



INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA EN
INGENIERÍA Y TECNOLOGÍAS AVANZADAS

Trabajo Terminal I

**“Optimización basada en métodos metaheurísticos
para la sintonización de controladores PID en
sistemas mecatrónicos”**

*Que para obtener el título de
“Ingeniero en Mecatrónica”*

Presentan:

**Cortéz Conde Alexander
Valdez Cruz Marco Antonio**

Asesores:

M. en C. Cuervo Pinto Victor Darío
M. en C. Fonseca Campos Jorge
Dr. Rodríguez Molina Alejandro



Diciembre 2021

INSTITUTO POLITÉCNICO NACIONAL



UNIDAD PROFESIONAL INTERDISCIPLINARIA EN
INGENIERÍA Y TECNOLOGÍAS AVANZADAS

Trabajo Terminal I

**“Optimización basada en métodos metaheurísticos
para la sintonización de controladores PID en
sistemas mecatrónicos”**

Que para obtener el título de

“Ingeniero en Mecatrónica”

Presenta:

Cortéz Conde Alexander

Valdez Cruz Marco Antonio

Asesores:

M. en C. Cuervo Pinto

Victor Darío

M. en C. Fonseca Campos

Jorge

Dr. Rodríguez Molina

Alejandro

Presidente del Jurado

Profesor titular

Dr. Adolfo Rojas Pacheco

M. en E. Rivas Bonilla
Elizabeth



DEDICATORIA

AGRADECIMIENTOS

Contenido

Resumen/Abstract	VII
Abreviaturas	XVII
Simbología	XIX
Introducción	XXIII
0.1. Definición del problema	XXIII
0.2. Justificación	XXVI
0.3. Objetivo	XXVIII
Objetivos específicos	XXVIII
0.4. Antecedentes	XXIX
1. Marco Referencial	1
1.1. Marco Histórico	1
Control Automático	1
Optimización	2

CONTENIDO

1. Marco teórico		3
Controladores automáticos	3	
Métodos de integración	7	
Dinámica	8	
Cinemática	9	
1.3. Marco conceptual	13	
Métodos metaheurísticos	16	
Algoritmos Evolutivos	18	
Algoritmos genéticos (GA)	21	
Algoritmos de evolución diferencial (DE)	22	
Optimización por Enjambre de partículas (PSO)	24	
Optimización multiobjetivo (MOP)	26	
Índices de desempeño de un sistema controlado	27	
Indicadores de Calidad	29	
Pruebas no paramétricas	31	
Método Analítico Jerárquico (AHP)	33	
1.4. Marco procedimental	35	
Sintonización de controladores multiobjetivo	35	
Optimización multiobjetivo con métodos metaheurísticos para la sin-		
tonización de controladores	37	
Metodología	38	
2. Diseño del sistema	43	
2.1. Diseño conceptual	43	
Necesidades y requerimientos	43	
Arquitectura funcional	45	



Arquitectura física	51
Matriz morfológica	52
Búsqueda morfológica	53
Selección de diseño conceptual	55
Propuesta solución	58
Validación	77
2.2. Diseño detallado	81
Modelado del sistema.	81
Sintonización con métodos metaheurísticos para optimización multi- objetivo.	81
Interfaz Gráfica (GUI).	81
Adaptación del péndulo simple.	87
2.3. Validación e integración de los sistemas	103
3. Conclusiones	109
3.1. Conclusiones	109
Referencias	113
Apéndices	121
Apéndice A: Modelos dinámicos	123
.1. Péndulo simple	123
.2. Péndulo invertido	126
.3. Péndulo doble	131
Apéndice B: Códigos de simulaciones de los pendulos	139

CONTENIDO

	❀❀❀	
.4.	Simulación de la dinámica del péndulo simple	139
.5.	Simulación de la dinámica del péndulo invertido	144
.6.	Simulación de la dinámica del péndulo doble	149
.7.	Algoritmo de control en Arduino	154
Apéndice C: Sintonización multi-objetivo.		159
.8.	Péndulo simple	159
	DE	159
Anexos		171
	Plano de montaje	173
	Plano de conjunto	174
	Plano de despiece	175
	Interfaz gráfica	178
Glosario		183

Resumen/Abstract

Resumen

La optimización en el proceso de sintonización permite encontrar el conjunto de valores de ganancias del controlador que cumpla con el comportamiento deseado en el sistema. De manera práctica, en la industria se busca que el control cumpla con varias características de rendimiento, sin embargo, cuando no se trata de una sintonización global (de un solo objetivo), es conveniente tratar el problema de sintonización como uno de optimización multiobjetivo, el cual puede ser resuelto usando métodos metaheurísticos.

En el presente trabajo se propone el estudio de la sintonización de controladores de tipo PID (Proporcional, Integral, Derivativo) para el control de distintos sistemas mecatrónicos comúnmente utilizados como lo son el péndulo simple, el péndulo doble y el péndulo invertido. Así, el problema de sintonización del controlador de tipo PID se aborda como uno de optimización multiobjetivo, en el que se pretende minimizar el error mediante la Integral del Error Cuadrático (ISE por sus siglas en inglés) y las variaciones en la señal de control mediante la Integral del valor Absoluto de la señal de control (IADU por sus siglas en inglés). Lo anterior, con la intención de



obtener un compromiso adecuado entre la precisión en la regulación de la salida y el tiempo de vida del sistema mecatrónico. Debido a su complejidad, el problema de optimización se resuelve utilizando alternativas multiobjetivo de distintos algoritmos metaheurísticos ampliamente conocidos: Evolución Diferencial, Optimización por Enjambre de Partículas y Algoritmo Genético (respectivamente DE, PSO y GA por sus siglas en inglés).

Este estudio permitirá al ingeniero mecatrónico observar el funcionamiento de la optimización metaheurística en la sintonización de controladores de tipo PID para sistemas mecatrónicos ampliamente usados.

Palabras Clave:

Optimización multiobjetivo, sintonización, algoritmo genético, evolución diferencial, enjambre de partículas, control PID, Frente de Pareto

Abstract

Optimization in the tuning process makes it possible to find the set of controller gain values that meets the desired behavior in the system. In a practical way, in the industry it is sought that the control meets several performance characteristics, however, when it is not a global tuning (of a single objective), it is convenient to treat the tuning problem as one of multi-objective optimization, which can be solved using metaheuristic methods.

In this work we propose the study of the tuning of PID type controllers (Proportional, Integral, Derivative) for the control of different commonly used mechatronic systems such as the simple pendulum, the double pendulum and the inverted pendulum. Thus, the tuning problem of the PID-type controller is approached as one of multiobjective optimization, in which it is intended to minimize the error through the Integral of the Quadratic Error (ISE) and the variations in the control signal through the Integral of the Absolute value of the control signal (IADU for its acronym in English). The above, with the intention of obtaining an adequate compromise between

the precision in the regulation of the output and the life time of the mechatronic system. Due to its complexity, the optimization problem is solved using multiobjective alternatives of different widely known metaheuristic algorithms: Differential Evolution, Particle Swarm Optimization and Genetic Algorithm (respectively DE, PSO and GA for its acronym in English).

This study will allow the mechatronic engineer to observe the performance of metaheuristic optimization in tuning PID-type controllers for widely used mechatronic systems.

Key words:

Multi-objective optimization, tuning, genetic algorithm, differential evolution, swarm particle, PID control.

Índice de figuras

1.1. Diagrama de bloques de un sistema de control industrial Fuente: Adaptado de	4
1.2. Perfil de velocidad triangular	12
1.3. Perfil de velocidad trapezoidal	12
1.4. Metodología de diseño por medio de la optimización	14
1.5. Diagrama de un EA	20
1.6. Hipervolumen (HV).	31
1.7. Pasos típicos para la sintonización de controladores multiobjetivo. .	36
1.8. MOP optimizado con un método metaheurístico	38
1.9. Modelo en V.	39
1.10. Macro ciclos y madurez del producto	41
2.1. Diagrama IDEF0.	47
2.2. Diagrama IDEF0 Nodo A0.	49
2.3. Diagrama IDEF0 Nodo A2.	50

ÍNDICE DE FIGURAS



2.4. Matriz morfológica.	52
2.5. Ruta 1	53
2.6. Ruta 2	54
2.7. Ruta 3	55
2.8. Péndulo simple.	58
2.9. Posición del péndulo simple en el plano XY.	60
2.10. Gráficas de salida con respecto al tiempo del péndulo simple a $\theta = 0$	60
2.11. Posición del péndulo en el plano XY.	61
2.12. Gráficas de salida con respecto al tiempo del péndulo simple a $\theta = \frac{\pi}{2}$	61
2.13. Péndulo invertido.	62
2.14. Posición del péndulo invertido en el plano XY.	64
2.15. Gráficas de salida respecto al tiempo péndulo invertido a $\theta = x = 0$	65
2.16. Posición en el plano XY del péndulo invertido.	65
2.17. Gráficas de salida con respecto al tiempo del péndulo invertido a $\theta = \frac{\pi}{2}$, $x = 0$	66
2.18. Posición en el plano XY (Péndulo doble).	69
2.19. Gráficas de salida con respecto al tiempo del Péndulo doble a condiciones iniciales $\theta_1 = \theta_2 = 0$	70
2.20. Posición en el plano XY (Péndulo doble).	70
2.21. Gráficas de salida con respecto al tiempo del péndulo doble a condiciones iniciales $\theta_1 = \frac{\pi}{2}$, $\theta_2 = 1$	71
2.22. ISE vs ISE.	78
2.23. IADU vs IADU.	79
2.24. Diagrama de flujo de la GUI.	83
2.25. Selección de materiales	87
2.26. Sistema péndulo simple.	90

ÍNDICE DE FIGURAS



2.27. Motor.	90
2.28. Eje.	91
2.29. Chumacera	91
2.30. Cople	92
2.31. Elemento de sujeción.	92
2.32. Brazo de aluminio.	93
2.33. Motor con encoder	
Fuente: Modificado de [1].	97
2.34. Teensy 4.1.	98
2.35. Configuración de pines.	100
2.36. Circuito de control.	101
2.37. Módulo L298N	102
2.38. Frente de Pareto	103
2.39. Circuito de control del péndulo.	104
2.40. Ejecución de simulación.	104
2.41. Convertidor de nivel lógico de 5 V a 3.3 V.	105
2.42. Péndulo simple (Control).	106
2.43. Péndulo simple (Control).	106
1. Péndulo simple.	123
2. Péndulo invertido.	126
3. Péndulo doble.	131
4. Plano de montaje.	173
5. Plano de conjunto.	174
6. Plano del eje.	175
7. Plano de la barra.	176

ÍNDICE DE FIGURAS



8.	Plano del bloque	177
9.	Pantalla de inicio.	178
10.	Ingreso de datos péndulo simple.	178
11.	Frente de Pareto péndulo simple.	179
12.	Simulación dinámica péndulo invertido.	179
13.	Ingreso de datos péndulo invertido.	180
14.	Simulación dinámica péndulo invertido.	180
15.	Ingreso de datos péndulo doble.	181
16.	Simulación dinámica péndulo doble.	181

Índice de Tablas

2.1. Necesidades del sistema	44
2.2. Requerimientos	44
2.3. Matriz de trazabilidad	45
2.4. Familia IDEF	46
2.5. Matriz de trazabilidad para la validación de funciones.	51
2.6. Variables y criterios.	55
2.7. Matriz de comparación de criterios.	56
2.8. Criterio: Número de elementos	56
2.9. Criterio: Comunicación motor-sensor	56
2.10. Criterio: Ensamble	57
2.11. Selección de ruta	57
2.12. Variables y criterios.	72
2.13. Matriz de comparación de criterios.	72
2.14. Criterio: Supresión de errores.	72
2.15. Criterio: Implementación digital.	73

ÍNDICE DE TABLAS



2.16. Criterio: Evaluación analítica.	73
2.17. Selección de índice de desempeño.	73
2.18. Análisis de ventajas y desventajas en la selección de materiales.	88
2.19. Matriz de comparación de criterios	95
2.20. Criterio: Resolución.	96
2.21. Criterio:Torque	96
2.22. Criterio:Precio	96
2.23. Criterio: Consumo energético	96
2.24. Selección de motor con encoder	97
1. Notación péndulo simple.	124
2. Notación péndulo invertido.	127
3. Notación péndulo doble.	132

Abreviaturas

<i>2D</i>	Bidimensional.
<i>AHP</i>	Analytic Hierarchy Process method (Método de Proceso Analítico Jeraquíco).
<i>DE</i>	Differential Evolution (Evolución Diferencial).
<i>FP</i>	Frente de Pareto.
<i>EA</i>	Evolutionary algorithm(Algoritmo evolutivo).
<i>GA</i>	Genetic Algorithm (Algoritmo Genético).
<i>HV</i>	Hipervolumen.
<i>IADU</i>	Integral of the Absolute value of the control action (Integral del valor absoluto de la variación de la señal de control).
<i>IDE</i>	Integrated Development Environment (Entorno de Desarrollo Integrado).
<i>IDEF</i>	Integration Definition for Function Modeling (Definición de integración de modelado de funciones).
<i>ISE</i>	Integral Square Error (Integral del error al cuadrado).
<i>MOEAs</i>	Multi-objective evolutionary algorithms (Algoritmos evolutivos multi-objetivo).

Abreviaturas



<i>MOO</i>	Multi-objective optimisation (Optimización multi-objetivo).
<i>MOP</i>	Multi-objective Problem (Problema multi-objetivo).
<i>NSGA-II</i>	Non-dominated Sorting Genetic Algorithm II (Algoritmo genético de clasificación no dominado II)
<i>PID</i>	Proporcional,Integral, Derivativo.
<i>PSO</i>	Particle Swarm Optimization (Optimización por enjambre de partículas).
<i>PWM</i>	Pulse-Width Modulation (Modulación por ancho de pulsos).
<i>S</i>	Conjunto solución óptimo.
<i>VDI</i>	Verein Deutscher Ingenieure (Asociación de Ingenieros Alemanes).

Simbología

b	Fricción viscosa del péndulo simple.
b_1	Fricción viscosa del carro (Péndulo invertido) \ Fricción viscosa del brazo 1 (Péndulo doble).
b_2	Fricción viscosa del brazo 2.
$C(\mathbf{q}), \dot{\mathbf{q}}$	Matriz de fuerzas centrípetas y de Coriolis.
Cr	Probabilidad de recombinación.
$\mathbf{D}(\dot{\mathbf{q}}, \mathbf{f}_e)$	Vector de fuerzas o pares de fricción.
$e(t)$	Señal de error.
F	Factor de escalado
\mathbf{f}_e	Fuerza de fricción estática.
$\mathbf{f}_f(\dot{\mathbf{q}}, \mathbf{f}_e)$	Vector de pares de fricción viscosa, Coulomb y estática.
G	Contador de generaciones.
$\mathbf{g}(\mathbf{q})$	Vector de fuerzas o pares gravitacionales.
g	Aceleración de la gravedad.
$g(x)$	Restricción de desigualdad.
$h(x)$	Restricción de igualdad.
I	Inercia de la barra.
I_1	Inercia del brazo 1.

I_2	Inercia del brazo 2.
$J(q)$	Jacobiano analítico.
$J(x) \in m$	Vector de objetivos de diseño (MOP).
J_{IADU}	Función objetivo IADU.
J_{ISE}	Función objetivo ISE
K	Energía cinética.
k_d	Ganancia derivativa.
k_i	Ganancia integral.
k_p	Ganancia proporcional.
L	Lagrangiano del sistema.
l_1	Longitud del brazo 1.
l_2	Longitud del brazo 2.
l_c	Longitud al centro de masa del barra.
l_{c1}	Longitud al centro de masa del brazo 1.
l_{c2}	Longitud al centro de masa del brazo 2.
M	Masa del carro.
$M(\mathbf{q})$	Matriz de inercia.
m	Masa de la barra.
m_1	Masa del brazo 1.
m_2	Masa del brazo 2.
\mathbf{q}	Vector de posiciones articuladas o coordenadas generalizadas.
$\dot{\mathbf{q}}$	Vector de velocidades articuladas.
$\tilde{\mathbf{q}}$	Error de posición.
$\dot{\tilde{\mathbf{q}}}$	Error de velocidad.
\mathbf{q}_d	Vector de posición deseada.
$P _0$	Población inicial.
$P _G$	Población actual .
$P _G^*$	Nueva población de GA.

$P _{G+1}$	Población siguiente.
$S1$	Sistema Róbótico.
$S2$	Sistema de Información.
$S3$	Sistema de administración de energía.
T_d	Tiempo derivativo.
T_i	Tiempo integral.
U	Energía potencial.
$u_{i,j,G0}$	Vector hijo.
$u(t)$	Señal de salida del controlador
v	Velocidad lineal.
$v_{i,G}$	Vector mutante.
\mathbf{x}	Vector de estado.
$\dot{\mathbf{x}}$	Ecuación de estado.
$x \in R^n$	Vector de variables de desición (MOP).
$x^j _{best}$	Mejor posición conocida para la partícula j (PSO).
$x^j _k$	Posición de una partícula en la generación k (PSO).
$x^{swarm} _{best}$	Mejor posición conocida por el enjambre (PSO).
$\dot{x}^j _k$	Velocidad de la paítucla en la generación k (PSO).
\underline{x}_i	Límite inferior del espacio de búsqueda (MOP).
\bar{x}_i	Límite superior del espacio de búsqueda (MOP).
β_1, β_2	Factores de redonde de crecimiento global.
θ	Posición angular de la barra.
θ_1	Posición angular del brazo 1.
θ_2	Posición angular del brazo 2.
τ	Vector de pares aplicados.
φ_1, φ_2	Factores de redondeo de crecimiento local
ω	Velocidad angular.

0.1. Definición del problema

La mecatrónica es una de las ingenierías que ha tomado un lugar muy importante en el mundo, al combinar distintas ramas como la electrónica, mecánica, computación y control, de modo que los procesos requeridos en la industria cada día requieren de una mayor participación de esta ingeniería. El control de sistemas involucra áreas de conocimiento de la mecatrónica, pues se requiere la obtención del modelo, ya sea mecánico, electrónico, o la combinación de ambos. Obtener el modelo es necesario para poder aplicar algún tipo de controlador. El controlador más usado en la industria es el PID, este incluye las ganancias: proporcional, integral y derivativa y el proceso de obtención de estas es conocido como sintonización.

Los métodos de sintonización se han clasificado según su naturaleza y uso en [2]:

- Métodos analíticos donde las ganancias de control se obtienen analizando la estabilidad del sistema en lazo cerrado.
- Métodos heurísticos donde la experiencia se considera con el ajuste manual del diseño del controlador para establecer los parámetros del controlador.
- Métodos de optimización donde se plantea un problema de programación ma-



temática y se optimiza utilizando técnicas de optimización para obtener las ganancias fijas del controlador.

- Métodos de ajuste adaptativo donde un proceso de identificación y combinación de los tres métodos anteriores se utilizan para ajustar las ganancias de control.

La Inteligencia Artificial y algunas ramas de ella, como lo son la lógica difusa, las redes neuronales, y el cómputo evolutivo, presentan una ventaja ante los métodos tradicionales de sintonización de controladores, pues estos buscan la optimización del sistema mediante algoritmos computacionales. Actualmente con la evolución de la informática y uso de los sistemas inteligentes, es mucho más conveniente usar algoritmos computacionales que nos proporcionen mejores soluciones a los problemas de control que se nos presenten. Debido al incremento de máquinas de precisión con un requerimiento de compensación, se ha abordado el problema del ajuste de control con el uso de métodos de optimización. En las últimas décadas, los algoritmos metaheurísticos y, en particular, los algoritmos evolutivos (EA, por sus siglas en inglés) [3], cuya inspiración se toma de la naturaleza de la teoría de la evolución y la supervivencia de los más aptos, han sido utilizados como una alternativa exitosa para el ajuste del controlador basado en métodos de optimización [4, 5, 6], ya que pueden manejar de manera eficiente las compensaciones altamente no lineales entre múltiples indicadores de rendimiento de lazo cerrado, e incorporan mecanismos (flexibilidad) para mejorar su convergencia y diversidad.

Una de las estrategias más utilizadas para evaluar el comportamiento del sistema de control, es el uso de indicadores de rendimiento, los cuales pueden medir características como la suavidad de la señal de control o la minimización del error mediante integraciones. Cabe recalcar que se puede considerar más de un indicador al mismo tiempo. Al plantearse más de un requerimiento o función objetivo en el diseño del controlador, obtenemos un problema de optimización multiobjetivo, pues se busca un conjunto de soluciones con distintos niveles de compromiso que cumpla con los requerimientos establecidos inicialmente.



Los problemas de optimización multiobjetivo se pueden resolver mediante la implementación de algoritmos metaheurísticos, como lo son la Evolución Diferencial, el Algoritmo Genético, y la técnica de Optimización por Enjambre de Partículas. Se proponen específicamente estos tres algoritmos debido a su uso extendido en la literatura y a su efectividad, además de realizar un estudio de la sintonización de controladores usando las técnicas metaheurísticas, para lograr una sintonización automática, tratando la sintonización como un problema de optimización multiobjetivo, y buscando que sea aplicable a sistemas mecatrónicos comunes, como son el péndulo, el péndulo doble, y el péndulo invertido. El propósito de usar los sistemas ya mencionados, es validar los resultados de manera sencilla, evaluando los objetivos mediante los indicadores de desempeño; ISE e IADU, por lo que en este proyecto se propone programar tres diferentes algoritmos evolutivos de uso recurrente en la literatura, que nos proporcionen las ganancias k_p, k_i, k_d adecuadas para satisfacer distintos niveles de compromiso que pueden presentarse en la sintonización de controladores PID. Como se ha mencionado con anterioridad, el péndulo simple, invertido y doble son sistemas usados con recurrencia en el sector industrial y por esta razón, han sido seleccionados para ejecutar un control de posición sobre ellos a través de una simulación computacional. Al mismo tiempo, se indaga en la implementación de un control PID para regular la posición de una barra de aluminio que fungirá como brazo de un péndulo simple, a través del movimiento de un motor, y con ayuda de un encoder, obtener los datos para su procesamiento por medio de una tarjeta de desarrollo. Los resultados obtenidos serán comparados con los de la simulación, buscando que el sistema sea estable en un tiempo reducido, así como minimizar el sobreimpulso y el error de las señales del sistema. Finalmente, se propone implementar una interfaz de usuario para los diseñadores de controladores, en la cual se podrán visualizar los resultados de la sintonización y una simulación 2D de la respuesta del sistema.



0.2. Justificación

En la mecatrónica, uno de los tópicos con mayor peso es la optimización, ya que la búsqueda de mejorar un determinado sistema es constante. Esto con el objetivo de aumentar la productividad y/o economizar algún proceso (generalmente industrial). Por lo tanto, el desarrollo de este estudio pretende aportar una relevancia significativa al área ya mencionada, de modo que se tratará a la sintonización de controladores como un problema de optimización. Dicho de otra forma, se trata de calcular o determinar el valor de las variables que intervienen en un sistema o proceso para minimizar o maximizar el valor de una función objetivo. Dado que el control y la automatización tienen un papel muy importante dentro de la mecatrónica, el implementar y corroborar nuevos métodos y técnicas para el diseño y sintonización de controladores puede incrementar la forma de controlar estos sistemas, pues al ser métodos comprobados brindan una mayor confianza al momento de implementarlos. En el planteamiento del problema, se mencionaron los sistemas que serán utilizados durante el estudio para comprobar el funcionamiento de la sintonización basada en optimización metaheurística multiobjetivo, por lo que es conveniente mencionar algunas aplicaciones que tienen estos sistemas o plantas, pues se encuentran en una amplia diversidad de contextos.

Aplicaciones del péndulo simple:

- En edificios para contrarrestar los fuertes vientos y posibles movimientos sísmicos.[7]
- En puentes colgantes para contrarrestar las fuerzas del viento y movimiento telúricos [7]
- Grúas de demolición [7].

Aplicaciones del péndulo doble:

- Sistemas con comportamiento caótico; Atractores de Lorentz, fractales, circuitos RLC, entre otros [8].



- Simulación de sismos [9].
- Modelos utilizados en economía, biología, meteorología, etc [9].

Aplicaciones del péndulo invertido:

En [10] se ejemplifican diversas aplicaciones en las siguientes áreas:

- Aeroespacial: Se requiere el control activo de un cohete para mantenerlo en la posición vertical invertida durante su despegue.
- Biomecánica: El péndulo invertido es frecuentemente utilizado para modelar bípedos caminantes, tal como el robot humanoide.
- Transporte: Segway Human Transporter, cuyo control está basado en entradas sensoriales de giroscopios montados en la base del mismo.

Por sí solos, estos sistemas son mecanismos, que, al agregar la parte de control, se convierten en sistemas más completos que tienen diversas aplicaciones en la industria, estos simples, pero muy representativos sistemas, al embeberse en sistemas más completos, conducen a excelentes resultados.

Para la codificación se seleccionó un lenguaje de alto nivel para desarrollar el proceso de sintonización. A continuación, se presentan algunas ventajas que presentan los lenguajes de alto nivel [11]:

Ventajas:

- Son más fáciles de leer, escribir y mantener por humanos.
- Se da forma a un código común que se podrá utilizar en diversos sistemas operativos y plataformas.
- Se pueden usar diferentes paradigmas de programación.
- Posibilidad de usar lenguajes interpretados.



0.3. Objetivo

Implementar mediante simulación tres métodos de sintonización para un controlador PID basados en optimización metaheurística multiobjetivo: Evolución Diferencial, Algoritmo Genético y Optimización por Enjambre de Partículas, (DE, GA Y PSO respectivamente, por sus siglas en inglés) en tres sistemas mecatrónicos comunes, mediante un estudio comparativo, con base en evidencia estadística, del comportamiento las técnicas metaheurísticas multiobjetivo ya mencionadas, y la aplicación física del algoritmo con mejor desempeño en un péndulo simple.

Objetivos específicos

Objetivos TT1

- Obtener los modelos en espacio de estados de los sistemas: péndulo simple, péndulo doble y péndulo invertido.
- Seleccionar del método numérico para simular los modelos en espacio de estados de los sistemas péndulo simple, péndulo doble y péndulo invertido.
- Seleccionar una estructura para el controlador PID, PD o PI.
- Determinar las variables de diseño del problema de optimización.
- Determinar las funciones objetivo del problema de optimización.
- Determinar las restricciones funcionales del problema de optimización.
- Determinar las restricciones de caja del problema de optimización.
- Plantear el problema de sintonización como un problema multiobjetivo.
- Determinar tres algoritmos metaheurísticos para sintonizar el controlador.



- Seleccionar técnicas estadísticas no paramétricas para evaluar el desempeño de los algoritmos metaheurísticos.
- Diseño del péndulo simple, circuito de control y acoplamiento.

Objetivos TT2

- Implementar la simulación de los modelos en espacio de estados del péndulo simple, péndulo doble y péndulo invertido.
- Programar los algoritmos metaheurísticos para sintonizar el controlador.
- Resolver numéricamente los problemas de optimización para las variables de diseño.
- Obtener las aproximaciones de los frentes de Pareto correspondientes a cada una de las nueve sintonizaciones del controlador.
- Evaluar el desempeño de los algoritmos metaheurísticos sobre las aproximaciones de los frentes de Pareto.
- Implementación de una interfaz de usuario que permita la visualización del comportamiento de los diferentes algoritmos aplicados al controlador en un péndulo simple.
- Implementar el control en un péndulo simple.
- Implementar las ganancias de control que proporcionen el mejor compromiso entre los objetivos en un péndulo simple.

0.4. Antecedentes



Año	Nombre	Descripción	Objetivos	Controlador y variables de decisión	Restricciones	Referencia
2018	Tuning of Fractional Order PID Controllers using Evolutionary Optimization for PID Tuned Synchronous Generator Excitation System	Se utiliza un NSGA-II para resolver el problema de contradicción entre minimizar el error, escalamiento de la estabilidad robusta y la minimización del consumo de energía encuadrando controladores IO PID- FO PID como un problema multiobjetivo.	Integral Squared Error (ISE), Maximum Sensitivity (MS).	PID+FOPID: ganancias $k_p^j, k_i^j, k_d^j, j = 1, 2$, <i>integro – differential orders,</i>	Condiciones de estabilidad de Routh.	[12]
2018	Control of refrigeration systems based on vapour compression using multi-objective optimization techniques.	Se hace uso de un algoritmo basado en DE para sintonizar las ganancias requeridas para un control de tipo PID usado en un sistema de refrigeración basado en compresión de vapor.	IAE (dos salidas), Total variation of the control signal (TV) (dos entradas), Log-Modulus (LM).	PID+PI: ganancias $k_p^j, k_i^j, j = 1, 2, k_d, filtrof.$	-	[13]
2014	Design of a fractional order pid controller for hydraulic turbine regulating system using chaotic nondominated sorting genetic algorithm ii.	Se diseña un controlador PID fraccionario (FOPID, por sus siglas en inglés) apoyándose de un NSGA-II para un sistema regulador de una turbina hidráulica (HTRS). Las funciones objetivo son la integral del error cuadrático (ISE) y la integral del error cuadrático del tiempo multiplicado (ITSE), constituyendo de esta forma un problema de optimización multiobjetivo.	ISE e ITSE.	PID: ganancias k_p, k_i, k_d	-	[14]
2014	A novel design method of multiobjective robust pid controller for industrial process.	Se presenta un control PID robusto multiobjetivo, para un proceso no lineal en un reactor de tanque agitado continuo (CSTR, por sus siglas en inglés). Se busca que el control se robusto, limitado y un buen rendimiento de salida. Para esto, se emplea un algoritmo evolutivo; Optimización por Enjambre de Partículas Multiobjetivo (MOPSO, por sus siglas en inglés).	ISE, IAE, MS.	PID: parámetros $, k_c, T_i, T_d, T_f$	Control máximo	[15]

Año	Nombre	Descripción	Objetivos	Controlador y variables de decisión	Restricciones	Referencia
2014	Design and performance analysis of pid controller for an avr system using multi-objective non-dominated shorting genetic algorithm-ii.	Se presenta el diseño y funcionamiento de un controlador PID para un sistema AVR usando NSGA-II , abordando el problema como uno de optimización multiobjetivo.	IAE, ITAE, ISE, ITSE.	PID: ganancias k_p, k_i, k_d	-	[16]
2011	Multi-objective pid controller tuning for a facts-based damping stabilizer using non-dominated sorting genetic algorithm-II.	Se estudia la aplicación de NSGA-II para la sintonización de un controlador PID para un Sistema Flexible de transmisión de CA (FACTS). El objetivo del diseño es mejorar la amortiguación del sistema de potencia cuando se somete a una perturbación con un esfuerzo de control mínimo.	SE index (para errores y señales de control).	PID: ganancias k_p, k_i, k_d	-	[17]
2015	Application of multiobjective controller to optimal tuning of pid gains for a hydraulic turbine regulating system using adaptive grid particle swam optimization.	Se emplea un algoritmo evolutivo multiobjetivo (AGPSO), para determinar las ganancias requeridas para sintonizar un HTRS. Se busca cuidar el tiempo de asentamiento y el nivel de sobreimpulso simultáneamente.	ISE, ITSE.	PID: ganancias k_p, k_i, k_d	-	[18]
2017	Multi-objective on-line optimization approach for the DC motor controller tuning using differential evolution.	Se propone la sintonización en línea de carácter multiobjetivo, usando algoritmos metaheurísticos, para poder ajustar los parámetros de control en la velocidad de un motor de imanes permanentes.	ISE, ISE.	Controlador por dinámica inversa: Parámetros dinámicos del motor CD.	Límites de la señal de control.	[19]
2008	A novel intelligent multiobjective simulated annealing algorithm for designing robust PID controllers.	Se hace uso de un algoritmo multiobjetivo para satisfacer las siguientes necesidades en un controlador PID: atenuación de perturbaciones, estabilidad robusta y seguimiento preciso del “setpoint”.	MS, MCS, ISE.	PI: ganancias k_p, k_i	Límites superiores de MS y MCS.	[20]
2019	Adaptive controller tuning method based on online multiobjective optimization: A case study of the four-bar mechanism.	Se busca la sintonización de un controlador adaptativo basado en optimización metaheurística multiobjetivo para regular la velocidad en un mecanismo de cuatro barras.	ISE, suavidad de la señal de control.	Dinámica inversa del controlador: parámetros dinámicos de un mecanismo de 4 barras.	-	[21]

Marco Referencial

1.1. Marco Histórico

Control Automático

El desarrollo de la humanidad puede reflejarse en sus avances tecnológicos y científicos. Entre los principales descubrimientos o inventos que han revolucionado la forma de vida del hombre, nos encontramos con la invención de la rueda, la imprenta, el desarrollo de la máquina de vapor -cuya invención impulsó la Revolución Industrial-, la electrónica y sus aplicaciones en la informática y la aparición de la red de redes, el Internet, como uno de los hitos con mayor impacto a nivel global.

El control automático representa una función vital en el avance de la ingeniería y la ciencia. Como primer trabajo importante en control automático, surge a finales del siglo XVIII el regulador de velocidad centrífugo de James Watt para el control de velocidad de una máquina de vapor [22]. Cuando la máquina de vapor fue inventada y se dió inicio a la Revolución Industrial, paralelamente surgió la necesidad de crear un sistema de control para poder manipular los diferentes parámetros de esta máquina. Un ejemplo de esto fue el desarrollo del regulador de presión con el fin de controlar este parámetro en el sistema. Posterior a la Revolución Industrial,



el diseño de sistemas de control realimentado estaba desarrollándose por medio de prueba y error junto con mucha intuición de la ingeniería [23]. En las matemáticas de mediados del siglo XIX, primero fue usado el análisis de la estabilidad de sistemas de control realimentado. Iba comenzando el siglo XX, cuando surgió la necesidad industrial de desarrollar instrumentos capaces de medir y controlar presiones, temperaturas y otras variables. Con el paso del tiempo, se dieron grandes avances en conceptos y procesos de control, entre los cuales podemos destacar: Minorsky mostró en 1922 que la estabilidad de un sistema puede determinarse a partir de sus ecuaciones diferenciales, al mismo tiempo, se dio a conocer la importancia de un controlador PID. La función proporcional ya se conocía desde el comienzo del relé, sin embargo, la función integral no se conoció hasta 1920 y la derivativa hasta 1930. Una década después, Nyquist diseñó un procedimiento simple en 1932 para determinar la estabilidad en sistemas de lazo cerrado. 1940 fue un gran año para el mundo del control automático, ya que G Ziegler y N.B. Nichols ponen en el mercado el primer controlador PID: Fulslope modelo 100 [24]. Fue durante las décadas de 1960 y 1970, que se investigaron a profundidad tópicos sobre control óptimo, tanto de sistemas determinísticos como estocásticos, como de control adaptable, mediante el aprendizaje de sistemas complejos.

Optimización

La optimización de problemas ha sido objeto de estudio desde el pasado y hasta nuestros días, ya que es un área activa de investigación. Debido a que la complejidad de los problemas de mundo real cada vez parece subir de nivel, ya sea en el campo de la ciencia o la ingeniería, es necesario contar con heurísticas capaces de resolver de forma rápida y eficiente estos problemas. En las últimas décadas, los algoritmos bio-inspirados han llamado la atención de los investigadores y han estado presentes en numerosas publicaciones. Estos métodos son estocásticos y funcionan bien optimizando problemas en donde se requiere minimizar/maximizar una función, sin



embargo, debido a que de forma global se busca una solución buena en un tiempo razonable, estos métodos no solucionan por sí solos el problema, pues únicamente se está optimizando un solo objetivo. Entonces, aquí es donde entra la optimización multiobjetivo basada en algoritmos evolutivos (MOEAs, por sus siglas en inglés) como una alternativa eficaz para optimizar problemas con más de un objetivo, garantizando soluciones óptimas o semi-óptimas en un tiempo razonable [25].

1.2. Marco teórico

Controladores automáticos

Los controladores automáticos comparan el valor real de la salida de una planta con la entrada de referencia (el valor deseado), determinan la desviación y producen una señal de control que reduce la desviación a cero o a un valor pequeño. La manera en la cual el controlador automático produce la señal de control se denomina acción de control. El controlador detecta la señal de error, que, por lo general, está en un nivel de potencia muy bajo, y la amplifica a un nivel lo suficientemente alto. La salida de un controlador automático alimenta a un actuador, como un motor, una válvula neumática, un motor hidráulico o un motor eléctrico. El sensor, o elemento de medición, es un dispositivo que convierte la variable de salida en otra variable manejable, como un desplazamiento, una presión o un voltaje, que pueda usarse para comparar la salida con la señal de entrada de referencia. En la figura 1.1, se muestra el diagrama de bloques de un sistema de control industrial [26].

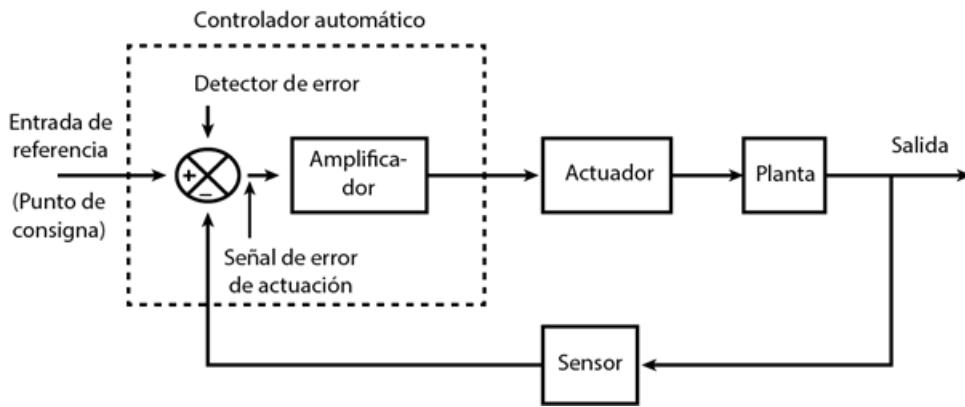


Figura 1.1: Diagrama de bloques de un sistema de control industrial Fuente: Adaptado de

Clasificación de los controladores industriales.

Los controladores industriales se clasifican, de acuerdo con sus acciones de control, como:

1. De dos posiciones o controladores on-off.
2. Controladores proporcionales.
3. Controladores integrales.
4. Controladores proporcionales, integrales, derivativos y sus variantes [26].

Este último tipo de controladores industriales se utilizan ampliamente en robots manipuladores industriales. En ausencia del conocimiento de la dinámica del robot, el control PID puede ser una buena elección para el seguimiento de trayectoria, ya que el controlador es libre de modelo, y sus parámetros se pueden ajustar por separado [27] [28]. Como se menciona en [29], las ventajas del controlador PID sobre los otros es que estos son simples y tienen significados físicos muy claros. Sin embargo, las ganancias del controlador PID deben ser sintonizadas para garantizar un buen



desempeño, así como sobreimpulso, tiempo de establecimiento y el error de estado estacionario adecuado. A pesar de no ser algo nuevo, este controlador es ampliamente usado en la industria, debido a su facilidad de implementación.

Algunos métodos de sintonización son los que se pueden consultar en [29]:

1. Modelo basado en sintonización analítica.
2. Métodos heurísticos.
3. Métodos en dominio de la frecuencia.
4. Métodos de optimización.
5. Métodos Adaptables.

En [30], se describen a los modelos heurísticos, como modelos inteligentes, basados en la experiencia, en este trabajo se propone tratar el problema de sintonización con metaheurística, lo que permite plantear el problema de sintonización como un problema de formal de optimización.

Acción de control proporcional

El controlador proporcional, es, en esencia, un amplificador con una ganancia ajustable.

Para un controlador con acción proporcional, la relación entre la salida del controlador $u(t)$ y la señal de error $e(t)$ es [26]:

$$u(t) = K_p e(t)$$

Acción de control integral

Es un controlador con acción de control integral, el valor de salida del controlador $u(t)$ se cambia a una razón proporcional a la señal de error $e(t)$ [26]. Es decir:

$$\frac{du(t)}{dt} = K_i e(t).$$



o bien:

$$u(t) = K_i \int_0^t e(t) dt$$

Acción de control proporcional-integral

La acción de control de un controlador proporcional-integral (PI) se define mediante [26]:

$$u(t) = K_p e(t) + K_i \int_0^t e(t) dt$$

Acción de control proporcional-derivativa

La acción de control de un controlador proporcional-derivativa (PD) se define mediante [26]:

$$u(t) = K_p e(t) + K_d \frac{de(t)}{dt}$$

Acción de control proporcional-integral-derivativa

La combinación de la acción proporcional, la acción integral, y la acción derivativa se denomina acción de control proporcional-integral-derivativa. Esta acción combinada tiene las ventajas de cada una de las tres acciones individuales.

$$u(t) = K_p e(t) + K_i \int_0^t e(t) dt + K_d \frac{de(t)}{dt}$$

O bien:

$$u(t) = K_p \left[e(t) + \frac{1}{T_i} \int_0^t e(t) dt + T_d \frac{de(t)}{dt} \right]$$

donde K_p es la ganancia proporcional, T_i es el tiempo integral y T_d es el tiempo derivativo [26].



Métodos de integración

Para resolver una ecuación diferencial, podemos apoyarnos de una gama de algoritmos cuyo propósito es calcular el valor numérico de una integral definida, a esta variedad de algoritmos se le conoce como integración numérica. Los métodos de integración numérica generalmente son descritos como una combinación de evaluaciones del integrando para obtener una aproximación a la integral [30]. Durante el análisis de cualquier método de integración numérica, estudiar el comportamiento del error de aproximación como una función del número de evaluaciones del integrando es fundamental. Como se ha mencionado antes, existen diversos métodos de integración, sin embargo el método de Euler destaca por ser uno de los más simples para resolver ecuaciones diferenciales ordinarias a partir de un valor inicial dado. Fue propuesto por Leonhard Euler en 1768. Consiste básicamente en aproximar la derivada por el cociente incremental en la ecuación mencionada:

Consiste en dividir el intervalo $t_0, t_0 + \alpha$ en N partes iguales:

$$t_1 = t_0 + h, t_2 = t_0 + 2h, \dots, t_N = t_0 + Nh = t_0 + \alpha, h = \frac{\alpha}{N}$$

Aplicando la definición de la derivada:

$$y'(t_k) = \lim_{h \rightarrow \infty} \frac{y(t_k + h) - y(t_k)}{h},$$

Deducimos que para h "suficientemente pequeño":

$$y'(t_k) = f(t_k, y(t_k)) \simeq \frac{y(t_k + h) - y(t_k)}{h}$$

Por tanto:

$$y(t_{k+1}) \simeq y(t_k) + hf(t_k, y(t_k)), k = 0, 1, \dots, N - 1,$$

partiendo de $y(0) = y_0$. La ecuación anterior se conoce como método de Euler [31]. Este método resulta adecuado para desarrollar este trabajo.

Dinámica

Ecuaciones de Euler-Lagrange

El lagrangiano $L(\mathbf{q}, \dot{\mathbf{q}})$ de n grados de libertad se define como la diferencia de la energía cinética $K(\mathbf{q}, \dot{\mathbf{q}})$ y la energía potencial $U(\mathbf{q})$.

$$L(\mathbf{q}, \dot{\mathbf{q}}) = K(\mathbf{q}, \dot{\mathbf{q}}) - U(\mathbf{q})$$

Las ecuaciones de movimiento de Euler-Lagrange de n grados de libertad están dadas por:

$$\frac{d}{dt} \left[\frac{\partial L(\mathbf{q}, \dot{\mathbf{q}})}{\partial \dot{\mathbf{q}}} \right] - \left[\frac{\partial L(\mathbf{q}, \dot{\mathbf{q}})}{\partial \mathbf{q}} \right] = \boldsymbol{\tau} - \mathbf{D}(\dot{\mathbf{q}}, \mathbf{f}_e)$$

donde $\mathbf{q} = [q_1, q_2, \dots, q_n]^T \in \mathbb{R}^n$ representa el vector de posiciones articuladas o coordenadas generalizadas, $\dot{\mathbf{q}} = [\dot{q}_1, \dot{q}_2, \dots, \dot{q}_n]^T \in \mathbb{R}^n$ es el vector de velocidades articuladas, $\boldsymbol{\tau} = [\tau_1, \tau_2, \dots, \tau_n]^T \in \mathbb{R}^n$ es el vector de pares aplicados, donde el i -ésimo par τ_i se encuentra asociado con la i -ésima coordenada generalizada q_i , y $\mathbf{D}(\dot{\mathbf{q}}, \mathbf{f}_e)$ es el vector de fuerzas o pares de fricción que depende de la velocidad articular $\dot{\mathbf{q}}$ y de la fricción estática \mathbf{f}_e que se encuentra presente en las articulaciones del mismo; $t \in \mathbb{R}_+$ representa el tiempo, $n \in N$ el número de grados de libertad [32].

Modelo dinámico

El modelo dinámico de un robot manipulador de n grados de libertad, que en su forma compacta y con la notación más ampliamente utilizada en el área de robótica esta dado por la siguiente ecuación [32]:

$$\boldsymbol{\tau} = M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \mathbf{f}_f(\dot{\mathbf{q}}, \mathbf{f}_e)$$

O alternativamente:

$$\begin{bmatrix} \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{q}} \\ M^{-1}(\mathbf{q})[\boldsymbol{\tau} - C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{g}(\mathbf{q}) - \mathbf{f}_f(\dot{\mathbf{q}}, \mathbf{f}_e)] \end{bmatrix}$$



Aquí se tiene que:

$\mathbf{q} \in \mathbb{R}^n$ es el vector de coordenadas generalizadas o posiciones articulares,

$\dot{\mathbf{q}} \in \mathbb{R}^n$ es el vector de velocidades articuladas,

$\ddot{\mathbf{q}} \in \mathbb{R}^n$ es el vector de aceleraciones articuladas,

$M(\mathbf{q}) \in \mathbb{R}^{nxn}$, es la matriz de inercia, la cual es simétrica, y definida positiva,

$C(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{nxn}$ es la matriz de fuerzas centrípetas y de Coriolis,

$$C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} = \dot{M}(\mathbf{q})\dot{\mathbf{q}} - \frac{\partial}{\partial \mathbf{q}} \left[\frac{1}{2} \dot{\mathbf{q}}^T M(\mathbf{q}) \dot{\mathbf{q}} \right]$$

$\mathbf{g}(\mathbf{q}) \in \mathbb{R}^n$ es el vector fuerzas o pares gravitacionales obtenido como el gradiente de la energía potencial, es decir:

$$\mathbf{g}(\mathbf{q}) = \frac{\partial \mathbf{U}(\mathbf{q})}{\partial \mathbf{q}}$$

debida a la acción de la gravedad,

$\mathbf{f}_f(\dot{\mathbf{q}}, \mathbf{f}_e) \in \mathbb{R}^n$ es el vector de pares de fricción viscosa, Coulomb, y estática (\mathbf{f}_e) de cada articulación del robot [32].

Cinemática

La cinemática es la parte de la física que aborda el problema de la descripción numérica del movimiento de sistemas mecánicos sin tomar en cuenta las fuerzas que lo producen.

Cinemática directa: Es una función vectorial que relaciona las coordenadas articulares $\mathbf{q} \in \mathbb{R}^n$ con las coordenadas cartesianas $[x, y, z]^T \in \mathbb{R}^3$ del robot $\mathbf{f}_R : \mathbb{R}^n \rightarrow \mathbb{R}^m$, así como la orientación $[\theta, \phi, \psi]^T \in \mathbb{R}^3$ de la herramienta colocada en el extremo final, tomando en cuenta las propiedades geométricas del sistema mecánico del robot. En esta definición f_R es una función continua en el vector de posiciones o desplazamiento articular $\mathbf{q} \in \mathbb{R}^n$, representan el número de grados de libertad y la dimensión del vector de posiciones o desplazamiento articular, $x, y, z \in \mathbb{R}$ son las



coordenadas cartesianas asociadas al extremo final del robot y $\tau, \phi, \psi \in \mathbb{R}$ son los ángulos de Euler, que representan la orientación de la herramienta colocada en el extremo final con respecto al sistema de referencias fijo en la base del robot, m es la dimensión de la función vectorial $\mathbf{f}_R(\mathbf{q}) = [x, y, z, \tau, \phi, \psi] \in \mathbb{R}^6$, para el caso general $m = 6$. Cuando $n > m$ se denominan robots redundantes [32].

Cinemática inversa: Es un problema no lineal que relaciona las coordenadas articulares en función de las coordenadas cartesianas. Dada la posición cartesiana y la orientación de la herramienta colocada en el extremo final del robot, obtener los ángulos de las articulaciones [32].

$$\mathbf{q} = \mathbf{f}_R^{-1}(x, y, z, \tau, \phi, \psi)$$

Cinemática diferencial

La cinemática diferencial directa es la derivada con respecto al tiempo de la cinemática directa:

$$\frac{d}{dt}[x \ y \ z \ \tau \ \phi \ \psi]^T = \begin{bmatrix} v \\ \omega \end{bmatrix} = \frac{d}{dt}\mathbf{f}_R(\mathbf{q}) = \frac{\partial \mathbf{f}_R(\mathbf{q})}{\partial \mathbf{q}} \dot{\mathbf{q}} = J(\mathbf{q})\dot{\mathbf{q}}$$

Como se ve, esta relaciona la velocidad articular $\dot{\mathbf{q}} \in \mathbb{R}^n$ con la velocidad lineal y la velocidad angular, además el mapeo descrito por una matriz $J(\mathbf{q}) = \frac{\partial \mathbf{f}_R(\mathbf{q})}{\partial \mathbf{q}} \in \mathbb{R}^{6xn}$ denominada jacobiano analítico:

$$J(\mathbf{q}) = \begin{bmatrix} J_v(\mathbf{q}) \\ J_\omega(\mathbf{q}) \end{bmatrix}$$

La cinemática diferencial inversa representa la relación entre la velocidad articular $\dot{\mathbf{q}}$ con la velocidad lineal de movimiento v y la velocidad angular ω , expresada en términos de la matriz inversa del jacobiano:

$$\dot{\mathbf{q}} = J^{-1}(\mathbf{q}) \begin{bmatrix} v \\ \omega \end{bmatrix}$$



donde $J^{-1}(\mathbf{q}) \in \mathbf{R}^{6xn}$ es la matriz inversa del jacobiano, la cual existe si es una matriz cuadrada y su determinante es diferente de cero. Si el determinante del jacobiano $J(\mathbf{q})$ es cero, entonces se dice que no es rango completo y se presentan problemas de singularidades [32].

Regulación de posición

La tarea de regular posición consiste en hacer que el sistema se mueva desde una condición inicial de posición hasta una posición deseada final [32].

Formalmente, el objetivo del problema de control de posición está determinado por encontrar una ley de control τ que proporcione los pares aplicados a los actuadores del sistema, de tal forma que la posición actual $\mathbf{q}(t)$ y la velocidad articular de movimiento $\dot{\mathbf{q}}(t)$ tiendan asintóticamente hacia la posición deseada \mathbf{q}_d y velocidad cero, respectivamente, sin importar las condiciones iniciales [32].

Perfiles de velocidad

Los perfiles de velocidad sirven para trazar la velocidad que deberá tomar un motor en un determinado tiempo, debido a la naturaleza de este trabajo, este tiempo será establecido por las necesidades del usuario. Los perfiles básicos son el triangular y el trapezoidal, los cuales describen a continuación.

Perfil de velocidad triangular: Este tipo de perfil tiene como característica que aumenta su velocidad con una aceleración constante en un determinado tiempo y en el momento en el que alcanza su velocidad máxima (v_m), comienza un decremento de velocidad igualmente con aceleración constante hasta llegar a una velocidad de 0 [33] , tal como se observa en la figura 1.2.

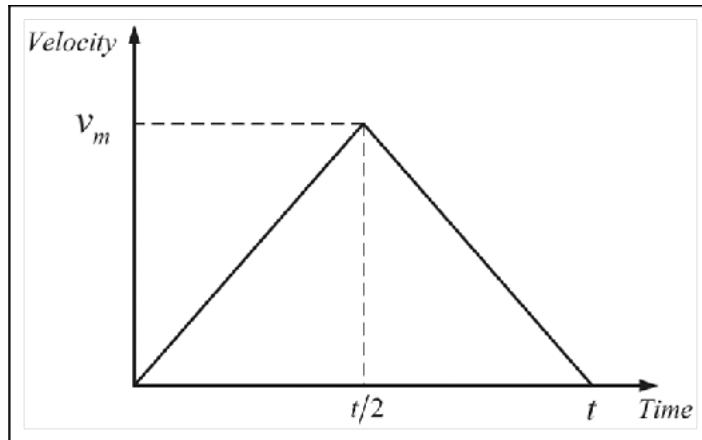


Figura 1.2: Perfil de velocidad triangular

Perfil de velocidad trapezoidal: El modelo del perfil trapezoidal es aquel en donde un motor es acelerado con una aceleración constante y cuando llega a una velocidad máxima (v_m), se mantiene constante por un tiempo, al terminar ese tiempo, es desacelerado a cero con una desaceleración constante, como se observa en la Figura 1.3.

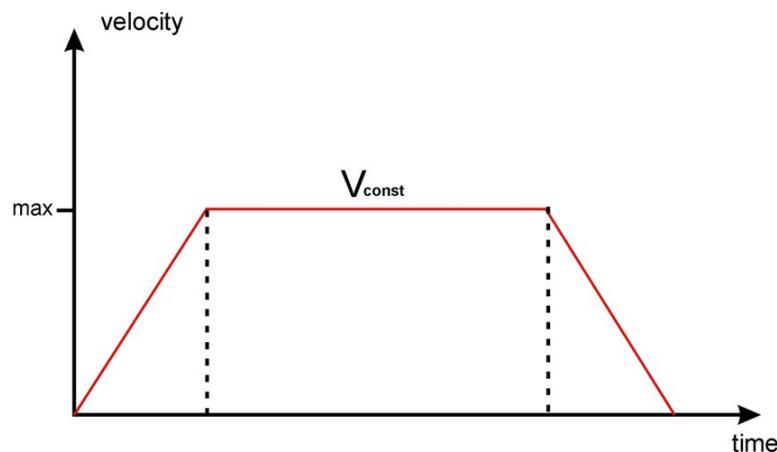


Figura 1.3: Perfil de velocidad trapezoidal



Control de trayectoria

El control de trayectoria radica en determinar una función vectorial τ de tal manera que las posiciones y las velocidades asociadas a las articulaciones del robot sigan con exactitud a las posiciones y velocidades deseadas, respectivamente.

El objetivo de control consiste en encontrar τ tal que:

$$\lim_{t \rightarrow \infty} \begin{bmatrix} \tilde{\mathbf{q}} \\ \dot{\tilde{\mathbf{q}}} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \in \mathbb{R}^{2n}$$

donde $\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}} \in \mathbb{R}^n$ representa el error de posición definido entre la posición o trayectoria deseada $\mathbf{q}_d(t)$ y la posición actual del robot $\mathbf{q}(t)$, es decir $\tilde{\mathbf{q}} = \mathbf{q}_d(t) - \mathbf{q}(t)$ y error de velocidad definido como $\dot{\tilde{\mathbf{q}}} = \dot{\mathbf{q}}_d(t) - \dot{\mathbf{q}}(t)$, es decir entre la velocidad deseada de movimiento $\dot{\mathbf{q}}_d(t)$ y la velocidad articular $\dot{\mathbf{q}}(t)$ [32].

1.3. Marco conceptual

Diseño planteado como problema de optimización

Sin pérdida de generalidad, un problema de diseño puede plantearse como el siguiente problema de optimización.

Diseño planteado:

$$\mathbf{J}(\mathbf{x}) = [J_1(x), \dots, J_m(x)] \dots (1.4)$$

Sujeto a:

$$g(x) \leq 0 \dots (1.5)$$

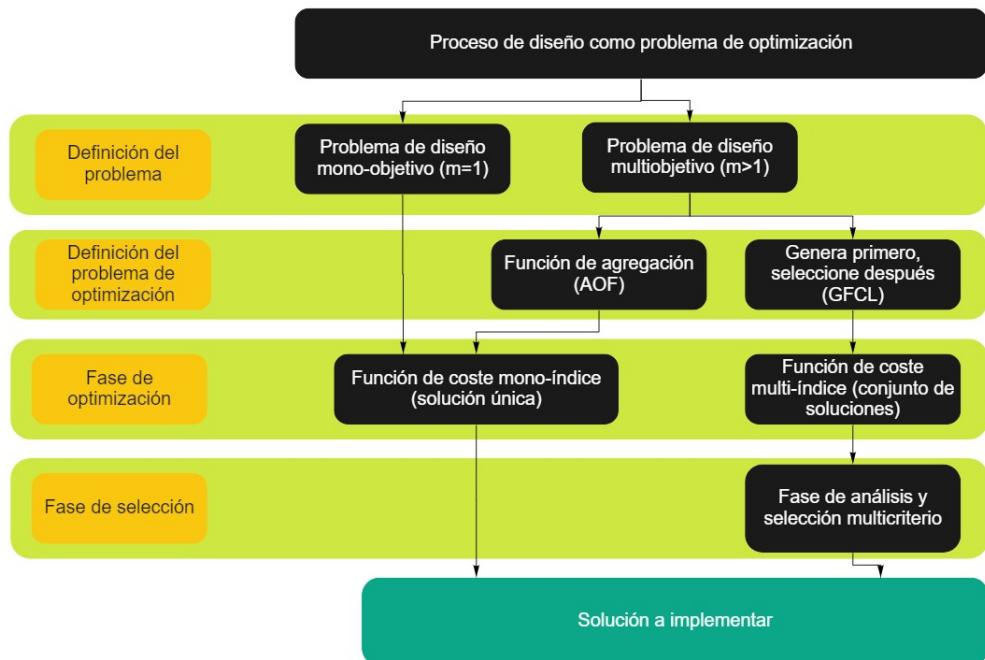


$$h(x) = 0 \dots(1.6)$$

$$\underline{x}_i \leq x_i \leq \bar{x}_i, i = [1, \dots n] \dots(1.7)$$

Donde $x \in R^n$ es el vector de variables de decisión; $J(x) \in m$, es el vector de objetivos de diseño; $g(x), h(x)$ el vector de restricciones de desigualdad e igualdad respectivamente, y $\underline{x}_i \leq x_i \leq \bar{x}_i$ cota de búsqueda del espacio de decisión para la variable x_i [34].

En la Figura 1.4 se muestra una metodología general para resolver un problema de diseño como un problema de optimización.



miro

Figura 1.4: Metodología de diseño por medio de la optimización

A continuación se aborda cada una de las etapas mostradas.



Definición del problema

Implica identificar las variables de decisión, los objetivos de diseño y las restricciones (si las hay) del problema. Si $m = 1$ se habla de un problema mono-objetivo (SOP por sus siglas en inglés) mientras que si $m > 1$ se habla de un problema multi-objetivo (MOP por sus siglas en inglés), donde se busca una solución de compromiso entre los objetivos de diseño.

Definición del problema de optimización

Se refiere a la forma en que se traduce el problema anterior en una expresión que el optimizador pueda interpretar. El concepto de optimización de sistemas de control abarca dos etapas, una de selección de índices de rendimiento y otra de diseño en base a la minimización o maximización de dichos índices. En el caso de un SOP, el índice a optimizar será el mismo (y a la vez el único) objetivo de diseño. En el caso de un MOP, su resolución se puede abordar desde dos perspectivas [35]: empleando funciones de agregación (Aggregate Objective Function, AOF) o calculando un conjunto de soluciones para seleccionar después la más indicada atendiendo a las preferencias del diseñador (Generate First, Choose Later, GFCL). En el caso de técnicas MOP/AOF, se busca especificar todos los requerimientos de diseño y las preferencias del diseñador en un solo índice. Con el planteamiento anterior el optimizador calcula una solución única para implementar. Por otra parte, la MOP/GFCL es una estrategia que busca determinar un conjunto de soluciones potenciales, para que el diseñador, a partir de ellas, elija la más indicada de acuerdo a sus preferencias de diseño. Lo anterior se justifica con el hecho de que, cuando se consideran simultáneamente todos los objetivos, generalmente no hay una solución mejor que las otras, pues no existe una solución única que mejore a todas las demás en todos los objetivos.



Fase de optimización

La fase de optimización se refiere al uso/ejecución del algoritmo de optimización; los algoritmos evolutivos (EA's por sus siglas en inglés) se han utilizado como una alternativa válida para resolver estos planteamientos en diversos campos de la ingeniería [36]. Los casos SOP y MOP/AOF derivan en la optimización de una función de coste mono-índice, y se espera en la mayoría de los casos una solución única. En el caso SOP, se espera que el único objetivo de diseño sea suficiente para encontrar una solución práctica para el diseñador. En el caso de MOP/AOF, se espera que el intercambio de prestaciones deseado entre objetivos de diseño sea adecuadamente reflejado en el diseño de la función de coste mono-índice. En el caso de MOP/GFCL, se busca (por lo general) un conjunto de soluciones que aproxime lo que se conoce como conjunto de Pareto \mathcal{P}^* , y su imagen en el espacio de objetivos, frente de Pareto \mathcal{PF}^* .

Fase de selección

La etapa de selección de la solución es exclusiva de la rama MOP/GFCL. En los casos SOP y MOP/AOF, no existe esta etapa, pues la solución que ofrece el optimizador supone una solución con las prestaciones deseadas que han sido definidas *a priori*. En el caso MOP/GFCL, la búsqueda de una solución con las prestaciones deseadas se efectúa (en general) *a posteriori*, analizando el conjunto de soluciones y el intercambio de sus prestaciones [34].

Métodos metaheurísticos

Un método heurístico es un procedimiento que trata de descubrir una solución factible muy buena, pero no necesariamente una solución óptima, para el problema específico bajo consideración. No puede darse una garantía acerca de la calidad de la solución que se obtiene, pero un método heurístico bien diseñado puede proporcionar



una solución que al menos está cerca de ser óptima (o concluir que no existen tales soluciones). El procedimiento también debe ser suficientemente eficiente como para manejar problemas muy grandes. Con frecuencia, el procedimiento es un algoritmo iterativo novedoso, donde cada iteración implica la realización de una búsqueda de una nueva solución que puede ser mejor que la solución que se encontró con anterioridad. Cuando el algoritmo termina después de un tiempo razonable, la solución que proporciona es la mejor que se pudo encontrar en cualquier iteración [37]. Por su parte, las metaheurísticas son una familia de algoritmos aproximados de propósito general. Suelen ser procedimientos iterativos que guían una heurística subordinada de búsqueda, combinando de forma inteligente distintos conceptos para explorar y explotar adecuadamente el espacio de búsqueda [38].

Clasificación de los métodos metaheurísticos

En la actualidad se conocen varias metaheurísticas que se han desarrollado exitosamente en la resolución de determinados problemas, tales como: los algoritmos voraces, la ascensión de colinas, el recocido simulado, colonias de hormigas, algoritmos de enjambre, la búsqueda tabú, los algoritmos genéticos, los algoritmos meméticos, entre otros. No obstante, aunque suele ser una tarea poco sencilla, algunos autores han recopilado algunas formas en las que se pudieran clasificar las metaheurísticas. A continuación se presenta una posible forma de clasificación de las metaheurísticas [38]:

1. *Basadas en métodos constructivos:* Parten de una solución inicial vacía y se van agregando componentes hasta construir una solución. En este grupo se pueden mencionar: GRASP, optimización basada en colonias de hormigas.
2. *Basadas en trayectorias:* Utilizan como heurística subordinada cualquier algoritmo de búsqueda local, que sigue una trayectoria en el espacio de búsqueda, mediante iteraciones que tratan de remplazar una solución inicial por otra de



mejor calidad. Allí se encuentran: búsqueda local, templado simulado, búsqueda tabú, búsqueda local iterativa, entre otras.

3. *Basadas en poblaciones:* El proceso contempla múltiples puntos de búsqueda en el espacio, que evolucionan en paralelo. Dentro de ellas están: algoritmos genéticos, algoritmos meméticos, algoritmos basados en nubes de partículas, búsqueda dispersa, entre otros.

Algoritmos Evolutivos

Dentro de los métodos metaheurísticos basados en poblaciones, se encuentran los algoritmos evolutivos, también conocidos como optimizadores metaheurísticos. Estos son métodos estocásticos que están basados o bien en las leyes de la selección natural o inspirados por el comportamiento de ciertas especies.

Los optimizadores metaheurísticos se han usado ampliamente en problemas de optimización, debido a operación relativamente simple, su capacidad para manejar problemas complejos a un coste computacional razonable, así como su flexibilidad en variedad de contextos [39]. Además, son preferentemente usados en optimización multiobjetivo debido a que su capacidad para obtener múltiples soluciones en el frente de Pareto en una sola ejecución completa del programa, así como aprovechar similitudes de soluciones mediante recombinación. El conjunto de soluciones no dominadas es llamado aproximación al conjunto de Pareto \mathcal{P}^A , estas soluciones en el espacio de objetivos forman la aproximación al frente de Pareto \mathcal{PF}^A . Las soluciones óptimas, que en la vida real son desconocidas, forman el conjunto de Pareto \mathcal{P}^* y en el espacio de objetivos forman el verdadero frente de Pareto \mathcal{PF}^* [34]. Es importante recalcar que los optimizadores metaheurísticos no garantizan encontrar el verdadero frente de Pareto, sino la aproximación al Frente de Pareto a cambio de un coste computacional razonable [39]. Todos estos enfoques operan en un conjunto de soluciones candidatas. Usando fuertes simplificaciones, este conjunto es posteriormente modificado por los dos principios básicos de evolución: selección y variación.



La selección representa la competencia por los recursos entre los seres vivos. Algunos son mejores que otros y es más probable que sobrevivir y reproducir su información genética. En algoritmos evolutivos, la selección natural se simula mediante un proceso de selección estocástica. Cada solución tiene la oportunidad de reproducirse un cierto número de veces, dependiendo de su calidad. De este modo, la calidad se evalúa evaluando a las personas y asignando ellos valores de aptitud escalar. El otro principio, la variación, imita la capacidad natural de crear "nuevos" seres vivos mediante recombinación y mutación. En la Figura 1.5 se puede observar de manera gráfica el proceso descrito.

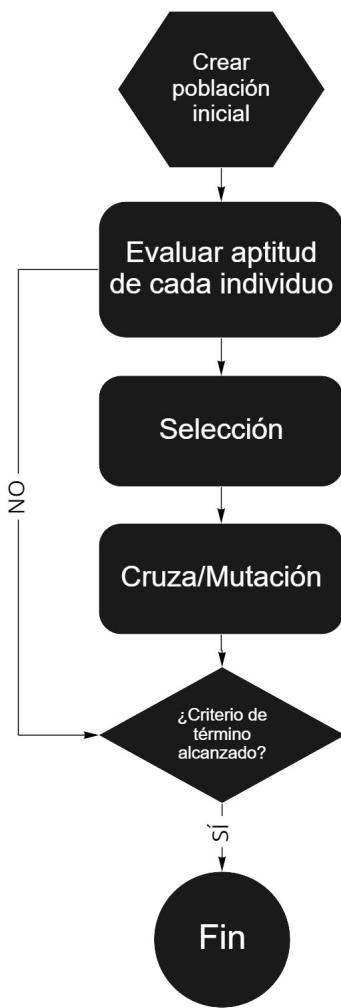


Figura 1.5: Diagrama de un EA

Las estrategias más populares (para la ingeniería de control) en los que se basan los diferentes EA's parecen ser los Algoritmos Genéticos (GA), la Optimización por Enjambre de Partículas (PSO) y la Evolución Diferencial (Differential Evolution, DE). En este punto, es importante señalar que no existe una técnica evolutiva mejor que las otras, dado que cada una de ellas tiene sus ventajas y desventajas.



Algoritmos genéticos (GA)

Los algoritmos genéticos están muy influidos por un fenómeno natural. En este caso, la analogía es con la teoría biológica de la evolución formulada por Charles Darwin a mediados del siglo XIX.

Las soluciones factibles de un problema específico corresponden a los miembros de una especie particular, donde la aptitud de cada miembro ahora se mide por el valor de la función objetivo. Se trabaja con una población completa de soluciones de prueba. En el caso de cada iteración (generación) de un algoritmo genético, la población actual consiste en el conjunto de soluciones de prueba que en la actualidad están bajo consideración. Estas soluciones de prueba se entienden como los miembros vivos de la especie. Algunos de los miembros más jóvenes de la población (en especial los miembros más aptos) sobreviven en la adultez y se convierten en padres (aparejados de manera aleatoria) que después tienen hijos (nuevas soluciones de prueba) que tienen algunas de las características (genes) de ambos padres. Como los miembros más aptos de la población tienen una mayor probabilidad de convertirse en padres que los otros, a medida que avanza, un algoritmo genético tiende a generar poblaciones mejoradas de soluciones de prueba. De vez en cuando ocurren mutaciones, de forma que los hijos también pueden adquirir características (algunas veces deseables) que no posee ninguno de los padres. Este fenómeno ayuda a los algoritmos genéticos a explorar una parte de la región factible, quizá mejor que la que se consideró antes. Por último, la supervivencia del más apto tiende a conducir al algoritmo genético hacia una solución de prueba (la mejor de todas las consideradas) que al menos es cercana a la óptima [37].

Los algoritmos genéticos [40] [41] se basan en la evolución de las especies. Cada individuo (possible solución del problema de optimización) está caracterizado por un cromosoma (codificación de esa solución). El algoritmo genético se basa en hacer evolucionar un conjunto de individuos (población) para conseguir que sus individuos se acerquen a la solución óptima con el paso de cada generación (iteración del



algoritmo). El mecanismo de evolución básico consiste en ir generando nuevos individuos modificando los cromosomas de los individuos de la población existente, bien mezclándolos entre sí (cruce), efectuando un cambio aleatorio (mutación). Además de la generación de nuevos individuos, existe un proceso de selección para determinar los individuos que continúan en la población.

En Algorithm 1, se puede ver el pseudo-código de un algoritmo genético básico para una optimización mono-objetivo.

Algorithm 1 GA Básico

- 1: Generar población inicial $P|_0$ con N_p cromosomas
 - 2: Evaluar $P|_0$
 - 3: **while** Criterio de terminación no alcanzado
 - 4: Leer contador de generaciones G
 - 5: Crear nueva población $P|_G^*$ a partir de $P|_G$
 - 6: Seleccionar cromosomas padres
 - 7: Efectuar el operador de cruce para crear un hijo a partir de los padres
 - 8: Efectuar el operador mutación en el hijo
 - 9: Seleccionar los cromosomas más aptos de $P|_G^* \cup P|_G$ para generar a población $P|_{G+1}$
 - 10: $G = G + 1$
 - 11: **end while**
-

Algoritmos de evolución diferencial (DE)

El algoritmo de Evolución Diferencial [42] [43] [44] es utilizado principalmente por su simplicidad y compactibilidad [44]. Existen muchas versiones del algoritmo; la versión original [42] emplea tres operadores: mutación, recombinación y selección.

Evolución diferencial es una técnica metaheurística basada en poblaciones de vectores numéricos. Entre las ventajas más destacadas de este algoritmo pueden mencionarse la simplicidad, eficiencia, propiedad de búsqueda local y velocidad. El



proceso que sigue este algoritmo para resolver un problema de optimización se caracteriza por iterar sobre una población de vectores para hacer evolucionar soluciones candidatas con respecto a una función de aptitud [42] [45].

En Algorithm 2, se describen los pasos a seguir para el algoritmo de DE básico para optimización mono-objetivo:

Algorithm 2 DE Básico

- 1: Generar población inicial $P|_0$ con N_p individuos
 - 2: Evaluar $P|_0$
 - 3: **while** Criterio de terminación no alcanzado
 - 4: Leer contador de generaciones G
 - 5: **do** Para cada $\in P|_G$
 - 6: Generar un vector mutante (1.1)
 - 7: Generar un vector hijo (1.2)
 - 8: Efectuar una comparación por pares Padre-Hijo (1.3)
 - 9: **end do**
 - 10: $G = G + 1$
 - 11: **end while**
-

$$\mathbf{v}_{i,G} = \mathbf{x}_{r_1,G} + F(\mathbf{x}_{r_2,G} - \mathbf{x}_{r_3,G}) \quad (1.1)$$

$$u_{i,j,G} = \begin{cases} v_{i,j,G} & \text{si } \text{rand}(0,1) \leq C_r \\ x_{i,j,G} & \text{de otra forma} \end{cases} \quad (1.2)$$

$$\mathbf{x}_{i,G+1} = \begin{cases} \mathbf{u}_{i,G} & \text{si } f(\mathbf{u}_{i,G}) \leq f(\mathbf{x}_{i,G}) \\ \mathbf{x}_{i,G} & \text{de otra forma} \end{cases} \quad (1.3)$$

Para la optimización multiobjetivo, la comparación se realiza mediante el concepto de dominancia. Este concepto será presentado en la sección de optimización multiobjetivo.



Optimización por Enjambre de partículas (PSO)

La optimización por enjambre de partículas (Particle Swarm Optimization) es un método de optimización heurística orientado a encontrar mínimos o máximos globales. Su funcionamiento está inspirado en el comportamiento que tienen las bandadas de pájaros o bancos de peces en los que, el movimiento de cada individuo (dirección, velocidad, aceleración...), es el resultado de combinar las decisiones individuales de cada uno con el comportamiento del resto [46].

PSO simula el comportamiento colaborativo de las especies en la búsqueda de recursos para su supervivencia. Cada partícula está determinada por su posición x y velocidad \dot{x} en la generación k [47] [48].

$$\dot{x}^j|_k = \dot{x}^j|_{k-1} + \varphi_1 \beta_1(x^j|_{best}) + \varphi_2 \beta_2(x^{swarm}|_{best})$$

$$x^j|_k = x^j|_{k-1} + xj|k$$

$$x^j|_k = x^j|_{k-1} + xj|k$$

donde φ_1, φ_2 son factores empleados para ponderar el conocimiento local y global; β_1, β_2 son números aleatorios; donde $x^{swarm}|_{best}$ es la mejor posición (solución) conocida por el enjambre (conocimiento colectivo) mientras que $x^j|_{best}$ es la mejor posición conocida para la partícula j (conocimiento individual). En Algorithm 3 se muestra el pseudo-código para un PSO básico.

**Algorithm 3** PSO Básico

```

1: Generar el enjambre inicial  $P|_0$  con  $N_p$  partículas
2: Evaluar  $P|_0$ 
3: Inicializar para cada partícula su mejor posición conocida  $\mathbf{x}^j|_{best}$ 
4: Inicializar la mejor posición conocida del enjambre  $\mathbf{x}^{swarm}|_{best}$ 
5: Inicializar la velocidad de cada partícula  $\dot{\mathbf{x}}^j|_0$ 
6: while Criterio de terminación no alcanzado
7:   Leer contador de generaciones G
8:   do para cada partícula  $\mathbf{x}^j \in P|_G$ 
9:     Actualizar su velocidad  $\dot{\mathbf{x}}^j|_G$ 
10:    Actualizar su posición  $\mathbf{x}^j|_G$ 
11:    Actualizar la mejor posición conocida  $\mathbf{x}^j|_{best}$ 
12:    Actualizar el conocimiento colectivo del enjambre  $\dot{\mathbf{x}}^{swarm}|_{best}$ 
13:   end do
14:    $G = G + 1$ 
15: end while

```

Adicionalmente, algunas de las características deseables de los EA's son [34]:

- Convergencia
- Diversidad de soluciones
- Manejo de preferencias
- Tratamiento de restricciones
- Escalabilidad
- Optimización en entornos dinámicos
- La optimización robusta
- Optimización de funciones con alto índice computacional



De la anterior lista, es importante señalar el tratamiento de restricciones, ya que este aspecto es relevante cuando se considera que la mayoría de los problemas de ingeniería incluyen diferentes niveles de restricción, que pueden ir desde restricciones de operación, implementación, físicas, entre otras. Una de las técnicas más populares para afrontar las restricciones son las *reglas de factibilidad*. Un ejemplo clásico es el presentado por Deb [49], el cual consiste en 3 reglas sencillas.

1. Al comparar dos soluciones factibles, aquella que tenga la mejor función objetivo es seleccionada.
2. Al comparar una solución factible con una no-factible, la primera es seleccionada sobre la segunda.
3. Al comparar dos soluciones no-factibles, aquella con el menor grado de violación de las restricciones es seleccionada.

Optimización multiobjetivo (MOP)

Un MOP se expresa como:

$$\min \vec{F}(\vec{p}) = [f_1(\vec{p}), \dots, f_m(\vec{p})]^T$$

sujeto a:

$$g_i(\vec{p}) \leq 0, i = 1, \dots, n_g$$

$$h_j(\vec{p}) = 0, j = 1, \dots, n_h$$

$$p_k^{\min} \leq p_k \leq p_k^{\max}, k = 1, \dots, d$$

donde $\vec{p} = [x_1, \dots, x_d]^T$ es el vector de variables de diseño, que debe de ser encontrado para minimizar un vector \vec{F} de $m > 1$ funciones objetivo, las cuales están en conflicto una con la otra. En muchas aplicaciones del mundo real, los MOP están sujetos a restricciones funcionales de la forma de $g_i(\vec{p})$ y $h_j(\vec{p})$ las cuales son llamadas respectivamente restricciones de desigualdad e igualdad. Adicionalmente, las variables de



diseño pueden ser limitadas, i.e., $p_k \in [p_k^{min}, p_k^{max}]$; todas estas restricciones están relacionadas con las limitaciones del mundo real (por ejemplo, energía y tamaño, entre otras). El espacio de soluciones factibles (donde se satisfacen todas las restricciones) se denota por Ω [39].

Dominancia de Pareto

Un vector $\vec{F}(\vec{p}) = [f_1(\vec{p}), \dots, f_m(\vec{p})]^T$ se dice que domina a $\vec{F}(\vec{q}) = [f_1(\vec{q}), \dots, f_m(\vec{q})]^T$ (denotado por $\vec{F}(\vec{p}) \preceq \vec{F}(\vec{q})$) si y solo si $\vec{F}(\vec{p})$ es tan bueno como $\vec{F}(\vec{q})$ para todo los objetivos, i.e., $f_i(\vec{p}) \leq_i (\vec{q}), \forall i \in \{1, \dots, m\}$, y para al menos un objetivo $f_i(\vec{p}) < f_i(\vec{q})$ [39].

Óptimo de Pareto

Un vector de decisión $\vec{p} \in \Omega$ es un óptimo de Pareto si no existe una función $f_i(\vec{p})$ que pueda mejorar sin empeorar el resto, i.e., $\nexists \vec{q} \in \Omega$ tal que $\vec{F}(\vec{p}) \preceq \vec{F}(\vec{q})$ [39].

Conjunto óptimo de Pareto

El conjunto óptimo de Pareto \mathcal{P}^* contiene lo más posible el vector de decisión óptimo $\vec{p} \in \Omega$, i.e., $\mathcal{P}^* = \{\vec{p} \in \Omega | \nexists \vec{q} \in \Omega, \vec{F}(\vec{p}) \preceq \vec{F}(\vec{q})\}$. [39]

Frente de Pareto

El frente de Pareto contiene los valores de los vectores objetivo evaluados en \mathcal{P}^* , i.e., $\mathcal{PF}^* = \{\vec{F}(\vec{p}) | \vec{p} \in \mathcal{P}^*\}$. [39]

Índices de desempeño de un sistema controlado

Los índices de desempeño son una medida cuantitativa del desempeño de un sistema y se eligen de manera que se de énfasis a las especificaciones importantes del mismo. Dicho de otra manera, son un número que nos indica cuan bien funciona



un sistema de control, y por esta razón , representan una buena opción para ser escogidos como función objetivo [50].

En la literatura han sido abordados diferentes criterios correspondientes a índices de desempeño basados en la integral del error y del control. Son utilizados en sistemas controlados para evaluar la respuesta del sistema frente a perturbaciones o cambios de parámetros. A continuación se mencionan algunos de los más usados:

Integral Absolute Error Criterion (IAE)

Este criterio es de fácil aplicación y proporciona una respuesta aceptable a la salida del lazo de control. Sin embargo, si se emplea este criterio en sistemas que son altamente subamortiguados o altamente sobreamortiguados, no conseguirá el óptimo. Un sistema óptimo basado en este criterio es un sistema que posea un amortiguamiento razonable y una respuesta transitoria satisfactoria [51].

$$IAE = \int_0^T |e(t)| dt$$

Integral of the Absolute value of the control action (IADU)

Se caracteriza por indicar la suavidad de la señal de control[52].

$$IADU = \int_0^T |u(t) - u(t - 1)| dt$$

Integral Time Absolute Error (ITAE)

Al estar el valor absoluto del error multiplicado por el tiempo, permite que errores grandes que ocurren al inicio de la respuesta donde aún los tiempos son pequeños tengan poco peso, y los errores pequeños que se dan durante la etapa final de la respuesta sean más grandes. Por otra parte, se caracteriza por ser un criterio muy



selectivo pero difícil de evaluar analíticamente [50].

$$ITAE = \int_0^T t|e(t)|dt$$

Integral square-error criterion(ISE)

Este criterio da mayor ponderación a los errores elevados del sistema, que comúnmente se dan al inicio de la respuesta, y con menos peso significativo evalúa los errores pequeños que se dan al final de la misma. [50].

Un sistema se considera óptimo si consigue minimizar este índice de desempeño. En particular el ISE se usa con frecuencia cuando se implementan entradas determinísticas (escalón) y entradas estadísticas dado a su facilidad de implementación tanto digital como analíticamente [50]. Este criterio está definido por:

$$ISE = \int_0^T e^2(t)dt$$

Indicadores de Calidad

Durante el proceso de diseño MOO, existen algunas consideraciones que deben de presentar los indicadores de rendimiento, los cuales son de suma importancia. Estos son, comúnmente: capacidad, convergencia y diversidad. Por lo tanto, podemos categorizar a estos indicadores MOO en cuatro grupos principales [53] :

Siendo S el conjunto solución óptimo:

- Indicadores de capacidad: Las métricas de este grupo cuentan el número o conjunto de soluciones no dominadas en S que satisface a los requisitos predefinidos.
- Indicadores de convergencia: Estas son métricas para medir la proximidad del conjunto de solución óptima en el FP.
- Indicadores de diversidad: Estas métricas indican dos tipos de información:
 - 1) La distribución mide qué tan uniformemente dispersas son las soluciones



de S en el espacio objetivo y 2) la propagación indica qué tan bien llegan las soluciones de S a los extremos del FP.

- Indicadores de convergencia-diversidad: Indican tanto la convergencia como diversidad de S en una sola escala.

Para este trabajo, estamos buscando medir convergencia y diversidad en un frente dado, por lo que haremos uso de alguna métrica del cuarto grupo mencionado. Por esta razón, usaremos la métrica llamada Hipervolumen (HV), que está definida como:

$$HV(S, R) = \text{volume} \left(\bigcup_{i=1}^{|S|} v_i \right)$$

Donde $R = \{W\}^2$

Esta métrica calcula el volumen (en el espacio de objetivos) cubierto por miembros de un conjunto dado S , de soluciones no dominadas para problemas donde todos los objetivos han de ser minimizados. Matemáticamente, para cada $i \in S$ se construye un hipercubo v_i con un punto de referencia W y la solución i que definen la diagonal del mismo. El punto W se puede obtener simplemente con los peores valores de las funciones objetivo. Entonces, la unión de todos los hipercubos es lo que define el hipervolumen.

Los algoritmos que alcanzan mayores valores para HV son mejores. Es necesario normalizar las soluciones no dominadas puesto que HV depende del escalado de los valores de la función objetivo [54]. En la figura 1.6 se puede observar esta métrica.

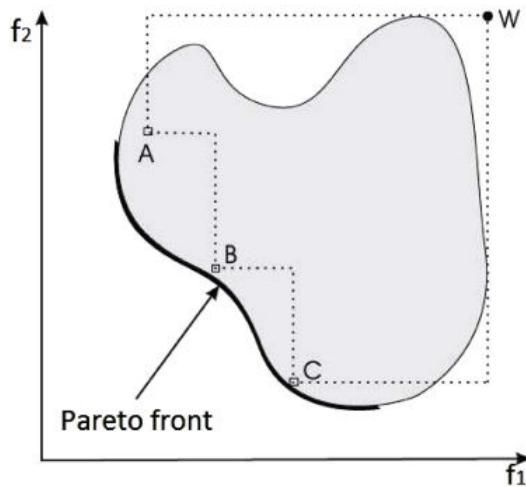


Figura 1.6: Hipervolumen (HV).

Pruebas no paramétricas

Son aquellas que no presuponen una distribución de probabilidad de datos, se conocen también como de distribución libre [55].

Pruebas del signo

Usa los signos + y - para establecer una diferencia en función de la dirección, no en magnitud de un par de observaciones, en la medición en lugar de cantidades.

Esta prueba es aplicable para el caso donde se tengan dos muestras relacionadas y se requiere mostrar que ambas condiciones son diferentes. El único requisito a la prueba es la continuidad a la variable considerada [56].

Procedimiento de la prueba

- Se determina el signo de diferencia entre los dos datos de cada pareja, pero



previamente se ha establecido un orden en la muestra que está constituida por las parejas, es decir, cual es el primer dato y cual es el segundo dato, de acuerdo con el contexto del problema o la naturaleza de la información.

- Se determina el valor de “n”, es decir, el número de parejas cuyas diferencias exhiben un signo, con lo cual se establece que aquellas parejas con igual valor tienen una diferencia de cero; luego entonces no deben ser consideradas en el conteo de signos.
- Se formula el planteamiento de la hipótesis.
- Se aplica la aproximación a una distribución normal con el resto de los pasos de una prueba de hipótesis [56].

Prueba de rango con signo de Wilcoxon

Se utiliza comúnmente como alternativa a la prueba paramétrica t de Student cuando la muestra necesite probar una hipótesis que tiene relación con un parámetro que refleje una tendencia central. Cuando se violan las suposiciones de la prueba t, la prueba de Wilcoxon, ofrece detectar diferencias significativas. La prueba de Wilcoxon con rangos de signo muestra que tan grande o pequeña es una muestra frente a una mediana hipotética [56].

Procedimiento de la prueba

- Mediante la diferencia, se le asigna un signo a cada pareja, donde el orden ya se ha establecido, es decir $z_i = y_i - x_i$ donde $i = 1, 2, 3, \dots, n$.
- Obtener un grupo de n diferencias absolutas , sin tomar en cuenta los signos.
- Se omiten los pares con diferencia absoluta 0, por lo que se forma un conjunto de \hat{n} , donde $\hat{n} \leq n$.
- Ordenar de manera incremental las diferencias absolutas entre pares de valores, esto con el fin de asignarle un rango R_i de 1 hasta \hat{n} , como las diferencias son



absolutas, es posible que se repitan valores, pero entonces se les asigna el rango promedio de sus posiciones que ocuparían si no hubiera ocurrido el empate.

- Una vez ordenados, se le asigna en su rango el signo que le corresponde a cada valor de los n valores, para posteriormente sumar los rangos positivos y negativos por separado.

Se suman los rangos positivos ($W^+ = \sum R_i(+)$) y negativos ($W^- = \sum R_i(-)$).

La prueba se fundamenta en la hipótesis nula $H_0 : \mu_A = \mu_B$, eso significa que los totales de las sumas de los rangos positivos y negativos son aproximadamente iguales, en caso contrario la hipótesis alternativa esta denotada por $H_1 : \mu_A \neq \mu_B$, lo que significa es, si ambos totales (W^+, W^-) son pequeños en una prueba de dos extremos [56]. En otras palabras:

$H_1 : \mu_A < \mu_B$, sucede si W^+ es pequeña y W^- es grande.

$H_1 : \mu_A > \mu_B$, sucede si W^+ es grande y W^- es pequeña.

Método Analítico Jerárquico (AHP)

Desarrollado por Thomas L. Saaty en 1980, se trata de un procedimiento de comparación por pares de los criterios que parte de una matriz cuadrada en la cual el número de filas y columnas está definido por el número de criterios a ponderar. De esta manera, se establece una matriz de comparación entre pares de criterios, comparando la importancia de cada uno de ellos con los demás. Despues, se establece el eigenvector principal, el cual establece los pesos que paralelamente proporcionan una medida cuantitativa de la consistencia de los juicios de valor entre pares de factores [57]. Este método requiere que quien toma las decisiones proporcione evaluaciones subjetivas respecto a la importancia relativa de cada uno de los criterios y que después especifique su preferencia con respecto a cada una de las alternativas de decisión y para cada criterio. El resultado es una jerarquización con prioridades que muestran la preferencia global para cada una de las alternativas de decisión [58].



Algunas de las ventajas de AHP frente a otros métodos de decisión multicriterio son:

- Permite analizar y descomponer un problema por partes.
- Presenta un sustento matemático.
- Permite medir criterios cuantitativos y cualitativos mediante una escala común.

Este método utiliza comparaciones pareadas como base fundamental, utiliza una escala subyacente con valores de 1 a 9 para calificar las preferencias relativas de los dos elementos. Saaty [57] establece una escala numérica del 1 al 9, donde dos elementos que se están comparando pueden tener diferentes niveles de importancia:

- 1 : Igualmente importante.
- 3 : Ligeramente más importante.
- 5 : Notablemente más importante.
- 7 : Demostrablemente más importante.
- 9 : Absolutamente más importante.

Los valores 2,4,6 y 8 son valores intermedios.



1.4. Marco procedimental

El propósito principal de los controladores es garantizar la estabilidad de la respuesta dinámica del sistema. El correcto establecimiento de los parámetros del controlador suele ser una tarea difícil, conocida como ajuste del controlador, y depende de las necesidades de la aplicación. El problema general de sintonización del controlador se refiere a la búsqueda de un conjunto de parámetros del controlador que estabilice la respuesta dinámica del sistema bajo un criterio de rendimiento establecido que se ajuste a las necesidades anteriores [39].

Sintonización de controladores multiobjetivo

Las aplicaciones de ingeniería se han vuelto cada vez más exigentes a medida que pasa el tiempo, ya que requieren sistemas dinámicos que satisfagan necesidades diferentes y a menudo conflictivas, aspecto de ingeniería que encaja correctamente con la teoría de la optimización multiobjetivo. Los pasos habituales para la sintonización del controlador multiobjetivo son descritos en [39] y se muestran en la Figura 1.7

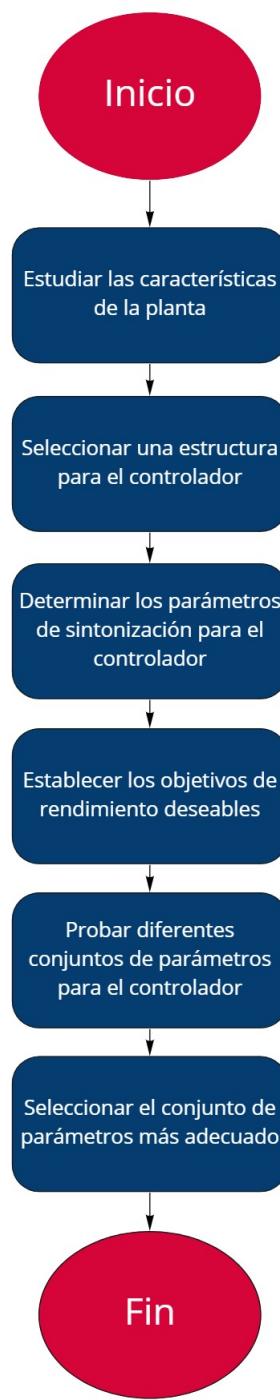


Figura 1.7: Pasos típicos para la sintonización de controladores multiobjetivo.

A continuación se describe brevemente cada uno de estos pasos:

1. **Estudiar las características de la planta:** Analizar el número de entradas y salida, comportamiento lineal o no lineal y la naturaleza de la actividad a realizar.
2. **Seleccionar una estructura para el controlador:** Escoger una estructura apropiada para el controlador, de tal manera que el comportamiento de la planta alcance la estabilidad.
3. **Determinar los parámetros de sintonización para el controlador:** Toda estructura de controlador lleva parámetros directamente relacionados con el comportamiento de la planta. Es de suma importancia que el diseñador determine dichos parámetros.
4. **Establecer los objetivos de rendimiento deseables:** Estos criterios son los que definen el problema multiobjetivo y están directamente relacionados con la minimización o maximización de indicadores cuantitativos.
5. **Probar diferentes conjuntos de parámetros para el controlador:** Como se ha mencionado antes, los métodos metaheurísticos buscan un conjunto de soluciones candidatas que satisfagan distintos niveles de compromiso entre los objetivos propuestos.
6. **Seleccionar el conjunto de parámetros más adecuado:** Del conjunto de soluciones obtenido en el paso anterior, se selecciona una de ellas de acuerdo con el criterio del diseñador, para finalmente ser implementada.

Optimización multiobjetivo con métodos metaheurísticos para la sintonización de controladores

En la Figura 1.8, se presenta un esquema de un MOP de sintonización de controladores basado en diferentes criterios de desempeño, donde se utiliza un optimizador



metaheurístico para encontrar los parámetros del controlador con los mejores compromisos. A través de una simulación dinámica, cada configuración de los parámetros debe probarse, con el objetivo de medir el rendimiento del controlador y seleccionar las alternativas más prometedoras y/o adecuadas. Una vez que se encuentran los mejores compromisos, el diseñador es responsable de elegir un solo parámetro de configuración a implantar en el controlador real [39].

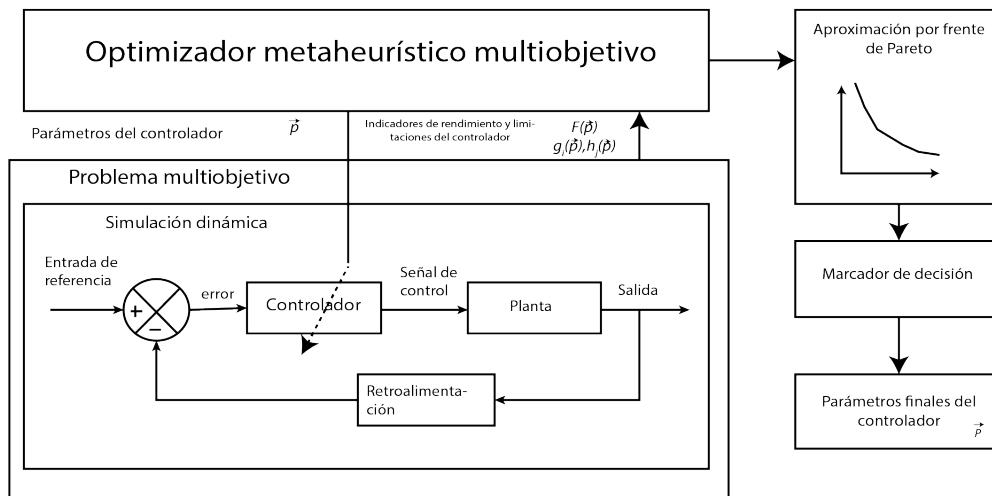


Figura 1.8: MOP optimizado con un método metaheurístico

Metodología

En este proyecto se propone usar la norma VDI 2206, la cual está destinada al desarrollo de productos mecatrónicos. Esta norma propone un modelo en V para describir el trayecto que tienen que seguir ciertas tareas de forma cíclica, y se basa en la concurrencia para el diseño de sistemas multifuncionales [59]. La Figura 1.9 representa la metodología propuesta en forma general.

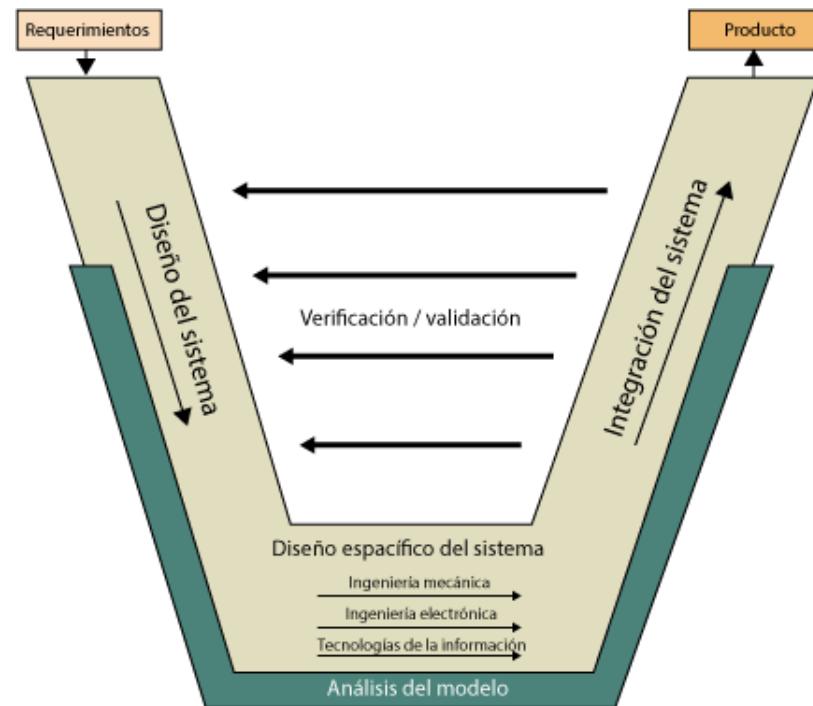


Figura 1.9: Modelo en V.

El proceso consta de tres fases.

- Diseño del sistema
- Diseño específico
- Integración del sistema

Estas fases se llevan a cabo de manera paralela, a partir de los requerimientos propuestos, con el objetivo de obtener un producto. Es muy importante señalar que en esta norma se está recibiendo retroalimentación continuamente para una mejora.

Una vez establecidas las fases que conforman el modelo V, se mencionan las principales tareas de la norma VDI 2206, las cuales son:

1. Definición de requisitos.

En esta etapa se investigan, analizan y enlistan los requerimientos más repre-



sentativos del proyecto. A lo largo del proceso pueden ser mejorados o alterados, pero no eliminados.

2. Diseño preliminar del sistema.

Durante esta tarea se realiza la división de los subsistemas del proyecto con base en los requerimientos planteados.

3. Diseño específico del sistema.

Al desarrollar cada subsistema se obtiene su completo dominio y entendimiento para permitir cualquier cambio o resolución del problema si es que fuera necesario. Esto garantiza el desarrollo de los ya mencionados con vista a una unión y formación integral del proyecto.

4. Integración del sistema.

Al completar el desarrollo individual de los subsistemas, estos son acoplados para formar el proyecto. El diseño del proyecto se basa en trabajar por partes pensando en su unión final para evitar problemas con la integración.

5. Verificación / Validación.

En cada etapa desde el diseño, desarrollo y comprobación, se realizará una validación constante para generar retroalimentación.

6. Modelado y análisis del modelo.

Se realizará un modelado del sistema a desarrollarse y analizará su funcionalidad, desempeño y mejoras. Esto permite un concepto claro del proyecto y futuras mejoras o desarrollo de investigaciones en base del mismo. El modelado es aplicado a cada subsistema y su complementación para formar uno solo.

Generalmente un producto mecatrónico complejo no se produce en un solo ciclo de diseño como en la Figura 1.9, sino que se requieren varios, como se puede observar en la Figura 1.10. El producto es el resultado de unos ciclos de diseño, no tiene por qué ser el producto final, sino versiones previas que hayan sido necesarias para su



desarrollo, como un prototipo de laboratorio, un prototipo funcional, un producto piloto, etc.

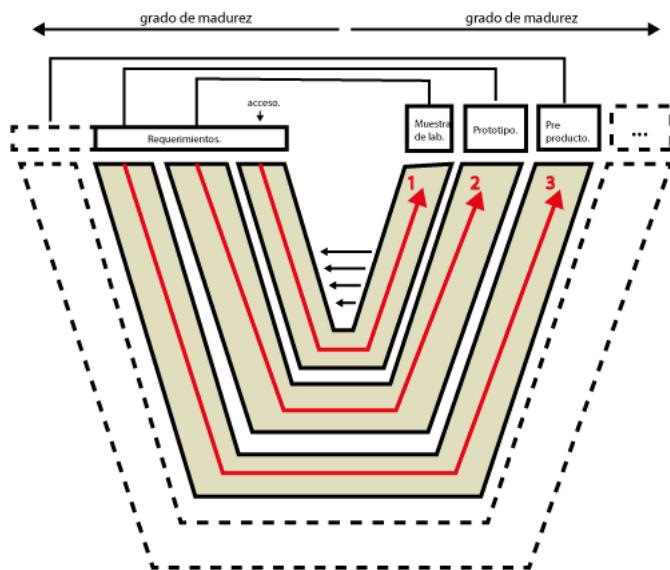


Figura 1.10: Macro ciclos y madurez del producto

Diseño del sistema

2.1. Diseño conceptual

En esta sección, se presentarán el proceso de diseño de concepto del sistema. Es decir, se realizará la identificación de necesidades y requerimientos con el fin de buscar y presentar una propuesta de concepto solución que satisfaga estas necesidades, posteriormente se realizará la validación del concepto y se presentará el concepto solución final.

Necesidades y requerimientos

Como primer paso, es necesario identificar las necesidades que debe resolver el sistema, esto permite plantear las especificaciones que tendrá el sistema. Las necesidades que se lograron plantear para este trabajo son las que contiene la tabla 2.1



TABLA 2.1: Necesidades del sistema.

Identificador	Necesidad
N1	Sintonización de controladores PID con métodos metaheurísticos.
N2	Agilizar el proceso de la sintinización.
N3	Mostrar en un frente de pareto el conjunto de soluciones entre los distintos niveles de compromiso de las funciones objetivo.
N4	Mostrar en una tabla el conjunto de soluciones entre los distintos niveles de compromiso de las funciones objetivo.
N5	Una interfaz intuitiva.
N6	Cumplir con tarea de regulación de posición.
N7	Una señal de control suave.
N8	Mostrar una simulación del resultado esperado.

Los requerimientos del sistema funcionan como indicadores de las especificaciones del sistema, esto con el fin de establecer rangos de valores permitidos y establecer que necesidades son alcanzables y hasta que punto se puede mejorar, los requerimientos que se identificaron se presentan en la Tabla 2.2

TABLA 2.2: Requerimientos

Identificador	Requerimientos	Valor
R1	Error	Bajo
R2	Suavidad	Alto
R3	Tiempo de ejecución	Adecuado
R4	Interfaz de usuario	Intuitiva
R5	Métodos metaheurísticos	3
R6	Consumo energético	Bajo
R7	Precisión	Alta
R8	Coincidencia en respuesta	Alta



Validación de requerimientos

En la Tabla 2.3 se muestra la verificación de necesidades con los requerimientos, pues de esta manera se valida que cada requerimiento se asocie al menos con una necesidad.

TABLA 2.3: Matriz de trazabilidad

	N1	N2	N3	N4	N5	N6	N7	N8
R1	X					X	X	
R2	X	X					X	
R3		X						X
R4			X	X	X			X
R5	X			X				
R6						X		
R7						X	X	X
R8	X		X	X		X	X	X

Arquitectura funcional

En esta sección se realizará la división por funciones, en funciones secundarias, esto con el propósito de hacer el proceso de diseño del concepto más claro. Para llevar a cabo esto, se utilizará la metodología gráfica IDEF, la cual consiste en el modelado de procesos y es utilizada en la implementación de sistemas y software en ingeniería. A pesar de que IDEF se desarrolló originalmente para el entorno de fabricación, actualmente esta metodología de modelado de procesos se ha aplicado para usos más amplios y para el desarrollo de software en general. Cabe resaltar que IDEF se refiere a una familia de lenguaje de modelado, y consiste en 16 métodos diferentes. Estos métodos de modelado de procesos cubren una amplia gama de usos y cada método captura un determinado tipo de datos. La tabla 2.4 enumera los 16 métodos IDEF, siendo IDEF0 a IDEF4 los métodos más utilizados [60] .

TABLA 2.4: Familia IDEF

IDEF0	Modelado De Funciones
IDEF1	Modelado De Información
IDEF1X	Modelado De Datos
IDEF2	Diseño De Simulación De Diseño
IDEF3	Captura De Descripción De Procesos
IDEF4	Diseño Orientado A Objetos
IDEF5	Captura De Descripción Ontológica
IDEF6	Diseño De Captura Racional
IDEF7	Revisión De Cuentas De Información Del Sistema
IDEF8	Modelado De Interfaz De Usuario
IDEF9	Diseño De Escenarios
IDEF10	Modelado De Artefactos De Implementación
IDEF11	Modelado De Artefactos De Información
IDEF12	Modelado De Organización
IDEF13	Diseño De Mapeo De Tres Esquemas
IDEF14	Diseño De Redes



Dado que nos interesa el modelado de funciones, IDEF0 resulta ser la metodología más adecuada. En la Figura 2.1 se describe la función general en un diagrama IDEF0.

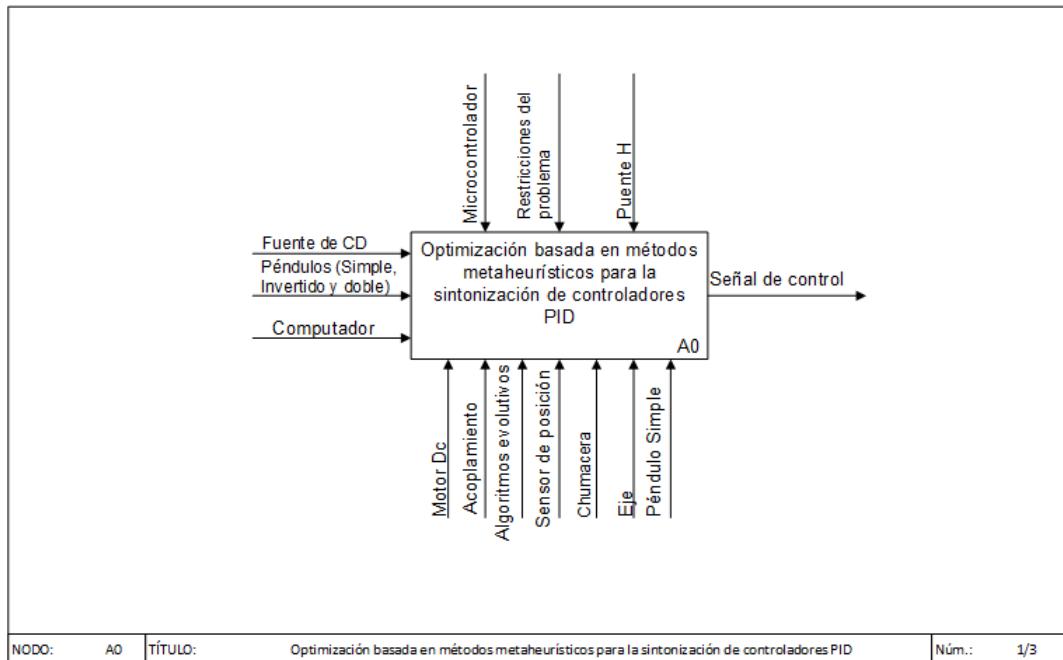


Figura 2.1: Diagrama IDEF0.

Para la sintonización de controladores PID es necesario el modelado de las plantas, convertir las características de la planta en un problema multiobjetivo, y para una verificación de los resultados se implementará el control en el péndulo físico de manera física. Las funciones se presentan a continuación:

A1. Modelado del sistema

El modelado del sistema permite tomar y verificar que la ecuación obtenida de la dinámica de cada péndulo sea correcta. Para esto se implementará una simulación 2D del sistema en Python, así como el monitoreo de las variables de control ya establecidas en el modelo dinámico del sistema.

A2. Sintonización con métodos metaheurísticos para optimización multiobjetivo.



La sintonización de los sistemas de control PID se realizará con algoritmos evolutivos, para eso es necesario plantear el problema de optimización, posteriormente resolver estos problemas con 3 algoritmos evolutivos, los cuales están inspirados en la teoría de la evolución. Los resultados se presentarán en frentes de Pareto, los cuales proporcionan un conjunto de soluciones que cumplen con diferentes niveles de compromisos con las funciones objetivo.

A3. Interfaz visual

Esta función cumple con la acción de presentar los resultados en una interfaz de usuario donde se estarán presentes los frentes de Pareto, así como los valores de las variables de diseño. Esto permitirá la selección de las ganancias de control que se adapten de mejor manera a las necesidades en cada péndulo.

A4. Adaptación del péndulo simple

El péndulo simple debe de ser adaptado para que se pueda mover con un actuador.

A5. Control del péndulo simple

Se implementará el control de regulación de posición del péndulo, pues ya se tienen los valores de las ganancias del controlador se debe verificar que los valores calculados cumplan con los objetivos deseados.

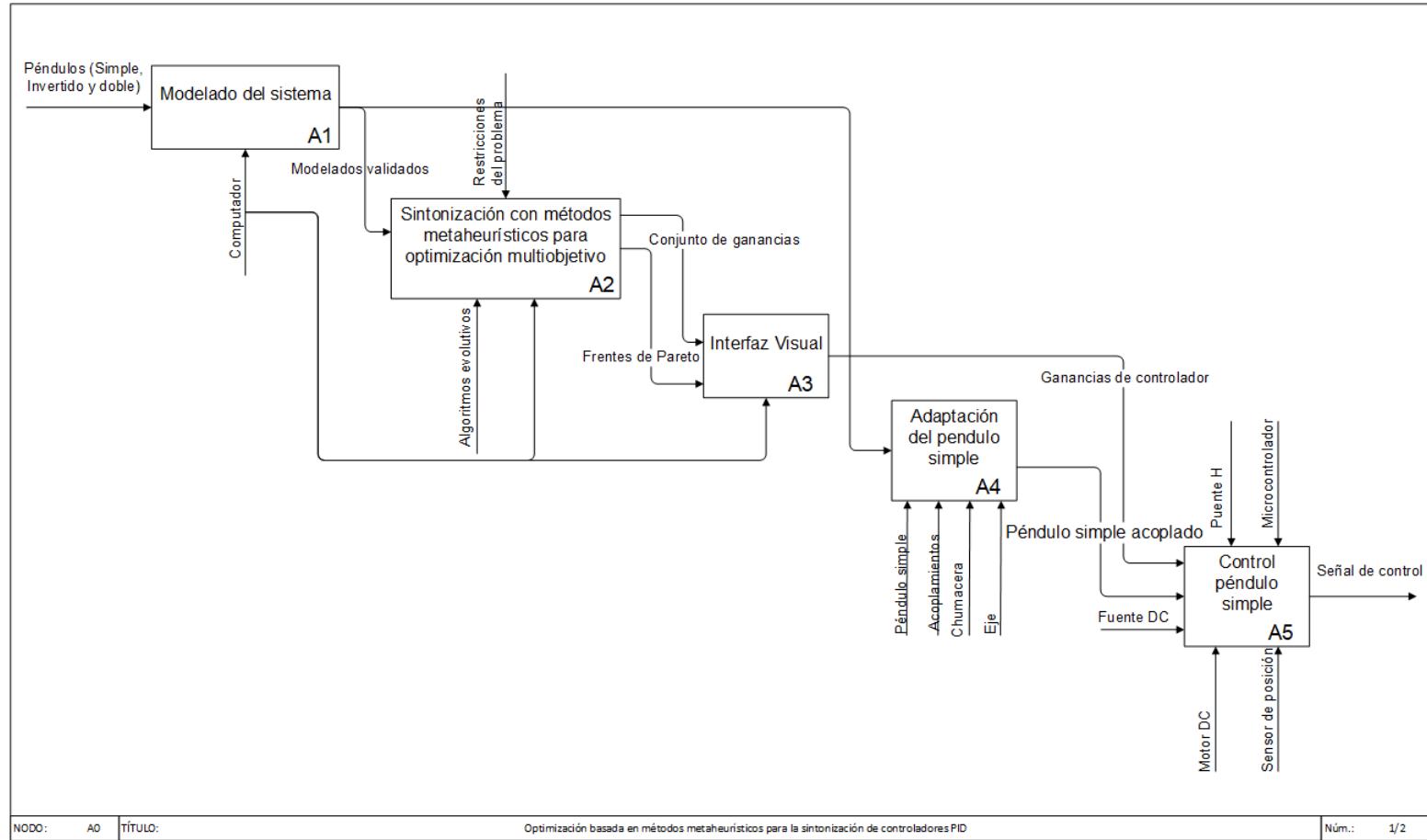


Figura 2.2: Diagrama IDEF0 Nodo A0.



Es necesario descomponer el bloque A2 del diagrama IDEF0 para una mejor comprensión de esta función, es por ello que, la figura 2.3 muestra la descomposición de este bloque.

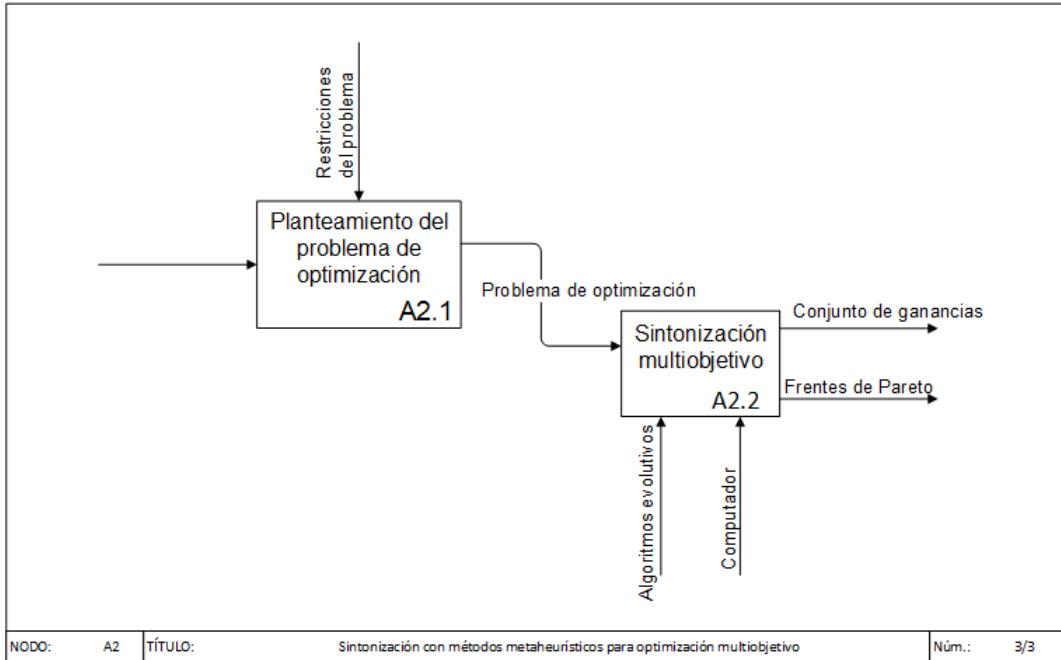


Figura 2.3: Diagrama IDEF0 Nodo A2.

Validación de funciones

Una vez que se han establecido las funciones que el sistema debe de llevar a cabo, es importante realizar una matriz de trazabilidad entre las funciones y los requerimientos del sistema, esto se puede observar en la tabla 2.5. La interfaz de usuario (R4) debe de mostrar la información más relevante del proceso de control, es por ello que este requerimiento, junto con R5 Y R8 están ligados con las funciones A1 y A3, puesto que ambas están enfocadas a la visualización de datos arrojados una vez que se ha ejecutado uno de los algoritmos evolutivos en busca de la sintonización multiobjetivo. Por otra parte, se busca que el sistema sobre el cual se probará la sintonización (A2) fuera de línea (A4), es decir, el sistema de péndulo simple, no



requiera de una gran cantidad de energía para funcionar (R6). Uno de los aspectos más importantes a evaluar, es que el resultado obtenido a través de las simulaciones realizadas en Python, coincide (R8) con los datos recolectados mediante un sensor implementado en el sistema físico, observando datos relevantes acerca del error (R1) y de la suavidad de la señal de control (R2)(A5). Por último, se espera que el costo computacional (R3) concuerde con la complejidad del problema de optimización.

TABLA 2.5: Matriz de trazabilidad para la validación de funciones.

	R1	R2	R3	R4	R5	R6	R7	R8
A1	X	X	X	X			X	X
A2	X	X	X		X			X
A3					X		X	X
A4						X		X
A5	X	X					X	

Arquitectura física

S1 Robótico

- Manipulador: Elemento que fungirá como brazo robótico de 1 grado de libertad (1GDL).
- Movimiento: Elementos que en conjunto permitan al sistema llevar a cabo una función específica.

S2 Información

- Percepción: Sensores cuyo propósito es obtener información del sistema, para posteriormente ser procesada y tomar decisiones con base en ella.
- Comunicación: Dispositivos que permitan realizar la transmisión y recepción de datos.



- Procesamiento: Componentes electrónicos que permitan analizar la información recopilada a través de los sensores implementados.

S3 Administración energética

- Fuente de alimentación que permita a los circuitos funcionar y realizar la tarea especificada para cada uno.

Matriz morfológica

En diseño ingenieril, el análisis morfológico es una herramienta conceptual de apoyo, ya que facilita visual y sistemáticamente la exploración de alternativas solución en el espacio de diseño. Una matriz morfológica es una herramienta usada para generar conceptos de diseño, combinando opciones solución con funciones inherentes al sistema [61]. En la figura 2.4 se muestra la matriz morfológica que será de utilidad para establecer rutas de diseño para el mecanismo.

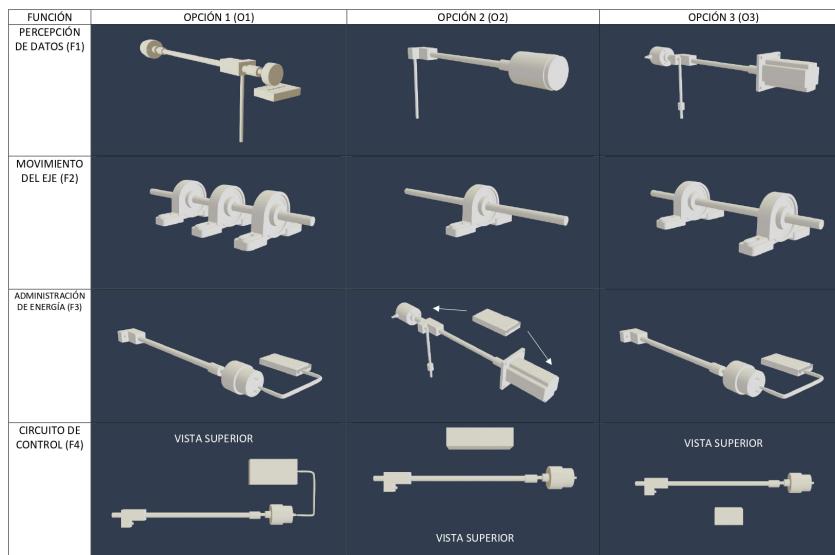


Figura 2.4: Matriz morfológica.



Búsqueda morfológica

- Ruta 1: O1F1-O1F2-O3F3-O3F4

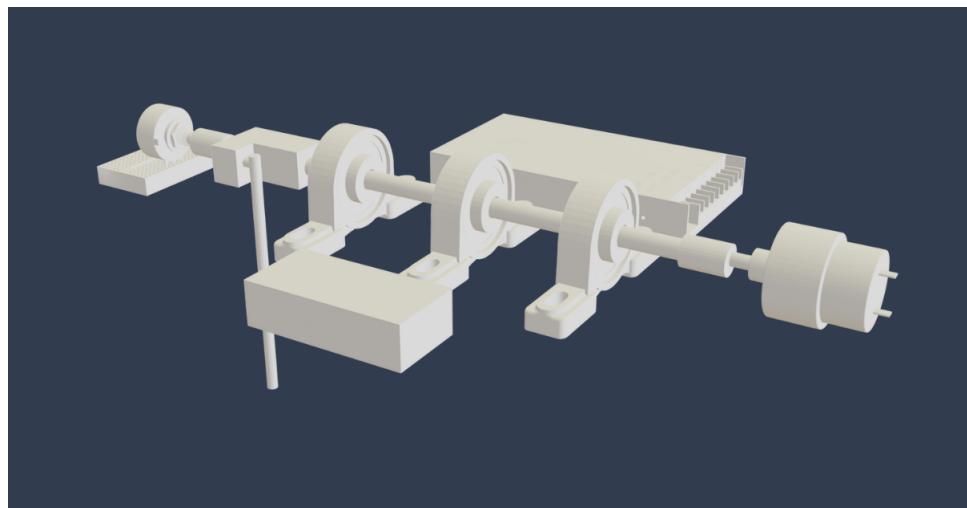


Figura 2.5: Ruta 1

Para la Ruta 1 que se observa en la Figura 2.5, se propone usar un motor en uno de los extremos, mientras que en el extremo opuesto se encuentra un sensor, como ejemplo, se visualiza un potenciómetro que estaría recibiendo la información proveniente del bloque que sujeta a la barra péndulo. Por otra parte, para mantener el eje fijo en traslación, se utilizarían tres chumaceras situadas a lo largo del mismo. Con esta distribución, la fuente de alimentación para el motor y circuito de control estaría situada a uno de los costados. Por último, la tarjeta de desarrollo estaría en otro de los costados, esperando que el computador se encuentre cerca de la ya mencionada.

- Ruta 2: O2F1-O3F2-O1F3-O2F4

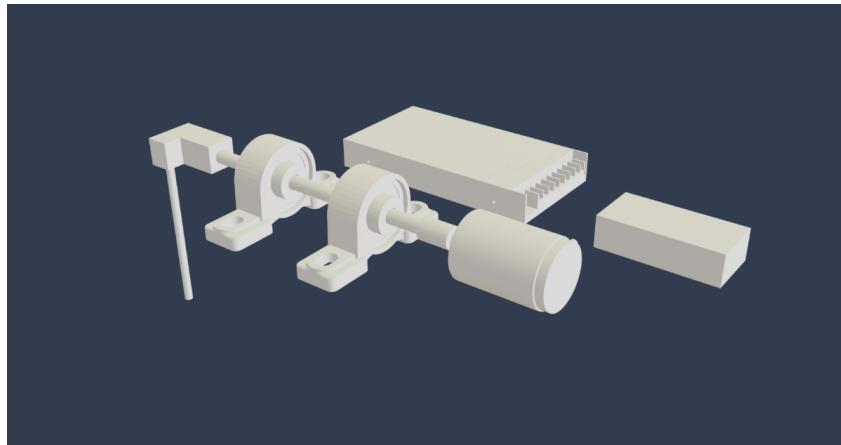


Figura 2.6: Ruta 2

En la Ruta 2 que se observa en la Figura 2.6, se propone utilizar un motor que tenga integrado un sensor de posición, como ejemplo, un encoder, de esta forma, tanto la fuente de alimentación como el circuito de control, estarían ubicados cerca del motor con sensor integrado, reduciendo el espacio ocupar por el mecanismo completo. Por otra parte, con el propósito de darle soporte al eje, basta con el uso de dos chumaceras distribuidas a lo largo del eje. Finalmente, el extremo opuesto al motor, se encontraría el bloque que sujet a la barra péndulo.



- Ruta 3: O3F1-O2F2-O2F3-O2F4

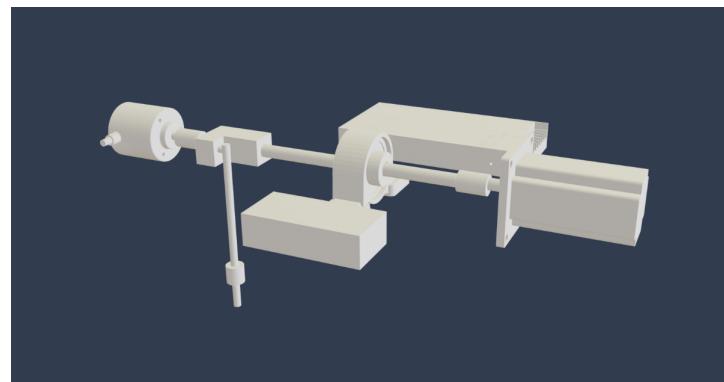


Figura 2.7: Ruta 3

Por último, en la Ruta 3 que se observa en la Figura 2.7, se sigue una distribución similar a la vista anteriormente en la figura 2.5, teniendo un motor en uno de los extremos y un sensor en el opuesto, sin embargo, en esta propuesta se tendría solo una chumacera para darle soporte al eje y tanto la fuente de alimentación como el circuito de control se encontrarían opuestos uno del otro sobre los costados del eje, teniendo en cuenta que la tarjeta de desarrollo encargada de recopilar los datos y mandar las señales al motor, se encuentre a una distancia cercana del computador.

Selección de diseño conceptual

Una vez descritas las rutas propuestas de diseño, se procede a utilizar el método AHP, descrito en la sección 1.3. La Tabla 2.6 contiene las variables a considerar durante el proceso, así como las rutas descritas en las Figuras 2.5, 2.6, y 2.7

TABLA 2.6: Variables y criterios.

	Número de elementos	Comunicación motor-sensor	Ensamble
Ruta 1	10	Media	Fácil
Ruta 2	8	Alta	Fácil
Ruta 3	8	Media	Fácil



En la Tabla 2.7 se comparan entre sí los criterios a evaluar, con el objetivo de obtener un vector de ponderación que será de utilidad más adelante.

TABLA 2.7: Matriz de comparación de criterios.

Criterios	Número de elementos	Comunicación motor-sensor	Ensamble	Matriz normalizada			Ponderación
Número de elementos	1.00	0.13	5.00	0.11	0.10	0.38	0.20
Comunicación motor-sensor	8.00	1.00	7.00	0.87	0.79	0.54	0.73
Ensamble	0.20	0.14	1.00	0.02	0.11	0.08	0.07
Total	9.20	1.27	13.00				

En la tabla 2.8 se tiene como primer criterio el número de elementos que principalmente conformarían el sistema, ya que se busca la reducción de elementos para facilitar el ensamble del ya mencionado.

TABLA 2.8: Criterio: Número de elementos

Alternativas	Ruta 1	Ruta 2	Ruta 3	Matriz normalizada			Vector promedio
Ruta 1	1	0.2	0.2	0.0909	0.1304	0.0476	0.0897
Ruta 2	5	1	3	0.4545	0.6522	0.7143	0.6070
Ruta 3	5	0.333333333	1	0.4545	0.2174	0.2381	0.3033
Total	11	1.533333333	4.2				

Es importante que la información proveniente del motor, sea recibida adecuadamente por el sensor, por lo que este punto es de gran importancia, la Tabla que tiene este criterio como eje central es la 2.9.

TABLA 2.9: Criterio: Comunicación motor-sensor

Alternativas	Ruta 1	Ruta 2	Ruta 3	Matriz normalizada			Vector promedio
Ruta 1	1	0.142857143	0.2	0.0769	0.1064	0.0323	0.0719
Ruta 2	7	1	5	0.5385	0.7447	0.8065	0.6965
Ruta 3	5	0.2	1	0.3846	0.1489	0.1613	0.2316
Total	13	1.342857143	6.2				



Por último, se busca que el ensamblaje del sistema sea sencillo, por lo que este punto está relacionado con el primer criterio. La comparación se muestra en la Tabla 2.10.

TABLA 2.10: Criterio: Ensamble

Alternativas	Ruta 1	Ruta 2	Ruta 3	Matriz normalizada			Vector promedio
Ruta 1	1	0.333333333	0.33333333	0.1429	0.2174	0.0526	0.1376
Ruta 2	3	1	5	0.4286	0.6522	0.7895	0.6234
Ruta 3	3	0.2	1	0.4286	0.1304	0.1579	0.2390
Total	7	1.533333333	6.33333333				

Finalmente, en la Tabla 2.11 se muestra el resultado obtenido con el método AHP, dando como resultado la **Ruta 2** mostrada en la Figura 2.6 como la ruta de diseño a desarrollar.

TABLA 2.11: Selección de ruta

Criterio/Alternativa	Número de elementos	Comunicación motor-sensor	Ensamble	Priorización
Ruta 1	0.089654307	0.071854707	0.13762668	0.07999996
Ruta 2	0.607001694	0.696531334	0.62340634	0.67371569
Ruta 3	0.303343999	0.231613959	0.23896698	0.24628435
Ponderación	0.20	0.73	0.07	

Propuesta solución

Modelado del sistema

En este apartado se presentan los modelos dinámicos de cada sistema obtenidos mediante la ecuación de Euler-Lagrange, posteriormente, se expresan estas ecuaciones en sus equivalentes en el espacio de estados, presentando el modelo final usando la expresión del modelo dinámico no lineal de un robot.

Péndulo Simple

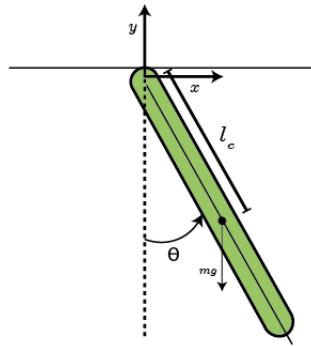


Figura 2.8: Péndulo simple.

La posición de la barra se describe mediante la siguiente ecuación:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} l_c \sin \theta \\ -l_c \cos \theta \end{bmatrix}$$

Derivando con respecto del tiempo, la velocidad está dada por la siguiente ecuación:

$$\dot{v} = \frac{d}{dt} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} l_c \cos \theta \\ l_c \sin \theta \end{bmatrix} \dot{\theta}$$



La energía cinética total del sistema:

$$K = \frac{1}{2}ml_c^2\dot{\theta}^2 + \frac{1}{2}I\dot{\theta}^2$$

La energía potencial del sistema:

$$U = mgh = -mgl_c \cos \theta$$

El lagrangiano que describe al sistema:

$$L = \frac{1}{2}ml_c^2\dot{\theta}^2 + \frac{1}{2}I\dot{\theta}^2 - (-mgl_c \cos \theta) = \frac{1}{2}ml_c^2\dot{\theta}^2 + \frac{1}{2}I\dot{\theta}^2 + (mgl_c \cos \theta)$$

$$D = \frac{1}{2}b\dot{\theta}^2$$

La ecuación de movimiento de Euler-Lagrange:

$$\ddot{\theta} = \frac{\tau - ((mgl_c \sin \theta) + b\dot{\theta})}{ml_c^2 + I}$$

Representación del modelo en ecuaciones de estado:

Las variables de estado son: $\theta, \dot{\theta}$

Ecuación de estados:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\theta} \\ \frac{u - ((mgl_c \sin \theta) + b\dot{\theta})}{ml_c^2 + I} \end{bmatrix}$$

La descripción paso a paso del desarrollo matemático se presenta en la sección de Apéndice A.

La validación del comportamiento del péndulo simple se realizó probando sus funcionamiento a diferentes condiciones iniciales, siendo el péndulo más sencillo y con menos grados de libertad es el más fácil de entender.

La programación de la simulación 2D de este péndulo se realizó en Python, el código fuente se puede consultar en la sección de Apéndice B.

A continuación, se presentan los resultados:



$$\theta = 0$$

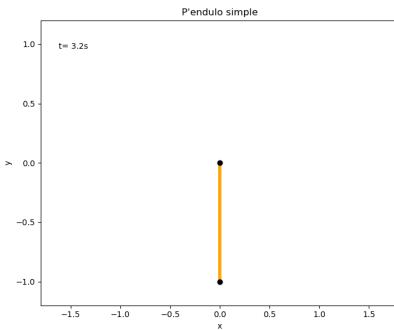


Figura 2.9: Posición del péndulo simple en el plano XY.

El péndulo no es forzado, por lo que, como se puede ver en la figura 2.10 en este punto el péndulo se encuentra en equilibrio, esto significa que el péndulo no se mueve, se mantiene estático en todo el periodo de tiempo de la simulación.

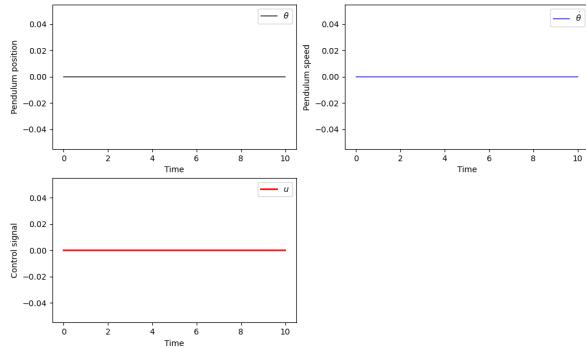


Figura 2.10: Gráficas de salida con respecto al tiempo del péndulo simple a $\theta = 0$.

También se puede observar en la figura 2.13 las gráficas de posición, velocidad del péndulo y, por último, la señal de control del actuador que proporciona movimiento al motor.

En todas las gráficas de posición y velocidad al nivel de 0, con esto se reafirma que



el péndulo se encuentra en su condición de equilibrio estático.

La señal de control en nivel 0 es debido a que no se le aplica ninguna entrada, es una oscilación libre, debido a que el péndulo se encuentra en una posición de equilibrio, el péndulo no se mueve.

$$\theta = \frac{\pi}{2}$$

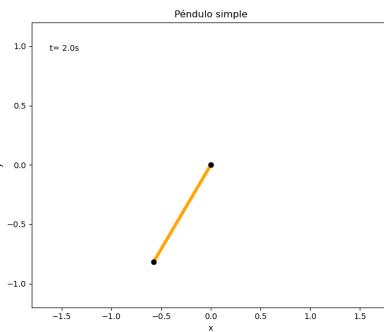


Figura 2.11: Posición del péndulo en el plano XY.

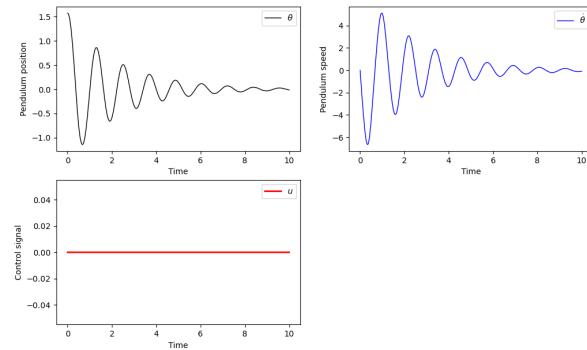


Figura 2.12: Gráficas de salida con respecto al tiempo del péndulo simple a $\theta = \frac{\pi}{2}$.

El péndulo liberado desde una posición diferente al equilibrio comienza con una oscilación y debido a que el sistema no se encuentra alimentado, busca detenerse hasta llegar al punto de equilibrio, en la figura 2.11 se observa al péndulo oscilando



libremente, y por efecto de la fricción se detendrá en el punto de equilibrio estático. En la figura 2.12 se puede ver la respuesta en el tiempo del péndulo, es decir, las gráficas de la posición y de velocidad. En la gráfica de posición se puede ver la oscilación libre del péndulo, y por efecto de la fricción el péndulo se comienza a detener hasta llegar a su posición de equilibrio, de igual manera la velocidad del péndulo oscila hasta reducir a 0.

La señal de control de u se mantiene en 0, pues en este momento aún no se le adapta un actuador y el péndulo oscila libremente.

Péndulo invertido

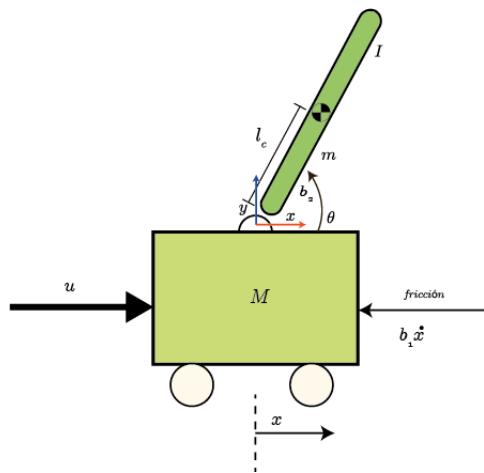


Figura 2.13: Péndulo invertido.

La posición de la barra se expresa de la siguiente manera:

$$p = \begin{bmatrix} l_c \cos \theta + x \\ l_c \sin \theta \end{bmatrix}$$



Derivando con respecto del tiempo, la velocidad es:

$$v = \frac{d}{dt} p = \dot{p} = \begin{bmatrix} -l_c \dot{\theta} \sin \theta + \dot{x} \\ l_c \dot{\theta} \cos \theta \end{bmatrix} = -l_c \dot{\theta} \sin \theta + l_c \dot{\theta} \cos \theta + \dot{x}$$

La energía cinética:

$$K = \frac{1}{2} (M + m) \dot{x}^2 + \frac{1}{2} (I + ml_c^2) \dot{\theta}^2 - ml_c \dot{x} \dot{\theta} \sin \theta$$

La energía potencial:

$$U = mg l \sin \theta$$

El lagrangiano que describe al sistema:

$$L = \frac{1}{2} (M + m) \dot{x}^2 + \frac{1}{2} (I + ml^2) \dot{\theta}^2 + ml \dot{x} \dot{\theta} \cos \theta + mg l \cos \theta$$

$$D = \frac{1}{2} b_1 \dot{x}^2 + \frac{1}{2} b_2 \dot{\theta}^2$$

La ecuación de movimiento de Euler-Lagrange:

$$\ddot{x} = \frac{u_1 - b_1 \dot{x} + ml_c \ddot{\theta} \sin \theta + ml_c \dot{\theta}^2 \cos \theta}{M + m}$$

$$\ddot{\theta} = \frac{u_2 - ml_c \ddot{x} \sin \theta - mg l_c \cos \theta - b_2 \dot{\theta}}{I + ml_c^2}$$

La representación en espacios de estados se puede expresar como:

Variables de estado: $x, \dot{x}, \theta, \dot{\theta}$

Vector de estado:

$$z = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix}$$

Ecuación de estado

$$\dot{z} = f(u, z, t)$$

$$= \begin{bmatrix} x \\ \theta \\ \ddot{x} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} x \\ \theta \\ M^{-1}(\mathbf{q})[\mathbf{u} - C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - g(\mathbf{q})] \\ ml_c \cos \theta \end{bmatrix} = \begin{bmatrix} x \\ \theta \\ \begin{bmatrix} M + m & ml_c \cos \theta \\ ml_c \cos \theta & I + ml_c^2 \end{bmatrix}^{-1} \left(\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} - \begin{bmatrix} b_1 & -ml_c \dot{\theta} \sin \theta \\ 0 & b_2 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{\theta} \end{bmatrix} - \begin{bmatrix} 0 \\ mg l_c \sin \theta \end{bmatrix} \right) \end{bmatrix}$$

La verificación se realizó en Python, a condiciones estáticas y condiciones fuera del equilibrio estático, esto con el fin de probar el comportamiento y verificación del modelo. A continuación, se presenta la operación a diferentes puntos de operación.

$$\theta = x = 0$$

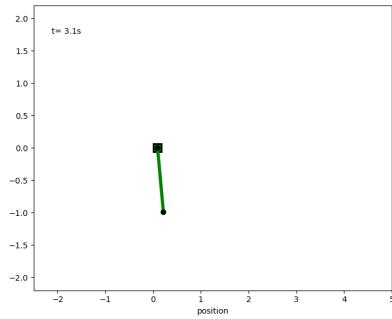


Figura 2.14: Posición del péndulo invertido en el plano XY.

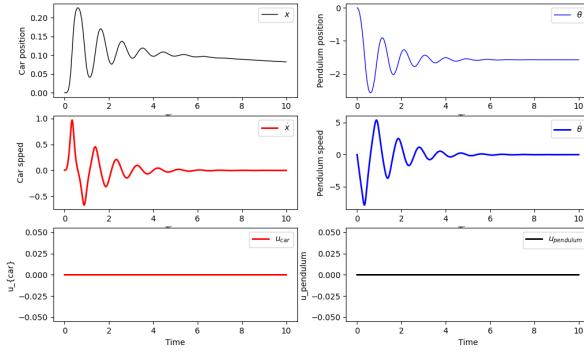


Figura 2.15: Gráficas de salida respecto al tiempo péndulo invertido a $\theta = x = 0$.

El péndulo invertido de la figura 2.14 se encuentra oscilando libremente, pues al no estar alimentado, al iniciar en un ángulo de 0, la posición del péndulo es sobre el eje positivo x , al soltarlo en una posición diferente la equilibrio en $\theta = 0$, el péndulo oscila, por efecto de la fricción el péndulo se detiene sobre la parte negativa del eje y , pues en esta posición se logra obtener el equilibrio estático, sin embargo, el carro mantiene oscilaciones muy pequeñas que tienden a desaparecer.

Lo descrito anteriormente se puede observar en la Figura 2.15. Se puede notar que el carrito no alcanza a detenerse, esto debido al tiempo de simulación, a diferencia del péndulo que se establece en la posición $\theta = \frac{-\pi}{2}$,

$$\theta = \frac{\pi}{2}, x = 0$$

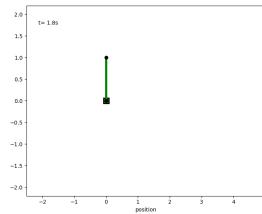


Figura 2.16: Posición en el plano XY del péndulo invertido.

Diseño del sistema

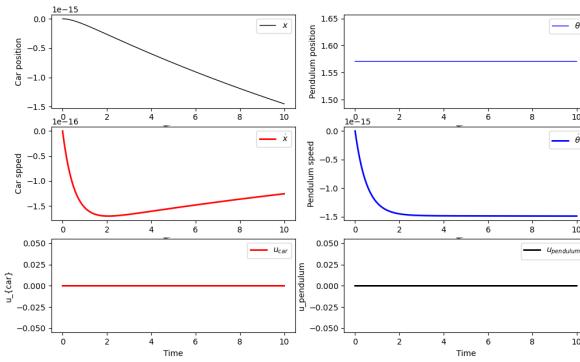


Figura 2.17: Gráficas de salida con respecto al tiempo del péndulo invertido a $\theta = \frac{\pi}{2}$, $x = 0$.

El comportamiento del péndulo invertido a condiciones iniciales $\theta = \frac{\pi}{2}$, $x = 0$ se muestra en la figura 2.16. El péndulo en este punto se encuentra en equilibrio y no presenta movimiento, en cambio el carrito presenta oscilaciones pequeñas.

El péndulo se mantiene estático sin alimentación, y el carrito muestra movimiento pequeños, en la gráfica de posiciones se pueden ver en la Figura 2.17, al igual que las señales de control, que se mantiene en 0, y al igual se puede ver en esta figura, las gráficas de velocidad con valores muy pequeños.



Péndulo doble

Es necesario comenzar a describir el movimiento obteniendo el Lagrangiano. Para la coordenada del centro de masa del brazo 1:

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} l_{c1} \sin \theta_1 \\ -l_{c1} \cos \theta_1 \end{bmatrix}$$

Para la coordenada del centro de masa del brazo 2:

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} l_1 \sin \theta_1 + l_{c2} \sin(\theta_1 + \theta_2) \\ -l_1 \cos \theta_1 - l_{c2} \cos(\theta_1 + \theta_2) \end{bmatrix}$$

La velocidad en ambas coordenadas es:

$$v_1 = \begin{bmatrix} l_{c1} \cos \theta_1 \\ l_{c1} \sin \theta_1 \end{bmatrix} \dot{\theta}_1$$

$$v_2 = \begin{bmatrix} l_1 \cos \theta_1 + l_{c2} \cos(\theta_1 + \theta_2) & l_{c2} \cos(\theta_1 + \theta_2) \\ l_1 \sin \theta_1 + l_{c2} \sin(\theta_1 + \theta_2) & l_{c2} \sin(\theta_1 + \theta_2) \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

La energía cinética total del sistema:

$$K = \frac{1}{2} \left(\dot{\theta}_1^2 (m_1 l_{c1}^2 + I_1 + I_2 + m_2 (l_1^2 + l_{c2}^2 + 2l_1 l_{c2} \cos(\theta_2))) + \right.$$

$$\left. \dot{\theta}_2^2 (m_2 l_{c2}^2 + I_2) + 2\dot{\theta}_1 \dot{\theta}_2 (m_2 l_1 l_{c2} \cos(\theta_2) + m_2 l_{c2}^2 + I_2) \right)$$

La energía potencial:

$$U = g(m_1 y_1 + m_2 y_2) = -g(m_1 l_{c1} \cos \theta_1 + m_2 (l_1 \cos \theta_1 + l_{c2} \cos(\theta_1 + \theta_2)))$$

A continuación, se presenta el Lagrangiano:

$$L = K - U = \frac{1}{2} \left(\dot{\theta}_1^2 (m_1 l_{c1}^2 + I_1 + I_2 + m_2 (l_1^2 + l_{c2}^2 + 2l_1 l_{c2} \cos(\theta_2))) + \right.$$

$$\left. \dot{\theta}_2^2 (m_2 l_{c2}^2 + I_2) + 2\dot{\theta}_1 \dot{\theta}_2 (m_2 l_1 l_{c2} \cos(\theta_2) + m_2 l_{c2}^2 + I_2) + g(m_1 l_{c1} \cos \theta_1 + m_2 (l_1 \cos \theta_1 + l_{c2} \cos(\theta_1 + \theta_2))) \right)$$

$$D = \frac{1}{2} \left(m_1 b_1 \dot{\theta}_1^2 + m_2 b_2 \dot{\theta}_2^2 \right)$$

La ecuación de movimiento de Euler-Lagrange:

$$-2m_2\dot{\theta}_1\dot{\theta}_2l_1l_{c2}\sin(\theta_2) - m_2\dot{\theta}_2^2l_1l_{c2}\sin(\theta_2) + \\ m_1b_1\dot{\theta}_1 + g(m_1l_{c1}\sin\theta_1 + m_2(l_1\sin\theta_1 + l_{c2}\sin(\theta_1 + \theta_2))) = \tau_1$$

$$\ddot{\theta}_2(m_2l_{c2}^2 + I_2) + \ddot{\theta}_1(m_2l_1l_{c2}\cos(\theta_2) + m_2l_{c2}^2 + I_2) + \\ m_2\dot{\theta}_1^2l_1l_{c2}\sin(\theta_2) + m_2b_2\dot{\theta}_2 + gm_2l_{c2}\sin(\theta_1 + \theta_2) = \tau_2$$

Agrupando los términos, se puede expresar la dinámica del péndulo doble en las siguientes matrices:

$$M = \begin{bmatrix} m_1l_{c1}^2 + I_1 + I_2 + m_2(l_1^2 + l_{c2}^2 + 2l_1l_{c2}\cos(\theta_2)) & m_2l_1l_{c2}\cos(\theta_2) + m_2l_{c2}^2 + I_2 \\ m_2l_1l_{c2}\cos(\theta_2) + m_2l_{c2}^2 + I_2 & m_2l_{c2}^2 + I_2 \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix}$$

$$C = \begin{bmatrix} -2m_2\dot{\theta}_1\dot{\theta}_2l_1l_{c2}\sin(\theta_2) + m_1b_1 & -\dot{\theta}_2m_2l_1l_{c2}\sin(\theta_2) \\ m_2\dot{\theta}_1l_1l_{c2}\sin(\theta_2) & m_2b_2 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

$$g = \begin{bmatrix} m_1l_{c1}\sin\theta_1 + m_2(l_1\sin\theta_1 + l_{c2}\sin(\theta_1 + \theta_2)) \\ m_2l_{c2}\sin(\theta_1 + \theta_2) \end{bmatrix}$$

$$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}$$

Modelo en espacio de estados:

Variables de estado: $(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2)$



Vector de estados:

$$x = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

La ecuación en espacios de estados:

$$\dot{x} = f(x, t, u)$$

$$\dot{x} = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ M^{-1}(\mathbf{u} - (C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + g(\mathbf{q}))) \end{bmatrix}$$

Se puede consultar el proceso detallado en el sección de Apéndice A.

El modelo del péndulo doble se simuló usando Python, ahora con ese modelo se simula en los siguientes puntos de condiciones iniciales, no está alimentado, es por eso que las señales de control para ambas gráficas es 0.

El código fuente de la simulación se puede consultar en la sección de Apéndice B.

$$\theta_1 = \theta_2 = 0$$

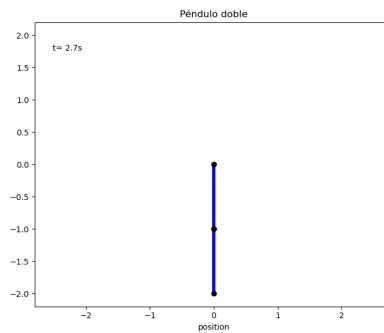


Figura 2.18: Posición en el plano XY (Péndulo doble).

Diseño del sistema

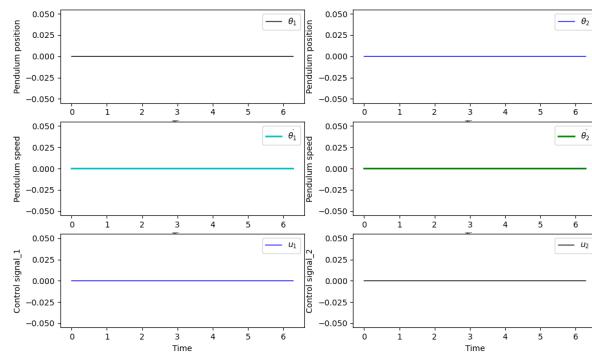


Figura 2.19: Gráficas de salida con respecto al tiempo del Péndulo doble a condiciones iniciales $\theta_1 = \theta_2 = 0$.

En la Figura 2.18 se muestra el comportamiento del péndulo en condiciones iniciales 0, al estar en un punto de equilibrio estático, el péndulo no muestra movimiento. Las gráficas de salida que se muestran en la Figura 2.19, todas se muestran en 0, pues al estar en equilibrio, el péndulo no se mueve, ambos brazos se mantienen en 0, al igual que las velocidades de los brazos se mantienen en 0.

$$\theta_1 = \frac{\pi}{2}, \theta_2 = 1$$

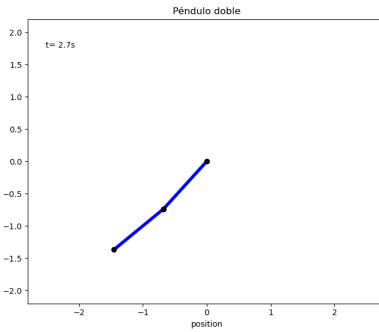


Figura 2.20: Posición en el plano XY (Péndulo doble).

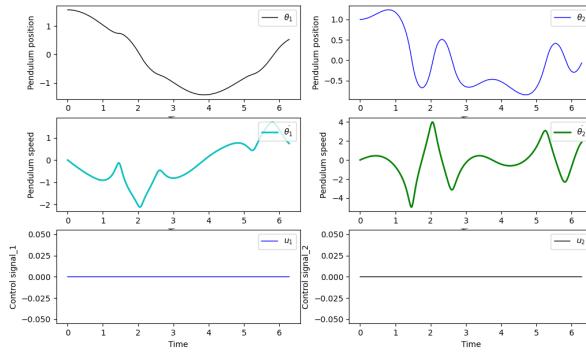


Figura 2.21: Gráficas de salida con respecto al tiempo del péndulo doble a condiciones iniciales $\theta_1 = \frac{\pi}{2}$, $\theta_2 = 1$.

El comportamiento mostrado en la Figura 2.21 es el péndulo oscilando libremente, ambos brazos oscilan libremente, este sistema es caótico, por lo que no llega a establecerse y tiende al desorden.

La animación del movimiento de los brazos del péndulo se puede ver en la Figura 2.20.

Selección de los indicadores de desempeño

Como se ha mencionado anteriormente, es importante seleccionar los indicadores de desempeño adecuados para el problema multiobjetivo, por lo que se hará uso de la metodología AHP, comúnmente utilizada para la toma de decisiones.

Indicadores basados en el error y la señal de control

En la tabla 2.12 se presentan tres indicadores de desempeño que son usados ampliamente en la literatura. Se compararon diferentes criterios que nos permitirán elegir la mejor opción.



TABLA 2.12: Variables y criterios.

	Supresión de errores	Implementación	Evaluación analítica
ISE	Buena	Fácil	Fácil
ITAE	Buena	Fácil	Difícil
IAE	Regular	Muy Fácil	Difícil

Tenemos como resultado la tabla 2.13

TABLA 2.13: Matriz de comparación de criterios.

Criterios	Supresión de errores	Facilidad de implementación	Facilidad Evaluación analítica	Matriz normalizada			Ponderación
Supresión de errores	1	7	5	0.7447	0.8537	0.4545	0.6843
Facilidad de implementación	0.14285714	1	5	0.1064	0.1220	0.4545	0.2276
Facilidad Evaluación analítica	0.2	0.2	1	0.1489	0.0244	0.0909	0.0881
Total	1.34285714	8.2	11				

Posteriormente, como lo indica la meteodología, en las tablas 2.14, 2.15 y 2.16 se comparan entre sí las opciones a elegir con cada uno de los criterios como eje central.

TABLA 2.14: Criterio: Supresión de errores.

Alternativas	ISE	ITAE	IAE	Matriz normalizada			Vector promedio
ISE	1	3	7	0.6774	0.6923	0.6364	0.6687
ITAE	0.33333333	1	3	0.2258	0.2308	0.2727	0.2431
IAE	0.14285714	0.33333333	1	0.0968	0.0769	0.0909	0.0882
Total	1.47619048	4.33333333	11				



TABLA 2.15: Criterio: Implementación digital.

Alternativas	ISE	ITAE	IAE	Matriz normalizada			Vector promedio
ISE	1	1	0.2	0.1429	0.4545	0.0323	0.2099
ITAE	1	1	5	0.1429	0.4545	0.8065	0.4680
IAE	5	0.2	1	0.7143	0.0909	0.1613	0.3222
Total	7	2.2	6.2				

TABLA 2.16: Criterio: Evaluación analítica.

Alternativas	ISE	ITAE	IAE	Matriz normalizada			Vector promedio
ISE	1	6	5	0.7317	0.7500	0.7143	0.7320
ITAE	0.16666667	1	1	0.1220	0.1250	0.1429	0.1299
IAE	0.2	1	1	0.1463	0.1250	0.1429	0.1381
Total	1.36666667	8	7				

Por último, en la tabla 2.17 se colocan los eigenvectores de cada criterio correspondientes a cada alternativa, mostrando que la opción con un peso de priorización mayor, es el ISE.

TABLA 2.17: Selección de índice de desempeño.

Criterio/Alternativa	Supresión de errores	Facilidad de implementación	Facilidad Evaluación analítica	Priorización
ISE	0.66869689	0.20988689	0.73199768	0.56983499
ITAE	0.24310099	0.4679514	0.12993612	0.28431552
IAE	0.08820212	0.32216171	0.1380662	0.14584949
Ponderación	0.68429495	0.22762655	0.0880785	

Por otra parte, en varios artículos [13, 21], el índice comúnmente usado para medir la suavidad de la señal de control, es el IADU, por lo que representa una buena elección como segunda función objetivo.



Selección de la estructura del controlador

Péndulo simple e invertido

Como propuesta solución, para estos péndulos se seleccionó la estructura PID, debido a su amplio uso en la literatura consultada en los antecedentes. Es importante mencionar que esta estructura se aplicará a cada actuador presente en estos sistemas.

Péndulo doble

En [32] se menciona que para el control de trayectoria uno de los controladores que presentan mejor desempeño en la resolución de este problema es la estructura PD+. Es por este motivo que se seleccionó el controlador PD+ como estructura de control para el péndulo doble.

Definición del problema multiobjetivo

La mayoría de los problemas de diseño de control PID son inherentemente problemas multiobjetivo, ya que naturalmente existen distintos objetivos que entran en conflicto al intentar satisfacerse simultáneamente dentro de las restricciones establecidas. Como se mencionó anteriormente, se propone usar como funciones objetivo los índices de rendimiento ISE e IADU, con el objetivo de encontrar un conjunto de compromisos entre la disminución del error y la suavidad de la señal.

Péndulo simple

De acuerdo con la ecuación 1.4, podemos definir el problema multiobjetivo como se muestra a continuación:

$$\min \mathbf{J}(\mathbf{x}) = [J_{ISE}(\mathbf{x}), J_{IADU}(\mathbf{x})]$$

Donde

$$\mathbf{x} = [K_p, K_d, K_i]$$



Sujeto a

$$0.001 \leq x_i \leq 10 \quad i = [1, 2, 3]$$

Restricciones

$$J_{ISE} \leq 0.1$$

$$J_{IADU} \leq 0.8$$

$$u(t) = \begin{cases} u(t) = 1.4 \text{ Nm} & \text{si } 1.4 \text{ Nm} \leq u(t) \\ u(t) = u(t) & \text{si } 3.5 \text{ mNm} \leq u(t) \leq 1.4 \text{ Nm} \\ u(t) = 3.5 \text{ mNm} & \text{si } u(t) \leq 3.5 \text{ mNm} \end{cases}$$

Se define J_{ISE} como $\int_0^T e^2(t)dt$ donde $e(t)$ es la diferencia entre la posición deseada y la posición de salida del sistema, es decir, $e(t) = q_d - q$.

J_{IADU} se puede definir de la forma $\int_0^T |u(t) - u(t-1)|dt$ donde u es la señal de control. En otras palabras, este indicador calcula la diferencia de la señal de control actual con respecto a la señal de control de un instante de tiempo anterior, con esto, lo que se medirá serán las variaciones de la señal de control y evitar variaciones abruptas para todo instante del tiempo.

Los límites de las variables de diseño K_i, K_d, K_i se establecieron con base en experimentos con el modelo ya establecido en el lenguaje Python, dando como resultado un buen comportamiento con ganancias menores a 10 y los valores de estas variables deben de estar definidas positivamente, es decir, deben de ser mayores a 0.

Los límites que se establecieron en J_{ISE} y J_{IADU} están en función de los resultados obtenidos en los artículos de referencia de los antecedentes, y con base en los requerimientos, donde se establece que se debe de minimizar el error y maximizar la suavidad de la señal.

Por último, el rango de valores de la señal de control u , se estableció tomando en

cuenta el torque que proporcionan los diferentes actuadores de DC comerciales, como lo son el motor DC, motor a pasos y el servomotor.

Péndulo invertido:

De forma similar, se establece el problema multiobjetivo para el péndulo invertido.

$$\min \mathbf{J}(\mathbf{x}) = [J_{ISE}(\mathbf{x}), J_{IADU}(\mathbf{x})]$$

Donde

$$\mathbf{x} = [K_{p_j}, K_{d_j}, K_{i_j}] \quad j = 1, 2$$

Sujeto a

$$0.001 \leq x_i \leq 20 \quad i = [1, 2, 3, 4, 5, 6]$$

Restricciones

$$J_{ISE} \leq 0.1$$

$$J_{IADU} \leq 0.8$$

Péndulo doble:

Por último, el problema multiobjetivo para el péndulo doble de establece a continuación.

$$\min \mathbf{J}(\mathbf{x}) = [J_{ISE}(\mathbf{x}), J_{IADU}(\mathbf{x})]$$

Donde

$$\mathbf{x} = [K_{p_j}, K_{d_j}] \quad j = 1, 2$$

Sujeto a

$$0.001 \leq x_i \leq 20 \quad i = [1, 2, 3, 4]$$



Selección de técnicas estadísticas para evaluar el desempeño de los algoritmos metaheurísticos.

La evaluación de los algoritmos metaheurísticos se realiza con indicadores de rendimiento, debido a su uso en la literatura el indicador seleccionado es el Hiper-volumen.

Los algoritmos al componerse de variables aleatorias son conocidos también como algoritmos estocásticos, debido a esto, la distribución sus resultados no siguen una distribución normal. Afortunadamente existen la probabilidad no paramétrica, estas pruebas pueden tratar los datos que no siguen una distribución normal.

Una prueba que se ajusta a las necesidades de este proyecto es la prueba de Wilcoxon, pues se usa para datos que muestran una relación entre sí e indican que tan diferentes son las muestras. Pero para saber realmente que algoritmo presenta mejor desempeño es necesario una prueba de signo.

La solución propuesta para evaluar y comparar el desempeño de los algoritmos es la prueba de rangos con signo de Wilcoxon, pues es ampliamente usada en la literatura consultada.

Validación

Problema de optimización

Para la validación del problema de optimización, se usará la librería *pymoo* de Python, esta librería permite resolver problemas de optimización con algoritmos evolutivos, tanto para problemas de un solo objetivo como para multiobjetivo. Cuenta con varios algoritmos para la optimización multiobjetivo, como lo es NSGA-II. Con este algoritmo se trabajarán las pruebas de validación del planteamiento del problema.

Solo se probarán el planteamiento del péndulo simple con un ajuste en la restricción de la señal de control u , pues el torque máximo que el motor ofrece es superior al

establecido, además se tomará en cuenta el signo de la señal, pues indica la dirección de giro del motor, por lo que esta restricción se plantea de la siguiente manera:

$$u(t) = \begin{cases} u(t) = 2.94 \text{ Nm} & \text{si } 2.94 \text{ Nm} \leq u(t) \\ u(t) = u(t) \text{ Nm} & \text{si } 0 \text{ Nm} \leq u(t) \text{ Nm} < 2.94 \text{ Nm} \\ u(t) = u(t) \text{ Nm} & \text{si } -2.94 \leq u(t) < 0 \text{ mNm} \\ u(t) = -2.94 \text{ mNm} & \text{si } u(t) < -2.94 \text{ Nm} \end{cases}$$

A continuación, se procede a la validación de las restricciones de los valores mínimos de ISE e IADU, de la misma forma, se procede a validar cada límite con el uso de la librería *pymoo*, para eso, en el diagrama de frente de Pareto se tiene como salida ISE vs ISE e IADU vs IADU, sin tomar en cuenta los valores establecidos en el planteamiento del problema, esto con el fin de verificar el valor mínimo de ambos objetivos.

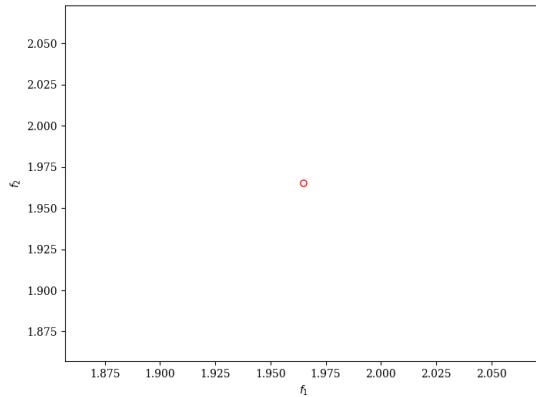


Figura 2.22: ISE vs ISE.

En la figura 2.22 se muestra el valor mínimo de ISE que se puede obtener usando algoritmos evolutivos, este valor es de 1.96 aproximadamente, y para no forzar al algoritmo el nuevo límite establecido será de 20.

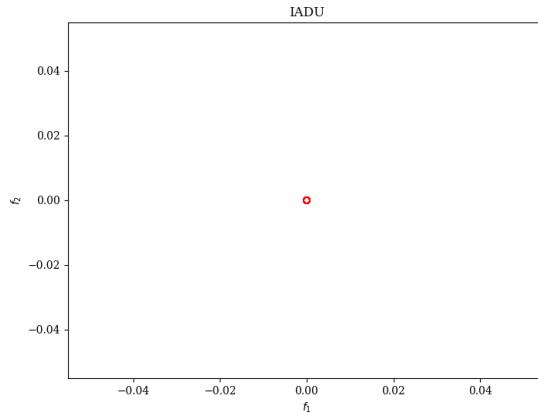


Figura 2.23: IADU vs IADU.

En la figura 2.23 se muestra el valor mínimo de IADU que se puede obtener usando algoritmos evolutivos, este valor es muy próximo a 0, por lo que el nuevo límite se mantiene en 0.8.

Reescribiendo el problema de optimización:

Péndulo simple

$$\min \mathbf{J}(\mathbf{x}) = [J_{ISE}(\mathbf{x}), J_{IADU}(\mathbf{x})]$$

Donde

$$\mathbf{x} = [K_p, K_d, K_i]$$

Sujeto a

$$0.001 \leq x_i \leq 10 \quad i = [1, 2, 3]$$

Restricciones

$$J_{ISE} \leq 20$$

$$J_{IADU} \leq 0.8$$

$$u(t) = \begin{cases} u(t) = 2.94 \text{ Nm} & \text{si } 2.94 \text{ Nm} \leq u(t) \\ u(t) = u(t) \text{ Nm} & \text{si } 0 \text{ Nm} \leq u(t) \text{ Nm} < 2.94 \text{ Nm} \\ u(t) = u(t) \text{ Nm} & \text{si } -2.94 \leq u(t) < 0 \text{ mNm} \\ u(t) = -2.94 \text{ mNm} & \text{si } u(t) < -2.94 \text{ Nm} \end{cases}$$

Péndulo invertido

$$\min \mathbf{J}(\mathbf{x}) = [J_{ISE}(\mathbf{x}), J_{IADU}(\mathbf{x})]$$

Donde

$$\mathbf{x} = [K_{p_j}, K_{d_j}, K_{i_j}] \quad j = 1, 2$$

Sujeto a

$$0.001 \leq x_i \leq 20 \quad i = [1, 2, 3, 4, 5, 6]$$

Restricciones

$$J_{ISE} \leq 20$$

$$J_{IADU} \leq 0.8$$

Péndulo doble

$$\min \mathbf{J}(\mathbf{x}) = [J_{ISE}(\mathbf{x}), J_{IADU}(\mathbf{x})]$$

Donde

$$\mathbf{x} = [K_{p_j}, K_{d_j}] \quad j = 1, 2$$

Sujeto a

$$0.001 \leq x_i \leq 20 \quad i = [1, 2, 3, 4]$$



2.2. Diseño detallado

Modelado del sistema.

El modelado del sistema se realizó con las ecuaciones de Euler-Lagrange para obtener el modelo dinámico del sistema, para posteriormente expresarlo en matrices de Inercia, Coriolis, y gravedad, por último, se realizó un cambio de variable para obtener las ecuaciones de estado del sistema, el cual permite expresar el modelo dinámico obtenido en un sistema de primer orden para su fácil resolución.

Las simulaciones del modelo dinámico de cada uno de los péndulos se realizaron en Python, donde se presenta una simulación 2D del movimiento de cada uno de los péndulos, así como las gráficas de velocidad, posición y señal de control de cada péndulo. Todo lo mencionado anteriormente se presento en la sección de apéndice A.

Sintonización con métodos metaheurísticos para optimización multiobjetivo.

La sintonización de los sistemas propuestos se realizará con DE, PSO Y GA, como se plateó en el apéndice A, el problema de optimización y su validación son correctas y muestran un buen resultado.

Interfaz Gráfica (GUI).

La GUI (Graphical User Interface) es una interfaz entre la persona y la máquina. El objetivo de una interfaz gráfica es representar el código programado de un sistema de una forma más clara y entendible para el usuario, con el propósito de simplificarle las tareas a realizar. Utilizan elementos gráficos como iconos, menús e imágenes para facilitar el manejo del usuario humano. Tanto los sistemas operativos como las aplicaciones utilizan una interfaz gráfica de usuario.



Para este proyecto, se propone la elaboración de una GUI en el ambiente de Python, mostrando en una pantalla de inicio los tres sistemas propuestos inicialmente (péndulo simple, péndulo invertido y péndulo doble). Posteriormente, dependiendo del sistema seleccionado, se solicita el ingreso de datos o variables del mismo, si los datos son correctos, se solicita seleccionar el método metaheurístico a ejecutar. Una vez realizado esto, se muestra el Frente de Pareto resultante de la ejecución del programa, así como los valores de cada punto en la gráfica. Se pretende que se pueda seleccionar un conjunto de datos para realizar la simulación dinámica del mecanismo y visualizar las gráficas obtenidas. El proceso descrito se muestra en un diagrama de flujo mostrado en la Figura 2.24. Por otra parte, en la sección Anexos, las Figuras 9, 10, 11, 12, 13, 14, 15, y 16, muestran un bosquejo de la interfaz en ejecución.



miro

Figura 2.24: Diagrama de flujo de la GUI



Búsqueda de opciones de bibliotecas.

- Tkinter

El paquete tkinter («interfaz Tk») es la interfaz estándar de Python para el kit de herramientas GUI de Tcl / Tk. Tanto Tk como tkinter están disponibles en la mayoría de las plataformas Unix, incluido macOS, así como en sistemas Windows.

- PyQt

Es uno de los modulos más utilizados en la creación de aplicaciones con interfaces gráficas en Python. Es desarrollado por Riverbank Computing Limited. PyQt proporciona enlaces para Qt4 y Qt5.

PyQt está disponible en dos ediciones: PyQt4, que se compilará contra Qt 4.xy 5.xy PyQt5, que solo se compilará contra 5.x. Ambas ediciones se pueden construir para Python 2 y 3. PyQt contiene más de 620 clases que cubren interfaces gráficas de usuario, manejo de XML, comunicación de red, bases de datos SQL, navegación web y otras tecnologías disponibles en Qt.

- PySimpleGUI

PySimpleGUI es un módulo para el diseño y creación de interfaces gráficas en Python. Es un conector de la API de Python con el módulo Tkinter. Permite programar los mismos elementos de la interfaz de usuario desarrollada con Tkinter, pero con una interfaz más intuitiva.

Permite desarrollar de manera mas sencilla interfaces gráficas, además de que agrega la capacidad de personalizar diseños.

Análisis de ventajas y desventajas.

- Tkinter



Ventajas.

- Disponible para sistemas operativos macOS, Windows, y plataformas Unix.
- Preinstalado en el sistema operativo Windows.
- Relativamente fácil de aprender.

Desventajas.

- Lento.

■ **PyQt**

Ventajas.

- PyQt4 se ejecuta en Windows, Linux, Mac OS X y varias plataformas UNIX. PyQt5 también se ejecuta en Android e iOS.
- Elementos gráficos completos, pues cuenta con más de 620 clases para interfaces gráficas.
- Manejo de XML, comunicación de red, bases de datos SQL, navegación web y otras tecnologías disponibles.

Desventajas.

- Más complejo de aprender.
- En ocasiones emerge la implementación en C++ subyacente, teniendo que hacer casts entre tipos de datos, etc.

■ **PySimpleGUI**

Ventajas.

- Utiliza los mismos controles que los de tkinter, Qt, WxPython y Remi.
- El código se reduce entre un 50 % y un 90 %.



- Tiene una interfaz amigable.
- Mejor interfaz.

Desventajas.

- Trata las ventanas como filas de elementos de la GUI.

Selección de biblioteca.

Por la facilidad y la optimización de código que se menciono en la sección anterior, la biblioteca que presenta mayores beneficios es PySimpleGUI, pues se pueden implementar los elementos de otras bibliotecas, y visualmente el resultado es más estético.



Adaptación del péndulo simple.

En la Figura 2.25 se muestra una matriz con los materiales propuestos para ser empleados en la construcción del mecanismo establecido en la Figura 2.6.

	Opciones		
Elemento	1	2	3
A) PÉNDULO	Madera	Aluminio	Nylamid
B) EJE	Acero inoxidable	Aluminio	Acero al carbono
C) BLOQUE	ABS	Nylamid	PLA
D) ADQUISICIÓN DE DATOS	Teensy 4.1	Arduino UNO	Raspberry Pi

Figura 2.25: Selección de materiales

Búsqueda de opciones en la selección de materiales

- Opción 1: A1-B2-C1-D2
- Opción 2: A2-B1-C2-D1
- Opción 3: A3-B3-C3-D3

Análisis de ventajas y desventajas

Con el propósito de comparar las opciones entre sí y seleccionar la más adecuada, se realizó una tabla de ventajas y desventajas, misma que se observa en la Tabla 2.18.

Opción	Ventajas	Desventajas
1	<ul style="list-style-type: none"> -Esta opción se caracteriza por su ligereza, ya que los materiales a considerar son menos pesados a comparación de las otras opciones. -Arduino Uno es una placa de microcontrolador de uso fácil, además, es de bajo costo y es altamente flexible. Otra de las ventajas de usar esta placa es que es multiplataforma, ya que el software de Arduino funciona en los sistemas operativos Windows, Macintosh OSX y Linux. 	<ul style="list-style-type: none"> -El material más usado para ejes es el acero. -El precio a pagar por el uso de las librerías es un retraso en la ejecución de las instrucciones, algunos microsegundos que en el caso de dispositivos de uso cotidiano son irrelevantes, pero significativos a la hora de hacer adquisición de datos.
2	<ul style="list-style-type: none"> -El aluminio es un material con una buena relación peso/resistencia. -El nylamid tiene una mayor durabilidad que los otros materiales a considerar. - La Teensy 4.1 posee un microcontrolador de 32 bits corriendo a 600MHz, al igual que algunas placas de Arduino, pero con un rendimiento muy superior. -Gracias al plug-in para el IDE de Arduino "Teensyduino", es posible utilizar no solo el IDE de Arduino, sino también la mayoría de librerías del entorno del mismo. 	<ul style="list-style-type: none"> - El nylamid presenta menor resistencia y rigidez que el PLA y ABS. -El precio de la placa Teensy es más elevado que el de una placa Arduino Uno.
3	<ul style="list-style-type: none"> -El nylamid es un material ligero a comparación de los otros materiales propuestos. -El acero es el material más usado en ejes. -El PLA es un termoplástico fácil de usar con mayor resistencia y rigidez que el ABS y el nylon. 	<ul style="list-style-type: none"> - La relación entre las especificaciones de la Raspberry Pi y los requerimientos del sistema es mayor de lo que se necesita.

TABLA 2.18: Análisis de ventajas y desventajas en la selección de materiales.



Selección de opción de materiales

A pesar de que la opción 1 es la más económica, fue descartada ya que la precisión en la adquisición de datos que conlleva usar la placa de Arduino UNO, es baja comparada con las otras opciones. De manera similar y como se menciona en la Tabla 2.18, en el caso de la opción 3, la relación entre las especificaciones de la Raspberry Pi y los requerimientos del sistema es mayor de lo que se necesita, además de que la adquisición de la Raspberry Pi representa un costo elevado. El punto de equilibrio entre las opciones ya mencionadas, está dado por la opción 2, la cual representa la mejor opción, ya que la Teensy 4.1 posee las especificaciones necesarias para el proyecto. Por otra parte, los materiales del eje, el péndulo y el bloque son similares a las otras opciones. En conclusión, se elige la **opción 2** debido a la precisión en la tarjeta con microcontrolador y a la resistencia y economía que conllevan los demás elementos.

Como se ha mencionado en secciones anteriores, el objetivo de los algoritmos evolutivos desarrollados en este trabajo, es encontrar un conjunto de soluciones (ganancias) que puedan implementarse en un control PID, para controlar la posición de un péndulo simple. Es por ello que en la figura 2.26 se presenta una descripción del sistema integrado.

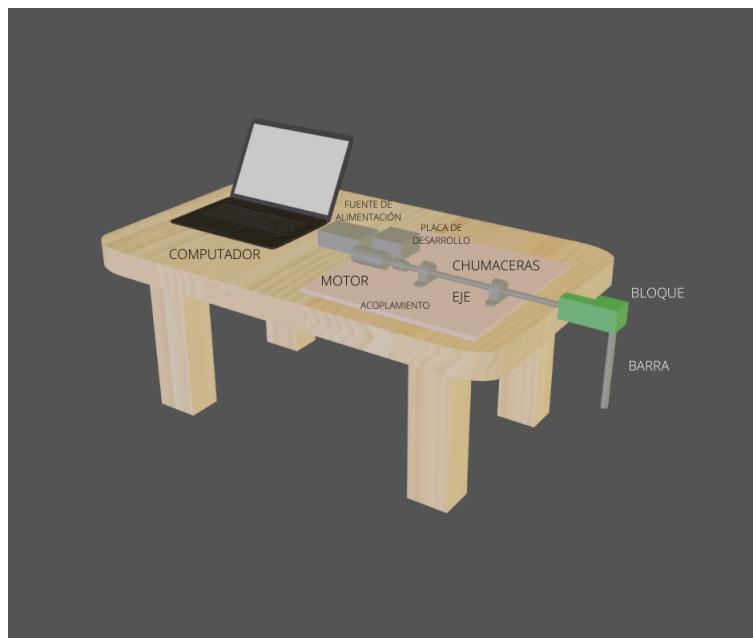


Figura 2.26: Sistema péndulo simple.

Los elementos que conforman el sistema se muestran a continuación:

Motor: Como se ha mencionado con anterioridad, el motor es pieza fundamental y será el encargado de suministrar el torque. Una aproximación a lo que sería el motor se muestra en la Figura 2.10.



Figura 2.27: Motor.



Eje: El elemento mostrado en la figura 2.28 será el encargado de transmitir el par del motor hacia el péndulo. Se busca que sea ligero y firme al mismo tiempo, por lo que una buena opción es que sea de aluminio 6061 con un diámetro de 8mm. El plano de esta pieza se muestra en la figura 5, en la sección Anexos.



Figura 2.28: Eje.

Chumacera: Con el propósito de darle estabilidad al eje, se ocuparán dos chumaceras de piso comerciales con rodamientos de 8 mm para eje lineal, como la que se muestra en la figura 2.29. Estos elementos cuentan con tornillos dentro del báculo para sujetar la barra para tolerar vibraciones o movimiento sin dañar el eje, manteniendo la precisión del mismo.



Figura 2.29: Chumacera

Acomplamientos: Para poder acoplar el motor al eje, se propone usar un cople



flexible de aluminio de 8 mm x 8 mm, como el que se muestra en la figura 2.30



Figura 2.30: Cople

Elemento de sujeción: Se utilizará una pieza impresa en 3D, que servirá como conexión entre el eje y el brazo que fungirá como péndulo. Se muestra en la figura 2.31. El plano de esta figura se muestra en la figura 8, en la sección Anexos.

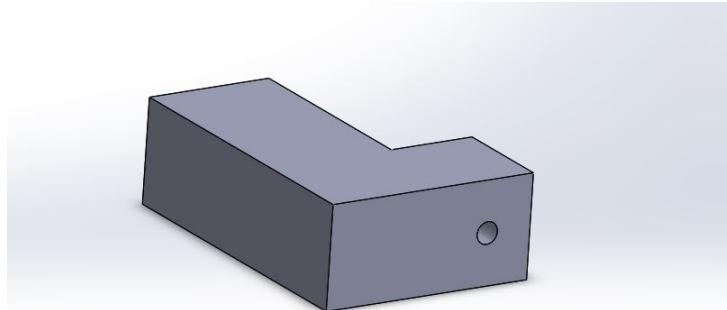


Figura 2.31: Elemento de sujeción.

Brazo: Elemento que nos servirá de referencia para visualizar la posición que se haya programado con el control PID. Lo podemos ver en la figura 2.32.

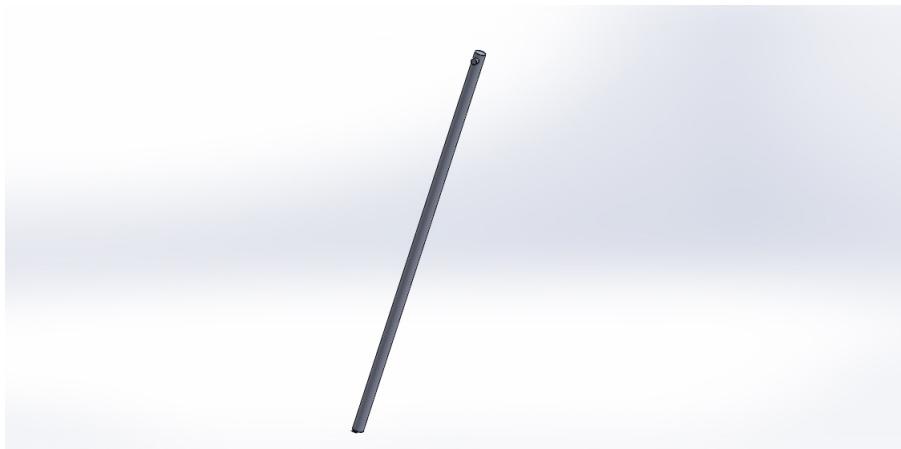


Figura 2.32: Brazo de aluminio.

Movimiento y sensor de posición

Debido a la construcción del sistema al cual se desea adaptar, se busca un motor de corriente directa que cuente con un sensor de movimiento. Los motores comerciales que cuenta con un sensor de movimiento acoplado son los motorreductores, y motores a pasos. Se requiere de un motor con sensor de movimiento, un driver que permita poder manipular el motor con el microcontrolador sin dañar algún componente. Las propuestas a considerar son las que se presentan a continuación:

- Motorreductor con Enconder 12V JGA25-370 con puente H.

Alimentación: 6-12 V.

Torque: 0.98 Nm.

Velocidad: 12 RPM.

Resolución del encoder: 5500 PPR.

Driver: Puente H L298.

Costo total: 795 MXN.

- Azssmuk Motor de engranaje de metal.



Alimentación: 12 V.
Velocidad: 300 RPM.
Torque: 0.78 Nm.
Resolución del encoder: 240 PPR.
Driver: Puente H L298.
Costo total: 593 MXN.

■ Rtelligent-Motor paso a paso Nema 23

Alimentación: 24-50 V.
Torque: 1 Nm.
Velocidad: 285 RPM.
Resolución del encoder: 1000.
Driver: Rtelligent T60.
Costo total: 1769 MXN.

■ Motorreductor con engranes planetarios y encoder.

Alimentación: 6-12 V.
Torque: 2.94 Nm.
Velocidad: 285 RPM.
Resolución del encoder: 250 PPR.
Driver: Puente H L298.
Costo total: 1053 MXN.



Los criterios para considerar son:

Resolución.

Torque.

Precio.

Consumo energético.

TABLA 2.19: Matriz de comparación de criterios

Criterios	Resolución	Torque	Precio	Consumo energético	Matriz normalizada					Ponderación
Resolución	1	0.33333333	7	5	0.230263158	0.21	0.35	0.375	0.291315789	
Torque	3	1	9	7	0.690789474	0.63	0.45	0.525	0.573947368	
Precio	0.14285714	0.11111111	1	0.33333333	0.032894737	0.07	0.05	0.025	0.044473684	
Consumo energético	0.2	0.14285714	3	1	0.046052632	0.09	0.15	0.075	0.090263158	
Total	4.34285714	1.58730159	20	13.33333333						

Las opciones se identificarán de la siguiente manera:

- A.Motorreductor con Enconder 12V JGA25-370 con puente H.
- B.Azsmuk Motor de engranaje de metal.
- C.Rtelligent-Motor paso a paso Nema 23.
- D.Motorreductor con engranes planetarios y encoder.

En las tablas 2.20, 2.21, 2.22 y 2.23 se comparan entre sí las opciones a elegir con cada uno de los criterios como eje central.

Las ponderaciones de los criterios se realizaron en la tabla 2.19, donde se establece el nivel de importancia que tiene cada criterio en la selección del motor y encoder.



TABLA 2.20: Criterio: Resolución.

Opciones	A	B	C	D	Matriz normalizada				Ponderación
A	1	5	3	5	0.577	0.5	0.643	0.5	0.554945055
B	0.2	1	0.333	1	0.115	0.1	0.071	0.1	0.096703297
C	0.333	3	1	3	0.192	0.3	0.214	0.3	0.251648352
D	0.2	1	0.333	1	0.115	0.1	0.071	0.1	0.096703297
Total	1.733	10	4.667	10					

TABLA 2.21: Criterio:Torque

Opciones	A	B	C	D	Matriz normalizada				Ponderación
A	1	3	0.333	0.143	0.088	0.167	0.051	0.098	0.101043911
B	0.333	1	0.2	0.111	0.029	0.056	0.031	0.076	0.047999695
C	3	5	1	0.2	0.265	0.278	0.153	0.138	0.208274867
D	7	9	5	1	0.618	0.5	0.765	0.688	0.642681527
Total	11.33	18	6.533	1.454					

TABLA 2.22: Criterio:Precio

Opciones	A	B	C	D	Matriz normalizada				Ponderación
A	1	0.333	7	3	0.223	0.203	0.35	0.321	0.274383882
B	3	1	9	5	0.67	0.608	0.45	0.536	0.56600879
C	0.143	0.111	1	0.333	0.032	0.068	0.05	0.036	0.046299187
D	0.333	0.2	3	1	0.074	0.122	0.15	0.107	0.113308141
Total	4.476	1.644	20	9.333					

TABLA 2.23: Criterio: Consumo energético

Opciones	A	B	C	D	Matriz normalizada				Ponderación
A	1	3	7	1	0.404	0.417	0.35	0.404	0.393589744
B	0.333	1	5	0.333	0.135	0.139	0.25	0.135	0.164529915
C	0.143	0.2	1	0.143	0.058	0.028	0.05	0.058	0.048290598
D	1	3	7	1	0.404	0.417	0.35	0.404	0.393589744
Total	2.476	7.2	20	2.476					



TABLA 2.24: Selección de motor con encoder

Criterios/ Opciones	Resolución	Torque	Precio	Consumo energético	Priorización
A	0.55494505	0.10104391	0.27438388	0.39358974	0.267387659
B	0.0967033	0.04799969	0.56600879	0.16452991	0.095743982
C	0.25164835	0.20827487	0.04629919	0.0482906	0.199265908
D	0.0967033	0.64268153	0.11330814	0.39358974	0.437602452
Ponderación	0.29131579	0.57394737	0.04447368	0.09026316	

En la tabla 2.24 se muestra el resultado final, donde se puede que la opción mejor evaluada es la opción D, pues ofrece una buena resolución en el encoder y además ofrece un torque de a 2.94 Nm.



Figura 2.33: Motor con encoderFuente: Modificado de [1].

El motor mostrado en la figura 2.33 es el motor seleccionado, es un motorreductor con relación de velocidad de 14:1, con una velocidad de 285 RPM, el encoder, es un encoder de efecto óptico, tiene una resolución de 1000 CPR, por último, puede operar con niveles de tensión de 6-12 V.

Microcontrolador

La tarjeta seleccionada para este proyecto es la tarjeta Teensy 4.1, pues tiene una buena capacidad de procesamiento, además de su facilidad de programar, en la figura 2.35, se puede observar esta tarjeta.

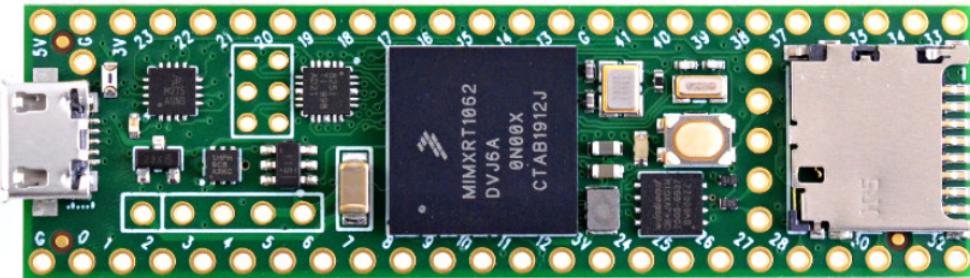


Figura 2.34: Teensy 4.1.

■ Características

- Cortex-M7 a 600 MHz.
- Unidad matemática de punto flotante, 64 y 32 bits.
- 7936K Flash, 1024K RAM (512K estrechamente acoplado), 4K EEPROM (emulado).
- Expansión de memoria QSPI, ubicaciones para 2 chips adicionales de RAM o Flash.
- Dispositivo USB 480 Mbit / seg y host USB 480 Mbit / seg.
- 55 pines de entrada / salida digital, 35 pines de salida PWM.
- 18 pines de entrada analógica.
- 8 puertos seriales, 3 SPI, 3 I2C.
- 2 puertos de audio digital I2S / TDM y 1 S / PDIF.
- 3 CAN Bus (1 con CAN FD)
- 1 puerto de tarjeta SD nativa SDIO (4 bits).
- Ethernet 10/100 Mbit con DP83825 PHY.
- 32 canales DMA de uso general.



- Aceleración criptográfica y generador de números aleatorios.
- RTC para fecha / hora
- FlexIO programable.
- Canalización de procesamiento de píxeles.
- Disparo cruzado periférico.
- Gestión de encendido / apagado.
- Software
 - Arduino IDE + Teensyduino.
 - Visual Micro.
 - PlatformIO.
 - CircuitPython.
 - Command Line with Makefile.
- Unidad de punto flotante (FPU por sus siglas en inglés)
 - La FPU realiza operaciones matemáticas flotantes de 32 bits y de doble precisión de 64 bits en hardware. La velocidad flotante de 32 bits es aproximadamente la misma velocidad que la matemática de números enteros. La precisión doble de 64 bits se ejecuta a la mitad de la velocidad de la flotante de 32 bits.
- Configuración de pines

Diseño del sistema

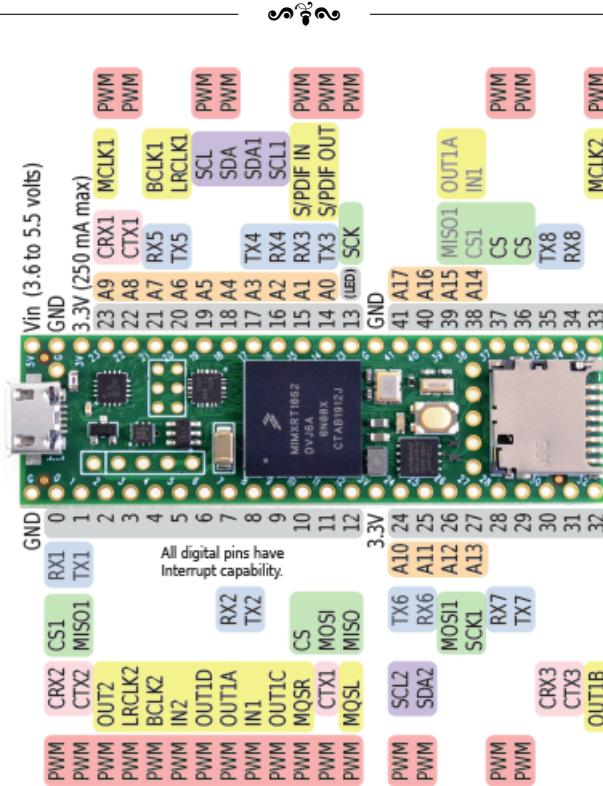


Figura 2.35: Configuración de pines.

La lógica es de 3.3 V, puede recibir señales de 0 a 3.3 V en los pines de entradas digitales, de igual manera su salida en alto es de 3.3 V.

Análogamente los pines analógicos solo leen de 0 a 3.3 V y cuenta con una resolución de 10 bits (rangos de entrada de 0 a 1023).

No tolera tensiones mayores a 3.3 V.



Control de péndulo simple

Circuito de control

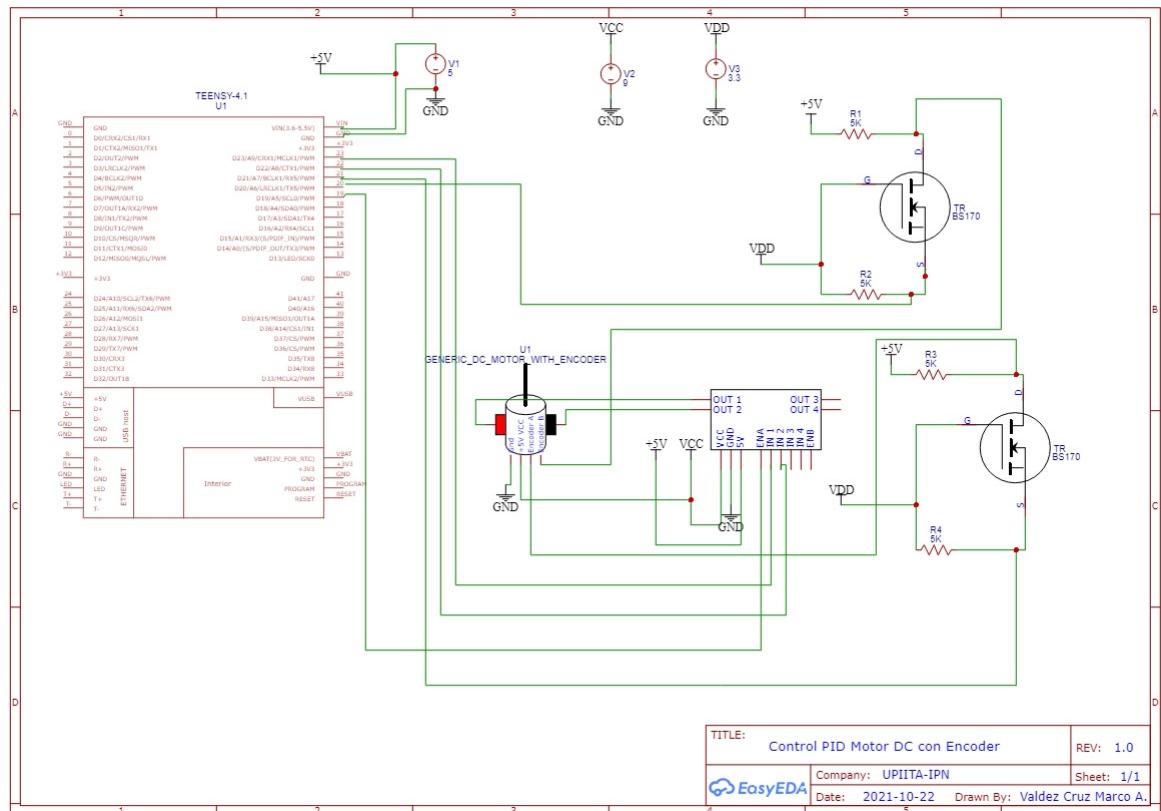


Figura 2.36: Circuito de control.

En la figura 2.36 se muestra el circuito de control completo, contempla:

Teensy 4.1

Es una tarjeta de desarrollo con una memoria Flash de 8 Mbyte, cuenta con 55 pines Input/Output, de los cuales 35 pueden ser usados como salida de PWM, un procesador ARM Cortex-M7 a 600 MHz, y una lógica de activación de 3.3 V, por lo que no soporta tensiones mayores a esta. Requiere un convertidor de nivel lógico que permita adaptar de 5 V a 3.3 V.

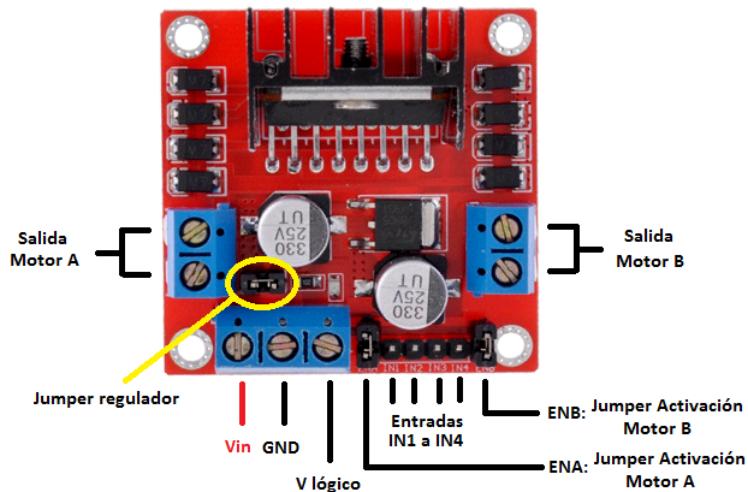
Modulo L298N

Figura 2.37: Módulo L298N

En la figura 2.37 se muestra el módulo L298N, este circuito es un driver que permite controlar dos motores DC o en su defecto un motor a pasos.

Este módulo será usado para regular la posición del péndulo, con las terminales EN1 y EN2, se puede controlar el sentido de giro del motor, la terminal ENA permite regular la velocidad del motor mediante una señal de PWM, permite controlar motores de hasta 35 V. Además, es compatible con tecnologías de 5V y de 3.3 V.

Convertidor de nivel lógico

Para esta tarea se usará un transistor MOSFET que tiene un tiempo de encendido y apagado de 4 ns, está polarizado de tal manera que al tener como entrada un alto lógico con nivel de tensión de 5V, a la salida se tenga 3.3 V como salida en alto.

2.3. Validación e integración de los sistemas

Validación del algoritmo de sintonización (MOP)

Se realizó la verificación del algoritmo de optimización con el algoritmo de evolución diferencial, tomando en cuenta el nuevo planteamiento del problema presentado en la sección de validación del problema de optimización, es decir, se codificó el problema de optimización acorde a las actualizaciones presentadas en la sección antes mencionada.

Se simuló el algoritmo con una población inicial de 100 individuos y 1000 generaciones, el resultado se presenta en la figura 2.38, donde se muestra el frente de Pareto.

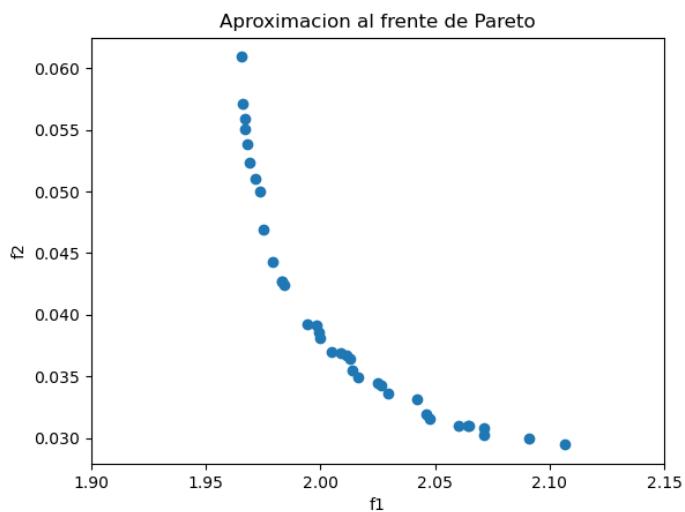


Figura 2.38: Frente de Pareto

Simulación del circuito de control

El circuito de control contempla una tarjeta Teensy 4.1 que es compatible con el IDE de Arduino, es por ello por lo que se seleccionó el Arduino para simular el funcionamiento final del subsistema de control, pues la tarjeta Tennsy no se puede

Diseño del sistema



simular. Como se puede ver en la figura 2.39, se generan 3 señales de control, 2 para controlar el sentido de giro, y una señal de PWM para regular la velocidad de giro del motor, y se recibe de retroalimentación los pulsos del encoder.

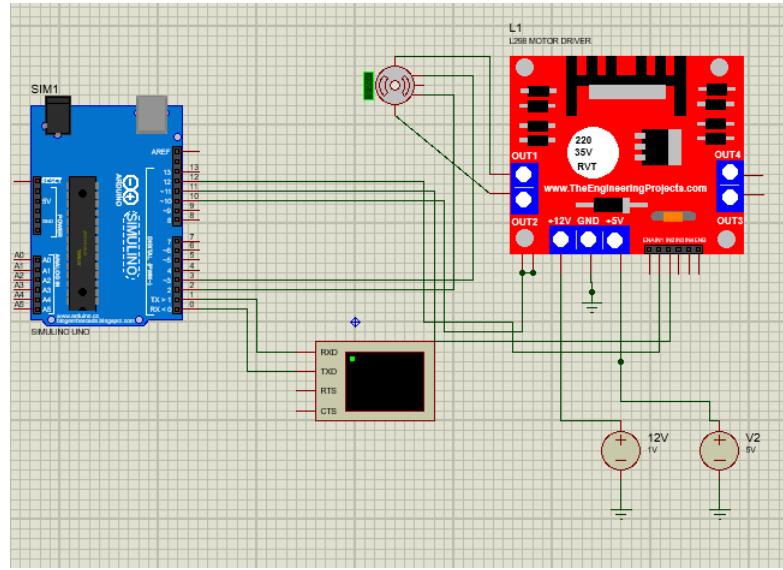


Figura 2.39: Circuito de control del péndulo.

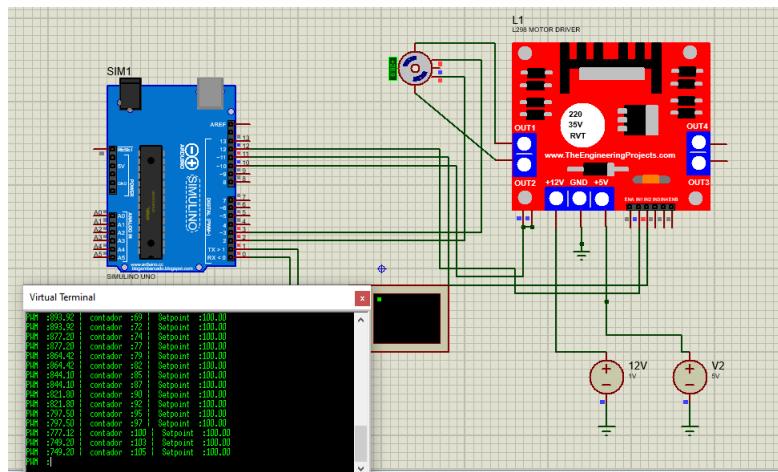


Figura 2.40: Ejecución de simulación.

La figura 2.40 muestra una simulación de un control PID con PWM en Arduino,



también se muestra la comunicación del Arduino con el puerto serie para observar la posición actual del péndulo.

Simulación del convertidor de nivel lógico.

Por último es necesario un convertidor de nivel lógico, que permita convertir un alto lógico de 5 V a un alto lógico de 3.3 V, pues la Teensy 4.1 no soporta tensiones mayores a 3.3 V, es por eso que se usa un circuito con MOSFET, el cual se muestra en la figura 2.41. Este circuito recibe como entrada una señal digital en alto de 5 V, y a la salida se obtiene una señal de en alto de 3.3 V, cuando se recibe una señal de 0 V, la salida es 0 V.

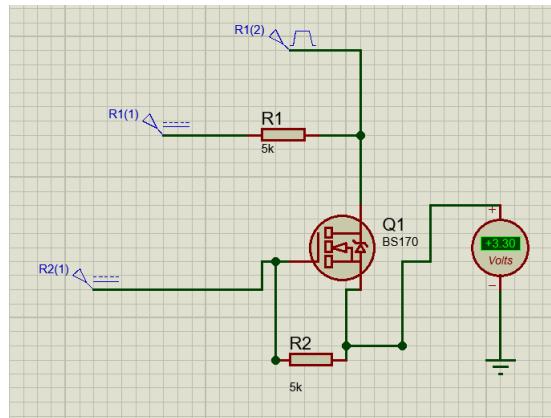


Figura 2.41: Convertidor de nivel lógico de 5 V a 3.3 V.

Simulación de la acción de control implementada (péndulo simple).

En la presente sección se realizó la prueba de sintonización con el algoritmo de evolución diferencial para el sistema de péndulo simple. Del FP obtenido, se seleccionó el punto donde $J_{ISE} = 1.9663022702552264$ y $J_{IADU} = 0.06091058$, este punto da como resultado los siguientes valores de ganancias.

$$k_p = 9.00809903857079.$$



$$k_d = 0.74331509706173.$$

$$k_i = 0.$$

Al hacer la prueba con el algoritmo presentado en el apéndice B, los resultados son una aproximación del comportamiento que se espera en el experimento físico.

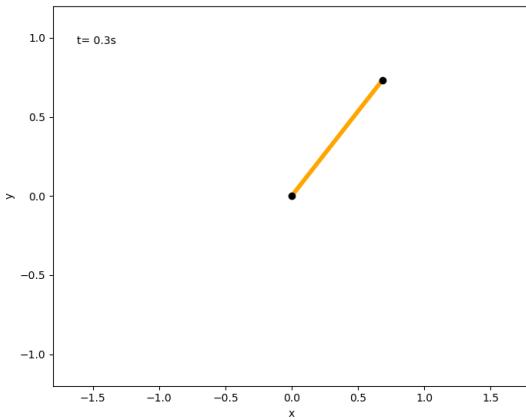


Figura 2.42: Péndulo simple (Control).

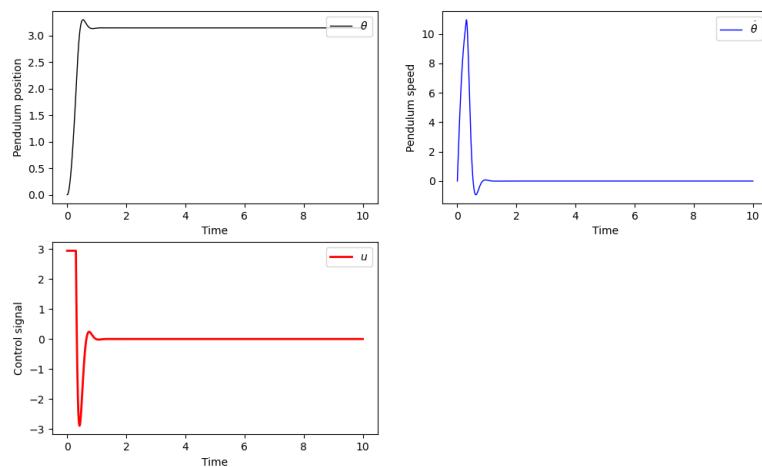


Figura 2.43: Péndulo simple (Control).



En la figura 2.42 se puede ver la simulación del péndulo simple con la acción de control como entrada al sistema. Se puede ver en la animación como el péndulo parte de posición inicial hasta la posición deseada $\theta = \frac{\pi}{2}$.

Las gráficas de posición del comportamiento del péndulo y su velocidad se pueden ver en la Figura 2.43, la gráfica de la señal de control también se muestra en esta figura, y se ve que por un pequeño instante del tiempo, el actuador se satura y cae hasta un pico negativo, posteriormente, el actuador permanece en 0, pues el punto seleccionado es un punto de equilibrio estático.

Conclusiones

3.1. Conclusiones

Los métodos metaheurísticos con procedimientos ampliamente usados en problemas de optimización como técnica que brinda buenos resultados. Los problemas de sintonización también pueden ser tratados como problemas de optimización.

En el presente trabajo se presenta una propuesta de un problema de diseño abordado como un problema de optimización multiobjetivo para la sintonización de controladores PID, el cual permita obtener un conjunto óptimo del cual se pueda extraer una solución que cumpla los compromisos planteados en el problema de acuerdo con las necesidades de la aplicación.

Para poder cumplir la propuesta, se modelaron los sistemas de péndulo simple, péndulo invertido, y péndulo doble. Se seleccionó la ecuación de Euler-Lagrange y se ordenaron los términos en las matrices que describen la dinámica no lineal de los robots, representando los sistemas en matrices de Inercia, Fuerza centrípeta y de Coriolis, y la matriz de gravedad, la representación en espacio de estados mediante estas matrices facilita la solución el planteamiento del problema porque no es necesario resolver todo el sistema para obtener el modelo final.

El método de integración permite resolver la ecuación diferencial planteada, pues la

Conclusiones



ecuación en espacios de estados permite representar a un sistema de orden mayor en un sistema de orden uno, que, al integrarlo completo, se obtiene la solución del sistema.

Debido a su facilidad de interpretación e implementación, y su buena aproximación, el método de integración seleccionado es el método de Euler, el cual mostró buenos resultados en las pruebas y simulaciones realizadas.

La estructura del controlador implementada fue la estructura PID, con la intención de no descartar ninguna acción en los sistemas de péndulo simple y péndulo invertido.

En el caso del péndulo doble, el problema se enfoca en el seguimiento de trayectoria, para este caso, la literatura recomienda usar una estructura de controlador PD+.

Una vez seleccionada la estructura del controlador del sistema, lo ideal es seleccionar las variables de diseño, es decir, las variables que se buscarán a través del algoritmo de optimización, pues así lo recomienda el proceso de planteamiento de un problema de diseño como un problema de optimización.

Las variables de diseño seleccionadas fueron las ganancias del controlador (k_p, k, d, k_i). Esto es crucial para la sintonización, pues el buscar las ganancias que permitan a la señal de control cumplir con los compromisos deseados, es en esencia el proceso de sintonización.

La verificación de las ganancias obtenidas mediante las primeras pruebas del algoritmo de optimización indica que estas ganancias ofrecen una buena respuesta en la simulación dinámica.

Una vez recopilada esta información, se plantea el problema de sintonización como uno de optimización multiobjetivo.

La implementación de evolución diferencial tuvo un resultado exitoso cuando este se probó con el problema de optimización planteado para la sintonización del controlador del péndulo simple, obteniendo un conjunto de soluciones en la aproximación al frente de Pareto y seleccionando una solución para la simulación dinámica del



sistema antes mencionado.

Cabe mencionar que el costo computacional (número de evaluaciones del algoritmo de optimización) fue bajo en relación con el buen resultado obtenido, pues se pudo notar que, a mayor número de evaluaciones, la aproximación al frente de Pareto era más precisa y, con una mayor diversidad en sus soluciones.

Un indicador para medir la calidad del frente de Pareto es el Hipervolumen, ya que los algoritmos evolutivos basados en este han demostrado un buen rendimiento en la resolución de problemas de optimización multiobjetivo, tal como se puede consultar en la literatura.

Debido a lo antes mencionado, este indicador resulta adecuado para evaluar la calidad de los frentes de Pareto obtenidos en la solución de los problemas planteados. Dado a que la naturaleza de los algoritmos evolutivos es no determinista, es decir, son algoritmos estocásticos, es necesario interpretar los datos que arroja el indicador de Hipervolumen con alguna técnica estadística no paramétrica. Dentro de la variedad de técnicas existentes, la que más se ajusta con las necesidades de este trabajo, es la prueba de signo de Wilcoxon.

Las pruebas físicas de la sintonización realizada se llevarán a cabo en el sistema del péndulo simple, para esto fue necesario determinar los componentes y materiales a usar, mismos que fueron seleccionados mediante el método de proceso analítico jerárquico y que conformaran el sistema físico a controlar.

Referencias

- [1] Seeed. Motor de engranes planetarios (1:14) con encoder. [Online]. Available: <https://www.seeedstudio.com/GP36-Planetary-Geared-Motor-with-Encoder-DC-Motor-1-14-1000CPR-p-4224.html>
- [2] M. G. Villarreal-Cervantes and J. Alvarez-Gallegos, “Off-line pid control tuning for a planar parallel robot using de variants,” *Exupert Syst. Appl.*, vol. 64, pp. 444–454, 2016.
- [3] A. Eiben and J. E. Smith, “Introduction to evolutionary computing,” 2003.
- [4] P. Fleming and R. Purshouse, “Evolutionary algorithms in control systems engineering: A survey,” *Control Eng. Pract.*, vol. 10, pp. 1223–1241, 2020.
- [5] G. Reynoso-Meza, X. Blasco, J. Sanchis, and M. Martínez, “Controller tuning using evolutionary multi-objective optimisation: Current trends and applications,” *Control Eng. Pract.*, vol. 28, pp. 58–73, 2014.

REFERENCIAS

- [6] A. E. Ruano, S. S. Ge, T. M. Guerra, F. L. Lewis, J. C. Principe, and M. Colnarie, “Computational intelligence in control,” *Annu. Rev. Control*, vol. 38, pp. 233–242, 2014.
- [7] “Aislación sísmica edificio de gobernantes (análisis comparativo de comportamiento y costos con un edificio tradicional),” *Universidad central del Ecuador, Facultad de ingeniería, ciencias físicas y matemática*, 2014.
- [8] R. Yáñez and O. Morales, “El péndulo doble como ejemplo de sistema caótico.”
- [9] M. Cruz and D. Oscar, “Simulación de un sismo mediante el movimiento de un péndulo doble,” *Instituto Tecnológico de Matamoros*.
- [10] D. J. Block, K. J. Åström, and M. W. Spong, “The reaction wheel pendulum,” vol. 1, pp. 112–114, 2007.
- [11] U. I. de Valencia, 2017. [Online]. Available: <https://www.universidadviu.com/los-lenguajes-programacion-alto-nivel/>
- [12] L. Kumar, P. Kumar, and D. Narang, “Tuning of fractional order $pi^\lambda d^\mu$ controllers using evolutionary optimization for pid tuned synchronous generator excitation system,” *IFAC-PapersOnLine*, vol. 51, no. 4, pp. 859–864, 2018.
- [13] G. Reynoso-Meza, H. Sánchez, and V. Ribeiro, “Control of refrigeration systems based on vapour compression using multi-objective optimization techniques,” vol. 51, no. 4, pp. 722–724, 2018.
- [14] Z. Chen, X. Yuan, B. Ji *et al.*, “Design of a fractional order pid controller for hydraulic turbine regulating system using chaotic nondominated sorting genetic algorithm ii,” vol. 84, pp. 390–404, 2014.
- [15] Y. Tian, Q. Wang, Y. Wang *et al.*, “A novel design method of multiobjective robust pid controller for industrial process,” in *2014 9th IEEE Conference on*



- Industrial Electronics and Applications.* Hangzhou, China: IEEE, June 2014, pp. 242–246.
- [16] N. Yegireddy and S. Panda, “Design and performance analysis of pid controller for an avr system using multi-objective non-dominated shorting genetic algorithm-ii,” in *2014 International Conference on Smart Electric Grid (ISEG)*. Guntur, India: IEEE, September 2014, pp. 1–47.
- [17] S. Panda, “Multi-objective pid controller tuning for a facts-based damping stabilizer using non-dominated sorting genetic algorithm-ii,” vol. 33, no. 7, pp. 1296–1308, 2011.
- [18] Z. Chen, X. Yuan, B. Ji *et al.*, “Application of multiobjective controller to optimal tuning of pid gains for a hydraulic turbine regulating system using adaptive grid particle swam optimization,” vol. 56, pp. 173–187, 2015.
- [19] M. Villarreal-Cervantes, A. Rodríguez-Molina, Peñaloza-Mejía *et al.*, “Multi-objective on-line optimization approach for the dc motor controller tuning using differential evolution,” pp. 20 393—20 407, 2017.
- [20] M. Hung, L. Shu, S. Ho *et al.*, “A novel intelligent multiobjective simulated annealing algorithm for designing robust pid controllers,” vol. 38, no. 2, pp. 319—330, 2008.
- [21] A. Rodríguez-Molina, M. Villarreal-Cervantes, , Mezura-Montes *et al.*, “Adaptive controller tuning method based on online multiobjectiveoptimization: A case study of the four-bar mechanism,” pp. 1—14, 2019.
- [22] D. G. Fernández, “Construcción de un regulador de watt para el control de un motor de corriente continua,” Ph.D. dissertation, Universidad Politécnica de Cartagena, Cartagena,España, 4 2019.
- [23] C. Lagos. Introducción histórica del control automático. [Online]. Available: <http://www.emb.cl/electroindustria/articulo>.

REFERENCIAS

- mvc?xid=474&ni=introduccion-historica-del-control-automatico&fbclid=IwAR0s7kkYXtjVCypIMXaEhgHFY9KSr5XWckMQL907FlL4mFrsKThSpa7AAuo
- [24] C. de Prada. Reguladores pid. [Online]. Available: <https://www.eii.uva.es/~prada/Pid.pdf>
 - [25] M. H. C. Botero, “Optimización multiobjetivo aplicada a problemas de planificación de tareas en ambientes de computo heterogéneo,” Ph.D. dissertation, Universidad de los Andes, Bogotá, 2010.
 - [26] K. Ogata, *Ingeniería de control moderna*, 5th ed. Madrid: PEARSON EDUCACIÓN, 2010.
 - [27] K. Chong and Y. Li, “Pid control system analysis, design, and technology,” *IEEE Trans. Control System Technology*, vol. 13, pp. 559–576, 2005.
 - [28] K. Aström and T. Hagglund, *PID Controllers: Theory, Design and Tuning*, 2nd ed. USA: ISA, 1994.
 - [29] R. Carmona, “Análisis de estabilidad en controladores pid y neuronales pid sobre robots,” Ph.D. dissertation, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional Unidad Zacatenco, 2013.
 - [30] G. Forsythe, M. Malcom, and C. Moler, *Computer Methods for Mathematical Computations*. Prentice-Hall, 1977.
 - [31] R. Kent, E. Saff, and A. Snider, *Ecuaciones diferenciales y problemas con valores en la frontera*. Pearson Educación en México, 2005.
 - [32] F. Reyes, *Robótica. Control de robots manipuladores*, 1st ed. México: Alfaomega, 2011.
 - [33] B. M. Escárcega, “Control de movimiento de una grúa lineal,” Ph.D. dissertation, Universidad de Sonora, Sonora, 2016.



- [34] G. R.-M. and Javier Sanchi and X. B. M. M. inez, “Algoritmos evolutivos y su empleo en el ajuste de controladores del tipo pid: Estado actual y perspectivas,” *Revista Iberoamericana de Automática e Informática industrial*, pp. 251–268, 20123.
- [35] C. Mattson and A. Messac, “Pareto frontier based concept selection under uncertainty, with visualization,” *Optimization and Engineering*, no. 6, pp. 85–115, 2005.
- [36] K. Saridakis and A. Dentsoras, “Soft computing in engineering design – a review,” *Advanced Engineering Informatics*, vol. 22, no. 2, pp. 202–221, 2008.
- [37] F. S. Hillier and G. J. Lieberman, *Introducción a la investigación de operaciones*, 9th ed. McGrawHill, 2010.
- [38] Heerrera. Introducción a los algoritmos metaheurísticos. [Online]. Available: <https://sci2s.ugr.es/sites/default/files/files/Teaching/OtherPostGraduateCourses/Metaheuristicas/Int-Metaheuristicas-CAEPIA-2009.pdf>
- [39] A. Rodríguez-Molina, E. Mezura-Montes, M. G. Villarreal-Cervantes, and M. Aldape-Pérez, “Multi-objective meta-heuristic optimization in intelligent control: A survey on the controller tuning problem,” *Applied Soft Computing Journal*, 2020.
- [40] J. Holland, “Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence,” *U. Michigan Press*, 1975.
- [41] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 13th ed., USA, 1988.

REFERENCIAS



- [42] R. Storn and K. Price, “Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces,” *Journal of Global Optimization*, pp. 341–359, 1997.
- [43] E. Mezura, M. Reyes, and C. Coello, “Multi-objective optimization using differential evolution: A survey of the state-of-the-art,” *Advances in Differential Evolution (SCI 143)*, pp. 173–196, 2008.
- [44] Das, S. Suganthan, and P. N., “Differential evolution: A survey of the state-of-the-art. evolutionary computation,” *IEEE Transactions on PP (99)*, p. 1 –28, 2010.
- [45] U. Rout, R. Sahu, and S. Panda, “Design ans analysis od differential evolution algorithm based automatic generation control for interconnected power system,” *Ahia Shams Engineering Journal*, pp. 409–421, 2013.
- [46] J. A. Rodrigo, “Optimización con enjambre de partículas (particle swarm optimization),” 2019. [Online]. Available: https://www.cienciadedatos.net/documentos/py02_optimizacion_pso
- [47] J. Kennedyand and R. Eberhart, “Particle swarm optimization. in: Neural networks. proceedings”,” *IEEE International Conference*, vol. 4, pp. 1942–1948, 1995.
- [48] C. Coello, “An introduction to multi-objective particle swarm optimizers,” *Soft Computing in Industrial Applications*, vol. 96, pp. 3–12, 2011.
- [49] K. Deb, “An efficient constraint handling method for genetic algorithms,” *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2, pp. 311–338, 2000.
- [50] V. del Cisne Orozco Orozco, “Desarrollo de una herramienta computacional para la sintonización de parámetros de controladores pid y smc para el seguimiento



- de trayectoria de un cuadricóptero basado en algoritmos genéticos,” Master’s thesis, Escuela Politécnica Nacional, Quito, Ecuador, 6 2018.
- [51] S. A. C. Goraldo. Índices de desempeño. [Online]. Available: <https://controlautomaticoeducacion.com/control-realimentado/indices-de-desempeno/>
- [52] H. S. S. Corrales, “Multi-objective optimization and multicriteria design of pi/-pid controllers,” Ph.D. dissertation, Universidad Autónoma de Barcelona, Barcelona, España, 6 2016.
- [53] S. Jiang, Y. Oong, J. Zhang, and L. Feng, “Consistencies and contradictions of performance metrics in multiobjective optimization,” *IEEE*, vol. 44, no. 12, pp. 2391–2404, 2014.
- [54] E. Alba *et al.* Metaheurísticas multiobjetivo para optimizar el proceso de difusión en manets metropolitanas. [Online]. Available: <https://neo.lcc.uma.es/staff/paco/pdfs/jaem2007.pdf>
- [55] M. L. Bautista, E. V. Rodríguez, L. B. Vargas, and C. C. Hernández, “Pruebas estadísticas parámetricas y no parámetricas: su clasificación, objetivos y características,” *Educación y Salud Boletín Científico Instituto de Ciencias de la Salud*, vol. 9, 2020.
- [56] L. G. Aragón, *Estadística en el área de las Ciencias Sociales y Administrativas*. México: Alfaomega, 206.
- [57] L. Saaty, *The Analytic Hierarchy Process*. McGraw-Hill, 1980.
- [58] T. Hurtado and G. Bruno, “El proceso de análisis jerárquico (ahp) como herramienta para la toma de decisiones en la selección de proveedores,” Ph.D. dissertation, Universidad Nacional Mayor de San Marcos, Lima, 2005.
- [59] J. Gausemeier and S. Moehringer, “Vdi 2206- a new guideline for the design of mechatronic systems,” vol. 35, no. 2, pp. 785–790, 2002.

REFERENCIAS



- [60] Wondershare. Qué es idef - definición, métodos, y beneficios. [Online]. Available: <https://www.edrawsoft.com/es/what-is-idef.html>
- [61] D. J. George, "Concept generation using morphological and option matrices," Ph.D. dissertation, Clemson University, Carolina del Sur, 2012.
- [62] L. Saaty, *The Analytic Hierarchy Process*, 1st ed. New York: McGraw-Hill, 1980.

Apéndices

Apéndice A: Modelos dinámicos

.1. Péndulo simple

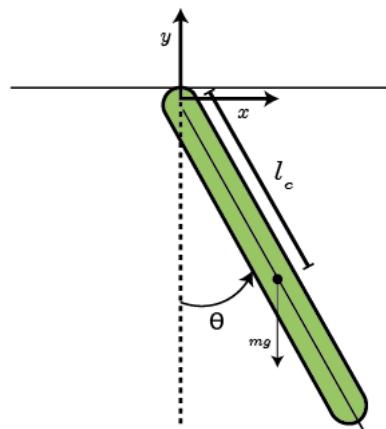


Figura 1: Péndulo simple.

De acuerdo con la Figura 1, los parámetros identificados en la misma se describen en la Tabla 1



TABLA 1: Notación péndulo simple.

Significado	Notación
Longitud al centro de masa	l_c
Masa de la barra	m
Inercia de la barra	I
Posición angular de la barra	θ
Aceleración debida a la gravedad	g

La posición de la barra es:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} l_c \sin \theta \\ -l_c \cos \theta \end{bmatrix}$$

Obteniendo la derivada con respecto del tiempo, la velocidad esta dada de la siguiente manera:

$$v = \frac{d}{dt} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} l_c \cos \theta \\ l_c \sin \theta \end{bmatrix} \dot{\theta}$$

La velocidad al cuadrado es:

$$v^T v = l_c^2 \dot{\theta}^2 (\cos^2 \theta + \sin^2 \theta) = l_c^2 \dot{\theta}^2$$

Calculando la energía cinética translacional:

$$K_T = \frac{1}{2} m v^T v = \frac{1}{2} m l_c^2 \dot{\theta}^2$$

La energía cinética de rotación

$$K_R = \frac{1}{2} I \dot{\theta}^2$$

La energía cinética total:

$$K = K_T + K_R = \frac{1}{2} m l_c^2 \dot{\theta}^2 + \frac{1}{2} I \dot{\theta}^2$$



La energía potencial del sistema:

$$U = mgh = -mgl_c \cos \theta$$

Una vez calculadas las energías del sistema, se calcula el Lagrangiano de la siguiente manera:

$$L = K - U$$

$$L = \frac{1}{2}ml_c^2\dot{\theta}^2 + \frac{1}{2}I\dot{\theta}^2 - (-mgl_c \cos \theta) = \frac{1}{2}ml_c^2\dot{\theta}^2 + \frac{1}{2}I\dot{\theta}^2 + (mgl_c \cos \theta)$$

$$D = \frac{1}{2}b\dot{\theta}^2$$

La ecuación del movimiento de Euler-Lagrange

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\theta}} \right) - \frac{\partial L}{\partial \theta} + \frac{\partial D}{\partial \dot{\theta}} = \tau$$

$$\frac{\partial L}{\partial \dot{\theta}} = \frac{\partial(\frac{1}{2}ml_c^2\dot{\theta}^2 + \frac{1}{2}I\dot{\theta}^2 + (mgl_c \cos \theta))}{\partial \dot{\theta}} = ml_c^2\dot{\theta} + I\dot{\theta}$$

$$\frac{\partial L}{\partial \theta} = \frac{\partial(\frac{1}{2}ml_c^2\dot{\theta}^2 + \frac{1}{2}I\dot{\theta}^2 + (mgl_c \cos \theta))}{\partial \theta} = -mgl_c \sin \theta$$

$$\frac{\partial D}{\partial \dot{\theta}} = \frac{\partial(\frac{1}{2}b\dot{\theta}^2)}{\partial \dot{\theta}} = b\dot{\theta}$$

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\theta}} \right) = \frac{d}{dt} (ml_c^2\dot{\theta} + I\dot{\theta}) = ml_c^2\ddot{\theta} + I\ddot{\theta}$$

$$ml_c^2\ddot{\theta} + I\ddot{\theta} - (-mgl_c \sin \theta) + b\dot{\theta} = \tau$$

$$ml_c^2\ddot{\theta} + I\ddot{\theta} + (mgl_c \sin \theta) + b\dot{\theta} = \tau$$

$$\ddot{\theta} = \frac{\tau - ((mgl_c \sin \theta) + b\dot{\theta})}{ml_c^2 + I}$$

Representación del modelo en ecuaciones de estado:

Las variables de estado son: $\theta, \dot{\theta}$

Vector de estado $\mathbf{x} = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}$

Ecuación de estado:

$$\dot{\mathbf{x}} = f(u, x, t)$$

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\theta} \\ \frac{u - ((mgl_c \sin \theta) + b\dot{\theta})}{ml_c^2 + I} \end{bmatrix}$$

.2. Péndulo invertido

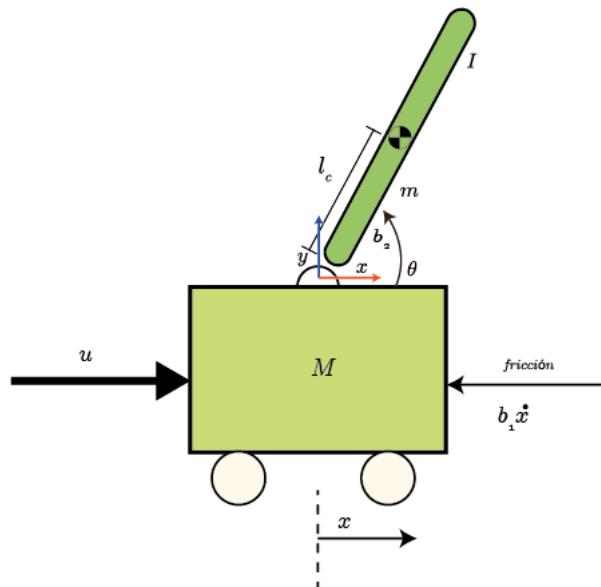


Figura 2: Péndulo invertido.



De acuerdo con la Figura 2, los parámetros identificados en la misma se describen en el la Tabla 2

TABLA 2: Notación péndulo invertido.

Significado	Notación
Longitud al centro de masa	l_c
Masa de la barra	m
Masa del carro	M
Inercia de la barra	I
Posición angular de la barra	θ
Coeficiente de fricción	$b_{1,2}$
Entrada	u

Es necesario comenzar a describir el movimiento obteniendo el Lagrangiano. Para la coordenada del centro de masa del brazo se tiene:

$$p = \begin{bmatrix} l_c \cos \theta + x \\ l_c \sin \theta \end{bmatrix}$$

Por lo que la velocidad queda expresada de la siguiente manera:

$$v = \frac{d}{dt} p = \dot{p} = \begin{bmatrix} -l_c \dot{\theta} \sin \theta + \dot{x} \\ l_c \dot{\theta} \cos \theta \end{bmatrix} = -l_c \dot{\theta} \sin \theta + l_c \dot{\theta} \cos \theta + \dot{x}$$

Se calcula la energía cinética del sistema:

$$K = \frac{1}{2} m v^2 + \frac{1}{2} I \dot{\theta}^2 + \frac{1}{2} M \dot{x}^2$$

Donde:

$$v = \begin{bmatrix} -l_c \dot{\theta} \sin \theta + \dot{x} \\ l_c \dot{\theta} \cos \theta \end{bmatrix} = -l_c \dot{\theta} \sin \theta + l_c \dot{\theta} \cos \theta + \dot{x}$$

$$v^2 = (l_c \dot{\theta} \cos \theta)^2 + (-l_c \dot{\theta} \sin \theta + \dot{x})^2 = l_c^2 \dot{\theta}^2 \sin^2 \theta + l_c^2 \dot{\theta}^2 \cos^2 \theta - 2l_c \dot{x} \dot{\theta} \sin \theta + \dot{x}^2 = l_c^2 \dot{\theta}^2 - 2l_c \dot{x} \dot{\theta} \sin \theta + \dot{x}^2$$

$$K = \frac{1}{2} m (l_c^2 \dot{\theta}^2 - 2l_c \dot{x} \dot{\theta} \sin \theta + \dot{x}^2) + \frac{1}{2} I \dot{\theta}^2 + \frac{1}{2} M \dot{x}^2$$

Simplificando

$$K = \frac{1}{2} (M + m) \dot{x}^2 + \frac{1}{2} (I + ml_c^2) \dot{\theta}^2 - ml_c \dot{x} \dot{\theta} \sin \theta$$

Ahora encontramos la energía potencial:

$$U = mg l \sin \theta$$

$$D = \frac{1}{2} b_1 \dot{x}^2 + \frac{1}{2} b_2 \dot{\theta}^2$$

$$L = K - U$$

$$L = \frac{1}{2} (M + m) \dot{x}^2 + \frac{1}{2} (I + ml^2) \dot{\theta}^2 + ml \dot{x} \dot{\theta} \cos \theta + mg l \cos \theta$$

$$q_1 = x$$

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{x}} \right) - \frac{\partial L}{\partial x} + \frac{\partial D}{\partial \dot{x}} = F$$

$$(M + m) \ddot{x} + b_1 \dot{x} - ml_c \ddot{\theta} \sin \theta - ml_c \dot{\theta}^2 \cos \theta = u_1$$



$$q_2 = \theta$$

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\theta}} \right) - \frac{\partial L}{\partial \theta} + \frac{\partial D}{\partial \dot{\theta}} = 0$$

$$-ml_c \ddot{x} \sin \theta + (I + ml_c^2) \ddot{\theta} + mgl_c \cos \theta + b_2 \dot{\theta} = u_2$$

Por lo tanto, se llega a las siguientes ecuaciones:

$$(M + m) \ddot{x} + b_1 \dot{x} - ml_c \ddot{\theta} \sin \theta - ml_c \dot{\theta}^2 \cos \theta = u_1$$

$$-ml_c \ddot{x} \sin \theta + (I + ml_c^2) \ddot{\theta} + mgl_c \cos \theta + b_2 \dot{\theta} = u_2$$

$$\ddot{x} = \frac{u_1 - b_1 \dot{x} + ml_c \ddot{\theta} \sin \theta + ml_c \dot{\theta}^2 \cos \theta}{M + m}$$

$$\ddot{\theta} = \frac{u_2 - ml_c \ddot{x} \sin \theta - mgl_c \cos \theta - b_2 \dot{\theta}}{I + ml_c^2}$$

Un sistema mecánico se puede escribir en su forma de espacios de estados, como se muestra a continuación:

$$M(\mathbf{q}) \ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}}) \dot{\mathbf{q}} + g(\mathbf{q}) = \mathbf{u}$$

Donde

$M(\mathbf{q}) \ddot{\mathbf{q}}$ es la matriz de inercia

$C(\mathbf{q}, \dot{\mathbf{q}}) \dot{\mathbf{q}}$ es la matriz de Coriolis y fuerza centrífuga

$g(\mathbf{q})$ es el vector de gravedad

\mathbf{u} es el vector de fuerzas y pares generalizados

$$\begin{bmatrix} M+m & -ml_c \sin \theta \\ -ml_c \sin \theta & I + ml_c^2 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} b_1 & -ml_c \dot{\theta} \cos \theta \\ 0 & b_2 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ mgl_c \cos \theta \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$\begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} M+m & -ml_c \sin \theta \\ -ml_c \sin \theta & I + ml_c^2 \end{bmatrix}^{-1} \left(\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} - \begin{bmatrix} b_1 & -ml_c \dot{\theta} \cos \theta \\ 0 & b_2 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{\theta} \end{bmatrix} - \begin{bmatrix} 0 \\ mgl_c \cos \theta \end{bmatrix} \right)$$

$$\ddot{\mathbf{q}} = M^{-1}(\mathbf{q}) [\mathbf{u} - C(q, \dot{\mathbf{q}}) \dot{\mathbf{q}} + g(\mathbf{q})]$$

Variables de estado: $x, \dot{x}, \theta, \dot{\theta}$

Vector de estado:

$$z = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix}$$

Ecuación de estado

$$\dot{z} = f(u, z, t)$$

$$\begin{bmatrix} x \\ \theta \\ \dot{x} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} x \\ \theta \\ M^{-1}(\mathbf{q}) [\mathbf{u} - C(\mathbf{q}, \dot{\mathbf{q}}) \dot{\mathbf{q}} - g(\mathbf{q})] \\ \dot{x} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} x \\ \theta \\ \begin{bmatrix} M+m & ml_c \cos \theta \\ ml_c \cos \theta & I + ml_c^2 \end{bmatrix}^{-1} \left(\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} - \begin{bmatrix} b_1 & -ml_c \dot{\theta} \sin \theta \\ 0 & b_2 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{\theta} \end{bmatrix} - \begin{bmatrix} 0 \\ mgl_c \sin \theta \end{bmatrix} \right) \\ \dot{x} \\ \dot{\theta} \end{bmatrix}$$



.3. Péndulo doble

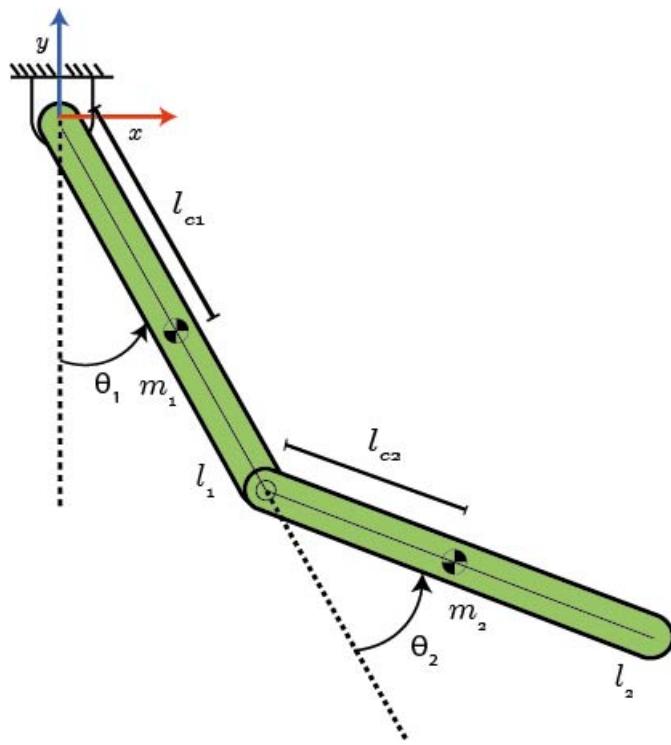


Figura 3: Péndulo doble.

La Figura 3 muestra el péndulo doble, los parámetros mostrados en ella se explican en la Tabla 3

Una vez identificados todos los parámetros, se puede comenzar a describir el modelo que describe el movimiento del sistema, el modelado se realizará con la ecuación de movimiento. Es necesario comenzar a describir el movimiento obteniendo el Lagrangiano. Para la coordenada del centro de masa del brazo 1:

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} l_{c1} \sin \theta_1 \\ -l_{c1} \cos \theta_1 \end{bmatrix}$$

Para la coordenada del centro de masa del brazo 2:



TABLA 3: Notación péndulo doble.

Brazo	Descripción	Notación
1	Masa del brazo 1	m_1
	Longitud al centro de masa del brazo 1	l_{c1}
	Longitud del brazo 1	l_1
	Posición angular del brazo 1	θ_1
	Inercia del brazo 1	I_1
2	Masa del brazo 2	m_2
	Longitud al centro de masa del brazo 2	l_{c2}
	Longitud del brazo 2	l_2
	Posición angular del brazo 2	θ_2
	Inercia del brazo 2	I_2
	Aceleración de la gravedad	g

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} l_1 \sin \theta_1 + l_{c2} \sin (\theta_1 + \theta_2) \\ -l_1 \cos \theta_1 - l_{c2} \cos (\theta_1 + \theta_2) \end{bmatrix}$$

La velocidad en ambas coordenadas es:

$$v_1 = \frac{d}{dt} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \frac{d}{dt} \begin{bmatrix} l_{c1} \sin \theta_1 \\ -l_{c1} \cos \theta_1 \end{bmatrix} = \begin{bmatrix} l_{c1} \cos \theta_1 \\ l_{c1} \sin \theta_1 \end{bmatrix} \dot{\theta}_1$$

$$v_2 = \frac{d}{dt} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \frac{d}{dt} \begin{bmatrix} l_1 \sin \theta_1 + l_{c2} \sin (\theta_1 + \theta_2) \\ -l_1 \cos \theta_1 - l_{c2} \cos (\theta_1 + \theta_2) \end{bmatrix} = \begin{bmatrix} l_1 \cos \theta_1 + l_{c2} \cos (\theta_1 + \theta_2) & l_{c2} \cos (\theta_1 + \theta_2) \\ l_1 \sin \theta_1 + l_{c2} \sin (\theta_1 + \theta_2) & l_{c2} \sin (\theta_1 + \theta_2) \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

Calculando la energía cinética del sistema, es necesario calcular el cuadrado de cada una de las velocidades, lo cual se presenta a continuación:

$$v_1^T v_1 = (l_{c1} \dot{\theta}_1 \cos \theta_1)^2 + (l_{c1} \dot{\theta}_1 \sin \theta_1)^2 = l_{c1}^2 \dot{\theta}_1^2 (\cos^2 \theta_1 + \sin^2 \theta_1) = l_{c1}^2 \dot{\theta}_1^2$$



$$\begin{aligned}
v_2^T v_2 &= (\dot{\theta}_1(l_1 \cos \theta_1 + l_{c2} \cos(\theta_1 + \theta_2)) + \dot{\theta}_2(l_{c2} \cos(\theta_1 + \theta_2)))^2 \\
&\quad + (\dot{\theta}_1(l_1 \sin \theta_1 + l_{c2} \sin(\theta_1 + \theta_2)) + \dot{\theta}_2(l_{c2} \sin(\theta_1 + \theta_2)))^2 \\
&= \dot{\theta}_1^2(l_1 \cos \theta_1 + l_{c2} \cos(\theta_1 + \theta_2))^2 + \dot{\theta}_2^2((l_{c2} \cos(\theta_1 + \theta_2))^2 + 2\dot{\theta}_1\dot{\theta}_2(l_1 \cos \theta_1 \\
&\quad + l_{c2} \cos(\theta_1 + \theta_2))(l_{c2} \cos(\theta_1 + \theta_2)) + \dot{\theta}_1^2(l_1 \sin \theta_1 + l_{c2} \sin(\theta_1 + \theta_2))^2 \\
&\quad + \dot{\theta}_2^2((l_{c2} \sin(\theta_1 + \theta_2))^2 + 2\dot{\theta}_1\dot{\theta}_2(l_1 \sin \theta_1 + l_{c2} \sin(\theta_1 + \theta_2))(l_{c2} \sin(\theta_1 + \theta_2))) \\
&= \dot{\theta}_1^2 l_1 (\cos^2 \theta_1 + \sin^2 \theta_1) + \dot{\theta}_1^2 l_{c2}^2 (\cos^2(\theta_1 + \theta_2) + \sin^2(\theta_1 + \theta_2)) \\
&\quad + 2\dot{\theta}_1^2 l_1 l_{c2} ((\cos \theta_1)(\cos(\theta_1 + \theta_2)) + (\sin \theta_1)(\sin(\theta_1 + \theta_2))) \\
&\quad + 2\dot{\theta}_1\dot{\theta}_2(l_1 l_{c2} (\cos \theta_1 \cos(\theta_1 + \theta_2) + \sin \theta_1 \sin(\theta_1 + \theta_2))) \\
&\quad + l_{c2}^2 ((\cos^2(\theta_1 + \theta_2) + \sin^2(\theta_1 + \theta_2))) + \dot{\theta}_2^2 l_{c2}^2 (\cos^2 \theta_1 + \sin^2 \theta_1)
\end{aligned}$$

$$v_2^T v_2 = \dot{\theta}_1^2 (l_1^2 + l_{c2}^2 + 2l_1 l_{c2} \cos(\theta_2)) + 2\dot{\theta}_1\dot{\theta}_2 (l_1 l_{c2} \cos(\theta_2) + l_{c2}^2) + \dot{\theta}_2^2 l_{c2}^2$$

La energía cinética total del sistema es:

$$\begin{aligned}
K &= \frac{1}{2} \left(m_1 v_1^T v_1 + I_1 (\dot{\theta}_1^2) + m_2 (v_2^T v_2) + I_2 (\dot{\theta}_1 + \dot{\theta}_2)^2 \right) \\
&= \frac{1}{2} \left(m_1 l_{c1}^2 \dot{\theta}_1^2 + I_1 (\dot{\theta}_1^2) + m_2 \left(\dot{\theta}_1^2 (l_1^2 + \right. \right. \\
&\quad \left. \left. l_{c2}^2 + 2l_1 l_{c2} \cos(\theta_2) + 2\dot{\theta}_1\dot{\theta}_2 (l_1 l_{c2} \cos(\theta_2) + l_{c2}^2) + \dot{\theta}_2^2 l_{c2}^2 + I_2 (\dot{\theta}_1 + \dot{\theta}_2)^2 \right) \right. \\
&\quad \left. \left. = \frac{1}{2} \left(\dot{\theta}_1^2 (m_1 l_{c1}^2 + I_1 + I_2 + m_2 (l_1^2 + l_{c2}^2 + 2l_1 l_{c2} \cos(\theta_2))) + \right. \right. \right. \\
&\quad \left. \left. \left. \dot{\theta}_2^2 (m_2 l_{c2}^2 + I_2) + 2\dot{\theta}_1\dot{\theta}_2 (m_2 l_1 l_{c2} \cos(\theta_2) + m_2 l_{c2}^2 + I_2) \right) \right)
\end{aligned}$$



$$U = g(m_1 y_1 + m_2 y_2) = -g(m_1 l_{c1} \cos \theta_1 + m_2 (l_1 \cos \theta_1 + l_{c2} \cos(\theta_1 + \theta_2)))$$

A continuación, se presenta el Lagrangiano:

$$\begin{aligned} L = K - U = & \frac{1}{2} \left(\dot{\theta}_1^2 (m_1 l_{c1}^2 + I_1 + I_2 + m_2 (l_1^2 + l_{c2}^2 + 2l_1 l_{c2} \cos(\theta_2))) + \right. \\ & \dot{\theta}_2^2 (m_2 l_{c2}^2 + I_2) + 2\dot{\theta}_1 \dot{\theta}_2 (m_2 l_1 l_{c2} \cos(\theta_2)) + \\ & \left. m_2 l_{c2}^2 + I_2 + g(m_1 l_{c1} \cos \theta_1 + m_2 (l_1 \cos \theta_1 + l_{c2} \cos(\theta_1 + \theta_2))) \right) \end{aligned}$$

$$D = \frac{1}{2} (m_1 b_1 \dot{\theta}_1^2 + m_2 b_2 \dot{\theta}_2^2)$$

Las ecuaciones de movimiento para la coordenada generalizada θ_1 :

$$\begin{aligned} \frac{\partial L}{\partial \dot{\theta}_1} = & \dot{\theta}_1 (m_1 l_{c1}^2 + I_1 + I_2 + \\ & m_2 (l_1^2 + l_{c2}^2 + 2l_1 l_{c2} \cos(\theta_2)) + \dot{\theta}_2 (m_2 l_1 l_{c2} \cos(\theta_2) + m_2 l_{c2}^2 + I_2)) \\ (1) \end{aligned}$$

$$\begin{aligned} \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\theta}_1} \right) = & (\ddot{\theta}_1 (m_1 l_{c1}^2 + I_1 + I_2 + \\ & m_2 (l_1^2 + l_{c2}^2 + 2l_1 l_{c2} \cos(\theta_2)) + \ddot{\theta}_2 (m_2 l_1 l_{c2} \cos(\theta_2) + m_2 l_{c2}^2 + I_2)) \end{aligned}$$

$$-2m_2 \dot{\theta}_1 \dot{\theta}_2 l_1 l_{c2} \sin(\theta_2) - m_2 \dot{\theta}_2^2 l_1 l_{c2} \sin(\theta_2)$$

$$\frac{\partial L}{\partial \theta_1} = -g(m_1 l_{c1} \sin \theta_1 + m_2 (l_1 \sin \theta_1 + l_{c2} \sin(\theta_1 + \theta_2)))$$



$$\begin{aligned}
 \frac{\partial D}{\partial \dot{\theta}_1} &= \frac{\partial}{\partial \dot{\theta}_1} \left(\frac{1}{2} \left(m_1 b_1 \dot{\theta}_1^2 + m_2 b_2 \dot{\theta}_2^2 \right) \right) = m_1 b_1 \dot{\theta}_1 \\
 \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\theta}_1} \right) - \frac{\partial L}{\partial \theta_1} + \frac{\partial D}{\partial \dot{\theta}_1} &= \\
 \left(\ddot{\theta}_1 (m_1 l_{c1}^2 + I_1 + I_2 + m_2 (l_1^2 + l_{c2}^2 + 2l_1 l_{c2} \cos(\theta_2))) \right. &+ \left. \ddot{\theta}_2 (m_2 l_1 l_{c2} \cos(\theta_2) + m_2 l_{c2}^2 + I_2) \right) \\
 - 2m_2 \dot{\theta}_1 \dot{\theta}_2 l_1 l_{c2} \sin(\theta_2) - m_2 \dot{\theta}_2^2 l_1 l_{c2} \sin(\theta_2) + \\
 m_1 b_1 \dot{\theta}_1 + g (m_1 l_{c1} \sin \theta_1 + m_2 (l_1 \sin \theta_1 + l_{c2} \sin(\theta_1 + \theta_2))) &= \tau_1
 \end{aligned}$$

Para la coordenada generalizada θ_2 :

$$\begin{aligned}
 \frac{\partial L}{\partial \dot{\theta}_2} &= \dot{\theta}_2 (m_2 l_{c2}^2 + I_2) + \dot{\theta}_1 (m_2 l_1 l_{c2} \cos(\theta_2) + m_2 l_{c2}^2 + I_2) \\
 \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\theta}_2} \right) &= \ddot{\theta}_2 (m_2 l_{c2}^2 + I_2) + \ddot{\theta}_1 (m_2 l_1 l_{c2} \cos(\theta_2) + m_2 l_{c2}^2 + I_2) - \dot{\theta}_1 \dot{\theta}_2 (m_2 l_1 l_{c2} \sin(\theta_2)) \\
 \frac{\partial D}{\partial \dot{\theta}_2} &= \frac{\partial}{\partial \dot{\theta}_2} \left(\frac{1}{2} \left(m_1 b_1 \dot{\theta}_1^2 + m_2 b_2 \dot{\theta}_2^2 \right) \right) = m_2 b_2 \dot{\theta}_2 \\
 \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\theta}_2} \right) - \frac{\partial L}{\partial \theta_2} + \frac{\partial D}{\partial \dot{\theta}_2} &= \\
 \ddot{\theta}_2 (m_2 l_{c2}^2 + I_2) &+ \ddot{\theta}_1 (m_2 l_1 l_{c2} \cos(\theta_2) + m_2 l_{c2}^2 + I_2) + \\
 m_2 \dot{\theta}_1^2 l_1 l_{c2} \sin(\theta_2) + m_2 b_2 \dot{\theta}_2 &+ g m_2 l_{c2} \sin(\theta_1 + \theta_2) = \tau_2
 \end{aligned}$$

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = u$$

Donde:

$M(q)\ddot{q}$ es la matriz de inercia

$C(q, \dot{q})\dot{q}$ es la matriz de Coriolis y fuerza centrífuga

$g(q)$ es el vector de gravedad

u es el vector de fuerzas y pares generalizados

Agrupando los términos:

$$M = \begin{bmatrix} m_1 l_{c1}^2 + I_1 + I_2 + m_2 (l_1^2 + l_{c2}^2 + 2l_1 l_{c2} \cos(\theta_2)) & m_2 l_1 l_{c2} \cos(\theta_2) + m_2 l_{c2}^2 + I_2 \\ m_2 l_1 l_{c2} \cos(\theta_2) + m_2 l_{c2}^2 + I_2 & m_2 l_{c2}^2 + I_2 \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix}$$

$$C = \begin{bmatrix} -2m_2 \dot{\theta}_2 l_1 l_{c2} \sin(\theta_2) + m_1 b_1 & -\dot{\theta}_2 m_2 l_1 l_{c2} \sin(\theta_2) \\ m_2 \dot{\theta}_1 l_1 l_{c2} \sin(\theta_2) & m_2 b_2 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

$$g = \begin{bmatrix} m_1 l_{c1} \sin \theta_1 + m_2 (l_1 \sin \theta_1 + l_{c2} \sin(\theta_1 + \theta_2)) \\ m_2 l_{c2} \sin(\theta_1 + \theta_2) \end{bmatrix}$$

$$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}$$

Modelo en espacio de estados:

Variables de estado: $(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2)$

Vector de estados:

$$x = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$



La ecuación en espacios de estados:

$$\dot{x} = f(x, t, u)$$

$$\dot{x} = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ M^{-1}(\mathbf{u} - (C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + g(\mathbf{q}))) \end{bmatrix}$$

Apéndice B: Códigos de simulaciones de los pendulos

.4. Simulación de la dinámica del péndulo simple

```
import os
import matplotlib.animation as animation
import numpy as np
import matplotlib.pyplot as plt

'''Función de límite de la señal de control'''
def limcontrol(u):
    if(u>=0):
        if(u>2.94):
            ur=2.94
        elif(u<=2.94):
            ur=u
    else:
        if(u>=-2.94):
            ur=u
        else:
            ur=-2.94
```

```

    return ur

'''Time parameters'''
dt = 0.005 # Tiempo de muestreo (5ms)
ti = 0.0 # Tiempo inicial de la simulación (0s)
tf = 10.0 # Tiempo final de la simulación (10s)
n = int((tf - ti) / dt) + 1 # Número de muestras
# Vector con los instantes de tiempo (en Matlab 0:0.005:10)
t = np.linspace(ti, tf, n)

'''Dynamic parameters'''
m = 0.5 # Masa del péndulo (kg)
l = 1.0 # Longitud de la barra del péndulo (m)
lc = 0.3 # Longitud al centro de masa del péndulo (m)
b = 0.05 # Coeficiente de fricción viscosa péndulo
g = 9.81 # Aceleración de la gravedad en la Tierra
I = 0.006 # Tensor de inercia del péndulo

'''State variables'''
x = np.zeros((n, 2))

'''Control vector'''
u = np.zeros((n, 1))

'''Initial conditions'''
x[0, 0] = 0 # Initial pendulum position (rad)
x[0, 1] = 0 # Initial pendulum velocity (rad/s)
ie_th = 0
ise=0#Inicial Jise actual
ise_next=0#Inicial Jise siguiente
iadu=0#Inicial Jiadu actual
iadu_next=0#Inicial Jiadu siguiente

'''State equation'''
xdot = [0, 0] #Ecuación de estado

```



```

'''Dynamic simulation'''
for i in range(n - 1):
    '''Current states'''
    th = x[i, 0] #Posición angular del péndulo
    th_dot = x[i, 1] #Velocidad angular del péndulo

    '''Controller'''
    e_th = np.pi-th #Error de posición
    e_th_dot = 0 - th_dot #Error de velocidad

    Kp = 9.00809903857079 #Ganancia proporcional
    Kd = 0.74331509706173#Ganancia derivativa
    Ki = 0# Ganancia integral

    u[i] = limcontro(Kp * e_th + Kd * e_th_dot + Ki * ie_th) #Ley
    de control

    '''System dynamics'''
    xdot[0] = th_dot
    xdot[1] = (u[i] - m * g * lc * np.sin(th) - b * th_dot) / (m *
        lc ** 2 + I)

    '''Integrate dynamics'''
    x[i + 1, 0] = x[i, 0] + xdot[0] * dt
    x[i + 1, 1] = x[i, 1] + xdot[1] * dt
    ie_th = ie_th + e_th * dt
    ise=ise_next+(e_th**2)*dt
    iadu=iadu_next+ (abs(u[i]-u[i-1]))*dt

    ise_next=ise
    iadu_next=iadu

u[n - 1] = u[n - 2]

```

Apéndice B: Códigos de simulaciones de los pendulos

898

```
print(x[:, 0]) #Posición del péndulo
print(ise) #Valor final de ISE
print(iadu) #Valor final de IADU

'''Plotting results'''
plt.figure(figsize=(12, 10))
plt.subplot(221)
plt.plot(t, x[:, 0], 'k', lw=1)
plt.legend([r'$\theta$'], loc=1)
plt.ylabel('Pendulum position')
plt.xlabel('Time')

plt.subplot(222)
plt.plot(t, x[:, 1], 'b', lw=1)
plt.legend([r'$\dot{\theta}$'], loc=1)
plt.ylabel('Pendulum speed')
plt.xlabel('Time')

plt.subplot(223)
plt.plot(t, u[:, 0], 'r', lw=2)
plt.legend([r'$u$'], loc=1)
plt.ylabel('Control signal')
plt.xlabel('Time')

plt.show()

'''Animation'''
plt.rcParams['animation.html'] = 'html5'

x0=np.zeros(len(t))
y0=np.zeros(len(t))

x1=l*np.sin(x[:,0])
y1=-l*np.cos(x[:,0])
```

```

fig = plt.figure(figsize=(8,6.4))
ax = fig.add_subplot(111, autoscale_on=False, \
                     xlim=(-1.8,1.8), ylim=(-1.2,1.2))
ax.set_xlabel('x')
ax.set_ylabel('y')

line, = ax.plot([],[],'o-',color='orange',lw=4, \
                 markerSize=6,markeredgecolor='k', \
                 markerfacecolor='k')

time_template = 't= %.1fs'
time_text = ax.text(0.05,0.9,'',transform=ax.transAxes)

def init():
    line.set_data([],[])
    time_text.set_text('')
    return line, time_text

def animate(i):

    line.set_data([x0[i],x1[i]],[y0[i],y1[i]])
    time_text.set_text(time_template % t[i])

    return line, time_text,

ani_a = animation.FuncAnimation(fig, animate, \
                                 np.arange(1,len(t)), \
                                 interval=40,blit=False,init_func=init)
plt.show()

```



5. Simulación de la dinámica del péndulo invertido

```

import os
import matplotlib.animation as animation
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import inv

'''Time parameters'''
dt = 0.005 # Tiempo de muestreo (5ms)
ti = 0.0 # Tiempo inicial de la simulación (0s)
tf = 10.0 # Tiempo inicial de la simulación (10s)
n = int((tf - ti) / dt) + 1 # Número de muestras
t = np.linspace(ti, tf, n) # Vector con los instantes de tiempo (en
                           # Matlab 0:0.005:10)

'''Dynamic parameters'''
m = 0.5 # Masa del péndulo (kg)
M = 0.7 # Masa del carro (kg)
l = 1.0 # Longitud de la barra del péndulo (m)
lc = 0.3 # Longitud al centro de masa del péndulo (m)
b1 = 0.05 # Coeficiente de fricción viscosa péndulo
b2 = 0.06 # Coeficiente de fricción del carro
g = 9.81 # Aceleración de la gravedad en la Tierra
I = 0.006 # Tensor de inercia del péndulo

'''State variables'''
z = np.zeros((n, 4))

'''Control vector'''
u = np.zeros((2, n))

'''Initial conditions'''
z[0, 0] = 0 # Posición inicial del carro (m)
z[0, 1] = 0 # Posición inicial del péndulo (rad)

```

.5 Simulación de la dinámica del péndulo invertido

☞

```
z[0, 2] = 0 # Velocidad inicial del carro (m/s)
z[0, 3] = 0 # Velocidad angular del péndulo (rad/s)
ie_th = 0 #Inicialización de error integral de posición de carro
ie_x = 0#Inicialización de error integral de posición de péndulo

'''State equation'''
zdot = [0, 0, 0, 0]

'''Dynamic simulation'''
for i in range(n - 1):
    '''Current states'''
    x = z[i, 0] # Posicion del carro
    th = z[i, 1] # Posición del péndulo
    x_dot= z[i, 2] # Velocidad del carro
    th_dot = z[i, 3] # Velocidad del péndulo

    '''Controller'''
    e_x = 0 - x #Error de posición de carro
    e_x_dot = 0 - x_dot #Error de velocidad de carro
    e_th = np.pi/2-th #Error de posición angular
    e_th_dot = 0 - th_dot #Error de velocidad angular

    '''Ganancias del controlador del carro'''
    Kp = 0
    Kd = 0
    Ki = 0

    '''Ganancias del controlador del péndulo'''
    Kp1 =0
    Kd1 = 0
    Ki1 = 0

    u[0, i] = Kp * e_x + Kd * e_x_dot + Ki * ie_x #Señal de control
    del actuador del carro
    u[1, i] = Kp1 * e_th + Kd1 * e_th_dot + Ki1 * ie_th #Señal de
```



control del actuador del péndulo

```

MI = np.array([[M + m, -m * lc * np.sin(th)], [-m * lc * np.sin(
    th), I + m * lc ** 2]]) # Matriz de inercia
MC = np.array([[b1, -m * lc * np.cos(th) * th_dot], [0, b2]]) # Matriz de Coriolis
MG = np.array([[0], [m * g * lc * np.cos(th)]]) # Vector de gravedad

array_dots = np.array([[x_dot], [th_dot]])#Vector de velocidades
MC2 = np.dot(MC, array_dots)

ua = np.array([[u[0, i]], [u[1, i]]])
aux1 = ua - MC2 - MG
Minv = inv(MI)
aux2 = np.dot(Minv, aux1) #Variables de segundo grado /(doble derivada)

'''System dynamics'''
zdot[0] = x_dot #Velocidad del carro
zdot[1] = th_dot #Velocidad del péndulo
zdot[2] = aux2[0, :] #Aceleración del carro
zdot[3] = aux2[1, :] #Aceleración del péndulo

'''Integrate dynamics'''
z[i + 1, 0] = z[i, 0] + zdot[0] * dt
z[i + 1, 1] = z[i, 1] + zdot[1] * dt
z[i + 1, 2] = z[i, 2] + zdot[2] * dt
z[i + 1, 3] = z[i, 3] + zdot[3] * dt
ie_th = ie_th + e_th * dt

```



```

ie_x = ie_x+e_x*dt

u[:,n - 1] = u[:,n - 2] #Actualizar señal de control

print(z[:, 0])

'''Plotting results'''
plt.figure(figsize=(12, 10))
plt.subplot(321)
plt.plot(t, z[:, 0], 'k', lw=1)
plt.legend([r'$x$'], loc=1)
plt.ylabel('Car position')
plt.xlabel('Time')

plt.subplot(322)
plt.plot(t, z[:, 1], 'b', lw=1)
plt.legend([r'$\theta$'], loc=1)
plt.ylabel('Pendulum position')
plt.xlabel('Time')

plt.subplot(323)
plt.plot(t, z[:, 2], 'r', lw=2)
plt.legend([r'$\dot{x}$'], loc=1)
plt.ylabel('Car speed')
plt.xlabel('Time')

plt.subplot(324)
plt.plot(t, z[:, 3], 'b', lw=2)
plt.legend([r'$\dot{\theta}$'], loc=1)
plt.ylabel('Pendulum speed')
plt.xlabel('Time')

plt.subplot(325)
plt.plot(t, u[0, :], 'r', lw=2)
plt.legend([r'$u_{car}$'], loc=1)

```

```

plt.ylabel('u_{car}')
plt.xlabel('Time')

plt.subplot(326)
plt.plot(t, u[1, :], 'k', lw=2)
plt.legend([r'$u_{\text{pendulum}}$'], loc=1)
plt.ylabel('u_pendulum')
plt.xlabel('Time')

plt.show()

'''Animation'''
x1 = z[:, 0]
y1 = np.zeros(len(t))

# suppose that l = 1
x2 = l * np.cos(z[:, 1]) + x1
y2 = l * np.sin(z[:, 1])

fig = plt.figure(figsize=(8, 6.4))
ax = fig.add_subplot(111, autoscale_on=False,
                     xlim=(-2.5, 5), ylim=(-2.2, 2.2))
ax.set_xlabel('position')
ax.get_yaxis().set_visible(True)

mass1, = ax.plot([], [], linestyle='None', marker='s',
                  markersize=10, markeredgecolor='k',
                  color='green', markeredgewidth=2)

line, = ax.plot([], [], 'o-', color='green', lw=4,
                 markersize=6, markeredgecolor='k',
                 markerfacecolor='k')
time_template = 't= %.1fs'
time_text = ax.text(0.05, 0.9, '', transform=ax.transAxes)

```

```

def init():

    mass1.set_data([], [])
    line.set_data([], [])
    time_text.set_text('')

    return line, mass1, time_text


def animate(i):

    mass1.set_data([x1[i]], [y1[i]])
    line.set_data([x1[i], x2[i]], [y1[i], y2[i]])
    time_text.set_text(time_template % t[i])
    return mass1, line, time_text


ani_a = animation.FuncAnimation(fig, animate, \
                                np.arange(1, len(t)), \
                                interval=1, blit=False, init_func=
                                init)
plt.show()

```

.6. Simulación de la dinámica del péndulo doble

```

import os
import matplotlib.animation as animation
import numpy as np
import matplotlib.pyplot as plt
from scipy import linalg

'''Time parameters''' #Parametros temporales
dt = 0.01 # Tiempo de muestreo (5ms)
ti = 0.0 # Tiempo inicial de la simulación (0s)
tf = 12.25 # Tiempo final de la simulación (12.25s)

```

Apéndice B: Códigos de simulaciones de los pendulos

898

```

n = int((tf - ti) / dt) + 1 # Número de muestras
t = np.linspace(ti, tf, n) # Vector con los instantes de tiempo (en
                           Matlab 0:0.005:10)

'''Dynamic parameters''' #Parametros dinamicos
m1 = 0.5 # Masa de la barra 1(kg)
m2= 0.5 #Masa de la barra 2 (kg)
l1 = 1.0 # Longitud de la barra 1 (m)
lc1 = 0.3 # Longitud al centro de masa de la barra 2 (m)
l2= 1.0 #Longitud de la barraa 2 (m)
lc2=0.3 #Longitud al centro de masa de la barra 2(m)
b1 = 0.05 # Coeficiente de fricción viscosa de la barra 1
b2= 0.02 #Coeficiente de fricción viscosa de la barra 2
g = 9.81 # Aceleración de la gravedad en la Tierra
I1 = 0.006 # Tensor de inercia del péndulo 1
I2= 0.004 #Tensor de inercia del péndulo 2

'''State variables'''#Variables de estado
x = np.zeros((n, 4))

'''Control vector'''#Señales de control
u = np.zeros((2,n))

'''Initial conditions'''#Condiciones iniciales
x[0, 0] = np.pi/2# Initial pendulum position 1 (rad)
x[0, 1] =0 # Initial pendulum position 2 (rad)
x[0, 2]=0 # Initial pendulum velocity (rad/s)
x[0, 3]=0 # Initial pendulum velocity (rad/s)

'''State equation'''#Ecuacion de estado
xdot = [0, 0, 0, 0]

'''Dynamic simulation'''
for i in range(n - 1):
    '''Current states'''
```



```

th1 = x[i, 0]
th2 = x[i, 1]
th1_dot=x[i,2]
th2_dot=x[i,3]
'''Propiedades del modelo dinámico'''
#Matriz de Inercia
M=np.array([[((m1*lc1)**2)+I1+I2+m2*((l1**2)+(lc2**2)+(2*l1*lc2*
    np.cos(th2))), (m2*lc2**2)+I2+m2*l1*lc2*np.cos(th2)],[((m2*lc2
    **2)+I2+m2*l1*lc2*np.cos(th2), (m2*lc2**2)+I2)])
#Fuerzas centripeta y de Coriolis
C=np.array([[[-2*m2*l1*lc2*th2_dot*np.sin(th2) + m1*b1, -m1*l1*
    lc2*np.sin(th2)*th2_dot],[m2*l1*lc2*th1_dot*np.sin(th2), m2
    *b2]]])
#Aporte gravitacional
gra=np.array([[m1*lc1*np.sin(th1)+m2*((l1*np.sin(th1))+lc2*np.
    sin(th1+th2))],[m2*lc2*np.sin(th1+th2)]])
#Vector de velocidades
v=np.array([[th1_dot],[th2_dot]])
C2=np.dot(C,v)
ua=np.array([[u[0,i]],[u[1,i]]])
aux1=ua-C2-gra
Minv=linalg.inv(M)
#Vector de aceleraciones
aux2=np.dot(Minv,aux1)
'''System dynamics'''
xdot[0] = th1_dot
xdot[1]= th2_dot
xdot[2]=aux2[0,:]
xdot[3]=aux2[1,:]
'''Integrate dynamics'''
x[i + 1, 0] = x[i, 0] + xdot[0] * dt
x[i + 1, 1] = x[i, 1] + xdot[1] * dt
x[i + 1, 2] = x[i, 2] + xdot[2] * dt
x[i + 1, 3] = x[i, 3] + xdot[3] * dt

```

Apéndice B: Códigos de simulaciones de los pendulos

898

```
print(x[:, 0])
print(x[:, 1])

'''Plotting results'''
plt.figure(figsize=(12, 10))
plt.subplot(321)
plt.plot(t, x[:, 0], 'k', lw=1)
plt.legend([r'$\theta_1$'], loc=1)
plt.ylabel('Pendulum position_1')
plt.xlabel('Time')

plt.subplot(322)
plt.plot(t, x[:, 1], 'b', lw=1)
plt.legend([r'$\theta_2$'], loc=1)
plt.ylabel('Pendulum position_2')
plt.xlabel('Time')

plt.subplot(323)
plt.plot(t, x[:, 2], 'c', lw=2)
plt.legend([r'$\dot{\theta}_1$'], loc=1)
plt.ylabel('Pendulum speed_1')
plt.xlabel('Time')

plt.subplot(324)
plt.plot(t, x[:, 3], 'g', lw=2)
plt.legend([r'$\dot{\theta}_2$'], loc=1)
plt.ylabel('Pendulum speed_2')
plt.xlabel('Time')

plt.subplot(325)
plt.plot(t, u[0, :], 'g', lw=2)
plt.legend([r'$u_1$'], loc=1)
plt.ylabel('$Control signal_1$')
plt.xlabel('Time')
```

```

plt.subplot(326)
plt.plot(t, u[1, :], 'g', lw=2)
plt.legend([r'$u_2$'], loc=1)
plt.ylabel('Control signal_2')
plt.xlabel('Time')

'''Animación'''

x0=np.zeros(len(t))
y0=np.zeros(len(t))

x1=l1*np.sin(x[:,0])
y1=-l1*np.cos(x[:,0])

x2=l2*np.sin(x[:,0])+ l2*np.sin(x[:,0]+x[:,1])
y2=-l2*np.cos(x[:,0])- l2*np.cos(x[:,0]+x[:,1])

fig = plt.figure(figsize=(8,6.4))
ax = fig.add_subplot(111, autoscale_on=False,
                     xlim=(-2.8,2.8), ylim=(-2.2,2.2))
ax.set_xlabel('position')

line, = ax.plot([],[],'o-',color='blue',lw=4,
                markersize=6,markeredgecolor='k',
                markerfacecolor='k')
line1, = ax.plot([],[],'o-',color='blue',lw=4,
                 markersize=6,markeredgecolor='k',
                 markerfacecolor='k')
time_template = 't= %.1fs'
time_text = ax.text(0.05,0.9,'',transform=ax.transAxes)

def init():
    line.set_data([],[])

```



```

        line1.set_data([],[])
        time_text.set_text('')

    return line, time_text, line1,
}

def animate(i):

    line.set_data([x0[i],x1[i]],[y0[i],y1[i]])
    line1.set_data([x1[i],x2[i]],[y1[i],y2[i]])
    time_text.set_text(time_template % t[i])

    return line, time_text, line1,
}

ani_a = animation.FuncAnimation(fig, animate, \
    np.arange(1,len(t)), \
    interval=100,blit=False,init_func=init)

plt.show()

```

.7. Algoritmo de control en Arduino

```

#include <PID_v1.h>
#include <TimerOne.h> /
/*-----Variables para Impresión en
consola-----*/
byte cmd = 0; // Use for serial
communication.
byte flags; // Flag for print values
in the serial monitor
/*-----Variables for LM298N
-----*/
int IN1 = 11;
int IN2 = 10;
int PWM1 = 9;

```



```

int forWARDS = 1;
int backWARDS = 0;
float start = 0;

/*-----Variables for incremental encoder
-----*/
volatile long contador = 0;
byte ant = 0;
byte act = 0;
const byte encA = 2; // Signal for channel
A
const byte encB = 3; // Signal for channel
B
int MIN_MAX_POST = 300; // Limit the maximum
position
/*-----We defined variables for PID
algorithm-----*/
TimerOne PID_compute; // Timer
double Setpoint, Input, Output;
double SampleTime = 100; // time in mili
seconds, RUN at 160MHZ ESP8266
//double Kp=5, Ki=2, Kd=0.001;
//double Kp=5.5, Ki=3.6, Kd=0.002;
double Kp=5.5, Ki=3.6, Kd=0.002; // PID gain
PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);

/*----- Interruption Function
-----*/
void PID_DCmotor_interrup(){
    //Serial.print("t: ");Serial.println(millis()-start) ;
    //start = millis();
    myPID.Compute(); // Calculus for PID algorithm
    RunMotor(Output); // PWM order to DC driver
}

```

Apéndice B: Códigos de simulaciones de los pendulos

898

```
/*-----SETUP-----*/  
  
*/  
void setup(){  
    Serial.begin(9600);  
    //Iniciando L298N  
    digitalWrite(IN1, LOW);  
    digitalWrite(IN2, LOW);  
    analogWrite(PWM1, LOW);  
    pinMode(IN1, OUTPUT);  
    pinMode(IN2, OUTPUT);  
    pinMode(PWM1, OUTPUT);  
  
    Setpoint = 100.0; // Init in position 0.0  
    //RUN the PID  
    myPID.SetMode(AUTOMATIC);  
    // Max Min values for PID algorithm  
    myPID.SetOutputLimits(-1023,1023);  
    // Sample Time for PID  
    myPID.SetSampleTime(SampleTime);  
  
    // Initializing Interruptions  
    PID_compute.initialize(SampleTime);  
    PID_compute.attachInterrupt(PID_DCmotor_interrup);  
    Serial.begin(9600);  
  
    // Pin Interruption  
    attachInterrupt(digitalPinToInterrupt(encA), encoder, CHANGE); //  
        rising and falling flank  
    attachInterrupt(digitalPinToInterrupt(encB), encoder, CHANGE); //  
        rising and falling flank  
}  
  
/*-----LOOP-----*/
```



```

/*
void loop(){
    Serial.print("PWM   :");Serial.print(Output);
    Serial.print(" |  contador   :");Serial.print(contador);
    Serial.print(" |  Setpoint   :");Serial.println(Setpoint);
    // Protection code
    //limit_pos();
    // Ask for input data for change PID gain or setpoint
    //input_data();
}

/*
-----
*/
// Function for run the motor, backward, forward or stop
void RunMotor(double Usignal){
    if (Setpoint-Input==0){
        shaftrev(IN1,IN2,PWM1,backWARDS, 0);
        //Serial.print("cero");
    }else if(Usignal>=0){
        shaftrev(IN1,IN2,PWM1,backWARDS, Usignal);
    }else{
        shaftrev(IN1,IN2,PWM1,forWARDS, -1*Usignal);
    }
}

// Function that set DC_driver to move the motor
void shaftrev(int in1, int in2, int PWM, int sentido,int Wpulse){
    if(sentido == 0){ //backWARDS
        digitalWrite(in2, HIGH);
        digitalWrite(in1, LOW);
        analogWrite(PWM,Wpulse);
    }
    if(sentido == 1){ //forWARDS
        digitalWrite(in2, LOW);
    }
}

```

```
digitalWrite(in1, HIGH);
analogWrite(PWM,Wpulse);
}

void encoder(void){
//Serial.println(ant);

ant=act; // Saved act (current read)

in ant (last read)

act = digitalRead(encA)<<1|digitalRead(encB);

if(ant==0 && act==1) contador++; // Increase the counter
for forward movement
else if(ant==1 && act==3) contador++;
else if(ant==3 && act==2) contador++;
else if(ant==2 && act==0) contador++;
else contador--; // Reduce the counter
for backward movement

// Enter the counter as input for PID algorith
Input=contador;
}
```

Apéndice C: Sintonización multi-objetivo.

.8. Péndulo simple

DE

```
import random
import numpy as np
import math
import matplotlib.pyplot as plt
import serial
from drawnow import *

#-----Parametros DE-----
limit=[(0,10),(0,10),(0,10)] #Limites inferior y
superior
poblacion = 200 # Tamaño de la población, mayor
>= 4
f_mut = 0.5 # Factor de mutacion [0,2]
recombinacion = 0.7 # Tasa de recombinacion [0,1]
generaciones =100 # Número de generaciones
D = 3 # Dimensionalidad O número de variables de diseño
M = 2 # Numero de objetivos
AMAX = 30 # Numero maximo de soluciones en el archivo
```

```

#
-----Función de dominancia-----
def dominates(_a, _b):
    for _j in range(M):           #Recorre el vector J de
        funciones objetivo
        if _b[_j] < _a[_j]:
            return False           #Regresa False si a domina b,
        en este caso seleccionamos b
    return True                    #Regresa True si b domina a,
    en este caso seleccionamos a
#
-----Función de límite del actuador
def limcontrol(u):
    if(u>=0):
        if(u>2.94):
            ur=2.94
        elif(u<=2.94):
            ur=u
    else:
        if(u>=-2.94):
            ur=u
        else:
            ur=-2.94
    return ur
#

```



```

#-----Problema de optimización-----
def pendulum_s(r):
    '''Time Parameters'''
    dt = 0.005 # Tiempo de muestreo (5ms)
    ti = 0.0 # Tiempo inicial de la simulación (0s)
    tf = 10.0 # Tiempo inicial de la simulación (10s)
    n = int((tf - ti) / dt) + 1 # Número de muestras
    t = np.linspace(ti, tf, n) # Vector con los instantes de tiempo
    (en Matlab 0:0.005:10)

    '''Dynamics Parameters'''
    m = 0.5 # Masa del péndulo (kg)
    l = 1.0 # Longitud de la barra del péndulo (m)
    lc = 0.3 # Longitud al centro de masa del péndulo (m)
    b = 0.05 # Coeficiente de fricción viscosa péndulo
    g = 9.81 # Aceleración de la gravedad en la Tierra
    I = 0.006 # Tensor de inercia del péndulo

    '''State variables'''
    x = np.zeros((n, 2))

    '''Control vector'''
    u = np.zeros((n, 1))

    ise=0
    ise_next=0
    iadu=0
    iadu_next=0

    '''Initial conditions'''
    x[0, 0] = 0 # Initial pendulum position (rad)
    x[0, 1] = 0 # Initial pendulum velocity (rad/s)

```

```

ie_th = 0

'''State equation'''
xdot = [0, 0]

'''Dynamic simulation'''
for o in range(n - 1):
    '''Current states'''
    th = x[o, 0]
    th_dot = x[o, 1]
    e_th = np.pi - th
    e_th_dot = 0 - th_dot

    '''Controller'''
    Kp = r[0]
    Kd = r[1]
    Ki = r[2]

    u[o, 0] = limcontro(Kp * e_th + Kd * e_th_dot + Ki * ie_th)

'''System dynamics'''

xdot[0] = th_dot
xdot[1] = (u[o] - m * g * lc * np.sin(th) - b * th_dot) / (
    m * lc ** 2 + I)

'''Integrate dynamics'''
x[o + 1, 0] = x[o, 0] + xdot[0] * dt
x[o + 1, 1] = x[o, 1] + xdot[1] * dt
ie_th = ie_th + e_th * dt

ise=ise_next+(e_th**2)*dt
iadu=iadu_next+ (abs(u[o]-u[o-1]))*dt

```



```

        if (ise>=20):
            ie=20
        else:

            ie=ise
            if (iadu>=0.8):
                ia=0.8
            else:
                ia=iadu

            ise_next=ie
            iadu_next=ia
            #print(u[o,o])

        return np.array([ise_next, iadu_next])
#-----Asegurar limites de caja
-----def asegurar_limites(vec, limit):

    vec_new = []
    # ciclo que recorren todos los individuos
    for i in range(len(vec)):

        # Si el individuo sobrepasa el límite mínimo
        if vec[i] < limit[i][0]:
            vec_new.append(limit[i][0])

        # Si el individuo sobrepasa el límite máximo
        if vec[i] > limit[i][1]:

```

Apéndice C: Sintonización multi-objetivo.

```
❀❀❀
```

```
    vec_new.append(limit[i][1])

    # Si el individuo está dentro de los límites
    if limit[i][0] <= vec[i] <= limit[i][1]:
        vec_new.append(vec[i])

    return vec_new
#-----Funcion main, DE
-----#
def main(function, limites, poblacion, f_mut, recombination,
generaciones):
    #----Poblacion
    -----#  

    population = np.zeros((generaciones,poblacion, D)) #poblacion
    actual
    population_next= np.zeros((generaciones,poblacion, D)) #
    poblacion siguiente
    #
    -----#
#-----F(x)
    -----#  

f_x = np.zeros((generaciones,poblacion, M)) # Valor de funcion
    objetivo de poblacion actual
f_x_next = np.zeros((generaciones,poblacion, M)) # Valor de
```



funcion objetivo de poblacion siguiente

#

```
#----- Inicialización de la población
-----
for i in range(0,poblacion): # cambiar tam_poblacion
    indv = []
    for j in range(len(limites)):
        indv.append(random.uniform(limites[j][0],limites[j][1]))
    #print(indv[0])
    population[0][i]=indv[0]
    population_next[0][i]=indv[0]

#print(population[0,:])
```

#

```
#----- Evaluación población
0-----
```

```
for i, xi in enumerate(population[0,:]): # Evalua objetivos

    f_x[0][i] = function(xi)
```

#

```
#-----Ciclo evolutivo-----  
  
for i in range(0,generaciones-1):  
    print ('Generación:',i)  
    for j in range(0, poblacion):  
  
        #Mutacion  
        # Seleccionamos 4 posiciones de vector aleatorios,  
        #range = [0, poblacion)  
        candidatos = range(0,poblacion)  
        random_index = random.sample(candidatos, 4)  
  
        r1 = random_index[0]  
        r2 = random_index[1]  
        r3 = random_index[2]  
  
        while r1 == j:  
            t=random.sample(candidatos, 1)  
            r1 = t[0]  
  
        while r2 == r1 or r2 == j:  
            t2=random.sample(candidatos,1)  
            r2=t2[0]  
  
        while r3 == r2 or r3 == r1 or r3 == j:  
            t3 =random.sample(candidatos, 1)  
            r3=t3[0]  
  
        x_1 = population[i][r1]  
        x_2 = population[i][r2]
```



```

x_3 = population[i][r3]
x_t = population[i][j]

# Restamos x3 de x2, y creamos un nuevo vector (x_diff)
x_diff =[x_2_i - x_3_i for x_2_i, x_3_i in zip(x_2, x_3
    )]

# Multiplicamos x_diff por el factor de mutacion(F) y
# sumamos x_1
v_mutante = [x_1_i + f_mut * x_diff_i for x_1_i,
    x_diff_i in zip(x_1, x_diff)]
v_mutante = asegurar_limites(v_mutante, limites)

#Vector hijo
v_hijo = np.copy(population[i][j])
jrand = random.randint(0, D)

for k in range(len(x_t)):
    crossover = random.uniform(0, 1)
    if crossover <= recombination or k == jrand:
        v_hijo[k]=v_mutante[k]
    else:
        v_hijo[k]=x_t[k]

#
# Evalua descendiente
f_ui = function(v_hijo)

# Selecciona el individuo que pasa a la siguiente
# generacion
if dominates(f_ui, f_x[i][j]):
    f_x_next[i][j] = np.copy(f_ui)
    population_next[i][j] = np.copy(v_hijo)
elif dominates(f_x[i][j], f_ui):

```



```

f_x_next[i][j] = np.copy(f_x[i][j])
population_next[i][j] = np.copy(population[i][j])

else:
    if random.uniform(0, 1) < 0.5:
        f_x_next[i][j] = np.copy(f_ui)
        population_next[i][j] = np.copy(v_hijo)

    else:
        f_x_next[i][j] = np.copy(f_x[i][j])
        population_next[i][j] = np.copy(population[i][j])

# Una vez que termina la generación actualizo x y f_x
f_x[i+1] = np.copy(f_x_next[i])
population[i+1] = np.copy(population_next[i])

# Filtrado no dominado
print(i)
f_x_fil = np.empty((0, M)) # Conjunto no dominado
population_fil = np.empty((0, D)) # Conjunto no dominado
population_fil2 = np.empty((0, D)) # Conjunto no dominado
for i1, f_x_1 in enumerate(f_x[i,:,:]):
    sol_nd = True

    for i2, f_x_2 in enumerate(f_x[i,:,:]):
        if i1 != i2:
            if dominates(f_x_2, f_x_1):
                sol_nd = False

        break
    if sol_nd:

        f_x_fil = np.append(f_x_fil, [f_x_1], axis=0)

        population_fil = np.append(population_fil, [population[i+1]])

```



```

    i ,i1 ,:]], axis=0)

print(f_x_fil)
print(population_fil)
print(len(f_x_fil))

#-----Guardar en archivo excel
-----

filename="fx1fill2.csv"
myFile=open(filename,'w')
myFile.write("kp,kd,ki,f1, f2 \n")
for l in range(len(f_x_fil)):
    myFile.write(str(population_fil[l, 0])+", "+str(
        population_fil[l, 1])+", "+str(population_fil[l, 2])+",
        str(f_x_fil[l, 0])+", "+str(f_x_fil[l, 1])+"\n")
myFile.close()
#-----Gráfica del Frente de Pareto
-----

plt.figure(1)
plt.title('Aproximacion al frente de Pareto')
plt.scatter(f_x_fil[:, 0], f_x_fil[:, 1])
plt.xlabel('f1')
plt.ylabel('f2')
plt.show()
return f_x_fil

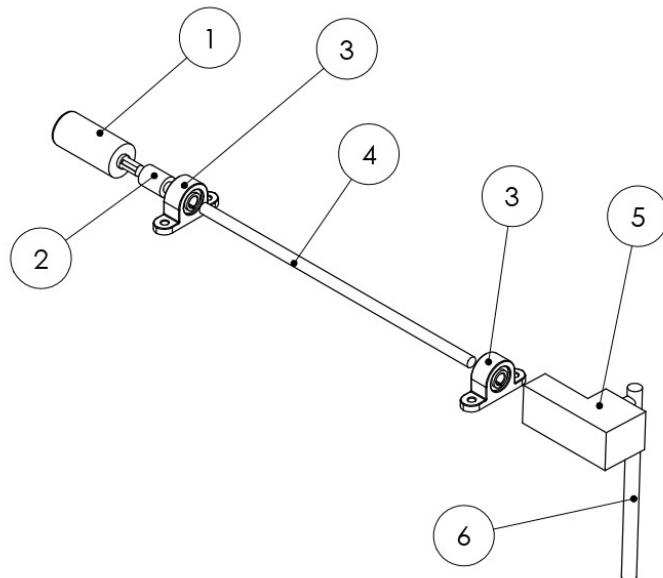
#llamado de la función main de DE
var=main(pendulum_s, limit, poblacion, f_mut, recombination,
generaciones)

```


Anexos



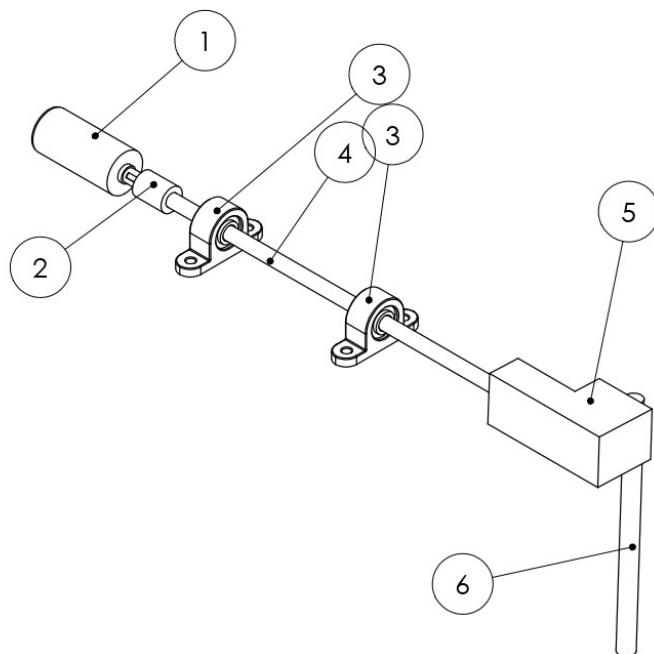
Plano de montaje



N.º DE ELEMENTO	N.º DE PIEZA	MATERIAL	CANTIDAD
1	MOTOR	ACERO	1
2	COPE	ALUMINIO	1
3	CHUMACERA	HIERRO COLADO	2
4	EJE	ACERO	1
5	BLOQUE	NYLAMID	1
6	BARRA PÉNDULO	ALUMINIO	1
NOMBRE	PÉNDULO SIMPLE		
AUTOR	CORTÉZ CONDE ALEXANDER & VALDEZ CRUZ MARCO ANTONIO		

Figura 4: Plano de montaje.

Plano de conjunto



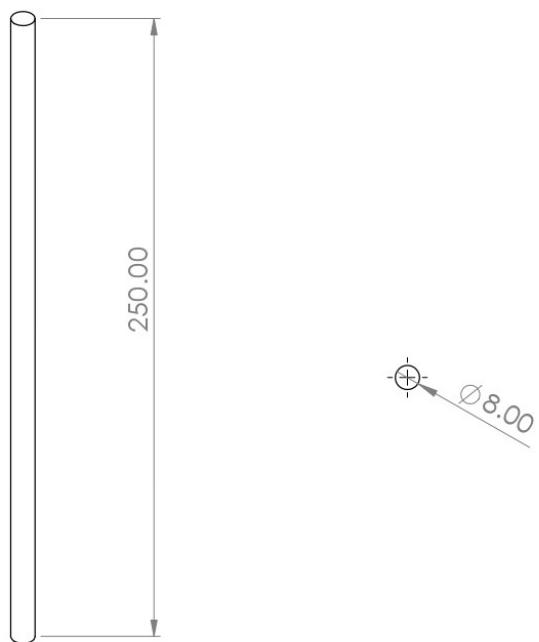
N.º DE ELEMENTO	N.º DE PIEZA	MATERIAL	CANTIDAD
1	MOTOR	ACERO	1
2	COPLE	ALUMINIO	1
3	CHUMACERA	HIERRO COLADO	2
4	EJE	ACERO	1
5	BLOQUE	NYLAMID	1
6	BARRA PÉNDULO	ALUMINIO	1
NOMBRE			
PÉNDULO SIMPLE			
AUTOR			
CORTÉZ CONDE ALEXANDER & VALDEZ CRUZ MARCO ANTONIO			

Figura 5: Plano de conjunto.



Plano de despiece

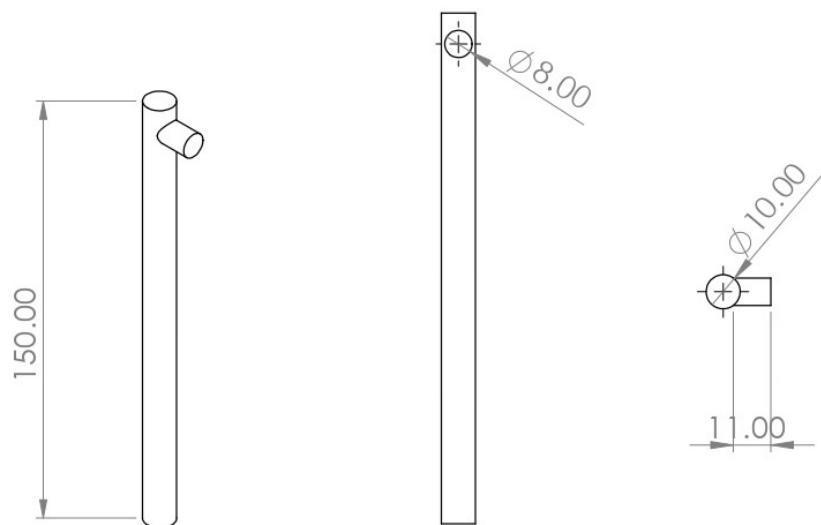
Eje



N.º DE ELEMENTO	N.º DE PIEZA	MATERIAL	CANTIDAD
4	EJE	ALUMINIO	1

Figura 6: Plano del eje.

Barra

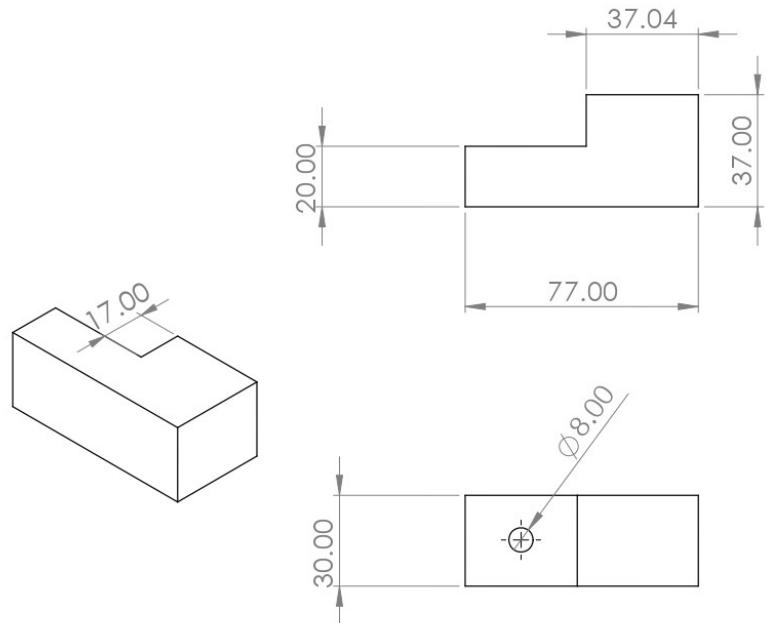


N.º DE ELEMENTO	N.º DE PIEZA	MATERIAL	CANTIDAD
6	BARRA	ALUMINIO	1

Figura 7: Plano de la barra.



Bloque



N.º DE ELEMENTO	N.º DE PIEZA	MATERIAL	CANTIDAD
5	BLOQUE	NYLAMID	1

Figura 8: Plano del bloque



Interfaz gráfica

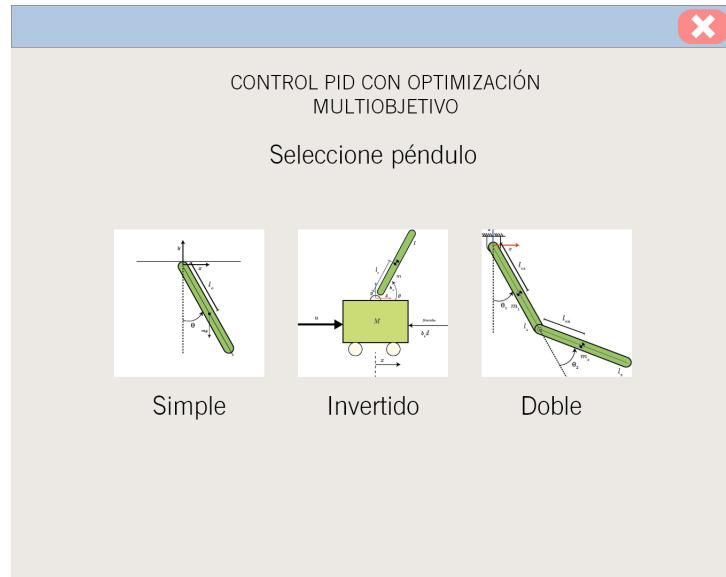


Figura 9: Pantalla de inicio.



Figura 10: Ingreso de datos péndulo simple.

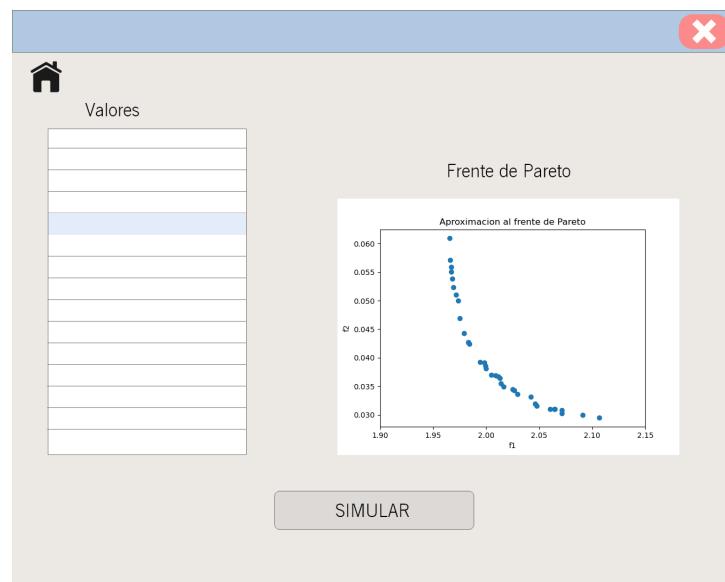


Figura 11: Frente de Pareto péndulo simple.

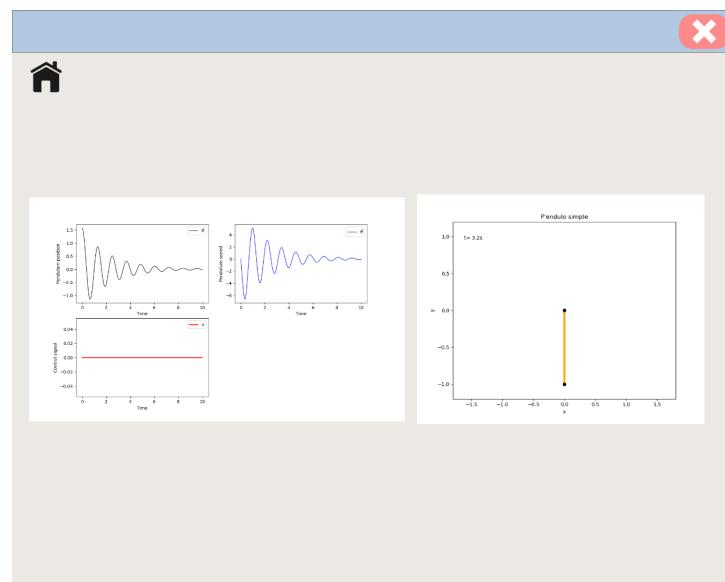


Figura 12: Simulación dinámica péndulo invertido.

**CONTROL PID CON OPTIMIZACIÓN
MULTIOBJETIVO**

Invertido

Ingrese los siguientes datos

Masa (m)	<input type="text"/>
Longitud (l)	<input type="text"/>
Fricción viscosa (b1)	<input type="text"/>
Fricción carro (b2)	<input type="text"/>
Tensor de inercia del péndulo (I)	<input type="text"/>
Set Point	<input type="text"/>
Condiciones iniciales	<input type="text"/>

Seleccione algoritmo metaheurístico

DE GA
 PSO

Figura 13: Ingreso de datos péndulo invertido.

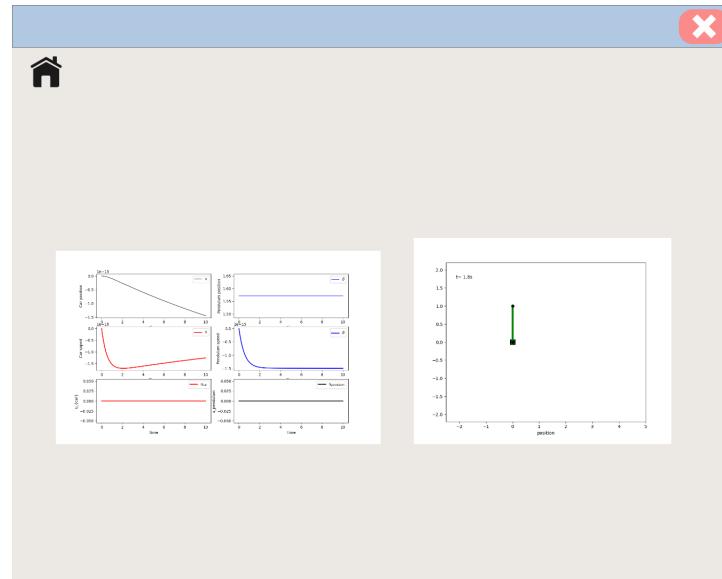


Figura 14: Simulación dinámica péndulo invertido.

CONTROL PID CON OPTIMIZACIÓN MULTIOBJETIVO

Doble

Ingrese los siguientes datos	Seleccione algoritmo metaheurístico
Masa barra 1 (m1)	<input type="text"/>
Masa barra 2 (m2)	<input type="text"/>
Longitud barra 1 (l1)	<input type="text"/>
Longitud centro de masa barra 1 (lc1)	<input type="text"/>
Longitud barra 1 (l2)	<input type="text"/>
Longitud centro de masa barra 1 (lc2)	<input type="text"/>
Fricción barra 1 (b1)	<input type="text"/>
Fricción barra 2 (b2)	<input type="text"/>
Inercia barra 1 (I1)	<input type="text"/>
Inercia barra 2 (I2)	<input type="text"/>

DE GA PSO

Figura 15: Ingreso de datos péndulo doble.

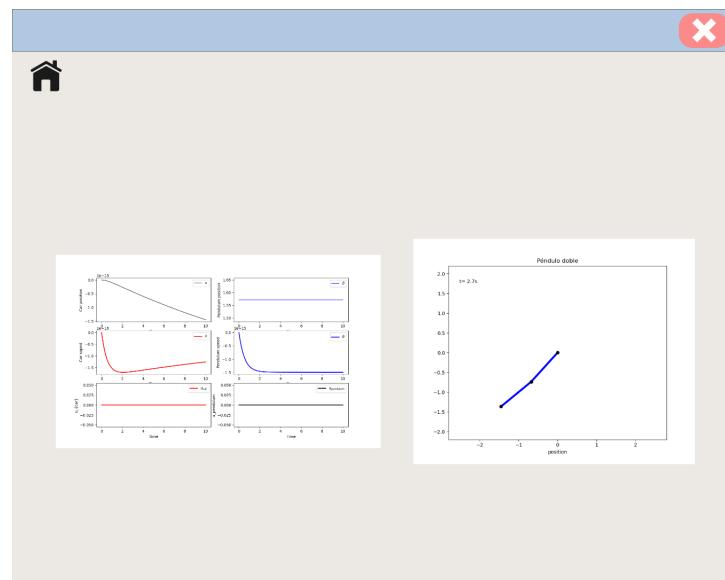


Figura 16: Simulación dinámica péndulo doble.

Glosario
