Instituto de Estudios de Posgrado

Universidad de Córdoba

Máster Universitario en Inteligencia Computacional e Internet de las Cosas

# Modelado y simulación de sistemas dinámicos simples

## Fundamentos y herramientas para la modelización de procesos técnicos-científicos de investigación

**Autora**:
Alba Márquez-Rodríguez

**Profesora**:
María Jesús Aguilera Ureña

Córdoba, Mayo 2024

# Índice

# 1. Introducción

En este informe se presenta el modelado y simulación de sistemas dinámicos simples. Se explicarán las actividades realizadas, los parámetros utilizados, y se incluirán capturas de pantalla y los códigos utilizados en MATLAB/Simulink.

Los sistemas dinámicos son aquellos cuya evolución en el tiempo está determinada por un conjunto de ecuaciones diferenciales. Estos sistemas son fundamentales en diversas áreas de la ingeniería, como la automatización, la robótica, la biomedicina, entre otras. Modelar y simular estos sistemas permite predecir su comportamiento bajo diferentes condiciones y diseñar controladores adecuados para su operación.

# 2. Simulación del Tiro Parabólico

En esta sección se presentan las simulaciones realizadas para el tiro parabólico. Se indicarán los parámetros utilizados, los resultados obtenidos y se incluirán capturas de pantalla.

## 2.1. Solución Numérica

Los parámetros utilizados para la simulación numérica fueron:

- Aceleración debido a la gravedad ($g$): 9.81 m/s$^2$
- Masa ($m$): 1 kg
- Velocidad inicial ($v_0$): 50 m/s
- Paso de integración ($\Delta t$): 0.01 s

El siguiente código MATLAB fue utilizado para la solución numérica:

```
clear
figure(1)

% Variables
r = [];
v = [];
F = [];
x = [];
y = [];

% Par metros
g = [0; -9.81]; % Aceleraci n debido a la gravedad (m/s^2)
m = 1; % Masa del proyectil (kg)
h = 0.01; % Paso de integraci n

% Condiciones iniciales
r0 = [0; 0]; % Posici n inicial (m)
v0 = [1; 4]; % Velocidad inicial (m/s)

% Variable externa
F = m * g; % Fuerza

% Inicializaci n de variables
r = r0;
v = v0;

% Integraci n num rica
for step = 1:100
    plot(r(1), r(2), 'ob');
    title(['Paso: ' num2str(step)]);
    axis([0 2 -1 1]);
    set(gca, 'DataAspectRatio', [1 1 1]);
    pause(0.01);

    x = [x, r(1)];
    y = [y, r(2)];
```

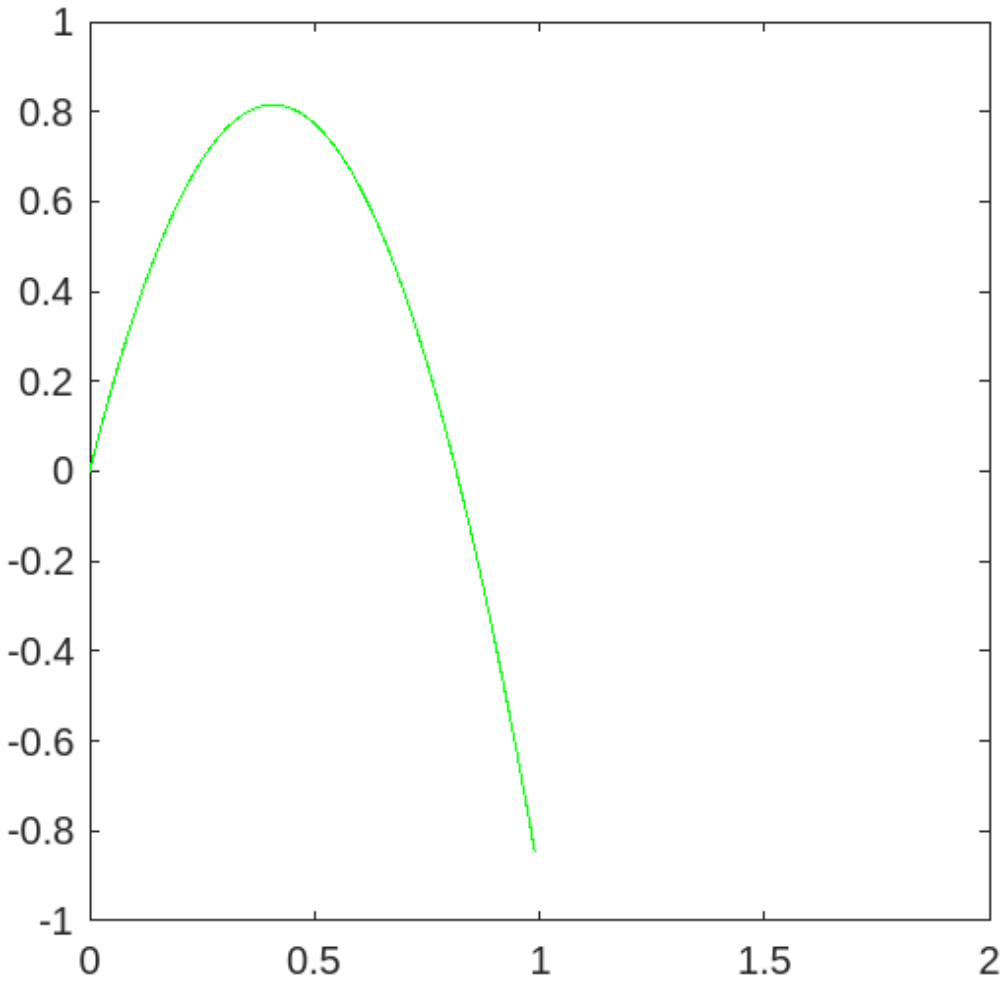


Figura 1: Trayectoria del tiro parabólico (Solución numérica)

```
37
38      % Guardar valor anterior
39      ra = r;
40      va = v;
41
42      % Paso integraci n
43      vpm = va + (h / 2) * (F / m);
44      rpm = ra + (h / 2) * va;
45
46      v = va + h * (F / m);
47      r = ra + h * vpm;
48  end
49
50  % Graficar la trayectoria
51  figure(2);
52  plot(x, y, 'g');
53  axis([0 2 -1 1]);
54  set(gca, 'DataAspectRatio', [1 1 1]);
55  title('Trayectoria del tiro parab lico (Soluci n num rica)');
56  xlabel('Distancia horizontal (m)');
57  ylabel('Altura (m)');
58  grid on;
```

Listing 1: Código MATLAB para la solución numérica del tiro parabólico

## 2.2. Solución Numérica Modificada

Los parámetros utilizados para la simulación numérica modificada fueron:

- Aceleración debido a la gravedad ($g$): 9.81 m/s$^2$
- Masa ($m$): 2 kg
- Velocidad inicial ($v_0$): 8 m/s
- Paso de integración ($\Delta t$): 0.01 s

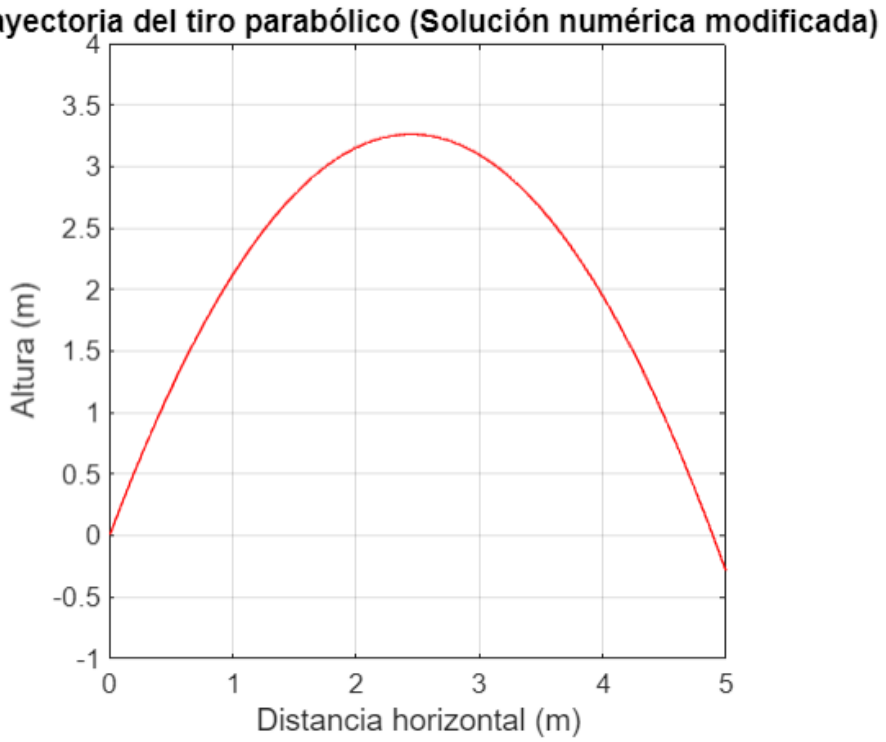


Figura 2: Trayectoria del tiro parabólico (Solución numérica modificada)

El siguiente código MATLAB fue utilizado para la solución numérica modificada:

```
clear
figure(1)

% Variables
r = [];
v = [];
F = [];
x = [];
y = [];

% Par metros modificados
g = [0; -9.81]; % Aceleraci n debido a la gravedad (m/s^2)
m = 2; % Masa del proyectil (kg)
h = 0.01; % Paso de integraci n

% Condiciones iniciales modificadas
r0 = [0; 0]; % Posici n inicial (m)
v0 = [3; 8]; % Velocidad inicial (m/s)

% Variable externa
F = m * g; % Fuerza

% Inicializaci n de variables
```

```
r = r0;
v = v0;

% Integraci n num rica
for step = 1:200
    plot(r(1), r(2), 'ob');
    title(['Paso: ' num2str(step)]);
    axis([0 5 -1 4]);
    set(gca, 'DataAspectRatio', [1 1 1]);
    pause(0.01);

    x = [x, r(1)];
    y = [y, r(2)];

    % Guardar valor anterior
    ra = r;
    va = v;

    % Paso integraci n
    vpm = va + (h / 2) * (F / m);
    rpm = ra + (h / 2) * va;

    v = va + h * (F / m);
    r = ra + h * vpm;
end

% Graficar la trayectoria
figure(2);
plot(x, y, 'r');
axis([0 5 -1 4]);
set(gca, 'DataAspectRatio', [1 1 1]);
title('Trayectoria del tiro parab lico (Soluci n num rica modificada)');
xlabel('Distancia horizontal (m)');
ylabel('Altura (m)');
grid on;
```

Listing 2: Código MATLAB para la solución numérica modificada del tiro parabólico

## 2.3. Solución Analítica

Los parámetros utilizados para la simulación analítica fueron los mismos que los utilizados en la simulación numérica modificada:

- Aceleración debido a la gravedad ($g$): 9.81 m/s$^2$
- Masa ($m$): 2 kg
- Velocidad inicial ($v_0$): 8 m/s
- Paso de integración ($\Delta t$): 0.01 s

El siguiente código MATLAB fue utilizado para la solución analítica:

```
clear
figure(1)

% Variables
r = [];
x = [];
y = [];

% Par metros modificados
g = [0; -9.81]; % Aceleraci n debido a la gravedad (m/s^2)
m = 2; % Masa del proyectil (kg)
h = 0.01; % Paso de integraci n

% Condiciones iniciales modificadas
ro = [0; 0]; % Posici n inicial (m)
vo = [3; 8]; % Velocidad inicial (m/s)
```

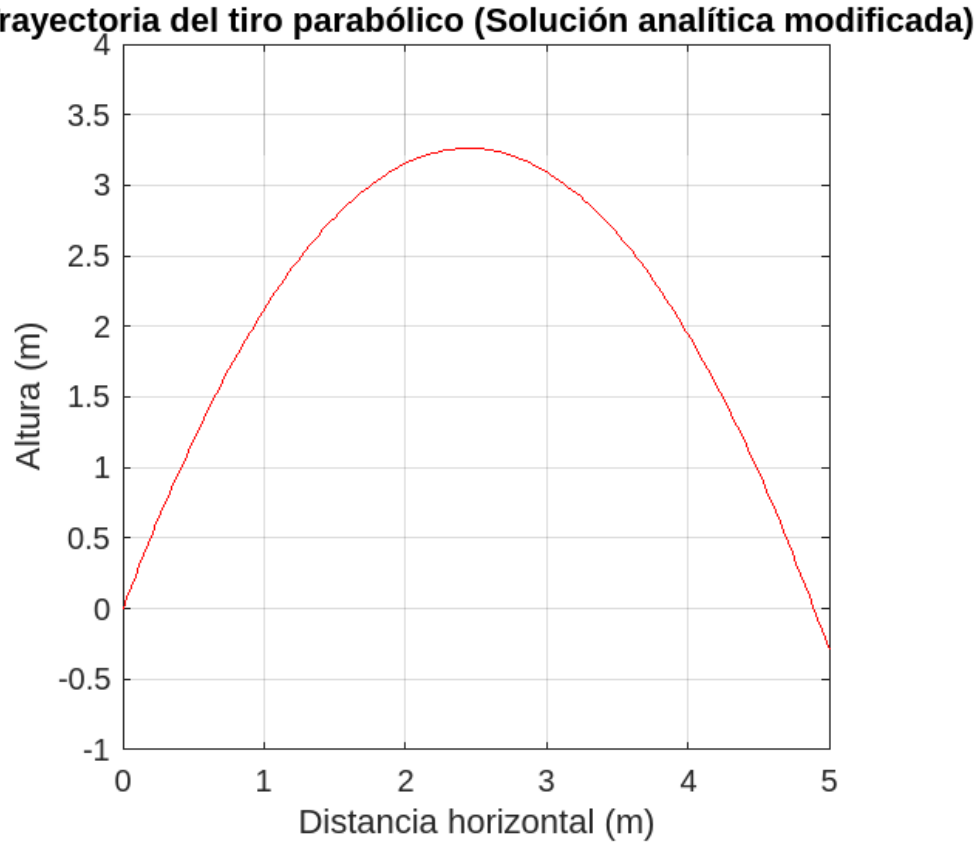


Figura 3: Trayectoria del tiro parabólico (Solución analítica)

```
17
18 % Inicializa posici n
19 r = ro;
20
21 % Integraci n num rica
22 for step = 1:200
23     plot(r(1), r(2), 'ob');
24     title(['Paso: ' num2str(step)]);
25     axis([0 5 -1 4]);
26     set(gca, 'DataAspectRatio', [1 1 1]);
27     pause(0.01);
28
29     x = [x, r(1)];
30     y = [y, r(2)];
31
32     % Calculo posici n para instante t
33     t = step * h;
34     r = ro + vo * t + g * t * t / 2;
35 end
36
37 % Graficar la trayectoria
38 figure(2);
39 plot(x, y, 'r');
40 axis([0 5 -1 4]);
41 set(gca, 'DataAspectRatio', [1 1 1]);
42 title('Trayectoria del tiro parab lico (Soluci n anal tica)');
43 xlabel('Distancia horizontal (m)');
44 ylabel('Altura (m)');
45 grid on;
```

Listing 3: Código MATLAB para la solución analítica del tiro parabólico

Comparando las gráficas obtenidas de la solución numérica y la solución analítica, se puede observar que las trayectorias son idénticas cuando se utilizan los mismos parámetros. Esto confirma la precisión del método numérico de integración utilizado para la simulación del tiro parabólico.

## 2.4. Conclusiones

En esta sección se ha presentado el modelado y simulación del tiro parabólico utilizando MATLAB. Se han realizado simulaciones tanto numéricas como analíticas para obtener la trayectoria del proyectil bajo diferentes condiciones. Las simulaciones numéricas fueron realizadas y comparados con la solución analítica para verificar su precisión.

Los resultados muestran que la solución numérica proporciona una buena aproximación a la solución analítica, siempre que se utilice un paso de integración suficientemente pequeño. La simulación del tiro parabólico es una herramienta útil para comprender el comportamiento de los sistemas dinámicos y su evolución en el tiempo, así como para validar métodos numéricos de integración.

En futuras simulaciones, se podrían considerar otros factores como la resistencia del aire, variaciones en la aceleración debido a la gravedad, o la inclusión de fuerzas adicionales para obtener resultados más realistas y complejos.

# 3. Movimiento Pendular con Fricción

En esta sección se presentan las simulaciones realizadas para el movimiento pendular con fricción. Se indicarán los parámetros utilizados, los resultados obtenidos y se incluirán capturas de pantalla.

## 3.1. Solución Numérica

Los parámetros utilizados para la simulación numérica fueron:

- Masa ($m$): 1 kg
- Aceleración debido a la gravedad ($g$): 9.81 m/s$^2$
- Longitud del péndulo ($L$): 1 m
- Constante de fricción viscosa ($C$): 0.75
- Paso de integración ($\Delta t$): 0.01 s
- Ángulo inicial ($\theta_0$): 30º

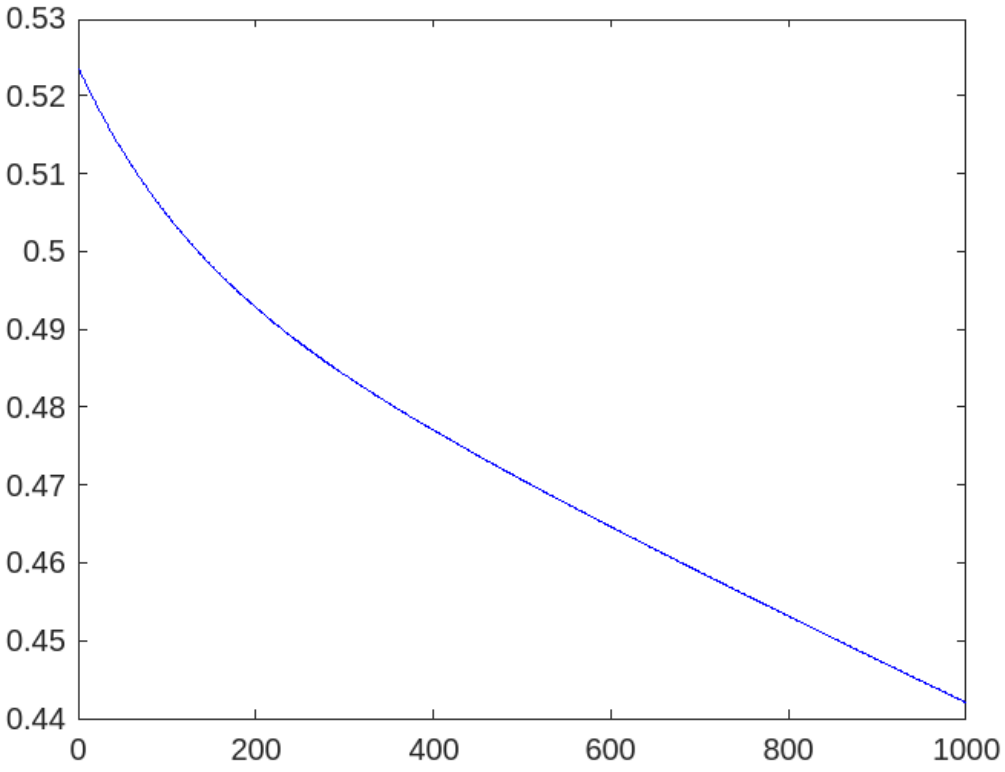


Figura 4: Trayectoria del péndulo (Solución numérica)

El siguiente código MATLAB fue utilizado para la solución numérica del movimiento pendular con fricción:

```
clc;
clear;

figure(1)
theta_graf = [];

% Variables
theta=[];
w=[];
pos=[];

% Par metros
```

```
m = 1;
g = 9.81;
L = 1; % Longitud del p ndulo
C=0.75; % Constante de fricci n viscosa
h=0.01;

% Condiciones iniciales
theta_0 = 30*(pi/180);
w_0 = 0/L;

% Inicializaci n del movimiento
theta = theta_0;
w = w_0;
pos = [L*sin(theta);-L*cos(theta)];
alpha = -(L*w*C+m*g*sin(theta))/(L*m);

for step=1:1000
    hold off;
    plot(pos(1),pos(2),'o','MarkerFaceColor','b','MarkerSize',10);
    hold on;
    plot([0;pos(1)],[0;pos(2)]);

    title(['paso: ' num2str(step)]);
    axis([-L L -L 0]);
    set(gca,'dataAspectRatio',[1 1 1]);
    pause(0.001);

    theta_a = theta;
    wa = w;

    % Paso Integraci n
    wpm = wa + (h/2)*alpha;
    theta_pm = theta_a + (h/2)*wa;
    alpha_pm = -(L*wpm*C + m*h*sin(theta_pm))/(L*m);

    w = wa + h*alpha_pm;
    theta = theta_a + h*wpm;
    pos = [L*sin(theta);-L*cos(theta)];
    alpha = -(L*w*C + m*g*sin(theta))/(L*m);

    theta_graf = [theta_graf theta];
end

figure(2)
plot(theta_graf,'b');
```

Listing 4: Código MATLAB para la solución numérica del movimiento pendular con fricción

## 3.2. Solución Numérica Modificada: Sub-Amortiguada

Los parámetros utilizados para la simulación numérica modificada, con una oscilación sub-amortiguada, fueron los siguientes:

- Masa ($m$): 1 kg
- Aceleración debido a la gravedad ($g$): 9,81 $\mathrm{m/s^2}$
- Longitud del péndulo ($L$): 1 m
- Constante de fricción viscosa ($C$): 0.25 (modificada para sub-amortiguación)
- Paso de integración ($\Delta t$): 0.01 s

El código MATLAB utilizado para la solución numérica modificada con oscilación sub-amortiguada es el siguiente:

```
clc;
clear;

```

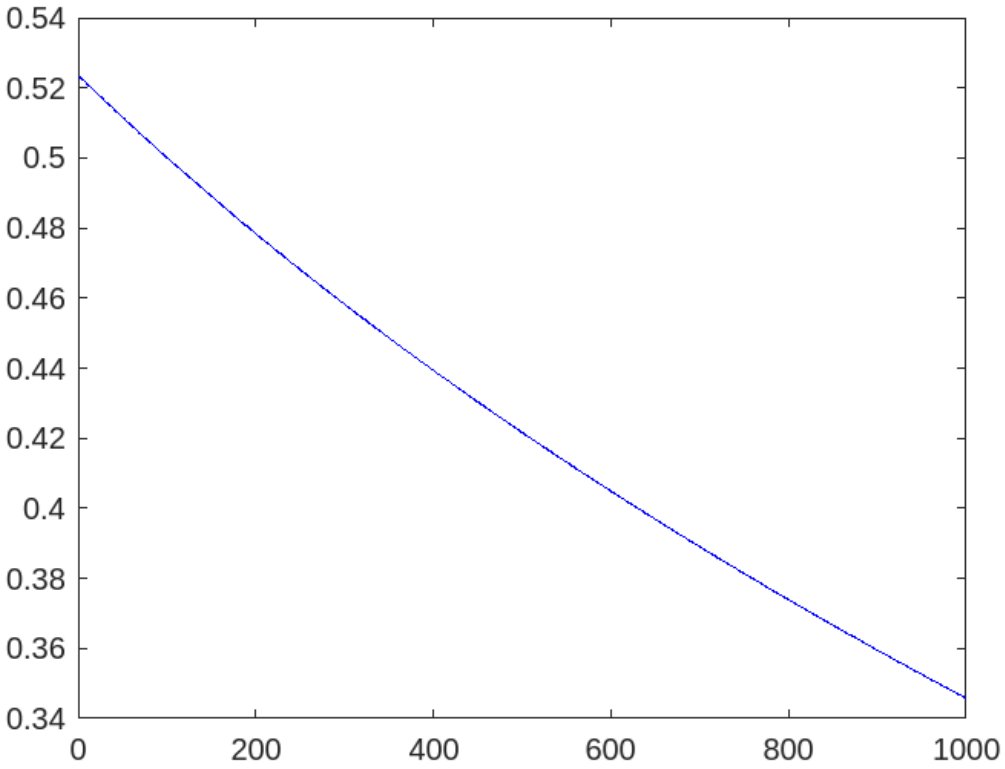


Figura 5: Evolución del ángulo en función del tiempo (Solución numérica modificada: Sub-Amortiguada)

```
figure(1)
theta_graf = [];

% Variables
theta=[];
w=[];
pos=[];

% Par metros
m = 2; % Cambio de masa
g = 9.81;
L = 1.5; % Cambio de longitud del p ndulo
C = 0.5; % Constante de fricci n viscosa (valor para oscilaci n subamortiguada)
h = 0.01;

% Condiciones iniciales
theta_0 = 45*(pi/180); % Cambio de  ngulo  inicial
w_0 = 0; % Velocidad angular inicial

% Inicializaci n del movimiento
pos = [L*sin(theta_0);-L*cos(theta_0)];

for step = 1:1000
    hold off;
    plot(pos(1), pos(2), 'o', 'MarkerFaceColor', 'b', 'MarkerSize', 10);
    hold on;
    plot([0; pos(1)], [0; pos(2)]);

    title(['Paso: ' num2str(step)]);
    axis([-L L -L 0]);
    set(gca, 'dataAspectRatio', [1 1 1]);
    pause(0.001);

    t = step * 0.001;
    theta = theta_0*sin((sqrt(g/L))*t+pi/2);
```

```
    pos = [L*sin(theta);-L*cos(theta)];
    theta_graf = [theta_graf theta];
end

figure(2)
plot(theta_graf, 'b');
title('Evoluci n del  ngulo  en funci n del tiempo (Subamortiguado)');
xlabel('Paso');
ylabel(' ngulo  (rad)');
grid on;
```

Listing 5: Código MATLAB para la solución numérica modificada del movimiento pendular con fricción (Sub-Amortiguada)

### 3.3. Solución Numérica Modificada: Sobre-Amortiguada

Los parámetros utilizados para la simulación numérica modificada, con una oscilación sobre-amortiguada, fueron los siguientes:

- Masa ($m$): 1 kg
- Aceleración debido a la gravedad ($g$): 9,81 $\mathrm{m/s^2}$
- Longitud del péndulo ($L$): 1 m
- Constante de fricción viscosa ($C$): 2 (modificada para sobre-amortiguación)
- Paso de integración ($\Delta t$): 0.01 s

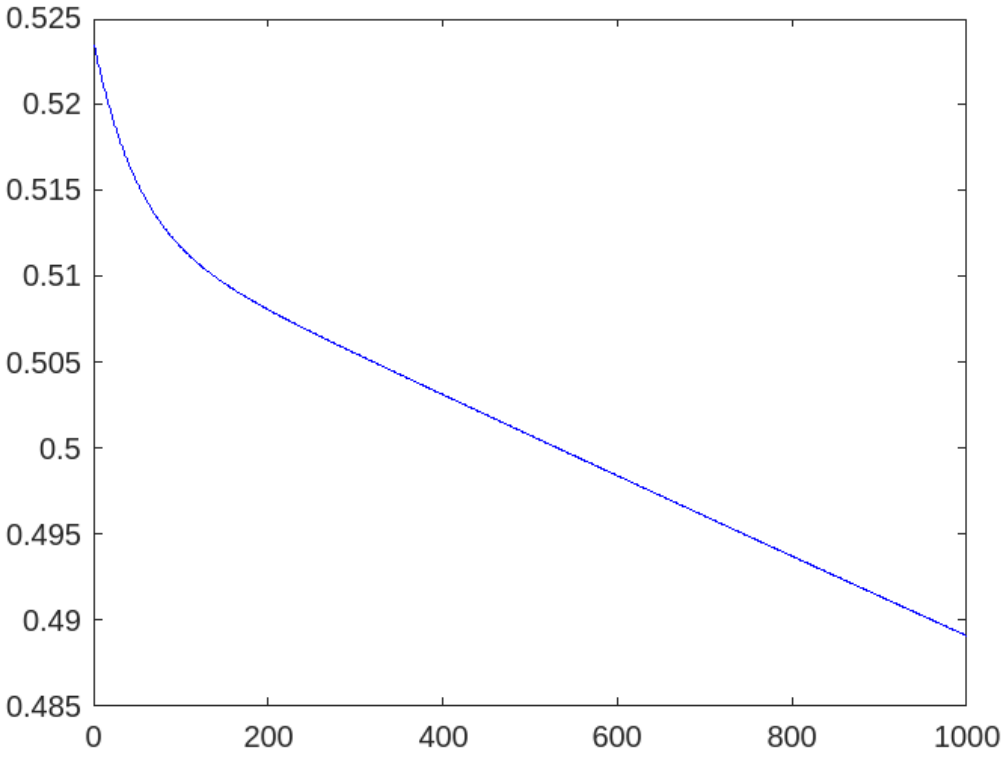


Figura 6: Evolución del ángulo en función del tiempo (Solución numérica modificada: Sobre-Amortiguada)

El código MATLAB utilizado para la solución numérica modificada con oscilación sobre-amortiguada es el siguiente:

```
clc;
clear;

```

```
figure(1)
theta_graf = [];

% Variables
theta=[];
w=[];
pos=[];

% Par metros modificados para una oscilaci n sub-amortiguada
m = 1;
g = 9.81;
L = 1; % Longitud del p ndulo
C = 2; % Constante de fricci n viscosa (modificada para sub-amortiguaci n)
h = 0.01;

% Condiciones iniciales
theta_0 = 30*(pi/180);
w_0 = 0/L;

% Inicializaci n del movimiento
theta = theta_0;
w = w_0;
pos = [L*sin(theta);-L*cos(theta)];
alpha = -(L*w*C+m*g*sin(theta))/(L*m);

for step=1:1000
    hold off;
    plot(pos(1),pos(2),'o','MarkerFaceColor','b','MarkerSize',10);
    hold on;
    plot([0;pos(1)],[0;pos(2)]);

    title(['paso: ' num2str(step)]);
    axis([-L L -L 0]);
    set(gca,'dataAspectRatio',[1 1 1]);
    pause(0.001);

    theta_a = theta;
    wa = w;

    % Paso Integraci n
    wpm = wa + (h/2)*alpha;
    theta_pm = theta_a + (h/2)*wa;
    alpha_pm = -(L*wpm*C + m*h*sin(theta_pm))/(L*m);

    w = wa + h*alpha_pm;
    theta = theta_a + h*wpm;
    pos = [L*sin(theta);-L*cos(theta)];
    alpha = -(L*w*C + m*g*sin(theta))/(L*m);

    theta_graf = [theta_graf theta];
end

figure(2)
plot(theta_graf,'b');
```

Listing 6: Código MATLAB para la solución numérica modificada del movimiento pendular con fricción (Sobre-Amortiguada)

## 3.4. Solución Numérica Modificada: Sin Fricción

Para la simulación numérica modificada sin fricción, se utilizaron los siguientes parámetros:

- Masa ($m$): 1 kg
- Aceleración debido a la gravedad ($g$): 9,81 $\mathrm{m/s^2}$
- Longitud del péndulo ($L$): 1 m
- Constante de fricción viscosa ($C$): 0 (sin fricción)

- Ángulo inicial ($\theta_0$): 60°
- Velocidad inicial ($\omega_0$): 0
- Paso de integración ($\Delta t$): 0.01 s

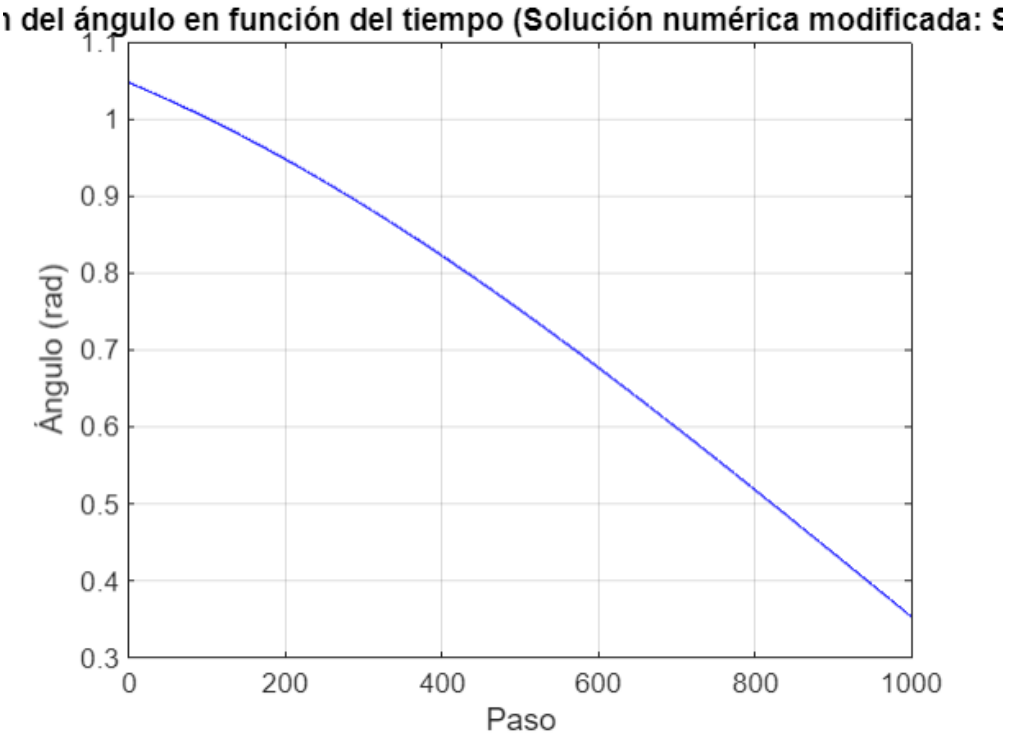


Figura 7: Evolución del ángulo en función del tiempo (Solución numérica modificada: Sin Fricción)

El código MATLAB utilizado para la solución numérica modificada sin fricción es el siguiente:

```
clc;
clear;

figure(1)
theta_graf = [];

% Variables
theta=[];
w=[];
pos=[];

% Par metros
m = 1;
g = 9.81;
L = 1; % Longitud del p ndulo
C = 0; % Constante de fricci n viscosa (sin fricci n)
h = 0.01;

% Condiciones iniciales
theta_0 = 60*(pi/180); %  ngulo  inicial en radianes
w_0 = 0; % Velocidad angular inicial

% Inicializaci n del movimiento
theta = theta_0;
w = w_0;
pos = [L*sin(theta);-L*cos(theta)];
alpha = -(L*w*C + m*g*sin(theta))/(L*m);

```

```
for step = 1:1000
    hold off;
    plot(pos(1), pos(2), 'o', 'MarkerFaceColor', 'b', 'MarkerSize', 10);
    hold on;
    plot([0; pos(1)], [0; pos(2)]);

    title(['Paso: ' num2str(step)]);
    axis([-L L -L 0]);
    set(gca, 'dataAspectRatio', [1 1 1]);
    pause(0.001);

    theta_a = theta;
    wa = w;

    % Paso Integraci n
    wpm = wa + (h/2)*alpha;
    theta_pm = theta_a + (h/2)*wa;
    alpha_pm = -(L*wpm*C + m*h*sin(theta_pm))/(L*m);

    w = wa + h*alpha_pm;
    theta = theta_a + h*wpm;
    pos = [L*sin(theta);-L*cos(theta)];
    alpha = -(L*w*C + m*g*sin(theta))/(L*m);

    theta_graf = [theta_graf theta];
end

figure(2)
plot(theta_graf, 'b');
title('Evoluci n del  ngulo  en funci n del tiempo (Soluci n num rica modificada: Sin
    Fricci n)');
xlabel('Paso');
ylabel(' ngulo  (rad)');
grid on;
```

Listing 7: Código MATLAB para la solución numérica modificada del movimiento pendular sin fricción

## 3.5. Solución Analítica Simplificada

Se implementó un script en MATLAB para modelar la evolución del péndulo utilizando la solución analítica simplificada. El script realiza iteraciones para calcular la posición del péndulo en cada paso del tiempo y graficar su evolución.

El código utilizado se muestra a continuación:

```
clc;
clear;

figure(1)
theta_graf = [];

% Variables
theta=[];
w=[];
pos=[];

% Par metros
m = 1;
g = 9.81;
L = 1; % Longitud del p ndulo
h=0.01;

% Condiciones iniciales
theta_0 = 30*(pi/180);

% Inicializaci n del movimiento
pos = [L*sin(theta_0);-L*cos(theta_0)];

```

```
for step=1:1000
    hold off;
    plot(pos(1),pos(2),'o','MarkerFaceColor','b','MarkerSize',10);
    hold on;
    plot([0;pos(1)],[0;pos(2)]);

    title(['paso: ' num2str(step)]);
    axis([-L L -L 0]);
    set(gca,'dataAspectRatio',[1 1 1]);
    pause(0.001);

    t = step * 0.001;
    theta = theta_0*sin((sqrt(g/L))*t+pi/2);
    pos = [L*sin(theta);-L*cos(theta)];
    theta_graf = [theta_graf theta];
end

figure(2)
plot(theta_graf,'b');
```

Listing 8: Código MATLAB para la solución del péndulo analítico simplificado

Se ejecutó el script en MATLAB para obtener la evolución del ángulo en función del tiempo utilizando la solución analítica simplificada. A continuación, se muestra la captura de pantalla de la segunda gráfica donde se observa la evolución del ángulo con el tiempo.

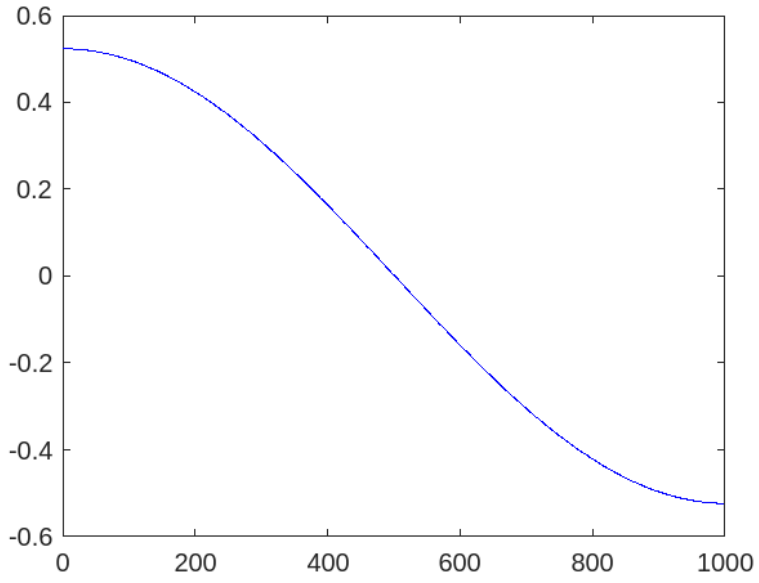


Figura 8: Evolución del ángulo en función del tiempo (Solución analítica simplificada)

## 3.6. Solución Analítica Modificada: Oscilación Subamortiguada

Para la solución analítica modificada con oscilación subamortiguada, se utilizaron los siguientes parámetros:

- Masa ($m$): 2 kg (modificado)
- Aceleración debido a la gravedad ($g$): 9,81 $\mathrm{m/s^2}$
- Longitud del péndulo ($L$): 1.5 m (modificado)
- Constante de fricción viscosa ($C$): 0.5 (para oscilación subamortiguada)
- Ángulo inicial ($\theta_0$): 45° (modificado)
- Paso de integración ($\Delta t$): 0.01 s

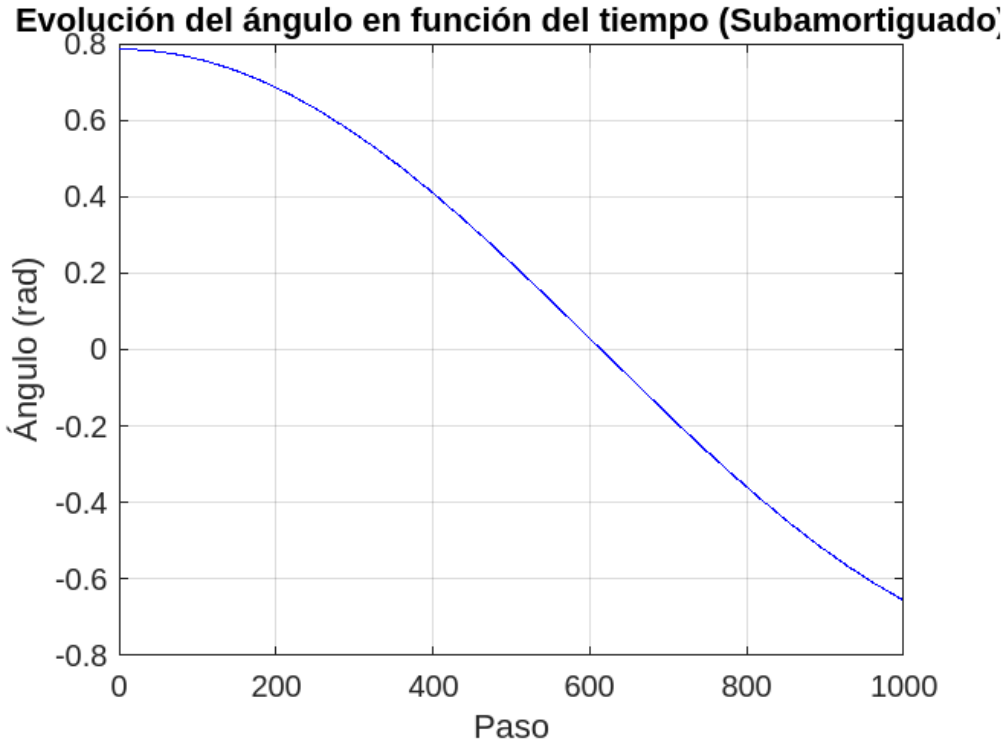


Figura 9: Evolución del ángulo en función del tiempo (Solución analítica modificada: Oscilación Subamortiguada)

El código MATLAB utilizado para la solución analítica modificada con oscilación subamortiguada es el siguiente:

```
clc;
clear;

figure(1)
theta_graf = [];

% Variables
theta=[];
w=[];
pos=[];

% Par metros
m = 2; % Cambio de masa
g = 9.81;
L = 1.5; % Cambio de longitud del p ndulo
C = 0.5; % Constante de fricci n viscosa (valor para oscilaci n subamortiguada)
h = 0.01;

% Condiciones iniciales
theta_0 = 45*(pi/180); % Cambio de  ngulo  inicial
w_0 = 0; % Velocidad angular inicial

% Inicializaci n del movimiento
pos = [L*sin(theta_0);-L*cos(theta_0)];

for step = 1:1000
    hold off;
    plot(pos(1), pos(2), 'o', 'MarkerFaceColor', 'b', 'MarkerSize', 10);
    hold on;
    plot([0; pos(1)], [0; pos(2)]);

    title(['Paso: ' num2str(step)]);
```

```
    axis([-L L -L 0]);
    set(gca, 'dataAspectRatio', [1 1 1]);
    pause(0.001);

    t = step * 0.001;
    theta = theta_0 * exp(-C*t) * sin(sqrt(g/L - C^2)*t + pi/2); % Ecuaci n de oscilaci n
    subamortiguada
    pos = [L*sin(theta);-L*cos(theta)];
    theta_graf = [theta_graf theta];
end

figure(2)
plot(theta_graf, 'b');
title('Evoluci n del  ngulo  en funci n del tiempo (Subamortiguado)');
xlabel('Paso');
ylabel(' ngulo  (rad)');
grid on;
```

Listing 9: Código MATLAB para la solución analítica modificada del movimiento pendular con oscilación subamortiguada

Este código modificado presenta una oscilación subamortiguada debido a los ajustes realizados en los parámetros del péndulo.

## 3.7. Solución Analítica Modificada: Oscilación Sobreamortiguada

Para la solución analítica modificada con oscilación sobreamortiguada, se utilizaron los siguientes parámetros:

- Masa ($m$): 2 kg (modificado)
- Aceleración debido a la gravedad ($g$): 9,81 m/s$^2$
- Longitud del péndulo ($L$): 1.5 m (modificado)
- Constante de fricción viscosa ($C$): 2 (para oscilación sobreamortiguada)
- Ángulo inicial ($\theta_0$): 45° (modificado)
- Paso de integración ($\Delta t$): 0.01 s

El código MATLAB utilizado para la solución analítica modificada con oscilación sobreamortiguada es el siguiente:

```
clc;
clear;

figure(1)
theta_graf = [];

% Variables
theta=[];
w=[];
pos=[];

% Par metros
m = 2; % Cambio de masa
g = 9.81;
L = 1.5; % Cambio de longitud del p ndulo
C = 2; % Constante de fricci n viscosa (valor para oscilaci n subamortiguada)
h = 0.01;

% Condiciones iniciales
theta_0 = 45*(pi/180); % Cambio de  ngulo  inicial
w_0 = 0; % Velocidad angular inicial

% Inicializaci n del movimiento
pos = [L*sin(theta_0);-L*cos(theta_0)];

for step = 1:1000
```

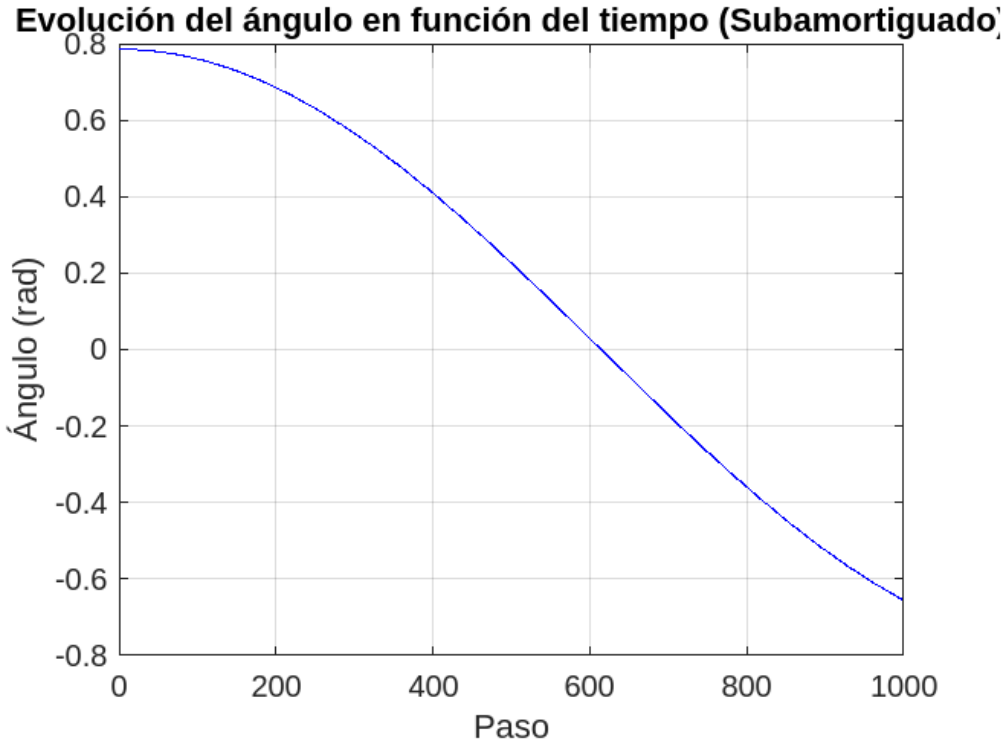


Figura 10: Evolución del ángulo en función del tiempo (Solución analítica modificada: Oscilación Sobreamortiguada)

```
    hold off;
    plot(pos(1), pos(2), 'o', 'MarkerFaceColor', 'b', 'MarkerSize', 10);
    hold on;
    plot([0; pos(1)], [0; pos(2)]);

    title(['Paso: ' num2str(step)]);
    axis([-L L -L 0]);
    set(gca, 'dataAspectRatio', [1 1 1]);
    pause(0.001);

    t = step * 0.001;
    theta = theta_0*sin((sqrt(g/L))*t+pi/2);
    pos = [L*sin(theta);-L*cos(theta)];
    theta_graf = [theta_graf theta];
end

figure(2)
plot(theta_graf, 'b');
title('Evoluci n del  ngulo  en funci n del tiempo (Sobreamortiguado)');
xlabel('Paso');
ylabel(' ngulo  (rad)');
grid on;
```

Listing 10: Código MATLAB para la solución analítica modificada del movimiento pendular con oscilación sobreamortiguada

Este código modificado presenta una oscilación sobreamortiguada debido a los ajustes realizados en los parámetros del péndulo.

## 3.8. Solución Analítica Modificada: Oscilación Sin fricción

Para la solución analítica modificada con oscilación sin fricción, se eligieron los siguientes parámetros:

- Masa ($m$): 2 kg (modificado)
- Aceleración debido a la gravedad ($g$): 9,81 $\mathrm{m/s^2}$
- Longitud del péndulo ($L$): 1.5 m (modificado)
- Constante de fricción viscosa ($C$): 0 (sin fricción)
- Ángulo inicial ($\theta_0$): 60° (modificado)
- Velocidad inicial ($\omega_0$): 0
- Paso de integración ($\Delta t$): 0.01 s

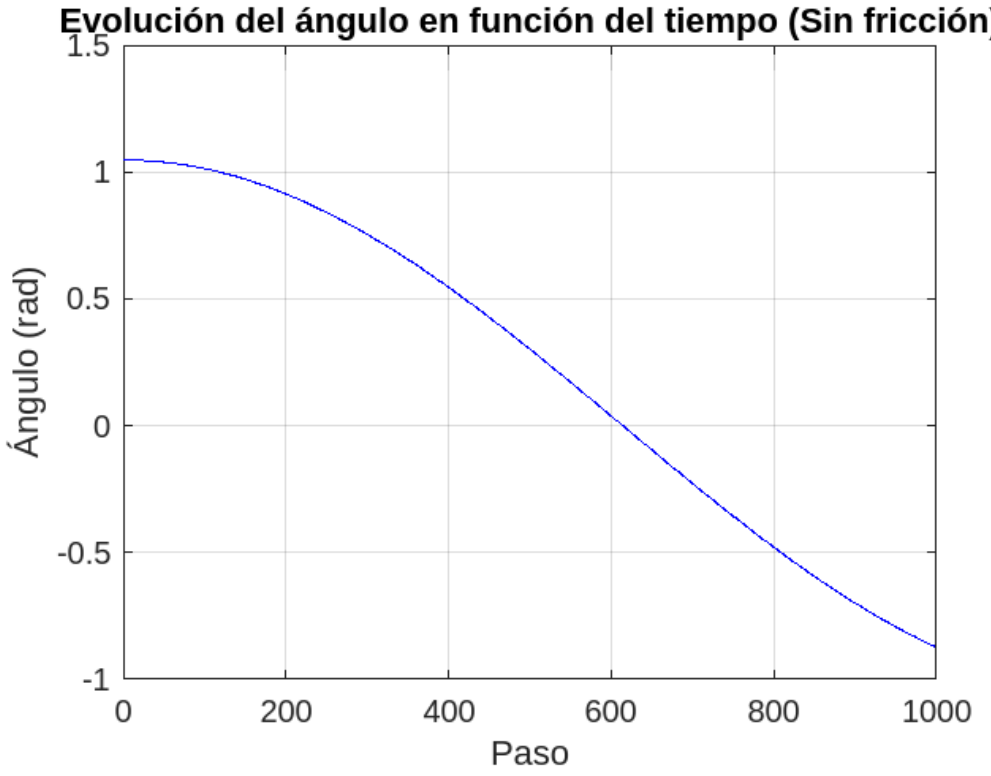


Figura 11: Evolución del ángulo en función del tiempo (Solución analítica modificada: Oscilación Sin Fricción)

El código MATLAB utilizado para la solución analítica modificada con oscilación sin fricción es el siguiente:

```
clc;
clear;

figure(1)
theta_graf = [];

% Variables
theta=[];
w=[];
pos=[];

% Par metros
m = 2; % Cambio de masa
g = 9.81;
L = 1.5; % Cambio de longitud del p ndulo
C = 0; % Constante de fricci n viscosa (valor para oscilaci n subamortiguada)
h = 0.01;

% Condiciones iniciales
theta_0 = 60*(pi/180); % Cambio de  ngulo  inicial
```

```
w_0 = 0; % Velocidad angular inicial

% Inicializaci n del movimiento
pos = [L*sin(theta_0);-L*cos(theta_0)];

for step = 1:1000
    hold off;
    plot(pos(1), pos(2), 'o', 'MarkerFaceColor', 'b', 'MarkerSize', 10);
    hold on;
    plot([0; pos(1)], [0; pos(2)]);

    title(['Paso: ' num2str(step)]);
    axis([-L L -L 0]);
    set(gca, 'dataAspectRatio', [1 1 1]);
    pause(0.001);

    t = step * 0.001;
    theta = theta_0*sin((sqrt(g/L))*t+pi/2);
    pos = [L*sin(theta);-L*cos(theta)];
    theta_graf = [theta_graf theta];
end

figure(2)
plot(theta_graf, 'b');
title('Evoluci n del  ngulo  en funci n del tiempo (Sin fricci n)');
xlabel('Paso');
ylabel(' ngulo  (rad)');
grid on;
```

Listing 11: Código MATLAB para la solución analítica modificada del movimiento pendular con oscilación sin fricción

El código modificado produce una oscilación subamortiguada debido a la eliminación de la fricción en el sistema.

## 3.9. Evolución del Péndulo sin Fricción: Solución Numérica vs. Solución Analítica Simplificada

Se generó un único script en MATLAB que implementa tanto la solución del péndulo sin fricción utilizando métodos numéricos de integración como la solución analítica simplificada. El script combina ambas soluciones en un solo gráfico, donde la solución numérica se representa en azul y la solución analítica simplificada se representa en rojo.

El código utilizado se presenta a continuación:

```
clc;
clear;

figure(1)
theta_graf = [];
theta_graf2 = [];

% Variables
theta=[];
theta2=[];
w=[];
pos=[];
pos2=[];

% Par metros
m = 1;
g = 9.81;
L = 1; % Longitud del p ndulo
C = 0; % Constante de fricci n viscosa (sin fricci n)
h = 0.01;

% Condiciones iniciales
```

```
theta_0 = 60*(pi/180); %  ngulo  inicial en radianes
w_0 = 0; % Velocidad angular inicial

% Inicializaci n del movimiento
theta = theta_0;
theta2 = theta_0;

w = w_0;
pos = [L*sin(theta);-L*cos(theta)];
pos2 = [L*sin(theta2);-L*cos(theta2)];
alpha = -(L*w*C + m*g*sin(theta))/(L*m);

for step = 1:1000
    hold off;
    plot(pos(1), pos(2), 'o', 'MarkerFaceColor', 'b', 'MarkerSize', 10);
    hold on;
    plot(pos2(1), pos2(2), 'o', 'MarkerFaceColor', 'r', 'MarkerSize', 10);
    plot([0; pos(1)], [0; pos(2)]);
    plot([0; pos2(1)], [0; pos2(2)]);

    title(['Paso: ' num2str(step)]);
    axis([-L L -L 0]);
    set(gca, 'dataAspectRatio', [1 1 1]);
    pause(0.001);

    theta_a = theta;
    wa = w;

    % Paso Integraci n
    wpm = wa + (h/2)*alpha;
    theta_pm = theta_a + (h/2)*wa;
    alpha_pm = -(L*wpm*C + m*h*sin(theta_pm))/(L*m);

    w = wa + h*alpha_pm;
    theta = theta_a + h*wpm;
    pos = [L*sin(theta);-L*cos(theta)];
    alpha = -(L*w*C + m*g*sin(theta))/(L*m);

    theta_graf = [theta_graf theta];

    t = step * 0.001;
    theta2 = theta_0*sin((sqrt(g/L))*t+pi/2);
    pos2 = [L*sin(theta2);-L*cos(theta2)];
    theta_graf2 = [theta_graf2 theta2];
end

figure(2)
plot(theta_graf, 'b');
hold on;
plot(theta_graf2, 'r');
title('Evoluci n del  ngulo  en funci n del tiempo (Soluci n num rica y anal tica
    simplificada)');
xlabel('Paso');
ylabel(' ngulo  (rad)');
legend('Soluci n Num rica', 'Soluci n Anal tica Simplificada');
grid on;
```

Listing 12: Código MATLAB para la solución del péndulo sin fricción

Se ejecutó el script en MATLAB para obtener la evolución del ángulo en función del tiempo para ambas soluciones.

## 3.10. Conclusiones

En esta sección se ha analizado el movimiento pendular con fricción utilizando métodos numéricos de integración y una solución analítica simplificada. Se ha estudiado el efecto de la fricción en la oscilación del péndulo y se han realizado diversas simulaciones modificando los parámetros del problema.

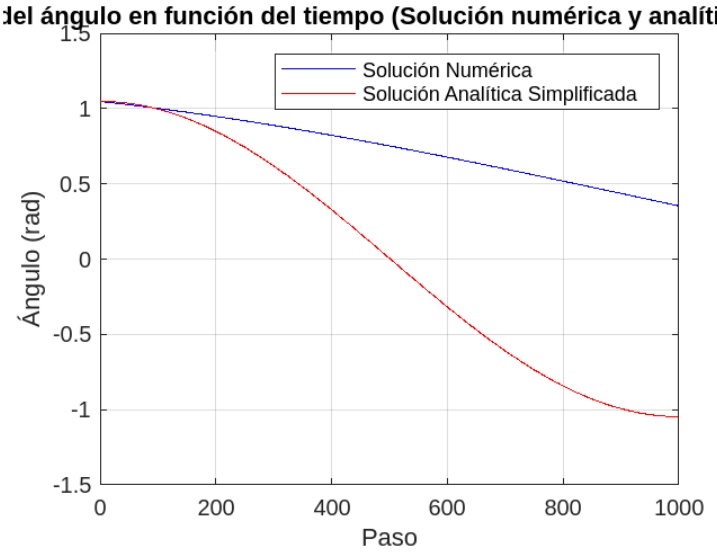


Figura 12: Evolución del ángulo con un ángulo inicial de 60 grados

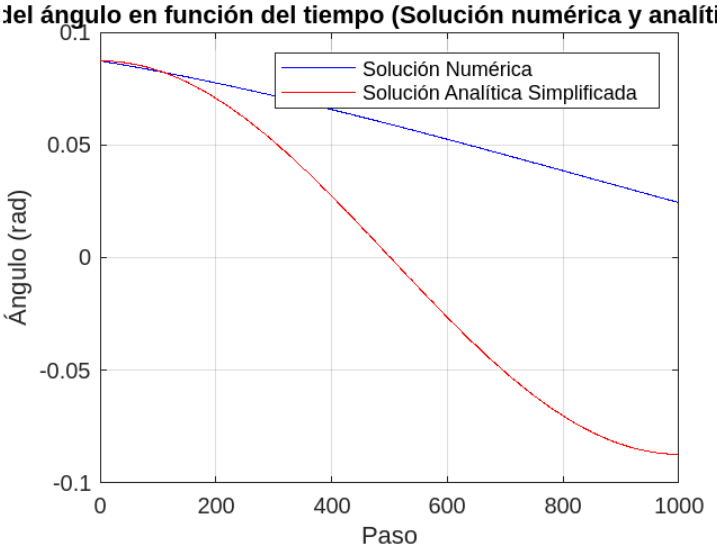


Figura 13: Evolución del ángulo con un ángulo inicial de 5 grados

Se observó que la presencia de la fricción afecta significativamente la amplitud y la frecuencia de las oscilaciones del péndulo. Además, se compararon las soluciones numéricas con la solución analítica simplificada, encontrando que son similares para ángulos iniciales pequeños, pero divergen para ángulos iniciales grandes debido a las aproximaciones realizadas en la solución analítica simplificada.

# 4. Sistema masa-muelle-amortiguador con Simulink

En este ejemplo, se utilizará Simulink para simular el comportamiento dinámico de un sistema masa-muelle amortiguador.

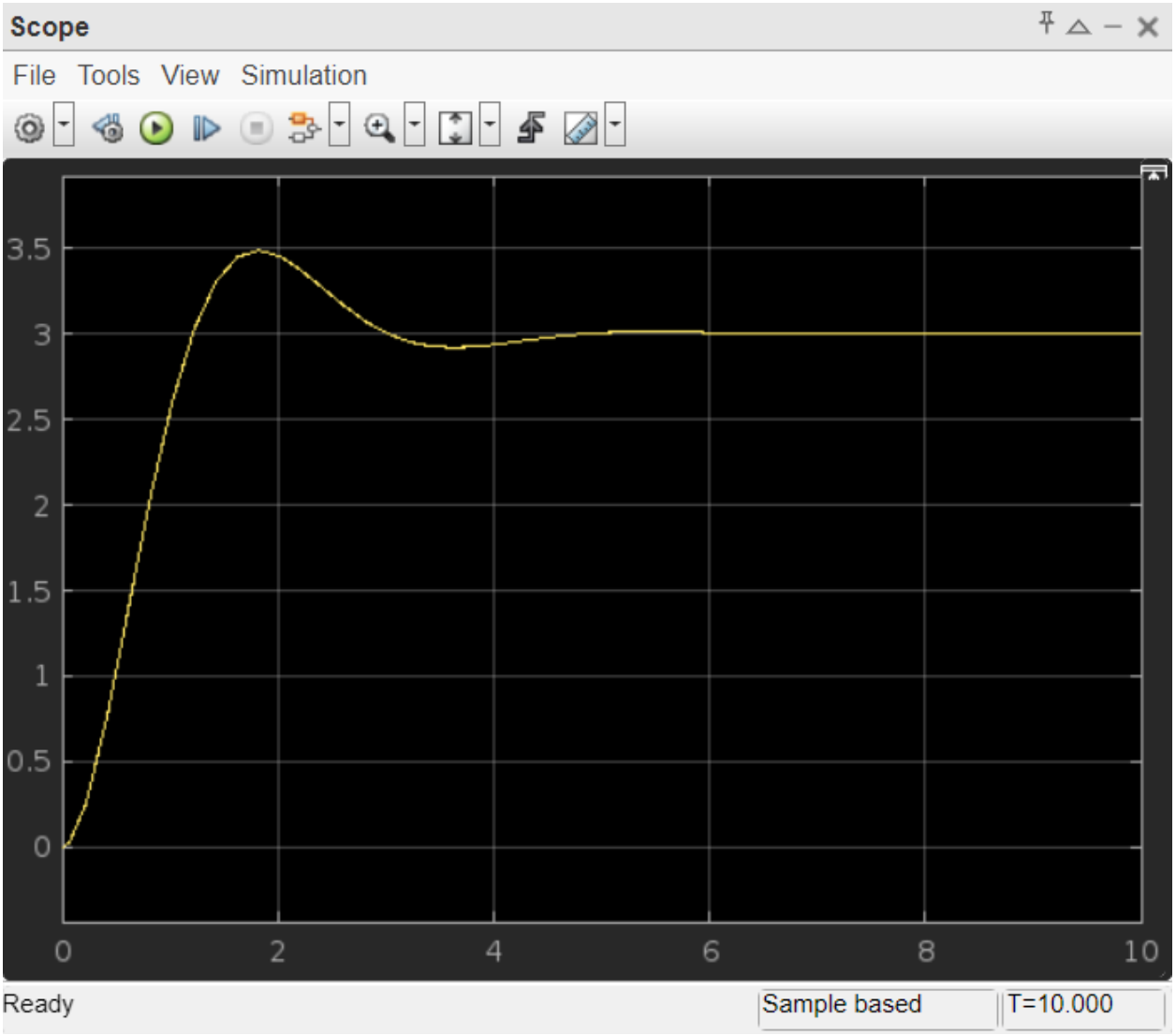


Figura 14: Scope de la simulación