MÉTODOS DE SIMULACIÓN Y MODELADO

Solución al Trabajo Práctico - Enero de 2021

EJERCICIO 1

Lea el artículo citado a continuación, que puede descargar de la página web de la asignatura, y conteste a las preguntas.

> Åström, K.J., Elmqvist, H., Mattsson, S.E. *Evolution of continuous-time modeling and simulation*. The 12^{th} European Simulation Multiconference, ESM'98, June 16–19, 1998, Manchester, UK.

1. ¿Qué analogías pueden establecerse entre el modelado basado en diagramas de bloques y el paradigma de la simulación analógica?
2. ¿Qué es el paradigma de modelado físico? ¿Qué tipo de modelos matemáticos se obtienen de aplicar el paradigma del modelado físico?
3. ¿Qué diferencias hay entre el paradigma de la simulación analógica y el paradigma del modelado físico?
4. Observe la Figura 1.1 y comente su contenido basándose en el artículo.

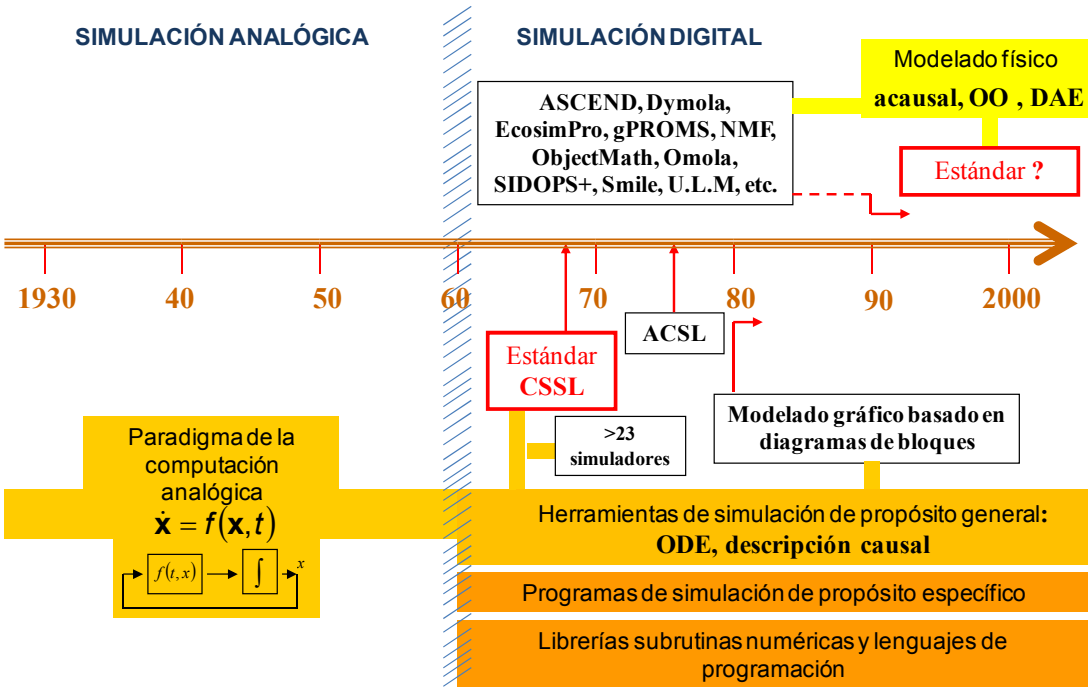


Figura 1.1: Evolución del modelado y simulación de tiempo continuo.

Solución al Ejercicio 1

En este ejercicio el alumno debe contestar, con sus propias palabras, a las cuestiones planteadas.

EJERCICIO 2

Consideremos el sistema mostrado en la Figura 1.2, que consiste en dos objetos de masa constante m_1 y m_2, dos muelles y un amortiguador. La pared lateral y el suelo se encuentran en reposo. El Objeto 1 desliza con rozamiento viscoso sobre el suelo, y se encuentra unido a la pared lateral mediante el primer muelle y el amortiguador. El Objeto 2 desliza con rozamiento viscoso sobre el Objeto 1. Un extremo del segundo muelle está unido al Objeto 1 y el otro extremo al Objeto 2.

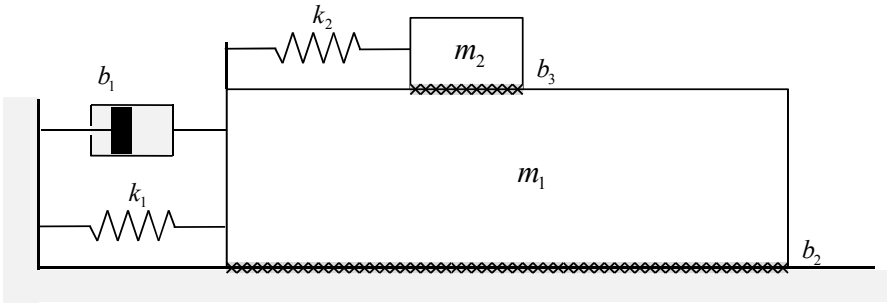


Figura 1.2: Diagrama del sistema mecánico.

El coeficiente del amortiguador (b_1), los coeficientes de fricción (b_2, b_3), los coeficientes de los muelles (k_1, k_2) y las masas (m_1, m_2) son parámetros de valor conocido.

$m_1 = 200$ kg	$m_2 = 10$ kg	$k_1 = k_2 = 1.2$ N/m
$b_1 = 1.8$ N·s/m	$b_2 = 3.2$ N·s/m	$b_3 = 1.4$ N·s/m

1. Escriba las ecuaciones del modelo del sistema mecánico. El modelo debe describir la evolución de la posición y la velocidad de los dos objetos.
2. Asigne la causalidad computacional. Indique cuántos grados de libertad tiene el modelo.
3. Escriba el diagrama de flujo del algoritmo para la simulación de este modelo. Emplee el método de integración de Euler explícito. La condición de finalización de la simulación es que el tiempo simulado alcance el valor 400 s.
4. Programe el algoritmo anterior en lenguaje R y ejecute la simulación. Represente gráficamente la posición de los dos objetos frente al tiempo y la velocidad de los dos objetos frente al tiempo. Explique qué criterio ha seguido para escoger el tamaño del paso de integración.

Solución al Ejercicio 2

Las ecuaciones del modelo se muestran a continuación. No se muestran las asignaciones de valor a las constantes y parámetros.

$$\begin{aligned}
\frac{dx_1}{dt} &= v_1 \\
m_1 \cdot \frac{dv_1}{dt} &= -b_1 \cdot v_1 - b_2 \cdot v_1 - b_3 \cdot (v_1 - v_2) - k_1 \cdot e_1 + k_2 \cdot e_2 \\
\frac{dx_2}{dt} &= v_2 \\
m_2 \cdot \frac{dv_2}{dt} &= = -b_3 \cdot (v_2 - v_1) - k_2 \cdot e_2 \\
\frac{de_1}{dt} &= v_1 \\
\frac{de_2}{dt} &= v_2 - v_1
\end{aligned}$$

Asumiendo que las variables que aparecen derivadas pueden ser seleccionadas como variables de estado, las variables del modelo pueden clasificarse de la forma siguiente:

– Parámetros y constantes: m_1, m_2, k_1, k_2, b_1, b_2, b_3
– Variables de estado: x_1, x_2, v_1, v_2, e_1, e_2
– Variables algebraicas:

Para asignar la causalidad computacional al modelo, se sustituyen las derivadas de las variables de estado por variables mudas:

$$\frac{dx_1}{dt} \rightarrow derx_1 \qquad \frac{dx_2}{dt} \rightarrow derx_2$$
$$\frac{dv_1}{dt} \rightarrow derv_1 \qquad \frac{dv_2}{dt} \rightarrow derv_2$$
$$\frac{de_1}{dt} \rightarrow dere_1 \qquad \frac{de_2}{dt} \rightarrow dere_2$$

Omitiendo las ecuaciones en las que se asigna valor a los parámetros y las constantes, se obtiene el modelo siguiente:

$$
\begin{aligned}
derx_1 &= v_1 \\
m_1 \cdot derv_1 &= -b_1 \cdot v_1 - b_2 \cdot v_1 - b_3 \cdot (v_1 - v_2) - k_1 \cdot e_1 + k_2 \cdot e_2 \\
derx_2 &= v_2 \\
m_2 \cdot derv_2 &= = -b_3 \cdot (v_2 - v_1) - k_2 \cdot e_2 \\
dere_1 &= v_1 \\
dere_2 &= v_2 - v_1
\end{aligned}
$$

Las variables del modelo pueden clasificarse en conocidas (el tiempo, los parámetros, las constantes y las variables de estado) y desconocidas (las variables algebraicas y las derivadas de las variables de estado):

– Conocidas: t
$m_1, m_2, k_1, k_2, b_1, b_2, b_3$
$x_1, x_2, v_1, v_2, e_1, e_2$
– Desconocidas: $derx_1, derx_2, derv_1, derv_2, dere_1, dere_2$

Con el fin de analizar si el modelo es *estructuralmente singular*, se comprueba que:

1. El número de ecuaciones y de variables desconocidas (incógnitas) es el mismo. Este modelo tiene 6 ecuaciones y 6 incógnitas.

2. Cada incógnita puede emparejarse con una ecuación en que aparezca y con la cual no se haya emparejado ya otra incógnita.

$$
\begin{aligned}
derx_1 &\rightarrow derx_1 = v_1 \\
derv_1 &\rightarrow m_1 \cdot derv_1 = -b_1 \cdot v_1 - b_2 \cdot v_1 - b_3 \cdot (v_1 - v_2) - k_1 \cdot e_1 + k_2 \cdot e_2 \\
derx_2 &\rightarrow derx_2 = v_2 \\
derv_2 &\rightarrow m_2 \cdot derv_2 = -b_3 \cdot (v_2 - v_1) - k_2 \cdot e_2 \\
dere_1 &\rightarrow dere_1 = v_1 \\
dere_2 &\rightarrow dere_2 = v_2 - v_1
\end{aligned}
$$

Aplicando el algoritmo de la partición, se obtiene el siguiente modelo ordenado y resuelto (existen otras posibles ordenaciones de las ecuaciones que son igualmente válidas):

$$
\begin{aligned}
[derx_1] &= v_1 \\
[derv_1] &= \frac{1}{m_1} \cdot (-b_1 \cdot v_1 - b_2 \cdot v_1 - b_3 \cdot (v_1 - v_2) - k_1 \cdot e_1 + k_2 \cdot e_2) \\
[derx_2] &= v_2 \\
[derv_2] &= \frac{1}{m_2} \cdot (-b_3 \cdot (v_2 - v_1) - k_2 \cdot e_2) \\
[dere_1] &= v_1 \\
[dere_2] &= v_2 - v_1
\end{aligned}
$$

El modelo posee seis variables de estado, por lo que tiene seis grados de libertad. En la Figura 1.3 se muestra el diagrama de flujo para la simulación del modelo, empleando el método de integración de Euler explícito y considerando como condición de finalización que el tiempo simulado alcance el valor 400 s. Se han asignado valores arbitrarios a los valores iniciales de las variables de estado.

Se ha seleccionado un tamaño del paso igual a 0.01 s. Para ello, se ha repetido la simulación empleando diferentes tamaños del paso, encontrándose que el error cometido escogiendo 0.01 s es admisible para el propósito de este estudio.

Se muestra a continuación el código R que implementa el algoritmo mostrado en la Figura 1.3, con un intervalo de comunicación de 0.5 s. En la Figura 1.4 se muestra la gráfica obtenida al ejecutar el código R.

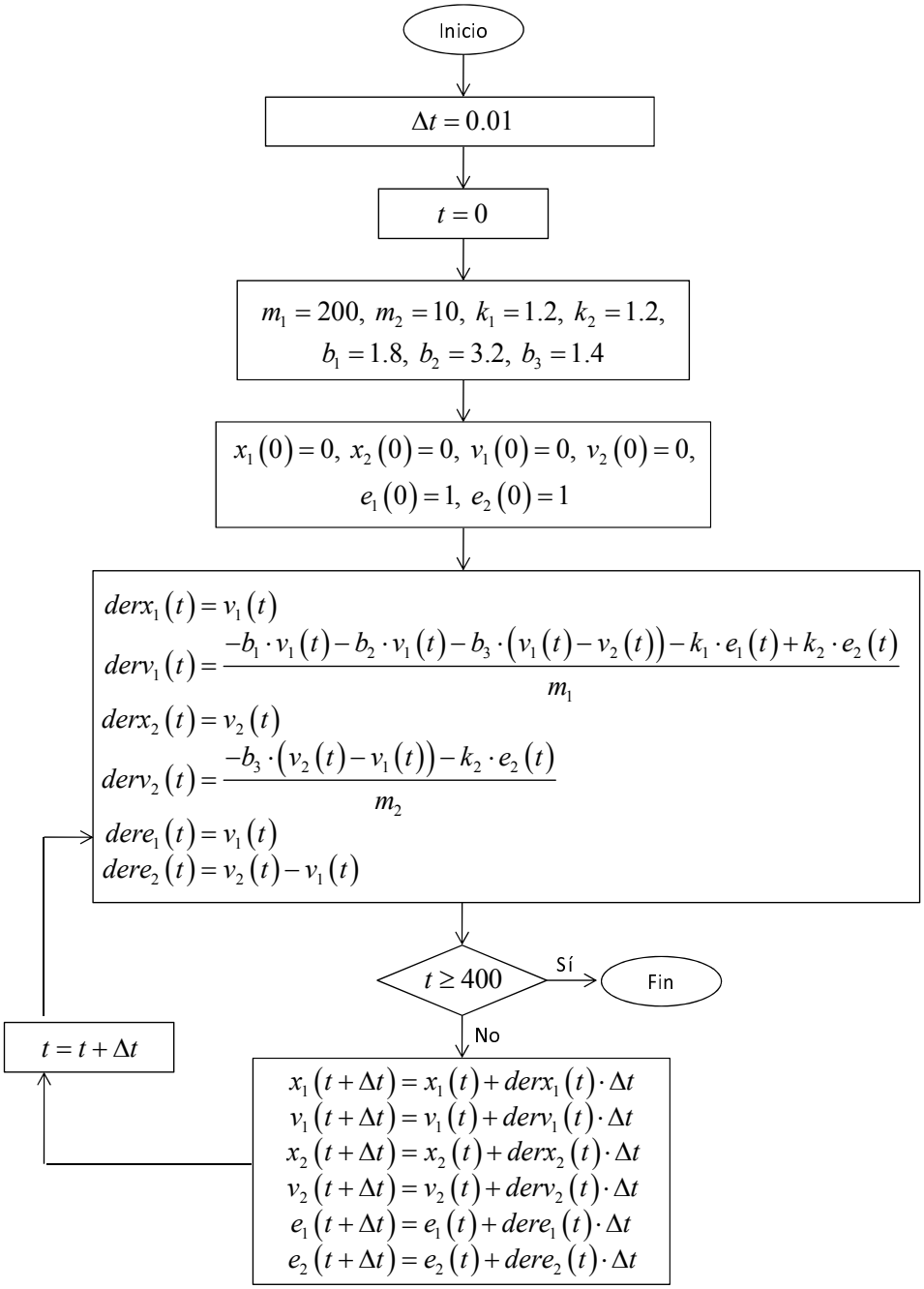


Figura 1.3: Diagrama de flujo de la simulación del modelo.

Código R del algoritmo de la simulación

```
t_fin <- 400   # Valor final del tiempo
h     <- 0.01  # Tamaño del paso de integración
t_com <- 0.5   # Intervalo de comunicación
# Parámetros
m1 <- 200
m2 <- 10
k1 <- 1.2
k2 <- 1.2
b1 <- 1.8
b2 <- 3.2
b3 <- 1.4
# Inicialización del tiempo
t <- 0
# Valor inicial del estado
x1 <- 0
x2 <- 0
v1 <- 0
v2 <- 0
e1 <- 1
e2 <- 1
# Inicialización resultados salida
t_plot      <- numeric(0)
x1_plot     <- numeric(0)
x2_plot     <- numeric(0)
v1_plot     <- numeric(0)
v2_plot     <- numeric(0)
t_ultimaCom <- -Inf

# Bucle de la simulación
termina <- FALSE
while ( !termina ) {
   termina <- (t + h/2 > t_fin)
   # Cálculo variables algebraicas en t
   derx1 <- v1
   derv1 <- (-b1*v1 -b2*v1 -b3*(v1-v2) -k1*e1 +k2*e2) / m1
   derx2 <- v2
   derv2 <- (-b3*(v2-v1) -k2*e2) / m2
   dere1 <- v1
   dere2 <- v2 - v1
   # Almacenar resultados de salida
   if ( t - t_ultimaCom + h/2 > t_com | termina ) {
      t_plot  <- c(t_plot, t)
      x1_plot <- c(x1_plot, x1)
      x2_plot <- c(x2_plot, x2)
      v1_plot <- c(v1_plot, v1)
      v2_plot <- c(v2_plot, v2)
      t_ultimaCom <- t
   }
   # Valor en t+h de los estados
   x1 <- x1 + h*derx1
   x2 <- x2 + h*derx2
   v1 <- v1 + h*derv1
   v2 <- v2 + h*derv2
   e1 <- e1 + h*dere1
   e2 <- e2 + h*dere2
   # Actualización del valor del tiempo
   t <-  t + h
}
```

```
# Representación gráfica
# Para poder usar minor.tick es necesario instalar el paquete Hmisc
# Configuración de la representación gráfica

par( mfrow = c(2,1) )
# x1 vs t
par( mai = c(0.4,1,0.8,0.2), cex = 1.2 )
plot(t_plot, x1_plot, lty = 1, type = "l",
     xlim = c(0,t_fin), ylim = c(-2,0.5),
     yaxt = "n", xlab = "", ylab = "x1 [m]", lwd = 2 )
axis(2, las = 2)
minor.tick(nx = 2, ny = 5, tick.ratio = 0.5)
abline(v = seq(0,t_fin,10*t_com), h = seq(-2,0.5,0.1),
       lwd = 0.5, lty = 3)
abline(h = 0, lwd = 1, lty = 1)
title("Posiciones", cex = 1)

# x2 vs t
par( mai = c(1,1,0.2,0.2), cex = 1.2 )
plot(t_plot, x2_plot, lty = 1, type = "l",
     xlim = c(0,t_fin), ylim = c(-3,0.5),
     yaxt = "n", xlab = "t [s]", ylab = "x2 [m]", lwd = 2 )
axis(2, las = 2)
minor.tick(nx = 2, ny = 5, tick.ratio = 0.5)
abline(v = seq(0,t_fin,10*t_com), h = seq(-3,0.5,0.1),
       lwd = 0.5, lty = 3)
abline(h = 0, lwd = 1, lty = 1)

dev.new()
par( mfrow = c(2,1) )
# v1 vs t
par( mai = c(0.4,1,0.8,0.2), cex = 1.2 )
plot(t_plot, v1_plot, lty = 1, type = "l",
     xlim = c(0,t_fin), ylim = c(-0.08,0.06),
     yaxt = "n", xlab = "", ylab = "v1 [m/s]", lwd = 2)
axis(2, las = 2)
minor.tick(nx = 2, ny = 5, tick.ratio = 0.5)
abline(v = seq(0,t_fin,10*t_com), h = seq(-0.08,0.06,0.004),
       lwd = 0.5, lty=3)
abline(h = 0, lwd = 1, lty=1)
title("Velocidades", cex = 1)

# v2 vs t
par( mai = c(1,1,0.2,0.2), cex = 1.2 )
plot(t_plot, v2_plot, lty = 1, type = "l",
     xlim = c(0,t_fin), ylim = c(-0.3,0.1),
     yaxt = "n", xlab = "t [s]", ylab = "v2 [m/s]", lwd = 2)
axis(2, las = 2)
minor.tick(nx = 2, ny = 5, tick.ratio = 0.5)
abline(v = seq(0,t_fin,10*t_com), h = seq(-0.3,0.1,0.02),
       lwd = 0.5, lty=3)
abline(h = 0, lwd = 1, lty=1)
```

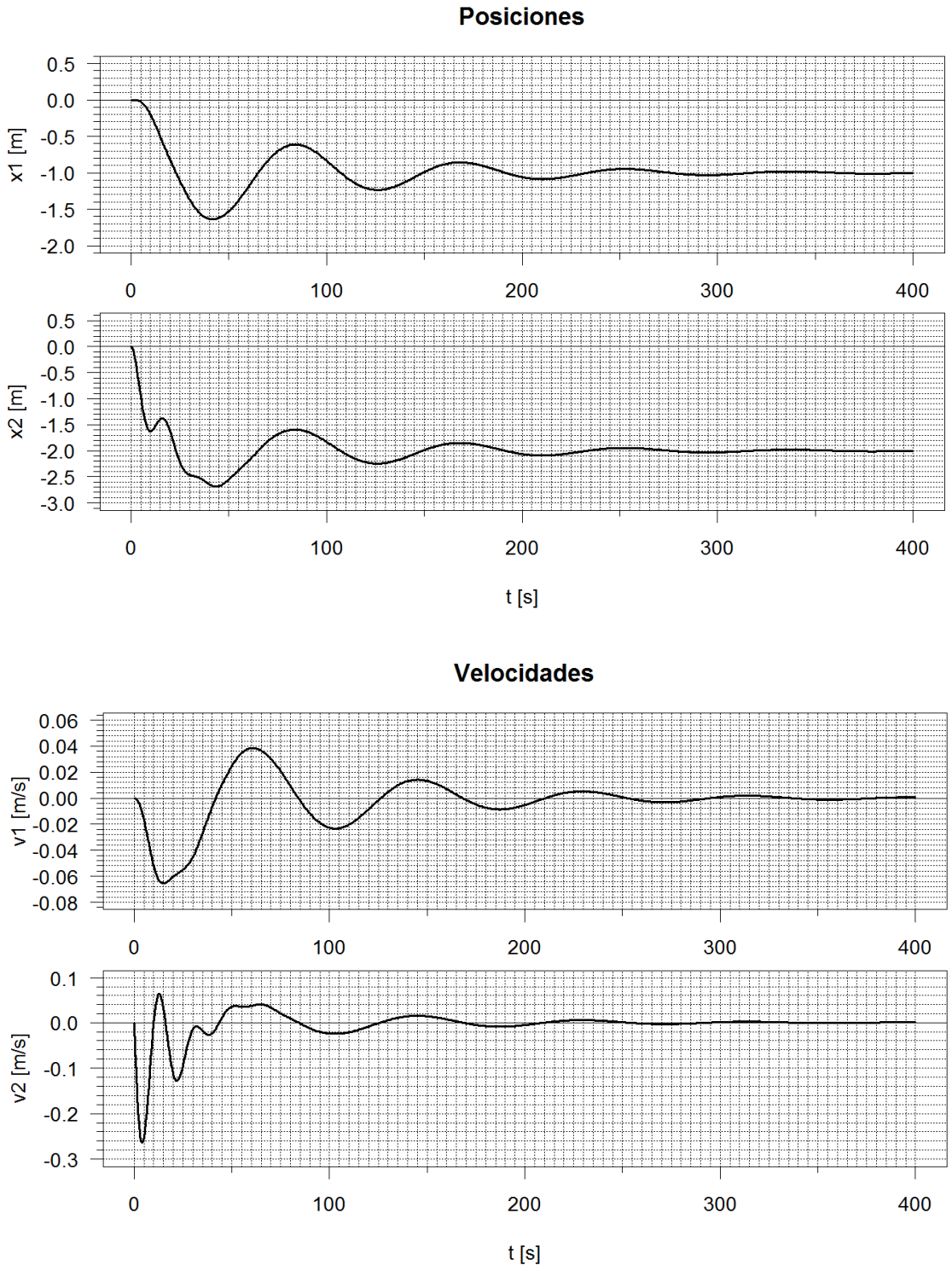


Figura 1.4: Resultado de la ejecución del código R.

EJERCICIO 3

Describa en lenguaje Modelica el sistema mecánico mostrado en la Figura 1.2 de las dos maneras siguientes:

1. Como un modelo atómico, que viene descrito por las ecuaciones que usted ha planteado al contestar a la pregunta anterior.
2. Programe una librería mecánica que contenga los componentes necesarios para componer el sistema mecánico de la Figura 1.2. A continuación, defina dicho sistema como un modelo compuesto, instanciando y conectando componentes de la librería que ha creado.

Asigne a los parámetros los valores indicados en el ejercicio anterior y simule los dos modelos anteriores durante 400 s. Represente gráficamente las posiciones y velocidades de los objetos frente al tiempo, mostrando las gráficas en la memoria.

Solución al Ejercicio 3

El modelo atómico en lenguaje Modelica se muestra en Código 1.1. El resultado de la simulación está representado en la Figura 1.5.

En el Código 1.2 y 1.3 se muestra una posible forma de modelar los componentes, organizándolos en una librería, y de componer el sistema mediante dichos componentes.

```
model DosObjetosDeslizantes
  import SI = Modelica.SIunits;
  parameter SI.Mass m1=200 "Masa del objeto 1";
  parameter SI.Mass m2=10  "Masa del objeto 2";
  parameter SI.TranslationalDampingConstant b1=1.8 "Amortiguador";
  parameter SI.TranslationalDampingConstant b2=3.2 "Obj.1 - Suelo";
  parameter SI.TranslationalDampingConstant b3=1.4 "Obj.1 - Obj.2";
  parameter SI.TranslationalSpringConstant k1=1.2 "Muelle 1";
  parameter SI.TranslationalSpringConstant k2=1.2 "Muelle 2";
  SI.Position x1(start=0,fixed=true) "Posicion del objeto 1";
  SI.Position x2(start=0,fixed=true) "Posicion del objeto 2";
  SI.Velocity v1(start=0,fixed=true) "Velocidad del objeto 1";
  SI.Velocity v2(start=0,fixed=true) "Velocidad del objeto 2";
  SI.Length e1(start=1,fixed=true) "Muelle 1";
  SI.Length e2(start=1,fixed=true) "Muelle 2";
equation
  // Objeto 1
  der(x1) = v1;
  m1*der(v1) = - b1*v1 - b2*v1 - b3*(v1 - v2) - k1*e1 + k2*e2;
  // Objeto 2
  der(x2) = v2;
  m2*der(v2) = - b3*(v2 - v1) - k2*e2;
  // Muelle 1
  der(e1) = v1;
  // Muelle 2
  der(e2) = v2-v1;
end DosObjetosDeslizantes;
```

Código 1.1: Ejercicio 3.1: modelo atómico del sistema.

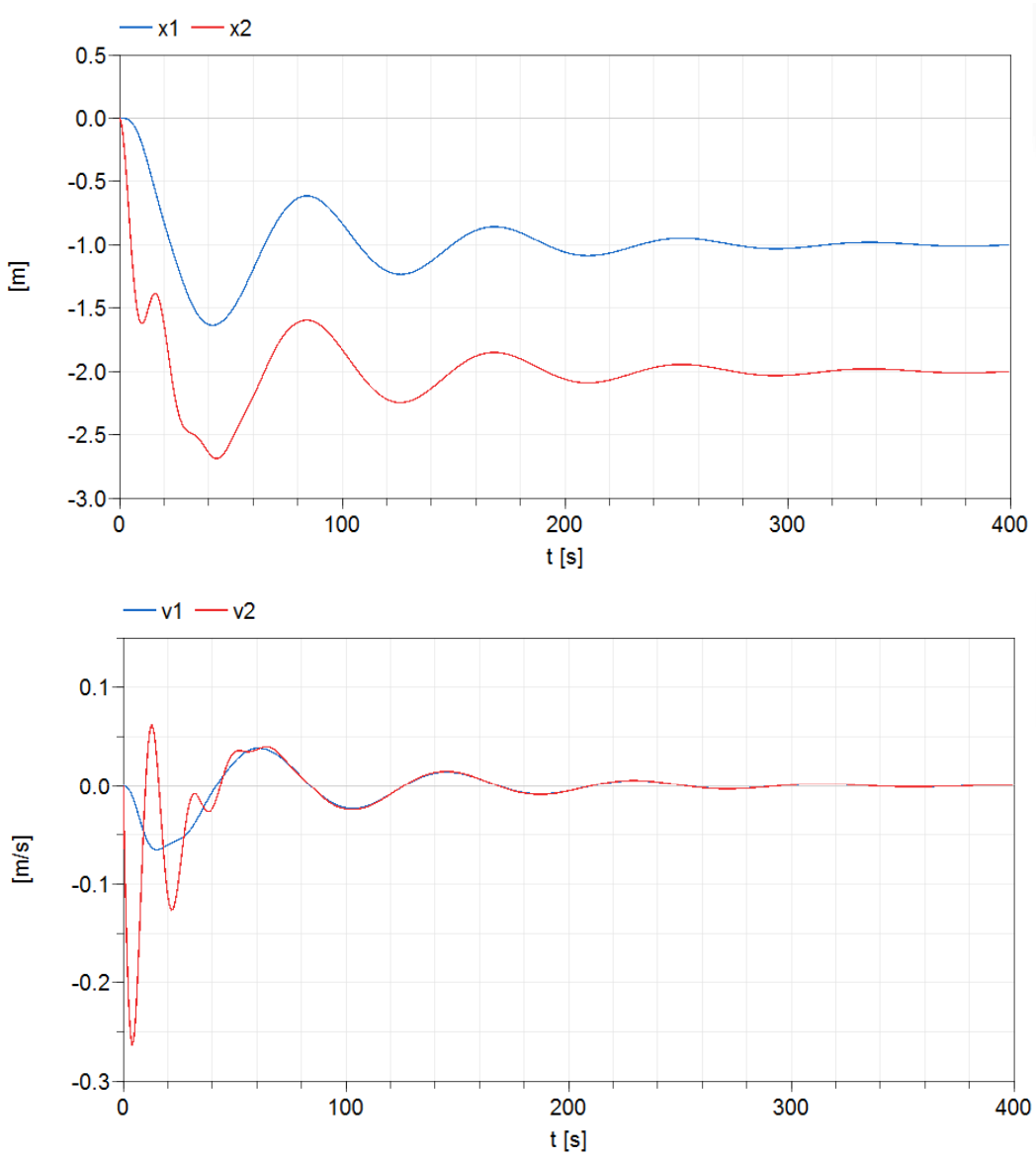


Figura 1.5: Resultado de la simulación usando Dymola.

```
package libMec1D "Libreria mecanica 1D"
import SI = Modelica.SIunits;

package Interfaz
  connector v_f "Velocidad / Fuerza"
     SI.Velocity v;
     flow SI.Force f;
  end v_f;
end Interfaz;

package Componentes

  model Muelle
    // (coord espacial p1) < (coord espacial p2)
    Interfaz.v_f p1;
    Interfaz.v_f p2;
    parameter SI.TranslationalSpringConstant k = 1;
    SI.Length e(start=eInicial,fixed=eInicialFixed) "Elongacion";
    parameter SI.Length eInicial "Elongacion inicial";
    parameter Boolean  eInicialFixed = true;
  equation
    der(e) = p2.v - p1.v;
    p1.f = k*e;
    p2.f = -p1.f;
  end Muelle;

  model Masa
    Interfaz.v_f p;
    parameter SI.Mass m=1 "Masa";
    SI.Position x(start=xInicial,fixed=xInicialFixed);
    parameter SI.Position xInicial=0;
    parameter Boolean xInicialFixed=true;
  equation
    -p.f = m*der(p.v);
    der(x) = p.v;
  end Masa;

  model Reposo
    Interfaz.v_f p;
  equation
    p.v = 0;
  end Reposo;

  model Amortiguador
    // (coord espacial p1) < (coord espacial p2)
    Interfaz.v_f p1;
    Interfaz.v_f p2;
    parameter SI.TranslationalDampingConstant b=1;
  equation
    p1.f = b*(p2.v - p1.v);
    p2.f = -p1.f;
  end Amortiguador;

end Componentes;
```

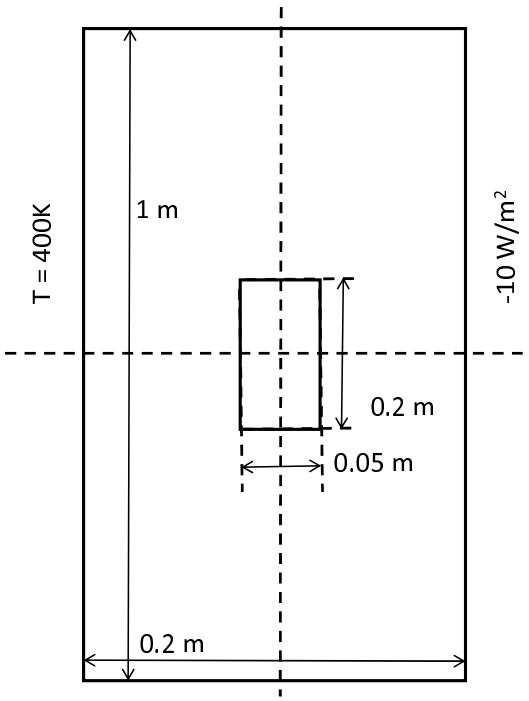
Código 1.2: Ejercicio 3.2: librería (1/2).

```
model SistemaMecanico
  import libMec1D.Componentes.*;
  parameter SI.Mass m1=200 "Masa del objeto 1";
  parameter SI.Mass m2=10  "Masa del objeto 2";
  parameter SI.TranslationalDampingConstant b1=1.8 "Amortiguador";
  parameter SI.TranslationalDampingConstant b2=3.2 "Obj.1 - Suelo";
  parameter SI.TranslationalDampingConstant b3=1.4 "Obj.1 - Obj.2";
  parameter SI.TranslationalSpringConstant k1=1.2 "Muelle 1";
  parameter SI.TranslationalSpringConstant k2=1.2 "Muelle 2";
  Reposo pared;
  Reposo suelo;
  Masa  masa1(m=m1);
  Masa  masa2(m=m2);
  Amortiguador amortiguador(b=b1);
  Amortiguador friccion_m1_suelo(b=b2);
  Amortiguador friccion_m2_m1(b=b3);
  Muelle muelle1(k=k1,eInicial=1);
  Muelle muelle2(k=k2,eInicial=1);
equation
  connect(pared.p,amortiguador.p1);
  connect(pared.p,muelle1.p1);

  connect(amortiguador.p2,masa1.p);
  connect(muelle1.p2,masa1.p);

  connect(suelo.p,friccion_m1_suelo.p1);
  connect(masa1.p,friccion_m1_suelo.p2);


  connect(masa1.p,muelle2.p1);
  connect(masa2.p,muelle2.p2);

  connect(masa1.p,friccion_m2_m1.p1);
  connect(masa2.p,friccion_m2_m1.p2);

end SistemaMecanico;

end libMec1D;
```

Código 1.3: Ejercicio 3.2: librería (2/2).

EJERCICIO 4

Consideremos el problema bidimensional de la distribución de calor en un bloque rectangular de un determinado material que tiene un hueco rectangular en su centro. El bloque rectangular tiene dos lados de longitud 0.2 m y 1 m. El hueco interior tiene forma rectangular y sus lados miden 0.05 m y 0.2 m. La geometría de este problema se muestra en figura siguiente.



El fenómeno térmico que tiene lugar en el bloque se describe por la ecuación diferencial escrita a continuación, donde $r_{cp} = 1$ en unidades del SI. La conductividad térmica del material (κ) es 1 en unidades del SI.

$$\nabla \cdot (-\kappa \cdot \nabla temp) + r_{cp} \cdot \frac{\partial temp}{\partial t} = 0$$

El lado izquierdo del bloque (véase la figura) se mantiene a una temperatura constante de 400 K. Existe una pérdida de calor en el lado derecho a una tasa constante de 10 W/m^2. El resto de lados están bien aislados. La temperatura inicial (en $t = 0$) del bloque es de 273 K.

Escriba un *script* en FlexPDE para obtener, en el instante $t = 5$ s, los dos gráficos siguientes: un gráfico con las curvas de nivel de la temperatura y un gráfico vectorial del flujo de calor.

Solución al Ejercicio 4

El *script* está a continuación. El gráfico de la temperatura se muestra en la Figura 1.6 y el gráfico vectorial del flujo de calor en la Figura 1.7.

```
TITLE 'Bloque con cavidad'  { Flujo de calor en un bloque con cavidad interior }
SELECT
  errlim=1e-3 spectral_colors
VARIABLES temp
DEFINITIONS
  Lx=0.1 Ly = 0.5 Cx = 0.025 Cy = 0.1
  rcp = 1
  k = 1
  heat = 0
  fluxd_x = -k*dx(temp) fluxd_y=-k*dy(temp)
  fluxd=vector(fluxd_x,fluxd_y)
INITIAL VALUES
  temp = 273
EQUATIONS
  div( fluxd)+ rcp*dt( temp)=heat
BOUNDARIES
region 'dominio'  {Define el dominio del problema}
  START (-Lx,Ly) VALUE(temp) = 400
      LINE TO (-Lx,-Ly)    NATURAL(temp) = 0
      LINE TO (Lx, -Ly) NATURAL(temp) = 10
      LINE TO (Lx, Ly)      NATURAL(temp)= 0  LINE TO CLOSE
 START(-Cx,Cy)      LINE TO (-Cx,-Cy) {Region excluida}
      line to (Cx,-Cy)          LINE TO (Cx, Cy)          LINE TO CLOSE
TIME FROM 0 TO 5 by 0.05
PLOTS
FOR T =  5
CONTOUR(temp)
VECTOR(fluxd)
end
END
```

Código 1.4: Ejercicio 4: modelo en FlexPDE.

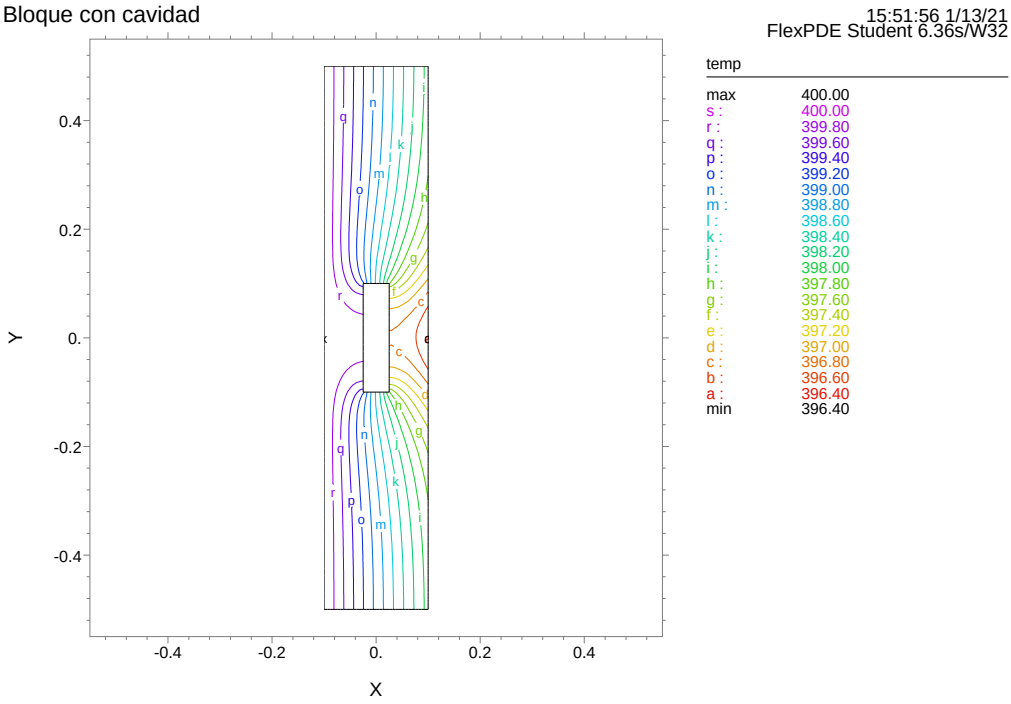


Figura 1.6: Curvas de temperatura.

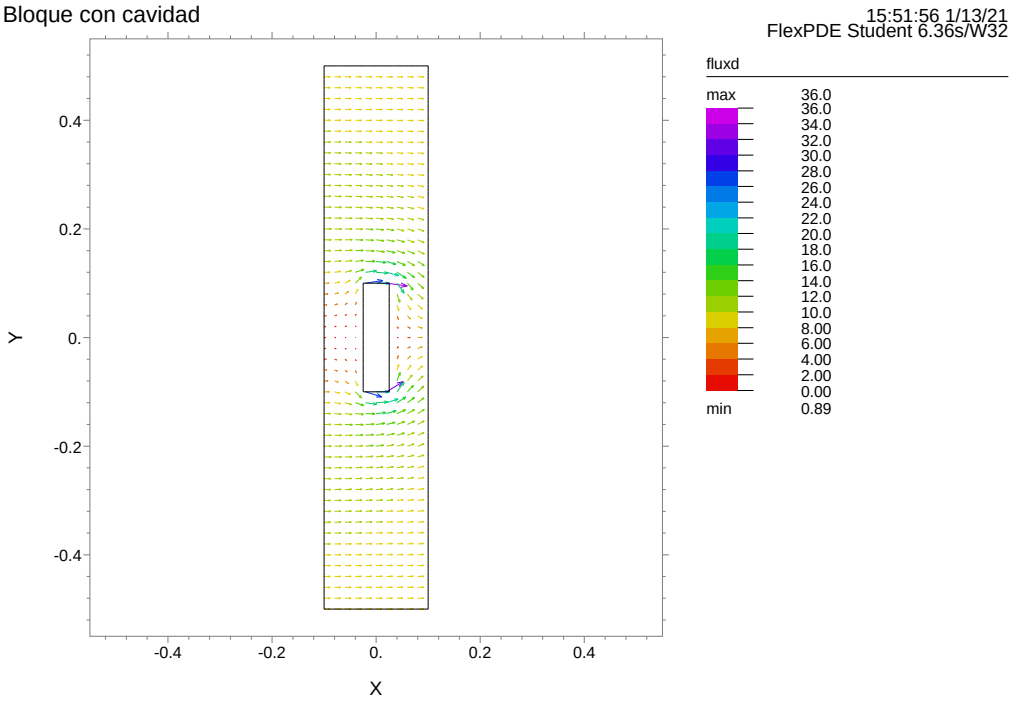


Figura 1.7: Flujo de calor