Modelamiento de Sistemas Dinámicos de Tipo Mecánico Basado en Euler Lagrange e Identificación de Parámetros Disipativos Mediante Algoritmos Evolutivos

Johan Sebastian Becerra Nova y Marco Tulio Mancipe Silva

Trabajo de Grado para Optar al Título de Ingeniero Mecánico

Director
Jabid Eduardo Quiroga Méndez
PhD Ingeniería Civil

Universidad Industrial de Santander
Facultad de Ingenierías Físico-Mecánicas
Escuela de Ingeniería Mecánica
Bucaramanga
2021

Dedicatoria

A mi familia en especial a mis padres y hermanos por estar siempre apoyándome, para que hoy uno de mis sueños se realidad.

A mis amigos y compañeros que en algún momento de mi vida fueron fuente de apoyo y alegrías.

A mi hermano Luis Fernando Mancipe por alentarme a conseguir cada uno de mis objetivos, por brindarme todo su apoyo y por brindarme sus consejos.

¡Gracias!

Marco Tulio Mancipe Silva

Dedicatoria

A mis padres y a mi hermana, que siempre estuvieron presentes en cada etapa de mi carrera, quienes siempre confiaron en mis capacidades, y sin condición alguna me brindaron su apoyo.

A mis amigos Osmar Fabian Barón, Daniel Cárdenas, Mary Luz Gil, Enrique Ortega, Marco Mancipe Silva, quienes me brindaron apoyo y alegría en el transcurso de mi vida universitaria.

A mi tío Luis Becerra Aguilar, quien confío en mis capacidades y me brindo su apoyo.

¡Gracias!

Johan Sebastian Becerra Nova

Agradecimientos

A la Universidad Industrial de Santander y a cada uno de los profesores de la Escuela de Ingeniería Mecánica que hicieron parte de nuestra formación integral y académica.

Agradecemos a nuestro director de trabajo de grado Jabid Eduardo Quiroga Méndez, por brindarnos todo su apoyo y asesoría en la realización de este proyecto de grado.

Agradecemos de manera general a cada una de las personas que de alguna manera aportaron su apoyo para que hoy nosotros podamos culminar esta etapa de nuestras vidas.

.

Tabla de Contenido

Pág.

Lista de Tablas

Pág.

Lista de Figuras

Pág.

Lista de Apéndices

Los apéndices están adjuntos y puede visualizarlos en la base de datos de la biblioteca UIS

Resumen

Título: Modelamiento de Sistemas Dinámicos de Tipo Mecánico Basado en Euler Lagrange e Identificación de Parámetros Disipativos Mediante Algoritmos Evolutivos[*]

Autor: Johan Sebastián Becerra Nova, Marco Tulio Mancipe Silva[**]

Palabras Clave: Modelos Matemáticos, Algoritmos Evolutivos, Sistemas dinámicos, programación.

Descripción: Con el paso de los años la educación se ha ido adaptando a los diferentes escenarios, integrando diferentes estrategias de enseñanza, que permiten un mejoramiento continuo en la formación de personas. Este proyecto busca desarrollar un programa en MATLAB y Python para modelamiento matemático, linealización, e identificación de parámetros disipativos de sistemas dinámicos de tipo mecánico, lineales o no lineales. Proporcionando apoyo a la catedra docente, brindando una herramienta software al profesor y estudiante de la asignatura sistemas dinámicos, que hace uso del cálculo simbólico, metodología de Euler Lagrange, series de Taylor y Algoritmos Evolutivos en MATLAB y Python. En la actualidad los recientes desarrollos tecnológicos permiten la adquisición de datos, procesamiento y tratamiento de señales, por lo tanto, este trabajo centra la atención en el modelamiento de sistemas de tipo mecánico lineales o no lineales para luego identificar sus parámetros de tipo disipativo, para obtener un modelo matemático ajustado que represente de manera real cada sistema mecánico. En base a esto se trabajó de manera directa en los dos ambientes de programación mostrando que se pueden optimizar procesos por medio de la programación y se incentiva a la comunidad educativa a encontrar estrategias que permitan el crecimiento y apropiación de estas.

[*] Trabajo de Grado

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería Mecánica. Director: Jabid Eduardo Quiroga Méndez. PhD Ingeniería Civil

Abstract

Title: Modeling of Mechanical-Type Dynamic Systems based on Euler Lagrange and Identification of Dissipative Parameters through Evolutionary Algorithms[*]

Author: Johan Sebastián Becerra Nova, Marco Tulio Mancipe Silva [**]

Key Words: Mathematical models, Evolutionary Algorithms, Dynamic Systems, programming.

Description: Over the years education has been adapted to different scenarios integrating different teaching strategies that allow a continuous improvement in the training of people. This project seeks to develop a program in MATLAB and Python for mathematical modeling, linearization, and identification of dissipative parameters of mechanical, linear, or non-linear dynamic systems. Providing support to the teaching professor, providing a software tool to the teacher and student of the subject dynamical systems, which makes use of symbolic calculus, Euler Lagrange methodology, Taylor series and Evolutionary Algorithms in MATLAB and Python. At present, recent technological developments allow data acquisition, signal processing and processing, Therefore, this work focuses attention on the modeling of linear or non-linear mechanical type systems and then identify their dissipative type parameters, to obtain a adjusted mathematical model that represents in a real way each mechanical system. Based on this, we worked directly in the two programming environments showing that we can optimize processes through programming and encourage the educational community to find strategies that allow the growth and appropriation of these.

[*] Degree Work

[**] Faculty of Physicomechanics. School of Mechanical Engineering. Director: Jabid Eduardo Quiroga Méndez. PhD Civil Engineer

Introducción

El modelado matemático de un sistema permite realizar una predicción del funcionamiento apoyándose en las características dinámicas del sistema, normalmente el modelo matemático de un sistema dinámico de tipo mecánico se trata de una serie de ecuaciones diferenciales que describen su comportamiento; muchos de los sistemas dinámicos de tipo mecánico son de interés en áreas como sistemas dinámicos, vibraciones, robótica e ingeniería de control entre otras, estos sistemas pueden ser lineales o lineales, se requieren tener en cuenta estrategias de linealización para los sistemas no lineales que permitan aplicar en gran medida muchos métodos de análisis lineal que a su vez proporcionen información acerca del comportamiento de sistemas no lineales; dado a que el modelado matemático de los sistemas permite la aproximación y predicción del funcionamiento de este tipo de sistemas, es necesario que el modelo se ajuste al comportamiento real de los sistemas, por ello se hace necesario la identificación de parámetros para desarrollar o mejorar la representación matemática de un sistema físico, usando datos experimentales.

En la actualidad los recientes desarrollos tecnológicos permiten la adquisición de datos, procesamiento y tratamiento de señales, por lo tanto este trabajo centra la atención en el modelamiento de sistemas de tipo mecánico lineales o no lineales para luego identificar sus parámetros de tipo disipativo, para obtener un modelo matemático ajustado que represente de manera real cada sistema de tipo mecánico brindando una herramienta software al profesor y estudiante de la asignatura sistemas dinámicos, que sirva de apoyo en el modelamiento matemático, linealización e identificación de parámetros disipativos en la dinámica de sistemas mecánicos, haciendo uso de cálculo simbólico, metodología de Euler Lagrange, series de Taylor

y Algoritmos Evolutivos en MATLAB y Python los cuales son dos de los entornos de programación más utilizados hoy en día por los científicos e ingenieros.

Al incluir una herramienta software de apoyo en modelamiento matemático usando la metodología de Euler Lagrange, el profesor y estudiante tendrán la posibilidad de comprobar u omitir algunos procesos matemáticos, que ya son de su conocimiento en cursos de niveles inferiores, estos procesos pueden ser extensos y con alta probabilidad de error. Al incluir una herramienta que identifique parámetros de sistemas dinámicos con Algoritmos Evolutivos, el estudiante podrá experimentar con sistemas reales, identificando la dinámica de estos.

El propósito de este proyecto es servir como herramienta de aprendizaje en el análisis dinámico de sistemas dinámicos de tipo mecánico, brindando la posibilidad de analizar ejercicios de alto grado de complejidad de una manera eficaz.

1. Objetivos

1.1. Objetivo General

Desarrollar un programa en MATLAB y Python para modelamiento matemático, linealización, e identificación de parámetros disipativos de sistemas dinámicos de tipo mecánico, lineales o no lineales. Contribuyendo a la misión de la escuela de Ingeniería Mecánica, en el desarrollo de investigación y formación de profesionales con alta calidad técnica y científica.

1.2. Objetivos Específicos

- Desarrollar un algoritmo en Python y MATLAB, que permita obtener modelos matemáticos (Ecuaciones diferenciales ordinarias lineales o no lineales), de sistemas dinámicos de tipo mecánico, a partir de las funciones de energía cinética y potencial de estos sistemas, utilizando la metodología de Euler-Lagrange.
- Desarrollar un algoritmo en Python y MATLAB, para linealizar ecuaciones diferenciales ordinarias, obtenidas del algoritmo del objetivo anterior, o proporcionadas por el usuario en forma simbólica, las cuales serán linealizadas aplicando las series de Taylor.
- Desarrollar un módulo de identificación de parámetros disipativos (constates de amortiguamiento, fricción), de sistemas dinámicos de tipo mecánico, partiendo de datos de un sistema dinámico (lineal o no lineal), y el modelo analítico que lo describe (ecuaciones diferenciales ordinarias), utilizando Algoritmos Evolutivos.
- Realizar material pedagógico en forma audiovisual, y presentaciones en PowerPoint para la asignatura sistemas dinámicos, que introduzca la herramienta de aprendizaje en el curso.

2. Modelos matemáticos

Para comprender el comportamiento de un sistema dinámico es importante realizar una descripción matemática de las características dinámicas del sistema; en este proyecto se muestra una metodología que resulta útil para describir modelos matemáticos de sistemas físicos con múltiples grados de libertad, haciendo uso de las ecuaciones de movimiento de Lagrange.

La mecánica de Lagrange permite modelar y simular las variables cinemáticas de máquinas, sistemas mecánicos o mecanismos complejos, su aplicación es general y se ha usado en máquinas agrícolas robótica entre otros. (Espinosa Bedoya & Gil parra, 2019)

2.1 Ecuaciones de movimiento de Lagrange

El modelado de sistemas puede realizarse mediante los principios básicos de la ley de conservación de energía para obtener las ecuaciones de movimiento, lo cual se hace fácil para sistemas simples, pero en sistemas con múltiples grados de libertad es conveniente usar el método de Lagrange donde se hace necesario definir las coordenadas generalizadas y el Lagrangiano para establecer el principio de Hamilton.

Mas adelante se mostrará el desarrollo de cada una de las ecuaciones y principios que serán necesarios para poder aplicar la siguiente ecuación:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} + \frac{\partial D}{\partial \dot{q}_i} = Q_{i(i=1,2,..n)} \tag{1}$$

Las n ecuaciones dadas por la ecuación (1) son las ecuaciones de Lagrange del movimiento para el sistema no conservativo y con fuerza de entrada Q_i; si se trabaja con sistemas conservativos se

suprime el uso de la función de disipación de Rayleigh y la fuerza de entrada (fuerza generalizada) Q_i de manera que la ecuación queda de la siguiente manera:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = 0_{(i=1,2,..n)} \tag{2}$$

De la cual se obtienen las n ecuaciones para el sistema conservativo.

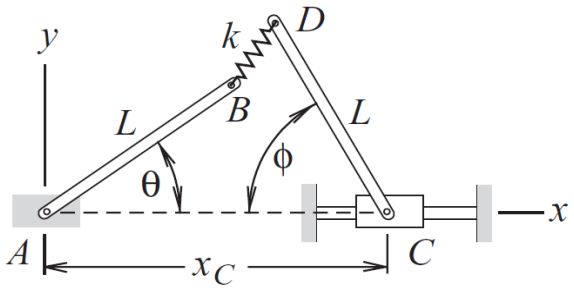
2.1.1 Coordenadas generalizadas

Son el conjunto de coordenadas independientes que se necesitan para describir completamente el movimiento de un sistema. El número de grados de libertad es igual al número de coordenadas generalizadas necesario para describir el movimiento del sistema.

Sí un sistema requiere n coordenadas generalizadas q_1, q_2, …, q_n necesitamos considerar n coordenadas generalizadas como coordenadas de un sistema n-dimensional en un espacio n-dimensional. Entonces en cualquier instante el sistema se caracteriza como un punto en este espacio n-dimencional. (Ogata, 1987, p. 596)

Figura 1

Coordenadas Generalizadas para un Mecanismo que está Conectado por un Resorte



Nota: en el mecanismo tiene tres grados de libertad (x_c θ ϕ) los cuales son el conjunto adecuado de coordenadas generalizadas. Tomado de (Ginsberg, 2008)

2.1.2 Lagrangiano

El Lagrangiano en términos generales se puede expresar de la siguiente manera:

Energía cinética del sistema – Energía potencial del sistema = Lagrangiano

El lagrangino (L) en forma general es una función de $q_i, \dot{q}_i$ (i=1,2,…,n) y del tiempo *t*.

2.1.3 Función de disipación de Rayleigh

En sistemas no conservativos o comúnmente llamados sistemas amortiguados la energía se disipa, por lo tanto, Rayleigh desarrollo una función de disipación D de la que puede derivarse la fuerza de amortiguamiento. Suponiendo que el sistema involucra *r* amortiguadores viscosos, la función de disipación se define mediante:

$$D = \frac{1}{2}(b_1\delta_1^2 + b_2\delta_2^2 + \cdots b_n\delta_n^2)$$

Donde b es la constante de amortiguamiento o coeficiente de amortiguamiento viscoso y δ es la diferencia de velocidad, así que δ puede expresarse como función de las velocidades generalizadas $\dot{q}_i$.(Ogata, 1987, p. 603)

2.2 Linealización de sistemas no lineales

El proceso de linealizar sistemas no lineales es importante debido a que mediante la linealización de ecuaciones no lineales es posible aplicar en gran medida muchos métodos de análisis lineal que a su vez proporcional información acerca del comportamiento de sistemas no lineales; la linealización de sistemas dinámicos usualmente se realiza eligiendo un punto de linealización el cual llamaremos punto de equilibrio, el cual debe ser estable para que la aproximación linealizada sea válida.

El proceso de linealización que se tomara en cuenta se basa en la expansión de la función no lineal en series de Taylor, tomando en cuenta solo el termino lineal de dicha expansión teniendo presente que los términos despreciados deben ser pequeños es decir que las variables se desvíen solo ligeramente de la condición de operación.

Si consideramos un sistema no lineal de dos entradas *x* y *y* cuya salida es *z* tenemos la relación $z=f(x,y)$ donde si la condición de operación normal corresponde al punto (x_0,y_0,z_0) se puede expandir en series de Taylor alrededor de ese punto de la siguiente manera:

$$z = f(x_0, y_0) + \left[\frac{\partial f}{\partial x}(x - x_0) + \frac{\partial f}{\partial y}(y - y_0)\right] + \frac{1}{2!}\left[\frac{\partial^2 f}{(\partial x^2)(x - x_0)^2} + 2\frac{\partial^2 f}{\partial x \partial y}(x - x_0)(y - y_0) + \frac{\partial^2 f}{(\partial y^2)(y - y_0)^2}\right] + \cdots$$

Evaluando las derivadas parciales en el punto de operación y despreciando los términos de más alto orden se llega al siguiente modelo lineal del sistema no lineal cercano al punto de operación:

$$z - z_0 = a(x - x_0) + b(y - y_0)$$

Donde:

$$a = \left.\frac{\partial f}{\partial x}\right|_{x=x_0, y=y_0, z=z_0}$$

$$b = \left.\frac{\partial f}{\partial y}\right|_{x=x_0, y=y_0, z=z_0}$$

Como se ha mencionado anteriormente es importante recordar que en el presente procedimiento de linealización las desviaciones de las variables de condición deben ser pequeñas de otra manera no se puede llevar a cabo el procedimiento.

3. Algoritmos Evolutivos

Los algoritmos Evolutivos son una técnica general, robusta capaz de proporcionar soluciones de alta calidad con amplia variación de componentes y parámetros. Su forma de procesamiento se inspira en la evolución de las especies y permiten abordar problemas con un alto grado de complejidad de optimación y búsqueda que surgen en la ingeniería y demás campos científicos. (Araujo & Cervigón, 2009)

De forma más formal y tomando la definición dada por Goldberg, “los Algoritmos Evolutivos son algoritmos de búsqueda basados en la mecánica de selección natural y de la genética natural. Combinan la supervivencia del más apto entre estructuras de secuencias con un intercambio de información estructurado, aunque aleatorizado, para constituir así un algoritmo de búsqueda que tenga algo de las genialidades de las búsquedas humanas” (Goldberg, 1989).

El principal objetivo de un Algoritmo Evolutivo es encontrar una soluciona aceptable a un problema por medio de mejoramiento de una población, cuya función de evaluación corresponde a una solución del problema. Al trabajar con algoritmos evolutivos se parte de un esquema general para la resolución de problemas donde solo se debe especificar la forma de ciertos componentes. En la Figura 2 se muestra el esquema general para un algoritmo Evolutivo en forma de pseudocódigo.

Figura 2

Pseudocódigo Esquema General para un Algoritmo Evolutivo

```
funcion Algoritmo_Genético()
{
     TPoblacion pob;    // población
     TParametros parámetros// tamaño población

     obtener_parametros(parametros);
     pob = poblacion_inicial();
     evaluacion(pob, tam_pob, pos_mejor, sumadaptacion);

     // bucle de evolución
     mientras no se alcanza condición de terminación hacer{
        seleccion(pob, parámetros);
        reproduccion(pob, parámetros);
        evaluacion(pob,parámetros,pos_mejor,sumadaptacion);
      }
     devolver pob[pos_mejor]
```

Al inicio del algoritmo se obtienen los datos de entrada del problema (parámetros) y se genera una población inicial, cuyos individuos se evalúan mediante la función de adaptación del algoritmo. El resto del algoritmo consiste en un bucle, que en cada iteración crea una generación en la que se produce un proceso de selección de los individuos mejor adaptados que tendrán mayor probabilidad de tener copias en una nueva población, seguido de un proceso de reproducción en el que se generan nuevos individuos a partir de los de la población mediante operaciones de mezcla y pequeñas alteraciones, y finalmente una evaluación de la nueva población. En muchas ocasiones se utilizan pequeñas variantes de este esquema.

Por lo tanto, para poder aplicar un Algoritmo Evolutivo se requieren los siguientes componentes básicos:

Representación de los individuos en los AG los individuos son cadenas binarias que se denotan por b que representan a los puntos x del espacio de búsqueda del problema. Si se toma la nomenclatura biológica b es denominada el genotipo del individuo y a x se le

denomina fenotipo. Existen tres tipos de codificación: binaria, no binaria y mixta; la sencillez de la representación binaria aporta características muy importantes de eficiencia, pero se debe tener en cuenta que es necesario disponer de un método para pasar de la representación binaria al espacio de búsqueda natural del problema. Cuando no es adecuada la representación binaria se debe recurrir a otro tipo de codificación, todo depende de la naturaleza del problema a solucionar.

Generación de la población inicial una forma de crear una población inicial de posibles soluciones está basada en un proceso aleatorio es decir que se va creando cada gen (posición de la cadena binaria) con una función que devuelve un cero o un uno con igual probabilidad. Es imprescindible para un buen funcionamiento del Algoritmo Evolutivo dotar a la población de sufímente variedad para poder explorar todas las zonas de búsqueda (Araujo & Cervigón, 2009, p. 29).

Función de evaluación (Fitness) juega el papel del ambiente clasificando las soluciones en términos de su aptitud, determinando que tan buenos o malos serán los resultados.

Condiciones de terminación en los Algoritmos Evolutivos es necesario especificar las condiciones donde el algoritmo deja de evolucionar y se presenta la mejor solución encontrada. Donde la condición de terminación más sencilla es indicando el número de generaciones de evolución.

Proceso de selección: Mecanismo de selección y muestreo en los Algoritmos Evolutivos se puede llevar el proceso de selección de múltiples maneras donde se tiende a favorecer la cantidad de copias de los individuos mejor adaptados las técnicas de selección más conocidas y aplicadas son las siguientes: proporcional, escolástica, selección por torneo y muestreo por retos.

Proceso de reproducción: operadores genéticos hace referencia a la forma en la que el Algoritmo Evolutivo va accediendo a nuevas regiones de búsqueda. Los nuevos individuos se crean aplicando algunos operadores genéticos como cruza, mutación y elitismo.

4. Metodología

4.1 Modelamiento matemático de sistemas dinámicos en Python y MATLAB

Como se mencionó en el capítulo 2, la forma más versátil para obtener el modelado matemático de sistemas dinámicos con múltiples grados de libertad es a partir de la ecuación de movimiento de Lagrange, conocida muchas veces en la literatura como la ecuación de Euler-Lagrange la cual se muestra a continuación

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} + \frac{\partial D}{\partial \dot{q}_i} = Q_{i(i=1,2,..n)}$$

Aprovechando las librerías de MATLAB y Python para calculo simbólico se decidió crear un módulo que integrara todos los términos de la ecuación de Euler-Lagrange, de manera que el usuario pueda obtener las ecuaciones que modelan el sistema dinámico a analizar, brindando el Lagrangiano del sistema en forma simbólica al programa. Se escogieron estos lenguajes de programación debido a sus características y ventajas.

Python es un lenguaje de programación abierto multiplataforma que permite ser usado en múltiples diciplinas convirtiéndose así en uno de los lenguajes de programación más utilizados en el mundo, por lo cual su comunidad es muy grande y proporciona soporte que facilita el desarrollo de aplicaciones; MATLAB es una plataforma de programación y calculo muy conocida por millones de ingenieros y científicos combina un entorno de escritorio perfeccionado para el análisis

iterativo y los procesos de diseño con un lenguaje de programación que expresa las matemáticas de matrices y arrays directamente, una desventaja frente a Python es que requiere una licencia para su uso.

Al desarrollar los módulos de modelado matemático y linealización en los dos lenguajes de programación permitimos al usuario escoger el entorno de programación donde tenga más experiencia puesto que se decidió que estos módulos sean utilizados de manera directa como código para fomentar en el estudiante el uso de la programación en la resolución de tareas de ingeniería.

Para el desarrollo del módulo de modelamiento matemático cada uno de los elementos de la ecuación de Euler-Lagrange estará contenido en una lista o vector según sea el caso, así que el vector X contendrá a cada una de las coordenadas generalizadas (q_i) y a sus primeras derivadas en el tiempo, el vector $\dot{X}$ contendrá a las derivadas en el tiempo del vector X y por último el vector o lista F contendrá la resta ($Q_i - F_{\text{amortiguamiento i}}$) que es equivalente a:

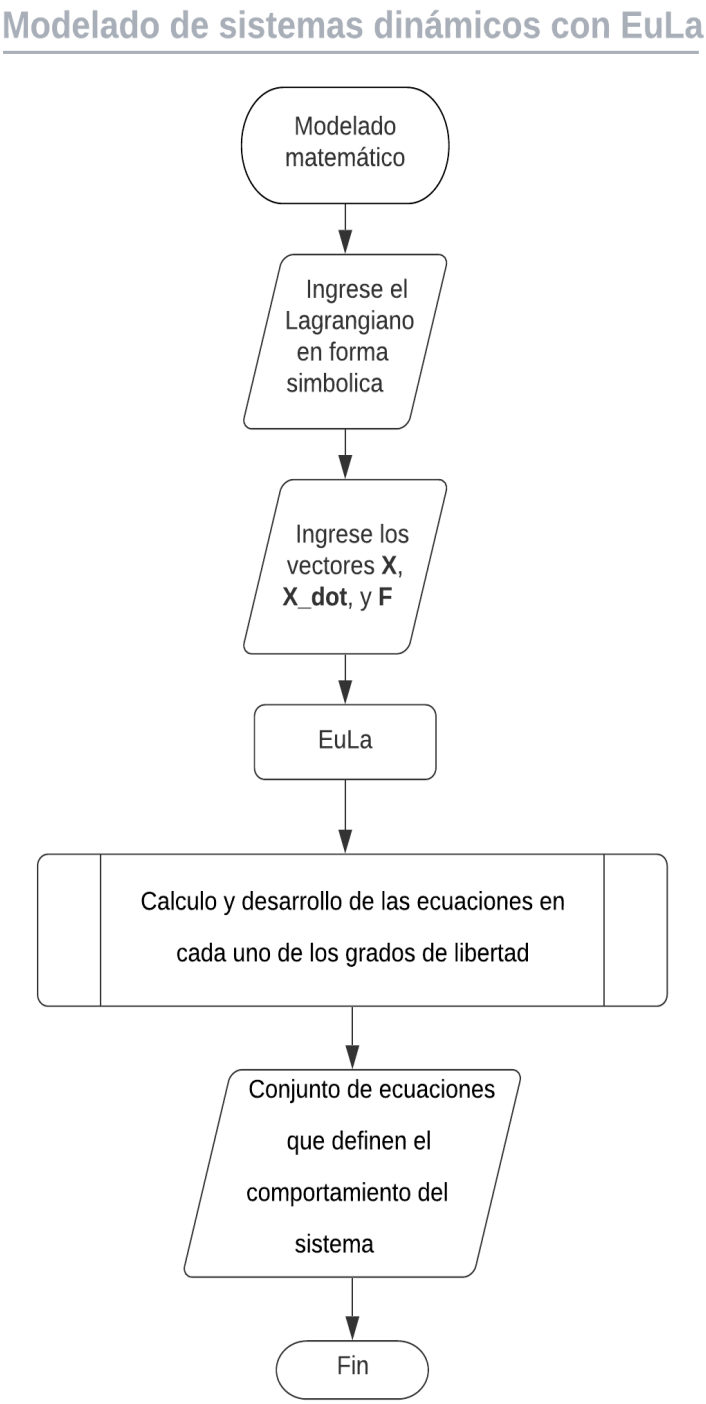
$$Q_{i(i=1,2,..n)} - \frac{\partial D}{\partial \dot{q}_i}$$

por lo tanto, el módulo que se desarrollo tiene la siguiente forma: $EuLa(L, X, \dot{X}, F)$.

En la Figura 3 se muestra el diagrama de flujo donde se expone la secuencia de calculo que se realiza haciendo uso de la función que se desarrolló para obtener las ecuaciones que modelan un sistema dinámico en forma simbólica.

Figura 3

Diagrama de Flujo Modelamiento con Función EuLa



El código de función EuLa en MATLAB y Python se muestra la Figura 4 y Figura 5, como se mencionó anteriormente esta función requiere que usuario ingrese el Lagrangiano en forma simbólica y los vectores o listas *X*, $\dot{X}$ y *F*, se debe tener en cuenta que al momento de ingresar los elementos de las listas o vectores $\dot{X}$ y *F* estos deben ir en el mismo orden del vector o lista *X*, guardando su correspondencia con el grado de libertad que está contenido en dicha lista o vector.

Figura 4

Módulo EuLa Desarrollado en MATLAB

```
function RTA = EuLa(L,X,X_dot,F)
    jac =jacobian(L,X);
    ddt_jac = jacobian(jac,X)*X_dot.';

RTA = [];
cont = 1;
for i = 1:length(X_dot)
    for j = 1:length(X)
       if X_dot(i) == X(j)
          X_ind = j;
          X_dind = i;
          RTA = [RTA,simplify(eq(ddt_jac(X_ind) - jac(X_dind),F(cont)))];
          cont = cont+1;
          break
       end
    end
end
end
```

Nota: La función Eula es una adaptación de las rutinas de cálculo de las ecuaciones de movimiento de Lagrange donde por cada grado de libertad se tendrá una ecuación diferencial, por lo tanto, se requiere derivar el Lagrangiano en términos de cada coordenada generalizada $\frac{\partial L}{\partial q_i}$ para ello se recurrió a sacar el Jacobiano, luego de esto para obtener cada termino de $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right)$ se hace necesario calcular el Jacobiano del Jacobino (ddt_jac) y multiplicar por la variable temporal de cada grado

de libertad contenida en el vector X_dot ($\dot{X}$) el cual el usuario debe ingresar previamente al igual que F y X, luego para mostrar las ecuaciones se ocuparon los ciclos for.

Figura 5

Módulo EuLa Desarrollado en Python

```python
import sympy as sym
def Eu_La(L,X,X_dot,F):
    L_M = sym.Matrix([L])
    X_M = sym.Matrix([X])
    X_dot_M = sym.Matrix([X_dot])
    jac = L_M.jacobian(X_M)
    ddt_jac = sym.Matrix.dot(jac.jacobian(X_M),X_dot_M.T)

    #Ecuación Euler lagrange:
    RTA = []
    cont = 0
    for i in X_dot:
        if X.count(i)>0:
            Xind = X.index(i)
            X_dind = X_dot.index(i)
            RTA.append(sym.Eq(ddt_jac[Xind] - jac[X_dind],F[cont]).simplify())
            cont += 1
    return RTA
```

Nota. Si se utiliza Python en un entorno como Google Colab solo se requiere importar las librerías que serán necesarias durante la ejecución de los programas, por otro lado, si se decide utilizar el intérprete de Python se debe tener en cuenta que existen librerías estándar que se instalan a la vez con el intérprete las cuales solo requieren ser importadas para usarse, las librerías externas las cuales son desarrolladas por la comunidad pueden ser instaladas. A continuación, se presenta una breve descripción de una de las librerías que se utilizó para el desarrollo del módulo EuLa: SymPy es una biblioteca de Python para matemáticas simbólicas. Su propósito es llegar a ser un sistema de álgebra por computadora (CAS) completo manteniendo el código tan simple como sea posible para poder ser legible y extensible de manera fácil.

En resumen, se realizó una adaptación de las rutinas de cálculo para el modelado de sistemas dinámicos mediante la metodología de las ecuaciones de movimiento de Lagrange desarrollando el módulo EuLa.

Luego de obtener las ecuaciones para cada grado de libertad para el sistema dinámico con múltiples grados de libertad en forma simbólica, se puede obtener la solución de forma numérica aplicando cualquier método de solución de sistemas de ecuaciones diferenciales, para su implementación; en MATLAB se aprovecha el módulo ODE45 el cual es un versátil solver de ODE (MATLAB, n.d.), y en Python se utilizó el módulo ODEINT de Scipy.integrate en el Apéndice A se muestran algunos ejemplos de uso del módulo EuLa junto a los módulos de solución de ODE45.

4.2 Linealización de sistemas no lineales en Python y MATLAB

La linealización de las ecuaciones diferenciales proporcionadas por el usuario o las arrojadas por el módulo EuLa se realiza aplicando la teoría de las series de Taylor mencionada en la sección 2.2, para esta tarea se realizó un módulo en Python y MATLAB donde el usuario debe proporcionar los siguientes parámetros sis, *X*, *X0* puesto que la función para linealización que llamaremos *lin_sis* tendrá la siguiente forman *lin_sis(sis,X,X0)* donde sis será el conjunto de ecuaciones a linealizar, *X* contendrá las variables o entradas del sistema de ecuaciones, y *X0* contendrá los puntos donde se va expandir las series de Taylor (puntos de linealización).

A continuación, se muestran los códigos de los módulos que se desarrollaron en los dos lenguajes de programación.

Figura 6

Módulo de linealización en MATLAB

```
function st_sis =  lin_sis(sis,X,X0)
names = fieldnames(sis)
m = length(names)
for i = 1:m
    st_sis.(names{i}) = lin(sis.(names{i}),X,X0)
end
end
```

Nota: El ciclo for calcula la derivada de cada una de las ecuaciones que modela el sistema dinámico en cada variable de estado y la evalúa en el punto de linealización X0.

Figura 7

Módulo Linealización Desarrollado en Python

```
import a sympy as sym
def lin(f,X,X0):
    m=len(X)
    pares=[]
    for i in range(m):
        pares.append((X[i],X0[i]))
    y=f.subs(pares)
    for i in range m:
        y=y+sym.diff(f,X[i].subs(pares)*(X[i]-X0[i]))
    return y

def lin_sis(sis,X,X0):
    sis_lin={}
    for i in sis.keys():
        sis_lin.__setitem__(i,lin(sis[i],X,X0))
    return sis_lin
```

Nota: la función lin_sis permite linealizar el conjunto de ecuaciones diferenciales dadas o las arrojadas por el módulo EuLa en el punto de linealización X0.

4.3 Identificación de parámetros, que componen ODE de sistemas dinámicos, haciendo uso de Algoritmos Evolutivos

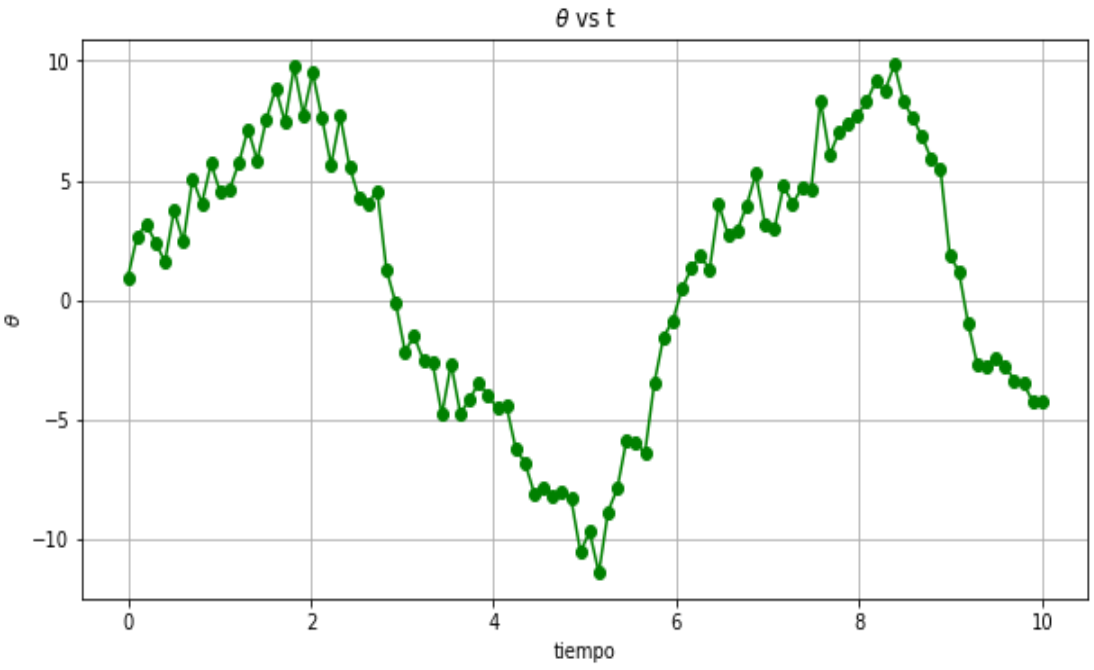
En esta etapa se desarrolló un módulo de identificación de parámetros disipativos (constates de amortiguamiento, fricción), de sistemas dinámicos de tipo mecánico, haciendo uso de algoritmos evolutivos, partiendo de datos de un sistema dinámico (lineal o no lineal), y el modelo analítico que lo describe (ecuaciones diferenciales ordinarias).

El problema de identificación de parámetros disipativos se puede explicar con este sencillo ejemplo:

Supongamos que tenemos un sistema dinámico de tipo mecánico, para el cual se tomaron datos experimentales de una de sus variables, en la siguiente grafica se muestran los datos experimentales:

Figura 8

Datos Experimentales Sistema Dinámico



Utilizando el método de Euler Lagrange, se modeló el sistema dinámico, y se encontró la familia de funciones que lo describen, las cuales se representan de la siguiente manera.

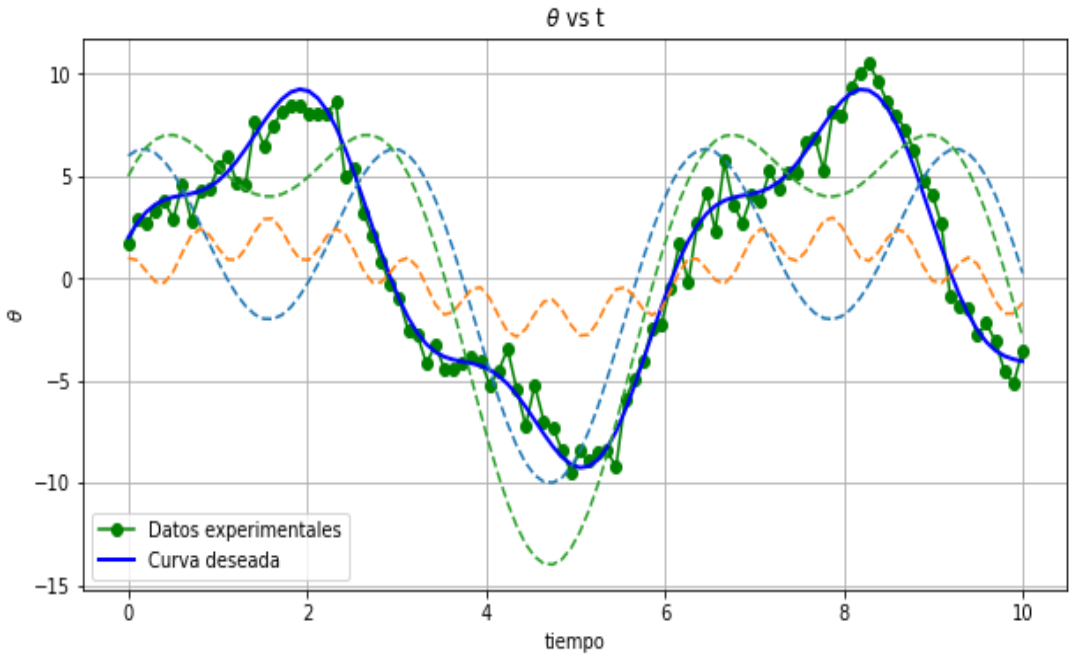
$$\theta = f(a, b, c, t)$$

Donde los parámetros a, b y c son los parámetros disipativos desconocidos, y t corresponde al tiempo.

La función $f(a, b, c, t)$ puede generar diferentes curvas, variando los parámetros a, b y c Sin embargo, la curva que mejor se adapta a los datos experimentales corresponde a los parámetros disipativos a, b y c del sistema dinámico de estudio.

Figura 9

Curvas de Adaptación de f(a,b,c,t)



Para hallar los parámetros disipativos a, b y c es necesario suponer los mismos, y evaluar el parecido de la curva generada por $f(a, b, c, t)$ y la curva de datos experimentales. para llevar a cabo este proceso, se propuso tratar la familia de curvas $f(a, b, c, t)$ como una especie, que evolucionara, ajustando las curvas a los datos experimentales.

Para ello fue necesario escribir un módulo de Algoritmos Evolutivos (AGS_R) que permitiera representar el genotipo de los individuos como un vector de números reales, los cuales

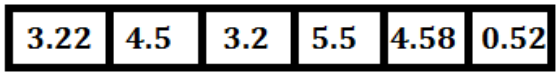
representan los valores que se quieren optimizar, y posteriormente utilizarlo para buscar los parámetros disipativos del sistema dinámico de estudio.

4.3.1 Módulo AGS_R

El módulo AGS_R es un algoritmo escrito en Python, inspirado en la teoría de evolución de Charles Darwin, que permite hallar una solución óptima a varios problemas de optimización, y búsqueda, tratando el dominio de solución del problema como una especie, la cual evolucionará hasta adaptarse a los requerimientos del problema. este módulo se diseñó para optimizar problemas cuyas soluciones puedan ser representadas por vectores de números reales, donde cada número representa una variable del problema, a continuación, se muestra un cromosoma o individuo.

Figura 10

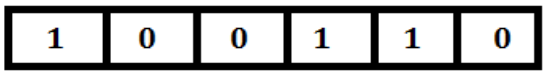
Cromosoma en Representación de Números Reales



La diferencia del módulo AGS_R a un algoritmo genético se basa en la representación del cromosoma, ya que un algoritmo genético representa el cromosoma como una cadena binaría, a continuación, se muestra la representación del cromosoma en un algoritmo genético

Figura 11

Cromosoma en Representación Binaria



En la identificación de parámetros disipativos se optó por la representación de cromosoma como una cadena de números reales donde cada posición representa una constante de disipación.

Función de cruce

En la construcción del módulo AGS_R se utilizó el cruce binario simulado (SBX) (Deb K, 1995, pp. 115–148), ya que es uno de los operadores más utilizados para codificaciones con números reales, este se aplica generando un numero α entre 0 y 1, este valor determina el modo de calcular otro valor β

- Si $\alpha < 0.5$ entonces $\beta = 2\alpha^{\left(\frac{1}{n+1}\right)}$
- Si $\alpha >= 0.5$ entonces $\beta = \left(\frac{1}{2(1-\alpha)}\right)^{\left(\frac{1}{n+1}\right)}$

El valor recomendado para n es 1 o 2, el valor para cada hijo se puede obtener mediante la siguiente formula, para cada uno de los hijos

$$h1_i = 0.5((P1_i + P2_i) - \beta|P2_i - P1_i|)$$

$$h2_i = 0.5((P1_i + P2_i) + \beta|P2_i - P1_i|)$$

Donde i hace referencia a un gen del cromosoma.

Función Mutación:

La función mutación evita que la población se estanque en un máximo o mínimo local, permitiendo la exploración del espacio de solución, aun cuando toda la población converge a un punto especifico, en el módulo de AGS_R se optó por la mutación uniforme, la cual se realiza modificando un elemento del cromosoma, por un valor al azar, dentro del rango permitido.

Función selección:

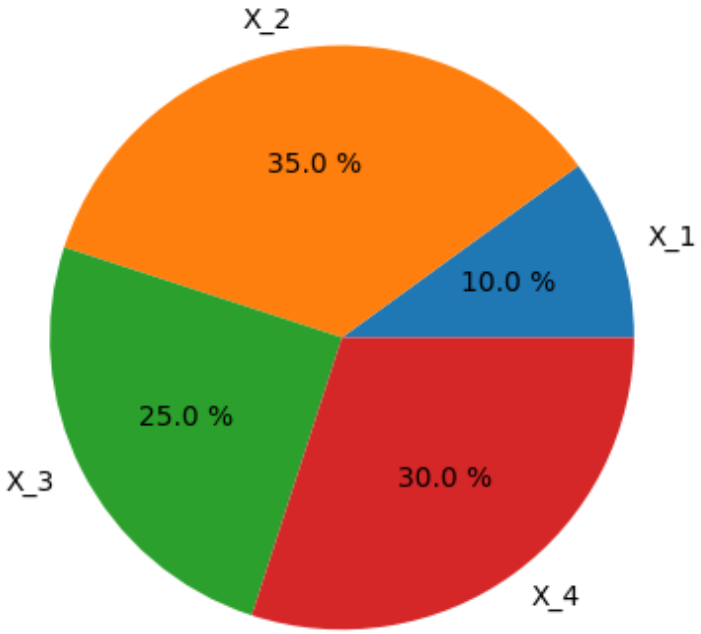
En la construcción del módulo AGS_R se define la selección por el método de la ruleta, la probabilidad de seleccionar un individuo con este método es proporcional a la adaptación relativa del mismo, se evalúa la adaptación de cada individuo en la población y se calcula su adaptación relativa a la población con la siguiente ecuación:

$$P_i = \frac{Adt(i)}{Adt_{prom}}$$

Donde $Adt(i)$ es la adaptación del individuo i, y Adt_{prom} es la adaptación promedio en la población. El área de la ruleta se divide proporcional a la adaptación relativa de los individuos, así al girar la ruleta los individuos más adaptados de la población tienen mayor probabilidad de seleccionarse para reproducción.

Figura 12

Método de la Ruleta



El procedimiento utilizado para simular la ruleta fue el siguiente:

- Se define la puntuación acumulada como $\text{Sum\_Adt}_i = \sum_{k=l}^{i} \text{Adt(k)}$ donde k hace referencia a los individuos anteriores al individuo i
- Se genera un numero “a” entre 0 y 1
- Se selecciona un individuo que cumpla $Sum\_Adt_{(i-1)} < a < Sum\_Adt_i$

Función Evolución

La función de evolución se encarga de ejecutar las demás funciones descritas anteriormente, en esta función se genera la población inicial y se evoluciona un numero de generaciones definidas por el usuario, se compone principalmente de un bucle for que repite el proceso de selección, cruce y mutación, por la cantidad de generaciones. También selecciona el mejor individuo de cada generación y se compara con el mejor de las generaciones anteriores, si este es más adaptado se remplaza, obteniendo el puesto de ELITE. La función Evolución, requiere como parámetro la función de adaptación, y retorna la Elite (Individuo más adaptado), y una lista que evidencia la evolución de la población, la cual contiene los mejores individuos de menor a mayor adaptación, presentados durante la ejecución.

El código del Módulo AGS_R se puede visualizar en el Apéndice B.

5. Resultados

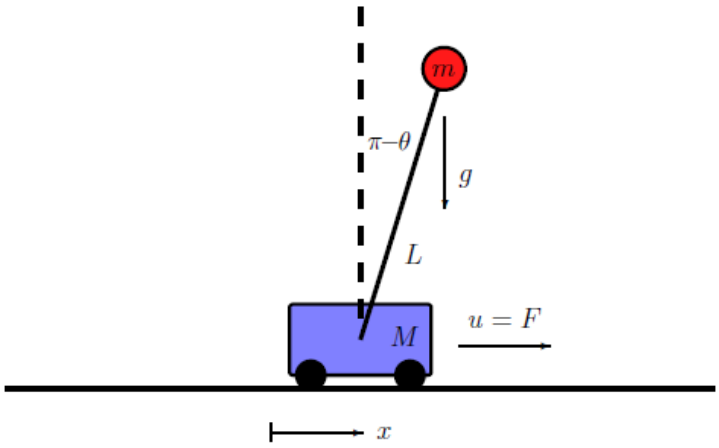
En este capítulo se comprueba el desarrollo de los diferentes módulos realizando pruebas y validación de los resultados. Para iniciar se realiza la obtención de los modelos matemáticos de algunos sistemas dinámicos de tipo mecánicos muy conocidos partiendo del análisis cinemático para encontrar la energía potencial y cinética de cada sistema para luego obtener el Lagrangiano y obtener las ecuaciones diferenciales que definen cada sistema mediante el módulo EuLa.

5.1 Modelamiento matemático de sistemas dinámicos en Python y MATLAB

El primer sistema es un mecanismo de péndulo invertido sobre un carro el cual es un problema común dentro del estudio de los sistemas dinámicos siendo un sistema de dos grados de libertad, donde las coordenadas generalizadas que lo describen son θ y x.

Figura 13

Péndulo Invertido



Nota. Tomada de Data Driven Science & Engineering Machine Learning, Dynamical Systems, and Control (p.352), por Brunton & Kutz, 2019, Cambridge University Press.

De acuerdo con la Figura 13 se define los siguientes parámetros:

Tabla 1

Parámetros péndulo Invertido

Parámetros	Descripción
m	Masa punctual del péndulo
M	Masa del carro
l	Longitude del péndulo
θ	Coordenada angular del péndulo
x	Coordenada horizontal del carro
u	Fuerza de entrada al sistema

Del análisis cinemático y geométrico se obtiene las siguientes expresiones que permitirán obtener la energía potencial y cinética del péndulo invertido para luego obtener el Lagrangiano que se define mediante la expresión $L=E_k-E_p$:

$\dot{x} = v$

$\dot{\theta} = w$

$y = lcos(\theta) \rightarrow$ altura del péndulo

$v_{px} = v + lwcos(\theta) \rightarrow$ velocidad del péndulo en la dirección horizontal

$v_{py} = -lwsin(\theta) \rightarrow$ velocidad del péndulo en la dirección vertical

De manera que la energía cinética y potencial del sistema queda de la siguiente manera:

$$E_k = \frac{M\,v^2}{2} + \frac{m\,((v + l\,\omega\,\cos(\theta))^2 + l^2\,\omega^2\,\sin(\theta)^2)}{2}$$

$$E_p = g\,l\,m\,\cos(\theta)$$

Obteniendo así, el Lagrangiano para el sistema de péndulo invertido

$$L = \frac{M\,v^2}{2} + \frac{m\,((v + l\,\omega\,\cos(\theta))^2 + l^2\,\omega^2\,\sin(\theta)^2)}{2} - g\,l\,m\,\cos(\theta)$$

El modelo matemático del sistema péndulo invertido se obtiene de la siguiente manera brindado el Lagrangiano (L), el vector de estados X, el vector de derivadas temporales de X (X_dot) y el vector de fuerzas F haciendo uso del módulo EuLa en MATLAB:

```
%se deben declarar los simbolos que definen el sistema
syms x v a theta omega alpha u M delta m l g

% Lagrangiano
L=(M*v^2)/2+(m*((v + l*omega*cos(theta))^2 + l^2*omega^2*sin(theta)^2))/2-
g*l*m*cos(theta);

X=[x,theta,v,omega]; %cordenadas generalizadas qi y qi_dot
X_dot=[v,omega,a,alpha]; %vector de derivadas en el tiempo de X
```

```
 F=[u-delta*v 0]; %vector de fuerzas que actuan en cada uno de los grados de
libertad

 sol=EuLa(L,X,X_dot,F); %funcion que resuelve las ecuaciones de Euler-Lagrange
 grad=solve(sol,a,alpha); %muestra cada Ecuacion diferencial
 grad.a
```

$$a = \frac{l\,m\,\sin(\theta)\,\omega^2 + u - \delta\,v - g\,m\,\cos(\theta)\,\sin(\theta)}{-m\,\cos(\theta)^2 + M + m} \tag{3}$$

```
grad.alpha
```

$$\frac{-l\,m\,\cos(\theta)\,\sin(\theta)\,\omega^2 - u\,\cos(\theta) + \delta\,v\,\cos(\theta) + g\,m\,\sin(\theta) + M\,g\,\sin(\theta)}{l\,(-m\,\cos(\theta)^2 + M + m)} \tag{4}$$

Como se comentó en el inicio de este capítulo el péndulo invertido es un problema muy conocido en la literatura por ello si comparamos las ecuaciones (3) y (4) proporcionadas por el módulo EuLa con las presentadas en la Figura 14 se evidencia que el módulo EuLa proporciona un modelo totalmente equivalente

Figura 14

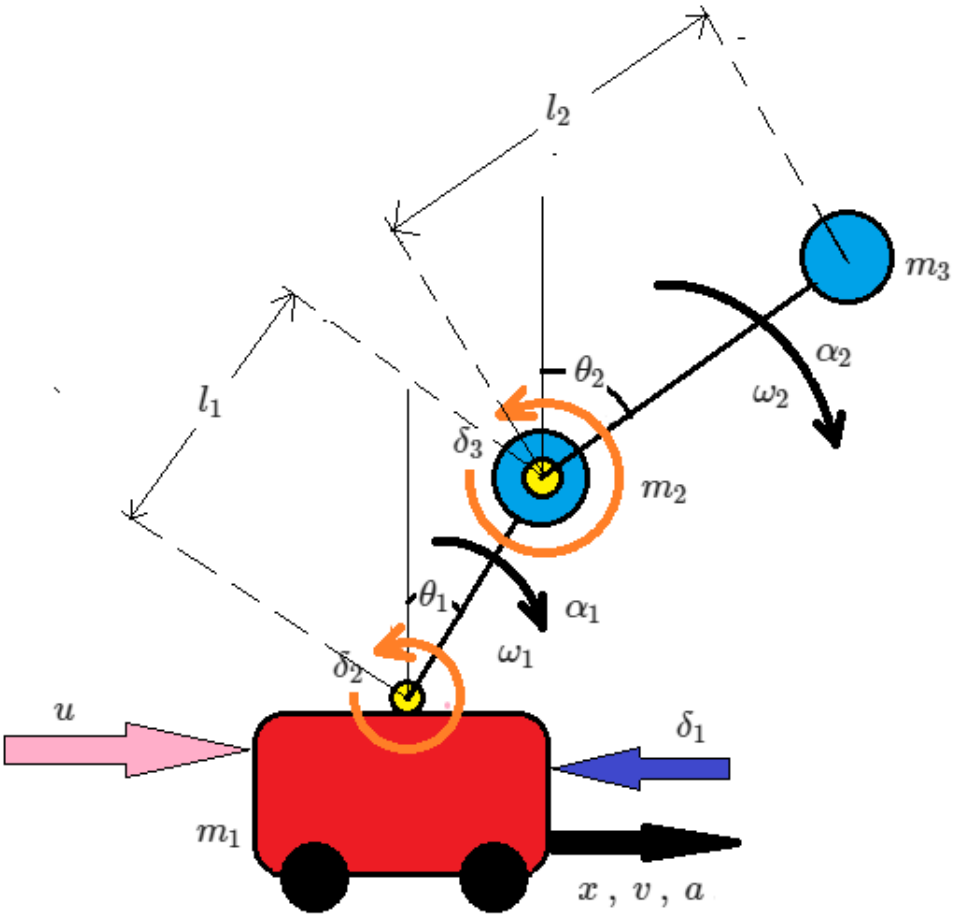
Ecuaciones péndulo invertido

$$\begin{aligned}
\dot{x} &= v \\
\dot{v} &= \frac{-m^2L^2g\cos(\theta)\sin(\theta) + mL^2(mL\omega^2\sin(\theta) - \delta v) + mL^2u}{mL^2(M + m(1 - \cos(\theta)^2))} \\
\dot{\theta} &= \omega \\
\dot{\omega} &= \frac{(m+M)mgL\sin(\theta) - mL\cos(\theta)(mL\omega^2\sin(\theta) - \delta v) + mL\cos(\theta)u}{mL^2(M + m(1 - \cos(\theta)^2))}
\end{aligned}$$

Nota. Tomada de Data Driven Science & Engineering Machine Learning, Dynamical Systems, and Control (p.352), por Brunton & Kutz, 2019, Cambridge University Press.

Figura 15

Carro con Péndulo Doble Invertido



En la figura 15 se presenta un carro con dos péndulos acoplados al mismo, este problema se deriva del péndulo invertido agregándole un grado de libertad, al acoplarle otro péndulo, a continuación, se procede a desarrollar el modelamiento del sistema.

```
# Velocidades
```

$$Vx_2 = v + \omega_1 l_1 cos(\theta_1)$$
$$Vy_2 = -\omega_1 * l_1 * sin(\theta_1)$$
$$Vx_3 = Vx_2 + \omega_2 * l_2 * cos(\theta_2)$$
$$Vy_3 = Vy_2 - \omega_2 * l_2 * sin(\theta_2)$$

```
# Alturas
```

$$y_1 = l_1 * cos(\theta_1)$$
$$y_2 = y_1 + l_2 * cos(\theta_2)$$

```
#Energías cinéticas
```

$$Ek1 = \frac{1}{2} * m1 * v^2$$
$$Ek2 = \frac{1}{2} * m2 * (Vx_2^2 + Vy_2^2)$$
$$Ek3 = \frac{1}{2} * m3 * (Vx_3^2 + Vy_3^2)$$

```
#Energías potenciales
```

$$Ep1 = m2 * g * y_1$$
$$Ep2 = m3 * g * y_2$$

```
# Lagrangiano
```

$$L = Ek1 + Ek2 + Ek3 - (Ep1 + Ep2)$$

Se define el sistema dinámico con los siguientes parámetros

$$m_1 = 1, \quad m_2 = 0.5, \quad m_3 = 0.3$$

$$L_1 = 0.2, \quad L_2 = 0.2, \quad g = 9.81$$

El Lagrangiano del sistema está dado por la siguiente ecuación:

$$\begin{aligned} L = {} & 0.01\omega_1^2 \sin^2(\theta_1) + 0.5v^2 + 0.006(-\omega_1 \sin(\theta_1) - \omega_2 \sin(\theta_2))^2 \\ & + 0.25(0.2\omega_1 \cos(\theta_1) + v)^2 + 0.15(0.2\omega_1 \cos(\theta_1) + 0.2\omega_2 \cos(\theta_2) + v)^2 \\ & - 1.5696 \cos(\theta_1) - 0.5886 \cos(\theta_2) \end{aligned}$$

Las ecuaciones diferenciales que modelan este sistema se hallaron usando el módulo de Euler Lagrange, escrito en Python, a continuación, se muestran los resultados.

$$\delta_1 v - u = -1.8a - 0.16\alpha_1 \cos(\theta_1) - 0.06\alpha_2 \cos(\theta_2) + 0.16\omega_1^2 \sin(\theta_1) + 0.06\omega_2^2 \sin(\theta_2)$$

$$\begin{aligned} \delta_2 \omega_1 = {} & -0.16a \cos(\theta_1) - 0.032\alpha_1 - 0.012\alpha_2 \cos(\theta_1 - \theta_2) - 0.012\omega_2^2 \sin(\theta_1 - \theta_2) \\ & + 1.5696 \sin(\theta_1) \end{aligned}$$

$$\begin{aligned} \delta_3 \omega_2 = {} & -0.06a \cos(\theta_2) - 0.012\alpha_1 \cos(\theta_1 - \theta_2) - 0.012\alpha_2 + 0.012\omega_1^2 \sin(\theta_1 - \theta_2) \\ & + 0.5886 \sin(\theta_2) \end{aligned}$$

Se procede a simular el sistema, para ello se definen las constantes de amortiguamiento como $\delta_1 = 0.1$, $\delta_2 = 0.2$, $\delta_3 = 0.3$, y se define la posición inicial en $x = 0$, $\theta_1 = 0$ y $\theta_2 = \frac{\pi}{8}$, de este modo los dos péndulos quedan arriba, con el propósito de que la energía potencial de estos sea la que produzca el movimiento.

Figura 16

Desplazamiento en Cada uno de los Grados de Libertad Carro con péndulo Doble Invertido

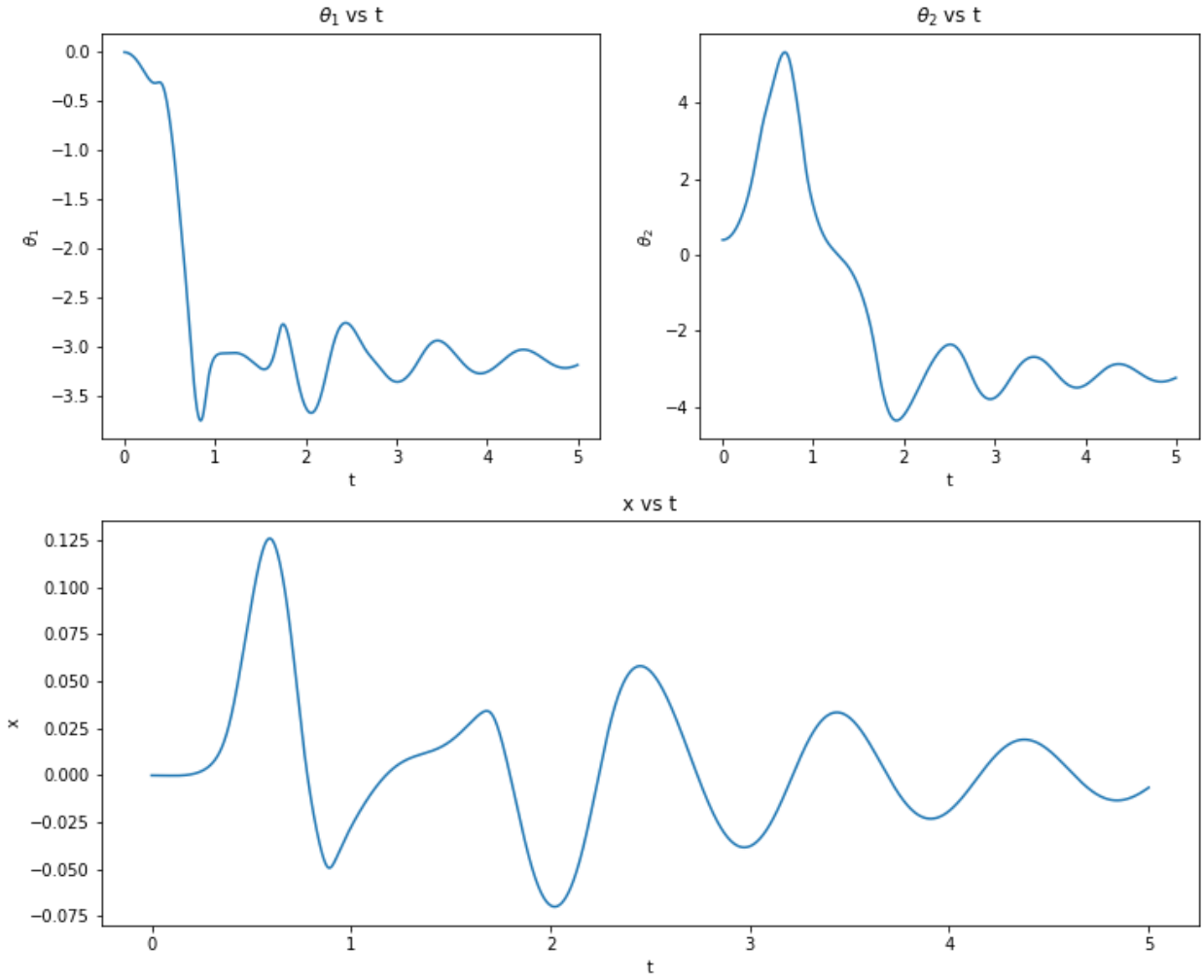


Figura 17

Secuencia de Movimiento Carro con Péndulo Doble Invertido

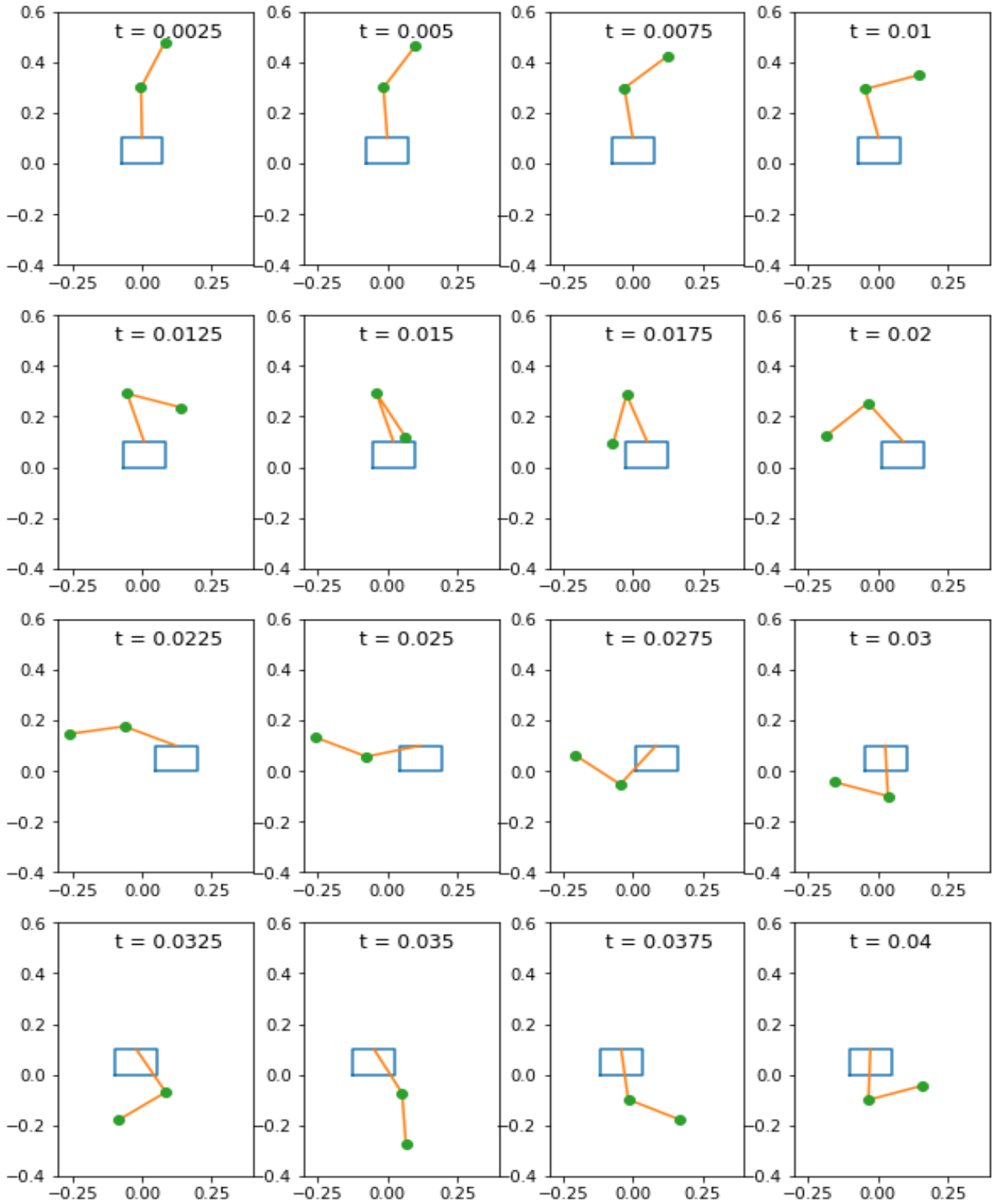
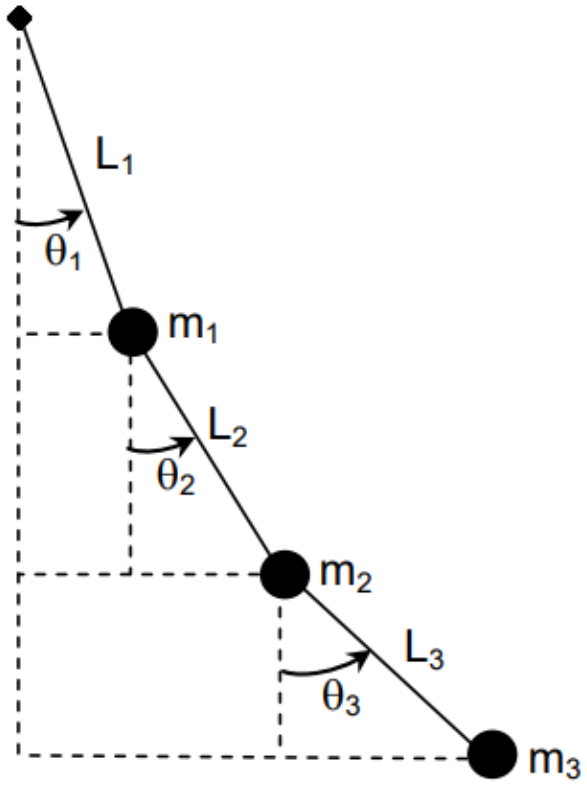


Figura 18

Péndulo Triple



En la Figura 18 se presenta un sistema dinámico compuesto por tres péndulos acoplados entre ellos, a este sistema se le conoce como péndulo triple y es un sistema representativo del caos, a continuación, se presenta el proceso de modelado y los resultados

```
#Velocidades
```

$$V1_x = \omega_1 * l_1 * cos(\theta_1)$$
$$V1_y = \omega_1 * l_1 * sin(\theta_1)$$
$$V2_x = V1_x + \omega_2 * l_2 * cos(\theta_2)$$
$$V2_y = V1_y + \omega_2 * l_2 * sin(\theta_2)$$
$$V3_x = V2_x + \omega_3 * l_3 * cos(\theta_3)$$
$$V3_y = V2_y + \omega_3 * l_3 * sin(\theta_3)$$

#Alturas

$$Y_1 = -l_1 * cos(\theta_1)$$
$$Y_2 = Y_1 - l_2 * sym.cos(\theta_2)$$
$$Y_3 = Y_2 - l_3 * cos(\theta_3)$$

#Energía cinética

$$EK_1 = \frac{1}{2} * m_1 * \left(V1_x^2 + V1_y^2\right)$$

$$EK_2 = \frac{1}{2} * m_2 * \left(V2_x^2 + V2_y^2\right)$$

$$EK_3 = \frac{1}{2} * m_3 * \left(V3_x^2 + V3_y^2\right)$$

#Energía potencial

$$Ep_1 = m_1 * g * Y_1$$
$$Ep_2 = -m_2 * g * Y_2$$
$$Ep3 = -m_3 * g * Y_3$$

#Lagrangiano#

$$L = EK_1 + EK_2 + EK_3 - Ep_1 - Ep_2 - Ep_3$$

Usando el módulo sympy de Python el cual es una herramienta de cálculo simbólico, se introducen las ecuaciones anteriores, y se simplifican. El Lagrangiano del sistema es:

$$\begin{aligned} L = {} & gl_1m_1\cos(\theta_1) + gm_2(-l_1\cos(\theta_1) - l_2\cos(\theta_2)) \\ & + gm_3(-l_1\cos(\theta_1) - l_2\cos(\theta_2) - l_3\cos(\theta_3)) \\ & + 0.5m_1(l_1^2\omega_1^2\sin^2(\theta_1) + l_1^2\omega_1^2\cos^2(\theta_1)) \\ & + 0.5m_2((l_1\omega_1\sin(\theta_1) + l_2\omega_2\sin(\theta_2))^2 + (l_1\omega_1\cos(\theta_1) + l_2\omega_2\cos(\theta_2))^2) \\ & + 0.5m_3((l_1\omega_1\sin(\theta_1) + l_2\omega_2\sin(\theta_2) + l_3\omega_3\sin(\theta_3))^2 \\ & + (l_1\omega_1\cos(\theta_1) + l_2\omega_2\cos(\theta_2) + l_3\omega_3\cos(\theta_3))^2) \end{aligned}$$

A continuación, se procede a realizar el procedimiento de Euler Lagrange al Lagrangiano del péndulo triple, de esta manera se obtienen las ecuaciones diferenciales que modelan el sistema dinámico, para ello se utiliza el módulo (EU_LA) al cual se le introduce el Lagrangiano del sistema, el vector de estado, el gradiente del vector de estado y el vector de fuerzas, como se muestra.

$X = [\theta_1, \theta_2, \omega_1, \omega_2]$ #Vector de estado
$\dot{X} = [\omega_1, \omega_2, \alpha_1, \alpha_2]$ #Gradiante del vector de estado
$F = [0,0,0]$ #vector de Fuerzas

$Sol = EL.EU_LA(L, X, \dot{X}, F)$ `#Solución por EU_LA`

Se definen las masas y las longitudes de los elementos como se muestra:

$$m_1 = 1\ , \qquad m_2 = 2, \qquad m_3 = 3$$

$$l1 = 0.2, \quad l_2 = 0.2, \qquad l_3 = 0.2$$

Inicialmente se colocan los péndulos hacia arriba, y se sueltan, de modo que la energía potencial de los mismos es la que genera el movimiento, no se considera fricción entre elementos, ni otro tipo de fuerzas, por lo tanto, el vector de fuerzas es F = [0,0,0], a continuación, se presentan los resultados.

Figura 19

Desplazamiento en Cada uno de los Grados de Libertad Péndulo Triple

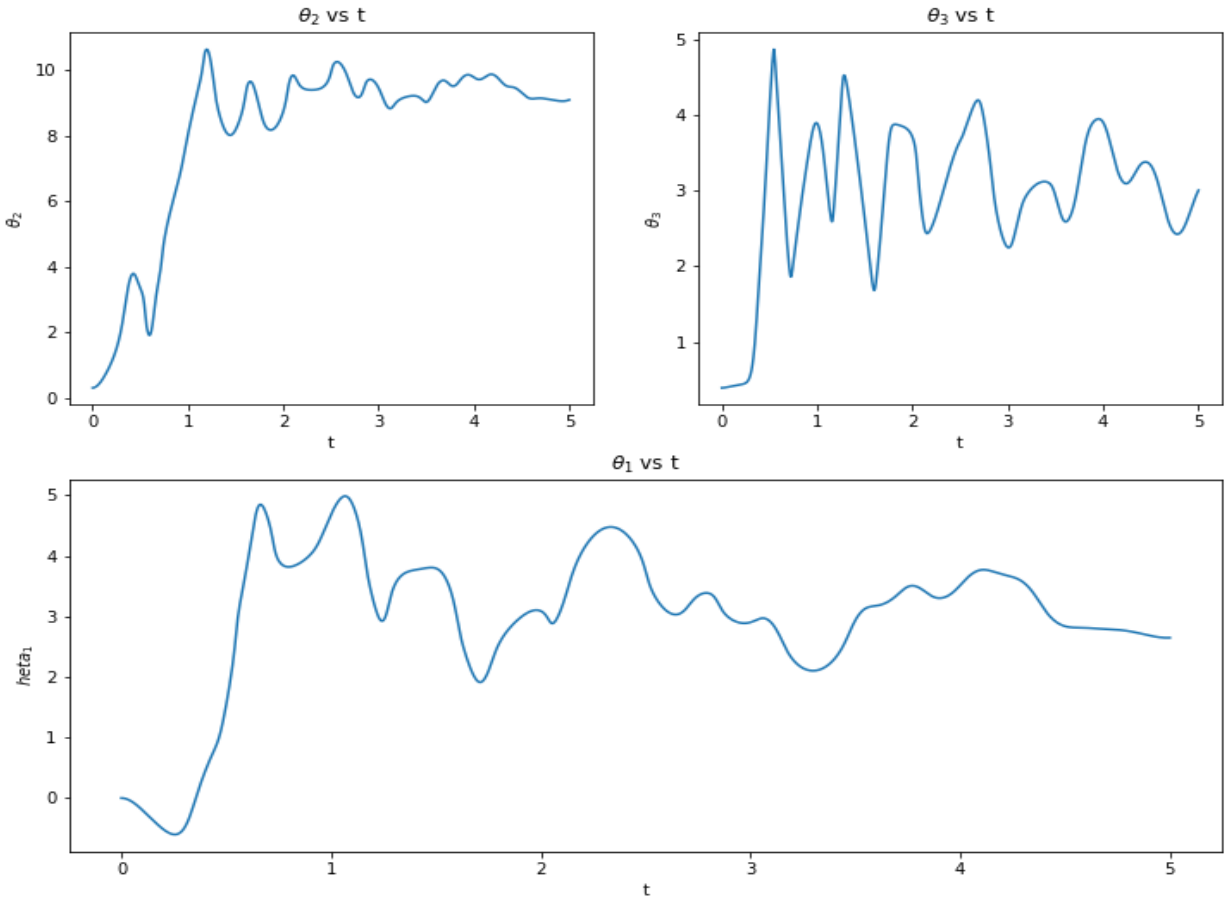
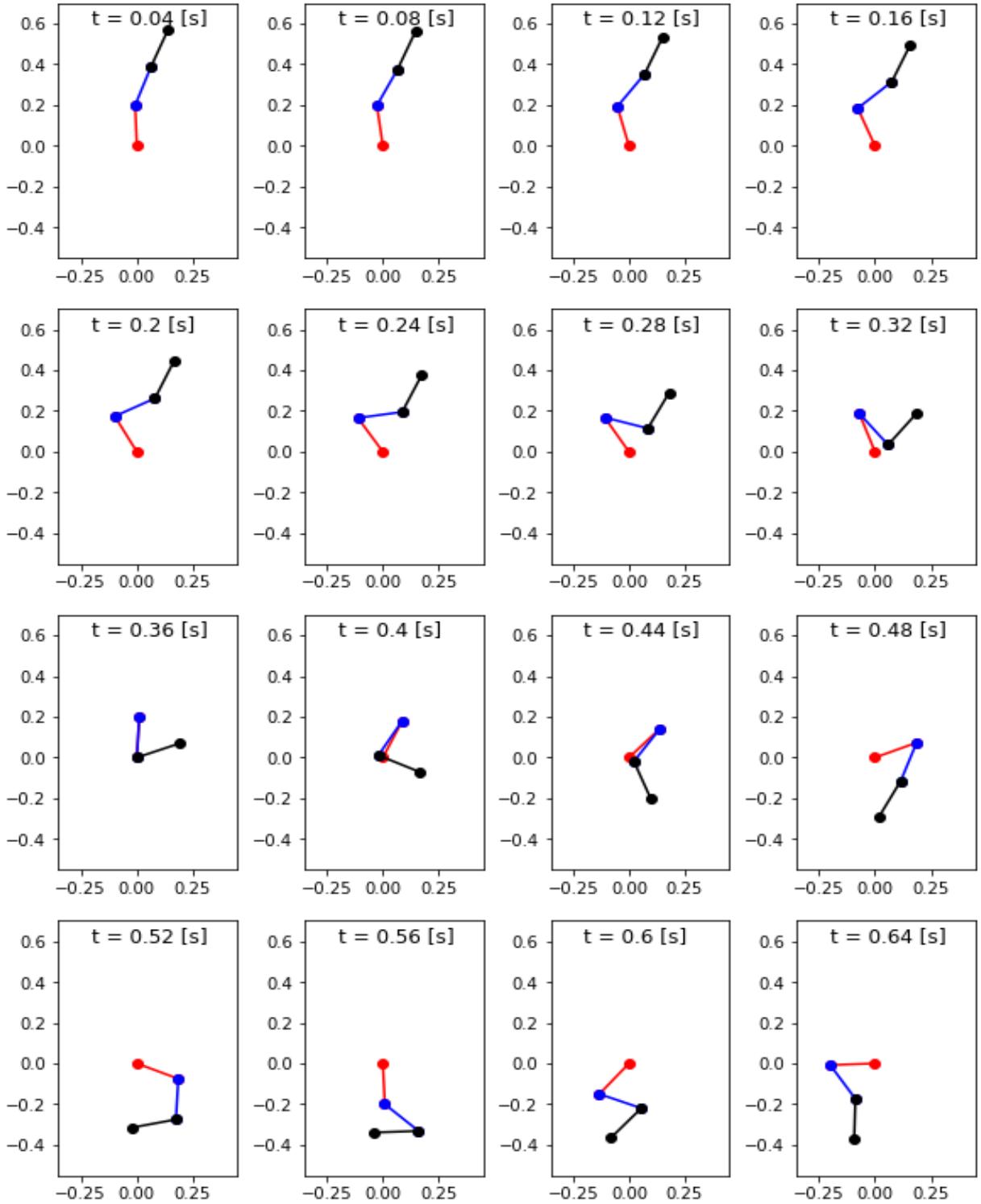


Figura 20

Secuencia de Movimiento Péndulo Triple



5.2 Linealización de sistemas no lineales en Python y MATLAB

El sistema no lineal que se tendrá en cuenta para linealizar corresponde al péndulo invertido de la Figura 13, se tienen dos puntos de equilibrio para el péndulo, cuando el péndulo está totalmente vertical hacia abajo $\theta = 0$ o cuando esta vertical hacia arriba $\theta = \pi$, en ambos casos la velocidad del carro y omega serán cero ($v = w = 0$) por lo tanto el punto de linealización corresponde a *X0=(x 0 0 0)* que recodando guarda relación de manera directa con vector *X=(x,θ,v,ω)* si se recuerda *x* corresponde a la coordenada horizontal del carro, para el cual se puede tomar cualquier punto para *x*, puesto que las ecuaciones (3) y (4) no dependen directamente de la variable de posición (*x*) del carro. Por lo tanto, el procedimiento para linealizar las ecuaciones (3) y (4) en punto *X0= (0 0 0 0)* es el siguiente en el que se recuerda el procedimiento donde se obtuvieron las ecuaciones del modelo no lineal:

```
 %se deben declarar los simbolos que definen el sistema
 syms x v a theta omega alpha u M delta m l g
 % se definen los parametros que requiere la funcion EuLa(L,X,X_dot,F)
 L=(M*v^2)/2+(m*((v + l*omega*cos(theta))^2 + l^2*omega^2*sin(theta)^2))/2-
g*l*m*cos(theta);
 X=[x,theta,v,omega]; %cordenadas generalizadas qi y qi_dot
 X_dot=[v,omega,a,alpha]; %vector de derivadas en el tiempo de X
 F=[u-delta*v 0]; %vector de fuerzas que actuan en cada uno de los grados de
libertad
 sol=EuLa(L,X,X_dot,F); %funcion que resuelve las ecuaciones de Euler-Lagrange
 grad=solve(sol,a,alpha); %muestra cada Ecuacion
 grad.a
```

$$a = \frac{l\,m\,\sin(\theta)\,\omega^2 + u - \delta\,v - g\,m\,\cos(\theta)\,\sin(\theta)}{-m\,\cos(\theta)^2 + M + m}$$

```
 grad.alpha
```

$$\frac{-l\,m\,\cos(\theta)\,\sin(\theta)\,\omega^2 - u\,\cos(\theta) + \delta\,v\,\cos(\theta) + g\,m\,\sin(\theta) + M\,g\,\sin(\theta)}{l\,(-m\,\cos(\theta)^2 + M + m)}$$

```
X0=[0 0 0 0]; %punto de linealización
lin=lin_sis(grad,X,X0);% linealización por series de Taylor
lin.a
```

$$\frac{u}{M} - \frac{\delta\, v}{M} - \frac{g\, m\, \theta}{M} \tag{5}$$

```
lin.alpha
```

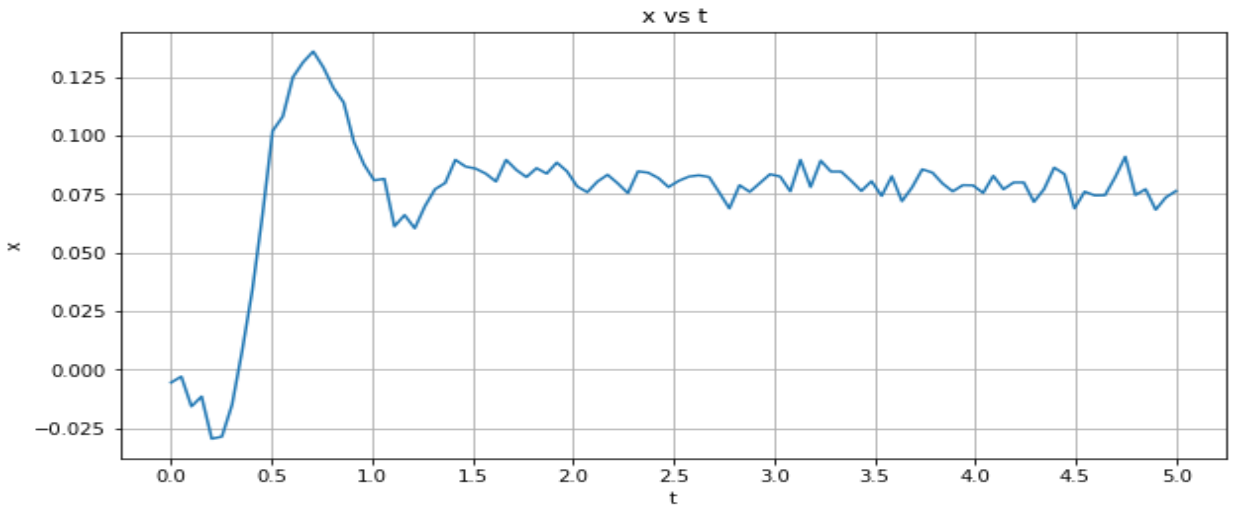
$$\frac{\delta\, v}{M\, l} - \frac{u}{M\, l} + \frac{\theta\,(M\, g + g\, m)}{M\, l} \tag{6}$$

5.3 Identificación de Parámetros Disipativos Mediante Algoritmos Evolutivos.

En esta sección se identifican los parámetros disipativos del sistema dinámico compuesto por un carro con dos péndulos acoplados a el mismo, como se muestra en la figura 15, se utilizan los datos correspondientes al desplazamiento del carro, y el modelo matemático que lo describe, lo cual se halló en la sección de modelamiento, los datos experimentales se generaron con $\delta_1 = 0.1, \delta_2 = 0.2, \delta_3 = 0.3$, y se le agrega ruido para simular datos experimentales.

Figura 21

Datos Experimentales



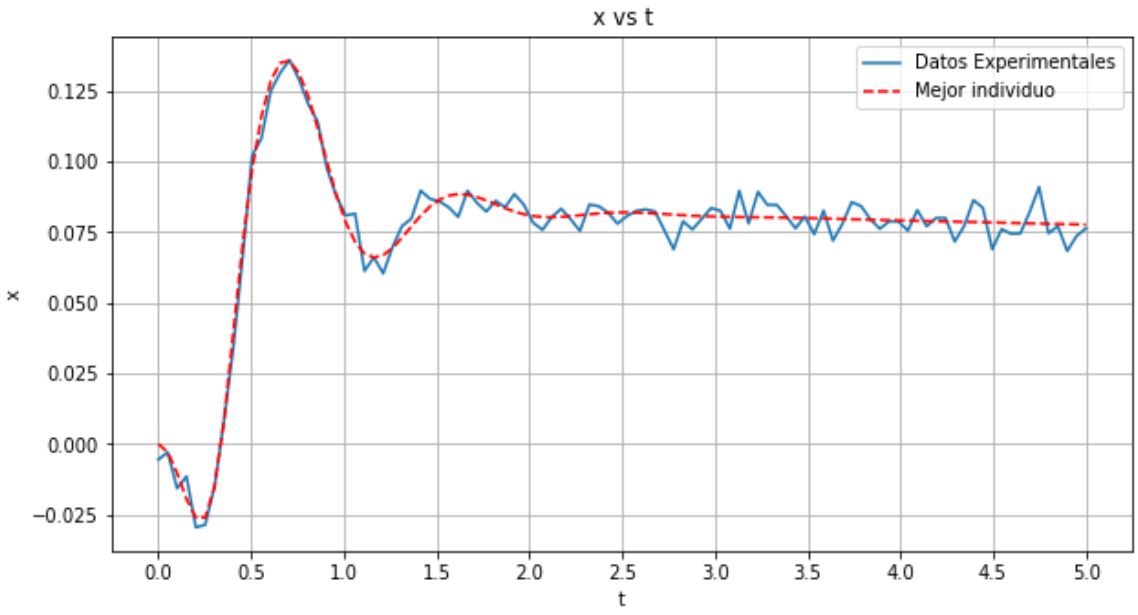
Este problema tiene 3 grados de libertad, donde por cada grado de libertad existe una constante disipativa, por lo tanto, la longitud del cromosoma es 3.

Como se requiere encontrar los parámetros disipativos, se varían los mismos evolucionando los individuos, hasta que se ajusten a la curva de datos, para ello se evalúa la población, con el coeficiente de determinación, el cual indica el grado de ajuste entre los datos y el individuo, de esta manera el mejor individuo, será el que mejor se ajuste a los datos. Los parámetros de ajuste del algoritmo evolutivo fueron los siguientes:

- Numero de generaciones = 30
- Probabilidad de cruce = 0.7
- Probabilidad de mutación = 0.1
- Espacio de búsqueda para todas las variables entre 0 y 10.
- Tamaño de población = 70

Figura 22

Respuesta del AGS_R a la Mejor Adaptación



El mejor individuo con mejor adaptación a los datos Experimentales según la gráfica de la Figura 22 corresponde a:

$$\delta_1 = 0.09562401, \qquad \delta_2 = 0.18538351, \qquad \delta_3 = 0.29663794$$

Se obtuvo una adaptación de 98.42%

El proceso de evolución que realiza el algoritmo evolutivo se muestra a continuación:

Figura 23

Curva de Evolución

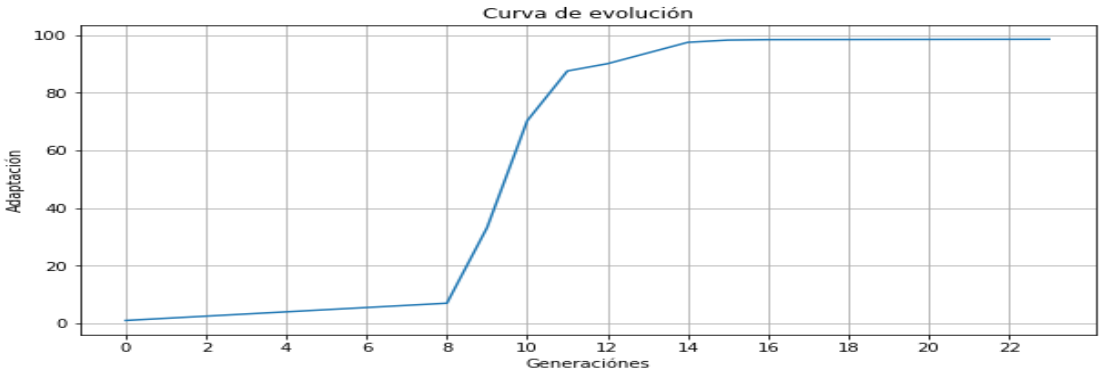
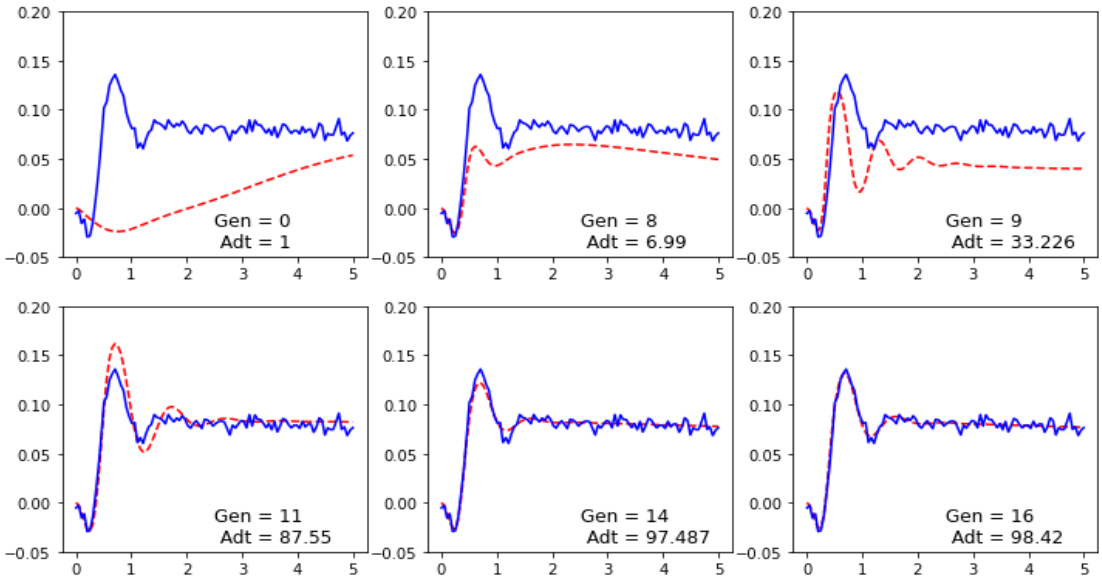


Figura 24

Secuencia de Evolución del AGS_R



6. Conclusiones

Se desarrolló una herramienta en Python y Matlab para facilitar el modelamiento matemático de sistemas dinámicos de tipo mecánico, utilizando la metodología de Euler Lagrange, la cual permite abordar el modelado con un enfoque energético, esta herramienta permite optimizar, el proceso de modelado, realizando automáticamente los procedimientos matemáticos que se requieren en el método de Euler Lagrange, permitiendo que el estudiante enfoque su atención en el planteamiento del problema.

Se desarrollo un módulo en Python y Matlab que permite linealizar sistemas de ecuaciones diferenciales haciendo uso de las series de Taylor, el cual recibe como entrada el sistema de ecuaciones, el vector de estado y el punto donde se quiere linealizar, este módulo realiza automáticamente el procedimiento de linealización, permitiendo al estudiante obtener las ecuaciones linealizadas del sistema en el punto de interés, este módulo es realmente útil para sistemas de múltiples grados de libertad ya que las ecuaciones diferenciales resultan ser muy extensas y el procedimiento muy tedioso.

Se elaboró un módulo de Algoritmos Evolutivos escrito en Python, para solucionar problemas de ingeniería, que se representen con números reales, como es el caso de la identificación de parámetros disipativos, este módulo permite que el estudiante encuentre una solución satisfactoria a problemas de alto grado de dificultad, donde realizar un análisis analítico profundo es impráctico.

En la identificación de parámetros disipativos, se utilizó el módulo descrito anteriormente, con el objetivo de evolucionar el modelo analítico, hasta ajustarlo a la curva de datos experimentales, se usó el coeficiente de determinación (r^2) para calificar el ajuste del modelo a la

curva de datos. Se prueba con el error cuadrático medio y el coeficiente de determinación, siendo este último el que presenta mejores resultados, también se prueba con algoritmos genéticos (Representación binaria del genotipo) pero resulta ser más efectivo los algoritmos Evolutivos de representación real ya que se ajustan mejor al problema.

Se desarrollo material pedagógico de las herramientas en Python y Matlab, para incentivar el uso de la programación en los estudiantes de la escuela de ingeniería mecánica.

Referencias Bibliográficas

Araujo, L., & Cervigón, C. (2009). *Algoritmos Evolutivos Un enfoque práctico* (Alfaomega & RA-MA (eds.); 1st ed.).

Brunton, S. L., & Kutz, J. N. (2017). *Data Driven Science & Engineering - Machine Learning, Dynamical Systems, and Control* (Cambridge University Press (ed.)). databook.uw.edu

Cortés cardona, G. (2017). *Diseño, Simulación y Control de un Péndulo Invertido Doble Lineal para el Laboratorio de Automática de la EIEE de UNIVALLE* [Universidad del Valle]. https://ci.nii.ac.jp/naid/40021243259/

Deb K, A. R. (1995). *Simulated binary crossover for continuous search space. Complex Systems*.

Espinosa Bedoya, A., & Gil parra, S. A. (2019). Modelo cinemático del martillo de un molino de martillos operando sin carga empleando la mecánica de Lagrange. *Ingenierías USBMed*, *10*, 28–33. https://doi.org/10.21500/20275846.3871

Ginsberg, J. (2008). *Engineering Dynamics* (CAMBRIDGE UNIVERSITY PRESS (ed.); 1st ed.).

MATLAB. (n.d.). *MATLAB ® para ingenieros*.

Ogata, K. (1987). *Dinamica de sistemas* (P. HALL (ed.); 1st ed.).

René, B., & Jaimes, A. (2014). Desarrollo de una herramienta en Matlab para Sintonización de Controladores PID, utilizando algoritmos genéticos basado en técnicas de optimización multiobjetivo Development of a tool for tuning in matlab PID controllers , using genetic algorithms based. *Sennova (Colombia)*, *Vol.1*, *No.*, 99. 80-103.

Sala Piqueras, A. (2019). *Linealización de sistemas dinámicos y paso a representación normalizada*. Universidad Politécnica de València. https://riunet.upv.es:443/handle/10251/116824