UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
FACULTAD DE INGENIERÍA

AUTÓNOMA UNIVERSIDAD DE QUERÉTARO

Facultad de Ingeniería

REPORTE DE TALLER DE DINÁMICA

Alumno: **Diego Joel Zuñiga Fragoso**

Expediente: 317684

Carrera: Ing. en Automatización

Periodo: 2023-2

Catedrático: Dr. Suresh Thenozhi

ÍNDICES

Diego Joel Zuñiga Fragoso
317684

Cinemática de partícula	PRACTICA # 1 FECHA 10/08/2023

1. Objetivo

El objetivo de esta práctica es familiarizarse con las funciones fundamentales de MATLAB, aprovechando su capacidad para facilitar cálculos y la generación de gráficos. Además, se busca aplicar los conocimientos adquiridos en el curso para implementar soluciones efectivas utilizando esta herramienta. A través de esta práctica, espero adaptarme más a la estructura y las funciones de MATLAB, pues estoy más acostumbrado a trabajar con C++.

2. Marco Teórico

Velocidad

La velocidad es una medida de la rapidez con la que un objeto cambia su posición. En términos matemáticos, es la derivada del desplazamiento respecto al tiempo. La velocidad es una cantidad vectorial, lo que significa que tiene tanto magnitud (la rapidez) como dirección. En el contexto de la dinámica, la velocidad puede ser constante (movimiento uniforme) o variable (movimiento acelerado).

Aceleración

La aceleración es una medida de cómo cambia la velocidad de un objeto con el tiempo. Matemáticamente, es la derivada de la velocidad respecto al tiempo. Al igual que la velocidad, la aceleración es una cantidad vectorial y puede ser positiva (cuando un objeto está aumentando su velocidad), negativa (cuando un objeto está disminuyendo su velocidad, también conocido como desaceleración) o cero (cuando un objeto mantiene una velocidad constante).

Posición

La posición es una medida de la ubicación de un objeto en un determinado instante. En un gráfico de posición contra tiempo, el desplazamiento del objeto se puede determinar calculando el área bajo la curva. La posición también es una cantidad vectorial y puede describirse en términos de coordenadas cartesianas (x, y, z) en un sistema de referencia dado.

MATLAB es un entorno de programación para cálculo numérico y gráficos desarrollado por MathWorks. Permite realizar análisis de datos, desarrollo de algoritmos, creación de modelos y aplicaciones, simulación, generación de código para diseño de sistemas y software embebidos, entre otras capacidades.

3. Formulación

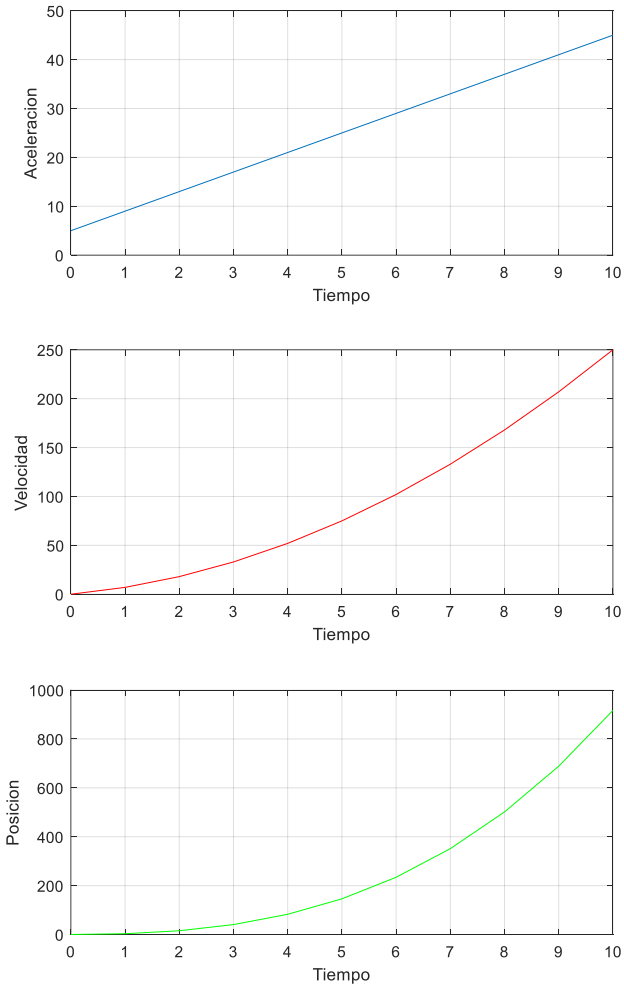
Diego Joel Zuñiga Fragoso
317684

Esta primer practica son problemas introductorios relativamente fáciles, donde las integrales se pueden hacer de manera mental por lo que no se requiere mucha formulación.

4. Implementación en MATLAB y Resultados

Codigo realizado en clase:

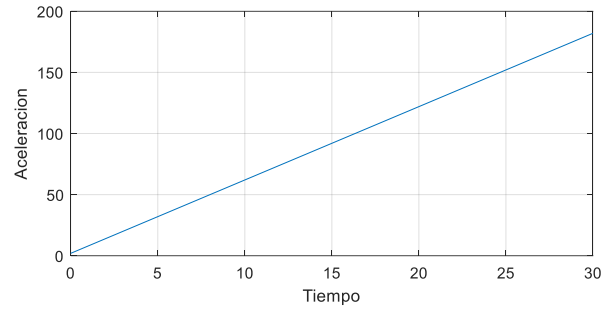
```
t = [0:10];

a = 4.*t+5;
subplot(3,1,1);
plot(t,a)
xlabel('Tiempo');
ylabel('Aceleracion');
grid on;


v = 2*(t.^2)+(5.*t);
subplot(3,1,2);
plot(t,v,'r')
xlabel('Tiempo');
ylabel('Velocidad');
grid on;


s = (2/3)*t.^3+(5/2).*t.^2;
subplot(3,1,3);
plot(t,s,'g')
xlabel('Tiempo');
ylabel('Posicion');
grid on;
```

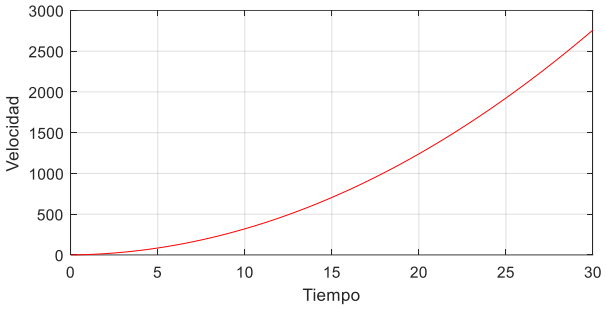


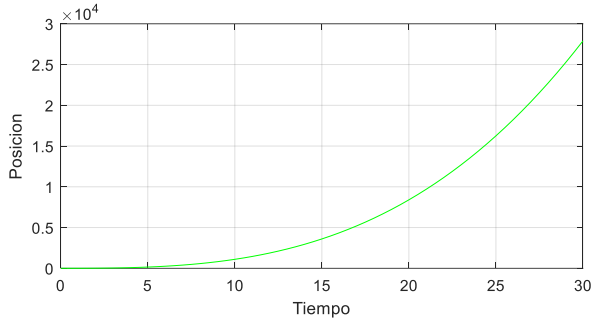
Diego Joel Zuñiga Fragoso
317684

Ejercicio 12.1

```
t = [0:0.5:30];

a = (6*t+2);
subplot(3,1,1);
plot(t,a)
xlabel('Tiempo');
ylabel('Aceleracion');
grid on;


v = (3*t.^2)+(2*t);
subplot(3,1,2);
plot(t,v,'r')
xlabel('Tiempo');
ylabel('Velocidad');
grid on;


s = (t.^3)+(t.^2);
subplot(3,1,3);
plot(t,s,'g')
xlabel('Tiempo');
ylabel('Posicion');
grid on;
```

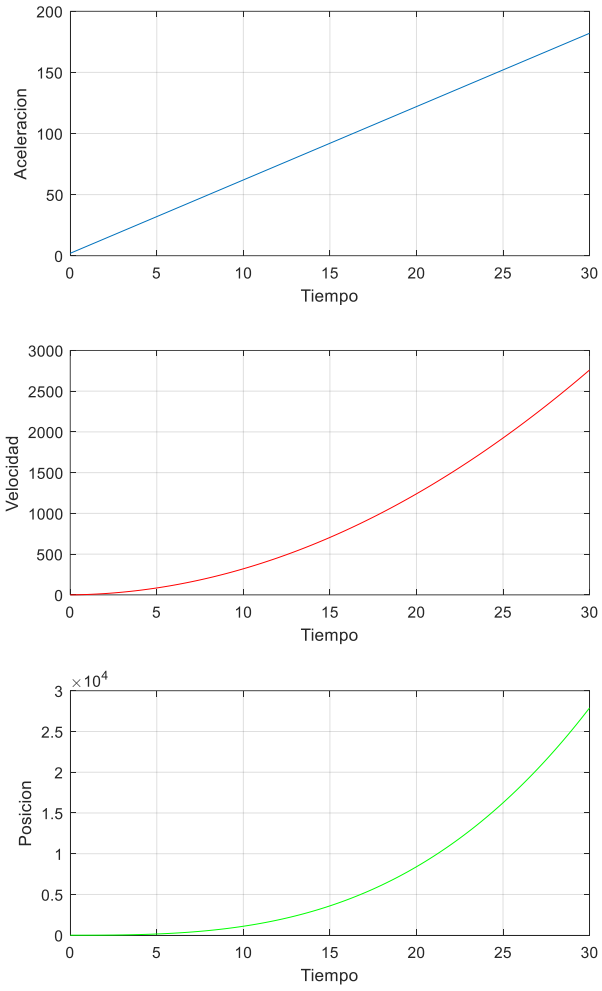






Ejercicio 12.2

```
t = [0:0.25:30];

a = (6*t+2);
subplot(3,1,1);
plot(t,a)
xlabel('Tiempo');
ylabel('Aceleracion');
grid on;


v = (3*t.^2)+(2*t);
subplot(3,1,2);
plot(t,v,'r')
xlabel('Tiempo');
ylabel('Velocidad');
grid on;


s = (t.^3)+(t.^2);
subplot(3,1,3);
plot(t,s,'g')
xlabel('Tiempo');
ylabel('Posicion');
grid on;
```



Como podemos observar obtenemos resultados muy precisos y de una manera muy rápida, realice varias evaluaciones para distintos puntos del tiempo y la gráfica de MATLAB muestra con precisión todos los datos.

5. Conclusión

El objetivo de la práctica se cumplió, ya me familiarice más con el software que utilizaremos a lo largo del semestre, aunque tiene ciertas variaciones con C++, se nota que es un software más especializado en matemáticas.

Diego Joel Zuñiga Fragoso
317684

Tiro Parabólico	PRACTICA # 2 FECHA 24/08/2023

1. Objetivo

El objetivo de la práctica es realizar cálculos y gráficos en MATLABA que ilustren el comportamiento de un tiro parabólico para tres ángulos de lanzamiento distintos. Además, se busca analizar cómo varía la distancia recorrida y la altura máxima alcanzada en relación con el ángulo de lanzamiento.

2. Marco Teórico

1. Tiro Parabólico:

El tiro parabólico es un concepto fundamental en la mecánica clásica que describe el movimiento de un objeto lanzado en un plano inclinado respecto a la horizontal. Este tipo de movimiento es una combinación de un movimiento rectilíneo uniforme en el eje horizontal (X) y un movimiento rectilíneo uniformemente acelerado en el eje vertical (Y). Los factores clave que determinan la trayectoria de un tiro parabólico son la velocidad inicial, el ángulo de lanzamiento y la aceleración debido a la gravedad.

2. Componentes de la Velocidad:

Cuando se realiza un tiro parabólico, la velocidad inicial del objeto se puede descomponer en dos componentes: una en la dirección horizontal (Vx) y otra en la dirección vertical (Vy). Estas componentes son cruciales para entender cómo varía la trayectoria del objeto con diferentes ángulos de lanzamiento.

3. Ángulo de Lanzamiento:

El ángulo de lanzamiento (θ) es el ángulo formado por la dirección inicial de lanzamiento y la horizontal. Este ángulo influye en la forma de la trayectoria y en las características del movimiento parabólico. Diferentes ángulos de lanzamiento producirán trayectorias y alcances diferentes.

4. Distancia Recorrida:

La distancia horizontal que un objeto alcanza en un tiro parabólico se llama alcance (R). El alcance depende del ángulo de lanzamiento y de la velocidad inicial. Se puede calcular mediante fórmulas específicas para el tiro parabólico.

5. Altura Máxima Alcanzada:

La altura máxima (h) es la máxima distancia vertical alcanzada por el objeto durante su trayectoria. Este valor también depende del ángulo de lanzamiento y la velocidad inicial.

6. Efecto de la Gravedad:

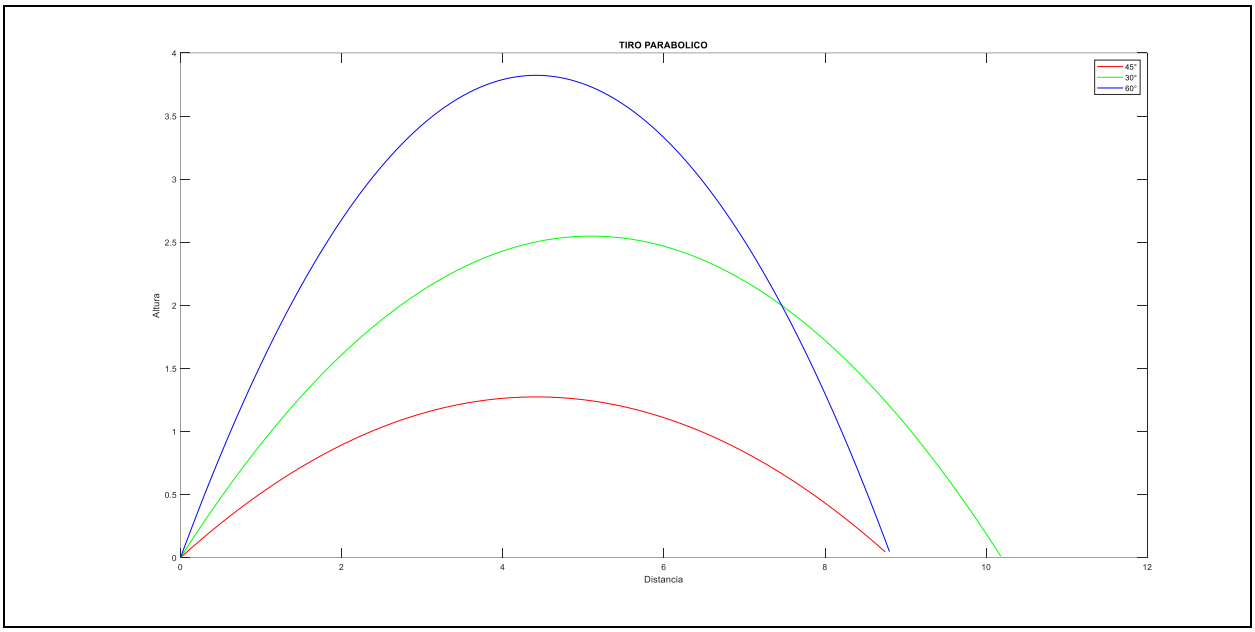
La aceleración debida a la gravedad (g) juega un papel fundamental en el movimiento parabólico. En la mayoría de los casos, se asume que la aceleración debida a la gravedad es constante y actúa hacia abajo, lo que afecta la componente vertical de la velocidad.

3. Implementación en MATLAB y Resultados

CODIGO

```
%Inicializar gravedad
g = 9.81;

% CASO 30°
Vx = 10*cosd(30);
Vy = 10*sind(30);
t = (2*Vy)/g;

disp('Valor de Vx:');
disp(Vx);
disp('Valor de Vy:');
disp(Vy);
disp('Valor del tiempo:');
disp(t);

x = Vx*t;
y = Vy*t-((1/2)*g)*t^2;
disp('Distancia:');
disp(x);
disp('Altura máxima:');
disp(y);
time = [0:0.01:t];
x1=Vx*time;
y1=Vy*time-(1/2)*(g)*(time.^2);
plot(x1,y1,'r');
title('Tiro parábolico a 30°');
xlabel('Distancia');
ylabel('Altura');
hold on;

% CASO 45°
Vx = 10*cosd(45);
Vy = 10*sind(45);
t = (2*Vy)/g;

disp('Vx: ');
disp(Vx);
```

```
disp('Vy:');
disp(Vy);
disp('Tiempo:');
disp(t);

x = Vx*t;
y = Vy*t-((1/2)*g)*t^2;
disp('Distancia:');
disp(x);
disp('Altura máxima:');
disp(y);
time = [0 : 0.01 : t];
x1 = Vx*time;
y1 = Vy*time-(1/2)*(g)*(time.^2);

plot(x1,y1,'g');

title('Tiro parábolico');
xlabel('Distancia');
ylabel('Altura');
hold on

% CASO 60°
Vx =10*cosd(60);
Vy = 10*sind(60);
t = (2*Vy)/g;
disp('Valor de Vx:');
disp(Vx);
disp('Valor de Vy:');
disp(Vy);
disp('Valor del tiempo:');
disp(t);

x = Vx*t;
y = Vy*t-((1/2)*g)*t.^2;
disp('Distancia:');
disp(x);
disp('Altura máxima:');
disp(y);

time = [0:0.01:t];
x1 = Vx*time;
y1 = Vy*time-(1/2)*(g)*(time.^2);
plot(x1,y1,'b');
xlabel('Distancia');
ylabel('Altura');
legend("Tiro parábolico 45°","Tiro parábolico 30°","Tiro parábolico 60°");
```



Como podemos observar, dependiendo del ángulo de inclinación al momento de realizar el tiro parabólico, este llegará más o menos lejos; cuanto menor sea el ángulo, menor altura máxima alcanzará. Esto se debe a la descomposición de las velocidades según el ángulo, ya que la velocidad inicial se distribuye de manera diferente en las componentes X e Y dependiendo del ángulo.

4. Conclusión

La práctica fue exitosa, ya que alcanzamos el objetivo planteado. Durante el desarrollo de esta, logré adquirir una comprensión más profunda sobre la influencia del ángulo de inclinación en un tiro parabólico. Además, esta experiencia me permitió fortalecer mis habilidades en el uso de MATLAB.

Diego Joel Zuñiga Fragoso
317684

Movimiento Tridimensional	PRACTICA # 3 FECHA 07/09/2023

1. Objetivo

El objetivo de esta práctica es emplear el software MATLAB para generar representaciones tridimensionales que visualicen las interacciones entre tres variables a lo largo del tiempo. Asimismo, se pretende llevar a cabo una comparativa de 3 distintos casos, con el fin de analizar y comprender las variaciones en estas relaciones en distintos contextos.

2. Marco Teórico

Coordenadas Tridimensionales:
Para representar puntos en el espacio 3D, se utilizan coordenadas tridimensionales, a menudo denotadas como (x, y, z), donde "x" representa la posición en el eje X, "y" en el eje Y y "z" en el eje Z. Estas coordenadas permiten ubicar un punto en el espacio tridimensional.

Espacio Tridimensional (3D):
En matemáticas y geometría, el espacio tridimensional se refiere a un espacio geométrico que tiene tres dimensiones: longitud, anchura y altura. En el contexto de la representación gráfica, el espacio 3D se utiliza para visualizar objetos tridimensionales.

Funciones Vectoriales:
Las funciones vectoriales son aquellas que asignan un vector a cada valor de un parámetro. En el contexto tridimensional, estas funciones pueden describir trayectorias y movimientos de objetos en el espacio. Las funciones vectoriales se representan comúnmente como r(t) = <x(t), y(t), z(t)>, donde r es el vector posición y t es el parámetro de tiempo.

3. Formulación

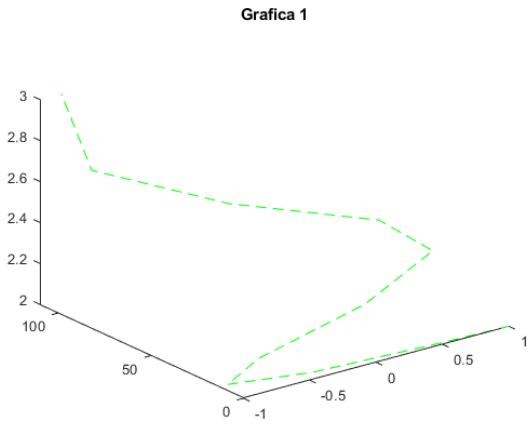
Para este código ocupamos la función plot3, la cual funciona exactamente como la función plot pero puede recibir hasta 3 funciones para graficar su movimiento tridimensional.

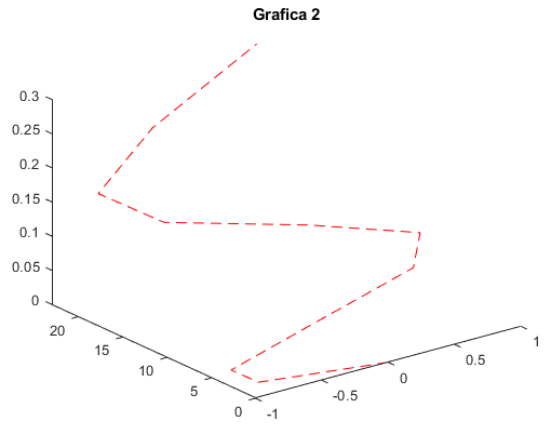
Todo lo demás son cosas que ya hemos utilizado anteriormente.

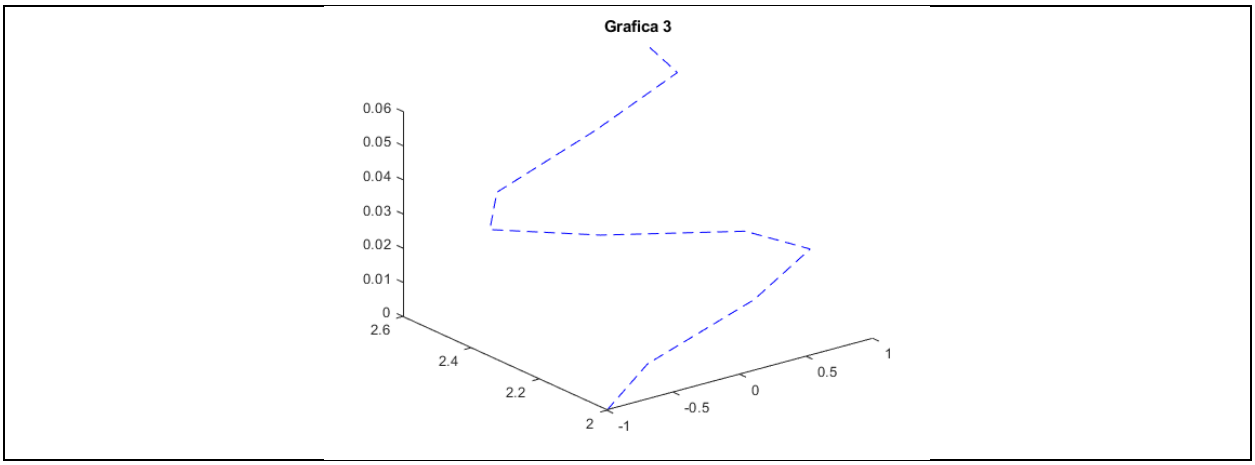
4. Implementación en MATLAB y Resultados

CODIGO

```
t = [0:1:10];
x1 = cos(t);
y1 = t.^2 +(0.01*t.^3);
z1 = 2 + (0.001*t.^3);

plot3(x1,y1,z1,'g--');
title('Grafica 1');
figure;

x2 = -sin(t);
y2 = (2*t)+(0.03*t.^2);
z2= (0.003*t.^2);
plot3(x2,y2,z2,'r--');
title('Grafica 2');
figure;

x3 = -cos(t);
y3 = (2+(0.06*t));
z3 = (0.006*t);
plot3(x3,y3,z3,'b--');
title('Grafica 3');
```







Como podemos observar MATHLAB puede graficar exitosamente el movimiento tridimensional de una partícula a lo largo del tiempo.

5. Conclusión

Se cumplió el objetivo de mediante el software MATLAB, aprender a realizar graficas tridimensionales, es una función que desconocía y me será muy útil en próximas prácticas, me he dado cuenta de que MATLAB es un software muy completo y aunque la forma de codificar es un poco diferente a lo que estoy acostumbrado con C++, suele ser muy intuitivo y fácil de aprender.

Diego Joel Zuñiga Fragoso
317684

Movimiento Erratico	PRACTICA # 4 FECHA 14/09/2023

1. Objetivo

El objetivo de la práctica es aprender a implementar funciones que representen un movimiento errático en el programa MATLAB. En este contexto, el término "movimiento errático" se refiere a la creación de una función de movimiento que está compuesta por varias funciones en diferentes intervalos de tiempo. Estas funciones variarán en su comportamiento y permitirán visualizar el movimiento de una partícula de una manera no lineal y, a veces, impredecible. El objetivo es adquirir habilidades para diseñar y programar este tipo de movimientos complejos, lo que puede ser útil en la simulación de situaciones donde el movimiento no sigue una trayectoria predecible y constante.

2. Marco Teórico

- Movimiento Errático:

En el contexto de esta práctica, el término "movimiento errático" se refiere a un tipo de movimiento que no sigue una trayectoria predecible y constante. En lugar de eso, el movimiento se modela mediante una serie de funciones que describen diferentes comportamientos en intervalos de tiempo específicos. Estos intervalos pueden tener variaciones en la velocidad, la dirección o la aceleración, lo que resulta en una trayectoria no lineal y a veces impredecible para una partícula en movimiento.

- Funciones Componentes:

Para simular el movimiento errático, se utilizarán múltiples funciones componentes. Cada una de estas funciones describirá el comportamiento de la partícula en un intervalo de tiempo particular. Estas funciones pueden ser polinomios, funciones trigonométricas, funciones aleatorias o cualquier otra función matemática que se adapte a los requisitos específicos del movimiento deseado.

- Transiciones entre Funciones:

Para crear un movimiento errático realista, es esencial definir cómo ocurren las transiciones entre las diferentes funciones componentes. Estas transiciones pueden ser suaves o abruptas, dependiendo de la naturaleza del movimiento que se desea modelar.

Por ejemplo, una transición suave puede representar una desaceleración gradual de la partícula antes de cambiar su dirección, mientras que una transición abrupta podría simular un cambio brusco en la velocidad o la dirección.

3. **Formulación**

Para utilizar funciones con movimiento erratico, utilizamos la siguiente función
piecewise('Intervalo 1', 'Funcion', 'Intervalo 2', 'Funcion');

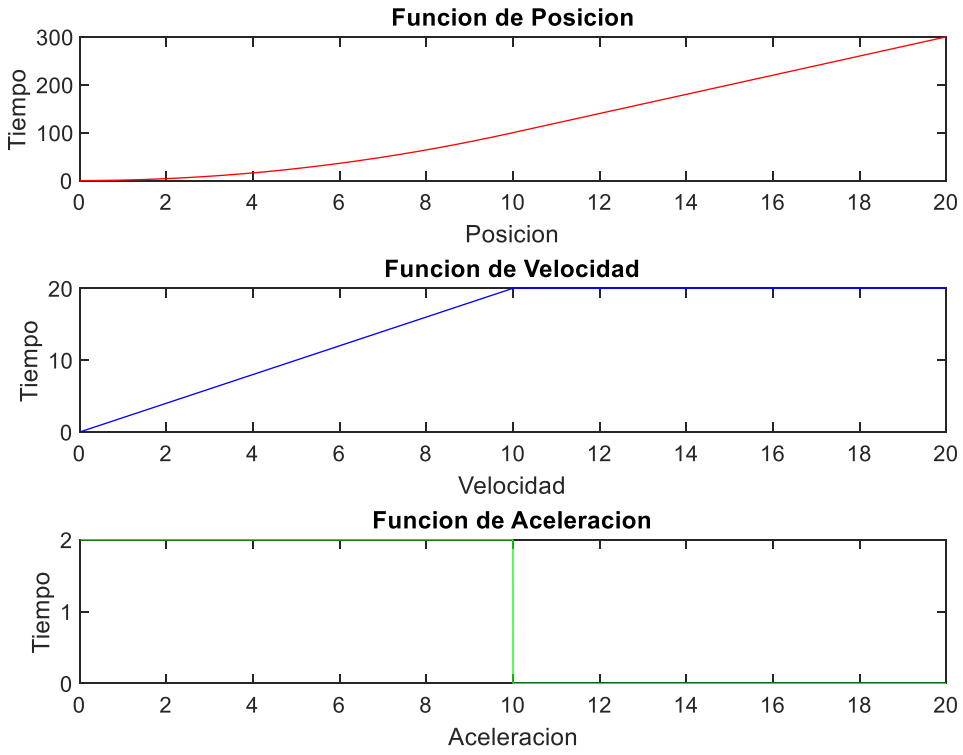
4. **Implementación en MATLAB y Resultados**

CODIGO

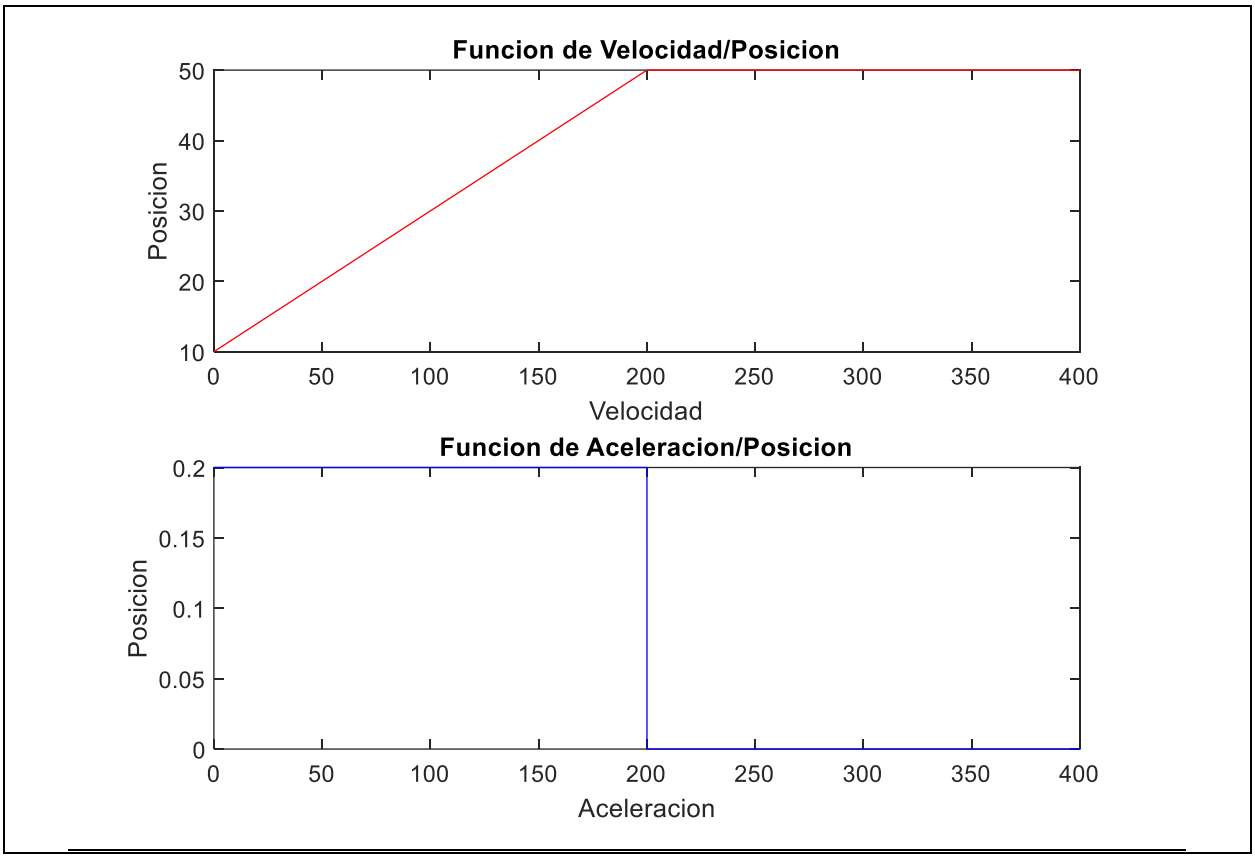
```
% EJERCICIO 12.6
syms t;
x1 = t^2;
x2 = 20*t-100;

x = piecewise(0 <= t < 10, x1, 10 <= t <= 20,x2);

figure;
subplot(3,1,1);
fplot(x,[0,20],'r');
title('Funcion de Posicion');
xlabel('Posicion');
ylabel('Tiempo');

v = diff(x,t);

subplot(3,1,2);
fplot(v,[0,20],'b');
title('Funcion de Velocidad');
xlabel('Velocidad');
ylabel('Tiempo');

a = diff(v,t);

subplot(3,1,3);
fplot(a,[0,20],'g');
title('Funcion de Aceleracion');
xlabel('Aceleracion');
ylabel('Tiempo');


% EJERCICIO 12.8
syms s;
v1 = 0.2*s+10;
```

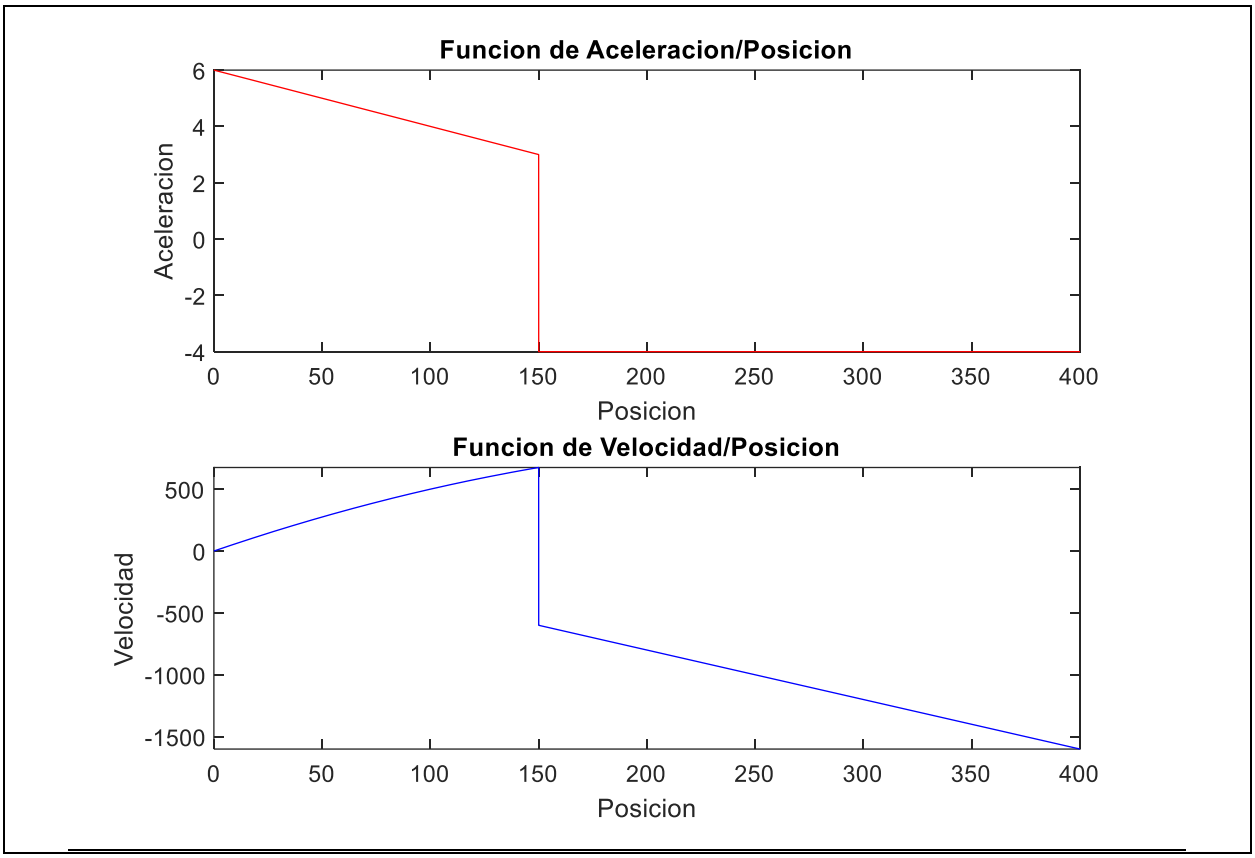
```
v2 = 50;

v = piecewise(0 <= s < 200, v1, 200 <= s <= 400,v2);

figure;
subplot(2,1,1);
fplot(v,[0,400],'r');
title('Funcion de Velocidad/Posicion');
xlabel('Velocidad');
ylabel('Posicion');

a = diff(v,s);

subplot(2,1,2);
fplot(a,[0,400],'b');
title('Funcion de Aceleracion/Posicion');
xlabel('Aceleracion');
ylabel('Posicion');

% EJERCICIO 12.62
a1 = -0.02*s+6;
a2 = -4;

a = piecewise(0 <= s < 150, a1, 150 <= s <= 400,a2);

figure;
subplot(2,1,1);
fplot(a,[0,400],'r');
title('Funcion de Aceleracion/Posicion');
xlabel('Posicion');
ylabel('Aceleracion');

v = int(a,s);

subplot(2,1,2);
fplot(v,[0,400],'b');
title('Funcion de Velocidad/Posicion');
xlabel('Posicion');
ylabel('Velocidad');

disp('Distancia antes de detenerse es de 150 metros');
disp('Velocidad Maxima es de 675');

% EJERCICIO 12.63
a1 = sqrt(36*t);
a2 = 4*t-18;

a = piecewise(0 <= t < 9, a1, 9 <= t <= 14,a2);

figure;
```

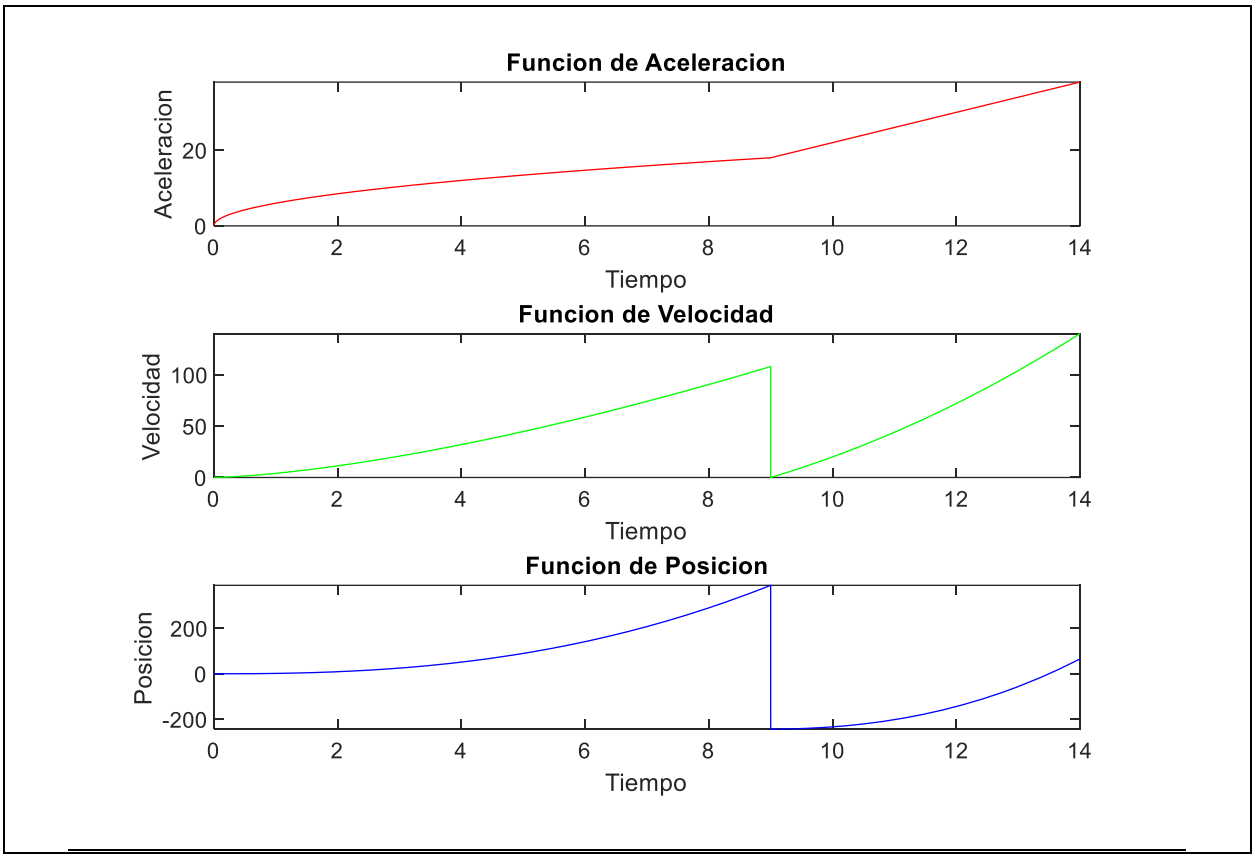
```
subplot(3,1,1);
fplot(a,[0,14],'r');
title('Funcion de Aceleracion');
xlabel('Tiempo');
ylabel('Aceleracion');

v = int(a,t);

subplot(3,1,2);
fplot(v,[0,14],'g');
title('Funcion de Velocidad');
xlabel('Tiempo');
ylabel('Velocidad');

s = int(v,t);

subplot(3,1,3);
fplot(s,[0,14],'b');
title('Funcion de Posicion');
xlabel('Tiempo');
ylabel('Posicion');
```



Diego Joel Zuñiga Fragoso
317684




Funcion de Aceleracion/Posicion
Aceleracion
Posicion
Funcion de Velocidad/Posicion
Velocidad
Posicion



5. Conclusión

Se ha alcanzado con éxito el objetivo de la práctica, que consistía en aprender a implementar funciones que representen un movimiento errático en MATLAB, utilizando funciones compuestas en intervalos de tiempo variables. A través de la aplicación de mis conocimientos previos en este software y su programación, pude cumplir con los objetivos de la práctica de manera eficiente, requiriendo un mínimo de investigación en Internet. Esto demuestra la utilidad de contar con una base sólida de conocimientos en MATLAB para abordar tareas de programación y simulación de movimientos complejos y no lineales.

Diego Joel Zuñiga Fragoso
317684

Obtención de ecuación de movimiento con regresión	PRACTICA # 5 FECHA 28/09/2023

1. Objetivo

En este trabajo, utilizaremos el software MATLAB para analizar el movimiento de una partícula a partir de los datos de velocidad que se nos proporcionan en el intervalo de 0 a 20 segundos. A partir de estos datos, realizaremos un regresion y obtendremos una ecuación de movimiento que describa la posición de la partícula en función del tiempo. Luego, graficaremos la posición, la velocidad y la aceleración de la partícula usando MATLAB y comentaremos los resultados obtenidos.

2. Marco Teórico

- Regresión no lineal:

La regresión no lineal es una técnica estadística que se utiliza para modelar la relación entre una variable dependiente y una o más variables independientes cuando esta relación no es lineal. En este caso, estamos tratando de modelar la posición de una partícula en función del tiempo a partir de los datos de velocidad proporcionados. Dado que la relación entre la posición y el tiempo no es lineal, se necesita una regresión no lineal para encontrar una función que describa con precisión el movimiento de la partícula.

- Ecuación de movimiento:

El objetivo principal de la regresión no lineal es obtener una ecuación de movimiento que relacione la posición de la partícula con el tiempo. Esta ecuación permitirá predecir la posición de la partícula en cualquier momento dentro del intervalo de tiempo especificado.

- Software MATLAB:

MATLAB es un software ampliamente utilizado en el campo de la ingeniería y la ciencia para realizar análisis numéricos y resolver problemas matemáticos. Proporciona herramientas y funciones poderosas para llevar a cabo análisis de regresión no lineal, ajustando modelos no lineales a datos experimentales.

3. **Formulación**

Para definir la función en la que realizamos la regresión usamos el siguiente comando:

inline('Función', 'Variable 1' , 'Variable 2');

Para realizar la regresión no lineal utilizamos el siguiente comando:

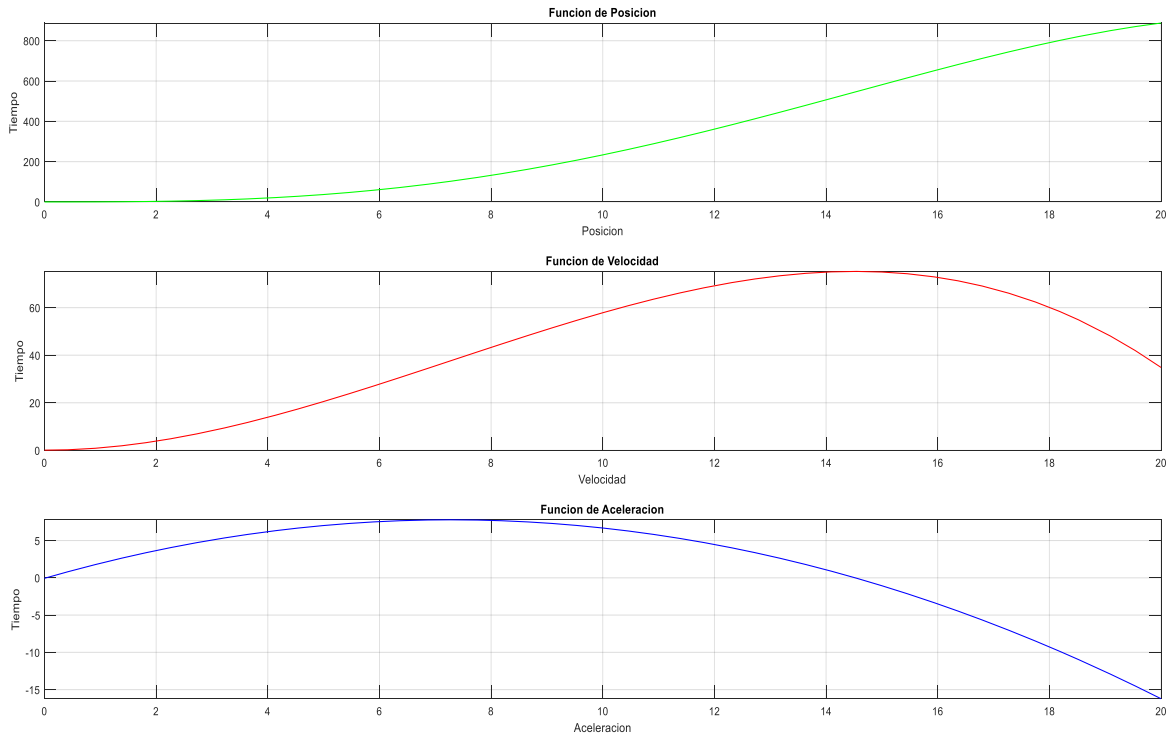
nlinfit('variable independiente' , 'Datos de variable dependiente' , 'Función de referencia' , 'Numero de coeficientes de función');

4. **Implementación en MATLAB y Resultados**

CODIGO

```
time = 0:20;
velocity = [0,1,4,8,14,21,28,35,43,51,58,64,69,73,75,75,73,68,60,49,35];

% Adaptamos los datos a una ecuacion de 2do grado

f2 = inline('x(1)*t.^2+x(2)*t+x(3)','x','t');
x2 = nlinfit(time,velocity,f2,[0 0 0]);
v2 = x2(1)*time.^2 + x2(2)*time + x2(3);

% No se adapto bien, por lo que adaptamos los datos a una ecuacion de 3er
grado

f3 = inline('x(1)*t.^3+x(2)*t.^2+x(3)*t+x(4)','x','t');
x3 = nlinfit(time,velocity,f3,[0 0 0 0]);
v3 = x3(1)*time.^3+x3(2)*time.^2+x3(3)*time+x3(4);

% Encaja perfercto por lo que ya tenemos la ecuacion

syms t;
v = x3(1)*t.^3+x3(2)*t.^2+x3(3)*t+x3(4);

% Calculamos funcion posicion y aceleracion

s = int(v,t);

a = diff(v,t);

% Graficamos

subplot(3,1,1);
```

```
fplot(s, [0 20], 'g');
hold on;
grid on;
title('Funcion de Posicion');
xlabel('Posicion');
ylabel('Tiempo');

subplot(3,1,2);
fplot(v, [0 20], 'r');
hold on;
grid on;
title('Funcion de Velocidad');
xlabel('Velocidad');
ylabel('Tiempo');

subplot(3,1,3);
fplot(a, [0 20], 'b');
hold on;
grid on;
title('Funcion de Aceleracion');
xlabel('Aceleracion');
ylabel('Tiempo');
```



Con la función de MATLAB pudimos obtener una función prácticamente perfecta para los datos de velocidad que nos brindaron, y a partir de esta pudimos obtener la ecuación de posición y velocidad.

5. Conclusión

Se cumplió el objetivo de realizar la regresión líneal a partir de los datos recibidos, me asombra la capacidad del software MATLAB y como realiza tantos cálculos de manera instantánea. Cada vez estoy mas familiarizado con el software y he aprendido más funciones.

Diego Joel Zuñiga Fragoso
317684

Cinética con ecuación del movimiento	PRACTICA # 6 FECHA 12/10/2023

1. **Objetivo**

A través del software MATLAB, abordaremos problemas de cinética utilizando los conceptos que hemos aprendido en clase, con el objetivo de aplicarlos de manera más práctica y efectiva en esta plataforma.

2. **Marco Teórico**

- Dinámica y Cinética:

La dinámica es una rama de la física que se enfoca en el estudio de los movimientos de los objetos y las fuerzas que los impulsan. La cinética, en particular, se centra en entender y predecir cómo los objetos cambian su posición, velocidad y aceleración en función del tiempo y las fuerzas involucradas.

- Segunda Ley de Newton:

La ecuación "Fuerza = Masa x Aceleración" representa la segunda ley de Newton, que establece que la aceleración de un objeto es directamente proporcional a la fuerza neta que actúa sobre él e inversamente proporcional a su masa. Esta ley es fundamental para comprender cómo los objetos reaccionan a las fuerzas que experimentan.

- Tipos de Fuerzas:

En esta práctica, se explorarán varios tipos de fuerzas, como la fuerza gravitatoria, la fricción, la tensión en cuerdas o resortes, entre otras. Comprender la magnitud y la dirección de estas fuerzas es esencial para aplicar la segunda ley de Newton de manera efectiva.

3. **Formulación**

Para esta practica empleamos un nuevo comando solve()

Este comando nos permite resolver ecuaciones, buscando una variable, los parámetros se ponen de la siguiente forma

a = solve (“Ecuación” , “Variable a despejar”);

La función nos devuelve el valor de la variable despejado de la ecuación.

4. Implementación en MATLAB y Resultados

- **Problema 13.7:**

Código Utilizado

```
%% Definicion de Variables
g = 9.807; % m/s^2
m = 250; %Kg
w = m*g; %N
d = 45; %m
v = 20*(1/3600)*(1000); %m/s

%% Solucion de la Problematica
t = d/v;

syms a;
a = solve(-v^2 == 2*a*d, a);
F = m*a;

% Display the results
fprintf('\n Aceleracion = %1.2f m/s^2', a);
fprintf('\n Fuerza F horizontal = %1.2f N', F);
```

Resultados

```
>> Practica6

 Aceleracion = -0.34 m/s^2
 Fuerza F horizontal = -85.73 N>>
```

- **Problema 13.9:**

Código Utilizado

```
%% Definicion de Variables
% Aceleracion
g = 9.807; % m/s^2

% Masa (Kg)
m_123 = 30000;
m_R = 12000;
```

```
% Fuerzas (N)
w_123 = m_123*g;
w_R = m_R*g;
F_h2o = 2000;
F_rem = 1500;

% Velocidad (m/s)
v = 4;

%% Solucion de la Problematica
syms F_T;   syms a;

F_T = solve(F_T - F_rem - 3*F_h2o == 0, F_T);
a = solve(F_T - F_rem - 2*F_h2o == (m_R + 2 * m_123)* a, a);

% Display the results
fprintf('\n Aceleracion = %1.4f m/s^2', a);
fprintf('\n Fuerza F_T horizontal = %1.2f N', F_T);
```

Resultados

```
Aceleracion = 0.0278 m/s^2
Fuerza F_T horizontal = 7500.00 N>>
```

- **Problema 13.71:**

Código Utilizado

```
%% Definicion de Variables

% Aceleracion
g = 9.807; % m/s^2
% Masa (Kg)
m = 5000;
% Fuerzas (N)
w = m*g;
% Velocidad (m/s)
v = 350*(1/3600)*(1000);
% Angulo (°)
theta = 15;

%% Solucion de la Problematica
syms F_L;   syms a;

F_L = w/cosd(theta);
r = (v^2*m)/(sind(theta)*F_L);

% Display the results
fprintf('\n Radio = %1.4f m', r);
fprintf('\n Fuerza de elevacion L = %1.2f N', F_L);
```

Resultados

```
Radio = 3597.0167 m
Fuerza de elevacion L = 50764.77 N>>
```

5. **Conclusión**

Se ha cumplido con éxito el objetivo de la práctica, y he tenido la oportunidad de fortalecer significativamente mis habilidades en el uso del software. Es impresionante la velocidad y precisión con la que MATLAB resuelve problemas matemáticos, además de la amplia variedad de funciones que ofrece.

Diego Joel Zuñiga Fragoso
317684

Sistema de masa, amortiguamiento y resorte	PRACTICA # 7 FECHA 16/11/2023

1. **Objetivo**

Los sistemas masa-resorte-amortiguador, omnipresentes en la ingeniería y la física, constituyen un modelo esencial para comprender fenómenos oscilatorios. Este sistema, compuesto por una masa, un resorte y un amortiguador, exhibe dinámicas vibracionales influenciadas por la amortiguación y la rigidez. La modificación de estos parámetros desempeña un papel clave en la manipulación de la amplitud y la frecuencia de las oscilaciones, aspectos cruciales para el diseño eficiente de estructuras y dispositivos vibracionales. En esta investigación, exploraremos cómo la variación de la amortiguación y la rigidez impacta en la dinámica de estos sistemas.

2. **Marco Teórico**

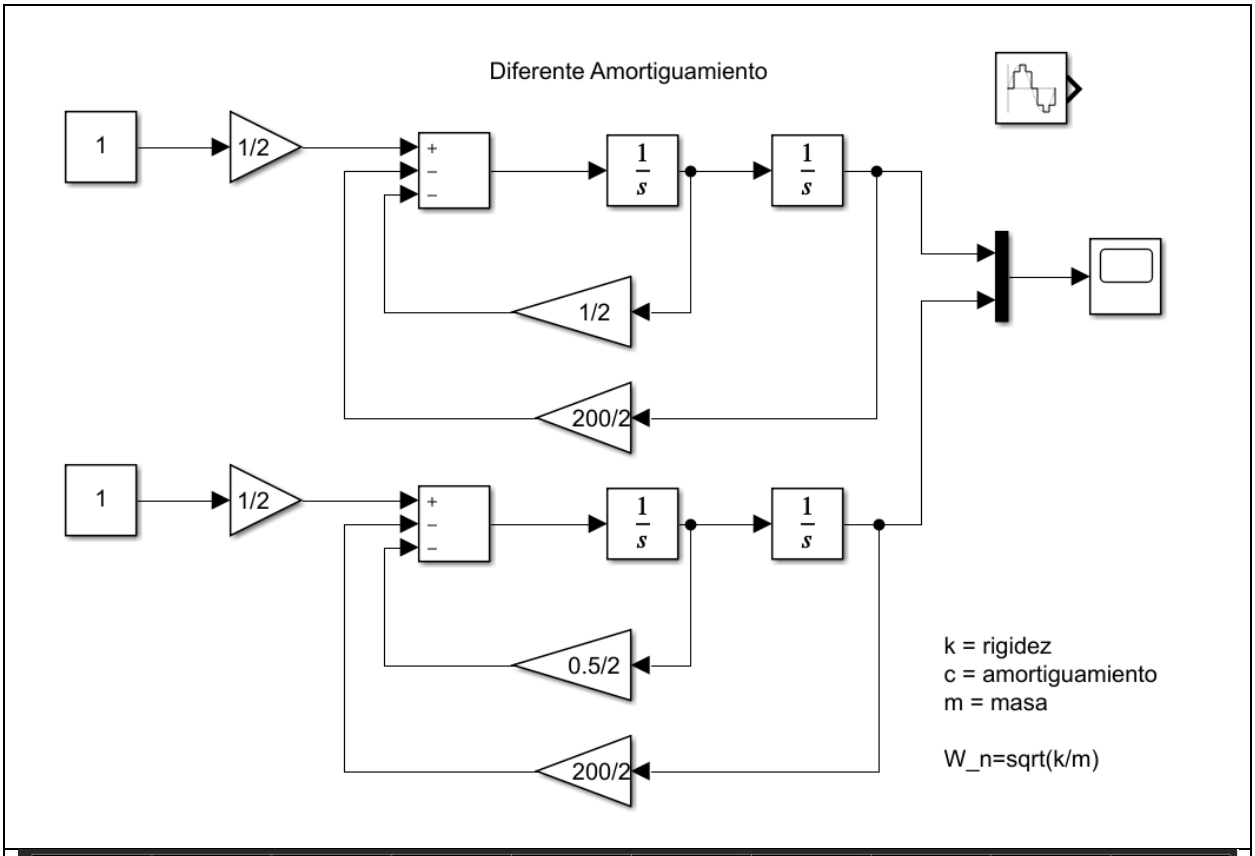
- Amortiguación y Amplitud:

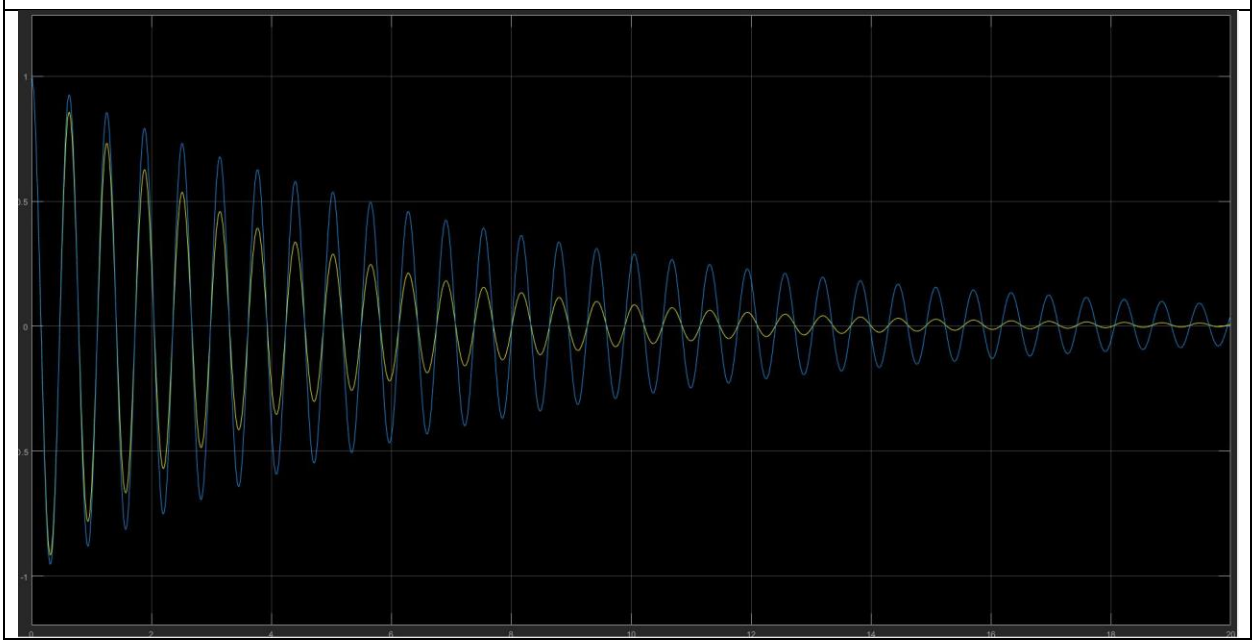
La amortiguación regula la magnitud de las oscilaciones en el sistema. Un mayor coeficiente de amortiguación conduce a una disminución más rápida de la amplitud de las oscilaciones, ya que la energía vibracional se disipa más eficientemente. Este fenómeno tiene implicaciones cruciales en el diseño de sistemas donde se busca controlar y minimizar la amplitud de las vibraciones, como en estructuras sismorresistentes o en la suspensión de vehículos.

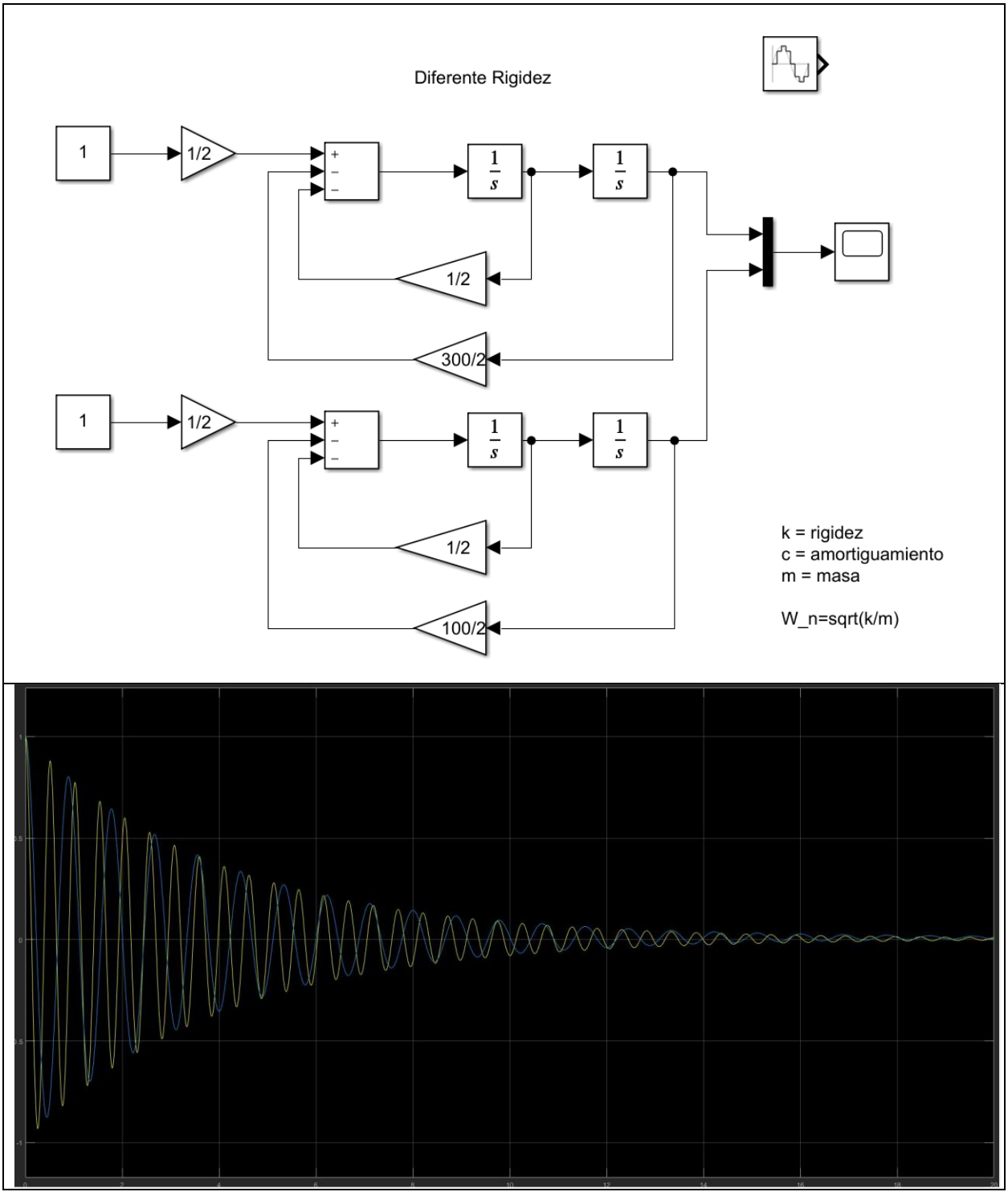
- Rigidez y Frecuencia:

La rigidez del resorte, representada por la constante elástica
k, influye en la frecuencia natural del sistema. Un resorte más rígido conduce a una frecuencia más alta de oscilación, ya que la fuerza restauradora es más intensa. Esta relación es vital en aplicaciones como la sintonización de dispositivos mecánicos resonantes, donde se busca optimizar la respuesta a frecuencias específicas.

3. Implementación en SIMULINK y Resultados






Diferente Rigidez
1
1/2
1/s
1/s
1/2
300/2
1
1/2
1/s
1/s
1/2
100/2
k = rigidez
c = amortiguamiento
m = masa
W_n=sqrt(k/m)

En sistemas oscilatorios como el masa-resorte-amortiguador, es evidente que la variación en la amortiguación tiene un impacto directo en la amplitud de la onda resultante. La amortiguación influye en la capacidad del sistema para disipar la energía vibracional, lo que a su vez afecta la magnitud de las oscilaciones. Un aumento en la amortiguación tiende a reducir la amplitud, ya que la energía se disipa más rápidamente, resultando en un decaimiento más rápido de las oscilaciones.

Por otro lado, la rigidez del resorte afecta la frecuencia del sistema. Un aumento en la rigidez conlleva a una mayor fuerza restauradora, lo que acelera el proceso de oscilación. En consecuencia, se observa un cambio en la frecuencia de las oscilaciones. Este fenómeno ilustra cómo la rigidez es fundamental para determinar la rapidez con la que el sistema regresa a su posición de equilibrio después de ser perturbado

4. **Conclusion**

La práctica alcanzó con éxito su objetivo. Aprovechando la herramienta de simulación Simulink, pudimos visualizar a través de un osciloscopio simulado las variaciones en el comportamiento de los sistemas masa-resorte-amortiguador al manipular ciertos parámetros. Además, esta experiencia significó mi primera incursión en el uso del software, brindándome la oportunidad de familiarizarme más con sus funcionalidades. Este ejercicio no solo amplió mi comprensión de los sistemas oscilatorios, sino que también consolidó mi habilidad en la utilización de Simulink como una valiosa herramienta de análisis y diseño en el ámbito de la ingeniería.