



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola Politècnica Superior d'Enginyeria
de Manresa



Trabajo Fin de Grado

Uso de herramientas avanzadas de modelización física en el diseño óptimo de sistemas de suspensión

Grado en Ingeniería de Automoción

Curso 20/21

Autor: Pau Alòs Pascual

Director: Dr. Francisco Palacios Quiñonero

Fecha: 06/07/2021

Localidad: Manresa

RESUMEN DEL PROYECTO

En este trabajo se emplea el entorno de simulación numérica *Simscape* para estudiar la respuesta dinámica de sistemas de suspensión. En el estudio se usan modelos simplificados de cuarto de vehículo y de medio vehículo con conductor, y se consideran las respuestas libres y las respuestas forzadas inducidas por un modelo de bache sinusoidal. En la construcción de los modelos computacionales y la ejecución de las simulaciones, se aplican recursos de programación de *Simscape* y *Matlab*, y se utilizan esquemas de computación paralela. Para validar los resultados de las simulaciones *Simscape*, se realiza una comparación con los resultados obtenidos en *Maple* mediante una formulación basada en sistemas de ecuaciones algebraico-diferenciales. Como aplicación, se estudia el diseño óptimo de un sistema de suspensión para el asiento del conductor en un modelo de medio vehículo. Los resultados obtenidos confirman el enorme potencial de *Simscape* en la modelización y simulación numérica de problemas complejos. También indican la relevancia del uso combinado de *Simscape* y estrategias de optimización aleatoria para abordar problemas de diseño óptimo con múltiples parámetros.

RESUMEN DEL PROYECTO (en inglés)

In this work, we use the simulation environment *Simscape* to study the dynamic response of suspension systems. In the study, we employ quarter-car and half-car simplified models and consider both the free responses and the forced responses induced by a sinusoidal bump model. In order to build the computational models and run the numerical simulations, we apply *Simscape* and *Matlab* programming tools and take advantage of parallel computing resources. To validate the results of the *Simscape* simulations, we perform a comparison with the results produced by *Maple* using a system of differential-algebraic equations. As an application, we study the optimal design of a suspension system for the driver's seat in a half-car model. The obtained results confirm the huge potential of *Simscape* for modeling and numerical simulation of complex problems. They also point out the relevance of combining *Simscape* and advanced random optimization strategies in multi-parameter design problems.

ÍNDICE

- 1. INTRODUCCIÓN1**
- 2. MODELOS DE VEHÍCULO.....3**
 - 2.1. MODELO DE UNA MASA.....3
 - 2.2. MODELO DE DOBLE MASA4
 - 2.3. MODELO DE MEDIO VEHÍCULO CON CONDUCTOR6
 - 2.4. MODELO DE ESPACIO DE ESTADO9
- 3. RECURSOS COMPUTACIONALES12**
 - 3.1. MAPLE12
 - 3.2. MATLAB13
 - 3.3. SIMULINK.....14
 - 3.4. SIMSCAPE16
 - 3.5. SIMULACIÓN EN PARALELO19
- 4. RESPUESTA NO FORZADA22**
 - 4.1. RESPUESTA NO FORZADA DEL MODELO DE UNA MASA.....23
 - 4.1.1. Solución exacta con Maple23
 - 4.1.2. Solución numérica con Matlab.....24
 - 4.1.3. Solución numérica con Simulink26
 - 4.1.4. Solución numérica con Simscape27
 - 4.1.5. Comparación de los resultados obtenidos en el modelo de masa simple con respuesta no forzada30
 - 4.2. RESPUESTA NO FORZADA DEL MODELO DE DOBLE MASA33
 - 4.2.1. Solución numérica con Maple33

4.2.2. Resolución numérica con Matlab, Simulink y Simscape	34
5. RESPUESTAS FORZADAS	39
5.1. EXCITACIÓN EXTERNA.....	39
5.2. RESPUESTA FORZADA DEL MODELO DE DOBLE MASA.....	40
5.3. RESPUESTA FORZADA DEL MODELO DE MEDIO VEHÍCULO CON CONDUCTOR.....	44
5.4. VALIDACIÓN DE LOS NUEVOS ELEMENTOS SIMSCAPE	50
6. OPTIMIZACIÓN DEL SISTEMA DE SUSPENSIÓN DEL ASIENTO	52
7. CONCLUSIONES.....	58
8. BIBLIOGRAFÍA.....	60
9. ANEXO. DOCUMENTOS MAPLE.....	61
9.1. CÓDIGO MAPLE PARA LA RESPUESTA FORZADA DEL MODELO DE DOBLE MASA	61
9.2. CÓDIGO MAPLE PARA LA RESPUESTA FORZADA DEL MODELO DE MEDIO VEHÍCULO	64

1. INTRODUCCIÓN

Hoy en día, la simulación numérica es un elemento fundamental en todas las ramas de la ingeniería. *Simscape* es un entorno de simulación numérica asociado a *Matlab* que permite modelar problemas físicos multidominio (mecánico traslacional, mecánico rotacional, eléctrico, térmico, magnético, etc.). Los modelos *Simscape* son diagramas que contienen elementos físicos enlazados por líneas que indican la transferencia de energía entre los distintos elementos. *Simscape* está completamente integrado con *Matlab* y *Simulink*, y dispone de un lenguaje de programación que permite definir nuevos dominios y elementos. Los sistemas de simulación numérica clásicos requieren la construcción previa de un modelo matemático y, en general, una pequeña modificación del problema implica la completa reformulación de las ecuaciones matemáticas y obliga a introducir cambios sustanciales en el modelo. En contraste, los elementos *Simscape* incorporan las ecuaciones algebraicas y diferenciales que definen su comportamiento, y el sistema de ecuaciones algebraico-diferenciales correspondiente a un diagrama *Simscape* se genera de forma automática en una etapa previa del proceso de simulación. Esta característica computacional y la posibilidad de combinar elementos de diferentes dominios físicos permiten la construcción eficiente de modelos complejos.

Debido a su novedad, el nivel de aplicación de *Simscape* es todavía muy limitado. También es muy reducido el número de publicaciones sobre el tema (de hecho, las publicaciones de carácter didáctico sobre el empleo de *Simscape* son prácticamente inexistentes). El objetivo de este trabajo es explorar el uso de *Simscape* en el modelado y simulación de sistemas de suspensión. En particular, se estudia la respuesta dinámica de sistemas de suspensión para modelos de un cuarto de vehículo y de medio vehículo. En la construcción de los modelos, se diseñan varios elementos usando el lenguaje de programación de *Simscape*. Para verificar el buen funcionamiento de los nuevos elementos, se comparan los resultados de las simulaciones *Simscape* con los obtenidos en *Maple* usando una formulación basada en el sistema de ecuaciones algebraico-diferenciales. Como aplicación, se estudia el diseño óptimo de un sistema de suspensión para el asiento del conductor en un modelo de medio vehículo. También se explora la aplicación de recursos de cálculo paralelo, que permiten aprovechar la potencia de cálculo proporcionada por los actuales procesadores multinúcleo.

El contenido del trabajo es el siguiente: En el Capítulo 2, se presentan los distintos modelos de vehículo y se discute brevemente la formulación de espacio de estado para sistemas de ecuaciones diferenciales lineales. En el Capítulo 3, se describen los principales elementos de los entornos de computación *Maple*, *Matlab*, *Simulink* y *Simscape*, y se resalta la importancia de la computación en paralelo en la simulación numérica de sistemas complejos. En el Capítulo 4, se estudia la respuesta no forzada de los modelos de cuarto de vehículo usando *Maple*, *Matlab*, *Simulink* y *Simscape*. En el Capítulo 5, se emplea *Simscape* para estudiar la respuesta forzada del modelo de cuarto de vehículo con doble masa y el modelo de medio vehículo con conductor, tomando como excitación externa un modelo de bache sinusoidal. En el Capítulo 6, se utilizan los modelos *Simscape* y los recursos de computación paralela para optimizar el diseño de un sistema de suspensión. En el Anexo, se incluyen los documentos de *Maple* usados para la validación de los resultados *Simscape* del Capítulo 5.

Los cálculos se han realizado con *Matlab* 2020a y *Maple* 2020, usando un ordenador portátil equipado con un procesador Intel i7 con cuatro núcleos, 32Gb de RAM, un disco sólido de 930Gb y Sistema Operativo Windows 10.

2. MODELOS DE VEHÍCULO

En este capítulo se presentan los modelos de vehículo que se usarán en el trabajo. En particular, se consideraran dos modelos de cuarto de vehículo y un modelo de medio vehículo. También se comenta brevemente la formulación de espacio de estado para ecuaciones diferenciales de orden superior.

2.1. Modelo de una masa

El modelo de una sola masa permite una representación simplificada del comportamiento dinámico de un cuarto de vehículo (*quarter-car*) [1]. Este modelo incluye una masa y un sistema muelle-amortiguador en paralelo que modela el sistema de amortiguación del vehículo. La variable respuesta es el desplazamiento vertical de la masa y la elevación de la carretera actúa como excitación dinámica externa.

La Figura 1 muestra una representación esquemática del modelo de masa simple.

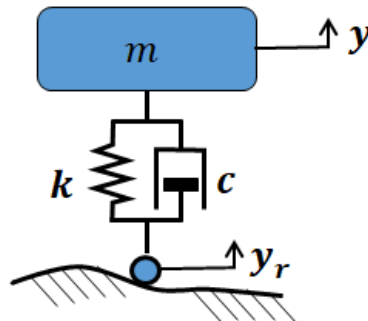


Figura 1. Esquema del modelo de una masa.

Los diferentes símbolos de la Figura 1 tienen el siguiente significado:

m es la masa del vehículo.

$y = y(t)$ es el desplazamiento vertical de la masa.

$y_r = y_r(t)$ es la elevación de la carretera.

c es el coeficiente de amortiguación.

k es el coeficiente elástico del muelle.

Si asumimos que el muelle y el amortiguador tienen un comportamiento lineal, obtenemos el diagrama del sólido libre (DSL) de la Figura 2.

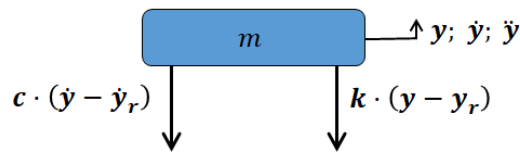


Figura 2. Diagrama de sólido libre correspondiente al modelo de masa simple.

Aplicando la 2ª ley de Newton, la respuesta dinámica del sistema puede describirse mediante la siguiente ecuación diferencial de segundo orden.

$$m \cdot \ddot{y} = -c \cdot (\dot{y} - \dot{y}_r) - k \cdot (y - y_r) \tag{1}$$

Separando la variable respuesta y la excitación, obtenemos:

$$m \cdot \ddot{y} + c \cdot \dot{y} + k \cdot y = c \cdot \dot{y}_r + k \cdot y_r \tag{2}$$

2.2. Modelo de doble masa

El modelo de doble masa permite un estudio más detallado del comportamiento dinámico de un cuarto de vehículo. En este caso, se añade una nueva masa, correspondiente a la rueda, y un segundo sistema muelle-amortiguador que modeliza la elasticidad y la amortiguación asociadas al neumático.

La Figura 3 muestra una representación esquemática del modelo de doble masa.

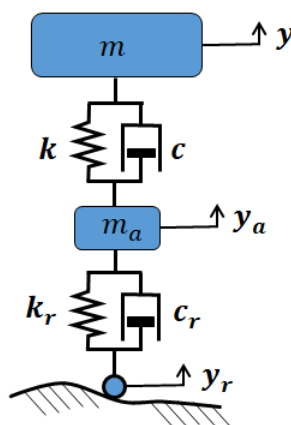


Figura 3. Esquema del modelo de doble masa.

Los distintos símbolos que aparecen en la Figura 3, tienen el siguiente significado:

$y = y(t)$ es el desplazamiento del cuerpo del vehículo.

$y_a = y_a(t)$ es el desplazamiento vertical del eje de la rueda.

$y_r = y_r(t)$ es la elevación de la carretera.

m es la masa del cuerpo del vehículo.

m_a es la masa del sistema eje-rueda.

c es el coeficiente de amortiguación del sistema de suspensión.

k es el coeficiente de elasticidad del sistema de suspensión.

c_r es el coeficiente de amortiguación del neumático.

k_r es el coeficiente de elasticidad del neumático.

En el modelo de doble masa, las variables respuesta son el desplazamiento vertical del cuerpo del vehículo y el desplazamiento vertical de la rueda. La excitación dinámica externa es la elevación de la carretera.

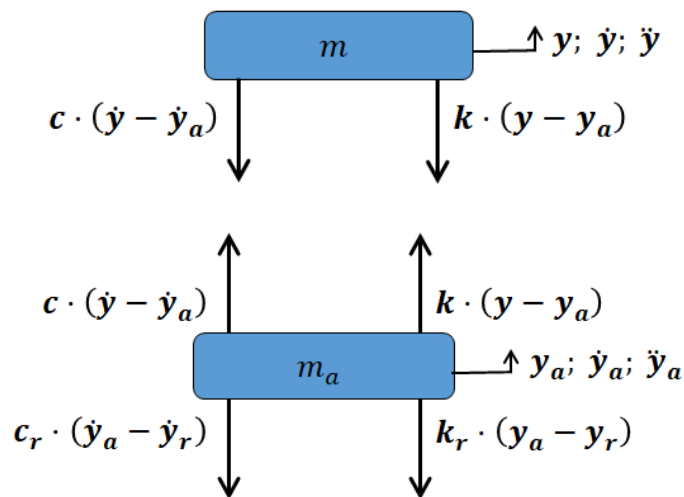


Figura 4. Diagramas del sólido libre correspondientes al modelo de doble masa.

A partir de los diagramas de sólido libre mostrados en la Figura 4, obtenemos el siguiente sistema de ecuaciones diferenciales de segundo orden:

$$\begin{aligned}
 m \cdot \ddot{y} &= -c \cdot (\dot{y} - \dot{y}_a) - k \cdot (y - y_a) \\
 m_r \cdot \ddot{y}_a &= -c_r \cdot (\dot{y}_a - \dot{y}_r) - k_r \cdot (y_a - y_r) + c \cdot (\dot{y} - \dot{y}_a) + k \cdot (y - y_a)
 \end{aligned}
 \tag{3}$$

Si separamos la excitación externa, obtenemos

$$\begin{aligned}
 m \cdot \ddot{y} + c \cdot (\dot{y} - \dot{y}_a) + k \cdot (y - y_a) &= 0 \\
 m_r \cdot \ddot{y}_a + c_r \cdot \dot{y}_a + k_r \cdot y_a + c \cdot (\dot{y}_a - \dot{y}) + k \cdot (y_a - y) &= c_r \cdot \dot{y}_r + k_r \cdot y_r
 \end{aligned}
 \tag{4}$$

Además de los desplazamientos verticales $y(t)$, $y_a(t)$, el modelo de la masa doble permite obtener otras variables respuesta de interés, como, por ejemplo, la aceleración vertical del cuerpo del vehículo $\ddot{y}(t)$, y el desplazamiento relativo del cuerpo del vehículo respecto de la rueda $r(t) = y(t) - y_a(t)$ (rattle space).

2.3. Modelo de medio vehículo con conductor

El modelo de medio vehículo (half car) con conductor permite estudiar el cabeceo (pitch) del cuerpo del vehículo, el desplazamiento vertical de su centro de gravedad, el desplazamiento vertical de los ejes delantero y trasero, y el desplazamiento vertical del sistema asiento-conductor.

Como se puede apreciar en la representación esquemática de la Figura 5, el modelo incluye la masa del sistema asiento-conductor (m_d), las masas de los sistema de eje-rueda delantero y trasero (m_{a1} , m_{a2}), y un cuerpo del vehículo de masa m y momento de inercia J (respecto el centro de gravedad).

Los puntos de anclaje de los sistemas de suspensión delantero y trasero están situados a una distancia a_1 y a_2 del centro de gravedad, respectivamente, y la distancia del centro de gravedad a la posición del conductor es a_3 .

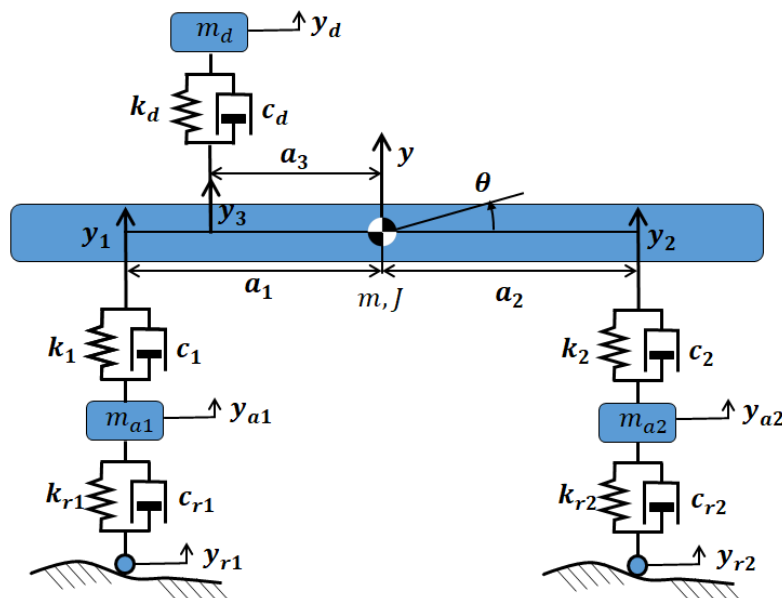


Figura 5. Esquema del modelo de medio vehículo con conductor.

Los símbolos de la Figura 5 tienen el siguiente significado:

$y_d = y_d(t)$ es el desplazamiento vertical del sistema asiento.

m_d es la masa del sistema asiento-conductor.

c_d, k_d son los coeficiente de amortiguación y de elasticidad del asiento.

$y_3 = y_3(t)$ es el desplazamiento vertical del punto de anclaje del asiento.

a_3 es la distancia del anclaje del asiento al centro de gravedad.

$y = y(t)$ es el desplazamiento vertical del centro de gravedad del cuerpo del vehículo.

m es la masa del cuerpo del vehículo.

$\theta = \theta(t)$ es el ángulo de cabeceo (pitch) del cuerpo del vehículo.

J es el momento de inercia del vehículo respecto del centro de gravedad.

$y_1 = y_1(t)$ es el desplazamiento vertical del punto de anclaje del sistema de suspensión delantero.

c_1, k_1 son los coeficiente de amortiguación y de elasticidad del sistema de suspensión delantero.

a_1 es la distancia del anclaje del sistema de suspensión delantero al centro de gravedad.

$y_2 = y_2(t)$ es el desplazamiento vertical del anclaje del sistema de suspensión trasero.

c_2, k_2 son los coeficiente de amortiguación y de elasticidad del sistema de suspensión trasero.

a_2 es la distancia del anclaje del sistema de suspensión trasero al centro de gravedad.

$y_{a1} = y_{a1}(t)$ es el desplazamiento vertical del sistema eje-rueda delantero.

m_{a1} es la masa del sistema eje-rueda delantero.

c_{r1}, k_{r1} son los coeficiente de amortiguación y de elasticidad del neumático delantero.

$y_{a2} = y_{a2}(t)$ es el desplazamiento vertical del sistema eje-rueda trasero.

m_{a2} es la masa del sistema eje-rueda trasero.

c_{r2}, k_{r2} son los coeficiente de amortiguación y de elasticidad del neumático trasero.

$y_{r1} = y_{r1}(t)$ es la elevación de la carretera en el neumático delantero.

$y_{r2} = y_{r2}(t)$ es la elevación de la carretera en el neumático trasero.

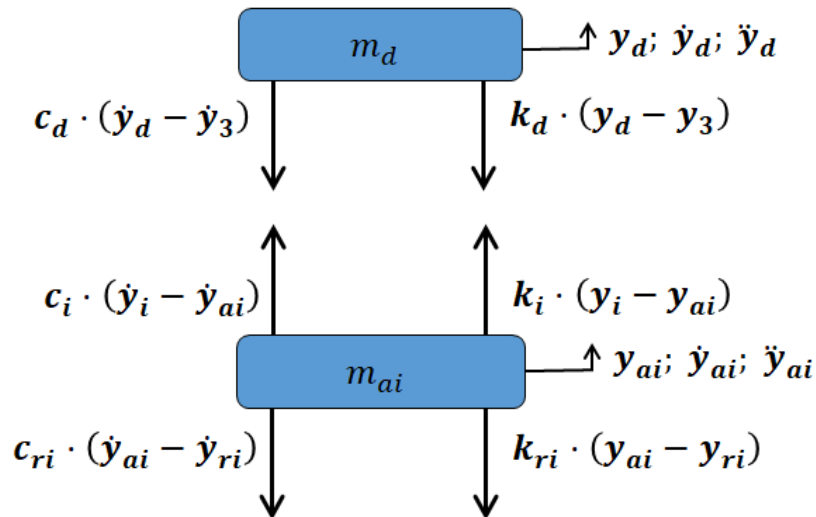


Figura 6. Diagramas del sólido libre de la masa del conductor y las masas de los ejes en el modelo de medio vehículo con conductor.

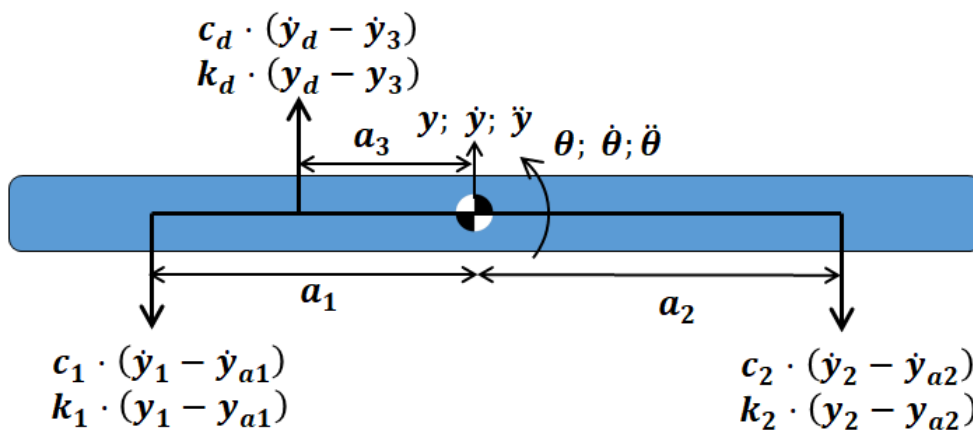


Figura 7. Diagrama del sólido libre correspondiente al cuerpo del vehículo para el modelo de medio vehículo con conductor.

A partir de los diagramas de sólido libre en la Figura 6 y la Figura 7, y aplicando las ecuaciones de Newton-Euler, podemos formular un sistema de 5 ecuaciones diferenciales de segundo orden con un total de 8 incógnitas: $y_d(t)$, $y(t)$, $\theta(t)$, $y_1(t)$, $y_2(t)$, $y_3(t)$, $y_{a1}(t)$, $y_{a2}(t)$.

$$m_d \cdot \ddot{y}_d = -k_d \cdot (y_d - y_3) - c_d \cdot (\dot{y}_d - \dot{y}_3) \tag{5}$$

$$m \cdot \ddot{y} = k_d \cdot (y_d - y_3) + c_d \cdot (\dot{y}_d - \dot{y}_3) - k_1 \cdot (y_1 - y_{a1}) - c_1 \cdot (\dot{y}_1 - \dot{y}_{a1}) - k_2 \cdot (y_2 - y_{a2}) - c_2 \cdot (\dot{y}_2 - \dot{y}_{a2}) \tag{6}$$

$$J \cdot \ddot{\theta} = -k_d \cdot a_3 \cdot (y_d - y_3) - c_d \cdot a_3 (\dot{y}_d - \dot{y}_3) + k_1 \cdot a_1 \cdot (y_1 - y_{a1}) + c_1 \cdot a_1 \cdot (\dot{y}_1 - \dot{y}_{a1}) - k_2 \cdot a_2 \cdot (y_2 - y_{a2}) - c_2 \cdot a_2 \cdot (\dot{y}_2 - \dot{y}_{a2}) \quad (7)$$

$$m_{a1} \cdot \ddot{y}_{a1} = k_1 \cdot (y_1 - y_{a1}) + c_1 (\dot{y}_1 - \dot{y}_{a1}) - k_{r1} \cdot (y_{a1} - y_{r1}) - c_{r1} (\dot{y}_{a1} - \dot{y}_{r1}) \quad (8)$$

$$m_{a2} \cdot \ddot{y}_{a2} = k_2 \cdot (y_2 - y_{a2}) + c_2 (\dot{y}_2 - \dot{y}_{a2}) - k_{r2} \cdot (y_{a2} - y_{r2}) - c_{r2} (\dot{y}_{a2} - \dot{y}_{r2}) \quad (9)$$

Si expresamos el ángulo de cabeceo $\theta(t)$ en radianes y aplicamos la aproximación de ángulo pequeño $\sin \theta \approx \theta$, se obtienen las siguientes relaciones algebraicas para el desplazamiento vertical de los puntos de anclaje $y_1(t)$, $y_2(t)$, $y_3(t)$:

$$y_1 = y - a_1 \cdot \theta \quad (10)$$

$$y_2 = y + a_2 \cdot \theta \quad (11)$$

$$y_3 = y - a_3 \cdot \theta \quad (12)$$

Finalmente combinando las ecuaciones (5) – (9) y (10) – (12) resulta un sistema algebraico-diferencial de 8 ecuaciones con 8 incógnitas.

Conviene destacar que actualmente existen recursos de computación numérica (DAE Solvers) que permiten resolver este tipo de problemas de forma eficiente. En el presente trabajo, usamos los DAE Solvers implementados en el comando *dsolve()* de *Maple* para simular la respuesta dinámica del modelo de medio vehículo con conductor.

2.4. Modelo de espacio de estado

En las últimas décadas se han desarrollado procedimientos numéricos que permiten resolver sistemas de ecuaciones diferenciales de forma eficiente (ODE Solvers). Usualmente, estos ODE Solvers, pueden aplicarse directamente a ecuaciones diferenciales de primer orden. En el caso de las ecuaciones diferenciales de orden superior, puede emplearse una formulación de espacio de estado, que permite reducir el orden de las ecuaciones diferenciales mediante el aumento del número de incógnitas. Usando este método, podemos definir una ecuación diferencial de segundo orden como un sistema de ecuaciones diferenciales de primer grado.

Así, por ejemplo, para la ecuación diferencial de segundo orden

$$\ddot{y}(t) + a \cdot \dot{y}(t) + b \cdot y(t) = g(t)$$

podemos tomar las variables de estado $x_1(t) = y(t)$, $x_2(t) = \dot{y}(t)$ y obtener el siguiente sistema de ecuaciones diferenciales de primer orden:

$$\begin{cases} \dot{x}_1(t) = x_2(t) \\ \dot{x}_2(t) = -b \cdot x_1(t) - a \cdot x_2(t) + g(t) \end{cases}$$

Este sistema puede escribirse de forma matricial como

$$\dot{X}(t) = A \cdot X(t) + B \cdot U(t) \quad (13)$$

donde

$$X(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

es el vector de estado,

$$U(t) = \begin{bmatrix} 0 \\ g(t) \end{bmatrix}$$

es el vector de entrada,

$$A = \begin{bmatrix} 0 & 1 \\ -b & -a \end{bmatrix}$$

es la matriz de estado, y

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

es la matriz de entrada.

La ecuación de estado (13) suele complementarse con una ecuación de salida

$$Z(t) = C \cdot X(t) + D \cdot U(t) \quad (14)$$

donde $Z(t)$ es un vector que contiene las variables de interés. Así, por ejemplo, si estamos interesados en las variables $y(t)$, $\dot{y}(t)$, $\ddot{y}(t)$, podemos tomar las variables de salida $z_1(t) = y(t)$, $z_2(t) = \dot{y}(t)$, $z_3(t) = \ddot{y}(t)$, que pueden obtenerse a partir del vector de estado $X(t)$ y el vector de entrada $U(t)$ mediante la matriz de salida

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -b & -a \end{bmatrix}$$

y la matriz de transmisión directa

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

El procedimiento descrito puede aplicarse de forma general a sistemas de ecuaciones diferenciales lineales con coeficientes constantes como los considerados en las secciones anteriores. Observamos que el modelo de espacio de estado formado por las ecuaciones (13) y (14) queda determinado por las matrices A , B , C y D . Una vez obtenidas estas matrices, pueden realizarse simulaciones numéricas de las respuesta del sistema usando ODE Solvers proporcionados por diversos entornos de computación, como, por ejemplo, los comandos *lsim()* y *initial()* incluidos en el *Control System Toolbox* de *Matlab* [2].

3. RECURSOS COMPUTACIONALES

En este capítulo se presenta una breve descripción de los entornos de computación *Maple*, *Matlab*, *Simulink* y *Simscape*. También se incluye un breve comentario sobre el uso de la computación paralela en *Matlab* para el uso eficiente de procesadores multinúcleo.

3.1. Maple

Maple [3] es un entorno de computación simbólica, numérica y gráfica de muy altas prestaciones. Tal como se muestra en la Figura 8, la interacción con el programa puede realizarse mediante documentos de trabajo que combinan fragmentos de texto con cálculos numéricos y simbólicos, y representaciones gráficas. La entrada del código puede realizarse en formato matemático, lo que facilita enormemente la definición de los problemas y la interpretación de los resultados. El comando *dsolve()* de *Maple* permite obtener soluciones simbólicas para problemas sencillos, y proporciona soluciones numéricas para problemas más complejos. Además puede aplicarse a problemas que combinen ecuaciones diferenciales y ecuaciones algebraicas, y que contengan ecuaciones diferenciales de orden superior.

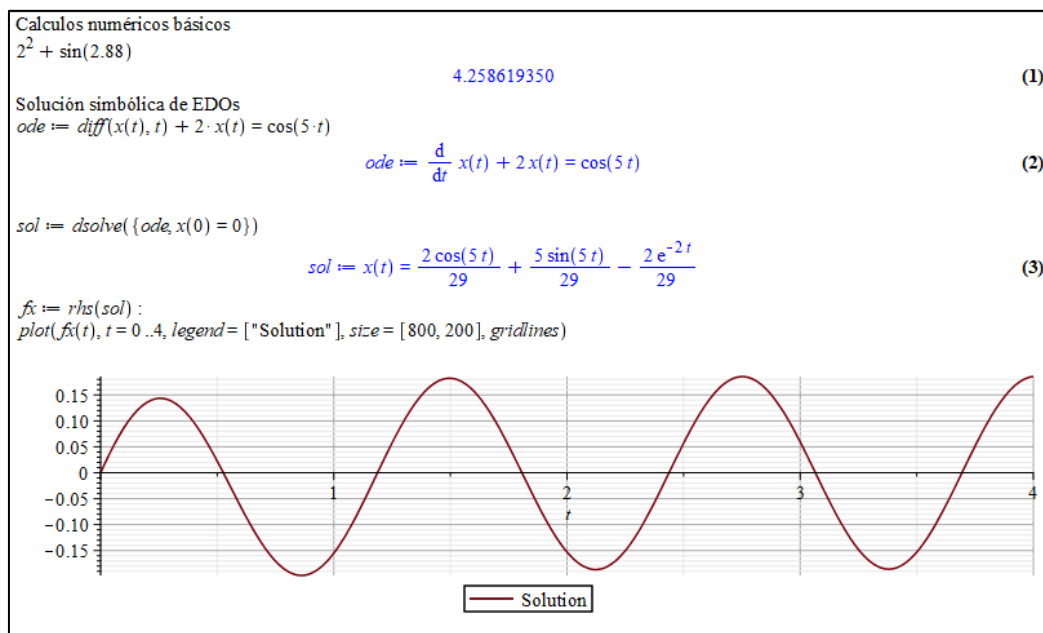


Figura 8. Hoja de trabajo de Maple.

3.2. Matlab

Matlab [4] es un entorno de computación numérica con un conjunto de librerías (Toolbox) que proporcionan herramientas computacionales avanzadas para una gran variedad de aplicaciones científicas y técnicas. También incluye herramientas para confeccionar gráficos de alta calidad y un completo lenguaje de programación.

La interfaz clásica de Matlab tiene la estructura típica de un entorno integrado de programación (IDE), con diversas ventanas para la ejecución de comandos, el seguimiento de las variables definidas, la revisión del historial de comandos ejecutados y la inspección del sistema de archivos (ver Figura 9). Los scripts de Matlab (m-files) son archivos de texto que combinan secuencias de órdenes y estructuras de programación, y son un recurso muy efectivo para la resolución de problemas complejos.

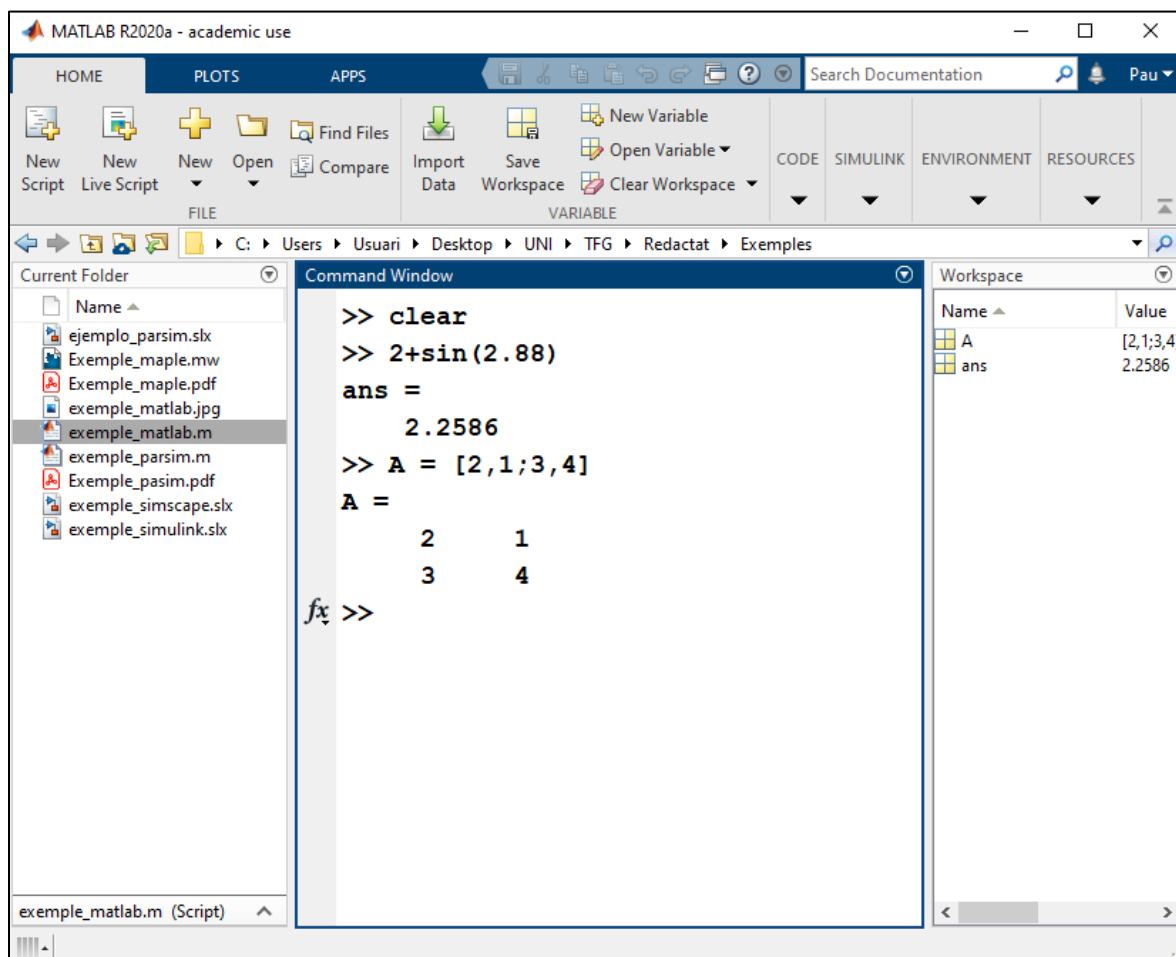


Figura 9. Interface clásico de Matlab.

Matlab incluye una librería de cálculo simbólico (Symbolic Math Toolbox [5]) que permite resolver problemas de dificultad moderada (ver Figura 10). En las últimas versiones de

Matlab, se ha incorporado un nuevo tipo de documento interactivo (live script) que combina bloques de texto, secuencias de órdenes y resultados.

```

Command Window
>> syms x(t)
>> ode = diff(x,t) + 2*x == cos(5*t)
ode(t) =
2*x(t) + diff(x(t), t) == cos(5*t)
>> IC = x(0) == 1
IC =
x(0) == 1
>> sol = dsolve(ode,IC)
sol =
0.9310*exp(-2*t) + 0.1857*cos(5*t - 1.1903)
fx >>
    
```

Figura 10. Cálculo simbólico con Matlab.

3.3. Simulink

Simulink [6] es un entorno de simulación numérica asociado a *Matlab*. Los modelos de *Simulink* se definen mediante diagramas gráficos que combinan bloques de entrada/salida, bloques de cálculo matemático y flechas de enlace que indican la dirección del flujo de información. La Figura 11 contiene un diagrama *Simulink* para la resolución numérica de una ecuación diferencial de primer orden. La representación gráfica de la solución obtenida se muestra en la Figura 12.

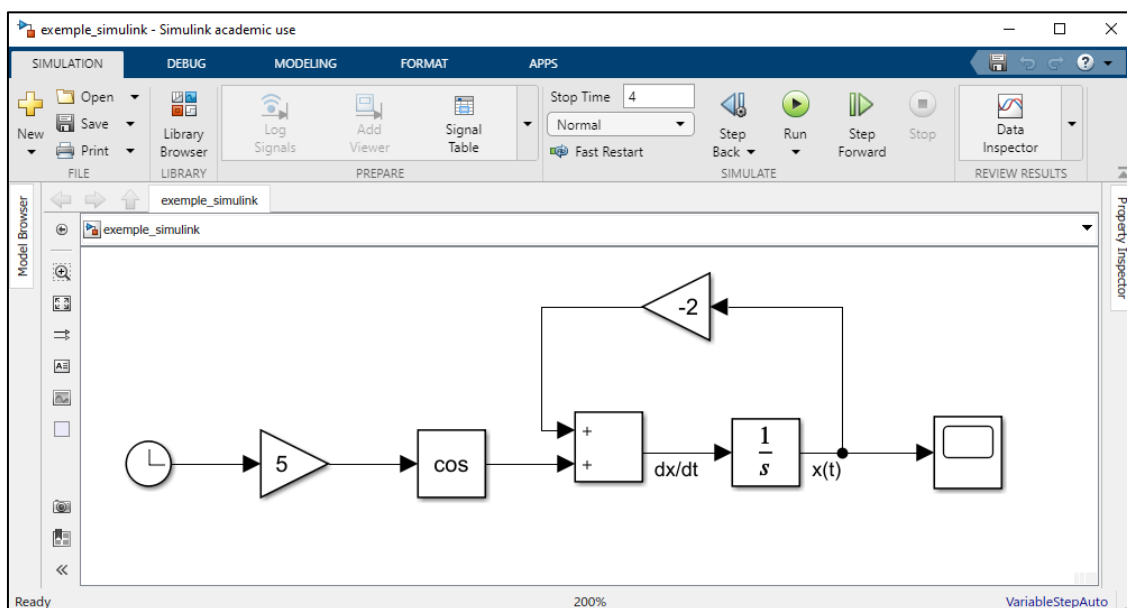


Figura 11. Diagrama *Simulink* para la resolución numérica de una ecuación diferencial.

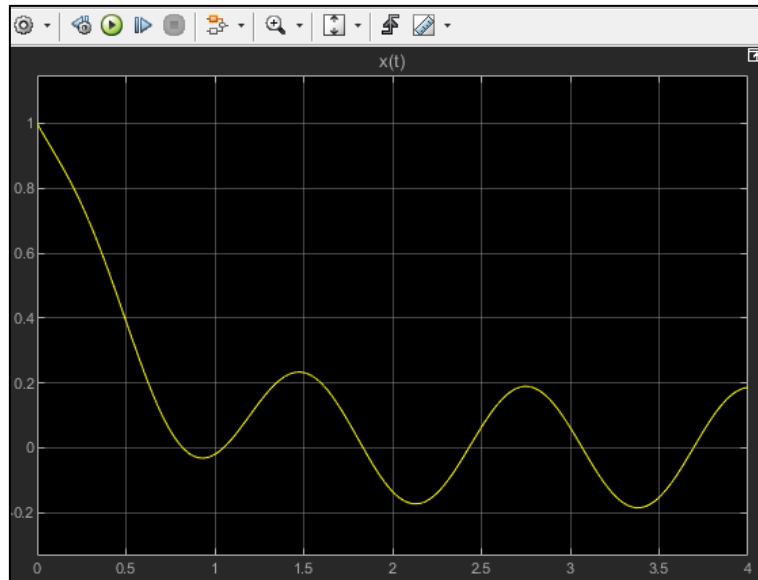


Figura 12. Representación gráfica de la solución calculada en el diagrama de la Figura 11.

Simulink permite definir varios niveles de subsistemas. También incorpora un buen número de librerías con bloques predefinidos para realizar tareas complejas correspondientes a diversos campos científicos y técnicos. Para adaptarse a las características numéricas de los diferentes problemas, *Simulink* dispone de varios ODE Solvers de paso fijo y variable, cuyos parámetros pueden ajustarse en el correspondiente cuadro de configuración (ver Figura 13). Una de las características fundamentales de *Simulink* es la integración total con *Matlab*. En particular, es posible controlar la configuración y ejecución de un modelo *Simulink* desde un *script* de *Matlab* (m-file). Una vez completada la simulación, los resultados numéricos obtenidos con el modelo *Simulink* pueden asignarse a variables del espacio de trabajo de *Matlab* para su procesado y análisis.

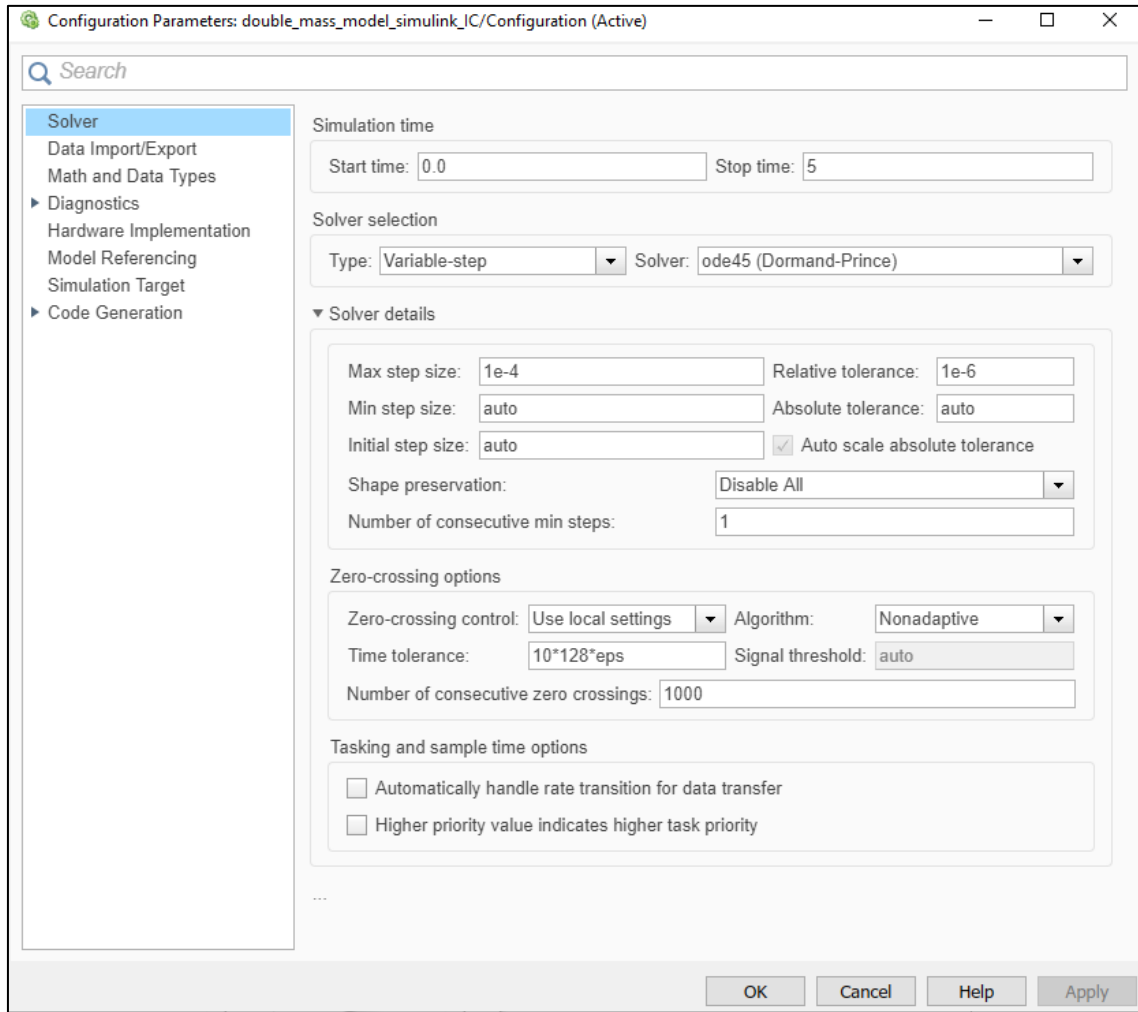


Figura 13. Cuadro de dialogo para las opciones de los ODE Solvers de Simulink.

3.4. Simscape

Simscape [7] es una extensión de *Simulink* para la simulación de modelos físicos y multidominio. Los modelos *Simscape* son diagramas que contienen elementos físicos enlazados por líneas (ramas) que indican la transferencia de energía entre los distintos elementos. En la Figura 14 se muestra un diagrama *Simscape* para la simulación de la respuesta dinámica de una masa con un sistema de muelle-amortiguador en paralelo.

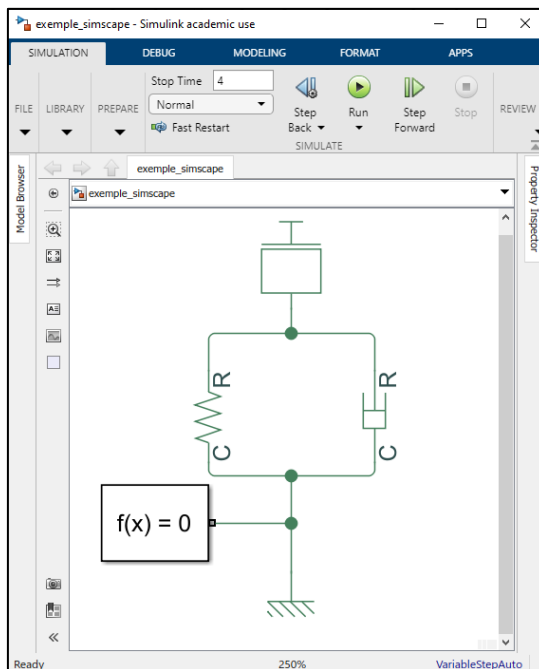


Figura 14. Diagrama Simscape para un sistema masa muelle-amortiguador.

Los elementos *Simscape* están organizados en diferentes dominios físicos (mecánico traslacional, mecánico rotacional, térmico, magnético, etc.). Cada dominio está caracterizado por un par de variables (*through* y *across*). Típicamente, los elementos de *Simscape* incluyen puertos conservativos para la transmisión de energía dentro de un mismo dominio y un conjunto de ecuaciones constitutivas que determinan el comportamiento del elemento. También pueden contener puertos de tipo señal para el intercambio de información. En el dominio mecánico traslacional, la variable *through* es la fuerza y la variable *across* es la velocidad. En el dominio mecánico rotacional la variable *through* es el momento angular y la variable *across* es la velocidad angular. En ambos casos, el producto de las variables *through* y *across* es igual a la potencia mecánica.

La definición de los dominios y bloques *Simscape* se realizan mediante *scripts* especiales (con extensión *.ssc*). *Simscape* incluye una librería básica de elementos predefinidos. También incorpora recursos de programación para definir nuevos bloques de usuario. En las Figuras 15 y 16 se muestran los elementos de las librerías de mecánica traslacional y sensores mecánicos.

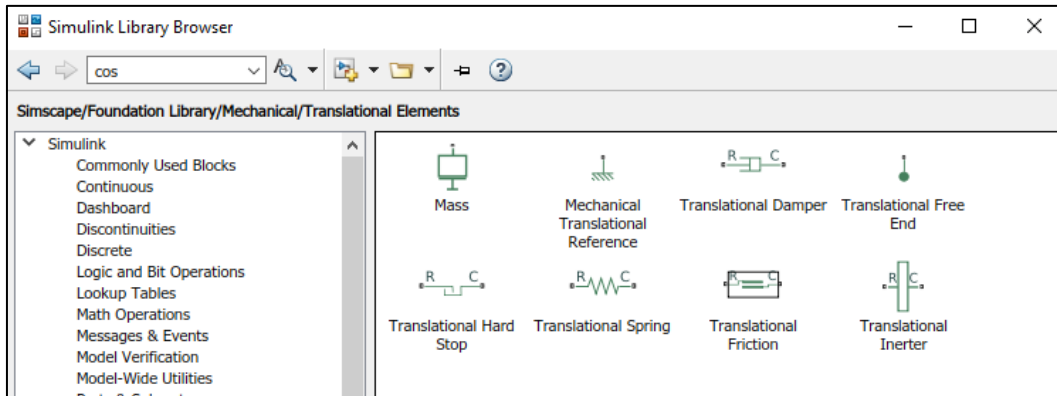


Figura 15. Librería Simscape de elementos mecánicos traslacionales.

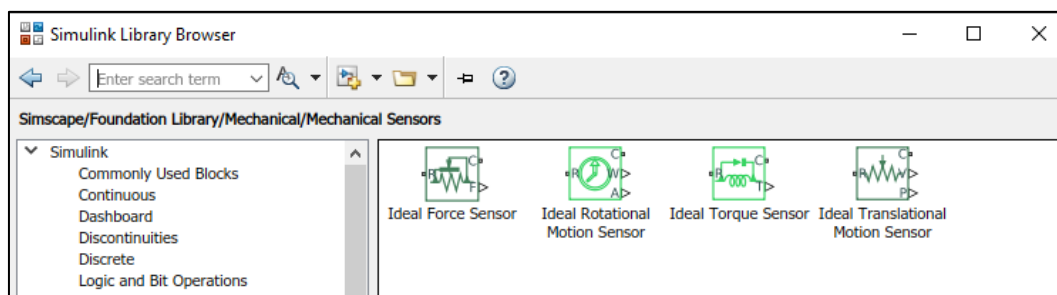


Figura 16. Librería Simscape de sensores mecánicos.

```

1 component acceleration_sensor
2 % Sensor de aceleración lineal ideal
3 nodes
4     R = foundation.mechanical.translational.translational; % R:left
5     C = foundation.mechanical.translational.translational; % C:right
6 end
7 outputs
8 % Nos define el valor A como la aceleración relativa entre nodos
9     A = { 0, 'm/s^2' }; % A:right
10 end
11 variables(Access = private)
12     v = {0, 'm/s'}; % velocity
13 end
14 equations
15     v == R.v - C.v;
16     % Define A como la derivada de la velocidad relativa
17     A == v.der;
18 end
19 end
    
```

Figura 17. Script Simscape para definir un sensor de aceleración en el dominio mecánico traslacional.

En la Figura 17, se muestra un *script Simscape* que permite definir un sensor de aceleración mediante un bloque de usuario [8].

Los modelos *Simscape* son un tipo particular de modelo *Simulink* y pueden combinarse con elementos *Simulink* estándar mediante los bloques de conversión *PS-Simulink* y

Simulink-PS que realizan la transformación entre las señales físicas de *Simscape* y las señales numéricas de *Simulink*. (ver Figura 18)

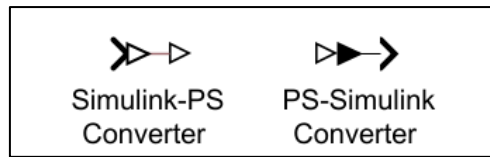


Figura 18. Bloques de enlace entre diagramas Simulink y diagramas Simscape.

Igual como sucede con los modelos *Simulink*, los modelos *Simscape* se pueden configurar y ejecutar desde un *script Matlab*. La relación entre *Matlab*, *Simulink* y *Simscape* se muestra de forma esquemática en la Figura 19.

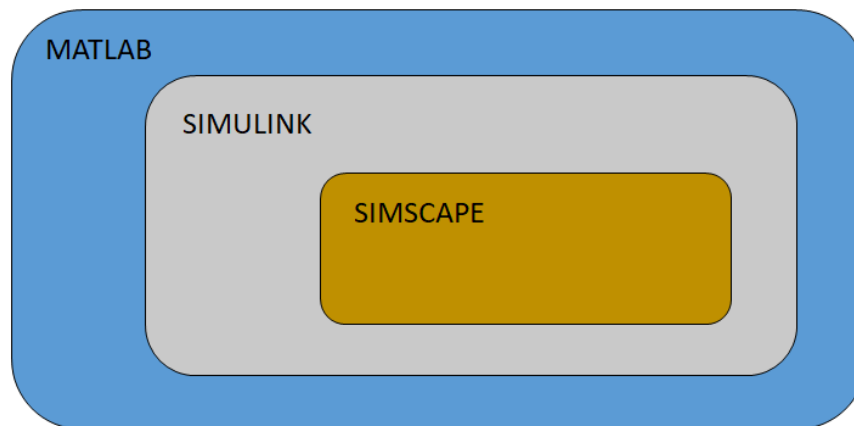


Figura 19. Relación entre Matlab, Simulink y Simscape.

Conviene destacar que los enlaces de los diagramas *Simulink* son direccionales y la evaluación de los distintos bloques se realiza de forma secuencial siguiendo el orden determinado por los enlaces. En cambio, los diagramas *Simscape* no tienen un orden establecido. Para su simulación numérica, se utiliza una etapa de preprocesado, que transforma el diagrama *Simscape* en un sistema de ecuaciones algebraico-diferenciales. Si el diagrama *Simscape* forma parte de un diagrama *Simulink*, el diagrama de *Simscape* puede considerarse como un solo bloque *Simulink*. La direccionalidad de los bloques de enlace *Simulink-PS* y *PS-Simulink* determina el momento de evaluación del bloque *Simscape* dentro del diagrama *Simulink*.

3.5. Simulación en paralelo

La potencia computacional es un factor decisivo en la simulación numérica de modelos complejos. En este contexto, el aprovechamiento de los recursos de computación

paralela proporcionados por los actuales procesadores multinúcleo es un elemento de gran importancia.

El *Parallel Computing Toolbox* [9] proporciona herramientas para implementar esquemas de cálculo paralelo en ordenadores con procesadores multinúcleo. En particular, el comando *parsim()* permite realizar simulaciones en paralelo de modelos *Simulink* en los diferentes núcleos del procesador, aprovechando de forma eficiente la capacidad computacional del ordenador. Para usar este comando, hay que generar previamente una estructura de datos que contenga los valores particulares correspondientes a las diferentes simulaciones. La Figura 20 muestra un ejemplo de *script Matlab* para la simulación en paralelo de un modelo *Simscape*. En la Figura 21 puede verse el proceso de inicialización del servidor de cálculo paralelo y la traza de ejecución en un lote de 10 simulaciones

```

1   % Nombre del modelo a simular
2   model_name = 'ejemplo_parsim';
3   % Vector de valores de la variable
4   gain_s = 1:10;
5
6   open_system(model_name,'loadonly')
7   for j = 1:length(gain_s)
8       % Estructura de inputs
9       in(j) = Simulink.SimulationInput(model_name);
10      % Definimos los parametros por cada estrucutra
11      in(j)=in(j).setVariable('gain',gain_s(j));
12  end
13  % Simulamos el modelo
14  sol = parsim(in,'ShowSimulationManager','off','ShowProgress','on');
```

Figura 20. Script Matlab para la simulación en paralelo de un modelo Simulink.

```

Command Window
[29-May-2021 17:16:54] Checking for availability of parallel pool...
[29-May-2021 17:16:54] Starting Simulink on parallel workers...
[29-May-2021 17:16:55] Building model references in parallel...
[29-May-2021 17:16:55] Configuring simulation cache folder on parallel workers...
[29-May-2021 17:16:55] Loading model on parallel workers...
[29-May-2021 17:16:58] Running simulations...
[29-May-2021 17:17:10] Completed 1 of 10 simulation runs
[29-May-2021 17:17:10] Completed 2 of 10 simulation runs
[29-May-2021 17:17:10] Completed 3 of 10 simulation runs
[29-May-2021 17:17:10] Completed 4 of 10 simulation runs
[29-May-2021 17:17:20] Completed 5 of 10 simulation runs
[29-May-2021 17:17:20] Completed 6 of 10 simulation runs
[29-May-2021 17:17:20] Completed 7 of 10 simulation runs
[29-May-2021 17:17:20] Completed 8 of 10 simulation runs
[29-May-2021 17:17:28] Completed 9 of 10 simulation runs
[29-May-2021 17:17:28] Completed 10 of 10 simulation runs
[29-May-2021 17:17:28] Cleaning up parallel workers...
fx >>
```

Figura 21. Progreso de la simulación en paralelo.

El *Parallel Computing Toolbox* también incluye una herramienta gráfica que permite realizar el seguimiento de forma visual del progreso de la simulación en paralelo (ver Figura 22).

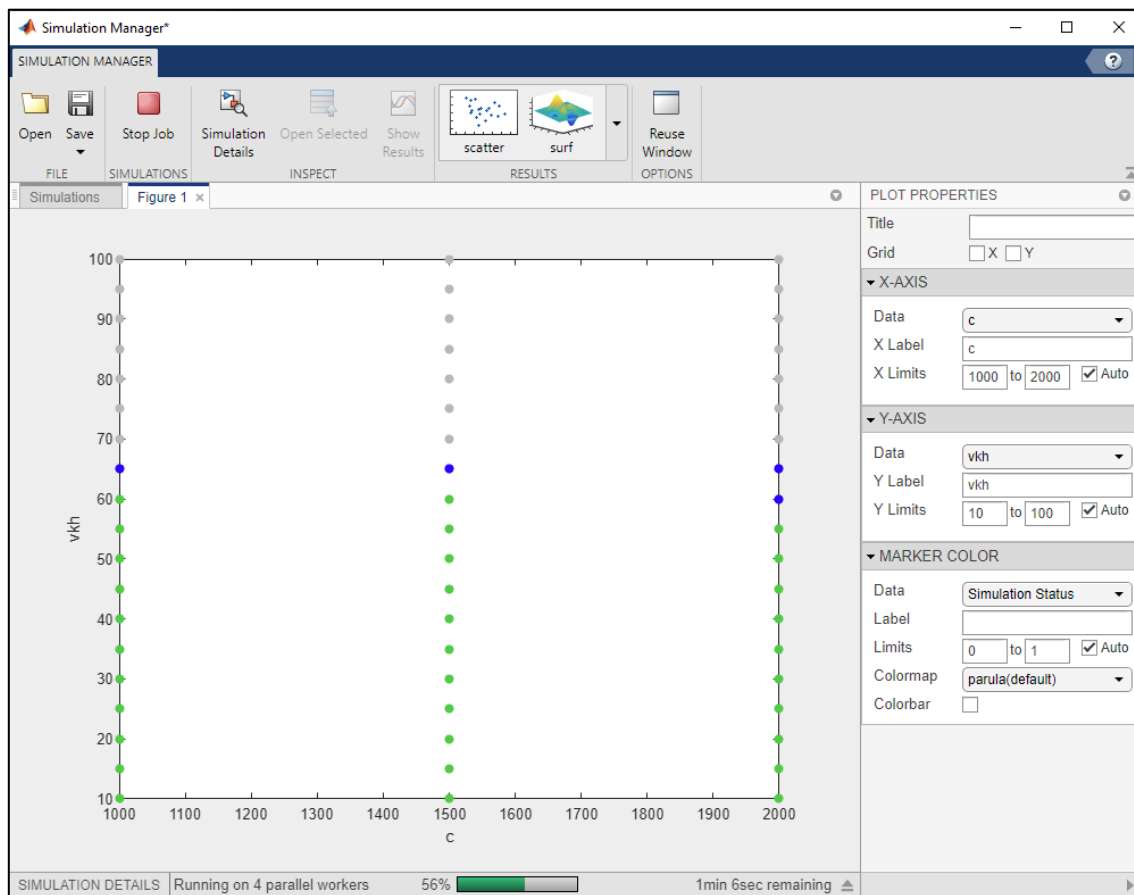


Figura 22. Herramienta gráfica para el seguimiento de simulaciones en paralelo.

Conviene observar que el comando *parsim()* permite aplicar una estrategia de “fuerza bruta” para explorar el comportamiento de modelos con uno o dos parámetros. Para modelos con un mayor número de parámetros, la realización de las simulaciones correspondientes a todas las configuraciones paramétricas deja de ser una estrategia viable. Así, por ejemplo, en un problema con 6 parámetros, la exploración del modelo tomando 10 valores por cada parámetro, conduciría a la realización de 10^6 simulaciones. Con 100 valores por parámetro, el número de simulaciones sería de 10^{12} . En este caso, pueden utilizarse estrategias de optimización avanzada que emplean un número reducido de configuraciones mediante selección aleatoria. Algunas de estas estrategias, como el algoritmo genético o el enjambre de partículas (particle swarm), pueden implementarse usando herramientas proporcionadas por el *Global Optimization Toolbox* de *Matlab*.

4. RESPUESTA NO FORZADA

En este capítulo se estudia la respuesta no forzada de los modelos de masa simple y masa doble presentados en el Capítulo 2. En el modelo de una masa se toma un desplazamiento inicial nulo y una velocidad de la masa de $1m/s$ como condiciones iniciales. En el modelo de doble masa, se considera un desplazamiento inicial nulo en ambas masas, una velocidad nula en la masa del sistema eje-rueda y una velocidad de $1m/s$ en la masa del cuerpo del vehículo. Los parámetros empleados en las simulaciones están recogidos en la Tabla 1 y son los valores propuestos en [10].

Parámetro	Definición	Valor	Unidades
m	Masa del vehículo	250	kg
m_r	Masa de la rueda	45	kg
k	Coeficiente de elasticidad de la suspensión	16000	N/m
c	Coeficiente de amortiguación de la suspensión	1000	$N \cdot s/m$
k_r	Coeficiente de elasticidad de la rueda	160000	N/m
c_r	Coeficiente de amortiguación de la rueda	0	$N \cdot s/m$

Tabla 1. Valores de los parámetros usados en la simulación del modelo de masa simple y de doble masa.

El objetivo del capítulo es estudiar la respuesta de los modelos usando diferentes herramientas computacionales y verificar la consistencia de los resultados obtenidos. Para el modelo de masa simple, se realiza la resolución simbólica con *Maple* y la resolución numérica con *Matlab*, *Simulink* y *Simscape*. En el modelo de masa doble, la solución en *Maple* también se obtiene de forma numérica. Las comparaciones de las respuestas se realizan de forma gráfica y numérica, tomando como referencia la solución proporcionada por *Maple*.

4.1. Respuesta no forzada del modelo de una masa

4.1.1. Solución exacta con Maple

El problema de valor inicial correspondiente a la respuesta no forzada del modelo de una masa es el siguiente:

$$\begin{cases} m \cdot \ddot{y} + c \cdot \dot{y} + k \cdot y = 0 \\ y(0) = 0, \dot{y}(0) = 1 \end{cases} \quad (15)$$

El código Maple para definir y resolver este problema de forma simbólica se muestra en las Figuras 23 y 24. La representación gráfica se presenta en la Figura 25.

```
restart
Definición de la EDO del modelo
De := m·ÿ + c·ẏ + k·y(t) = 0
De := m ( d² y(t) / dt² ) + c ( d y(t) / dt ) + k y(t) = 0 (1)
Condiciones iniciales
IC := {y(0) = 0, D(y)(0) = 1}
IC := {y(0) = 0, D(y)(0) = 1} (2)
Solución simbólica
sol := dsolve( {De} union IC )
sol := y(t) = ( m e^{(-c + sqrt(c² - 4km)) t} / sqrt(c² - 4km) - m e^{(c + sqrt(c² - 4km)) t} / sqrt(c² - 4km) ) (3)
```

Figura 23. Solución simbólica general del problema de valor inicial.

```
Parámetros del sistema
param := {m = 250, k = 16000, c = 1000 }
param := {c = 1000, k = 16000, m = 250} (4)
Solución particular
solp := subs(param, sol)
solp := y(t) = - ( sqrt(-15000000) e^{(-1000 + sqrt(-15000000)) t} / 60000 + sqrt(-15000000) e^{(1000 + sqrt(-15000000)) t} / 60000 ) (5)
Evaluación float
evalf(rhs(solp))
-0.06454972245 I e^{(-2.000000000 + 7.745966692 I) t} + 0.06454972245 I e^{(-2.000000000 - 7.745966692 I) t} (6)
```

Figura 24. Solución correspondiente a los valores de los parámetros de la Tabla 1.

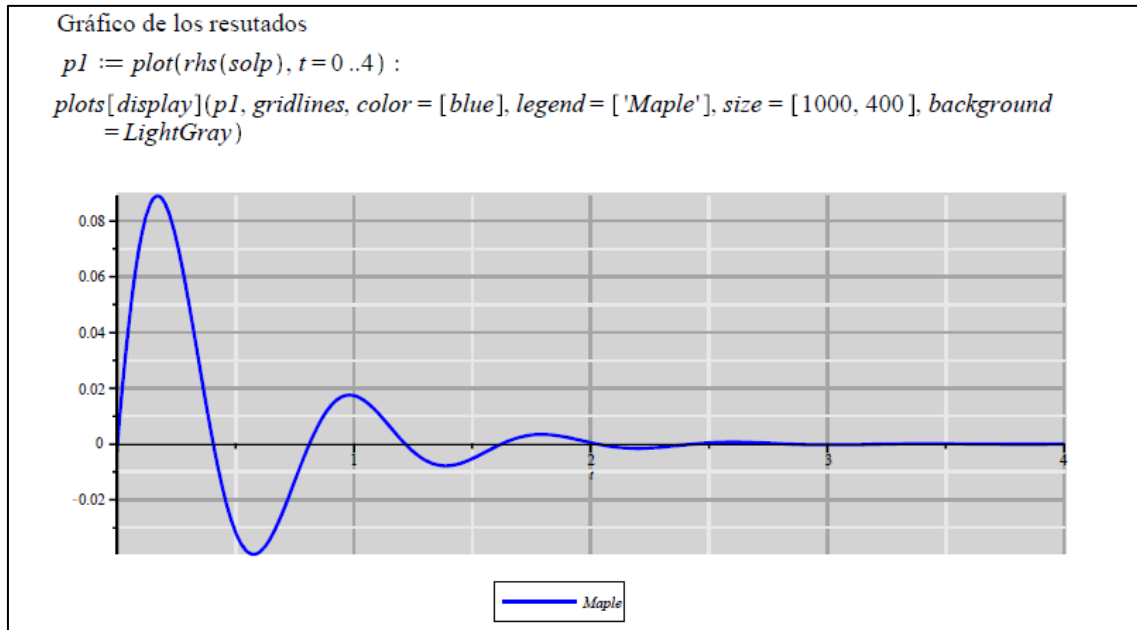


Figura 25. Representación gráfica de la solución simbólica obtenida con Maple para la respuesta no forzada del modelo de una masa.

4.1.2. Solución numérica con Matlab

Para calcular la respuesta del modelo de una masa usando *Matlab*, formulamos el problema en forma de espacio de estado. Tomando las variables de estado $x_1(t) = y(t)$, $x_2(t) = \dot{y}(t)$ y la variable de salida $z = y(t)$, obtenemos la formulación de espacio de estado

$$\begin{cases} \dot{X} = A \cdot X + B \cdot u \\ Z = C \cdot X + D \cdot u \end{cases} \quad (16)$$

con

$$A = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

$$C = [1 \quad 0], \quad D = [0]$$

Usando las herramientas del *Control System Toolbox* de *Matlab*, el modelo de espacio de estado puede definirse con el comando `ss()` y la solución numérica del problema de valor inicial puede calcularse con el comando `initial()`. La Figura 26 muestra un *script Matlab* que permite realizar la simulación, y el gráfico de la respuesta se presenta en la Figura 27.

```

single_mass_IC.m x +
15 %% State Space
16 % Vector de tiempos
17 t_grid = 0:1e-3:5;
18 % Definimos las matrices del espacio de estado
19 A = [0, 1;
20      -k/m, -c/m];
21 B = [0, 0]';
22 C = [1, 0];
23 D = [0];
24 % Modelo de espacio de estado
25 sys = ss(A,B,C,D);
26 % Condiciones iniciales
27 IC = [0,1];
28 % Solución
29 y_ss = initial(sys,IC,t_grid);
30 % Graficos
31 plot(t_grid,y_ss)

```

Figura 26. Código Matlab para la simulación con espacio de estado de la respuesta no forzada en el modelo de una sola masa.

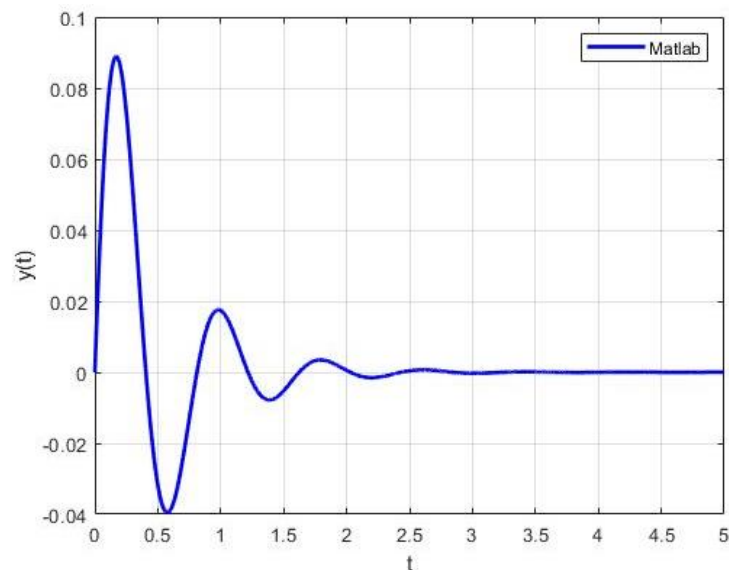


Figura 27. Respuesta no forzada del modelo de una sola masa calculada con la formulación de espacio de estado y el comando initial().

4.1.3. Solución numérica con Simulink

La respuesta no forzada del modelo de una masa puede calcularse numéricamente usando el diagrama *Simulink* de la Figura 28, que define la ecuación diferencial

$$\ddot{y} = -\frac{c}{m} \cdot \dot{y} - \frac{k}{m} \cdot y$$

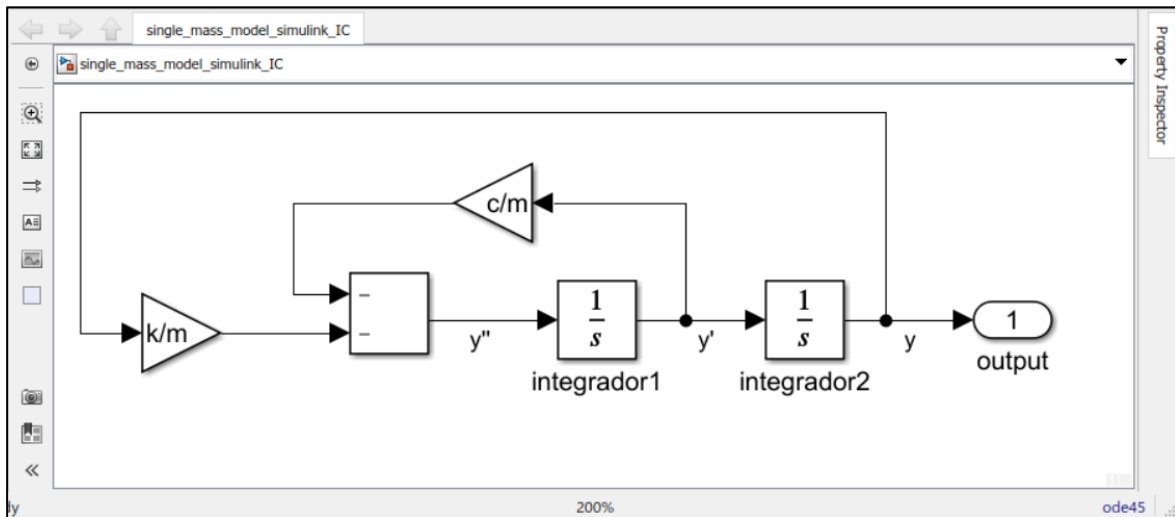


Figura 28. Diagrama Simulink para la respuesta libre del modelo de una masa.

Para especificar la condición inicial $\dot{y}(0)$, usamos la variable *dy0* en el cuadro de configuración del bloque *integrador1* (ver Figura 29). La condición inicial $y(0)$ se establece de forma análoga en el *integrador2* mediante la variable *y0*.

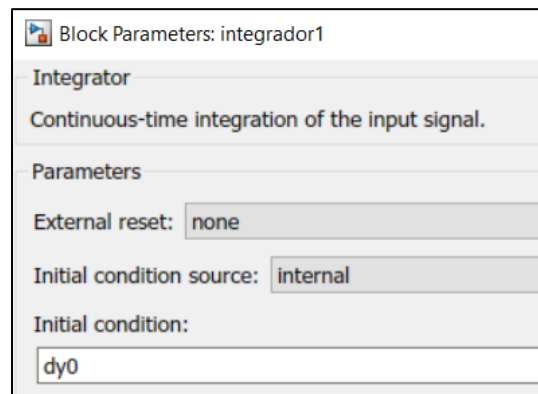


Figura 29. Condición inicial del bloque *integrador1*.

El *script Matlab* de la Figura 30 define el valor de los tiempos de simulación, asigna los valores particulares de los parámetros y las condiciones iniciales, establece algunas opciones para el ODE Solver, ejecuta la simulación, define el vector de respuesta y representa gráficamente la solución calculada. Al ejecutarse el *script*, los valores

asignados a las variables m , c , k , dy_0 , y_0 se cargan en el área de variables de *Matlab* (*workspace*) y son asignados a los parámetros de los bloques *Simulink* para realizar la simulación. El gráfico obtenido es prácticamente idéntico al mostrado en la Figura 27.

```
run_single_mass_IC_simulink.m  x +
1 - clear; clc
2 - %% Vector de tiempos
3 - stop_time = 5; sample_time = 1e-3;
4 - t_grid = 0:sample_time:stop_time;
5 - %% Parámetros y CI
6 - m = 250; c = 1000; k = 16000;
7 - dy0 = 1; y0 = 0;
8 - %% Simulación
9 - % Nombre del modelo de Simulink
10 - model_link = 'single_mass_model_simulink_IC';
11 - open_system(model_link, 'loadonly');
12 - set_param(model_link, ...
13 -           'Solver', 'rk45', ...
14 -           'RelTol', '1e-6', ...
15 -           'MaxStep', '1e-4');
16 - link_out = sim(model_link, t_grid);
17 - %% Extracción de los resultados
18 - y_link = link_out.yout;
19 - %% Gráficos
20 - plot(t_grid, y_link)
```

Figura 30. Código Matlab para simular la respuesta libre del modelo de una masa con Simulink.

4.1.4. Solución numérica con Simscape

Con *Simscape*, podemos calcular la respuesta no forzada del modelo de una masa usando el diagrama mostrado en la Figura 31 [11].

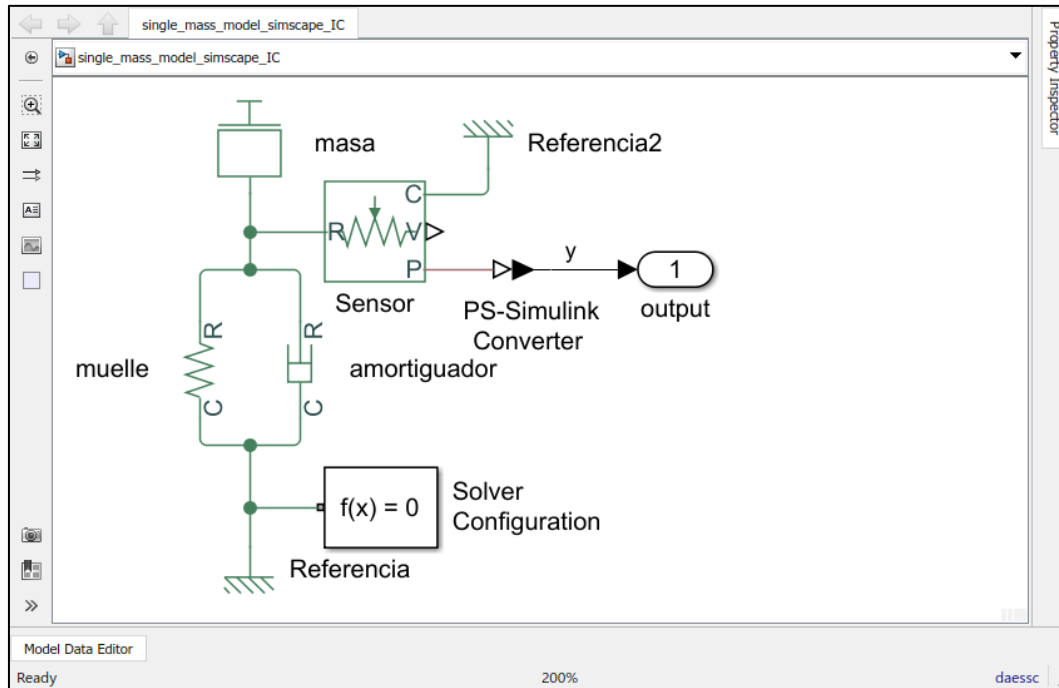


Figura 31. Diagrama Simscape de la respuesta no forzada del modelo de una sola masa.

En la configuración del elemento masa, establecemos la variable m para determinar el valor de la masa y la variable $dy0$ para la condición inicial $\dot{y}(0)$ (ver Figura 32). En las Figuras 33 y 34 se muestra la configuración de los elementos muelle y amortiguador, respectivamente.

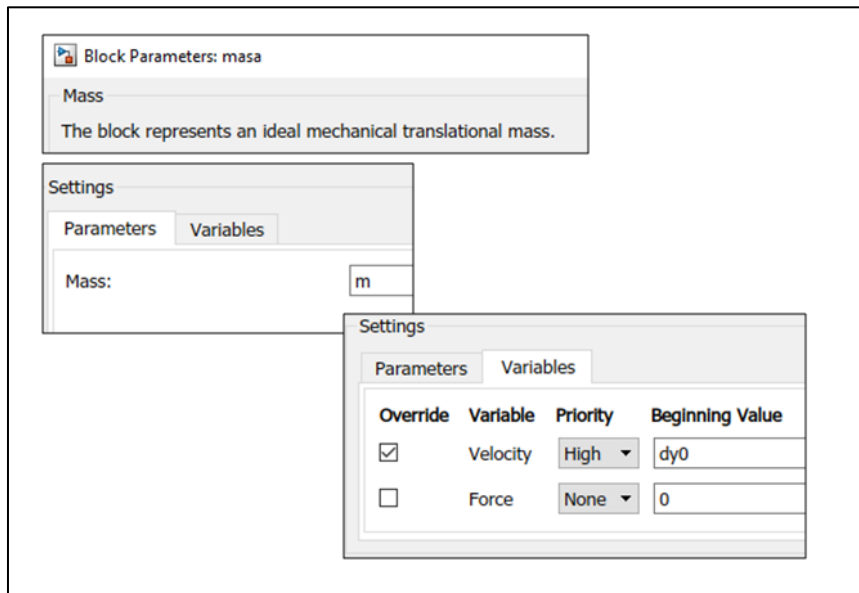


Figura 32. Configuración del elemento masa.

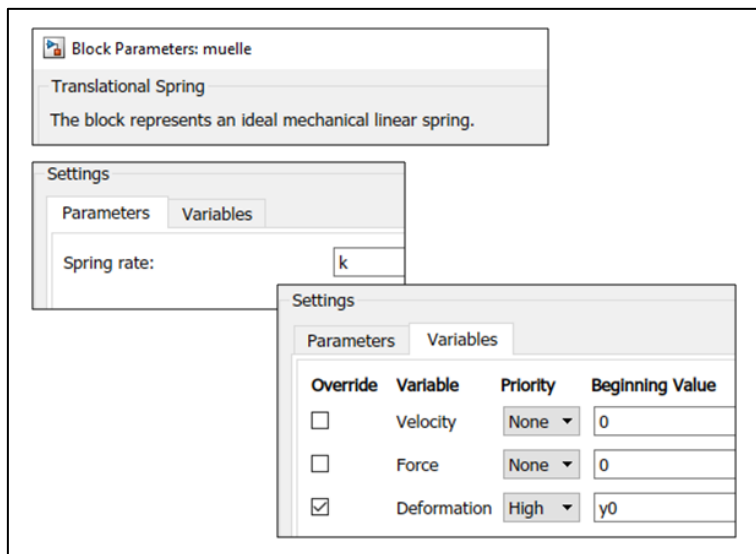


Figura 33. Configuración del elemento muelle.

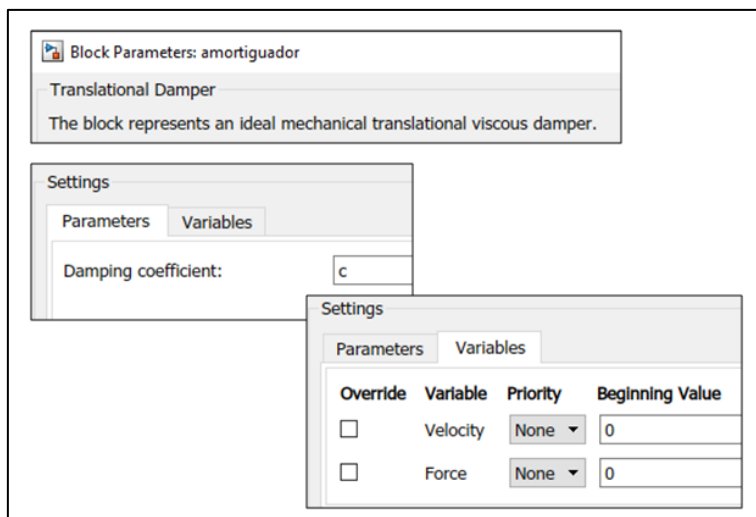


Figura 34. Configuración del elemento amortiguador.

La condición inicial $y(0)$ se establece en el elemento muelle mediante la variable y_0 . En los valores iniciales correspondientes a las variables de las condiciones iniciales especificadas por las variables dy_0 , y_0 , se fija el nivel de prioridad alto. En el resto de variables, se deja un nivel de prioridad bajo (none). En una etapa previa a la simulación, los valores iniciales con un nivel de prioridad bajo se autoajustan para que sean consistentes con los valores iniciales de alta prioridad.

El sensor mecánico traslacional de *Simscape* devuelve la posición y la velocidad relativa entre los puntos R (rod) y C (case). Como el puerto C está enlazado a un bloque de referencia y el puerto R está conectado al elemento masa, el sensor devuelve la posición relativa de la masa respecto de una referencia inercial. Antes de enlazar con el puerto de

salida *Simulink* (bloque output), la señal física de desplazamiento se transforma en una señal numérica *Simulink* mediante el bloque de conversión *PS-Simulink*. Para simular el modelo usamos el código *Matlab* de la Figura 35. Observamos que en este caso se usa el solver *daessc*, que está especialmente indicado para los sistemas de ecuaciones algebraico-diferenciales generadas por *Simscape*. El gráfico de la respuesta obtenida con *Simscape* es prácticamente idéntico al presentado en la Figura 27.

```

run_single_mass_IC_simscape.m  x  +
1 - clear; clc
2 - %% Vector de tiempos
3 - stop_time = 5; sample_time = 1e-3;
4 - t_grid = 0:sample_time:stop_time;
5 - %% Parámetros y CI
6 - m = 250; c = 1000; k = 16000;
7 - dy0 = 1; y0 = 0;
8 - %% Simulación
9 - % Nombre del modelo de Simulink
10 - model_sims = 'single_mass_model_simscape_IC';
11 - open_system(model_sims, 'loadonly');
12 - set_param(model_sims, ...
13 -         'Solver', 'daessc', ...
14 -         'RelTol', '1e-6', ...
15 -         'MaxStep', '1e-4');
16 - sims_out = sim(model_sims, t_grid);
17 - %% Extracción de los resultados
18 - y_sims = sims_out.yout;
19 - %% Gráficos
20 - plot(t_grid, y_sims)

```

Figura 35. Código Matlab para simular la respuesta libre del modelo de una masa con Simscape.

4.1.5. Comparación de los resultados obtenidos en el modelo de masa simple con respuesta no forzada

Para ver la consistencia de las distintas simulaciones, generamos un archivo CSV que contiene en columnas los datos del vector tiempo (*t_grid*), y los vectores de resultados (*y_ss*, *y_link*, *y_sims*) calculados con *Matlab*, *Simulink* y *Simscape*, respectivamente (ver Figura 36). A continuación, usando el código *Maple* de la Figura 37, importamos la matriz de resultados en *Maple* y construimos los *splines* cúbicos *mat*, *link* y *sims* correspondientes a las diferentes soluciones. Finalmente, usando el código *Maple* de la Figura 38, definimos una función que implementa la solución exacta y representamos gráficamente los errores.

```

63     %% Exportar resultados
64     % Nombre del fichero CSV
65     file_IC = 'single_mass_IC.csv';
66     % Matriz de valores a exportar
67     sol = [t_grid',y_ss,y_link,y_sims];
68     % Exportamos la matriz al fichero
69     writematrix(sol,file_IC)
    
```

Figura 36. Código Matlab para generar el archivo de resultados en formato CSV.

```

import := ImportMatrix("single_mass_IC.csv")

import := 
$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0.001 & 0.0009979920187 & 0.0009979920187 & 0.0009979888927 \\ 0.002 & 0.001991936299 & 0.001991936299 & 0.001991933059 \\ 0.003 & 0.002981785515 & 0.002981785515 & 0.002981782163 \\ 0.004 & 0.003967492789 & 0.003967492789 & 0.003967489328 \\ 0.005 & 0.004949011699 & 0.004949011699 & 0.004949008129 \\ 0.006 & 0.005926296273 & 0.005926296273 & 0.005926292597 \\ 0.007 & 0.006899300992 & 0.006899300992 & 0.006899297212 \\ 0.008 & 0.007867980796 & 0.007867980796 & 0.007867976913 \\ 0.009 & 0.008832291077 & 0.008832291077 & 0.008832287093 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{7}$$

5001 × 4 Matrix

mat := Interpolation:-CubicInterpolation(import( ..., 1), import( ..., 2))

mat := 
$$\left[ \begin{array}{l} \text{a cubic interpolation object} \\ \text{with 5001 points in 1-D} \end{array} \right] \tag{8}$$


link := Interpolation:-CubicInterpolation(import( ..., 1), import( ..., 3))

link := 
$$\left[ \begin{array}{l} \text{a cubic interpolation object} \\ \text{with 5001 points in 1-D} \end{array} \right] \tag{9}$$


sims := Interpolation:-CubicInterpolation(import( ..., 1), import( ..., 4))

sims := 
$$\left[ \begin{array}{l} \text{a cubic interpolation object} \\ \text{with 5001 points in 1-D} \end{array} \right] \tag{10}$$

    
```

Figura 37. Código Maple para importar las respuestas no forzadas del modelo de una sola masa y generar los splines cúbicos de aproximación.

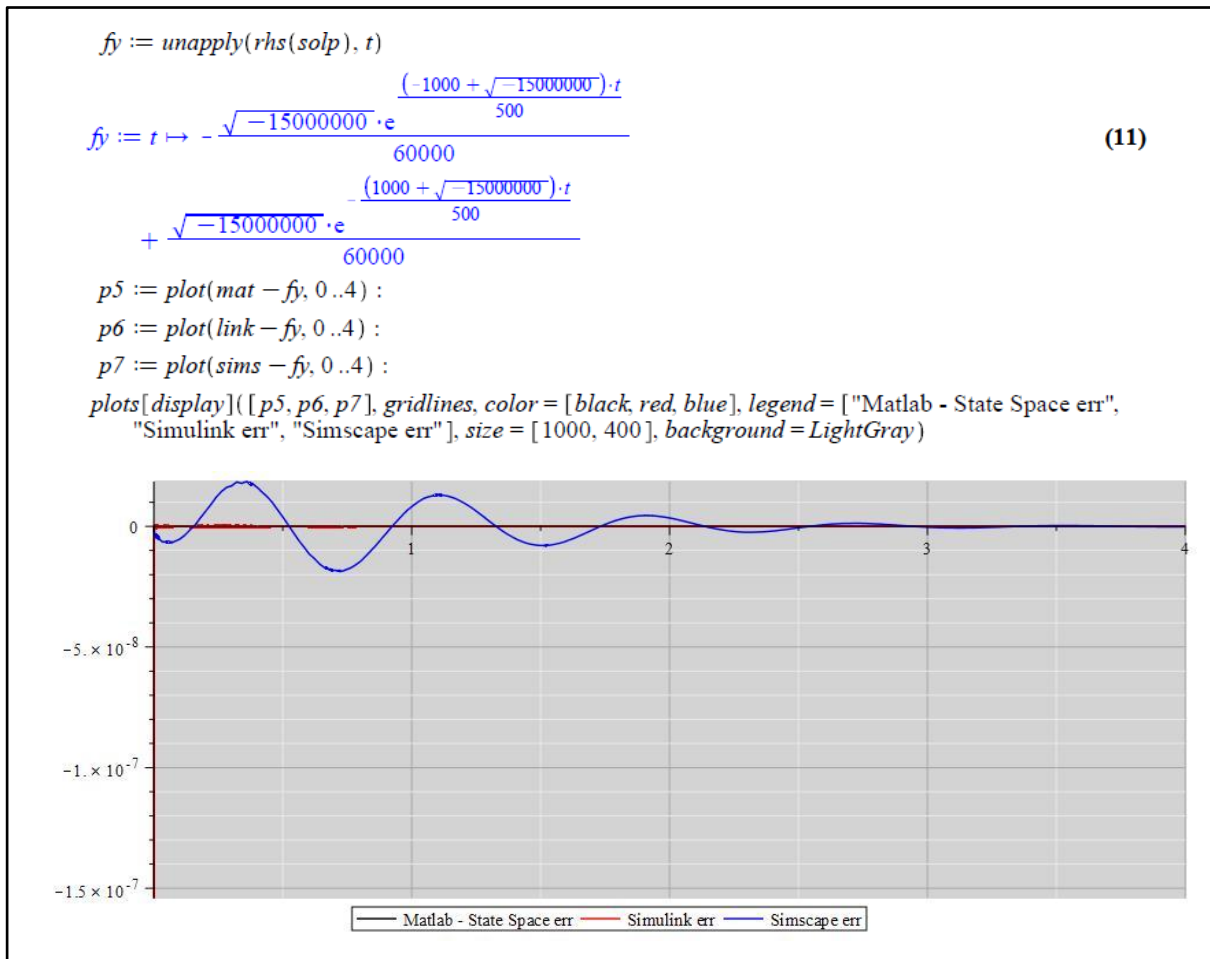


Figura 38. Error en las respuestas no forzadas del modelo de masa simple.

Observamos que en todos los casos la magnitud del error es inferior a $1.6 \cdot 10^{-7}$, lo que confirma la consistencia de las soluciones obtenidas. También se aprecia el distinto comportamiento de los errores, que puede estar causado por el empleo de distintos solvers y sus configuraciones.

4.2. Respuesta no forzada del modelo de doble masa

4.2.1. Solución numérica con Maple

Considerando la Ecuación (4), la respuesta libre del modelo de doble masa correspondiente a una velocidad inicial de 1 m/s en el cuerpo del vehículo puede describirse mediante el siguiente problema de valor inicial:

$$\begin{cases} m \cdot \ddot{y} + c \cdot (\dot{y} - \dot{y}_a) + k \cdot (y - y_a) = 0 \\ m_r \cdot \ddot{y}_a + c_r \cdot \dot{y}_a + k_r \cdot y_a + c \cdot (\dot{y}_a - \dot{y}) + k \cdot (y_a - y) = 0 \\ y(0) = 0, \dot{y}(0) = 1 \\ y_a(0) = 0, \dot{y}_a(0) = 0 \end{cases} \quad (17)$$

Este problema puede resolverse numéricamente con *Maple* usando el código mostrado en las Figuras 39 y 40. La resolución numérica con *Maple* nos proporciona procedimientos numéricos que permiten aproximar el valor de las soluciones (ver líneas 10 y 11 en la Figura 40. La representación gráfica del desplazamiento vertical del cuerpo del vehículo $y(t)$ y del sistema eje-rueda $y_a(t)$ se presenta en la Figura 41.

```
restart
Definición del sistema de EDO del modelo
De1 := m·ÿ + c·(ẏ - ẏa) + k·(y(t) - ya(t)) = 0
De1 := m ( d²/dt² y(t) ) + c ( d/dt y(t) - d/dt ya(t) ) + k(y(t) - ya(t)) = 0 (1)
De2 := mr·ÿa + c·(ẏa - ẏ) + k·(ya(t) - y(t)) + kr·ya(t) = 0
De2 := mr ( d²/dt² ya(t) ) + c ( d/dt ya(t) - d/dt y(t) ) + k(ya(t) - y(t)) + kr·ya(t) = 0 (2)
Condiciones iniciales
IC := { y(0) = 0, D(y)(0) = 1, ya(0) = 0, D(ya)(0) = 0 }
IC := { y(0) = 0, ya(0) = 0, D(y)(0) = 1, D(ya)(0) = 0 } (3)
Parámetros del sistema
param := { m = 250, k = 16000, c = 1000, mr = 45, kr = 160000 }
param := { c = 1000, k = 16000, kr = 160000, m = 250, mr = 45 } (4)
Definición de las ecuaciones particulares
De1p := subs(param, De1)
De1p := 250 d²/dt² y(t) + 1000 d/dt y(t) - 1000 d/dt ya(t) + 16000y(t) - 16000ya(t) = 0 (5)
De2p := subs(param, De2)
De2p := 45 d²/dt² ya(t) + 1000 d/dt ya(t) - 1000 d/dt y(t) + 176000ya(t) - 16000y(t) = 0 (6)
```

Figura 39. Definición con Maple del problema de valor inicial correspondiente a la respuesta libre del modelo de doble masa con los valores de parámetros dados en la Tabla 1.

```

Solución del sistema de EDOs de forma numerica
sol := dsolve( {De1p, De2p} union IC, {y(t), ya(t)}, numeric, maxstep=1e-4, relerr=1e-8, output
= operator, maxfim=0)
                                [Length of output exceeds limit of 100000] (7)

Extracción de los resultados
sol_y := subs(sol, y)
                                sol_y := proc(t) ... end proc (8)

sol_ya := subs(sol, ya)
                                sol_ya := proc(t) ... end proc (9)

Evaluación de y(t), ya(t)
sol_y(0.25)
                                0.0850734060557423 (10)

sol_ya(0.25)
                                0.00634592267869082 (11)
    
```

Figura 40. Resolución numérica con Maple del problema de valor inicial.

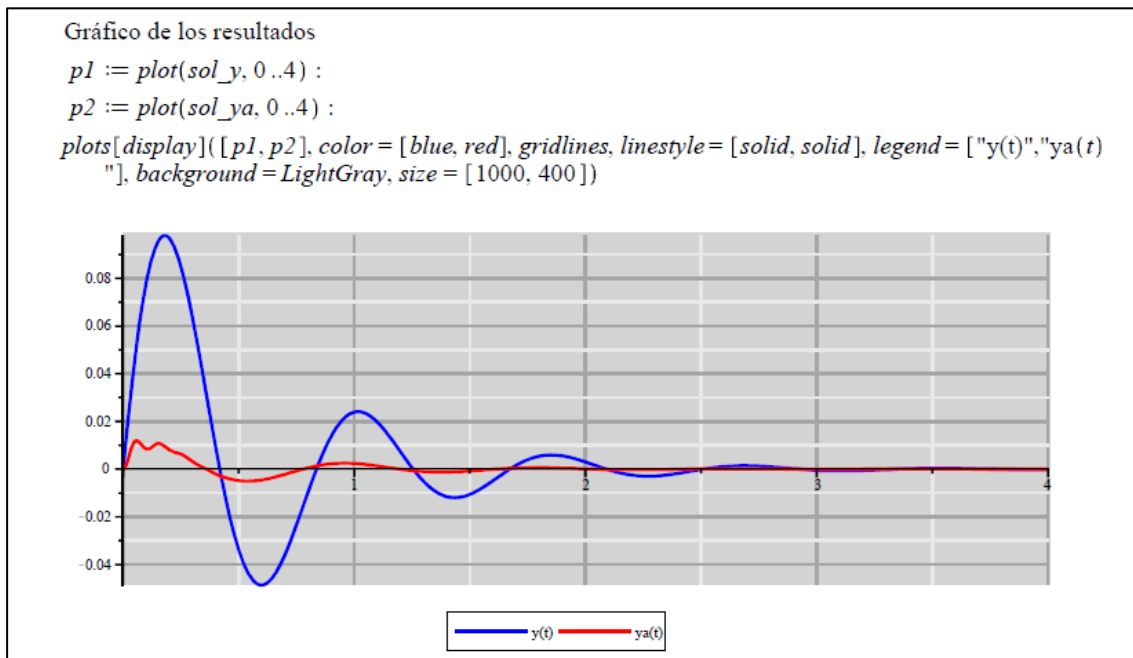


Figura 41. Representación gráfica de la solución numérica obtenida con Maple para la respuesta no forzada del modelo de doble masa.

4.2.2. Resolución numérica con Matlab, Simulink y Simscape

Para resolver el problema de valor inicial de la Ecuación 17 con el comando *initial()* de *Matlab*, es preciso escribir el problema en forma de espacio de estado

$$\begin{cases} \dot{X}(t) = A \cdot X(t) + B \cdot U(t) \\ Z(t) = C \cdot X(t) + D \cdot U(t) \\ X(0) = X_0 \end{cases}$$

Considerando las variables de estado $x_1(t) = y(t)$, $x_2(t) = \dot{y}(t)$, $x_3(t) = y_a(t)$, $x_4(t) = \dot{y}_a(t)$, y tomando como variables de salida los desplazamientos verticales $z_1(t) = y(t)$, $z_2 = y_a(t)$, se obtienen las matrices

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k}{m} & -\frac{c}{m} & \frac{k}{m} & \frac{c}{m} \\ 0 & 0 & 0 & 1 \\ \frac{k}{m_r} & \frac{c}{m_r} & -\left(\frac{k_r + k}{m_r}\right) & -\frac{c}{m_r} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

y el vector de condiciones iniciales $X_0 = [0 \ 1 \ 0 \ 0]^T$. La resolución numérica del problema de valor inicial con *Simulink* puede realizarse con el diagrama de la Figura 42.

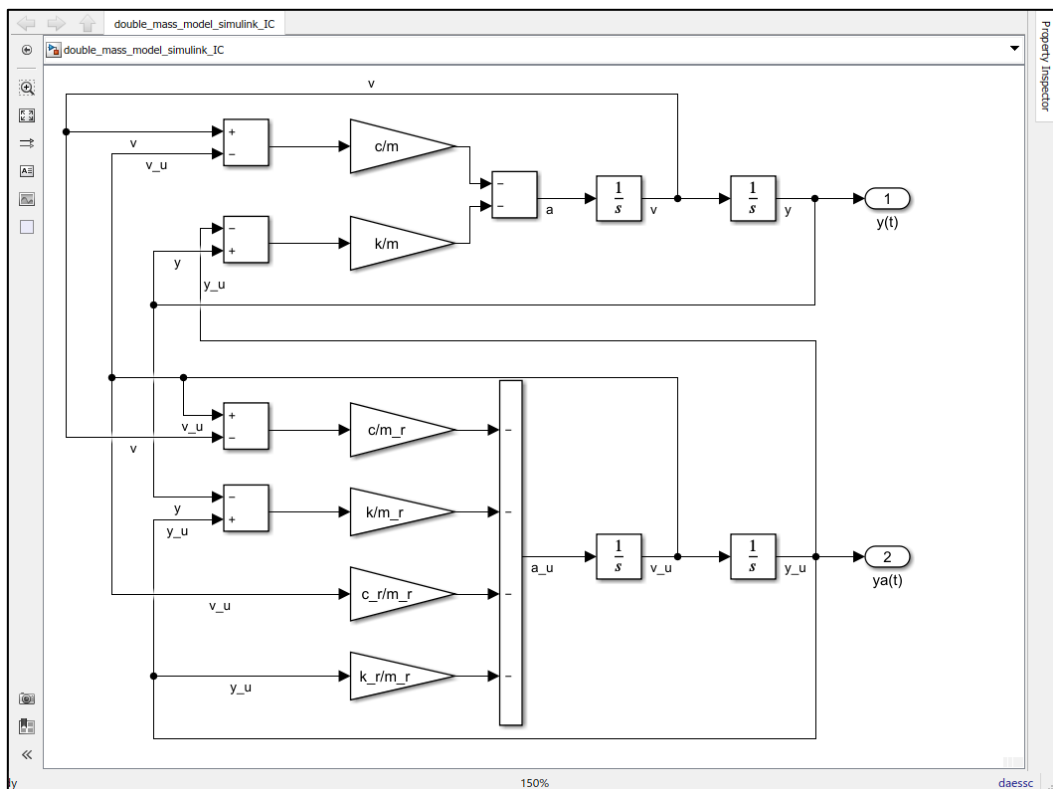


Figura 42. Diagrama Simulink para calcular la respuesta no forzada del modelo de doble masa.

La solución con *Simscape* puede obtenerse usando el diagrama de la Figura 43 y el *script Matlab* de la Figura 44. La representación gráfica de las respuestas obtenidas con *Simscape* se muestran en la Figura 45.

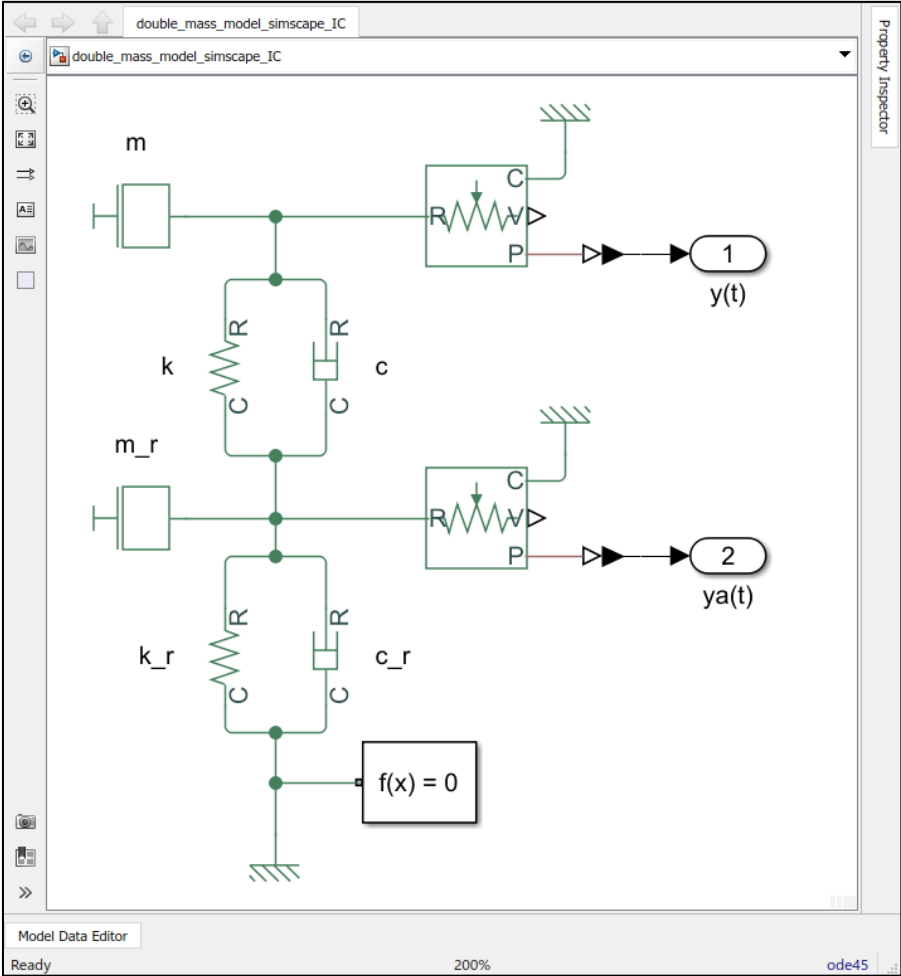


Figura 43. Diagrama Simscape para calcular la respuesta no forzada del modelo de masa doble.

```

run_double_mass_IC_simscape.m x +
1  clear; clc
2  %% Vector de tiempos
3  stop_time = 5; sample_time = 1e-3;
4  t_grid = 0:sample_time:stop_time;
5  %% Parámetros y CI
6  m = 250; c = 1000; k = 16000;
7  m_r = 50; c_r = 0; k_r = 160000;
8  dy0 = 1; y0 = 0;
9  dya0 = 0; ya0 = 0;
10 %% Simulación
11 % Nombre del modelo de Simscape
12 model_sims = 'double_mass_model_simscape_IC';
13 open_system(model_sims,'loadonly');
14 set_param(model_sims,...
15           'Solver','rk45',...
16           'RelTol','1e-6',...
17           'MaxStep','1e-4');
18 sims_out = sim(model_sims,t_grid);
19 %% Extracción de los resultados
20 y_sims = sims_out.yout(:,1);
21 ya_sims = sims_out.yout(:,2);
22 %% Gráficos
23 plot(t_grid,[y_sims,ya_sims])
    
```

Figura 44. Script Matlab para simular la respuesta libre del modelo de doble masa con Simscape.

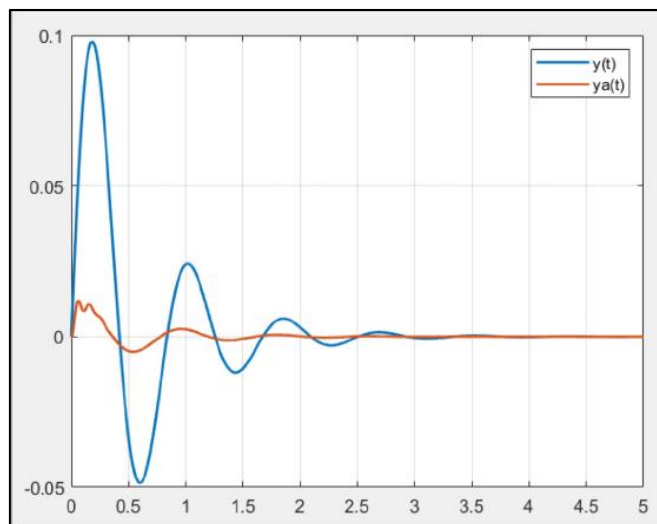


Figura 45. Respuesta libre del modelo de doble masa calculado con Simscape.

Conviene observar que para abordar la resolución del modelo de doble masa con *Maple*, *Matlab* y *Simulink*, es preciso reformular completamente el problema de valor inicial. Además, el aumento de un grado de libertad en el nuevo modelo incrementa de forma notable la complejidad del proceso de solución.

En contraste, la resolución con *Simscape* está basada en elementos físicos y no requiere la manipulación de las ecuaciones diferenciales del problema. En este caso, para realizar la extensión al problema de doble masa, solo necesitamos añadir los nuevos elementos al diagrama *Simscape*, ajustar la configuración de los bloques y modificar ligeramente el *script* de ejecución.

Finalmente, observamos que los resultados obtenidos con *Simscape* son similares a los calculados con *Maple*, *Matlab* y *Simulink*. Así, por ejemplo, si procedemos como en la sección anterior y tomamos como referencia el resultado proporcionado por *Maple*, los errores correspondientes al desplazamiento vertical $y(t)$ calculado por *Simscape*, *Simulink* y *Matlab* tienen una magnitud inferior a $1.5 \cdot 10^{-7}$ (ver Figura 46).

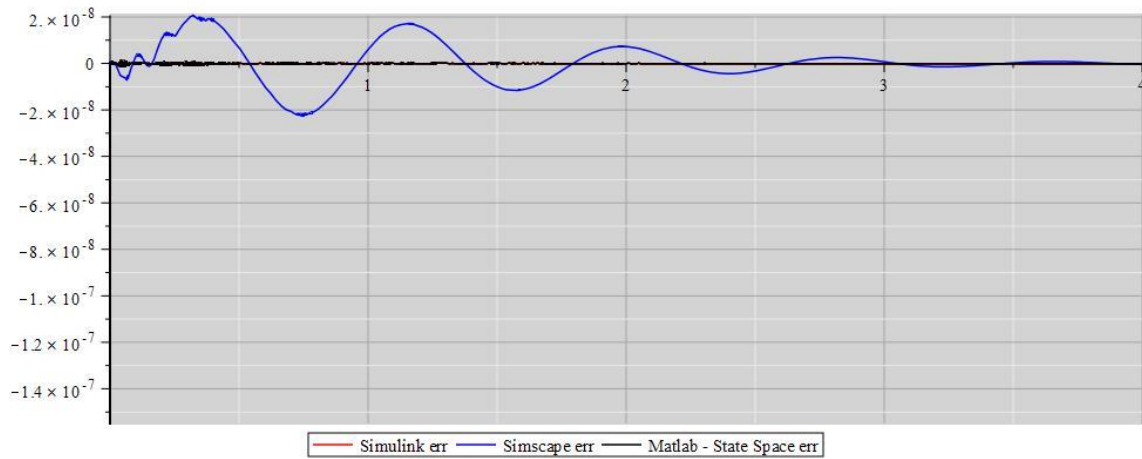


Figura 46. Error en las respuestas no forzadas del modelo de masa doble tomando como referencia el resultado proporcionado por Maple.

5. RESPUESTAS FORZADAS

En este capítulo usamos *Simscape* para simular la respuesta forzada del modelo de cuarto de vehículo con doble masa y el modelo de medio vehículo con conductor. La excitación externa se genera mediante un modelo sinusoidal de bache. Para introducir la excitación externa se programa un nuevo bloque *Simscape* que produce el desplazamiento en el dominio mecánico traslacional. En el modelo de medio vehículo, se programa otro nuevo bloque *Simscape* que permite incorporar el efecto dinámico de los desplazamientos verticales y la rotación del cuerpo del vehículo. Para verificar el buen funcionamiento de los nuevos bloques, se contrastan las respuestas *Simscape* con los resultados obtenidos en *Maple* a partir de las ecuaciones diferenciales. El código completo de la resolución *Maple* se incluye en el Anexo.

5.1. Excitación externa

Para generar las respuestas forzadas de este capítulo, consideramos un modelo de bache sinusoidal como el mostrado esquemáticamente en la Figura 47.

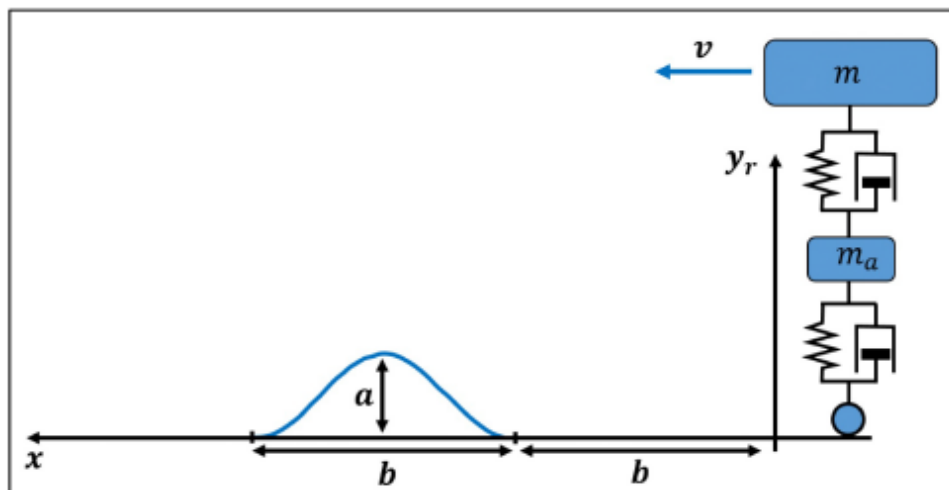


Figura 47. Perfil geométrico del bache usado como excitación externa.

El perfil geométrico del bache puede describirse por la función:

$$y_r(x) = \begin{cases} 0 & \text{si } x < b \\ \frac{a}{2} \cdot \left(1 - \cos\left(\frac{2 \cdot \pi}{b} \cdot x\right)\right) & \text{si } b < x < 2 \cdot b \\ 0 & \text{si } x > 2 \cdot b \end{cases} \quad (18)$$

Cuando el vehículo avanza con una velocidad constante v en la dirección positiva del eje x , la elevación de la carretera que actúa sobre el vehículo es

$$y_r(t) = \begin{cases} 0 & \text{si } t < b/v \\ \frac{a}{2} \cdot \left(1 - \cos\left(\frac{2 \cdot \pi}{b} \cdot v \cdot t\right)\right) & \text{si } b/v < t < 2 \cdot b/v \\ 0 & \text{si } t > 2 \cdot b/v \end{cases} \quad (19)$$

donde:

a es la altura del bache (en m),

b es la longitud del bache y la distancia del origen a su inicio (en m),

v es la velocidad de avance del vehículo (en m/s).

Si tomamos el tiempo en segundos y los valores particulares $a = 0.1 \text{ m}$, $b = 0.3 \text{ m}$ y $v = 11.11 \text{ m/s}$ (40 km/h), obtenemos

$$y_r(t) = \begin{cases} 0 & \text{si } t < 0.027 \\ 0.05 - 0.05 \cdot \cos(232.717 \cdot t) & \text{si } 0.027 < t < 0.054 \\ 0 & \text{si } t > 0.054 \end{cases}$$

5.2. Respuesta forzada del modelo de doble masa

La respuesta forzada del modelo de doble masa puede calcularse con el diagrama *Simscape* de la Figura 48. El nuevo diagrama es una versión modificada del diagrama *Simscape* usado para simular la respuesta libre. Para generar la excitación externa, se ha usado un bloque *Simulink* que permite definir funciones matemáticas mediante *scripts Matlab* (ver Figura 49). El empleo de este bloque requiere un bloque “reloj”, que proporciona acceso al tiempo de simulación, y un elemento de conversión *Simulink-PS* para enlazar con los elementos *Simscape*.

Como la librería *Simscape* solo contiene generadores de fuerza y velocidad (ver Figura 50), se ha programado un generador de desplazamiento en el dominio mecánico traslacional mediante el código presentado en la Figura 51.

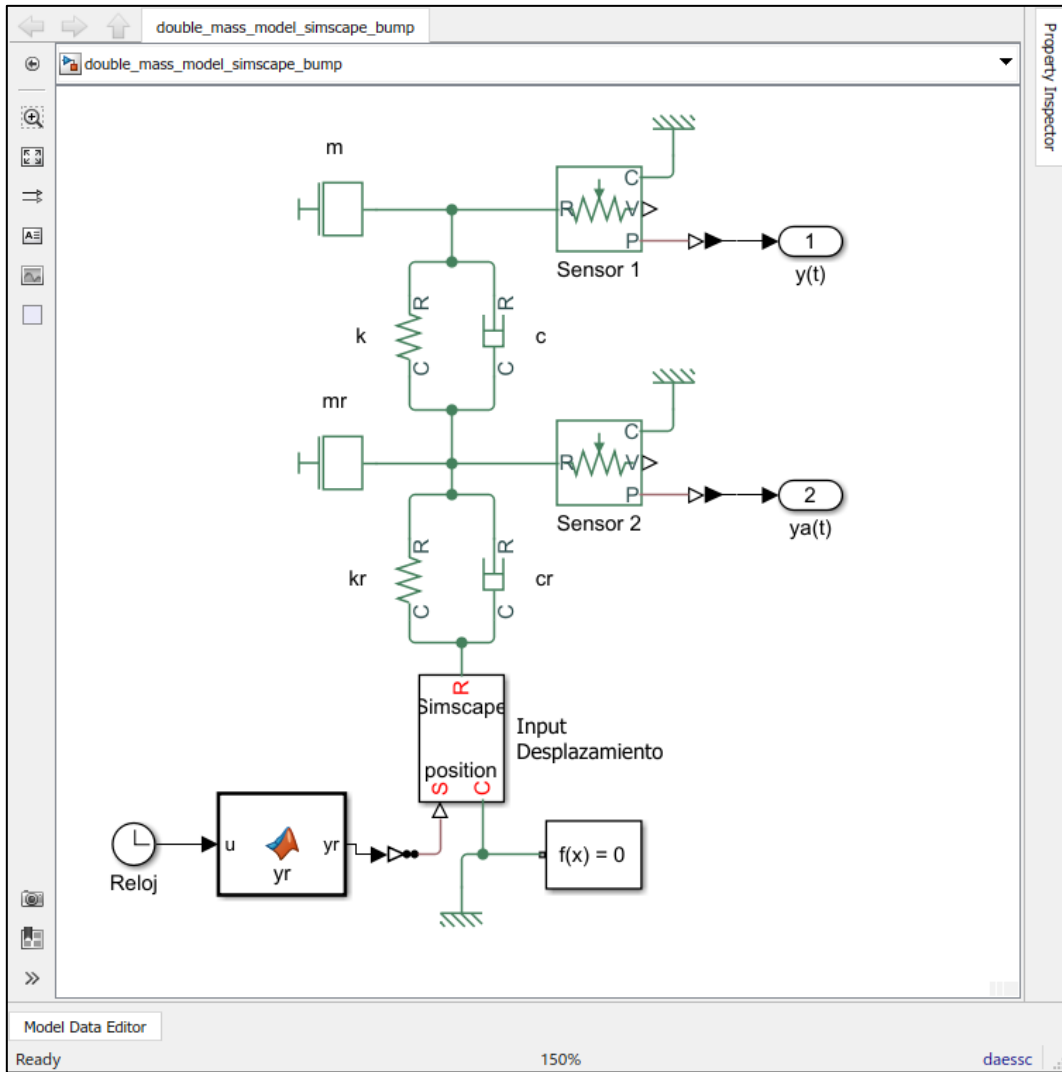


Figura 48. Diagrama Simscape para simular la respuesta forzada del modelo de doble masa.

```

1  function y = yr(t)
2  if t < 0.027
3      y = 0;
4  elseif t <= 0.054
5      y = 0.05*(1-cos(232.71057*t));
6  else
7      y = 0;
8  end
    
```

Figura 49. Función Matlab para la definición del bache usada en el modelo de doble masa.

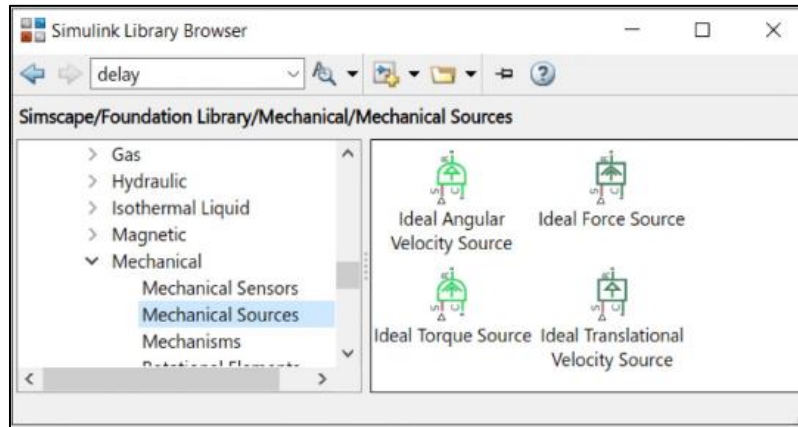


Figura 50. Elementos generadores en los dominios mecánicos de Simscape.

```

1  component position
2  % Fuente de posición lineal ideal
3  inputs
4      S = { 0, 'm' }; % S:bottom
5  end
6  nodes
7      C = foundation.mechanical.translational.translational; % C:bottom
8      R = foundation.mechanical.translational.translational; % R:top
9  end
10 variables(Access = protected)
11     v = { 0, 'm/s' }; % Velocidad
12     x = { 0, 'm' }; % Posición
13     f = { 0, 'N' }; % Fuerza
14 end
15 branches
16     f : R.f -> C.f;
17 end
18 equations
19     v == R.v - C.v;
20     x.der == v;
21     x == S;
22 end
23 end
    
```

Figura 51. Script Simscape para definir un generador de desplazamiento en el dominio mecánico traslacional.

La simulación se realiza desde el *script* de *Matlab* de la Figura 52, donde pueden observarse las condiciones iniciales nulas (línea 8) y la generación del archivo CSV para la verificación del resultado con *Maple* (líneas 27 - 29). La representación gráfica de las respuestas obtenidas se muestra en la Figura 53.

```

run_double_mass_bump.m
1 clear; clc
2 %% Vector de tiempos
3 stop_time = 3; data_points = 3000;
4 t_grid = linspace(0,stop_time,data_points)';
5 %% Parametros y CI
6 m = 250; c = 1000; k = 16000;
7 m_r = 45; c_r = 0; k_r = 160000;
8 dy0 = 0; y0 = 0; dya0 = 0; ya0 = 0;
9 %% Simulación
10 % Nombre del modelo
11 model_name = 'double_mass_model_simscape_bump';
12 open_system(model_name,'loadonly');
13 % Configuración del modelo
14 set_param(model_name, ...
15     'StopTime',string(stop_time),...
16     'Solver','daessc',...
17     'MaxStep','1e-4',...
18     'RelTol','1e-6');
19 sol_bump = sim(model_name,t_grid);
20 %% Extracción de los resultados
21 y_sims = sol_bump.yout(:,1);
22 ya_sims = sol_bump.yout(:,2);
23 %% Representación gráfica
24 plot(t_grid,[y_sims,ya_sims],'LineWidth',1.2)
25 grid, legend("y(t)","ya(t)")
26 %% Archivo CSV
27 file_bump = '../double_mass_bump_simscape.csv';
28 sol = [t_grid,y_sims,ya_sims];
29 writematrix(sol,file_bump);

```

Figura 52. Script Matlab para realizar la simulación de la respuesta forzada del modelo de doble masa.

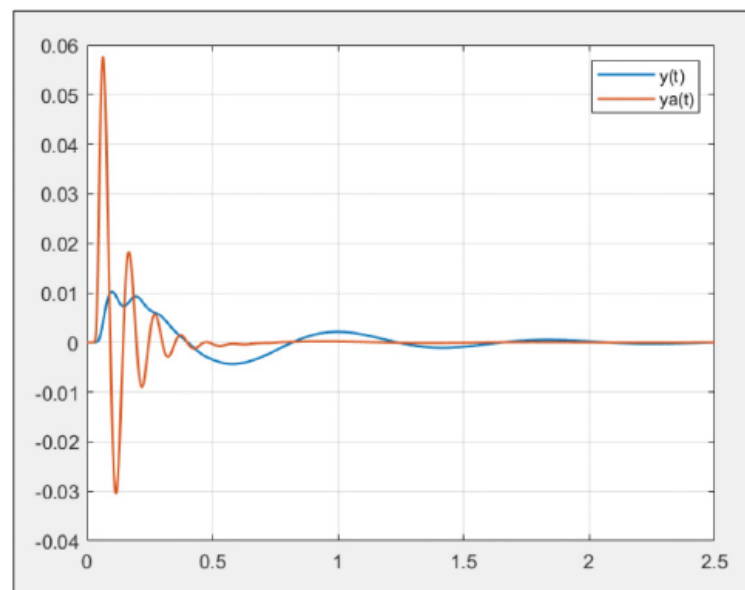


Figura 53. Representación gráfica de la respuesta forzada del modelo de doble masa calculado con Simscape.

5.3. Respuesta forzada del modelo de medio vehículo con conductor

En esta sección usamos *Simscape* para simular la respuesta forzada del modelo de medio vehículo con conductor. Los valores particulares de los parámetros usados en la simulación son similares a los presentados en [10], y están recogidos en la Tabla 2.

Parámetro	Definición	Valor	Unidades
m	Masa del cuerpo del vehículo	1450	kg
m_d	Masa asiento-conductor	100	kg
m_{r1}, m_{r2}	Masa eje-rueda	45	kg
J	Momento de inercia de cuerpo del vehículo	2460	$kg \cdot m^2$
k_1, k_2	Coefficiente de elasticidad de la suspensión	16000	N/m
c_1, c_2	Coefficiente de amortiguación de la suspensión	1000	$N \cdot s/m$
k_d	Coefficiente de elasticidad del asiento	16000	N/m
c_d	Coefficiente de amortiguación del asiento	1000	$N/m \cdot s$
k_{r1}, k_{r2}	Coefficiente de elasticidad de los neumáticos	160000	N/m
c_{r1}, c_{r2}	Coefficiente de amortiguación de los neumáticos	0	$N \cdot s/m$
a_1	Distancia del eje delantero al CG	1.2	m
a_2	Distancia del eje trasero al CG	1.6	m
a_3	Distancia del asiento al CG	0.7	m

Tabla 2. Valores de los parámetros usados en la simulación de la respuesta forzada del modelo de medio vehículo.

Como excitación externa del eje delantero, se toma el modelo de bache discutido en la sección anterior

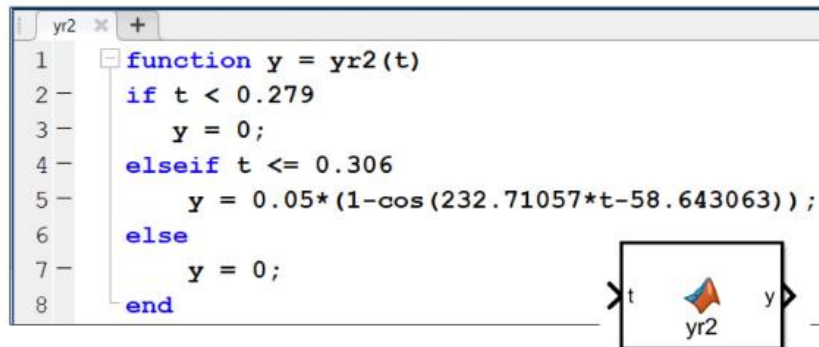
$$y_{r1}(t) = \begin{cases} 0 & \text{si } t < 0.027 \\ 0.05 - 0.05 \cdot \cos(232.717 \cdot t) & \text{si } 0.027 < t < 0.054 \\ 0 & \text{si } t > 0.054 \end{cases}$$

Si el vehículo avanza con velocidad constante, la excitación externa que actúa en el eje posterior puede escribirse en la forma $y_{r2}(t) = y_{r1}(t - \delta)$, donde $\delta = (a_1 + a_2)/v$ es el

tiempo necesario para recorrer la distancia entre ejes. En particular, para una velocidad $v = 11.11 \text{ m/s}$, el valor del retardo es $\delta = 0.252 \text{ s}$ y obtenemos

$$y_{r2}(t) = \begin{cases} 0 & \text{si } t < 0.279 \\ 0.05 - 0.05 \cdot \cos(232.717 \cdot t - 58.643) & \text{si } 0.279 < t < 0.306 \\ 0 & \text{si } t > 0.306 \end{cases}$$

Esta función puede implementarse mediante el bloque *Simulink* de la Figura 54.



```

1 function y = yr2(t)
2   if t < 0.279
3     y = 0;
4   elseif t <= 0.306
5     y = 0.05*(1-cos(232.71057*t-58.643063));
6   else
7     y = 0;
8   end
  
```

Figura 54. Bloque Simulink para implementar la excitación externa en el eje posterior.

El diagrama *Simscape* construido para esta simulación se muestra en la Figura 55. En este caso, la principal novedad es el elemento *Simscape half_car_driver* que permite modelar el desplazamiento vertical y la rotación del cuerpo del vehículo. El nuevo bloque está definido mediante el *script Simscape* de la Figura 57. El elemento define la distancia del centro de gravedad a los ejes y al asiento del conductor como parámetros configurables e incorpora cuatro puertos del dominio mecánico traslacional y un puerto del dominio mecánico rotacional (ver Figura 56). Tal como puede verse en el diagrama de la Figura 55, los puertos $y1$, $y2$ permiten enlazar con los sistemas de suspensión, el puerto yd facilita la adición del sistema asiento-conductor, y los puertos yCG y th incorporan la masa y el momento de inercia del cuerpo del vehículo, respectivamente. Las variables de salida de la simulación son la aceleración vertical del conductor ($ad(t)$ en el puerto 5), el desplazamiento del centro de gravedad ($y(t)$ en el puerto 1), el ángulo de cabeceo del vehículo ($th(t)$ en el puerto 2) y las elongaciones de las suspensiones delantera y trasera ($r1(t)$ en el puerto 3 y $r2(t)$ en el puerto 4, respectivamente). Las variables $y(t)$, $th(t)$, $r1(t)$ y $r2(t)$ pueden obtenerse con los sensores de desplazamiento incluidos en las librerías de *Simscape* (ver Figura 16). Para la variable $ad(t)$ se ha diseñado un sensor de aceleración traslacional usado en el *script Simscape* de la Figura 17. La ejecución de la simulación se realiza con el código *Matlab* de la Figura 58 y los gráficos de respuesta se obtienen con el código de la Figura 59.

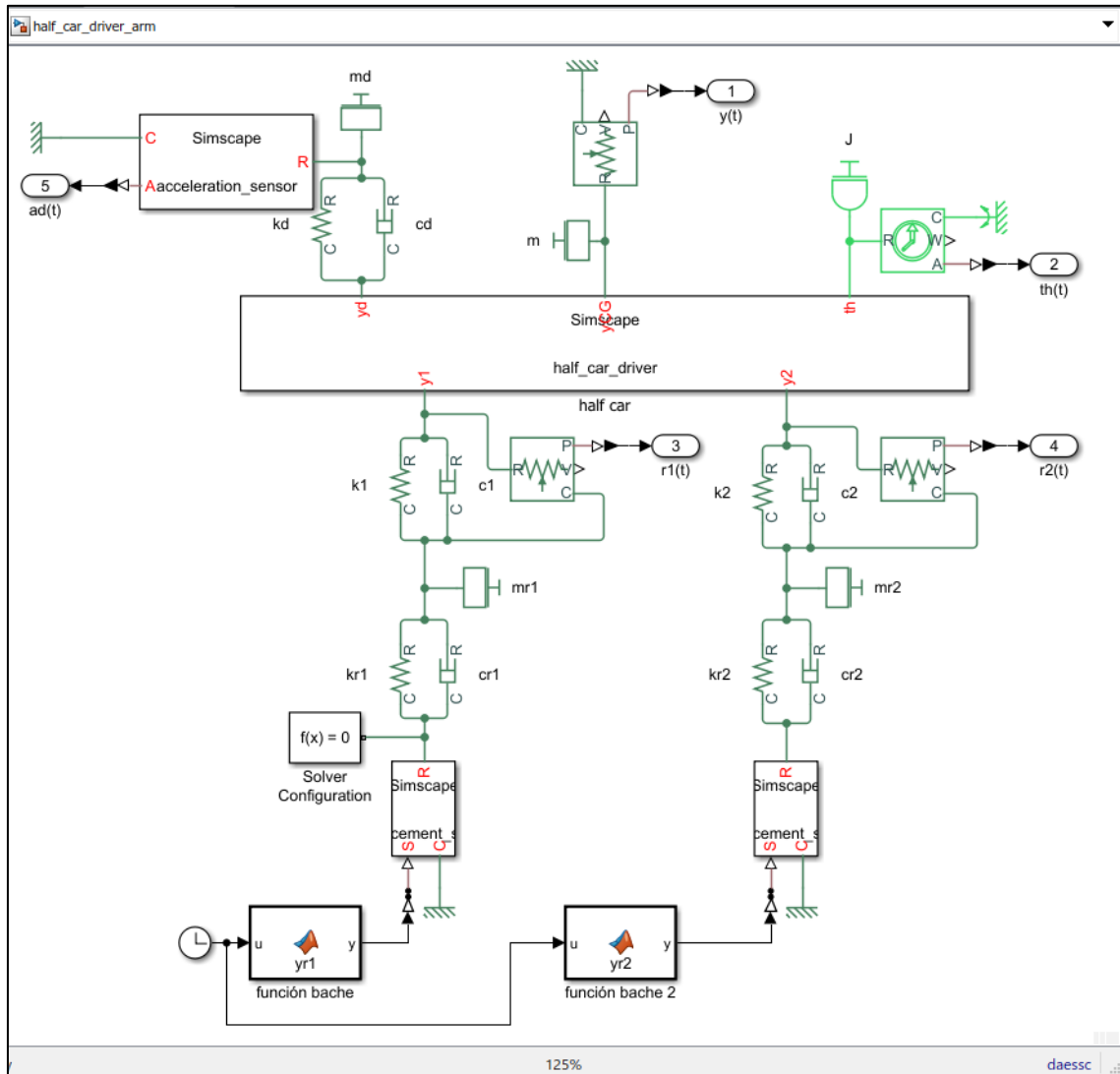


Figura 55. Diagrama Simscape para simular la respuesta forzada del modelo de medio vehículo.

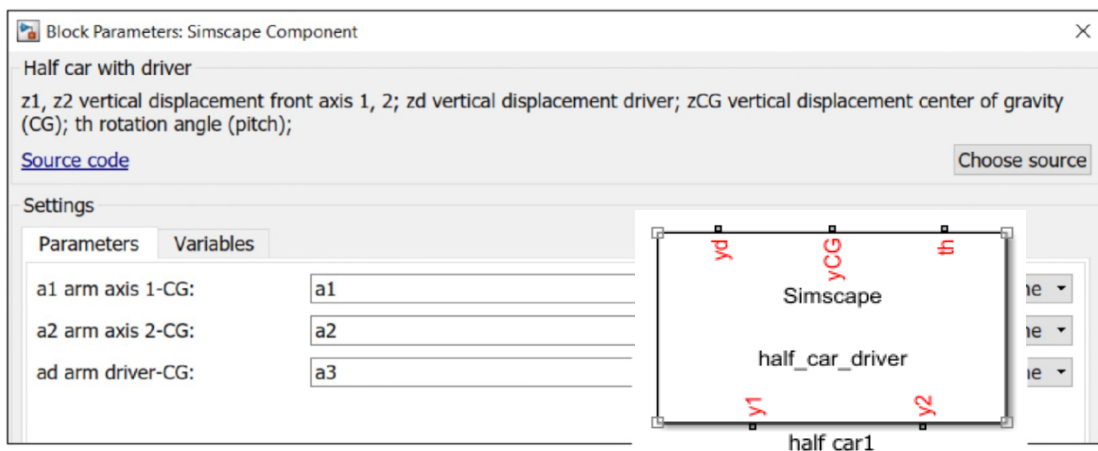


Figura 56. Cuadro de configuración y bloque Simscape del elemento half_car_driver.

```

1  component half_car_driver
2  % Half car with driver
3  % y1, y2 desplazamiento vertical del eje trasero
4  % yd desplazamiento vertical del conductor
5  % yCG desplazamiento vertical del centro de gravedad
6  % th rotación
7  nodes
8      y1 = foundation.mechanical.translational.translational; % y1:left
9      y2 = foundation.mechanical.translational.translational; % y2:left
10     yd = foundation.mechanical.translational.translational; % yd:right
11     y  = foundation.mechanical.translational.translational; % yCG:right
12     th = foundation.mechanical.rotational.rotational;      % th:right
13 end
14 parameters
15     a1 = { 1.4, 'm' }; % a1 eje delantero - CG
16     a2 = { 1.4, 'm' }; % a2 eje trasero - CG
17     ad = { 0.7, 'm' }; % ad conductor - CG
18 end
19 variables
20     f_y1 = { 0, 'N' }; % fuerza eje delantero
21     f_y2 = { 0, 'N' }; % fuerza eje trasero
22     f_y  = { 0, 'N' }; % fuerza centro de gravedad
23     f_yd = { 0, 'N' }; % fuerza al conductor
24     t    = { 0, 'N*m' }; % par en el centro de gravedad
25 end
26 branches
27     f_y1 : y1.f -> *;
28     f_y2 : y2.f -> *;
29     f_y  : y.f -> *;
30     f_yd : yd.f -> *;
31     t    : th.t -> *;
32 end
33 equations
34     assert(a1 > 0)
35     assert(a2 > 0)
36     f_y1 + f_y2 + f_y + f_yd == 0; % fuerzas del chasis
37     t == ad*f_yd + a1*f_y1 - a2*f_y2; % momentos del chasis
38     yd.v == y.v - ad * th.w; % velocidad en el conductor
39     y1.v == y.v - a1 * th.w; % velocidad en el eje delantero
40     y2.v == y.v + a2 * th.w; % velocidad en el eje trasero
41 end
42 end

```

Figura 57. Script Simscape para definir el elemento *half_car_driver*.

```

run_half_car_driver_arm.m  ×  +
2   %% Vector de tiempos
3   stop_time = 5; n_points = 5000;
4   t_grid = linspace(0,stop_time,n_points)';
5   %% Parámetros
6   md = 100; cd = 1000; kd = 16000;
7   m = 1460; J = 2460;
8   a1 = 1.2; a2 = 1.6; a3 = 0.7;
9   c1 = 1000; k1 = 16000; c2 = 1000; k2 = 16000;
10  ma1 = 45; cr1 = 0; kr1 = 160000;
11  ma2 = 45; cr2 = 0; kr2 = 160000;
12  %% Simulación
13  % Nombre del modelo
14  model_name = 'half_car_driver_arm';
15  open_system(model_name,'loadonly');
16  % Configuración del modelo
17  set_param(model_name,...
18           'Solver','daessc',...
19           'RelTol','1e-6',...
20           'MaxStep','1e-4');
21  sol_bump = sim(model_name,t_grid);
22  %% Extracción de los resultados
23  y_sims = sol_bump.yout(:,1);
24  th_sims = sol_bump.yout(:,2);
25  r1_sims = sol_bump.yout(:,3);
26  r2_sims = sol_bump.yout(:,4);
27  ad_sims = sol_bump.yout(:,5);
28  % Exportamos los resultados
29  file = '../half_car_driver_arm.csv';
30  sol = [t_grid,y_sims,th_sims,r1_sims,r2_sims,ad_sims];
31  writematrix(sol,file)

```

Figura 58. Código Matlab para la simulación del modelo Simscape de medio vehículo.

```

32  %% Gráfico aceleración del conductor
33  figure(100)
34  plot(t_grid,ad_sims,'LineWidth',1.2)
35  grid
36  %% Gráficos del desplazamiento del CG y rotación del cuerpo
37  figure(200), subplot(2,1,1)
38  plot(t_grid,y_sims,'b','LineWidth',1.2)
39  grid
40  subplot(2,1,2)
41  plot(t_grid,th_sims,'k','LineWidth',1.2)
42  grid
43  %% Gráficos del rattle space
44  figure(300)
45  plot(t_grid,[r1_sims,r2_sims],'LineWidth',1.2)
46  grid, legend('r1(t)','r2(t)')

```

Figura 59. Código para generar los gráficos de la respuesta forzada del modelo de medio vehículo.

Observamos que en el caso de la respuesta forzada, todas las condiciones iniciales se consideran nulas y los valores correspondientes se han definido directamente en los distintos bloques (por defecto, todos los bloques tienen condiciones iniciales nulas). Las gráficas de las respuestas obtenidas se presentan en las Figuras 60 ,61 y 62.

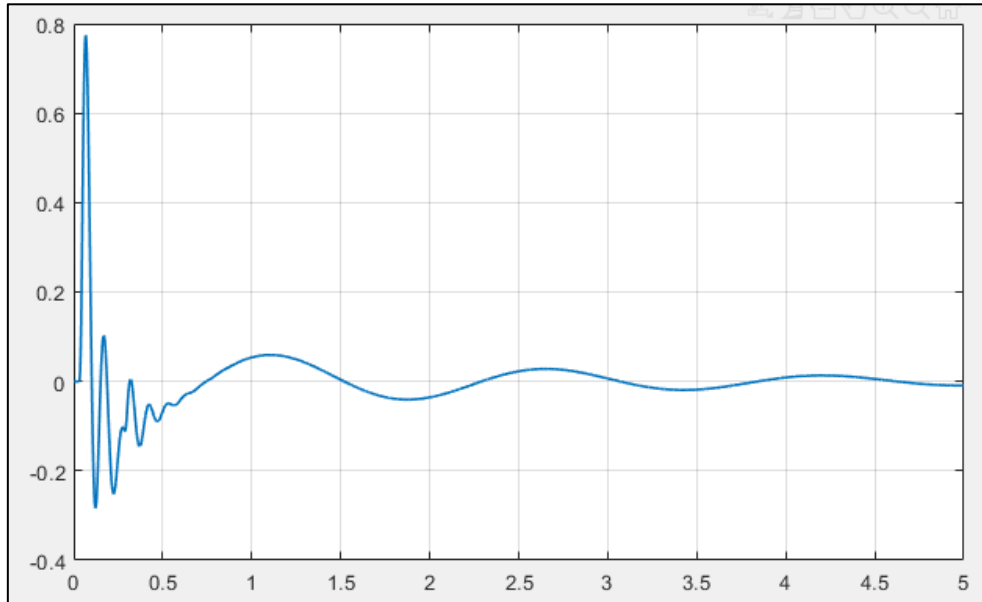


Figura 60. Aceleración vertical del conductor (en m/s^2) en la respuesta forzada del modelo de medio vehículo.

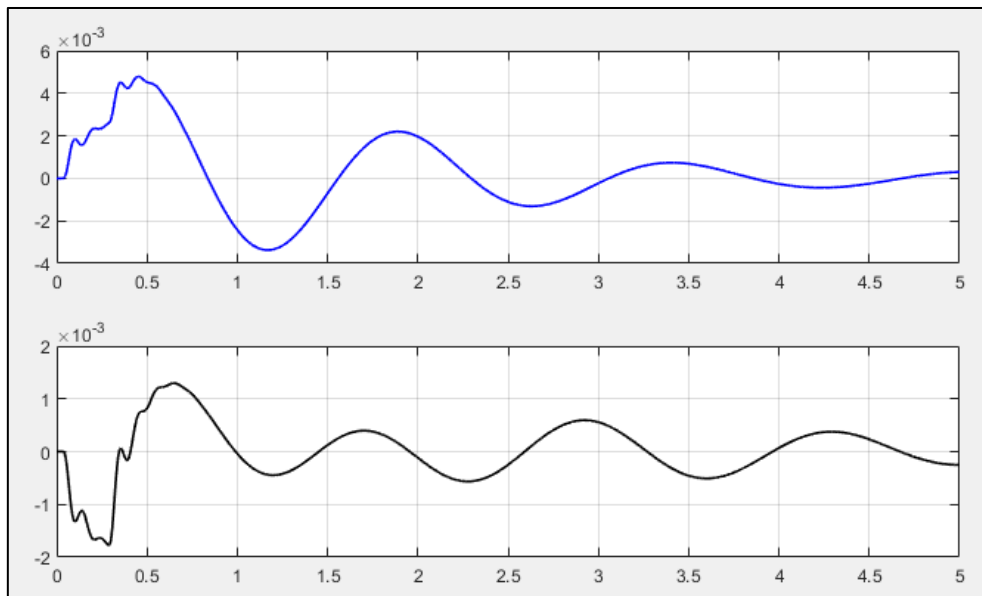


Figura 61. Desplazamiento vertical del centro de gravedad (en m) y ángulo de cabeceo (en rad) en el modelo de medio vehículo.

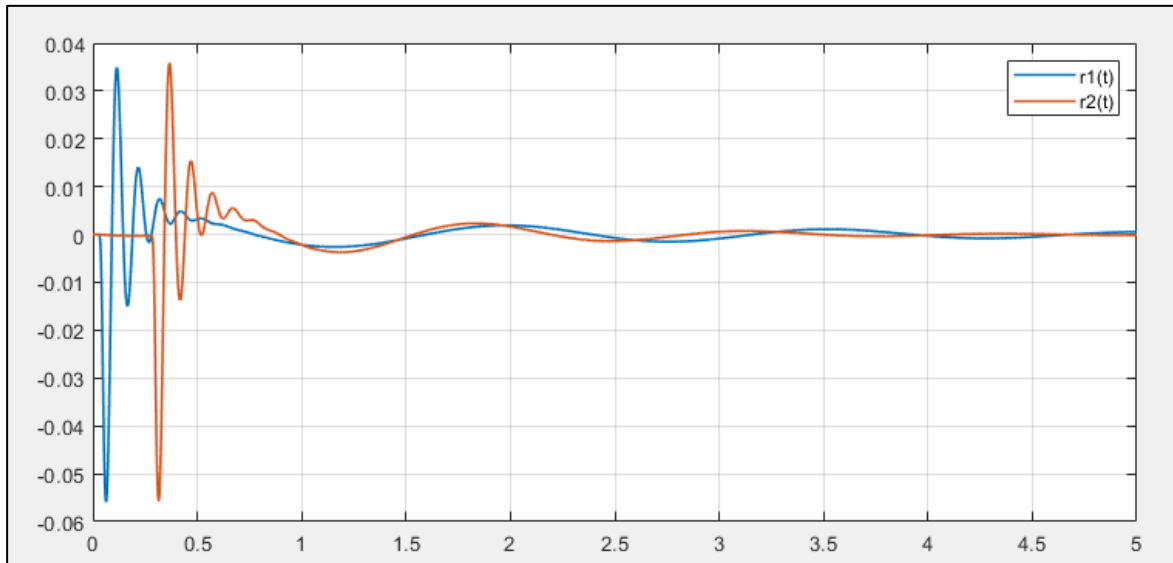


Figura 62. Elongación (en m) del sistema de suspensión delantero ($r1$) y trasero ($r2$) en la respuesta forzada del modelo de medio vehículo.

El gráfico de la Figura 60 muestra que el paso sobre el bache produce un pico de aceleración vertical de 0.8 m/s^2 sobre el conductor. En la Figura 62, observamos que la perturbación externa produce una compresión de los sistemas de suspensión de unos 6 cm , seguida de una elongación inferior a 4 cm . El efecto de la perturbación sobre el desplazamiento vertical del centro de gravedad y el cabeceo del cuerpo del vehículo es muy reducido, con valores inferiores a 0.5 cm y 0.2 deg , respectivamente (ver Figura 61).

5.4. Validación de los nuevos elementos Simscape

En las simulaciones de respuesta forzada realizadas en este capítulo se han programado nuevos elementos *Simscape*. En particular, se ha diseñado un generador de posición y un sensor de aceleración en el dominio mecánico traslacional y un elemento que permite modelar el desplazamiento vertical y el cabeceo del cuerpo del vehículo.

Para validar el buen funcionamiento de los nuevos bloques, se ha calculado la respuesta forzada de los modelos de doble masa y de medio vehículo usando *Maple*. El código completo de la resolución con *Maple* y la comparación con los resultados de *Simscape* se ha incluido en el Anexo.

La comparación de los resultados obtenidos con *Maple* y *Simscape* confirma el buen funcionamiento de los nuevos elementos *Simscape*. En concreto, tomando como referencia los valores proporcionados con *Maple* a partir de las ecuaciones algebraico-

diferenciales de los modelos, el error estimado en el desplazamiento vertical del cuerpo del vehículo y el ángulo de cabeceo es inferior $1.7 \cdot 10^{-8}$. Para la deflexión de los sistemas de suspensión, la magnitud de los errores es del orden de $1.5 \cdot 10^{-6}$. En la aceleración vertical del conductor, se observan errores inferiores a $1.5 \cdot 10^{-5}$. La diferencia de precisión de los diferentes casos es consistente con el orden de magnitud y la rapidez de la variación de las respuestas correspondientes.

6. OPTIMIZACIÓN DEL SISTEMA DE SUSPENSIÓN DEL ASIENTO

Para ilustrar el potencial práctico de la simulación con *Simscape*, en este capítulo estudiamos la mejora del sistema de suspensión del asiento del conductor en el modelo de medio vehículo. El objetivo es determinar valores adecuados del coeficiente de elasticidad k_d y el coeficiente de amortiguación c_d que permitan reducir la aceleración vertical del conductor sin incrementar el desplazamiento relativo del conductor respecto del cuerpo del vehículo.

Para ello, consideramos un conjunto de valores de cada parámetro $Kd = \{kd_1, \dots, kd_{n1}\}$, $Cd = \{cd_1, \dots, cd_{n2}\}$ y calculamos la respuesta forzada del modelo de medio vehículo, tomando como excitación externa el modelo de bache sinusoidal discutido en el capítulo anterior, y manteniendo los valores de la Tabla 2 para el resto de los parámetros.

Las simulaciones pueden realizarse con el diagrama *Simscape* de la Figura 63, que proporciona los valores de la aceleración del conductor (señal $ad(t)$ en el puerto 1) y el desplazamiento relativo del conductor respecto del centro de gravedad del cuerpo del vehículo (señal $yrel(t)$ en el puerto 2). Para evaluar el desempeño de una configuración particular (kd_i, cd_j) , definimos la función de coste

$$\text{coste}(kd_i, cd_j) = \omega_1 \cdot \widehat{ad}_{ij} + \omega_2 \cdot \widehat{yrel}_{ij} \quad (20)$$

donde

$$\widehat{ad}_{ij} = \max\{|ad_{ij}(t)|, t \in [0, t_{max}]\}$$

$$\widehat{yrel}_{ij} = \max\{|yrel_{ij}(t)|, t \in [0, t_{max}]\}$$

representa la magnitud de los picos obtenidos en la respuesta correspondiente a (kd_i, cd_j) . Los coeficientes ω_1 y ω_2 permiten compensar las diferencias en el orden de magnitud y priorizar la reducción de una de las variables de salida.

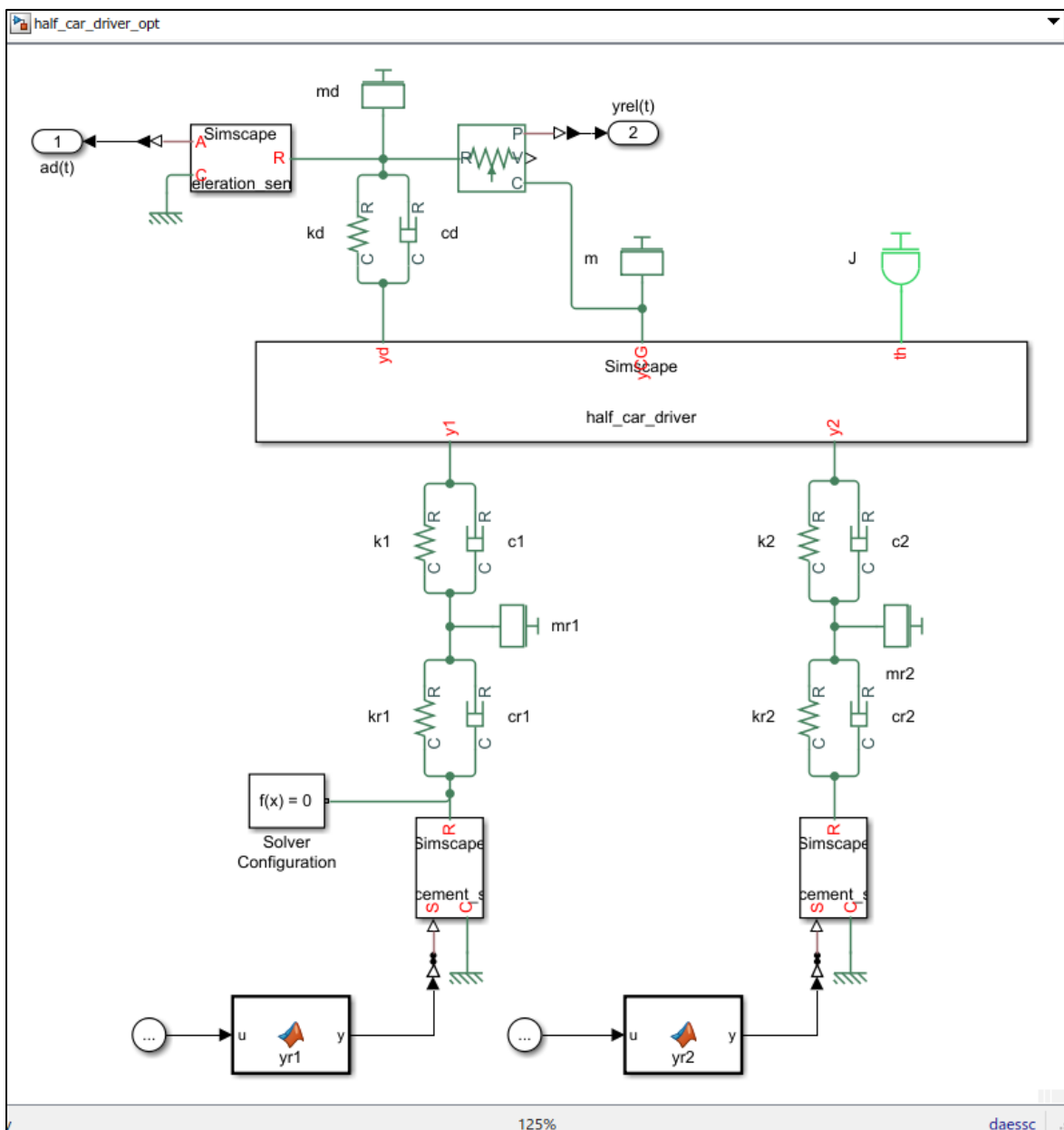


Figura 63. Diagrama Simscape para el modelo de medio vehículo con conductor usado en la simulación en paralelo.

Observamos que, en total, hay que realizar un conjunto de $n = n_1 \cdot n_2$ simulaciones, que pueden ejecutarse en paralelo en los diferentes núcleos del procesador usando el comando `parsim()`. Este comando requiere la creación de una estructura de datos que contenga los valores particulares de los parámetros. En la Figura 64 se muestra el código *Matlab* que ejecuta la simulación en paralelo para un conjunto de $n = 609$ pares (kd_i, cd_j) .

```

run_half_car_parsim_opt.m
1  clear, clc
2  %% Parámetros de la simulación
3  kds = 2000:500:16000; % valores de kd
4  cds = 0:50:1000;      % valores de cd
5  %% Configuración del modelo
6  model = 'half_car_driver_opt';
7  open_system(model, 'loadonly')
8  config = getActiveConfigSet(model);
9  set_param(model, ...
10         'Solver', 'daessc', ...
11         'RelTol', '1e-6', ...
12         'MaxStep', '1e-3', ...
13         'StopTime', '1', ...
14         'OutputTimes', '[0:0.01:1]') % vector de tiempos
15  %% Estructura de Inputs de la simulación
16  for i = 1:length(kds)
17      for j = 1:length(cds)
18          in(i,j) = Simulink.SimulationInput(model);
19          in(i,j) = in(i,j).setVariable('kd',kds(i));
20          in(i,j) = in(i,j).setVariable('cd',cds(j));
21      end
22  end
23  %% Simulación
24  xout=parsim(in, 'ShowSimulationManager','on',...
25              'ShowProgress','on');

```

Figura 64. Script Matlab para la ejecución en paralelo de un lote de 609 simulaciones.

Teniendo en cuenta que, en el problema considerado, los valores pico de las variables respuesta se producen durante el primer segundo, el vector de tiempos se ha restringido a 101 valores en el intervalo $[0,1]$ (línea 14 del script), lo que permite reducir notablemente la carga computacional. La Figura 65 muestra la herramienta gráfica de seguimiento de la simulación en paralelo, que ha requerido un tiempo de computación de 5 min y 28 seg.

Una vez completado el lote de simulaciones, el código *Matlab* de la Figura 66 permite calcular los valores de la función de coste, y realizar las representaciones gráficas de las Figuras 67 y 68. Para determinar el valor de los coeficientes ω_1 y ω_2 , se ha tomado la magnitud de los picos $ad^* = 0.78 \text{ m/s}^2$ y $yrel^* = 0.0019 \text{ m}$ correspondiente a los valores de referencia $kd = 16000 \text{ N/m}$ y $cd = 1000 \text{ N} \cdot \text{s/m}$. De donde se obtienen los factores de normalización $\hat{\omega}_1 = 2/ad^* = 2.6$, $\hat{\omega}_2 = 1/yrel^* = 522$, que priorizan la reducción de la aceleración del conductor. Finalmente, se define $\omega_1 = \hat{\omega}_1/\hat{\omega}_2 \approx 0.005$, $\omega_2 = 1$.

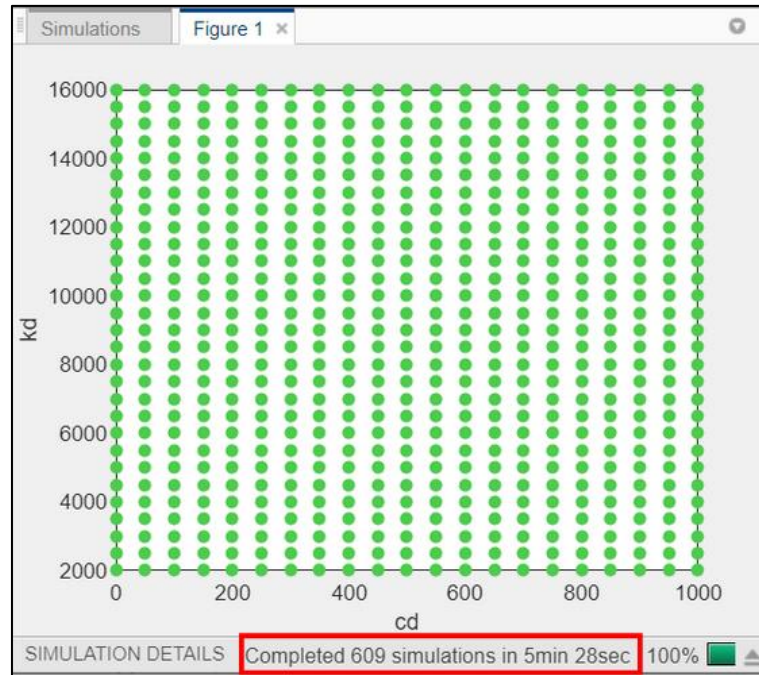


Figura 65. Herramienta gráfica de seguimiento de la simulación en paralelo.

```

26  %% Función coste
27  w1=0.005; w2=1;
28  cost=zeros(length(kds),length(cds));
29  for k=1:length(kds)
30      for j=1:length(cds)
31          ad_peak=max(abs(xout(k,j).yout(:,1)));
32          y_peak=max(abs(xout(k,j).yout(:,2)));
33          cost(k,j)=w1*ad_peak + w2*yd_peak;
34      end
35  end
36  %% Gráfico de superficie de la función coste
37  figure(100)
38  [Mkd,Mcd] = meshgrid(kds,cds);
39  p_surf = surfc(Mkd,Mcd,cost');
40  p_surf(2).LineWidth = 1;
41  p_surf(2).LevelList = 0.003:1e-4:0.005;
42  colormap(cool); colorbar
43  %% Gráfico de controno de la función coste
44  figure(200)
45  [M,pc] = contourf(Mkd,Mcd,cost',[0.003:1e-4:0.005]);
46  grid on; pc.Fill = 'on';
47  pc.LineWidth = 1;
48  colormap(cool), colorbar
    
```

Figura 66. Código Matlab para calcular la función coste.

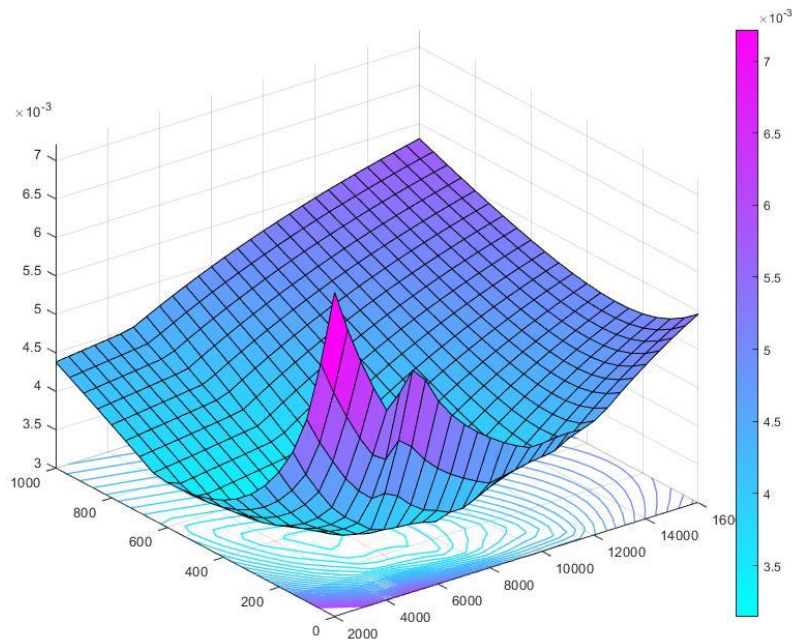


Figura 67. Superficie definida por los puntos (kd_i, cd_j, z_{ij}) con $z_{ij} = coste(kd_i, cd_j)$.

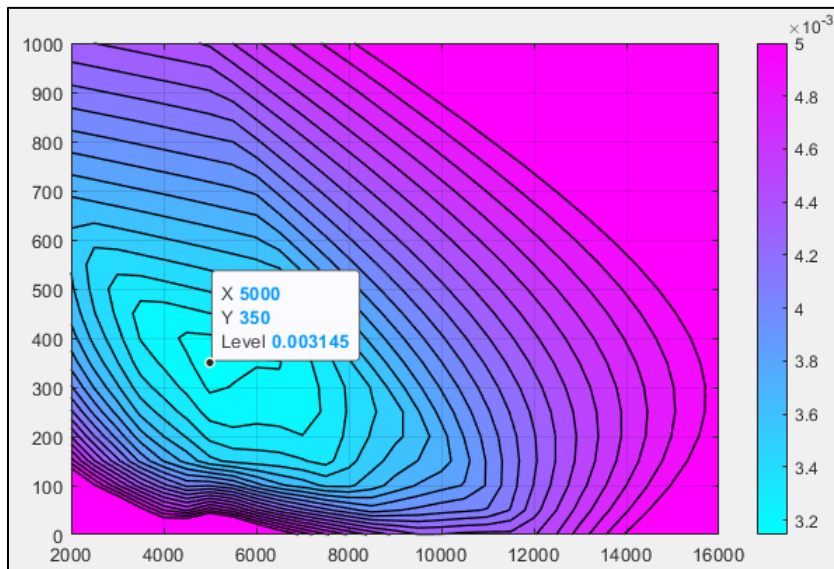


Figura 68. Curvas de nivel definidas por los puntos (kd_i, cd_j, z_{ij}) con $z_{ij} = coste(kd_i, cd_j)$.

Observando las gráficas de las Figuras 67 y 68, podemos tomar como configuración óptima $k_d^* = 5000 \text{ N/m}$ y $c_d^* = 350 \text{ N} \cdot \text{s/m}$. Las respuestas correspondientes a la configuración de referencia y a la configuración óptima se muestran en las Figuras 69 y 70, donde podemos ver que la configuración (kd^*, cd^*) proporciona una notable reducción del pico de aceleración, sin introducir efectos adversos en el desplazamiento

relativo del conductor respecto del cuerpo del vehículo. La reducción de la aceleración está alrededor del 60% de valor máximo del pico.

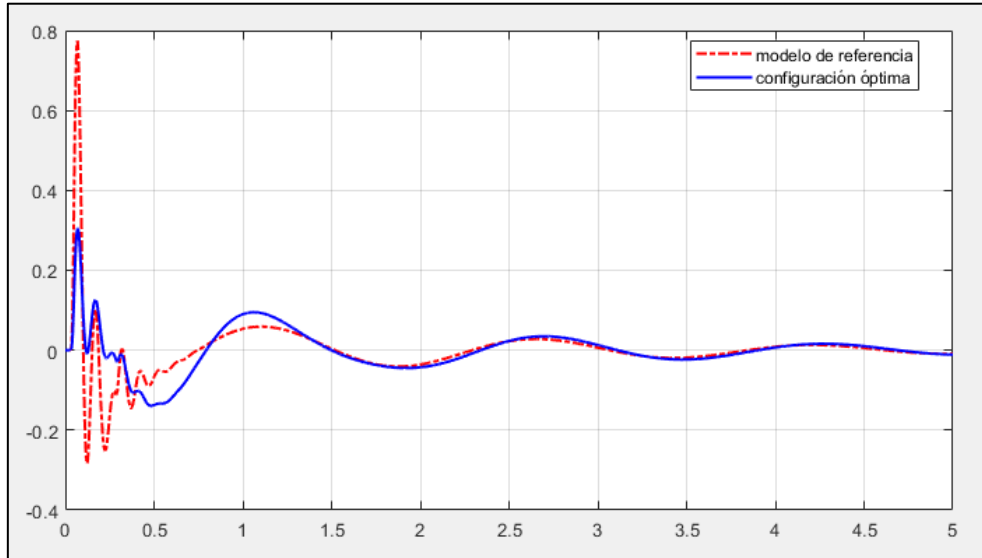


Figura 69. Aceleración del conductor (en m/s^2) correspondiente a la configuración de referencia y la configuración óptima (kd^*, cd^*).

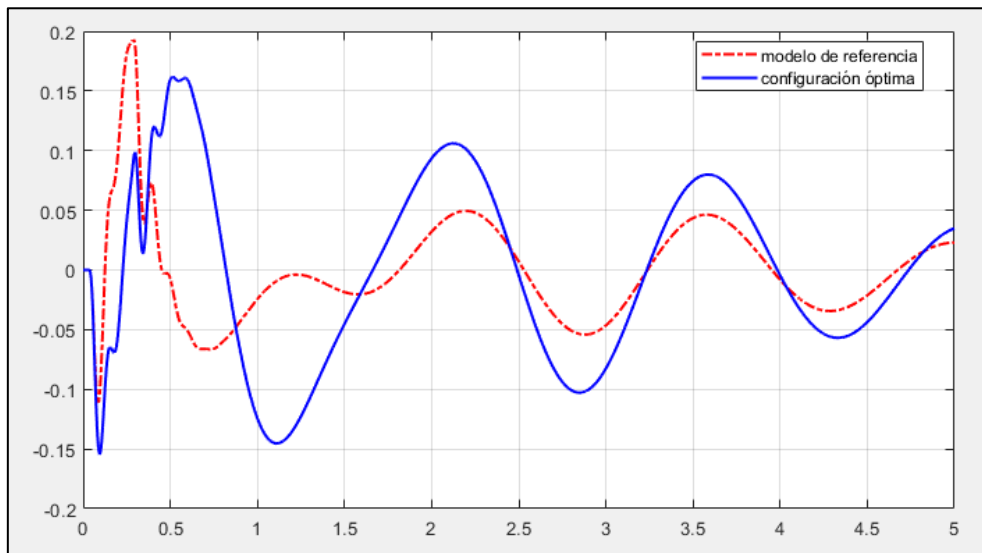


Figura 70. Desplazamiento relativo del conductor (en cm) correspondiente a la configuración de referencia y la configuración óptima (kd^*, cd^*).

7. CONCLUSIONES

En este trabajo se ha empleado *Simscape* para estudiar la respuesta dinámica de sistemas de suspensión y la resolución de un problema de diseño óptimo. La principal característica de *Simscape* es la posibilidad de construir modelos computacionales para problemas complejos mediante la combinación de elementos físicos de diferentes dominios (mecánico traslacional, mecánico rotacional, eléctrico, térmico, magnético, etc.). Después de realizar el trabajo y considerando las dificultades encontradas y los resultados obtenidos, podemos resaltar los siguientes puntos positivos:

- Existencia de un buen número de bibliotecas, con elementos predefinidos para diversos campos científicos y técnicos.
- Posibilidad de realizar modelos computacionales que combinen elementos de *Simscape* y *Simulink*.
- Posibilidad de configurar y realizar simulaciones de modelos *Simscape* mediante *scripts Matlab*.
- Posibilidad de interactuar con *Matlab* mediante *scripts*.
- Existencia de un completo lenguaje de programación, que permite definir nuevos dominios y elementos.
- Posibilidad de definir subsistemas para abordar problemas complejos.
- Posibilidad real de abordar problemas multidominio.
- Posibilidad de usar recursos de cálculo paralelo.

También podemos destacar dos aspectos negativos fundamentales:

- La construcción de modelos computacionales basados en la combinación de elementos físicos es un enfoque muy distinto del tradicional. Es preciso realizar un cambio de mentalidad y entender con claridad los elementos fundamentales del nuevo enfoque.
- No existen publicaciones de carácter didáctico que expliquen e ilustren la aplicación práctica de *Simscape*.

En resumen, resulta claro que *Simscape* puede ser una herramienta de gran relevancia en muchos campos de la ingeniería. Sin embargo, para que ese impacto se produzca, sería preciso realizar un importante esfuerzo de difusión y formación.

Para finalizar, queremos destacar una futura línea de trabajo de particular interés que se nos ha planteado en la resolución del ejemplo de diseño óptimo presentado en el Capítulo 6. Esta línea consiste en el uso combinado de modelos computacionales de Simscape con estrategias avanzadas de optimización aleatoria, como el Algoritmo Genético o el Enjambre de Partículas incluidos en el *Global Optimization Toolbox* de *Matlab*, que podría permitir abordar problemas de diseño óptimo con múltiples parámetros.

8. BIBLIOGRAFÍA

1. **Salem, Mahmoud Hosny.** Investigation of a non-linear suspension in a quarter car model. s.l. Aston Univerity, 2018.
2. **MathWorks.** *Control System Toolbox*. Natick. The MathWorks Inc., 2020. Vol. R2020a.
3. **Maplesoft.** *Maple User Manual*. s.l. Waterloo Maple Inc., 2021.
4. **MathWorks.** *MatLab Documentation*. Natick. The MathWorks Inc., 2020. Vol. R2020a.
5. —. *Symbolic Math Toolbox*. Natick. The MathWorks Inc., 2020. Vol. R2020a.
6. —. *Simulink User's Guide*. Natick. The MathWorks Inc., 2020. Vol. R2020a.
7. —. *Simscape User's Guide*. Natick. The MathWorks Inc., 2020. Vol. R2020a.
8. —. *Simscape Lenguage Guide*. Natick. The MathWorks Inc., 2020. Vol. R2020a.
9. —. *Parallel Computing Toolbox*. Natick. The MathWorks Inc., 2020. Vol. R2020a.
10. **Rajamani, Rajesh.** Design and analysis of passive automotive suspensions. *Vehicle dynamics and control*. Minneapolis. Springer, 2011.
11. *Simulación de la dinámica vertical de un cuarto de vehiculo mediante MatLab, Simulink y Simscape.* **Reula, Jorge A., Peralta, Alejandro O. y Heidenreich, Elvio.** Santa Fe : AMCA, 2019, Mecánica Computacional, Vol. XXXVII, págs. 1521 - 1530.

9. ANEXO. DOCUMENTOS MAPLE

9.1. Código Maple para la respuesta forzada del modelo de doble masa

Excitación externa

$v\omega h := 40 :$

$v := \frac{v\omega h}{3.6} :$ # velocidad

$h := 0.1 :$ # altura del bache

$b := 0.3 :$ # longitud del bache

$T := \frac{b}{v} :$ # periodo del bache

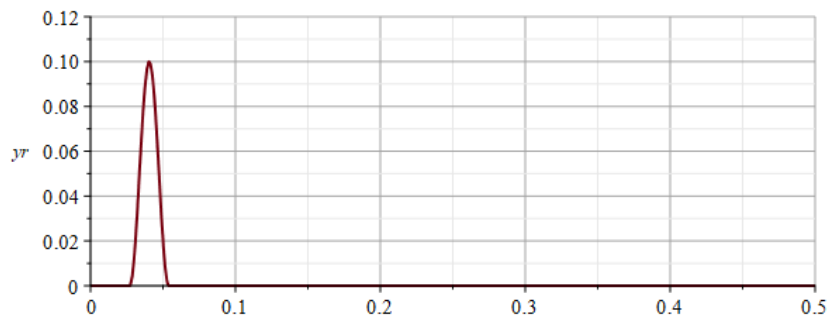
$\omega := \frac{2 \cdot \pi}{T} :$ # frecuencia del bache

$yr := \text{piecewise}\left(t < T, 0, t \leq 2 \cdot T, \frac{h}{2} \cdot (1 - \cos(\omega \cdot t)), 0\right)$

$$yr := \begin{cases} 0 & t < 0.02700000000 \\ 0.05000000000 - 0.05000000000 \cos(232.7105670 t) & t \leq 0.05400000000 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Representación gráfica de $yr(t)$

$p0 := \text{plot}(yr, t = 0 .. 0.5, yr = 0 .. 0.12, \text{thickness} = 2, \text{gridlines}, \text{size} = [600, 250])$



Sistema de EDOs del modelo

$De1 := m \cdot \ddot{y} + c \cdot (\dot{y} - \dot{y}\alpha) + k \cdot (y(t) - y\alpha(t)) = 0 :$

$De2 := m_r \cdot \ddot{y}\alpha + c \cdot (\dot{y}\alpha - \dot{y}) + k \cdot (y\alpha(t) - y(t)) + k_r \cdot (y\alpha(t) - y_r) = 0 :$

Condiciones iniciales

$IC := \{y(0) = 0, D(y)(0) = 0, y\alpha(0) = 0, D(y\alpha)(0) = 0\} :$

Parámetros del modelo

$param := \{m = 250, k = 16000, c = 1000, m_r = 45, k_r = 160000\} :$

$De1p := \text{subs}(param, De1)$

$$De1p := 250 \frac{d^2}{dt^2} y(t) + 1000 \frac{d}{dt} y(t) - 1000 \frac{d}{dt} y\alpha(t) + 16000 y(t) - 16000 y\alpha(t) = 0 \quad (2)$$

$De2p := \text{subs}(param, De2)$

$$De2p := 45 \frac{d^2}{dt^2} y\alpha(t) + 1000 \frac{d}{dt} y\alpha(t) - 1000 \frac{d}{dt} y(t) + 176000 y\alpha(t) - 16000 y(t) \quad (3)$$

$$- 160000 \left(\begin{cases} 0 & t < 0.02700000000 \\ 0.05000000000 - 0.05000000000 \cos(232.7105670 t) & t \leq 0.05400000000 \\ 0 & \text{otherwise} \end{cases} \right) = 0$$

Solución numerica del modelo

```
sol := dsolve({De1p, De2p} union IC, {y(t), ya(t)}, mnumeric, maxstep=1e-4, relerr=1e-8, output=operator, maxfun=0) :
sol(0.25)
```

$$[t = 0.25, y(0.25) = 0.00655409057111220, D(y)(0.25) = -0.0379320910450918, ya(0.25) = 0.00158117940717295, D(ya)(0.25) = 0.394383874173620] \tag{4}$$

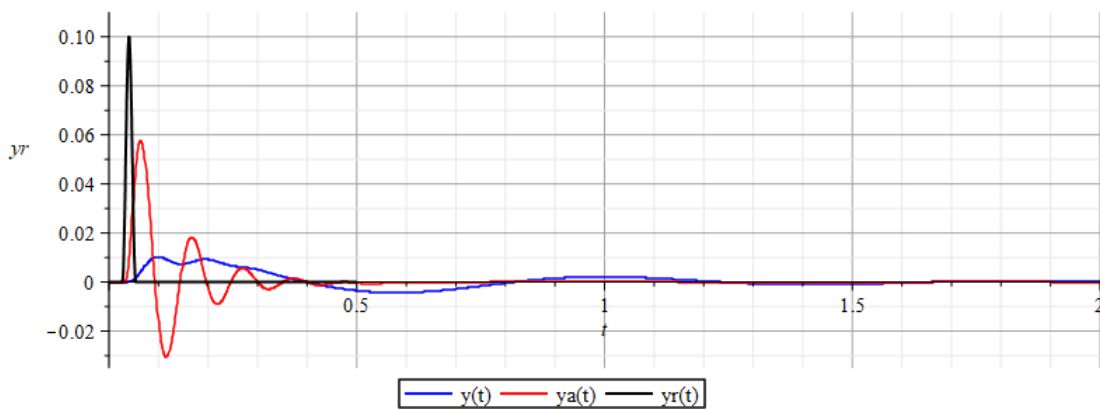
Extracción de la respuesta

```
y_sol := subs(sol, y)
y_sol := proc(t) ... end proc \tag{5}
```

```
ya_sol := subs(sol, ya)
ya_sol := proc(t) ... end proc \tag{6}
```

Gráficos de la respuesta

```
p1 := plot(y_sol, 0..3, numpoints = 2000) :
p2 := plot(ya_sol, 0..3, numpoints = 2000) :
plots[display]([p1, p2, p0], view = [0..2, -0.035..0.11], color = [blue, red, black], gridlines, legend = ["y(t)", "ya(t)", "yr(t)"], size = [800, 300])
```



Importamos la respuesta del sistema simulado usando Simscape

```
bump_simscape := ImportMatrix("double_mass_bump_simscape.csv") :
```

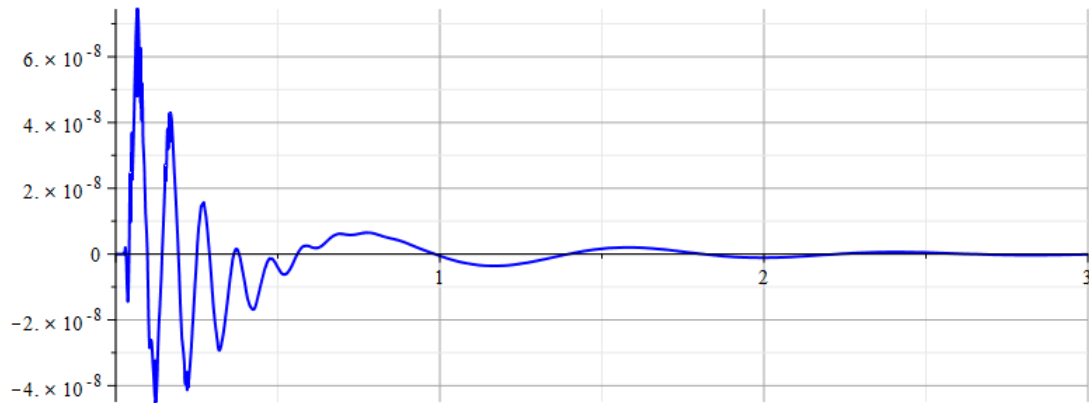
Interpolamos los puntos de los valores importados para generar una función temporal

```
sims_y := Interpolation:-CubicInterpolation(bump_simscape[ .., [1, 2]])
sims_y := [ a cubic interpolation object
            with 3000 points in 1-D ] \tag{7}
```

```
sims_ya := Interpolation:-CubicInterpolation(bump_simscape[ .., [1, 3]])
sims_ya := [ a cubic interpolation object
            with 3000 points in 1-D ] \tag{8}
```

Gráfica error para $y(t)$, $ya(t)$

`p6 := plot(y_sol - sims_y, 0..3, color=blue, thickness=2, gridlines, size=[800, 300])`



`p7 := plot(ya_sol - sims_ya, 0..3, color=red, thickness=2, gridlines, size=[800, 300])`



9.2. Código Maple para la respuesta forzada del modelo de medio vehículo

Parámetros del bache

$vkm := 40 :$

$v := \frac{vkm}{3.6} :$ # velocidad

$b := 0.3 :$ # longitud del bache

$a := 0.1 :$ # altura del bache

Delay entre ejes

$L1 := 1.2 :$ # posición del eje delantero respecto el C.G.

$L2 := 1.6 :$ # posición del eje trasero respecto el C.G.

$\delta := \frac{(L1 + L2)}{v}$ # delay entre ejes

$$\delta := 0.2520000000 \tag{1}$$

Función bache

$u := t \rightarrow \text{piecewise}\left(t < \frac{b}{v}, 0, t \leq \frac{2 \cdot b}{v}, \frac{a}{2} \cdot \left(1 - \cos\left(\frac{2 \cdot \pi}{b} \cdot v \cdot t\right)\right), 0\right) :$

Excitación externa del primer eje

$yr1 := u(t)$

$$yr1 := \begin{cases} 0 & t < 0.02700000000 \\ 0.05000000000 - 0.05000000000 \cos(232.7105670 t) & t \leq 0.05400000000 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Excitación externa del segundo eje

$yr2 := u(t - \delta)$

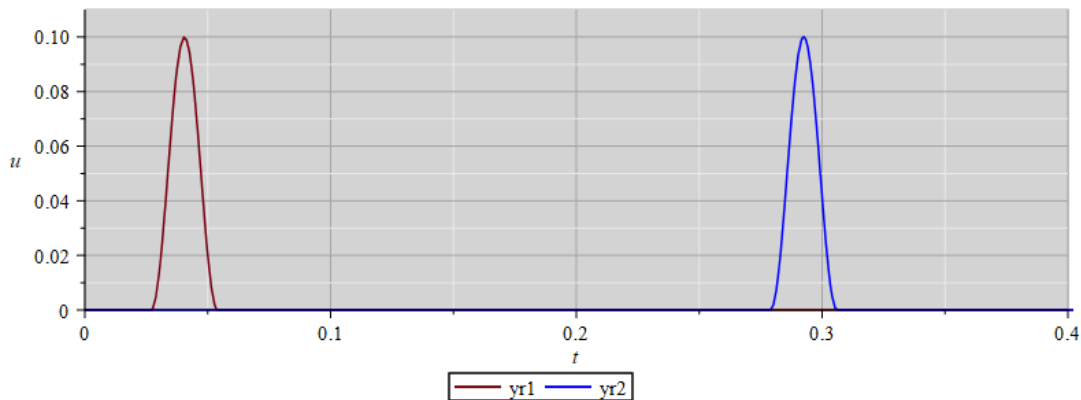
$$yr2 := \begin{cases} 0 & t < 0.27900000000 \\ 0.05000000000 - 0.05000000000 \cos(232.7105670 t - 58.64306288) & t \leq 0.30600000000 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Gráfica de la excitación externa

$pb1 := \text{plot}(yr1, t = 0 .. 0.5, 'u' = 0 .. 0.15) :$

$pb2 := \text{plot}(yr2, t = 0 .. 0.5, 'u' = 0 .. 0.15, \text{color} = \text{blue}) :$

$\text{plots}:-\text{display}(pb1, pb2, \text{legend} = ["yr1", "yr2"], \text{view} = [0 .. 0.4, 0 .. 0.11], \text{size} = [800, 300], \text{background} = \text{LightGray}, \text{gridlines})$



Definimos el conjunto de EDO

$$DEdriver := md \cdot \ddot{y}d + kd \cdot (y_d(t) - y_3(t)) + cd \cdot (\dot{y}d - \dot{y}3) = 0 :$$

$$DE := m \cdot \ddot{y} = kd \cdot (y_d(t) - y_3(t)) + cd \cdot (\dot{y}d - \dot{y}3) - kl \cdot (y_1(t) - ya_1(t)) - k2 \cdot (y_2(t) - ya_2(t)) - c1 \cdot (\dot{y}1 - \dot{y}a_1) - c2 \cdot (\dot{y}2 - \dot{y}a_2) :$$

$$DE1 := J \cdot \ddot{\theta} = -kd \cdot (y_d(t) - y_3(t)) \cdot a_3 - cd \cdot (\dot{y}d - \dot{y}3) \cdot a_3 + kl \cdot (y_1(t) - ya_1(t)) \cdot a_1 - k2 \cdot (y_2(t) - ya_2(t)) \cdot a_2 + c1 \cdot (\dot{y}1 - \dot{y}a_1) \cdot a_1 - c2 \cdot (\dot{y}2 - \dot{y}a_2) \cdot a_2 :$$

$$DEw1 := ma_1 \cdot \ddot{y}a_1 - kl \cdot (y_1(t) - ya_1(t)) - c1 \cdot (\dot{y}1 - \dot{y}a_1) + kr1 \cdot (ya_1(t) - yr1) :$$

$$DEw2 := ma_2 \cdot \ddot{y}a_2 - k2 \cdot (y_2(t) - ya_2(t)) - c2 \cdot (\dot{y}2 - \dot{y}a_2) + kr2 \cdot (ya_2(t) - yr2) :$$

Ecuaciones de geometria adicionales

$$EQ1 := y_1(t) = y(t) - a_1 \cdot \theta(t) :$$

$$EQ2 := y_2(t) = y(t) + a_2 \cdot \theta(t) :$$

$$EQ3 := y_3(t) = y(t) - a_3 \cdot \theta(t) :$$

Variables de salida

$$EQout1 := ad(t) = \ddot{y}d : \# \text{aceleración del conductor}$$

$$EQout2 := r1(t) = y_1(t) - ya_1(t) : \# \text{rattle space del eje 1}$$

$$EQout3 := r2(t) = y_2(t) - ya_2(t) : \# \text{rattle space del eje 2}$$

Condiciones iniciales

$$ICs := \{y_d(0) = 0, D(y_d)(0) = 0, y(0) = 0, \dot{y}(0) = 0, \theta(0) = 0, D(\theta)(0) = 0, y_1(0) = 0, y_2(0) = 0, y_3(0) = 0, ya_1(0) = 0, D(ya_1)(0) = 0, ya_2(0) = 0, D(ya_2)(0) = 0\} :$$

Parámetros del sistema

$$pars := [md = 100, cd = 1000, kd = 16000, m = 1460, J = 2460, kl = 16000, k2 = 16000, a_1 = L1, a_2 = L2, a_3 = 0.7, c1 = 1000, c2 = 1000, ma_1 = 45, ma_2 = 45, kr1 = 160000, kr2 = 160000] :$$

Ecuaciones con parámetros

$$DEdriverp := subs(pars, DEdriver) :$$

$$DEp := subs(pars, DE) :$$

$$DE1p := subs(pars, DE1) :$$

$$DEw1p := subs(pars, DEw1) :$$

$$DEw2p := subs(pars, DEw2) :$$

$$EQ1p := subs(pars, EQ1) :$$

$$EQ2p := subs(pars, EQ2) :$$

$$EQ3p := subs(pars, EQ3) :$$

Solución numerica del sistema

$$sol := dsolve(\{DEdriverp, DEp, DE1p, DEw1p, DEw2p, EQ1p, EQ2p, EQ3p, EQout1, EQout2, EQout3\} \text{union } ICs, \{ad(t), y_d(t), y(t), \theta(t), y_1(t), y_2(t), y_3(t), ya_1(t), ya_2(t), r1(t), r2(t)\}, \text{numeric}, \text{maxstep} = 1e-4, \text{maxfun} = 0, \text{relerr} = 1e-8, \text{output} = \text{operator}) :$$

sol(0.25) # ejemplo evaluación

$$\begin{aligned} [t = 0.25, ad(0.25) = -0.165285454716638, r1(0.25) = 0.00201595164208998, r2(0.25) \\ = -0.000276623722116845, \theta(0.25) = -0.00164511312830160, D(\theta)(0.25) \\ = -0.00217513522386360, y(0.25) = 0.00235887858619573, D(y)(0.25) = 0.00429079792677752, \\ y_1(0.25) = 0.00433301434015766, y_2(0.25) = -0.000273302419086826, y_3(0.25) \\ = 0.00351045777600685, ya_1(0.25) = 0.00231706269806767, D(ya_1)(0.25) = 0.401798674925099, \\ ya_2(0.25) = 3.32130303001866 \cdot 10^{-6}, D(ya_2)(0.25) = 0.000278172255474775, y_d(0.25) \\ = 0.00417000868758548, D(y_d)(0.25) = 0.0117891234698878] \end{aligned} \quad (4)$$

Extracción de los resultados

$$fad := subs(sol, ad) :$$

$$fy := subs(sol, y) :$$

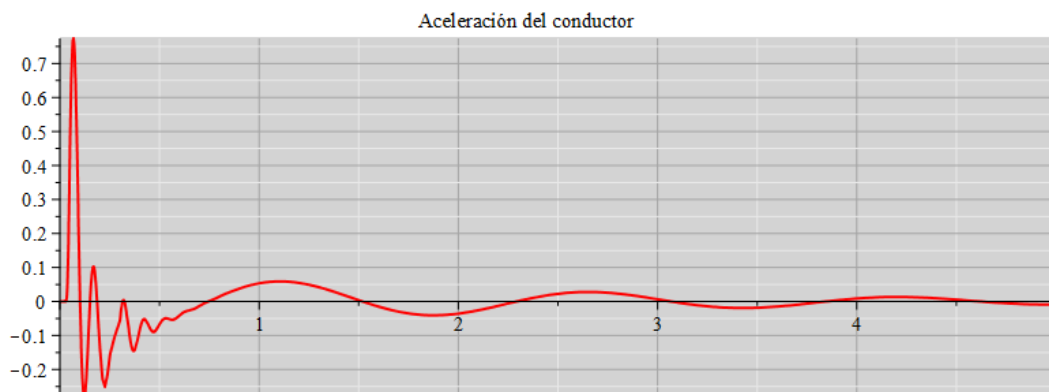
$$f\theta := subs(sol, \theta) :$$

$$fr1 := subs(sol, r1) :$$

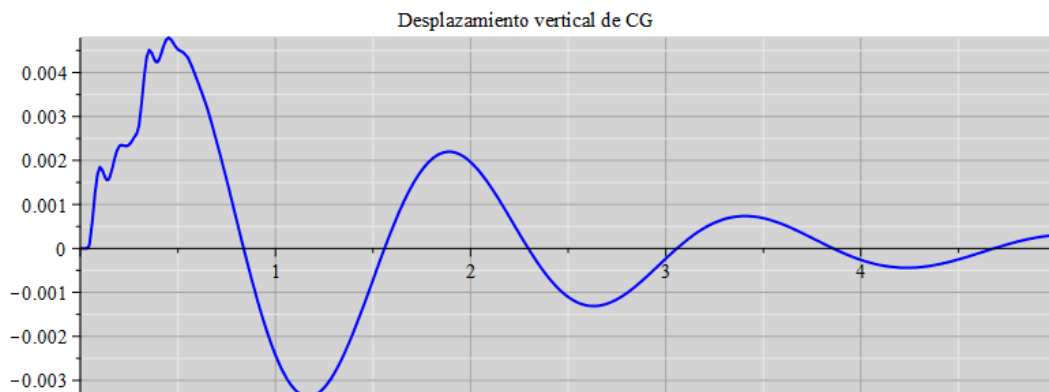
$$fr2 := subs(sol, r2) :$$

Gráficas de los resultados

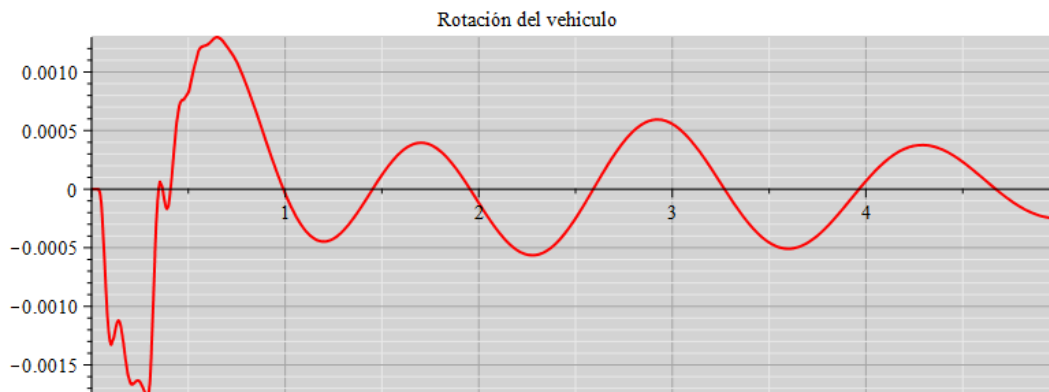
```
plot(fad, 0 ..5, color = red, thickness = 2, gridlines, background = LightGray, title = "Aceleración del conductor", size = [ 800, 300 ])
```



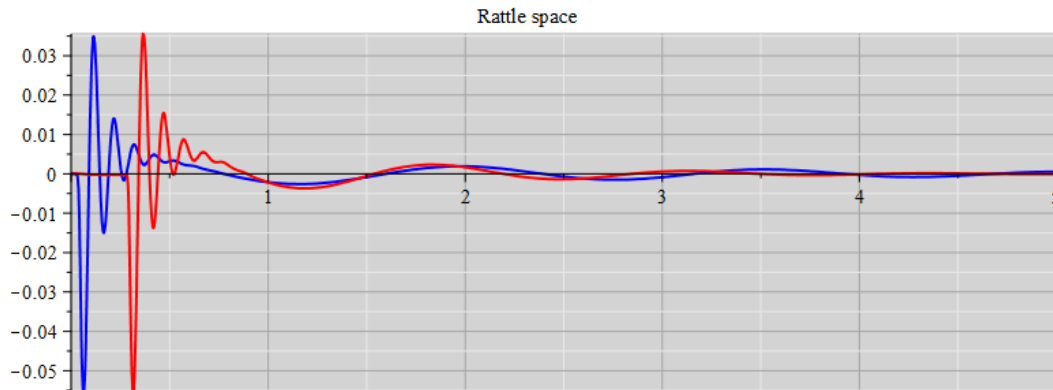
```
plot(fy, 0 ..5, color = blue, thickness = 2, gridlines, background = LightGray, title = "Desplazamiento vertical de CG", size = [ 800, 300 ])
```



```
plot(fθ, 0 ..5, color = red, thickness = 2, gridlines, background = LightGray, title = "Rotación del vehículo", size = [ 800, 300 ])
```



```
plot([fr1,fr2], 0..5, color=[blue,red], thickness=[2,2], gridlines, background=LightGray, title
="Rattle space", size=[800,300])
```



Importamos los datos de Simscape

2a columna -> Desplazamiento del centro de gravedad del chasis

3a columna -> Rotación

4a y 5a columna -> Rattle space

6a columna -> Aceleración del conductor

```
xdat2 := ImportMatrix("../half_car_driver_arm.csv", datatype=double) :
```

Interpolamos los puntos de los valores importados

```
syms_ad := Interpolation[CubicInterpolation](xdat2[ ..., [1, 6]]) :
```

```
syms_y := Interpolation[CubicInterpolation](xdat2[ ..., [1, 2]]) :
```

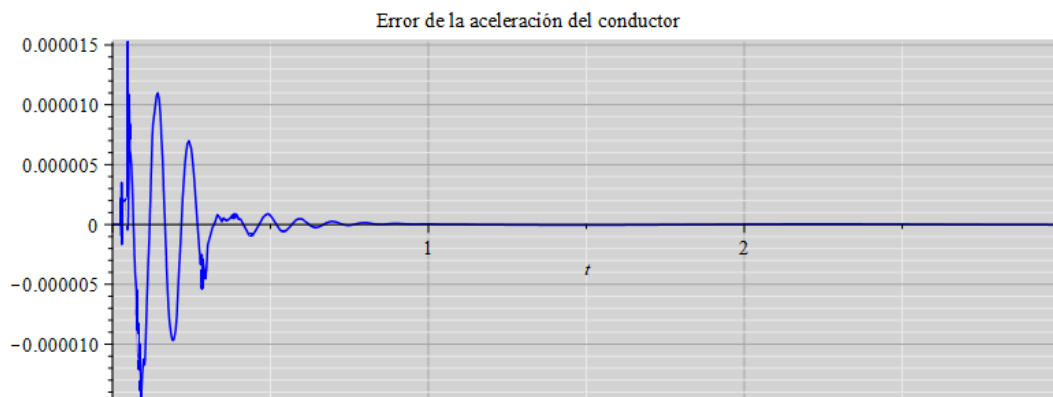
```
syms_th := Interpolation[CubicInterpolation](xdat2[ ..., [1, 3]]) :
```

```
sims_r1 := Interpolation[CubicInterpolation](xdat2[ ..., [1, 4]]) :
```

```
sims_r2 := Interpolation[CubicInterpolation](xdat2[ ..., [1, 5]]) :
```

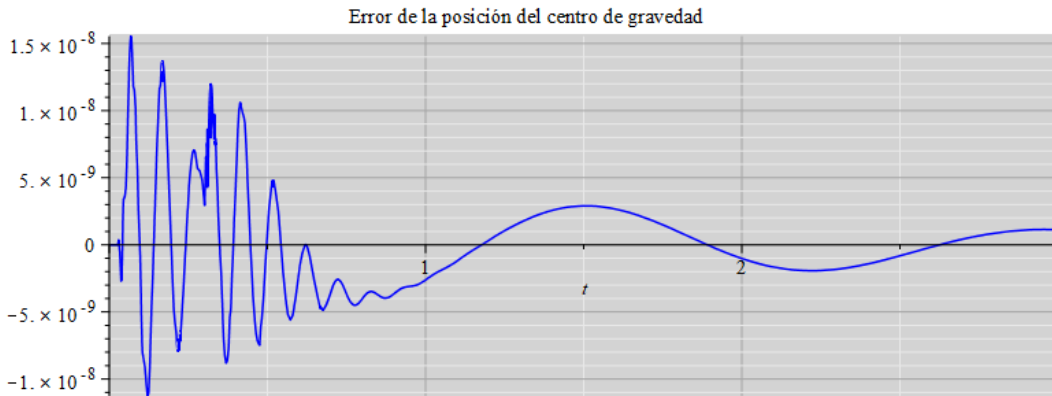
Error en la aceleración del conductor

```
plot(fad(t) - syms_ad(t), t=0..3, color=blue, gridlines, size=[800,300], title
="Error de la aceleración del conductor", background=LightGray)
```



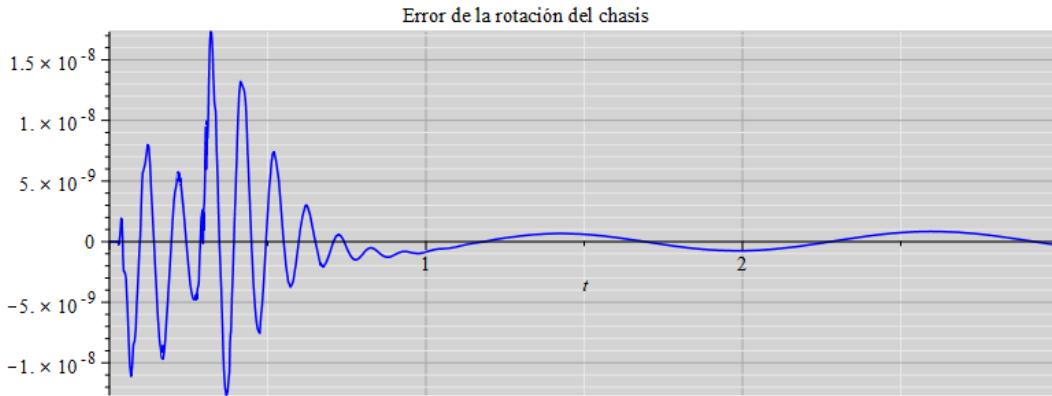
Error en la posición del centro de gravedad

```
p6 := plot(fy(t) - syms_y(t), t=0..3, color=blue, gridlines, size=[800,300], title="Error de la posición del centro de gravedad", background=LightGray)
```



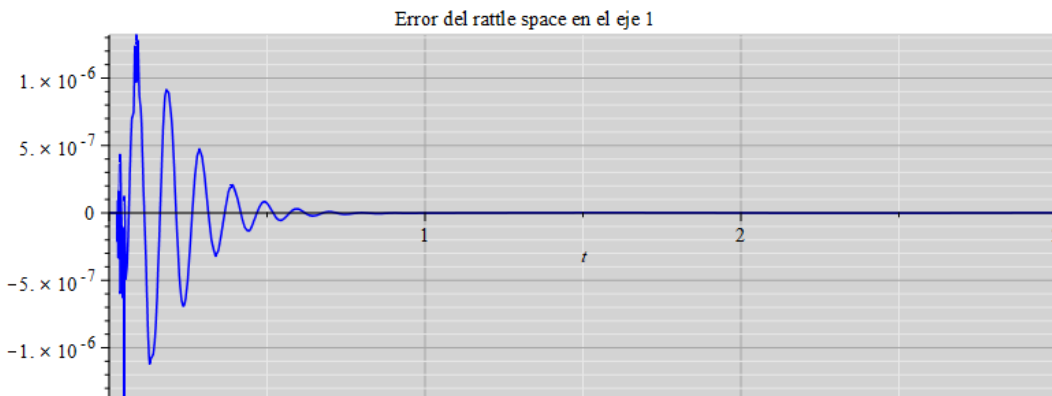
Error en la rotación del chasis

```
p7b := plot(ftheta(t) - syms_th(t), t=0..3, color=blue, linestyle=solid, gridlines, size=[800,300], title="Error de la rotación del chasis", background=LightGray)
```



Error en el rattle space del primer eje

```
p8 := plot(fr1(t) - sims_r1(t), t=0..3, color=blue, linestyle=solid, gridlines, size=[800,300], title="Error del rattle space en el eje 1", background=LightGray)
```



Error en el rattle space del segundo eje

```
p9 := plot(fr2(t) - sims_r2(t), t=0..3, color=red, linestyle=solid, gridlines, size=[800, 300], title
="Error del rattle space en el eje 2", background=LightGray)
```

