(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit13 as
text
%        str2double(get(hObject,'String')) returns contents of
edit13 as a double

function ktr_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function cr_Callback(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit19 as
text
%        str2double(get(hObject,'String')) returns contents of
edit19 as a double

function cr_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

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

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

function kf_CreateFcn(hObject, eventdata, handles)
% hObject    handle to kr (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

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

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

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

```
% hObject    handle to cr (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function ktf_Callback(hObject, eventdata, handles)
% hObject    handle to edit18 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit18 as
text
%        str2double(get(hObject,'String')) returns contents of
edit18 as a double

function ktf_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit18 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

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

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

function mf_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ktf (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called
```

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

function mr_Callback(hObject, eventdata, handles)
% hObject    handle to m2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of m2 as text
%        str2double(get(hObject,'String')) returns contents of
m2 as a double

function mr_CreateFcn(hObject, eventdata, handles)
% hObject    handle to m2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function m_Callback(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit13 as
text
%        str2double(get(hObject,'String')) returns contents of
edit13 as a double

function m_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: edit controls usually have a white background on Win-
dows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'de-
faultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function Ix_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of
edit9 as a double

function Ix_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

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

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

function Iy_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ktr (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

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

```
% hObject    handle to m2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of m2 as text
%        str2double(get(hObject,'String')) returns contents of
m2 as a double

function cf_CreateFcn(hObject, eventdata, handles)
% hObject    handle to m2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function axes1_CreateFcn(hObject, eventdata, handles)
imshow(imread('7gdl.PNG'));

function figure1_CloseRequestFcn(hObject, eventdata, handles)
opc=questdlg('¿Seguro que desea cerrar esta ventana?','SA-
LIR','Si','No','No');
if strcmp(opc,'No')
    return;
end

% Hint: delete(hObject) closes the figure
delete(hObject);

function calcular_Callback(hObject, eventdata, handles)

global m7 mf7 mr7 Ix7 Iy7 cf7 cr7 kf7 kr7 ktf7 ktr7 kR7 a17 a27
b17 b27

m7 = str2double(get(handles.m,'String'));
mf7 = str2double(get(handles.mf,'String'));
mr7 = str2double(get(handles.mr,'String'));

Ix7 = str2double(get(handles.Ix,'String'));
Iy7 = str2double(get(handles.Iy,'String'));

cf7 = str2double(get(handles.cf,'String'));
cr7 = str2double(get(handles.cr,'String'));

kf7 = str2double(get(handles.kf,'String'));
kr7 = str2double(get(handles.kr,'String'));
kR7 = str2double(get(handles.kR,'String'));
ktf7 = str2double(get(handles.ktf,'String'));
```

```
ktr7 = str2double(get(handles.ktr,'String'));

a17 = str2double(get(handles.a1,'String'));
a27 = str2double(get(handles.a2,'String'));
b17 = str2double(get(handles.b1,'String'));
b27 = str2double(get(handles.b2,'String'));

% Matriz de masa [M]

M=[m7 0 0 0 0 0 0
    0 Ix7 0 0 0 0 0
    0 0 Iy7 0 0 0 0
    0 0 0 mf7 0 0 0
    0 0 0 0 mf7 0 0
    0 0 0 0 0 mr7 0
    0 0 0 0 0 0 mr7];

% Matriz de amortiguamiento [C]

c11 = 2*cf7+2*cr7;
c12 = b17*cf7-b27*cf7-b27*cr7+b17*cr7;
c13 = 2*a27*cr7-2*a17*cf7;
c14 = -cf7;
c15 = -cf7;
c16 = -cr7;
c17 = -cr7;

c21 = c12;
c22 = b17^2*cf7+b27^2*cf7+b17^2*cr7+b27^2*cr7;
c23 = a17*b27*cf7-a17*b17*cf7-a27*b27*cr7+a27*b17*cr7;
c24 = -b17*cf7;
c25 = b27*cf7;
c26 = b27*cr7;
c27 = -b17*cr7;

c31 = c13;
c32 = c23;
c33 = 2*cf7*a17^2+2*cr7*a27^2;
c34 = a17*cf7;
c35 = a17*cf7;
c36 = -a27*cr7;
c37 = -a27*cr7;

c41 = -cf7;
c42 = -b17*cf7;
c43 = a17*cf7;
c44 = cf7;
c45 = 0;
c46 = 0;
c47 = 0;

c51 = -cf7;
c52 = b27*cf7;
c53 = a17*cf7;
c54 = 0;
c55 = cf7;
```

```
c56 = 0;
c57 = 0;

c61 = -cr7;
c62 = b27*cr7;
c63 = -a27*cr7;
c64 = 0;
c65 = 0;
c66 = cr7;
c67 = 0;

c71 = -cr7;
c72 = -b17*cr7;
c73 = -a27*cr7;
c74 = 0;
c75 = 0;
c76 = 0;
c77 = cr7;

C=[c11 c12 c13 c14 c15 c16 c17
    c21 c22 c23 c24 c25 c26 c27
    c31 c32 c33 c34 c35 c36 c37
    c41 c42 c43 c44 c45 c46 c47
    c51 c52 c53 c54 c55 c56 c57
    c61 c62 c63 c64 c65 c66 c67
    c71 c72 c73 c74 c75 c76 c77];

% Matriz de rigidez [K]

h = b17+b27;

k11 = 2*kf7+2*kr7;
k12 = b17*kf7-b27*kf7-b27*kr7+b17*kr7;
k13 = 2*a27*kr7-2*a17*kf7;
k14 = -kf7;
k15 = -kf7;
k16 = -kr7;
k17 = -kr7;

k21 = k12;
k22 = kR7+b17^2*kf7+b27^2*kf7+b17^2*kr7+b27^2*kr7;
k23 = a17*b27*kf7-a17*b17*kf7-a27*b27*kr7+a27*b17*kr7;
k24 = -b17*kf7-(kR7/h);
k25 = b27*kf7+(kR7/h);
k26 = b17*kr7;
k27 = -b27*kr7;

k31 = k13;
k32 = k23;
k33 = 2*kf7*a17^2+2*kr7*a27^2;
k34 = a17*kf7;
k35 = a17*kf7;
k36 = -a27*kr7;
k37 = -a27*kr7;

k41 = -kf7;
```

```
k42 = k24;
k43 = a17*kf7;
k44 = kf7+ktf7+(kR7/h^2);
k45 = -kR7/h^2;
k46 = 0;
k47 = 0;

k51 = -kf7;
k52 = k25;
k53 = a17*kf7;
k54 = -kR7/h^2;
k55 = kf7+ktf7+(kR7/h^2);
k56 = 0;
k57 = 0;

k61 = -kr7;
k62 = b27*kr7;
k63 = -a27*kr7;
k64 = 0;
k65 = 0;
k66 = kr7+ktr7;
k67 = 0;

k71 = -kr7;
k72 = -b17*kr7;
k73 = -a27*kr7;
k74 = 0;
k75 = 0;
k76 = 0;
k77 = kr7+ktr7;

K=[k11 k12 k13 k14 k15 k16 k17
    k21 k22 k23 k24 k25 k26 k27
    k31 k32 k33 k34 k35 k36 k37
    k41 k42 k43 k44 k45 k46 k47
    k51 k52 k53 k54 k55 k56 k57
    k61 k62 k63 k64 k65 k66 k67
    k71 k72 k73 k74 k75 k76 k77];

% Frecuencias naturales

Mr = sqrtm(M);
Kt = inv(Mr)*K*inv(Mr);
[V,D] = eig(Kt);

W = sqrt(diag(D)); % frecuencias naturales en rad/s
wn1 = W(1);
wn2 = W(2);
wn3 = W(3);
wn4 = W(4);
wn5 = W(5);
wn6 = W(6);
wn7 = W(7);

Wz = (1/(2*pi))*W; % frecuencias naturales en Hz
wn1z = Wz(1);
```

```
wn2z = Wz(2);
wn3z = Wz(3);
wn4z = Wz(4);
wn5z = Wz(5);
wn6z = Wz(6);
wn7z = Wz(7);

set(handles.wn1r,'String',wn1);
set(handles.wn2r,'String',wn2);
set(handles.wn3r,'String',wn3);
set(handles.wn4r,'String',wn4);
set(handles.wn5r,'String',wn5);
set(handles.wn6r,'String',wn6);
set(handles.wn7r,'String',wn7);

set(handles.wn1h,'String',wn1z);
set(handles.wn2h,'String',wn2z);
set(handles.wn3h,'String',wn3z);
set(handles.wn4h,'String',wn4z);
set(handles.wn5h,'String',wn5z);
set(handles.wn6h,'String',wn6z);
set(handles.wn7h,'String',wn7z);

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

% Hints: contents = cellstr(get(hObject,'String')) returns FT
contents as cell array
%        contents{get(hObject,'Value')} returns selected item
from FT

function FT_CreateFcn(hObject, eventdata, handles)
% hObject    handle to FT (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

function representar_Callback(hObject, eventdata, handles)
global valor7
valor7 = get(handles.FT,'Value');
dof7i_res;

function kR_Callback(hObject, eventdata, handles)
% hObject    handle to kR (see GCBO)
```

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

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

function kR_CreateFcn(hObject, eventdata, handles)
% hObject    handle to kR (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

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

Ventada de gráficos del modelo de siete grados de libertad.

```
function varargout = dof7i_res(varargin)
% DOF7I_RES MATLAB code for dof7i_res.fig
%      DOF7I_RES, by itself, creates a new DOF7I_RES or raises
the existing
%      singleton*.
%
%      H = DOF7I_RES returns the handle to a new DOF7I_RES or
the handle to
%      the existing singleton*.
%
%      DOF7I_RES('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in DOF7I_RES.M with the given in-
put arguments.
%
%      DOF7I_RES('Property','Value',...) creates a new DOF7I_RES
or raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before dof7i_res_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes prop-
erty application
%      stop.  All inputs are passed to dof7i_res_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI al-
lows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help dof7i_res

% Last Modified by GUIDE v2.5 29-Aug-2017 12:18:11

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @dof7i_res_OpeningFcn, ...
                   'gui_OutputFcn',  @dof7i_res_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT


% --- Executes just before dof7i_res is made visible.
function dof7i_res_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to dof7i_res (see VARARGIN)

% Centrar ventana

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)]);

% Choose default command line output for dof7i_res
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes dof7i_res wait for user response (see UIRESUME)
% uiwait(handles.figure1);

function varargout = dof7i_res_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

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

function figure1_CloseRequestFcn(hObject, eventdata, handles)
opc=questdlg('¿Seguro que desea cerrar esta ventana?','SA-
LIR','Si','No','No');
if strcmp(opc,'No')
    return;
end

% Hint: delete(hObject) closes the figure
delete(hObject);

function axes1_CreateFcn(hObject, eventdata, handles)
```

```
global m7 mf7 mr7 Ix7 Iy7 cf7 cr7 kf7 kr7 ktf7 ktr7 kR7 a17 a27
b17 b27 valor7

F1 = 1;
F2 = 1;
F3 = 1;
F4 = 1;

% Matriz de masa [M]

m11 = m7;
m22 = Ix7; %phi
m33 = Iy7; %theta
m44 = mf7;
m55 = mf7;
m66 = mr7;
m77 = mr7;

M=[m11 0 0 0 0 0 0
    0 m22 0 0 0 0 0
    0 0 m33 0 0 0 0
    0 0 0 m44 0 0 0
    0 0 0 0 m55 0 0
    0 0 0 0 0 m66 0
    0 0 0 0 0 0 m77];

% Matriz de amortiguamiento [C]

c11 = 2*cf7+2*cr7;
c12 = b17*cf7-b27*cf7-b27*cr7+b17*cr7;
c13 = 2*a27*cr7-2*a17*cf7;
c14 = -cf7;
c15 = -cf7;
c16 = -cr7;
c17 = -cr7;

c21 = c12;
c22 = b17^2*cf7+b27^2*cf7+b17^2*cr7+b27^2*cr7;
c23 = a17*b27*cf7-a17*b17*cf7-a27*b27*cr7+a27*b17*cr7;
c24 = -b17*cf7;
c25 = b27*cf7;
c26 = b27*cr7;
c27 = -b17*cr7;

c31 = c13;
c32 = c23;
c33 = 2*cf7*a17^2+2*cr7*a27^2;
c34 = a17*cf7;
c35 = a17*cf7;
c36 = -a27*cr7;
c37 = -a27*cr7;

c41 = -cf7;
c42 = -b17*cf7;
c43 = a17*cf7;
c44 = cf7;
```

```
c45 = 0;
c46 = 0;
c47 = 0;

c51 = -cf7;
c52 = b27*cf7;
c53 = a17*cf7;
c54 = 0;
c55 = cf7;
c56 = 0;
c57 = 0;

c61 = -cr7;
c62 = b27*cr7;
c63 = -a27*cr7;
c64 = 0;
c65 = 0;
c66 = cr7;
c67 = 0;

c71 = -cr7;
c72 = -b17*cr7;
c73 = -a27*cr7;
c74 = 0;
c75 = 0;
c76 = 0;
c77 = cr7;

C=[c11 c12 c13 c14 c15 c16 c17
    c21 c22 c23 c24 c25 c26 c27
    c31 c32 c33 c34 c35 c36 c37
    c41 c42 c43 c44 c45 c46 c47
    c51 c52 c53 c54 c55 c56 c57
    c61 c62 c63 c64 c65 c66 c67
    c71 c72 c73 c74 c75 c76 c77];

% Matriz de rigidez [K]

h = b17+b27;

k11 = 2*kf7+2*kr7;
k12 = b17*kf7-b27*kf7-b27*kr7+b17*kr7;
k13 = 2*a27*kr7-2*a17*kf7;
k14 = -kf7;
k15 = -kf7;
k16 = -kr7;
k17 = -kr7;

k21 = k12;
k22 = kR7+b17^2*kf7+b27^2*kf7+b17^2*kr7+b27^2*kr7;
k23 = a17*b27*kf7-a17*b17*kf7-a27*b27*kr7+a27*b17*kr7;
k24 = -b17*kf7-(kR7/h);
k25 = b27*kf7+(kR7/h);
k26 = b17*kr7;
k27 = -b27*kr7;
```

```
k31 = k13;
k32 = k23;
k33 = 2*kf7*a17^2+2*kr7*a27^2;
k34 = a17*kf7;
k35 = a17*kf7;
k36 = -a27*kr7;
k37 = -a27*kr7;

k41 = -kf7;
k42 = k24;
k43 = a17*kf7;
k44 = kf7+ktf7+(kR7/h^2);
k45 = -kR7/h^2;
k46 = 0;
k47 = 0;

k51 = -kf7;
k52 = k25;
k53 = a17*kf7;
k54 = -kR7/h^2;
k55 = kf7+ktf7+(kR7/h^2);
k56 = 0;
k57 = 0;

k61 = -kr7;
k62 = b27*kr7;
k63 = -a27*kr7;
k64 = 0;
k65 = 0;
k66 = kr7+ktr7;
k67 = 0;

k71 = -kr7;
k72 = -b17*kr7;
k73 = -a27*kr7;
k74 = 0;
k75 = 0;
k76 = 0;
k77 = kr7+ktr7;

K=[k11 k12 k13 k14 k15 k16 k17
    k21 k22 k23 k24 k25 k26 k27
    k31 k32 k33 k34 k35 k36 k37
    k41 k42 k43 k44 k45 k46 k47
    k51 k52 k53 k54 k55 k56 k57
    k61 k62 k63 k64 k65 k66 k67
    k71 k72 k73 k74 k75 k76 k77];

% matriz del sistema, a

a = [ 0 1 0 0 0 0 0 0 0 0 0 0 0 0
    -k11/m7 -c11/m7 -k12/m7 -c12/m7 -k13/m7 -c13/m7 -k14/m7 -
c14/m7 -k15/m7 -c15/m7 -k16/m7 -c16/m7 -k17/m7 -c17/m7
    0 0 0 1 0 0 0 0 0 0 0 0 0 0
```

```
    -k21/Ix7 -c21/Ix7 -k22/Ix7 -c22/Ix7 -k23/Ix7 -c23/Ix7 -
k24/Ix7 -c24/Ix7 -k25/Ix7 -c25/Ix7 -k26/Ix7 -c26/Ix7 -k27/Ix7 -
c27/Ix7
    0 0 0 0 0 1 0 0 0 0 0 0 0 0
    -k31/Iy7 -c31/Iy7 -k32/Iy7 -c32/Iy7 -k33/Iy7 -c33/Iy7 -
k34/Iy7 -c34/Iy7 -k35/Iy7 -c35/Iy7 -k36/Iy7 -c36/Iy7 -k37/Iy7 -
c37/Iy7
    0 0 0 0 0 0 0 1 0 0 0 0 0 0
    -k41/mf7 -c41/mf7 -k42/mf7 -c42/mf7 -k43/mf7 -c43/mf7 -
k44/mf7 -c44/mf7 -k45/mf7 -c45/mf7 -k46/mf7 -c46/mf7 -k47/mf7 -
c47/mf7
    0 0 0 0 0 0 0 0 0 1 0 0 0 0
    -k51/mf7 -c51/mf7 -k52/mf7 -c52/mf7 -k53/mf7 -c53/mf7 -
k54/mf7 -c54/mf7 -k55/mf7 -c55/mf7 -k56/mf7 -c56/mf7 -k57/mf7 -
c57/mf7
    0 0 0 0 0 0 0 0 0 0 0 1 0 0
    -k61/mr7 -c61/mr7 -k62/mr7 -c62/mr7 -k63/mr7 -c63/mr7 -
k64/mr7 -c64/mr7 -k65/mr7 -c65/mr7 -k66/mr7 -c66/mr7 -k67/mr7 -
c67/mr7
    0 0 0 0 0 0 0 0 0 0 0 0 0 1
    -k71/mr7 -c71/mr7 -k72/mr7 -c72/mr7 -k73/mr7 -c73/mr7 -
k74/mr7 -c74/mr7 -k75/mr7 -c75/mr7 -k76/mr7 -c76/mr7 -k77/mr7 -
c77/mr7];

% matriz de entrada, b

b = [0 0 0 0
    0 0 0 0
    0 0 0 0
    0 0 0 0
    0 0 0 0
    0 0 0 0
    0 0 0 0
    F1/mf7 0 0 0
    0 0 0 0
    0 F2/mf7 0 0
    0 0 0 0
    0 0 F3/mr7 0
    0 0 0 0
    0 0 0 F4/mr7];

% matriz de salida, c

c = eye(14,14);

% matriz de transmision directa, d

d = 0;

% valores y vectores propios del sistema

[xm,omega] = eig(a);

% Definición de un vector de frecuencias para utilizar, radia-
nes/seg.
```

```
w = 0:0.01:10e2;

% utilizar la función "ss" para definir el sistema de espacio de
estado

sssys = ss(a,b,c,d);

% cálculo de la magnitud y la fase mediante el comando bode.
% primer índice 1-14: z zdot phi phidot theta thetadot z1 z1dot
z2 z2dot z3 z3dot z4 z4dot
% segundo índice 1-4: F1 F2 F3 F4
% tercer índice 1-200: extrae todos los puntos de la frecuencia
utilizando ":"

[mag,phs] = bode(sssys,w);

z51mag = mag(1,1,:);
z51phs = phs(1,1,:);

z52mag = mag(1,2,:);
z52phs = phs(1,2,:);

z53mag = mag(1,3,:);
z53phs = phs(1,3,:);

z54mag = mag(1,4,:);
z54phs = phs(1,4,:);

p51mag = mag(3,1,:);
p51phs = phs(3,1,:);

p52mag = mag(3,2,:);
p52phs = phs(3,2,:);

p53mag = mag(3,3,:);
p53phs = phs(3,3,:);

p54mag = mag(3,4,:);
p54phs = phs(3,4,:);

t51mag = mag(5,1,:);
t51phs = phs(5,1,:);

t52mag = mag(5,2,:);
t52phs = phs(5,2,:);

t53mag = mag(5,3,:);
t53phs = phs(5,3,:);

t54mag = mag(5,4,:);
t54phs = phs(5,4,:);

z11mag = mag(7,1,:);
z11phs = phs(7,1,:);

z12mag = mag(7,2,:);
```

```
z12phs = phs(7,2,:);

z13mag = mag(7,3,:);
z13phs = phs(7,3,:);

z14mag = mag(7,4,:);
z14phs = phs(7,4,:);

z21mag = mag(9,1,:);
z21phs = phs(9,1,:);

z22mag = mag(9,2,:);
z22phs = phs(9,2,:);

z23mag = mag(9,3,:);
z23phs = phs(9,3,:);

z24mag = mag(9,4,:);
z24phs = phs(9,4,:);

z31mag = mag(11,1,:);
z31phs = phs(11,1,:);

z32mag = mag(11,2,:);
z32phs = phs(11,2,:);

z33mag = mag(11,3,:);
z33phs = phs(11,3,:);

z34mag = mag(11,4,:);
z34phs = phs(11,4,:);

z41mag = mag(13,1,:);
z41phs = phs(13,1,:);

z42mag = mag(13,2,:);
z42phs = phs(13,2,:);

z43mag = mag(13,3,:);
z43phs = phs(13,3,:);

z44mag = mag(13,4,:);
z44phs = phs(13,4,:);

% calculo de la magnitud en decibelios, db

z11magdb = 20*log10(z11mag);

z12magdb = 20*log10(z12mag);

z13magdb = 20*log10(z13mag);

z14magdb = 20*log10(z14mag);

z21magdb = 20*log10(z21mag);
```

```
z22magdb = 20*log10(z22mag);

z23magdb = 20*log10(z23mag);

z24magdb = 20*log10(z24mag);

z31magdb = 20*log10(z31mag);

z32magdb = 20*log10(z32mag);

z33magdb = 20*log10(z33mag);

z34magdb = 20*log10(z34mag);

z41magdb = 20*log10(z41mag);

z42magdb = 20*log10(z42mag);

z43magdb = 20*log10(z43mag);

z44magdb = 20*log10(z44mag);

z51magdb = 20*log10(z51mag);

z52magdb = 20*log10(z52mag);

z53magdb = 20*log10(z53mag);

z54magdb = 20*log10(z54mag);

p51magdb = 20*log10(p51mag);

p52magdb = 20*log10(p52mag);

p53magdb = 20*log10(p53mag);

p54magdb = 20*log10(p54mag);

t51magdb = 20*log10(t51mag);

t52magdb = 20*log10(t52mag);

t53magdb = 20*log10(t53mag);

t54magdb = 20*log10(t54mag);

% representación de las funciones de transferencia

switch valor7
    case 1
        subplot(2,1,1)
        semilogx(w,z51magdb(1,:))
        title('Respuesta en frecuencia para 7 GDL, x/F1')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
```

```
        semilogx(w,z51phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('fase, deg')
        grid
    case 2
        subplot(2,1,1)
        semilogx(w,z52magdb(1,:))
        title('Respuesta en frecuencia para 7 GDL, x/F2')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,z52phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('fase, deg')
        grid
    case 3
        subplot(2,1,1)
        semilogx(w,z53magdb(1,:))
        title('Respuesta en frecuencia para 7 GDL, x/F3')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,z53phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('fase, deg')
        grid
    case 4
        subplot(2,1,1)
        semilogx(w,z54magdb(1,:))
        title('Respuesta en frecuencia para 7 GDL, x/F4')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,z54phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('fase, deg')
        grid
    case 5
        subplot(2,1,1)
        semilogx(w,p51magdb(1,:))
        title('Respuesta en frecuencia para 7 GDL, \phi/F1')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,p51phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('fase, deg')
        grid
    case 6
        subplot(2,1,1)
        semilogx(w,p52magdb(1,:))
        title('Respuesta en frecuencia para 7 GDL, \phi/F2')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,p52phs(1,:))
```

```
    xlabel('Frecuencia, rad/s')
    ylabel('fase, deg')
    grid
case 7
    subplot(2,1,1)
    semilogx(w,p53magdb(1,:))
    title('Respuesta en frecuencia para 7 GDL, \phi/F3')
    ylabel('magnitud, db')
    grid
    subplot(2,1,2)
    semilogx(w,p53phs(1,:))
    xlabel('Frecuencia, rad/s')
    ylabel('fase, deg')
    grid
case 8
    subplot(2,1,1)
    semilogx(w,p54magdb(1,:))
    title('Respuesta en frecuencia para 7 GDL, \phi/F4')
    ylabel('magnitud, db')
    grid
    subplot(2,1,2)
    semilogx(w,p54phs(1,:))
    xlabel('Frecuencia, rad/s')
    ylabel('fase, deg')
    gridend
case 9
    subplot(2,1,1)
    semilogx(w,t51magdb(1,:))
    title('Respuesta en frecuencia para 7 GDL, \theta/F1')
    ylabel('magnitud, db')
    grid
    subplot(2,1,2)
    semilogx(w,t51phs(1,:))
    xlabel('Frecuencia, rad/s')
    ylabel('fase, deg')
    grid
case 10
    subplot(2,1,1)
    semilogx(w,t52magdb(1,:))
    title('Respuesta en frecuencia para 7 GDL, \tehta/F2')
    ylabel('magnitud, db')
    grid
    subplot(2,1,2)
    semilogx(w,t52phs(1,:))
    xlabel('Frecuencia, rad/s')
    ylabel('fase, deg')
    grid
case 11
    subplot(2,1,1)
    semilogx(w,t53magdb(1,:))
    title('Respuesta en frecuencia para 7 GDL, \theta/F3')
    ylabel('magnitud, db')
    grid
    subplot(2,1,2)
    semilogx(w,t53phs(1,:))
    xlabel('Frecuencia, rad/s')
```

```
    ylabel('fase, deg')
    grid
case 12
    subplot(2,1,1)
    semilogx(w,t54magdb(1,:))
    title('Respuesta en frecuencia para 7 GDL, \theta/F4')
    ylabel('magnitud, db')
    grid
    subplot(2,1,2)
    semilogx(w,t54phs(1,:))
    xlabel('Frecuencia, rad/s')
    ylabel('fase, deg')
    grid
case 13
    subplot(2,1,1)
    semilogx(w,z11magdb(1,:))
    title('Respuesta en frecuencia para 7 GDL, x1/F1')
    ylabel('magnitud, db')
    grid
    subplot(2,1,2)
    semilogx(w,z11phs(1,:))
    xlabel('Frecuencia, rad/s')
    ylabel('fase, deg')
    grid
case 14
    subplot(2,1,1)
    semilogx(w,z12magdb(1,:))
    title('Respuesta en frecuencia para 7 GDL, x1/F2')
    ylabel('magnitud, db')
    grid
    subplot(2,1,2)
    semilogx(w,z12phs(1,:))
    xlabel('Frecuencia, rad/s')
    ylabel('fase, deg')
    grid
case 15
    subplot(2,1,1)
    semilogx(w,z13magdb(1,:))
    title('Respuesta en frecuencia para 7 GDL, x1/F3')
    ylabel('magnitud, db')
    grid
    subplot(2,1,2)
    semilogx(w,z13phs(1,:))
    xlabel('Frecuencia, rad/s')
    ylabel('fase, deg')
    grid
case 16
    subplot(2,1,1)
    semilogx(w,z14magdb(1,:))
    title('Respuesta en frecuencia para 7 GDL, x1/F4')
    ylabel('magnitud, db')
    grid
    subplot(2,1,2)
    semilogx(w,z14phs(1,:))
    xlabel('Frecuencia, rad/s')
    ylabel('fase, deg')
```

```
    grid
case 17
    subplot(2,1,1)
    semilogx(w,z21magdb(1,:))
    title('Respuesta en frecuencia para 7 GDL, x2/F1')
    ylabel('magnitud, db')
    grid
    subplot(2,1,2)
    semilogx(w,z21phs(1,:))
    xlabel('Frecuencia, rad/s')
    ylabel('fase, deg')
    grid
case 18
    subplot(2,1,1)
    semilogx(w,z22magdb(1,:))
    title('Respuesta en frecuencia para 7 GDL, x2/F2')
    ylabel('magnitud, db')
    grid
    subplot(2,1,2)
    semilogx(w,z22phs(1,:))
    xlabel('Frecuencia, rad/s')
    ylabel('fase, deg')
    grid
case 19
    subplot(2,1,1)
    semilogx(w,z23magdb(1,:))
    title('Respuesta en frecuencia para 7 GDL, x2/F3')
    ylabel('magnitud, db')
    grid
    subplot(2,1,2)
    semilogx(w,z23phs(1,:))
    xlabel('Frecuencia, rad/s')
    ylabel('fase, deg')
    grid
case 20
    subplot(2,1,1)
    semilogx(w,z24magdb(1,:))
    title('Respuesta en frecuencia para 7 GDL, x2/F4')
    ylabel('magnitud, db')
    grid
    subplot(2,1,2)
    semilogx(w,z24phs(1,:))
    xlabel('Frecuencia, rad/s')
    ylabel('fase, deg')
    grid
case 21
    subplot(2,1,1)
    semilogx(w,z31magdb(1,:))
    title('Respuesta en frecuencia para 7 GDL, x3/F1')
    ylabel('magnitud, db')
    grid
    subplot(2,1,2)
    semilogx(w,z31phs(1,:))
    xlabel('Frecuencia, rad/s')
    ylabel('fase, deg')
    grid
```

```
case 22
    subplot(2,1,1)
    semilogx(w,z32magdb(1,:))
    title('Respuesta en frecuencia para 7 GDL, x3/F2')
    ylabel('magnitud, db')
    grid
    subplot(2,1,2)
    semilogx(w,z32phs(1,:))
    xlabel('Frecuencia, rad/s')
    ylabel('fase, deg')
    grid
case 23
    subplot(2,1,1)
    semilogx(w,z33magdb(1,:))
    title('Respuesta en frecuencia para 7 GDL, x3/F3')
    ylabel('magnitud, db')
    grid
    subplot(2,1,2)
    semilogx(w,z33phs(1,:))
    xlabel('Frecuencia, rad/s')
    ylabel('fase, deg')
    grid
case 24
    subplot(2,1,1)
    semilogx(w,z34magdb(1,:))
    title('Respuesta en frecuencia para 7 GDL, x3/F4')
    ylabel('magnitud, db')
    grid
    subplot(2,1,2)
    semilogx(w,z34phs(1,:))
    xlabel('Frecuencia, rad/s')
    ylabel('fase, deg')
    grid
case 25
    subplot(2,1,1)
    semilogx(w,z41magdb(1,:))
    title('Respuesta en frecuencia para 7 GDL, x4/F1')
    ylabel('magnitud, db')
    grid
    subplot(2,1,2)
    semilogx(w,z41phs(1,:))
    xlabel('Frecuencia, rad/s')
    ylabel('fase, deg')
    grid
case 26
    subplot(2,1,1)
    semilogx(w,z42magdb(1,:))
    title('Respuesta en frecuencia para 7 GDL, x4/F2')
    ylabel('magnitud, db')
    grid
    subplot(2,1,2)
    semilogx(w,z42phs(1,:))
    xlabel('Frecuencia, rad/s')
    ylabel('fase, deg')
    grid
case 27
```

```
        subplot(2,1,1)
        semilogx(w,z43magdb(1,:))
        title('Respuesta en frecuencia para 7 GDL, x4/F3')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,z43phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('fase, deg')
        grid
    case 28
        subplot(2,1,1)
        semilogx(w,z44magdb(1,:))
        title('Respuesta en frecuencia para 7 GDL, x4/F4')
        ylabel('magnitud, db')
        grid
        subplot(2,1,2)
        semilogx(w,z44phs(1,:))
        xlabel('Frecuencia, rad/s')
        ylabel('fase, deg')
        grid
end
```