UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA QUÍMICA, BIOTECNOLOGÍA Y MATERIALES

APLICACIÓN DE TÉCNICAS DE *MACHINE LEARNING* EN EL MODELAMIENTO DE SISTEMAS QUÍMICOS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL QUÍMICO

PEDRO ANDRÉS JOFRÉ ESCÁRATE

PROFESOR GUÍA:
J. CRISTIAN SALGADO HERRERA

MIEMBROS DE LA COMISIÓN:
FRANCISCO GRACIA CAROCA
ALVARO OLIVERA NAPPA

SANTIAGO DE CHILE
2020

RESUMEN DE LA MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL QUÍMICO
POR: PEDRO ANDRÉS JOFRÉ ESCÁRATE
FECHA: DICIEMBRE 2020
PROF. GUÍA: J. CRISTIAN SALGADO HERRERA

Aplicación de técnicas de *Machine Learning* en el modelamiento de sistemas químicos

El *Machine Learning* ha generado avances en varias áreas, siendo una de ellas la modelación matemática de sistemas o fenómenos, el cual es una herramienta que se usa comúnmente en las industrias para predecir o determinar las condiciones de operación, por mencionar algunos usos. A pesar de las ventajas que ha generado, el requerimiento de conocimiento para construir el modelo, así como conocimiento computacional para realizar la simulación numérica se mantienen como limitantes de este proceso.

El objetivo de este trabajo de memoria consiste desarrollar una metodología con la cual se propongan bloques o ecuaciones faltantes en modelos matemáticos descritos por sistemas de ecuaciones diferenciales ordinarias para sistemas químicos de diversa complejidad.

La metodología diseñada consiste, en primer lugar, en una recolección de información y estudio de los datos, seguido de la implementación del algoritmo de programación genética para obtener propuestas del modelo. Finalmente, se realiza un análisis de estos con el objetivo de mejorar la calidad de la siguiente iteración. Sumado a lo anterior, se modifica el algoritmo para agregar características que ayudan a mejorar su desempeño ya sea en la calidad del resultado obtenido o el tiempo de ejecución.

Con esta metodología se logró diseñar los modelos que representan la altura de un líquido dentro de un estanque, Lotka-Volterra y la reacción de oxidación del monóxido de carbono en base a datos obtenidos a partir de simulaciones de dichos modelos. Para los 2 primeros se logran diseñar modelos similares a los originales con diferencia solo en los parámetros; mientras que, para el tercero, se logra obtener información relevante con relación al fenómeno.

Se concluye que la metodología es capaz de proponer modelos de distinta complejidad y puede ser aplicada en otras áreas, ya que las modificaciones realizadas no son específicas para el área de Ingeniería Química. Sin embargo, es necesario tener ciertas consideraciones al momento de usarla, siendo las más importante que la elección de una función de *fitness* adecuada e informativa, así como el análisis de las expresiones matemáticas propuesta por el modelo.

Agradecimientos

Quiero agradecer a mi familia, en especial a mis padres, por apoyarme en este duro proceso, ser un pilar de apoyo y hacer lo que estuviera a su alcance para que yo esté en buenas condiciones. Mi abuela que siempre se preocupa de mí, se asegura que tenga comida al alcance cuando estoy trabajando. A mi tía Verónica, tío Bernardo, al Martín y Pablito por leer mi tesis en distintos momentos y darme sus consejos. A Tatán por apañar en realizar una semana de escritura intensa. A mi profesor guía por el apoyo entregado durante la realización de la memoria.

1 TABLA DE CONTENIDO

Índice de figuras

Índice de Algoritmos

Índice de tablas

Capítulo 1

1 Introducción

1.1 Antecedentes generales

El modelamiento de fenómenos, tanto naturales como artificiales, consiste en representar a través de ecuaciones matemáticas o sistemas de ecuaciones los comportamientos característicos de éstos [1]. Conocer un modelo es de gran utilidad, ya que permite a los profesionales utilizarlos para predecir el estado en que estos se encontrarán en un momento en específico basado en las condiciones en que estos modelos son simulados. Por ejemplo, en la medicina se utiliza el DTI (*Diffusion Tensor Imaging*) para determinar el estado de tractos de materia blanca en el cerebro [1]. Otra utilidad es obtener las condiciones de operación para llegar a los resultados que se desean. Un uso común ocurre en empresas de petróleo donde se puede usar el modelo de Hubbert para simular la producción de aceite [2].

Dentro de los tipos de modelos encontramos los fenomenológicos, que se crean a partir de información y conocimientos del fenómeno de interés. En el caso que sólo exista conocimiento del tipo de variables, pero no de su interacción, se pueden utilizar métodos que obtengan la ecuación a través de un modelo empírico [3].

Existen diversas maneras de construir modelos que tendrán diferentes requisitos y condiciones que están suscritos al fenómeno en estudio y el enfoque que se utilice. Para los modelos fenomenológicos se requiere un conocimiento del fenómeno, mientras que los empíricos requieren datos recopilados. En este último, dependiendo de la forma que tengan los datos, será posible observar comportamientos conocidos que permite diseñar un modelo usando un número limitado de operaciones. Tal es el caso de los modelos lineales donde se utilizan sumas y multiplicaciones. Cuando la forma sea desconocida será posible la utilización de algoritmos computacionales dedicados a resolver ecuaciones matemáticas.

En la ingeniería de proceso es una práctica común el diseño de modelos que son usados para diseñar equipos que deben representar un gran rango de complejidad. Al diseñar es posible encontrar una situación en que la información o características no se encuentren disponibles o donde existan diversas variables (Flujos, concentraciones, tiempo) que no entreguen

resultados óptimos y que interfieran en el trabajo. Por esto, es necesario el desarrollo de un algoritmo que pueda crear modelos y evaluarlos de manera automática reiterando este proceso hasta que encuentre una expresión que logre representar la fenomenología[1] [3] [4].

Machine Learning, como aplicación de la inteligencia artificial, aporta en el diseño de modelos en forma de implementación computacional de algoritmos que aprenden del comportamiento de los datos a medida que se ejecuta, llegando a proponer modelos más complejos y que requiere un menor conocimiento previo sobre la situación de estudio[5]. Uno de los métodos más estudiados y conocidos de esta área es la programación genética, la cual se basa en la teoría de la evolución y que se ha aplicado con éxito en numerosas áreas. Ejemplos de esto incluye el modelamiento de la producción de aceite [6], el modelamiento de la absorción de hierro por el intestino en el área biológica [7], procesamiento de imagen para la mejora y segmentación [8], afinar los parámetros en los controladores PID utilizado en la robótica [9], entre otros.

Gracias al avance del *Machine Learning* en el área de la ingeniería química se pueden obtener modelos de distinta complejidad al usar métodos genéricos para posteriormente analizar el comportamiento representado en el resultado. Esto puede ser mejorado si el método tiene en cuenta los comportamientos conocidos y características del problema al buscar la solución. A partir de lo anterior, el objetivo general es crear una metodología que permita aplicar los métodos de *Machine Learning* en el área química para obtener modelos capaces de representar los fenómenos en estudio.

El presente trabajo se divide en cuatro apartados. El primero de ellos corresponde al marco teórico donde se proporcionan los antecedentes que sustentan la propuesta. Posteriormente, el apartado que hace referencia a la metodología implementada en la investigación. Luego se presentan los resultados junto con una discusión de estos, para finalmente entregar las conclusiones y proyecciones factibles a partir del estudio.

1.2 Motivación

El uso de técnicas basadas en: la inteligencia artificial, el tratamiento de extensas cantidades de datos (o *Big Data*) y algoritmos computacionales complejos, han generado lo que se conoce como industria 4.0. Esto ha generado que las empresas se deban adaptar a nuevas tecnologías para aplicarla en sus respectivas áreas.

Debido a esta nueva implementación, existe una situación donde un conocimiento mínimo en el área de informática sea necesario en todo ámbito, creando una limitante que se encuentra en el número de personas capacitadas o bien, genera un gasto de tiempo al entrenarlas.

El presente estudio ofrece una propuesta metodológica para la aplicación de técnicas de *Machine Learning,* en la obtención de modelos en la ingeniería de procesos para así facilitar su implementación en la industria.

Lo anterior se realiza a través del uso de algoritmos para diseñar modelos de distinta complejidad en sistemas químicos. Esto tiene como finalidad reducir costos en tiempo humano y simplificar el análisis de los resultados de manera que se pueda obtener información que no se poseía originalmente del modelo de interés.

1.3 Objetivos

1.3.1 Objetivo General

- Desarrollar una metodología capaz de proponer bloques o ecuaciones faltantes en modelos matemáticos descritos por sistemas de ecuaciones diferenciales ordinarias para sistemas químicos de diversa complejidad.

1.3.2 Objetivos Específicos

- Estudiar los parámetros más importantes que afectan el funcionamiento de la programación genética y las dificultades que pueden afectar su desempeño.
- Buscar e implementar técnicas para disminuir el efecto de los problemas comunes que enfrentan este algoritmo.
- Evaluar el uso de la programación genética al devolver modelos en su totalidad o parte de estos en fenómenos químicos.

Capítulo 2

2 Antecedentes y Marco teórico

En el presente apartado, se revisarán las bases teóricas necesarias para comprender los capítulos posteriores. En primer lugar, se proporcionan antecedentes teóricos en el área de la matemática; por lo que se destacarán contenidos asociados al modelamiento y tipos de modelos, algoritmos y teorema utilizados. En segundo lugar, el capítulo destaca los sistemas químicos que utiliza el presente estudio, mientras que el subapartado de *Machine Learning* expone la información central que justifica el uso de esta aplicación en la investigación. Finalmente, se exponen problemas comunes que afectan metodologías similares y posibles soluciones que ofrece la literatura.

2.1 Modelamiento

El modelamiento consiste en representar los fenómenos de interés de manera que permita un mejor análisis del fenómeno en cuestión [10]. Los modelos se han utilizado en la industria (incluyendo de alimentos, eléctrica y química) para diseñar equipos. Entre estos se encuentran los intercambiadores de calor, hornos y condensadores que son necesarios para controlar la temperatura requerida en un procedimiento específico y se diseñan a partir de modelos creados a partir de diferentes ecuaciones como la transferencia de calor y convección[11][12]. En este contexto, los modelos se ponen a prueba para analizar cómo reaccionan frente a las nuevas condiciones expuestas, entregando resultados variables. Aquellos modelos que representan de forma fidedigna el fenómeno de interés son usados para comprender el comportamiento del fenómeno. En caso de que no represente el resultado final, es posible volver a diseñar un modelo con la ventaja que no se parte desde cero, lo que aumentaría la eficacia del resultado. Esta acción puede llevarse a cabo hasta obtener un resultado satisfactorio [1].

Los modelos permiten obtener un mayor conocimiento de los fenómenos y el comportamiento que tendrán. De esta manera, es posible utilizarlos con distintos objetivos entre los que se deben mencionar: ser capaz de predecir los resultados cuando se cambian las condiciones iniciales o calcular las condiciones de ejecución para obtener el resultado esperado [1][2]. Debido a que la aplicación de modelos no depende de un área en particular, existe una gran variedad de ellos. Incluso la Biblioteca de Stanford hace mención solo a 25 tipos de modelos distintos[13].

Los modelos matemáticos, denominados en adelante como "modelos", fueron definidos por Bender Edward como "una construcción matemática abstracta, simplificada y relacionada a una parte de la realidad y creada para un fin en específico" [10]. La ventaja de este tipo de modelo recae en que utiliza expresiones matemáticas precisas, universales y ampliamente modificables. Además, su implementación puede lograrse por medio de múltiples teoremas y/o nuevas tecnologías centradas en realizar cálculos matemáticos.

En la ingeniería de procesos, desde el inicio del siglo 20, los modelos se utilizan de manera rutinaria para el diseño de equipos [14]. Un uso de esta aplicación es diseñar una red de intercambio de calor óptima dentro de una planta con el objetivo de obtener la mayor recuperación de calor posible, lo cual se traduce en un ahorro de recursos económicos [15] [16] Otro ejemplo es la destilación, uno de los procesos químicos más utilizados, que se caracteriza por separar y purificar mezclas de compuestos. Por ejemplo, en la industria del vino es necesario un diseño minucioso para la destilación del etanol, debido a que contiene una cantidad de sustancias que sobrepasa los 200 compuestos.[17].

El tipo de modelo que se utilizará en este estudio corresponden a los sistemas químicos.

2.2 Sistemas químicos

S.K. Gupta y M. Ramteke definen los sistemas químicos como un set de ecuaciones del tipo Y=F(X,P), el cual representa un modelo para determinar el estado "Y" del sistema en un momento determinado, en base a los valores de las variables "X" y los parámetros "P" [14] . La complejidad de los modelos es amplia, por lo que podría variar la identificación de sistemas simples como determinar la altura de llenado de un estanque en un periodo de tiempo, como también sistemas más complejos como la reacción de oxidación de monóxido de carbono (CO). A continuación, se presentan los modelos específicos que se trabajan en el desarrollo de la memoria.

2.2.1 Altura de un líquido almacenado en un tanque

Para monitorear y controlar la producción, es frecuente que en la industria se resalte la importancia de conocer el estado de funcionamiento de equipos o los flujos de materiales[18]. En este contexto, los modelos pueden ser útiles para obtener estos valores y compararlos con los resultados que se obtengan a través de mediciones dentro de la empresa con la ayuda de equipo especializado. Esto permite identificar discrepancias tempranas que pueden entregar información falsa asociada con desperfectos en el sistema[19].

Para modelar los estados de un equipo, se puede utilizar la conservación de masa y energía que están representados por las ecuaciones (1) y (2) respectivamente

$$\frac{dM}{dt} = F_e - F_s + rxn \tag{1}$$

$$\frac{dQ}{dt} = E_{ent} - E_{sal} \tag{2}$$

En las ecuaciones anteriores, el cambio de masa a través del tiempo está determinado por, "F_e " que es el flujo másico que entra, "F_s " el flujo másico de salida, y "rxn" la reacción presente en el equipo. El cambio de calor "Q" está dado por la diferencia entre "E_{ent}" la energía que entra al sistema y "E_{Sal}" la energía que sale del sistema.

Dependiendo del equipo, proceso y objetivo del modelo, este puede variar de complejidad, usando ambas ecuaciones o solo una de ellas. Este trabajo se centra en un sistema simple, ya que el cambio de energía es 0, no posee reacción dentro del estanque, el flujo de entrada es constante y el flujo de salida depende de la altura de llenado (h) del estanque y el coeficiente de caudal (c). A partir de lo anterior, la ecuación queda como se muestra en (3). Como el flujo no sufre cambio en la densidad los flujos másicos se pueden expresar como flujo Volumétrico (Fve) dividido por la densidad, obteniendo la ecuación (4). Además, si el estanque mantiene un área superficial constante se obtiene la forma de (5), la cual al ordenarla se obtiene finalmente la ecuación (6).

$$\frac{dM}{dt} = Fe - Fs \tag{3}$$

$$\rho * \frac{dV}{dt} = \rho * Fve - \rho * h * c \tag{4}$$

$$A * \frac{dh}{dt} = Fve - h * c \tag{5}$$

$$\frac{dh}{dt} = \frac{Fve - h * c}{A} \tag{6}$$

2.2.2 Lotka-Volterra

El modelo de Lotka-Volterra, también conocido como el modelo de presa-depredador, es un sistema de ecuaciones diferenciales, ecuación (7) y (8), que se usan comúnmente para representar la dinámica biológica entre 2 especies, donde "y" representa un depredador y las presa con "x" a través del tiempo "t", donde "α" y "β" son la reproducción y muerte por depredadores de las presas respectivamente, mientras que "σ" y "γ" representan la reproducción y

muerte de los depredadores respectivamente. La ecuación recibe su nombre de las dos personas que realizaron el mismo descubrimiento en distintas circunstancias. Primero fue estudiada por Lotka en 1920, al observar un comportamiento oscilatorio en ciertos experimentos de laboratorio; mientras que, Volterra, postuló el mismo modelo para explicar el comportamiento entre la población de peces y tiburones en 1925 [20]. Existen dos estados estacionarios para la ecuación:

1. Cuando x=y=0: en este caso ambas especies están extintas.
2. Cuando $x = \frac{\sigma}{\gamma}$ e $y = \frac{\alpha}{\beta}$: si en algún momento se cumple esta condición entonces el sistema entra en un estado estacionario ya que, la reproducción será igual a la muerte.

$$\frac{dx}{dt} = \alpha * x - \beta * x * y \quad (7)$$

$$\frac{dy}{dt} = \sigma * x * y - \gamma * y \quad (8)$$

Para cualquier otro caso las poblaciones mostraron un comportamiento oscilatorio con un periodo de $\frac{2*pi}{\sqrt{\alpha*\sigma}}$ lo cual se puede observar en la Figura 1.

En un inicio, las cifras asociadas a las presas van disminuyendo ya que su consumo es mayor que su reproducción. En cuanto a los depredadores, estos aumentan llegando a un punto en el que las presas están por desaparecer. En este momento, se revierte la situación, es decir, aumentan las presas y disminuyen los depredadores, ya que estos últimos no poseen alimento suficiente para sostenerse.

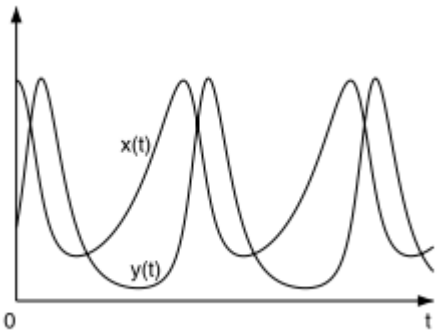

Figura 1 Representación de un modelo Lotka-Volterra[20]

La ecuación de Lotka-Volterra se relaciona con el área química a través de las reacciones catalíticas, las cuales corresponden a las mayorías de los sistemas que presentan oscilaciones en esta área [21], que son aquellas donde la velocidad de reacción aumenta con el tiempo alcanzando un *peak* para luego disminuir. Esto ocurre porque la velocidad está relacionada con la

concentración de los reactantes y en cierto tipo de reacciones algunos son productos de alguna de la reacciones del sistema, siguiendo el comportamiento mostrado en las ecuaciones (9), (10) y (11) donde "A", "B" y "C" son compuestos químicos y "k1", "k2", "k3" son la velocidad de la reacción, en este caso "C" el cual se puede obtener por la reacción presentada en las ecuaciones (9) o (11), al inicio solo se posee "A" pero a medida avanza la reacción la concentración de "B" aumenta provocando la segunda reacción generando "AB", el cual a su vez puede reaccionar para formar C, aumentando la obtención de este producto. [22]

$$\mathbf{A} \overset{k_1}{\rightarrow} \boldsymbol{B} + \boldsymbol{C} \tag{9}$$

$$\mathbf{A} + \mathbf{b} \overset{k_2}{\rightarrow} \boldsymbol{AB} \tag{10}$$

$$\mathbf{AB} \overset{k_3}{\rightarrow} \boldsymbol{B} + \boldsymbol{C} \tag{11}$$

El comportamiento anteriormente descrito, se puede apreciar en la Figura 2. En la imagen se muestra la concentración de distintos compuestos en una reacción de cloración del benceno.

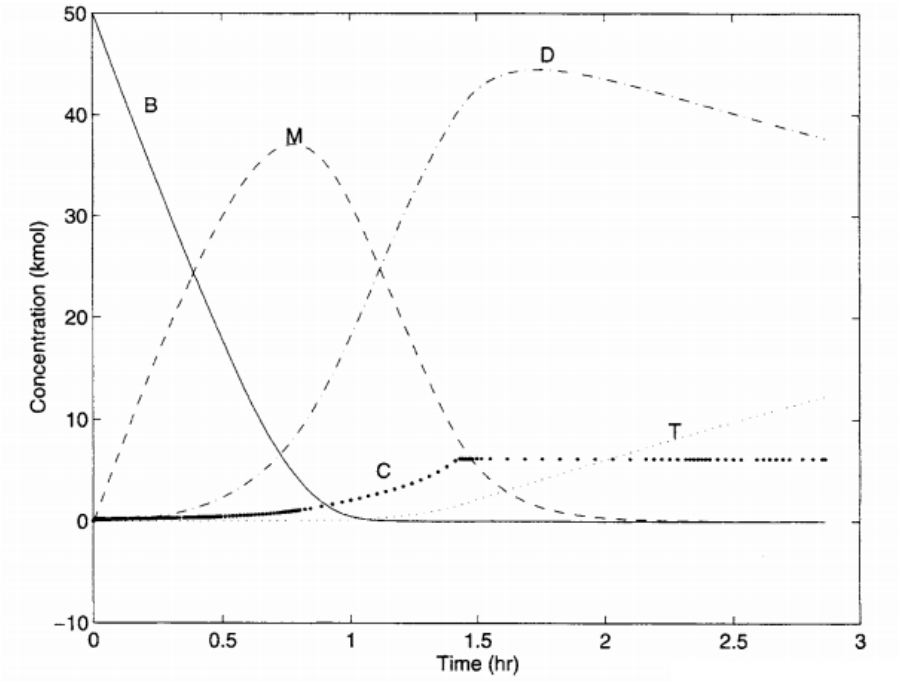

Figura 2 Concentración de compuestos en una reacción de cloración de benceno[22]

2.2.3 Oxidación del monóxido de carbono

La reacción de oxidación de CO sobre catalizador de platino fue un tema de interés y estudio en el siglo XX, ya que originalmente se pensaba que la

reacción seguía un mecanismo simple. Sin embargo, a medida que la tecnología avanzaba se descubrió que las cualidades del catalizador jugaban un rol importante en el mecanismo [23]. Así, en 1981 J. R. Crelghton, F.H. Tseng, J. M. White, y J. S. Turner se propusieron el sistema para la reacción mostrado a continuación:

$$\frac{d\theta_{CO}}{dt} = \frac{p_{CO}}{\left(2\pi m_{CO} \hbar T_g\right)^{1/2} N_s} S_{CO}\left(\theta_{CO}, \theta_O, T\right) - \upsilon \theta_{CO} e^{-\frac{E_d + \beta\theta_{CO}}{RT}} - A_{LH} e^{-\frac{E_{LH}}{RT}} \theta_{CO} \theta_O \quad (12)$$

$$\frac{d\theta_O}{dt} = \frac{p_{O_2}}{\left(2\pi m_{CO} \hbar T_g\right)^{1/2} N_S} S^0_{O_2}(T)(1 - \theta_{CO} - \theta_O)^2 - A_{LH} e^{-\frac{E_{LH}}{RT}} \theta_{CO} \theta_O \quad (13)$$

$$\frac{dp_{CO_2}}{dt} = -\frac{\bar{S}(CO_2)}{V} p_{CO_2} + \frac{A\hbar T_g}{V} N_s A_{LH} e^{-\frac{E_{LH}}{RT}} \theta_{CO} \theta_O \quad (14)$$

El sistema de ecuaciones anterior representa el cambio de moléculas presentes en el catalizador. El primer término de las ecuaciones (12) y (13) corresponde a la absorción de las moléculas del gas, la que depende de la frecuencia de colisiones de las partículas la cual se representa por, $\frac{p_i}{2\pi m_i k T_g}$ donde"m_i" es el peso, "T_g" la temperatura, "p_i" la presión con el subíndice "i" indica el gas que está representado, Tg es la temperatura de la placa y k constante de Boltzmann. Por su parte "N_s" es el número de sitios activos en el catalizador, "S_i" es el coeficiente de *sticking*, el cual es propio de cada compuesto y depende de "T" que representa la temperatura del sustrato.

Para la ecuación (12), el segundo término corresponde a la desorción del CO con "v" un factor pre-exponencial, $E_d + \beta * \theta_{CO}$ la energía de desorción, con E_d la energía de activación y β constatne de, "T" la temperatura del substrato, R la constante de los gases ideales. Es necesario agregar que este modelo asume una desorción nula para las moléculas de oxígeno. El último término en las ecuaciones (12) y (13) representa la velocidad de reacción que posee la forma de Arrhenius. La ecuación (14) muestra el cambio de la presión parcial para el CO2 donde el primer término muestra la disminución debido a una bomba que extrae el dióxido de carbono desde el interior $\bar{S}(CO_2)$, con V el volumen del reactor, mientras que el segundo término muestra el aumento de presión debido a la reacción de monóxido con el oxígeno. Es necesario destacar que el modelo asume que una vez se forma el CO2 este se separa instantáneamente del catalizador [24].

2.3 Algoritmos

Un algoritmo, en términos generales, corresponde a una secuencia de pasos o instrucciones que deben seguirse para obtener la solución de algún problema. Existe una gran variedad de tipos de algoritmos, pero esta memoria se centra en los algoritmos de optimización que son aplicados en diversas áreas. Por ejemplo, en la industria pueden usarse para tomar decisiones con respecto a la producción [25], planificar los horarios de trabajo [26], obtener el modelo de la operación unitaria, entre otras.

Existen diversos tipos de algoritmos. Una clasificación plausible distingue algoritmos estocásticos o determinísticos según la presencia de aleatoriedad. El primero de ellos corresponde a los algoritmos donde por lo menos uno de sus pasos está determinado por el azar, provocando que probablemente ninguna iteración de este sea similar a otra. En cambio, los determinísticos siguen los pasos de forma rigurosa lo que conlleva que al mantener las mismas condiciones se obtendrán los mismos resultados [27].

2.3.1 Teorema No Free Lunch

Existe un gran número de algoritmos para diseñar modelos. Estos algoritmos requieren un tiempo de ejecución y un gasto computacional, lo que varía dependiendo del algoritmo. Por este motivo, los resultados obtenidos, es decir el modelo, pueden ser distintos y con diferente eficacia, más aún no existe ningún algoritmo que sea óptimo para todo tipo de problema a la vez. Lo anterior fue demostrado en 1997 por David H. Wolpert y William G. Macready quienes propusieron un nuevo teorema denominado "No Free Lunch" (NFL) [28], el cual establece que, si se posee un algoritmo de optimización cualquiera y este realiza un buen trabajo para un tipo de problema específico, entonces su eficiencia en el resto de los problemas se ve afectada y existirán otros algoritmos que darán un mejor resultado.

Este teorema justifica la metodología propuesta en la presente investigación, ya que es preciso realizar modificaciones en un algoritmo para adaptarlo a la situación requerida. Es preciso tener en consideración que los algoritmos utilizados para dar solución a los casos propuestos en el presente estudio no serán óptimos si se replican en otros contextos, puesto que podrían comprometer los resultados.

2.4 Machine Learning

El *Machine Learning* es una aplicación de la inteligencia artificial que se centra en la creación de *softwares* que aprenden a medida que son ejecutados. Dependiendo del tipo de dato que se posea y el objetivo establecido, el *Machine Learning* se clasifica en 4 enfoques [29]:

1. Aprendizaje supervisado. Aquel donde se posee una clasificación entre datos input que se le entrega al programa y output deseado. Entre los métodos se encuentra la regresión simbólica, árboles de decisión, *support vector machine* (SVMs), entre otros [30]. Sus aplicaciones más comunes incluyen la clasificación y las regresiones [5].

2. Aprendizaje no supervisado. Se utiliza principalmente cuando se posee un gran número de datos sin clasificación [31]. Ejemplos de algoritmos incluyen: *clustering*, que se dedican a juntar los datos que poseen similitudes entre ellos; reducción de dimensión, donde el algoritmo trata de simplificar los datos lo mayor posible sin perder generalidad; la visualización, donde el objetivo del programa es crear una representación 2D o 3D de los datos [5].

3. Aprendizaje reforzado. En este caso, el programa no posee un set de datos objetivos a los que llegar, sino que a través de ensayo y error mejora su comportamiento frente a un problema específico. Un símil en su ejecución es el entrenamiento de un perro donde, si se comporta de la manera deseada, se le dará un premio, reforzando el actuar [5]. Entre las aplicaciones más comunes está la robótica, área en que los algoritmos permiten que se logre el manejo automático de autos [32].

4. Redes neuronales. Para este tipo de aprendizaje, el objetivo es simular el funcionamiento del cerebro a través de neuronas. Las redes neuronales pueden ser utilizadas para distintos objetivos, desde recomendaciones de videos basados en el historial de visualización en YouTube, hasta crear una red neuronal que pueda enfrentarse a los campeones de distintas disciplinas. Recientemente, Google diseñó Alphastar y Alphago, dos redes neuronales que fueron capaces de vencer a los actuales campeones del mundo en los juegos estratégicos de StarCraft II y Go respectivamente [33] [34].

El *Machine Learning* no solo sirve para mejorar o encontrar soluciones a problemas conocidos como en los ejemplos presentados, sino que, además, pueden ser utilizado para enfrentar y/o analizar situaciones impredecibles. Un

ejemplo de ello es la propuesta asociada a la implementación de taxis que se manejen de forma autónoma en un contexto de pandemia, proyecto en fase de prueba durante el 2020 [35].

A continuación, se detallará la Regresión simbólica la cual hace referencia a un método del aprendizaje supervisado. Esto resulta relevante puesto que es el método en que se basa el diseño del algoritmo de programación genética, propuesta metodológica de esta memoria.

2.5 Regresión simbólica

La regresión simbólica corresponde a un método de Machine Learning enfocada en la búsqueda de funciones en un espacio matemático, donde las ecuaciones son representadas por árboles binarios. Los nodos internos representan una función, mientras que los terminales, que pueden ser las constantes, variables y/o parámetros, se ubican en los nodos externos conocidos como hojas. El primer nodo, identificado como la raíz del árbol, y la profundidad de éste, se mide como la máxima distancia entre la raíz y sus hojas. De esta forma, el árbol de la Figura 3 representa la función Max(x+x, x+3*y) siendo la función Max() el nodo raíz con una profundidad de 4. Estos árboles son creados aleatoriamente para ser posteriormente evaluados y basado en su representatividad de los datos en un proceso denominado *fitness* [36].

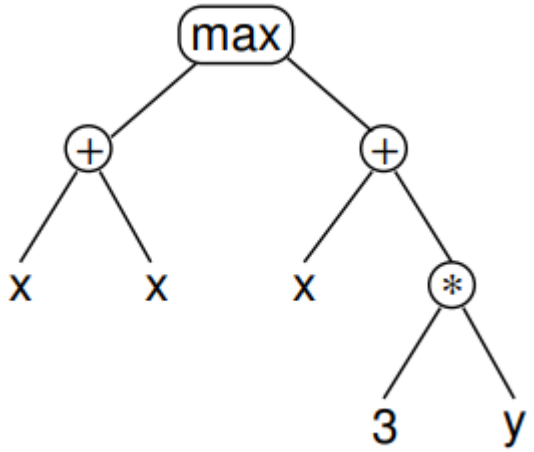

*Figura 3 Representación de un árbol binario para la función Max(x+x, x+3*y)*

Una vez analizados, es necesario crear un nuevo set de árboles. Estos son creados a través de operadores genéticos, los cuales se pueden observar en la Figura 4. El procedimiento puede utilizar la mutación que consiste en reemplazar parte de un subárbol del individuo con una nueva función aleatoria. También, se puede utilizar el cruzamiento, operador genético que elige dos

ecuaciones para crear una nueva. Este último operador escoge un subárbol de una de las ecuaciones previas y lo reemplaza en otro individuo.

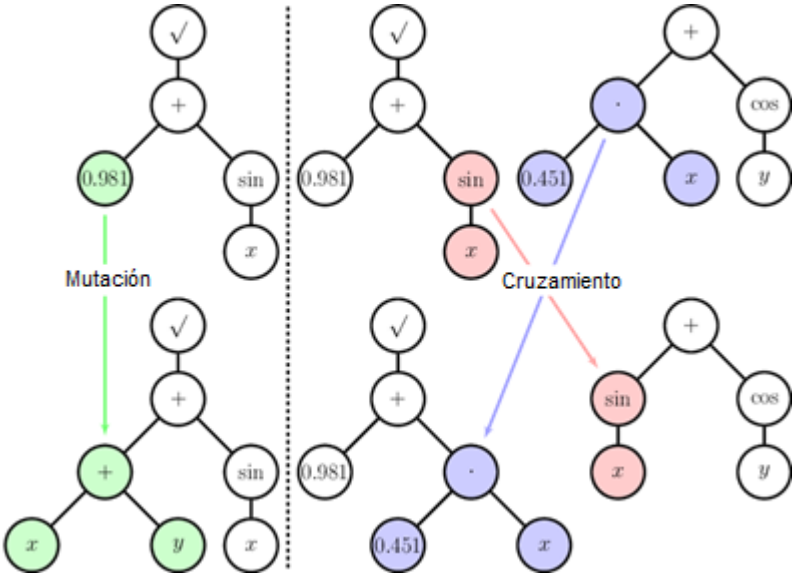

Figura 4 Operadores genéticos. Adaptada de [37]

El efecto de la mutación ayuda a que la solución no quede atrapada en un mínimo local [38]. Lo anterior se debe a que, si encuentra un mínimo local, lo cual está representado por la letra azul en la Figura 5, los subárboles se empiezan a repetir en los nuevos sets, reduciendo la diversidad lo que evita encontrar el óptimo global, cual está representado por la letra roja en la Figura 5. La mutación soluciona esto, ya que asegura una diversidad al crear algunos árboles utilizando la aleatoriedad.

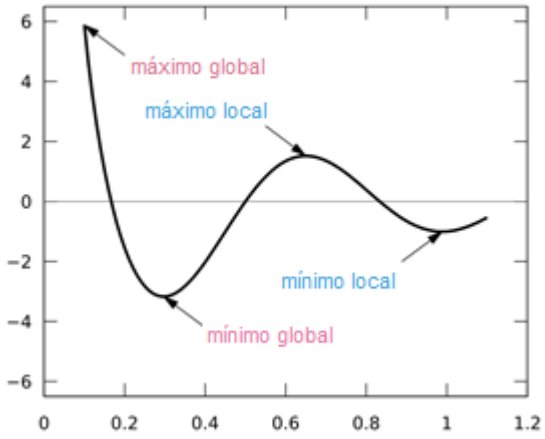

Figura 5 Óptimos locales y globales de una funcion.

Dentro de la regresión simbólica existen distintos métodos basados en un comportamiento propio de la naturaleza. Dentro de la más popular está la programación genética (P.G.), una adaptación de los algoritmos genéticos (A.G.) basada en la teoría de la evolución [27]. También se encuentran

aquellos basados en la inteligencia de enjambre, imitando el comportamiento colectivo de un grupo de individuos. Ejemplos incluyen el algoritmo de colonia de hormigas [39] y la programación artificial de colonia de abejas [40]. También existen métodos menos estocásticos como el *Fast Function Extraction* (FFX) que busca la mejor ecuación y parámetros a través de un algoritmo de red elástica, de modo que contrarreste el problema de la naturaleza estocástica de los demás algoritmos teniendo un método más determinístico [41].

2.6 Programación genética

La programación genética (P.G.) es una forma de regresión simbólica que se basa en la teoría de la evolución de Darwin, la cual postula que, los animales que mejor se adaptan al medio tienen mayor posibilidad de entregar sus genes a la siguiente generación [27]. De esta forma, si consideramos a los posibles modelos como los individuos, representados como árboles binarios, el *fitness*, como el índice de adaptación al medio y los genes como los subárboles; se puede aplicar el mismo método para encontrar la expresión que mejor representa los comportamientos del fenómeno.

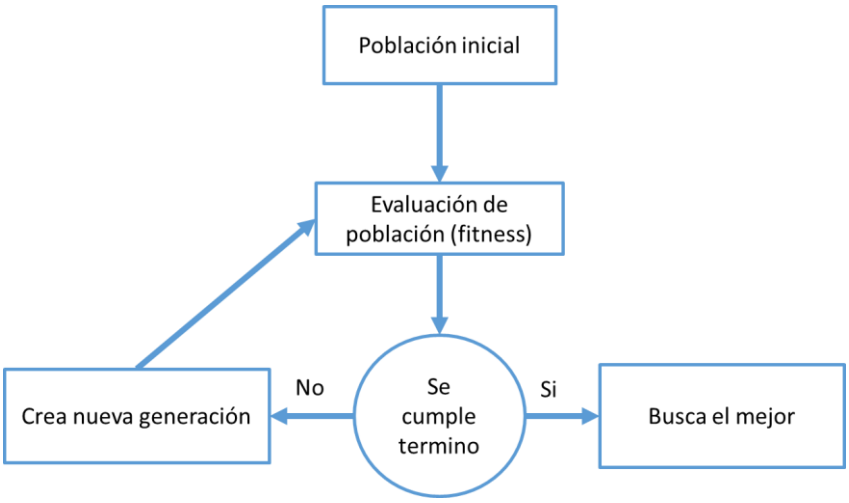

Figura 6 Esquema simplificado de la Programación Genética.

La programación genética sigue el algoritmo mostrado en la Figura 6, donde se crea una población inicial de posibles soluciones las que se evalúan en el proceso de *fitness*; después, se examina si se cumple alguna o varias condiciones de término. Estas condiciones pueden estar relacionadas, pero no limitadas, por lo que podrían existir otras condiciones de término como, por ejemplo: máximo número de generaciones y/o la convergencia de solución, donde una convergencia del 95% en la población se considera suficiente [42]. Si ninguna de estas condiciones es alcanzada, entonces se crea una nueva generación utilizando distintos métodos entre los que se encuentran los operadores genéticos o el elitismo que consiste en mantener un porcentaje de

los mejores individuos de la generación. En caso de que las condiciones de término sean alcanzadas, se devuelve el mejor individuo de la generación.

La Programación Genética (PG) tiene ciertas ventajas con respecto a otros algoritmos [27]:

1. No depende del gradiente (valor vectorial que representa la dirección en que más rápido varía la función), esto le permite trabajar con problemas complejos y/o discontinuos, aumentando el espectro de problemas donde puede ser aplicada la PG.

2. Exploratorio: dada la naturaleza estocástica de los métodos utilizados para crear la población, le permite a la PG un mayor rango de posibles soluciones a probar.

3. Paralelismo: como el cálculo de *fitness* de cada individuo es independiente del resto, al implementar la aplicación computacional del algoritmo, es posible calcular el *fitness* de la generación al mismo tiempo en que se reduce el tiempo de ejecución.

4. Es un método estocástico ya que las ecuaciones son creadas de manera aleatoria como una combinación de diversas funciones, constantes y parámetros, esto disminuye la posibilidad que el programa se atasque en mínimos locales

También posee ciertas desventajas, entre las cuales las más importantes son:

1. Tiempo de ejecución, el cual puede variar desde segundos hasta varias horas dependiendo de la complejidad del problema.

2. Necesidad de especificar buenos valores de parámetros del programa, los que van desde elegir número de individuo y generación hasta los valores de las constantes a utilizar.

A continuación, se presentan los parámetros que afectan el resultado de la programación genética y cómo se generan nuevas poblaciones.

2.6.1 Parámetros

Una de las desventajas de la Programación Genética (PG) es que el resultado se ve afectado por parámetros, entendidos como factores que inciden en la ejecución del algoritmo, por lo que se vuelve necesario estudiar los valores que mejor se le asignen, tales como el espacio matemático, la función de adaptación, número de generaciones e individuos, profundidad máxima y las condiciones de término.

2.6.1.1 Espacio matemático

El espacio matemático se refiere a los distintos conjuntos de valores que son utilizados para rellenar los árboles binarios, los cuales se dividen en 2, funciones y terminales, que se ubican en los nodos internos y las hojas, respectivamente.

2.6.1.1.1 Funciones:

Las funciones que utiliza el algoritmo deben ser especificadas por el usuario y deben cumplir con la propiedad de clausura [42], en la cual se deben cumplir dos condiciones:

1. Capacidad de evaluación. Debido a que la PG se utiliza en diferentes tipos de problema, existen distintos tipos de primitivas (tipos de datos utilizados en la programación). En la Tabla 1 se muestran algunos de los que pueden ser utilizados. Debido a esto, es necesario que las funciones sean consistentes entre ellas y entreguen el mismo tipo de resultado, ya que, al sustituir subárboles dentro de otro árbol, estas funciones pueden chocar al recibir un tipo de argumento no válido.

Como los árboles se crean de manera aleatoria, se pueden crear ecuaciones que no debieran ser evaluadas, por ejemplo, Log(1-X), donde "X" son los datos de las variables. Al ser evaluada, se puede tener un valor X=1 que provoca que el programa deba evaluar la expresión Log(0). Lo cual matemáticamente hablando es incorrecto, ya que el dominio de la función logaritmo consiste en los valores desde cero hasta el infinito positivo, sin incluir los extremos. Para evitar este problema, es necesario crear una nueva versión de la función, denominadas funciones protegidas, las que realizan el mismo proceso, pero tienen una condición extra para tratar los casos complejos. Siguiendo con el ejemplo del logaritmo natural, la forma protegida primero examina que el valor

a evaluar no sea 0 o negativo. Si lo es, debe devolver un valor predeterminado. En el caso contrario, calcula el logaritmo del valor.

Primitiva	Ejemplo
Int	[-1,-2,0,1,2,3]
Float	[pi,0.1]
Booleano	[Verdadero,Falso]
Char	[a,b,c,d…]
String	[abc,manzana]

Tabla 1 Tipos de primitivas usada en la programación

2. De suficiencia. La obligatoriedad en que las funciones entregadas deben ser suficientes para poder representar el fenómeno en estudio.[42]

2.6.1.1.2 Terminales

Los terminales corresponden a los valores que se alojan en las hojas de los árboles binarios y pueden ser divididos en 3 tipos:

1. Variables: Son los valores que se le entrega a los programas

2. Constantes: Son los valores fijados por el usuario con anterioridad.

3. Funciones sin argumentos: también conocidas como operación nularia y corresponden a las funciones que no necesitan de un parámetro para entregar un valor. Un ejemplo es la función rand() de Matlab que devuelve un número aleatorio entre 0 y 1.

En general, una aplicación computacional del algoritmo puede dar un buen resultado utilizando un set de constantes predeterminados, ya que a través de la evolución de los árboles puede crear los valores de los parámetros que busca. Si bien lo anterior puede entregar buenos resultados, proporcionar valores cercanos desde un inicio, ayuda a disminuir el esfuerzo de encontrar estos parámetros permitiendo poner más énfasis en el diseño del modelo.

2.6.1.2 Función de adaptación

La función de adaptación, o función de *fitness*, define el proceso según el cual se evalúa cada individuo de la población, entregando un valor numérico que representa qué tan fidedigna es la solución propuesta para el problema que se enfrenta. Como el *fitness* varía dependiendo la situación, el análisis del valor del *fitness* también cambia. Por ejemplo, en los problemas de regresión es

común utilizar el error cuadrático medio que entrega el promedio de las diferencias cuadráticas entre los valores obtenidos $f(X_i)$ y los deseados Y_i: en este caso, mientras menor sea el valor, mejor será el resultado.

$$E = \frac{1}{n}\sum_{i}^{n}(f(x_i) - Y_i)^2 \quad (15)$$

2.6.1.3 Número de generaciones

Este parámetro indica la cantidad máxima de generaciones que puede alcanzar el algoritmo. Este parámetro tiene un efecto directo en el resultado tanto en el tiempo como en su complejidad. Por una parte, un número pequeño tomará menor tiempo si es que se ocupan todas las generaciones, pero entregará resultados no tan complejos que podrían no ser representativos para los datos. Por otra parte, un valor elevado del parámetro no solo aumentará el tiempo de ejecución, sino que, además, incrementará el riesgo de incurrir en problemas relacionados con la regresión en la solución, como el *bloat* y la sobreestimación [43]. Estos casos serán explicados en la sección 2.7 Problemas de la regresión".

A diferencia de los demás parámetros que poseen ciertos criterios para ser escogidos, este normalmente se escoge por referencia a algún experimento similar o también se puede utilizar la afinación [44] [45].

2.6.1.4 Número de individuos:

Este parámetro especifica el número de individuos que tendrá la población en cada generación. Es necesario escoger un valor adecuado, ya que existe un *trade-off* entre ejecución y la calidad de la solución, es decir, una mayor cantidad de individuos tendrá mayor probabilidad de entregar una buena solución, pero demanda un mayor gasto, tanto computacional como de tiempo en su ejecución. Por su parte, una población menor aumenta la posibilidad de entregar soluciones de baja calidad, ya que reduce la diversidad de una población.

Para enfrentar el problema de escoger un número adecuado de individuos se puede abordar de diversas formas, entre las que destacan: utilizar un valor fijo proveniente de literatura (durante 1992 un buen número a utilizar era 50); y usar álgebra para calcular y abarcar cada posible solución, entre otras [43].

Sin importar la forma que se utilice para seleccionar este parámetro, es necesario tener en cuenta distintos factores, que incluyen: la diversidad de la población, la complejidad del problema, la selectividad y el espacio matemático que se utiliza en el algoritmo [46]. Dichos factores se explican a continuación:

1. La diversidad: refiere a qué tan diferentes son los individuos entre sí. Para una mejor exploración de las soluciones se requiere una mayor diversificación, la cual se obtiene con un mayor número de individuos. La falta de diversidad conduce a una temprana convergencia devolviendo un óptimo local [47].

2. La selectividad: Es la capacidad de obtener una mejor diversidad tras la selección de porcentajes de individuos. Mientras mayor sea la muestra, existe una mayor probabilidad de obtener amplia diversidad, frente a un mismo porcentaje de un universo más pequeño.[44]

3. La complejidad del problema: mientras más complejo sea un problema, mayor población se requiere. Sin embargo, utilizar un valor que exceda al óptimo, dada la complejidad del problema, entregará una solución que no agrega nueva información, además incurriría en un gasto extra de recursos computacionales [48].

4. Espacio matemático**:** mientras mayor sea el espacio matemático mayor será el número de individuos requeridos para probar las posibles combinaciones. Un número reducido de individuos podría dejar tantas funciones y/o terminales sin ser representadas.

2.6.1.5 Profundidad del árbol

Este parámetro indica la profundidad máxima que tendrá un árbol en todo momento, es decir, la distancia máxima entre el nodo principal (nodo raíz) y sus terminales (sus hojas). Si un individuo sobrepasa esta profundidad entonces no es considerado al momento de escoger a los mejores de la generación. Al igual que los parámetros mostrados anteriormente, el efecto de este parámetro afecta directamente el tiempo de ejecución del algoritmo y, más aún, la complejidad del árbol.

2.6.1.6 Condición de término

La condición de término indica los requisitos que se deben cumplir para buscar el mejor individuo y finalizar el algoritmo. Las condiciones más utilizadas son aquellas que están relacionadas con el número de generaciones y convergencia de las ecuaciones:

1. El número máximo de generaciones no puede ser superado, por lo que una vez que se alcanza, sin importar la calidad del mejor individuo, este se entregará como la mejor solución.
2. A medida que avanza el algoritmo, se comienzan a repetir los mejores genes, por lo que se obtienen modelos similares con un buen *fitness*, es decir, se empieza a identificar una similitud entre los individuos. Se escoge un porcentaje de convergencia con el objetivo de que, una vez el número de ecuaciones similares supere este valor, se detenga y entregue el mejor individuo.

2.6.2 Generación de la población

A continuación, se detallan los procesos relacionados con la creación de los individuos desde cero.

2.6.2.1 Creación de la población

Existen distintos métodos para crear las poblaciones con la condición de que sean aleatorios y limitados, ya que el parámetro de profundidad del árbol impide superar el valor asignado. A continuación, se presentan 3 métodos:

1. *Full:* consiste en asignar una función elegida de manera aleatoria a los nodos del árbol hasta alcanzar el límite máximo para, finalmente, completar el árbol con un terminal aleatorio (Figura 7). Este método asegura la creación de modelos más complejos.

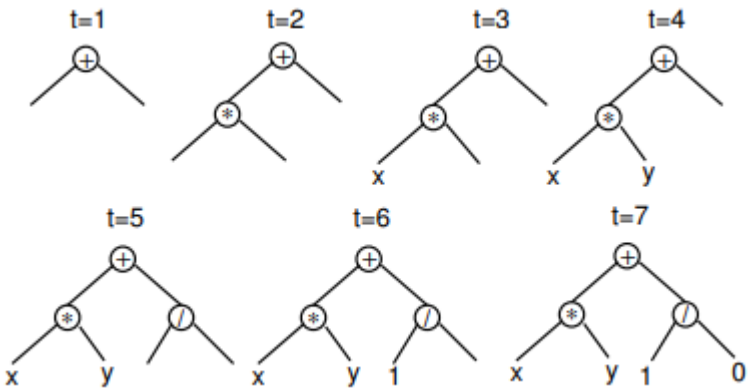

Figura 7 Creación de un arbol binario a traves del método full [42].

2. *Grow*: se diferencia del anterior en que los terminales no se escogen necesariamente cuando se alcanza la profundidad máxima, es decir, en cada nodo se selecciona aleatoriamente desde un conjunto compuesto por funciones y terminales, si los nodos están en la profundidad máxima entonces se escoge entre los terminales. Este método se ve afectado por la proporción entre terminales y funciones, si las primeras son mayores entonces se tendrán árboles de menor tamaño; mientras que lo opuesto, entregaría árboles de mayor tamaño acercándose al método full (Figura 8). Además, posee una ventaja sobre el primero, en que permite la creación de funciones con un mayor rango de complejidad, ya que puede crear la misma ecuación que el anterior, pero no a la inversa. Sin embargo, trae consigo la desventaja de que se pueden obtener árboles que tengan profundidad de 1 hasta el máximo permitido por el parámetro.

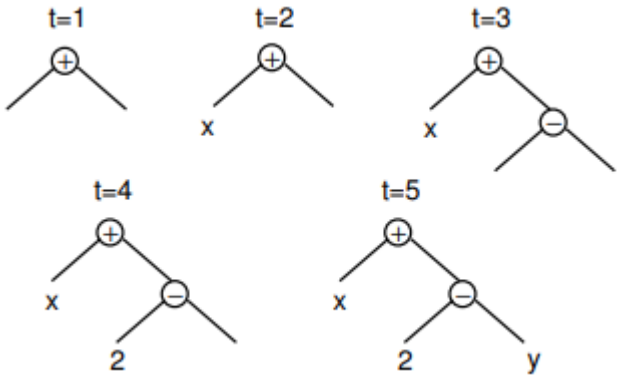

Figura 8 Creación de un arbol binario a traves del método Grow [42].

3. *Half-and-half*: los métodos anteriores tienen sus ventajas y desventajas. Por esto existe un tercer método denominado *half-and-half* con el cual la mitad de la población se crea a partir del método *full* mientras que el resto se crean con el método *grow* para así aprovechar lo mejor de ambos procesos.[49]

2.6.3 Nuevas generaciones

Para crear una nueva generación de individuos a partir de una existente, se utilizan los operadores genéticos (cruzamiento y mutación) los cuales requieren un individuo para el caso de la mutación y 2 individuos para el cruzamiento. En este apartado se presentan diferentes métodos que se utilizan para escoger los individuos, en específico, el método de la ruleta y torneo.

La ruleta consiste en ordenar los individuos según el *fitness* y posteriormente asignarles un rango entre 0 y 1 según su orden. Posteriormente, se escoge un número al azar y se devuelve el individuo cuyo rango posea el número seleccionado. De esta manera, mientras mejor fitness, mayor posibilidad tendrá de ser seleccionado (Figura 9). Existe una variación en la cual a los individuos se les asigna el mismo tamaño de espacio en la ruleta, entregando un mismo resultado que si todos los individuos tuviesen igual fitness (Figura 10).

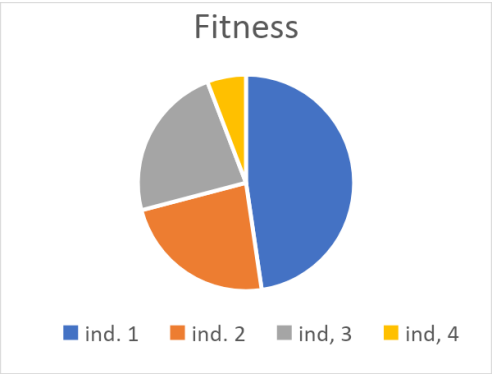

Figura 9 Ruleta construida a partir de los fitness de los individuos.

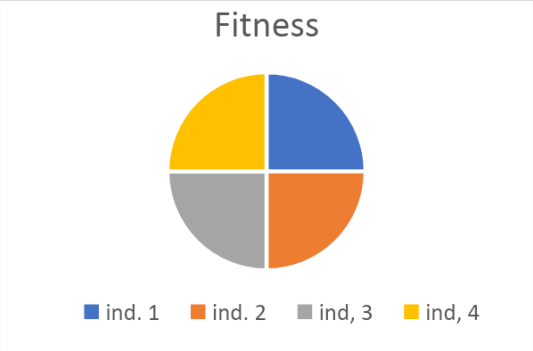

Figura 10 Ruleta donde el fitness de cada individuo es el mismo.

El torneo consiste en separar en grupos de individuos aleatorios donde se selecciona el mejor de cada grupo en base al *fitness*. El vencedor es usado para los operadores genéticos. Aun cuando el objetivo de los operadores genéticos es crear una generación superior a la anterior, existe la posibilidad que algunos sean peores, por esto es necesario copiar directamente a la nueva generación reemplazando los de menor calidad. Existen distintas maneras de escoger estos individuos, entre las que se destaca el elitismo. Este método consiste en conservar una cierta cantidad de los mejores individuos obtenidos en la generación, lo que normalmente se asocia con un porcentaje que va entre 0% y 1%. A continuación, se presenta un pseudocódigo de la programación genética expuesto en Algoritmo 1.

Algoritmo 1: Algoritmo Programación genética (Modificado de [42])

```
1: Create a initial population of random solution
2: For 1 to N_gen{
3:     Calculate each fitness
4:     If ending condition ==yes
5:     return best individual
5:     Select one or two individual
6:     Create new individual using genetic operations
7:     Select individual for the current generation
8:     Return best individual }
```

A partir del pseudocódigo expuesto es posible precisar, en primer lugar, la formación de una población inicial de individuos creada a partir de alguno de los métodos mencionados. Luego, entran en un proceso reiterativo donde se calcula el *fitness* de cada uno y se examina si se cumplen las condiciones de término. De ser positivo, se devuelve el mejor individuo y se termina el algoritmo. En caso contrario, es necesario escoger los individuos que actuarán como los padres de la nueva generación. En caso de esta decisión, se generan nuevos individuos utilizando los operadores genéticos. Finalmente se escogen los individuos que compondrán la nueva generación.

2.7 Problemas de la regresión

¿Cuál es la probabilidad de que, un mono apretando teclas aleatorias en una máquina de escribir, logre replicar cualquier texto existente?

La respuesta a esta pregunta es conocida en la cultura popular como el teorema de monos infinitos, donde la probabilidad tiende a uno mientras mayor sea el tiempo, aunque se utilice el término de "mono", este se refiere a cualquier objeto que entregue valores aleatorios[50]. En el caso de programación genética, si estuviese disponible la cantidad suficiente de individuos para probar todas las combinaciones posibles de ecuaciones, es seguro que la programación genética encontraría la solución en la generación inicial.

La mayor desventaja es el tiempo que se demora. Por ejemplo, asumiendo que se poseen cuatro terminales (dos variables y dos constantes) y cuatro funciones matemáticas simples (suma, resta, división y multiplicación) para un árbol de profundidad 1, se obtendrán cuatro posibilidades que correspondan a cada terminal, lo cual no tomaría demasiado tiempo. En otro caso, si fuera de profundidad 2 se tendrían todas las combinaciones del tipo F (A, B), donde F es una función y A, B son terminales no necesariamente distintos. En este caso es necesario probar 64 ecuaciones, demostrando que el aumento es exponencial con respecto a la profundidad del árbol, el número de terminales y funciones. Debido a lo anterior, si se agregara un terminal adicional, el número de ecuaciones aumentaría a 100, mientras que una nueva función aumentaría el número total a 80 ecuaciones.

Sumado a lo anterior, es preciso considerar que los algoritmos presentan ciertos obstáculos al ejecutarse. Esta situación tiene como consecuencia una reducción en la utilidad que tienen los algoritmos. Los contratiempos más comunes son:

2.7.1 Error de generalización

Los algoritmos de regresión simbólica requieren de datos para poder ejecutarse. Estos, en la mayoría de los casos, tienen un error asociado que puede deberse a diversos motivos. El ejemplo más común ocurre en los equipos de medición, puesto que mientras mayor sea el desgaste de este es más probable que entregue un cálculo incorrecto.

Como los programas aprenden el comportamiento de los datos, si se mantiene el mismo set de información con el tiempo llegarán a aprender el error

asociado; entregando una ecuación que puede representar al 100% los datos. Sin embargo, si se utiliza para predecir el resultado y se cambian las condiciones, el resultado se aleja de la realidad.[51]

Lo anterior se conoce como error de generalización y es cuando el resultado se obtiene al memorizar los valores de los datos. Por el contrario, se dice que el resultado posee una buena generalización si es que se crea al aprender las distintas reglas que definen el fenómeno de interés.[52]

Lo anterior refleja una contradicción del objetivo del programa, ya que se requiere que represente el comportamiento del fenómeno por sobre los datos experimentales. Por ejemplo, en la Figura 11 el grafico de la izquierda muestra un resultado generalizado deseado (línea naranja) a partir de un set de datos (circulo azul), mientras que el grafico derecho se observa un error de generalización, donde el resultado (línea roja) se obtiene a partir del mismo set de datos.

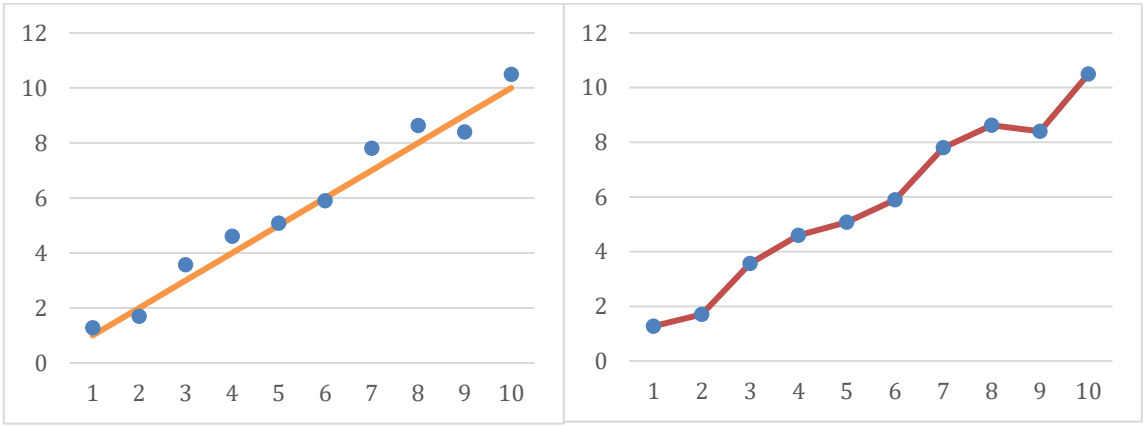

Figura 11 Ejemplo de generalización aceptable(izquierda) y una generalización equivocada(derecha).

2.7.2 Bloat

El *bloat* se refiere a un crecimiento desmedido de los árboles binarios [51]. Esto implica que términos simples son representados de manera compleja. Como se aprecia en la Figura 12 que representa la ecuación $2 * x$, la cual tiene la posibilidad de representarse en tres nodos, pero termina siendo presentada con nueve. Los nodos en rojo representados en la figura no aportan información útil, lo cual trae como consecuencia un mayor tiempo de ejecución y dificulta el análisis de la ecuación.

El *bloat* ha sido un tema ampliamente estudiado, ya que es una de las causas que disminuye la eficiencia del algoritmo que se produce en la programación genética. Lo anterior se debe a la forma de la función de *fitness* que, en la

mayoría de los casos, se basa solo en buscar un modelo que mejor represente los datos sin considerar otras características de este [53].

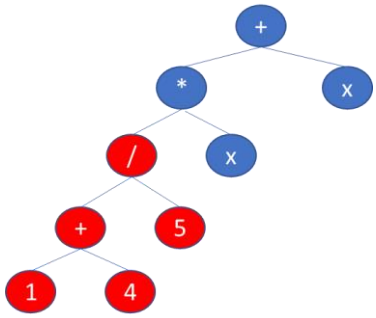

Figura 12 Ejemplo de un árbol binario que sufre Bloat.

Provocar una creación de individuos de mayor tamaño genera un número de impedimentos. En primer lugar, se traduce en un mayor tiempo de ejecución al momento de evaluarlos, siendo esta una de las razones que evitan que el PG se utilice a gran escala en sistemas en tiempo real, donde se espera que la respuesta sea en el menor tiempo posible.

En segundo lugar, es posible identificar un mayor uso de recursos donde, si no se prevé este caso de *bloat*, el programa se detendrá por quedarse sin espacio para calcular. Finalmente, una vez que se produce el *bloat,* las ecuaciones pierden calidad ya que este es un problema que solo va en aumento, lo que provoca que se deba obtener la ecuación antes del evento [54].

2.8 Herramientas

Se han desarrollado distintos métodos para combatir los problemas que presenta la regresión. En esta sección se presentará el indicado para disminuir el efecto de los problemas mencionados anteriormente.

2.8.1 Snipping

Esta herramienta consiste en simplificar los árboles binarios al sustituir los subárboles por una constante reduciendo el efecto del bloat. Al evaluar los subárboles, se puede observar si el rango en que varían es significativo o no, en caso de que sean bajos, pueden ser reemplazados por una constante.

En la Figura 13 se puede observar que el snipping reemplaza los nodos rojos por la constante 1, ya que sin importar qué datos se le entregará a la regresión, el subárbol en rojo sigue siendo 1.

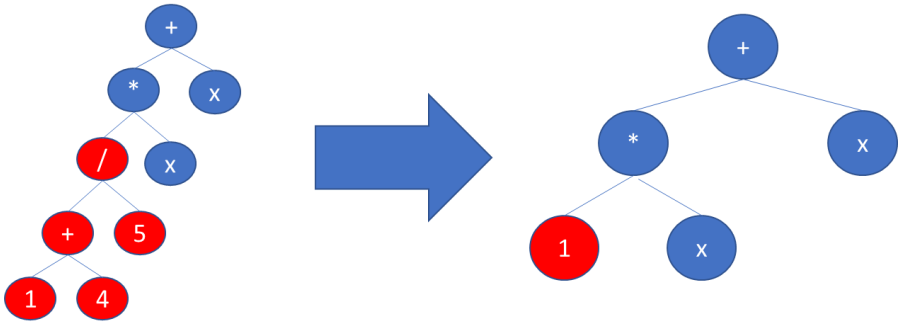

Figura 13 Efecto de Snipping sobre un árbol binario.

Capítulo 3

3 Metodología

En el presente capítulo se aborda los materiales necesarios en la realización de la memoria, donde se presentan los pasos necesarios de la metodología que propone este trabajo junto con los diferentes elementos que permiten mejorar en algún aspecto el algoritmo seleccionado. Finalmente se presenta el proceso por el que se corrobora la utilidad de la metodología en el área química.

3.1 Materiales y equipos

Para el desarrollo de la tesis se utilizarán los programas a continuación:

1. Matlab 2017b: se utiliza el lenguaje de programación de Matlab, ya que su uso se ha convertido en estándar para ingenieros y científicos.

2. Gplab 1.4: Toolbox para Matlab 2017b desarrollada por Sara Silva que permite aplicar programación genética[55].

3. Excel 2019: programa para crear gráficos para ejemplificar y demostrar información de manera sencilla.

Además de esto, se utiliza el siguiente equipo para realizar las simulaciones:

Computador perteneciente al laboratorio PMDC (*Process Modeling Distributed Computing*) del departamento de química, biotecnología y materiales que cuenta con un clúster de 120 núcleos. 80 núcleos CPU Quad Core AMD Opteron™, 16 x 10 GB RAM, espacio en disco 1TB en RAID 1, conectado mediante un switch de red de 1Gbps, 40 núcleos Intel Xeon, 2 x GBi RAM, espacio en disco 3 TB en RAID 1.[56]

3.2 Metodología propuesta

El teorema NFL, explicado en el marco teórico, exige la edición del algoritmo según las condiciones en las que se aplica, con la finalidad de mejorar el resultado obtenido. Teniendo en cuenta lo anterior la metodología propuesta se observa en la Figura 14.

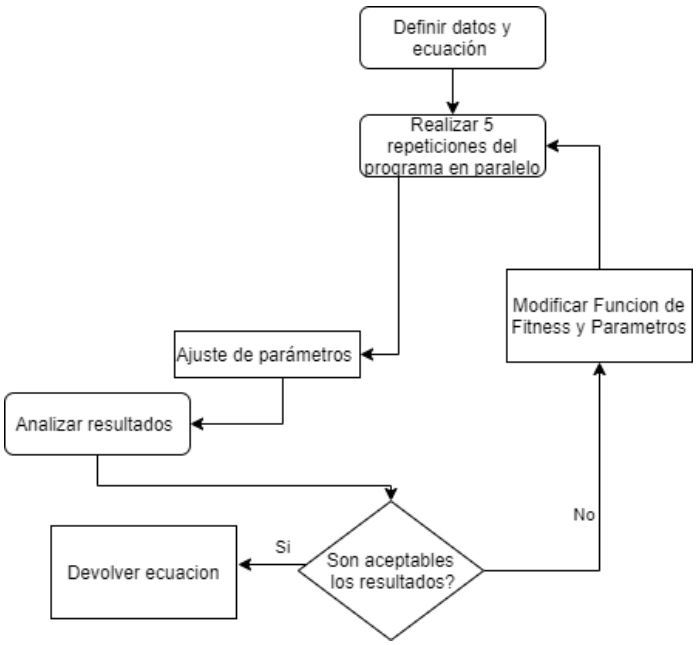

Figura 14 Metodología propuesta.

Primero, comienza con la recopilación de información. Este paso consiste en un estudio por parte del operador en el que se desea obtener el mayor conocimiento posible del fenómeno que se va a estudiar, además de realizar un análisis y recopilación de los datos, ya que los programas diseñados con esta metodología tienen el objetivo de proponer un modelo que represente los datos entregados, los cuales incluyen posibles errores en los datos. Esto se conoce como sobreestimación y, aunque el modelo represente al 100% los datos entregados, al utilizarlo en nuevas situaciones, puede que entregue una predicción errada del fenómeno. Esto se observará y discutirá con mayor detalle en los ejemplos a explorar.

El segundo paso consiste en ejecutar varias repeticiones, donde al individuo obtenido por el algoritmo de programación genética modificado se le realiza un ajuste de parámetros, para la primera iteración se utilizan los parámetros mostrados en la tabla 2, los cuales se escogieron en base a información bibliográfica, ya que dependiendo del problema en que se implemente estos pueden cambiar. A continuación, es necesario un análisis de los resultados por parte del usuario, momento en el cual es necesario observar las ecuaciones obtenidas, poniendo especial atención en que estas no solo se asemejen a los datos experimentales, sino que además entreguen información relevante y consistente con el fenómeno en estudio.

Tabla 2 Parámetros utilizados en la programación genética

Parámetros	Valor o criterio	Referencia
Número de generaciones	10	[42]
Número de individuos por generación	100	[42][46]
Funciones	+,-,*,/,exp(),log(),ln(),a^b Sin(),cos()	[42]
Terminales	0.01,5,10,1000	[44]
Límite de profundidad	17	[42]
Selección de individuos	Torneo	[55]
Creación de población inicial	Half and half	[49]
Probabilidad de mutación	0.1	[42]
Probabilidad de recombinación	0.9	[42]

Si los modelos son aceptables, se escoge el mejor de ellos. En caso contrario, se modifica la función de *fitness*, donde el programa no solo evaluará la proximidad con los datos teóricos, sino que además confirma si las ecuaciones cumplen las condiciones explicitadas por el usuario.

Al momento de llevar a cabo la metodología propuesta en este trabajo, es necesario considerar que, al realizar la implementación computacional del algoritmo, probablemente se deba editar un código fuente complejo o extenso (en este caso aplica a Gplab programado en Matlab, utilizado en la realización de este trabajo). Lo anterior puede generar complicaciones debido a que depende de otros *softwares*. Por esto, se aconseja que, al escribir las condiciones a revisar, no se modifique al individuo, sino que se resuelvan como se muestra en el pseudocódigo (Algoritmo 2).

Algoritmo 2: tipo de condiciones sugeridos

```
1. Fitness(ind){
2. For i: Cond_n
3.       If Cond_i(ind) == False
4.             F_ind = inf
5.             End Fitness
6. F_ind = Fit(ind)
7. }
```

A cada individuo (ind) de la generación se le aplica la función de *fitness* donde primero se comprueba si se cumple cada una de las condiciones impuestas (Cond_i). Si alguna de estas no se cumple, entonces, el *fitness* del individuo (F_ind) se fija en un valor, de tal manera que sin importar qué tan bien puede llegar a representar los datos, este sea descartado y se termina la función de *fitness* para ese individuo. Finalmente, si se cumplen todas las condiciones impuestas se calcula el valor de *fitness* de manera normal (Fit(ind)).

En el desarrollo de esta memoria, se diseñaron e implementaron distintas características en el algoritmo de programación genética con el objetivo de: permitir una mayor customización en las características deseadas, disminuir el efecto del *bloat* y el tiempo de ejecución, y facilitar el entendimiento del resultado. Aun cuando todas estas características fueron implementadas en Matlab, a continuación, se presentan los motivos y el proceso de pensamiento que llevaron a la creación de ellas.

3.3 Características agregadas

3.3.1 Snipping

En primer lugar, se implementa el *snipping*. En el caso de Gplab, implementarlo no es tan simple como solo editar el árbol binario, ya que los individuos se representan como estructuras, un tipo de data donde se agrupa información en contenedores, denominados campos. Para el caso de los individuos, estos poseen 24 (Figura 15), los que deben ser coherentes entre ellos. Así como se modifica el árbol (*tree*), se debe modificar también la ecuación (str), el número de nodos totales (*nodes*) y la profundidad del árbol (*level*).

```
                id: 99
            origin: 'crossover'
              tree: [1x1 struct]
               str: '0.01'
           parents: [94 92]
            xsites: [1 1]
             nodes: 1
           introns: []
             level: 1
           fitness: 105.9186
   adjustedfitness: 105.9186
            result: [91x1 double]
       testfitness: []
testadjustedfitness: []
        testresult: []
        dimensions: []
            pruned: 0
           mapping: []
  covariancematrix: {}
     inversematrix: {}
         centroids: []
    distancematrix: []
     countclusters: []
       mapclusters: []
```

Figura 15 Ejemplo de la Estructura de los individuos utilizada en Gplab.

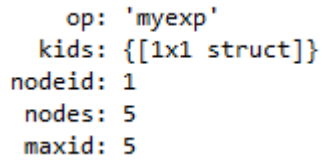

Figura 16 Ejemplo de la Estructura de los árboles binarios utilizada en Gplab.

A su vez, el árbol es una estructura que posee cinco campos (Figura 16). Al utilizar snipping es necesario modificarlo en su totalidad para evitar inconsistencia con la estructura del individuo al que pertenece. Cada nodo posee: un id propio (nodeid), el número de nodos presente en el árbol (nodes), el id más alto del árbol (maxid), y el valor del nodo raíz (op). Finalmente, están los nodos hijos que son los subárboles con conexión directa a la raíz (kids). Estos pueden ser 1 o 2, ya que, si op es una función, los kids pueden tener a su vez sus propios subárboles. A continuación, se presentan los pseudocódigos a partir de los cuales se creó la implementación computacional del *snipping* en la resolución de esta memoria.

Algoritmo 3: Tree Range

```
1. Y= evaluate_tree(tree,X)
2. Inf=Min(Y)
3. Sup=Max(Y)
4. Return [inf, Sup]
```

Algoritmo 4: snipping

```
5.  tree=Snipping(tree,X){
6.  While nodo=! Hoja
7.        [inf, sup]=tree_range(tree,X)
8.        If inf-sup <σ
9.              newtree= mean(inf,sup)
10.                   tree=newtree
11.                   Return tree
12.                   End snipping
13.             Else
14.             For i=1 to kids
15.                   tree.kid_i =Snipping(tree.kid_i,X)
16.                   tree.der = Snipping(tree.der,X)
17.       Return tree
```

Para facilitar la implementación del *snipping,* se diseña la función tree *range*, expresada en el Algoritmo 3, el cual tiene como objetivo devolver el valor máximo (sup) y mínimo (inf) obtenido al evaluar la función representada por el árbol (tree) con los valores experimentales (X) que se le entregan.

El *snipping* (Algoritmo 4), es un algoritmo recursivo. Al momento de implementarlo, primero se observa si el nodo raíz del árbol no es una hoja, lo cual se puede saber si el valor de este no es una constante o variable. Luego, se utiliza el *tree range* para observar el rango en el que se mueve la función. Si la diferencia entre sup e inf es menor a un valor definido por el usuario (σ), entonces todo el árbol se reemplaza por un nodo cuyo valor será el promedio del rango (esto se aplica debido a que, si una función no posee gran variabilidad, puede ser representada como una constante sin perder generalidad). En el caso que el rango sea mayor, entonces es necesario examinar los hijos del nodo raíz (kids), donde a cada uno de estos árboles (kid_i) se le aplica la función de *snipping*. Una vez examinados se devuelve el árbol simplificado.

3.3.2 Limitar el número de nodos

Gplab posee parámetros y funciones para controlar el crecimiento desmedido de los árboles, los que trabajan específicamente con la profundidad. Debido que no todas las funciones necesitan dos argumentos, puede existir el caso de un árbol que crezca solo por una rama (Figura 17) el cual sea más cercano al resultado deseado que un árbol de menor profundidad que crezca de manera uniforme (Figura 18). Independiente del resultado, si el primero de estos árboles supera el límite de profundidad, será rechazado. Por esto, se crea un nuevo parámetro, el cual se denomina maxnodo y toma valores numéricos enteros positivos, lo que hace que los individuos que posean un número de nodos que superen este valor sean descartados.

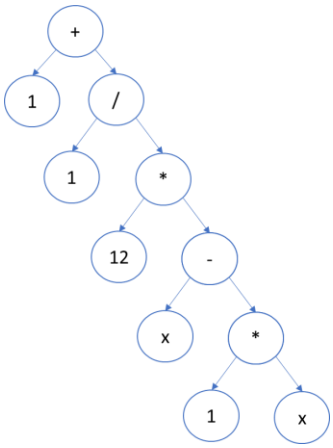

Figura 17 Caso extremo de un árbol desbalanceado.

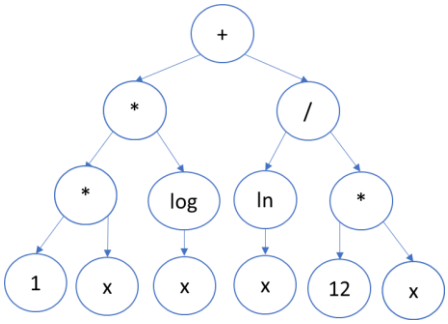

Figura 18 Árbol balanceado.

3.3.3 Ajuste de parámetros

Si el tiempo y poder de cómputo no son una limitante, la implementación computacional creará subárboles para encontrar la constante que mejor se adapte a los datos. Este comportamiento es el objetivo que se tiene al realizar un ajuste de parámetros, por lo que, para evitar que el programa gaste generaciones en esto, se implementa un ajuste dentro de la función de *fitness*. La función fue extraída de un trabajo similar donde se utiliza la programación genética para proponer un modelo de absorción[57].

3.3.4 Integrar ecuación

Existen ecuaciones complejas que no pueden ser expresadas de manera integrada, como la ecuación de Lotka-Volterra que es una de las ecuaciones que se utilizan para probar la metodología propuesta, por lo que, en vez de evaluar la ecuación de manera normal, se utiliza un integrador Runge-Kutta tipo 2 3. La razón para escoger esta opción por sobre el integrador Runge-Kutta 4 5, el cual es más robusto, se debe a que las ecuaciones son creadas de manera aleatoria, por lo que es probable que más de un individuo posea una función *stiff*, la que retrasa el funcionamiento del programa al aumentar de manera desmedida el tiempo necesario para integrarlo.

Algoritmo 5: *Fitness* con integración

```
1. Fitness(tree,X){
2. F=treetofuncion(tree)
3. Time=tic
4. While Time<t
5.         Y=ode(F,X)
6.         Return Y
7. Y=zeros(X)
8. Return Y
9. }
```

El Algoritmo 5 muestra un pseudocódigo donde primero se transforma el árbol binario en un tipo de dato útil que pueda ser fácilmente manipulado (en el caso de Matlab se transforma en una *function_handle)*. Luego, se inicia un temporizador (Time) y, mientras no supere un tiempo (t) especificado por el usuario (10 segundos en el trabajo realizado), se integra la función (F) construida a partir del árbol del individuo, devolviendo así el resultado. Si la función no alcanza a terminar en el tiempo especificado, entonces se devuelve una matriz del mismo tamaño que los datos con valores igual a 0 u otro valor que sea relevante para el usuario. El objetivo de esta última acción es que el individuo no sea considerado para crear la siguiente población.

3.3.5 Bloques conocidos

Al crear o proponer un modelo, la información recolectada puede ser utilizada por la implementación computacional del algoritmo para formar partes del mismo modelo. Una primera alternativa de implementar esta idea de manera no invasiva es entregar la información en forma de función para que el algoritmo lo seleccione en algún nodo. Este método no es confiable, ya que la creación de árboles es aleatoria, por lo que puede ocurrir que una pequeña parte de la población o incluso ninguna utilice la función. Debido a este inconveniente, se propone un segundo método que hace uso de la función de *fitness*.

Algoritmo 6: Agregar ecuación conocida

```
1. Fitness(tree,X){
2. Auxtree= totree(ec)
3. Newtree=Unir(Auxtree,tree,func)
4. F=treetofuncion(Newtree)
5. Time=tic
6. ….
```

En el Algoritmo 6 se encuentra el pseudocódigo del método utilizado. Primero transforma la información de interés (ec) que se quiere agregar en un tipo de dato que sea compatible (en el caso de Matlab se transforma en *string).* Luego, se une con la ecuación del individuo que se va a evaluar (Unir), especificando con qué función (func) se relacionan (suma, resta, multiplicación, división o cualquier función relevante). El resto del proceso es el mismo que en el mostrado en el Algoritmo 5.

3.3.6 Criterio de información de Akaike (AIC)

La función de *Fitness* que trae Gplab se puede simplificar en dos puntos. El primero de ellos es evaluar la ecuación, mientras que el segundo es calcular el error absoluto entre los valores obtenidos por el modelo y los resultados experimentales.

Para el presente estudio, la función de *fitness* se modifica para utilizar el Criterio de información de Akaike, el cual se utiliza como un valor de comparación de modelos estadísticos[58], donde el de menor valor será el mejor modelo propuesto.

El AIC trabaja con un *trade-off* entre la bondad del ajuste y la complejidad del modelo, que es medida en el número de parámetros que este utiliza. Esto se puede observar en la ecuación (16) donde *k* es el número de parámetros y L es el máximo valor de verosimilitud del modelo estimado. En el caso de que el número de datos sea reducido, es necesario agregar una corrección, obteniendo el Criterio de Información de Akaike corregido (AICc) que se muestra en la ecuación (17)[59] donde *n* es el número de datos usados.

$$AIC = 2 * k - 2Ln(L) \tag{16}$$

$$AICc = AIC + \frac{2 * k^2 + 2k}{n - k - 1} \tag{17}$$

3.4 Modelos

Los modelos que se utilizan para comprobar la metodología corresponden a los expuestos en el apartado de sistemas químicos en el Marco teórico de este trabajo. En el siguiente apartado se exponen los parámetros que se utilizan en este estudio.

3.4.1 Creación de datos

Los datos utilizados en el estudio se obtienen a partir de simular los modelos de altura del líquido y Lotka-Volterra. Este procedimiento se lleva a cabo al integrar dichos modelos en matlab utilizando la función ode45 obteniendo 100 valores, los parámetros y valores iniciales utilizados se observan en las Tabla 3 y Tabla 4. Para simular un caso más cercano a la realidad se le agrega un error aleatorio del 5% a cada dato, con lo anterior se obtienen 2 set de datos para cada modelo preestablecido para este trabajo.

Tabla 3 Parámetros para el modelo de altura de líquido dentro de un estanque

Parámetro	Valor
F_{ent}	2
d	0.8
A	10
h(t=0)	0
t	100 s

Tabla 4 Parámetros escogidos para el modelo de Lotka-Volterra

Parámetro	Valor
α	0.1
β	0.02
σ	0.04
γ	0.4
X(t=0)	10
Y(t=0)	15
t	100 días

Para el caso del modelo de reacción de oxidación de monóxido de carbono, primero es necesario simplificarlo, lo que se logra al unir las constantes en una sola cuando sea posible. Esto se realiza ya que el algoritmo no identifica los parámetros que componen las constantes del modelo, sino que busca la forma en que interactúan las constantes y las variables del fenómeno.

A partir de la simplificación anterior, el modelo queda representado por las ecuaciones (18) a la (20), donde k_1, k_5 es la constante relacionada con la adsorción del CO y O en el catalizador correspondientemente. k_2 y k_3 están relacionada con la desorción del CO, k_6 es la desorción del oxígeno, k_4 se relaciona con la velocidad de reacción, k_7 es la constante para transformar número de moles en presión y k_8 es el porcentaje extraído de CO_2 desde el reactor. Los parámetros utilizados en este modelo están expresados en la

Tabla 5.

$$\frac{d\theta_{CO}}{dt} = k_1 * S_{CO}(\theta_{Co}, \theta_O, T) - k_2 * \theta_{CO} * e^{k_3 * \theta_{CO}} - k_4 * \theta_{CO} * \theta_O \quad (18)$$

$$\frac{d\theta_O}{dt} = k_5 * (1 - \theta_{Co} - \theta_O)^2 - k_6 * \theta_O - k_4 * \theta_{CO} * \theta_O \quad (19)$$

$$\frac{dp_{CO_2}}{dt} = -k_8 * p_{CO_2} + k_7 * k_4 * \theta_{CO} * \theta_O \quad (20)$$

Tabla 5 Parámetros utilizados al simular la oxidación de CO

Parámetros	Valor
k_1	0.04
k_2	0.015
k_3	3.97
k_4	0.03
k_5	0.012
k_6	0
k_7	0.5
k_8	0

3.5 Comprobación de la metodología

Se comprueba la eficiencia de la metodología al intentar recuperar los modelos preseleccionados. El procedimiento se realiza a partir de datos con y sin error, los cuales son creados a través de condiciones fijas, y a los que se les aplica la metodología propuesta. Durante esta acción, los modelos originales y los parámetros se consideran desconocidos. Finalmente, las características diseñadas para el presente trabajo son analizadas tomando en consideración su efecto durante la ejecución y resultado del algoritmo.

La primera iteración en esta corroboración de la metodología utiliza los parámetros mostrados en la Tabla 6.

Tabla 6 Parámetros utilizados en la programación genética

Parámetros	Valor o criterio	Referencia
Número de generaciones	10	[42]
Número de individuos por generación	100	[42][46]
Funciones	+,-,*,/,exp(),log(),ln(),a^b Sin(),cos()	[42]
Terminales	0.01,5,10,1000	[44]
Límite de profundidad	17	[42]
Selección de individuos	Torneo	[55]
Creación de población inicial	Half and half	[49]
Probabilidad de mutación	0.5	[42]
Probabilidad de recombinación	0.5	[42]

Como los datos utilizados se obtuvieron a partir de simulaciones de modelos conocidos, el resultado obtenido a través de la metodología se evalúa al compararlos con el original, lo cual se realiza de 2 enfoques. Por un lado se cuantifica la similitud al simular el modelo obtenido con los datos originales a

través del coeficiente de correlación de Pearson el cual se calcula como se muestra en la ecuación (21) donde se espera que un modelo adecuado entregue valores cercano a 1, dentro de las razones para utilizarlo se encuentra que es independiente del orden de magnitud de los valores.

$$r = \frac{\sum Z_x * Z_y}{N - 1} \tag{21}$$

$$z_x = \frac{x - \bar{x}}{s_x} \tag{22}$$

Donde N es el número de datos, mientras que $Z_x \, y \, Z_y$ representan las variables normalizadas a través de la ecuación (22), con x el valor de la variable, $\bar{x}$ el valor promedio de la variable y s_x la desviación estándar de esta, para la variable y se ¡utiliza la misma fórmula.

En segundo se realiza una comparación de las expresiones matemática haciendo énfasis en el tipo de funciones y parámetros obtenidos junto con un análisis visual comparando las trayectorias obtenida a simular el modelo obtenido con respecto al original. Por último, dependiendo del fenómeno de estudio se pueden proponer y realizar otras comparaciones.

Capítulo 4

4 Resultados y Discusión

En este apartado se presentan los resultados obtenidos durante la realización del trabajo de título junto con sus discusiones. Este capítulo se divide en cuatro partes: las primeras tres corresponden al análisis de los modelos predefinidos donde la única característica adicional a utilizar en todo momento es la de simular los modelos, ya que los modelos están compuestos de ecuaciones diferenciales. El último apartado se hace cargo del cierre del capítulo con una síntesis de los resultados más importantes.

La literatura especializada ha enfatizado en que la programación genética es capaz de devolver ecuaciones de distintas complejidades en distintas áreas [60]. A partir de lo anterior, es relevante estudiar el comportamiento del algoritmo con modelos químicos de diferente complejidad para evaluar la metodología propuesta.

Cada modelo será presentado con una pequeña introducción, donde se exhiben los datos utilizados junto con los requisitos necesarios, si es que existen, para utilizar la metodología y/o el algoritmo. Seguido de esto, se validan los resultados al utilizar un set de parámetros predefinidos donde se considera que la información que se posee del fenómeno es mínima.

El primer modelo, al ser el más simple de los tres, aparte de ilustrar el funcionamiento de la metodología y el modelo que se obtiene al utilizarla, también es usada para comprobar el efecto de las diferentes modificaciones propuestas para el algoritmo de la programación genética implementado en la suite Gplab, las que corresponden al *snipping*, limitar el número de nodos de los árboles binarios, realizar un ajuste de parámetro a cada individuo de cada generación, que el algoritmo agregue expresiones conocidas previamente del modelo a los individuos y el uso del Criterio de información de Akaike como medida del *fitness*.

Los resultados entregados corresponden a los obtenidos al realizar cinco repeticiones en paralelo en cada caso, donde dependiendo de los valores que se entreguen y si es relevante se mostrarán los 5 o un resumen de estos.

4.1 Modelo 1: Altura de un líquido almacenado en un tanque

A continuación, se muestran los resultados obtenidos al simular el perfil de la altura de líquido en un estanque de nivel inicialmente vacío durante 100 segundos. El estanque de nivel está descrito por el siguiente modelo.

$$\frac{dh}{dt} = 0.2 - 0.08 * h \tag{23}$$

El modelo anterior expresa el cambio de altura del líquido dentro del estanque a través del tiempo, el cual está sujeto a una diferencia entre los flujos de entrada y salida. El primer término de la ecuación representa el aumento de h por efecto del flujo de entrada sobre el área transversal del estanque obteniéndose 0.2 [m/s]. Por su parte, el segundo se relaciona al flujo de salida, donde el coeficiente 0.08 [1/s] representa el cociente entre el coeficiente de pérdida, con unidades de [m^2/s], y el área del equipo.

La Figura 19 muestra la trayectoria teórica del modelo junto a una señal a la cual se le ha agregado un error aditivo tal como se expresa en la ecuación (24), donde a cada valor de los datos (x_i) se le agrega un error aleatorio que lo puede aumentar o disminuir hasta un 5%, donde se obtiene un nuevo valor $x_{i_{error}}$. El modelo fue simulado utilizando los parámetros en la Tabla 3 mediante una implementación del método de Runge Kutta orden 45 a través del software de Matlab.

$$x_{i_{error}} = x_i * random(0.95, 1.05\ ) \tag{24}$$

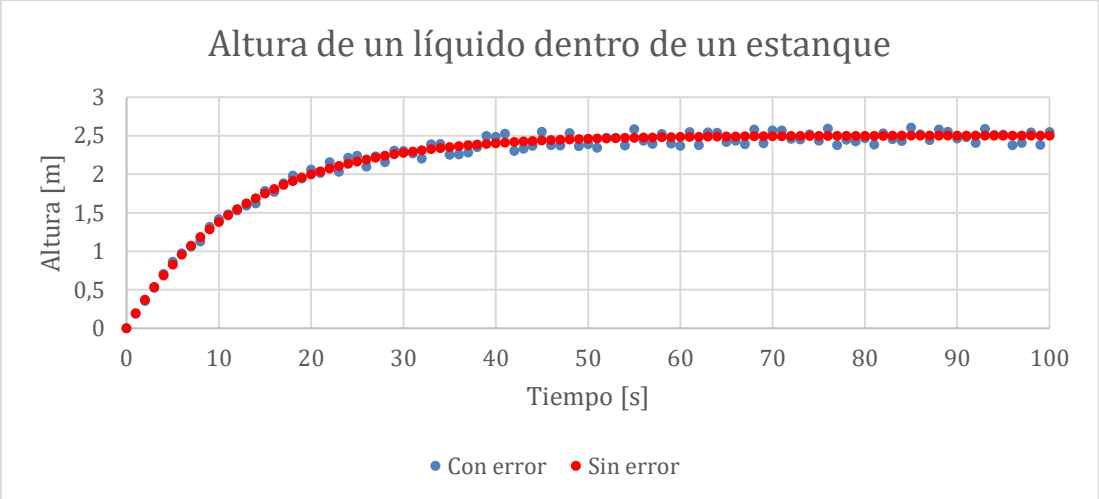

Figura 19 Datos obtenidos al simular el modelo de altura de liquido dentro de un estanque.

En la figura se observa que la trayectoria de la altura en función del tiempo se puede dividir en dos zonas. La primera comprende desde el inicio hasta los 60 segundos, donde se observa un incremento de tipo exponencial que converge a un valor de 2.5 metros. La segunda parte de los 60 segundos, donde el modelo alcanza un estado estacionario con la altura del líquido manteniéndose a los 2.5[m].

Al realizar un análisis del modelo matemático se puede extraer diferente información. Primero, que la altura máxima que alcanza el líquido dentro del estanque es 2.5[m]. Esto se debe a que a esa altura la derivada se vuelve nula, ya que la diferencia entre el flujo de entrada (el cual es constante) y el de salida (que cambia dependiendo de la altura) es 0. A partir de este momento el modelo alcanza y se mantiene en un estado estacionario.

La metodología propuesta en este trabajo depende de 2 factores:

1) El algoritmo: En este caso la programación genética es la que propone modelos a partir de los datos.
2) El usuario: Es quien debe realizar un análisis de los modelos propuestos para decidir cuál le resulta más útil en su trabajo.

Para poder crear un modelo con mejores resultados es necesario que ambos factores trabajen correctamente. En particular, el funcionamiento del algoritmo depende en gran medida de los parámetros con que se trabajan, por lo que se estudia el efecto de distintos parámetros y las características diseñadas en la metodología.

Es fundamental, como punto de partida, realizar un estudio sobre el comportamiento del algoritmo que se utiliza en la realización de este trabajo de memoria con un set de parámetros fijos, el cual corresponde a la programación genética (PG). Posteriormente, se vuelve a ejecutar el algoritmo cambiando uno de los parámetros para evaluar el efecto que genera. Los parámetros involucrados en esta acción reiterativa son:

1) Individuos y generación
2) Programación en paralelo
3) Presencia de error en los datos
4) Número de datos

4.1.1 Resultados Iteración 1

En primer lugar, se considera que el modelo es desconocido, por lo que se usa la metodología propuesta para diseñar un modelo que represente los datos expuestos en la Figura 19. El primer paso corresponde a la recopilación de datos e información, seguido de la utilización del algoritmo. Este proceso se realiza cinco veces para un primer acercamiento se utiliza un set de parámetros predeterminados, los cuales se muestran en la siguiente tabla.

Tabla 5 Parámetros utilizado en PG

Parámetros	Valor o criterio
Número de generaciones	10
Número de individuos por generación	40
Funciones	+,-,*,/,exp(),log(),ln(), sin(), cos(),a^b
Terminales	0.01,5,10,1000
Límite de profundidad	17
Selección de individuos	Torneo
Creación de población inicial	Half and half
Probabilidad de mutación	0.5
Probabilidad de recombinación	0.5

A continuación, se resumen los resultados obtenidos al realizar cinco repeticiones del algoritmo al utilizar los parámetros mostrados anteriormente.

Se observa que el algoritmo tiende a proponer modelos que siguen uno de los tres comportamientos más frecuentes. Los mejores modelos, para cada uno de estos comportamientos, se muestran en la Tabla 6, donde la primera columna expresa el modelo propuesto por el algoritmo, seguido del *fitness* que se calcula como el error cuadrático medio. Finalmente, el tiempo que se demoró en ejecutarse:

Tabla 7 Mejores modelos obtenidos de cinco iteraciones de PG para la Altura del líquido

modelo	***Fitness***	**Tiempo [s]**
$\boldsymbol{dh/dt} = 0.14^h$	0,1837	239
$\boldsymbol{dh/dt} = \frac{0.01}{h}$	0,1396	304
$dh/dt = 0.0353$	0,43	215

Al simular los modelos descritos en la Tabla 6 se obtienen las trayectorias para la altura en función del tiempo mostrado en la Figura 20.

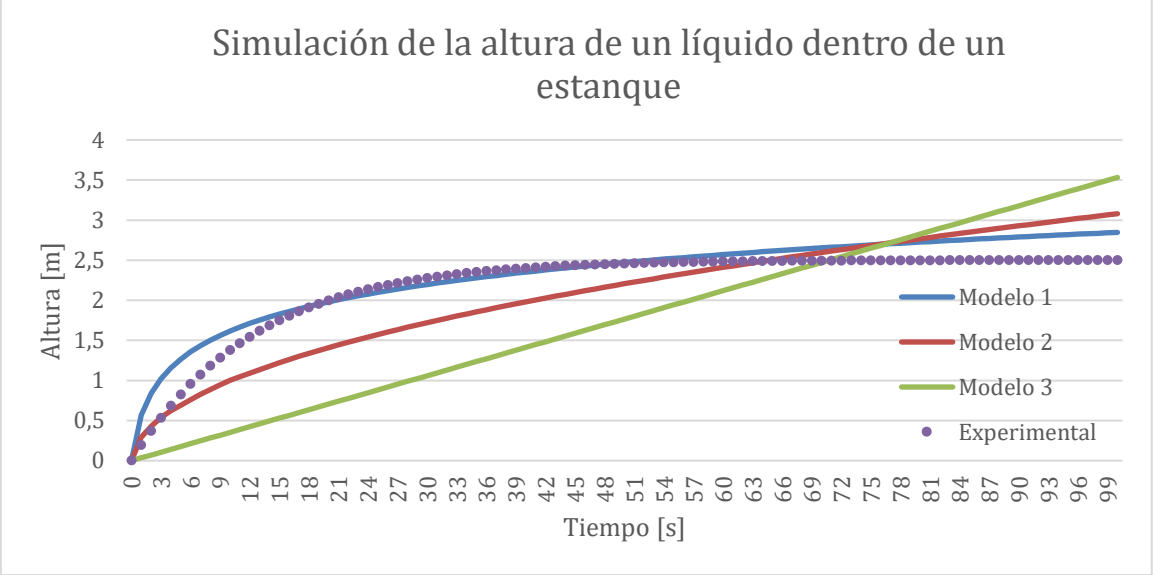

Figura 20 Modelos obtenidos por PG en la primera iteración de la metodología.

El primer comportamiento asociado a la trayectoria del Modelo 1, corresponde a una función exponencial que sigue un comportamiento similar a la primera zona de la simulación original, pero a diferencia de los datos que se mantienen constantes, una vez llegado a 2.5[m], el modelo seguirá aumentando indefinidamente a través del tiempo.

El Modelo 2, se caracteriza por contener un cociente donde la variable de estado se encuentra en el denominador. Si este modelo se integra analíticamente, se obtiene el modelo integrado representado en la ecuación siguiente, el cual se comporta de manera similar a los datos, es decir, se presenta un aumento rápido de la altura en un inicio, pero que luego decae en el tiempo.

$$h(t) = sqrt(t * 2 * 0.01) \quad (25)$$

Si bien el comportamiento observado gráficamente podría sugerir que la forma del modelo es la deseada, matemáticamente es incorrecto, ya que la simulación corresponde al llenado del estanque. En otras palabras, para t=0 la altura del estanque es cero provocando que en este punto el modelo se indetermine. Esta inconsistencia se produce dado que el algoritmo base usa funciones protegidas, las que se establecen para evitar errores como la división por cero, tal como ocurre en este caso.

El modelo anteriormente mencionado, en primera instancia, parece no ser de utilidad para el fenómeno de estudio, porque este puede presentar datos que provocan que el modelo se indetermine. Sin embargo, esto no es razón suficiente para que, en posteriores iteraciones del algoritmo, no se utilice la función división al crear los árboles binarios. Esto último se debe a que la presencia de la variable en el denominador puede representar un modelo cercano al deseado, podría darse el caso por ejemplo que el diseño esperado sea:

$$\frac{dh}{dt} = \frac{10}{h + 0.2} \tag{26}$$

Para obtener un mejor modelo, en este caso es mejor agregar parámetros a la ecuación. De esta manera, el denominador pasa de ser solo la variable a una suma entre una constante y una variable. Esto se puede observar en la siguiente ecuación:

$$\frac{dh}{dt} = \frac{10}{a + b * h} \tag{27}$$

El tercer comportamiento corresponde al Modelo 3, representado por una constante. El modelo integrado, en este caso, corresponde a un modelo lineal con intercepto igual a cero y con pendiente igual al valor de la constante propuesta por el algoritmo. A diferencia de los datos experimentales que se mantienen constantes a partir de cierto punto, este modelo aumenta infinitamente.

El modelo que se desea recuperar corresponde a una ecuación lineal, pero el programa tiende a ignorar este tipo de ecuación en favor de las presentadas anteriormente. Esto puede deberse a que los modelos obtenidos pueden ser representados por árboles con un menor número de nodos. En la Figura 21 se observan dos árboles binarios que representan distinto tipo de funciones: una lineal (izquierda) y una exponencial (derecha). Para ellas, se requiere un mínimo de 7 y 5 nodos respectivamente. Para el caso de la división y constante (modelo 2 y 3 revisado anteriormente) se requiere un número menor de nodos; 3 y 1 respectivamente. Esta diferencia se traduce en que funciones que requieren una menor cantidad de nodos, tienen mayor posibilidad de crearse, ya que un árbol de 7 nodos puede contener todas las funciones mencionadas, mientras que un árbol de un solo nodo puede crear la función constante. Esto implica que las funciones con menos nodos podrían gastar más poblaciones de individuos en el ajuste de los parámetros.

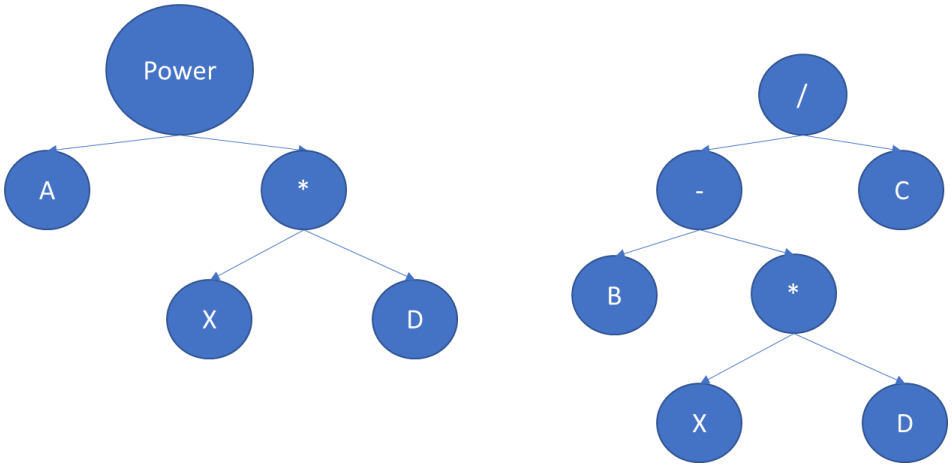

Figura 21 Representación de 2 funciones como arboles binarios

Sumado a lo anterior, se observa que una vez encontrado un modelo que tenga un comportamiento similar a los datos, los subárboles que lo representan se empiezan a expandir a través de las nuevas generaciones, provocando que un mayor número de individuo tenga una forma similar, diferenciándose en los valores de los parámetros que poseen. Esto es similar al objetivo de un ajuste.

Los modelos encontrados hasta el momento no sirven para representar los datos experimentales por lo que es necesario una nueva iteración la cual, según la metodología propuesta, se debe entregar información al algoritmo y cambiar los parámetros para obtener una mejor solución. Los cambios necesarios son, en primer lugar, limitar el uso de funciones exponenciales o simplemente eliminarlas según lo comprobado; en segundo lugar, las funciones trigonométricas pueden ser omitidas ya que, al observar los resultados, estas no fueron utilizadas para representar el comportamiento. Como tercer cambio, es necesario exigir la presencia de la variable de altura en los árboles, lo que evitaría la creación de una constante como resultado.

Un último cambio se asocia con agregar una constante negativa al conjunto de terminales. Esto tiene como finalidad que el programa no gaste tiempo en encontrar un parámetro negativo y se enfoque en trabajar la forma de la función. Los cuatro cambios detallados son aplicados desde un inicio en los siguientes experimentos.

Dado que los datos se obtuvieron a partir de simulaciones, se posee una cantidad mayor a la que se logra obtener al realizar trabajos similares en la práctica. Lo anterior se traduce en una facilidad para que el algoritmo proponga soluciones de mayor calidad, ya que una mayor cantidad de datos asegura que se consideren todos los estados del fenómeno.

4.1.2 Efecto de programación en paralelo

Una de las grandes ventajas de la programación genética (PG) es que la implementación computacional puede ser realizada en paralelo, es decir, se realizan simultáneamente varios cálculos, lo cual permite obtener una mayor cantidad de resultados. Por ejemplo, se puede aplicar en el cálculo del *fitness* de cada individuo, o ejecutar diversas iteraciones de la PG de manera concurrente. En la realización de la memoria se estudió el efecto que provoca utilizar esta última forma, donde se ejecutan cinco iteraciones en paralelo a través de la función "*parfor*" del software Matlab donde cada una consta de 100 individuos con 10 generaciones.

Durante las primeras pruebas, tres de los cinco modelos propuestos resultaron ser idénticos, tanto en los nodos que utilizan como en la posición de estos. Dada la naturaleza estocástica de la PG, la probabilidad de que ocurra lo anterior es baja, por lo que se examina la población inicial de cada set donde se obtienen los resultados mostrados en la Tabla 8 en que se presenta el porcentaje de similitud entre las diferentes poblaciones.

Tabla 8 Porcentaje de Similitud Entre los Individuos de Distintos Set

	1	**2**	**3**	**4**	**5**
1		21	21	11	21
2	21		100	12	100
3	21	100		12	100
4	11	12	12		25
5	21	100	100	25	

Se observa que los sets de población 2, 3 y 5 son idénticos entre ellos. Es importante recalcar que a partir de estos se originaron los modelos mencionados con anterioridad. Dado que el número de funciones y terminales entregado al algoritmo es finito, se espera que exista cierta similitud. Por ejemplo, si se le pide generar 7 individuos a partir de 5 terminales se obtendrá que por lo menos dos de ellos serán copias de alguno de los otros. Sin embargo, que se obtenga un 100% indica que los pasos aleatorios de PG están

entregando los mismos valores al utilizarla en paralelo, esto se puede deber a que en la computación una aleatoriedad real no es posible ya que el valor se crea a partir de alguna variable que depende del software utilizado, por ejemplo, utilizar el reloj interno del equipo o el número de fotogramas. En el caso de la realización de la memoria, las funciones de aleatoriedad de Matlab están determinadas por el reloj interno del equipo.

4.1.3 Efecto Individuos y generación

Para una buena ejecución del algoritmo es necesario escoger los parámetros adecuados, esta sección se centra en los parámetros de número de individuos y generaciones. Existen distintos métodos para calcular estos valores, uno de ellos consiste en basarse en experimentos similares [43] o ajustarlo a través de prueba y error.

Este trabajo de memoria se enfoca en este último, donde se estudia el efecto de aumentar o disminuir el valor de individuos: inicialmente se usan 100 individuos para luego repetir el proceso con 10, 500 y 1000. En cada uno de estos experimentos se utiliza un número de generación igual a 20 y se realizan cinco repeticiones. A medida que se ejecuta el algoritmo, se guarda la información de cada generación para analizar cómo mejora a través del tiempo. Cada proceso es cronometrado, ya que un aumento de individuos conlleva a un aumento del tiempo de ejecución.

Los resultados obtenidos se resumen desde la Figura 22 a la Figura 24 donde el primero muestra el tiempo de ejecución promedio y su desviación estándar, para cada experimento de individuos. Las siguientes 2 figuras muestran el cambio del *fitness* del mejor individuo a través de las generaciones. Los valores son el promedio y la desviación estándar obtenidos de las 5 repeticiones.

Tabla 9 Tiempo promedio de ejecucción del algoritmo para distinto numero de individios

Numero de individuos	Tiempo
10	103
100	269
500	1860
1000	4782

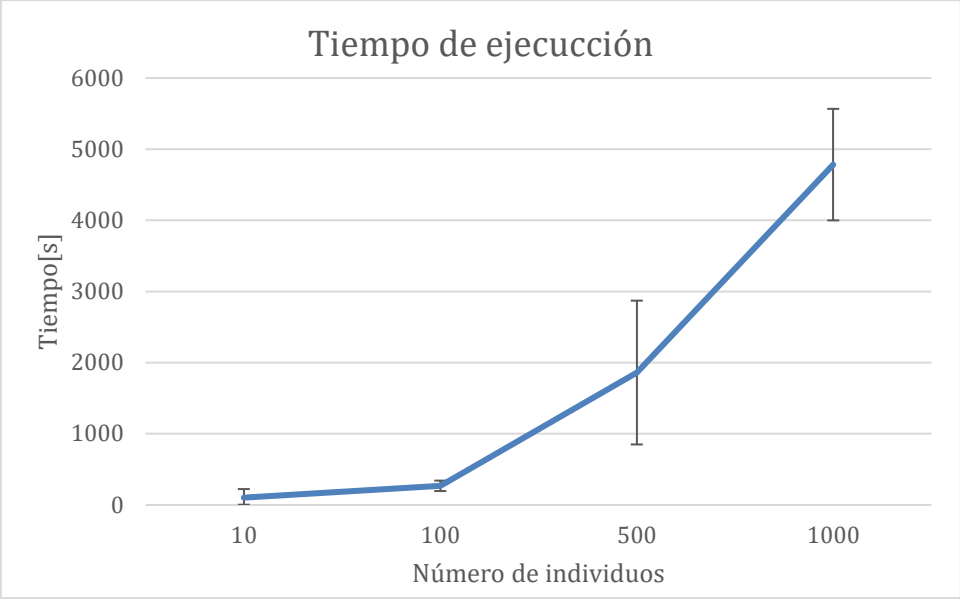

Figura 22 Tiempo de ejecuccion promedio del algoritmo para distinto numero de individuos.

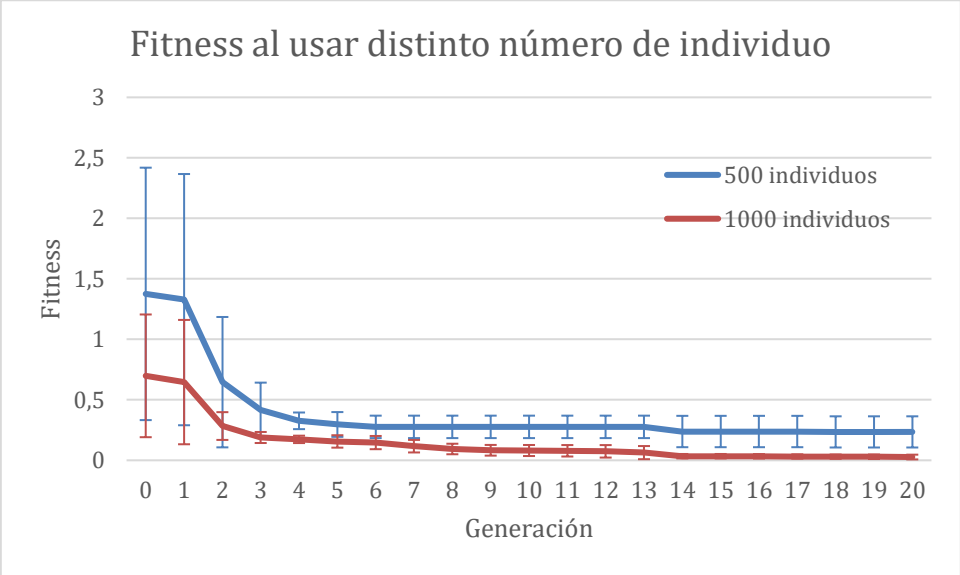

Figura 23 Evolución del fitness en 20 generaciones para 500 y 1000 individuos.

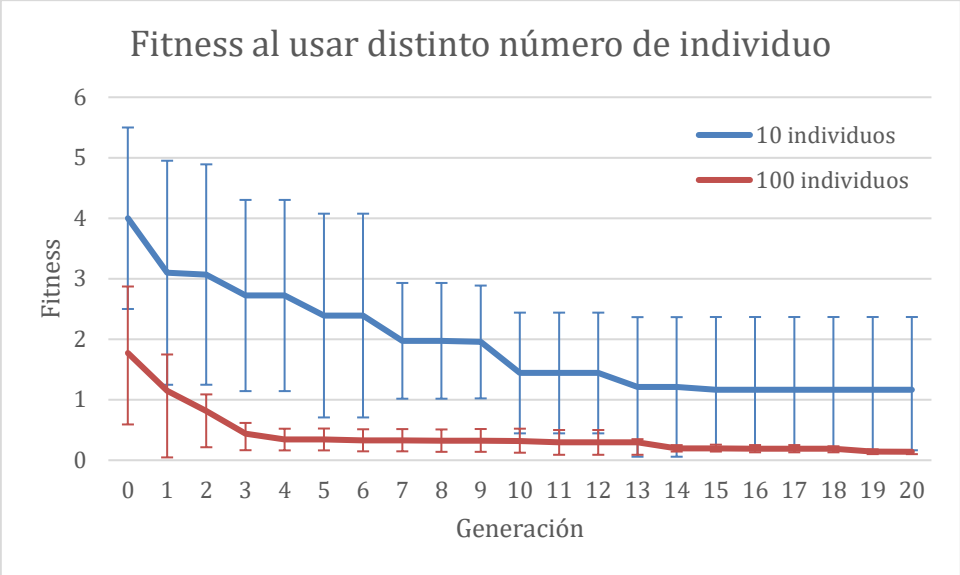

Figura 24 Evolución del fitness en 20 generaciones para 10 y 100 individuos.

En las Figura 23 y Figura 24 se observa que un mayor valor en cualquiera de los parámetros mencionados entrega mejores resultados, pero requiere más tiempo. La variación de 10 a 100 individuos entrega un mejor *fitness* con un incremento de tiempo menor que el caso de variar de 500 a 1000. Sumado a esto, se puede observar que la última mejora de *fitness* en la mayoría de los casos se obtiene entre las generaciones 10 y 14. Con estos resultados, se observa que utilizar 100 individuos con 10 generaciones entrega un valor satisfactorio en un tiempo razonable para el caso de estudio, por lo que desde aquí en adelante serán los parámetros utilizados.

4.1.4 Efecto del número de datos

El siguiente experimento consiste en estudiar el efecto del número de datos que se le entregan a la programación genética, como se expone al inicio de la sección, en los que se pueden apreciar dos zonas, por lo que se realizan dos experimentos donde el primero utiliza una parte de los datos originales que van desde el tiempo 0 hasta los 17 segundos, el resto de los datos se utilizan en el segundo.

La simulación de los modelos obtenidos de estos experimentos se observa en la Figura 25 y Figura 26, donde para el primero se obtuvo un modelo que expresa que el cambio de altura es constante a través del tiempo obteniéndose la línea sólida. Por su parte, el segundo caso es similar al primero en que independiente del momento, el cambio de altura fue nulo. Para ambos casos, si solo se observa el *fitness* obtenido y las figuras, se puede concluir que los modelos son suficientes para representar los datos, cuando en realidad los

modelos se obtuvieron solo porque los datos no fueron suficientes para representar el fenómeno en su totalidad.

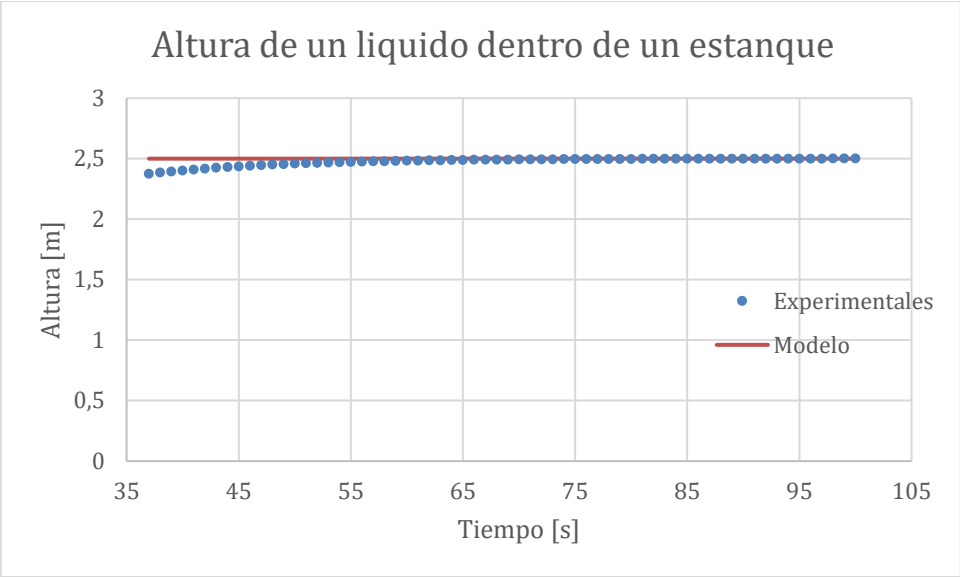

Figura 25 Modelo obtenido si los datos están concentrados en el estado estacionario.

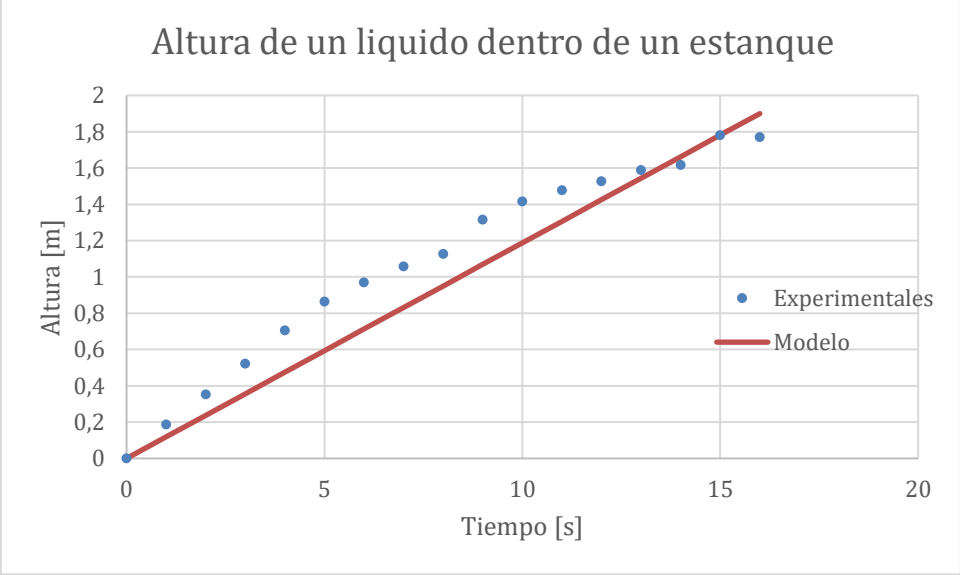

Figura 26 Modelo obtenido si los datos están concentrados en el estado transiente.

Obviamente, un mayor número de datos requiere más tiempo de ejecución al calcular el *fitness*. Este efecto se estudia utilizando tres sets de datos, uno con 90 datos, otro de 100 y uno de 200. Todos fueron obtenidos al simular el mismo modelo con los mismos parámetros. Los tiempos obtenidos se observan en la Tabla 10.

Tabla 10 Efecto del tiempo al aumentar o disminuir la cantidad de datos

Números de datos	Tiempo promedio [s]	Desviación estándar
90	223	54
100	262	88
200	282	100

Los modelos obtenidos para los gráficos corresponden a:

$$\frac{dh}{dt} = 0.12 \tag{28}$$

$$\frac{dh}{dt} = 0 \tag{29}$$

Estos modelos, aunque representan cada estado por separado, no bastan para representar el fenómeno completo. Por esto, siempre es necesario asegurarse que los datos a utilizar sean suficientes para observar todas las características del fenómeno. La desventaja de emplear un mayor número de datos se refleja en un aumento del tiempo necesario para la ejecución del algoritmo, ya que este requiere ejecutar más comparaciones con el modelo propuesto. Esto se puede apreciar en la Tabla 10, donde la variación de 100 a 200 datos aumenta solo un 7% el tiempo de ejecución.

4.1.5 Efecto de las características

En este apartado se presentan los resultados obtenidos al aplicar las diferentes modificaciones propuestas para el algoritmo de la programación genética implementado en la suite Gplab diseñadas en la metodología, las cuales serán discutidas siguiendo el mismo orden en el que fueron diseñadas en la metodología, para lo cual se estudian los siguientes casos:

1. Ninguna modificación
2. Cada modificación por sí sola
3. Una donde se utilizan todas con excepción de la de agregar bloques conocidos.

Antes de revisar los resultados es necesario recordar que una de las características, agregar bloque conocido, consiste en que la información reunida con anterioridad se le entrega al algoritmo para que lo una a los modelos que propone, de manera que solo se centre en diseñar los términos desconocidos del fenómeno. Para este caso, se consideró conocida la relación entre el flujo de entrada y el área transversal del estanque, datos que normalmente son manejados en la industria. Por lo que PG debe proponer el

término desconocido que será denotado por f(h). De esta manera, el modelo a diseñar queda como:

$$\frac{dh}{dt} = 0.2 - f(h) \quad (30)$$

$$f(h) = 0.08 * \mathrm{h} \quad (31)$$

Para cada caso se realizan cinco iteraciones con un set de datos sin error y otro con error, además de aplicar las modificaciones discutidas en los experimentos anteriores. A continuación, se muestra un resumen de los mejores modelos que representan el cambio de altura de un líquido para cada uno de los casos expuestos en la Tabla 11 , donde para cada caso se tiene un resultado para los sets de datos con y sin error. Además, se toman en consideración elementos como: tiempo que se demora en llegar a la solución, la complejidad en forma del número de nodos requeridos para representar el modelo y presencia de error en los datos. Para el caso de entregarle bloques conocidos del modelo al algoritmo, el término entregado por este último se representa como f(h).

Tabla 11 Resultados PG Altura de Estanque en Diversos Casos

Característica activada	modelo propuesto	Tiempo [s]	Complejidad [nº nodos]	Datos con error
Se activan todas las características con excepción de la de bloques conocidos.	$dh/dt = 4.75 - \frac{h}{43.98}$	661	7	No
	$5.375 - \frac{h^2}{1.075}$	1290	7	Si
Bloques conocidos	$f(h) = -\frac{h}{11}$	204	7	No
	$f(h) = -\frac{h^2}{25}$	212	9	Si
Snipping	$5 - h^2$	235	5	No
	$4.99 - h^2$	248	7	Si
Ajuste	$2.35 - h$	3301	5	No
	$0.3 - 0.13 * h$	2165	7	Si
Limitar el máximo numero de nodos permitido en los arboles binarios	$-\frac{1}{10^6 * (h + \frac{1}{h})}$	131	15	No
	$\frac{4{,}39}{100 * h}$	140	7	Si
Criterio de información de akaike como valor de fitness	$0.05 - \frac{0.25}{h}$	150	9	No
	$\frac{5}{101 * h}$	183	7	Si

Como lo anterior solo muestra el resultado obtenido en la mejor iteración de cada caso, dada la aleatoriedad del algoritmo no significa que estos valores demuestren que las modificaciones son útiles, sino que en la iteración los valores obtenidos por azar fueron adecuados. Por esto se analizan los valores obtenidos en las cinco iteraciones de cada caso. En ellas se calculan los valores promedio y desviación estándar, tanto para el *fitness* del mejor individuo, como para el tiempo de ejecución y complejidad de la solución medida en el número de nodos, lo cual se puede apreciar en las Figura 27, Figura 28 y Figura 29, respectivamente.

El fitness mostrado se calculó en base al error cuadrático medio, por lo que para comparar la modificación que consiste en utilizar el criterio de Akaike (AIC) para calcularlo. Para la modificación anterior se calcula el error cuadrático medio a partir de las simulaciones de los modelos obtenidos.

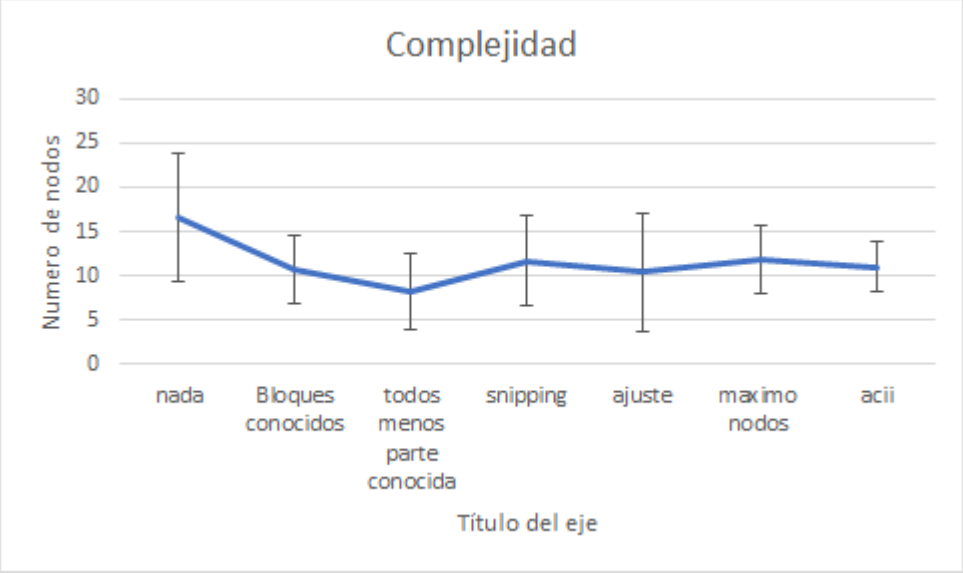

Figura 27 Fitness Promedio al Utilizar Distintas Características

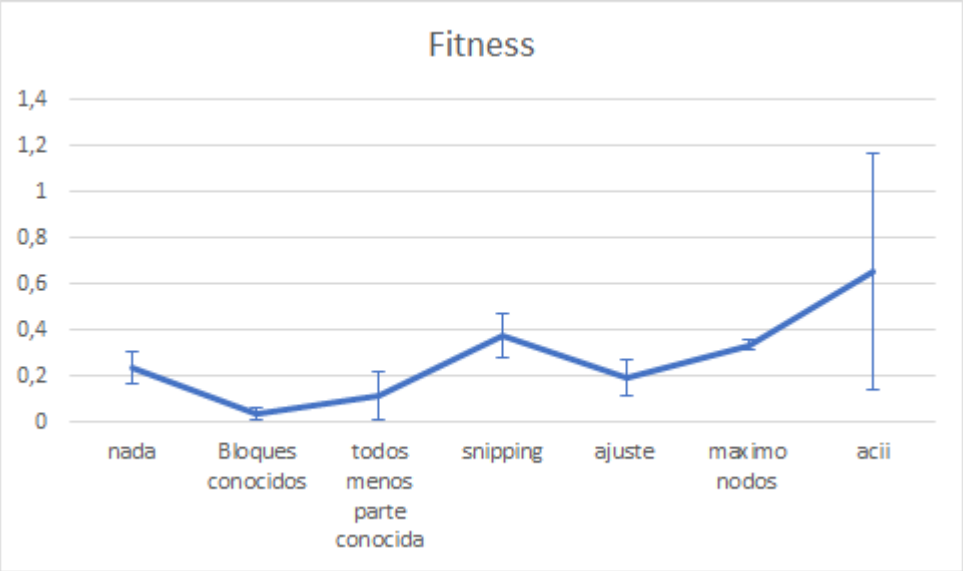

Figura 28 Complejidad de la Solución Entregada Utilizando Distintas Características

Figura 29 Tiempo de Ejecución al Utilizar Distintas Características

La implementación del *snipping* tiene como objetivo disminuir el tamaño de los árboles binarios que conforman el modelo que crean los individuos, lo que ocurre al unir diferentes parámetros en uno solo. Lo anterior se muestra en la Figura 27, donde se aprecia una mejora obteniendo modelos que requieren una menor cantidad de nodos. Sumado a lo anterior, esta modificación provoca una disminución del tiempo de ejecución debido a que la PG ahora está evaluando árboles de menor tamaño.

Asimismo, el *snipping* actúa similar a un ajuste de parámetros, donde el programa al tener varios individuos que diseñan un modelo de manera similar escoge aquel donde los parámetros obtenidos se asemejan más a los datos,

los cuales no necesariamente corresponden a un terminal, sino que a una combinación de estos con funciones. Posteriormente, estos serán reducidos a un solo terminal sin perder información.

La segunda modificación por analizar corresponde a limitar el número de nodos que puede llegar a poseer un árbol binario. Esto tiene como objetivo evitar el crecimiento desmedido y asegurar que al simular los modelos el tiempo necesario no sea desproporcionado. En este caso, el tiempo no sufre un cambio significativo debido a que al ser una ecuación sencilla es más probable que un árbol con pocos nodos sea suficiente para representar el fenómeno. Con relación al *fitness*, existe un aumento al analizar las soluciones. Se observa que el algoritmo encuentra expresiones similares a las deseadas, pero no alcanza los parámetros adecuados. Lo anterior se fundamenta en que una vez el algoritmo diseña un árbol que tenga forma similar, no le quedan nodos para intentar mejorar los parámetros, lo que provoca que el algoritmo deba representar el resultado de manera más eficiente.

En tercer lugar, se tiene la modificación que consiste en utilizar el ajuste de parámetro en cada individuo y cada generación, a diferencia de solo usarlo con el modelo que entrega el algoritmo. Esto tiene un efecto positivo en el *fitness* debido a que es el objetivo de esta función. Como el ajuste se encarga de encontrar los mejores parámetros, el programa no debe gastar nodos en formarlos provocando que las soluciones tengan una disminución en la complejidad. La gran desventaja de utilizar de esta manera el ajuste de parámetro está dada por un aumento desmedido del tiempo de ejecución del algoritmo donde, en este estudio, en promedio el algoritmo se puede demorar hasta 100 veces más, lo que depende del método utilizado. Esto muestra que, al menos que el tiempo no sea una limitante, es más eficiente realizar el ajuste solo al mejor individuo entregado por el algoritmo, tal y como lo propone esta metodología.

Los mejores modelos son obtenidos al utilizar la modificación de entregarle bloques conocidos al algoritmo, de manera que este se encargue de encontrar las partes desconocidas del fenómeno en estudio. En este caso para la altura del líquido dentro del estanque, se posee tanto la información con respecto al flujo de entrada como el área transversal del equipo. El término faltante, denominado f(h), corresponde a una multiplicación entre una constante y la variable de altura. Los mejores modelos se obtienen cuando se usa esta modificación, lo que se debe a que el árbol binario que debe generar el algoritmo requiere un tamaño menor, permitiendo que se utilice un mayor número de generaciones modificando los parámetros para encontrar los más adecuados. Pese a esto, la aplicación de este método trae consigo una

desventaja, relacionada con un aumento en el tiempo de ejecución del algoritmo, lo cual se debe a que sin importar lo útil que sea el individuo al momento de calcular el *fitness*, el modelo se expande.

La última modificación propuesta consiste en utilizar el Criterio de Información de Akaike (AIC) al calcular el valor de *fitness* de cada individuo en la generación, el cual posee un *trade-off* entre la bondad y complejidad del modelo, por lo que la solución propuesta por el algoritmo, aunque se asemeje a los datos, puede no ser escogida por la cantidad de parámetros que utilice. Para comparar los valores de fitness se calcula el error cuadrático medio para estos casos donde el gráfico de fitness (Figura 28) muestra que esta modificación entrega la mayor variación estándar. A pesar de lo anterior, entrega las soluciones más simples con excepción de utilizar el ajuste de parámetro en cada individuo. En cuanto al tiempo, es la segunda característica que entrega los resultados más eficientemente luego del *snipping*.

Como resultado del análisis, entregar información al algoritmo, es decir, agregar bloques conocidos, es el mejor avance para obtener resultados de mejor calidad; mientras que, si el tiempo no es un impedimento, aplicar el ajuste de parámetros en cada individuo entrega las mejores soluciones, a pesar de que el conocimiento del fenómeno sea mínimo. Si lo que se busca es una disminución del tiempo de ejecución, la utilización del *snipping* y AIC son las mejores opciones.

A partir de los resultados y discusiones realizadas con relación a la utilización de la metodología propuesta, al diseñar un modelo con base en los datos de la altura de líquido dentro de un estanque, se puede resumir que de las modificaciones realizadas al algoritmo es aconsejable mantener activada las correspondiente al *snipping* y AIC, ya que ambas ayudan a disminuir la complejidad del resultado entregado, además de disminuir el tiempo de ejecución. Al mismo tiempo, estas modificaciones no requieren de conocimiento sobre el comportamiento del fenómeno. Las demás de las modificaciones deben ser usadas con criterio ya que dependiendo de la necesidad pueden ser contraproducentes, por ejemplo, en el caso de limitar el número de nodos, escoger un valor reducido puede llegar a evitar que el algoritmo genere un modelo adecuado; en el caso contrario, un valor elevado deja sin eficiencia esta característica. Realizar el ajuste del parámetro a cada individuo en cada generación aumenta considerablemente el tiempo de ejecución. Finalmente, agregar bloques conocidos trae mayor beneficio siendo la única restricción la necesidad de información por parte de la persona que utilice el algoritmo.

Con relación a la metodología propuesta se puede apreciar la utilidad de que sea reiterativa, lo que se aprecia en los diferentes experimentos realizados. Cada uno de ellos posee un análisis de los resultados previos aumentando la calidad de estos en cada nueva iteración.

4.2 Modelo 2: Lokta-Volterra

A continuación, se muestran los resultados obtenidos al simular la dinámica entre dos especies biológicas, donde una es un depredador y la otra corresponde a su presa durante 100 días. Este sistema de ecuaciones se denomina Lotka- Volterra y está descrito por el siguiente modelo.

$$\frac{dx}{dt} = 0.1 * x - 0.02 * \ x * y \tag{32}$$

$$\frac{dy}{dt} = 0.04 * x * y - 0.4 * y \tag{33}$$

Este define el cambio a través del tiempo de 2 especies una presa (x) y un depredador (y). Donde los parámetros representan lo siguiente:

1)0.1[1/día]: La reproducción de las presas.
2)0.02[1/día*nºdepredadores]: velocidad de consumo de las presas por parte de los depredadores.
3)0.04[1/día*nºpresas]: La reproducción de los depredadores.
4)0.4 [1/día]: el porcentaje de decesos de los depredadores.

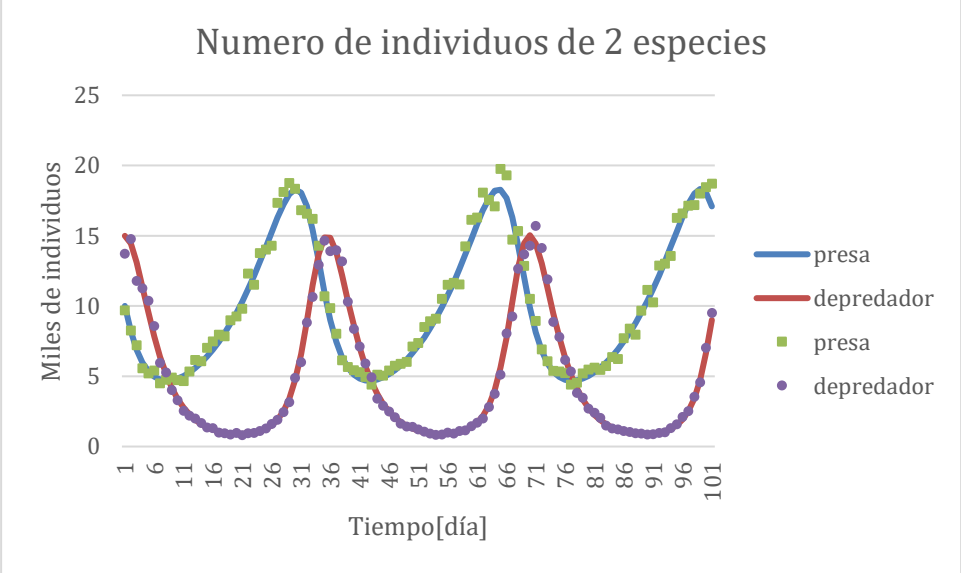

Figura 30 datos obtenidos al simular el modelo de Lotka-Volterra.

En este gráfico se observa que tanto los depredadores como las presas poseen un comportamiento oscilatorio con un periodo de 2*pi/sqrt(0.04*0.1). Este comportamiento se debe a la relación de las especies: los depredadores disminuyen, ya que no poseen la suficiente comida para sustentarse, llegando casi a la extinción, momento en el cual la población empieza a aumentar gracias a la presencia de las presas. Estas, por su parte, tienen un ciclo similar que parte disminuyendo, debido a que son consumidas, para luego empezar a aumentar en el momento en el que el consumo es menor al nacimiento de las presas, el máximo se alcanza cuando los depredadores son suficiente para igualar el consumo a la reproducción de las presas.

Un análisis matemático muestra que, si en algún momento los depredadores y presas poseen una población de 10 y 5 respectivamente, el cambio de población para ambas especies se vuelve 0, provocando que a partir de ese momento se mantengan con esos valores alcanzando un estado estacionario. Además de esto, si en algún momento las presas se extinguen (x=0), los depredadores empiezan a disminuir a un paso que cada vez es más lento hasta el punto en el que desaparecen. En el caso contrario, es decir, si se extinguen los depredadores, las presas empiezan a aumentar a un ritmo que va acelerando con el tiempo. Sumado a lo anterior también se puede obtener que el valor del periodo se obtiene al evaluar el jacobiano de la función en los punto de equilibrio [x=10 y=5], obteniendo que los valores propios son x1=sqrt(0.4*0.1)*i x2=-sqrt(0.4*0.1)*i con lo cual se obtiene que el periodo está dado por 2*pi/sqrt(0.4*0.1), es decir, depende solo de los parámetros relacionado con la reproducción de las presas y los decesos de los depredadores.

La implementación computacional del algoritmo al momento de realizar el trabajo de memoria no permite la obtención del sistema de ecuaciones completo, sino que PG puede proponer uno de los 2 sistemas que lo compone.

Por esto es necesaria la utilización de la modificación de agregar bloques conocidos, acción que en este caso corresponde a una de las ecuaciones del modelo de Lotka-Volterra. Debido a lo anterior, y a diferencia del Modelo 1 (Altura del líquido en un estanque) donde en un inicio se considera que la información que se posee es mínima o nula, para este caso es necesario tener los conocimientos suficientes para proponer uno de los sistemas. En el estudio

de este modelo se considera conocido el cambio de los depredadores a través del tiempo. Por lo anterior, si se define el lado derecho de la ecuación de las presas como f(x), siendo este el término que debe proponer el algoritmo se tiene:

$$\frac{dx}{dt} = f(x, y) \tag{34}$$

$$\frac{dy}{dt} = \sigma * x * y - \gamma * y \tag{35}$$

4.2.1 Resultado iteración 1

Para la primera iteración se considera que la información con respecto a la variación de las presas es desconocida, por lo que, aplicando la metodología propuesta se diseña un modelo que represente los datos expuestos en la Figura 30. El primer paso corresponde a la recopilación de datos e información, seguido de la utilización del algoritmo. Este proceso se realiza cinco veces para un primer acercamiento, donde se utiliza un set de parámetros predeterminados, los que se expresan en la Tabla 5. En la Tabla 11, se presentan los resultados obtenidos de cada repetición, donde solo se conoce la ecuación que define el cambio de los depredadores a través del tiempo.

Tabla 12 Resultados de la PG obtenido para la ecuación de las presas del modelo Lokta-Volterra

Ecuación	**Iteración**	***Fitness***	**Tiempo (segundos)**
$\frac{(\frac{y}{-0.91} - ((cos\ (sin\ (11.2747)) + x) * (-0.011) + (y - 11{,}3)))}{x - -0.91}$	1	24.88	412
$\frac{x}{y * y * y}$	2	47.6	417
$cos\ (y - e^{cos\ (y - e^{x*y})*y})$	3	56.27	345
$sin\ (x - (x - sin\ sin\ (y)\ ) * y - y)$	4	54.7	475
$\frac{y}{\frac{sin\ (x * y)}{y}}$	5	48.1	493

En primer lugar, se observa que la mayoría de los modelos posee un cociente con alguna de las variables de población en el denominador. Esto no genera problema en este experimento, ya que los datos utilizados no poseen valores que indeterminen la ecuación. Sin embargo, si las condiciones iniciales fueran distintas, se puede tener el caso de que alguna especie se extinga, es decir,

que la población tenga 0 individuos, valor que indetermina algunos de los modelos obtenidos en este experimento.

En segundo lugar, se observa que las funciones trigonométricas no son utilizadas y, en el caso de estar presentes, no generan los cambios necesarios. Por ejemplo, los Modelos 3 y 4 pueden entregar valores entre -1 y 1, pero los datos muestran que el cambio que presentan ambas especies es mayor a estas cifras. A partir de esto se puede sugerir que los efectos oscilatorios no son provocados por las funciones trigonométricas, por lo que para futuras interacciones es conveniente disminuir la frecuencia con que se utilizan o incluso descartarlas. Finalmente, dado que es una ecuación más compleja, el tiempo de ejecución es mayor con respecto al modelo anterior.

Al simular los modelos descritos en la Tabla 11 se obtienen las trayectorias para la población de los depredadores y las presas, las que están representadas en las Figura 31 y Figura 32 respectivamente, donde cada línea representa los datos obtenidos al utilizar uno de los resultados mostrados en la tabla anterior.

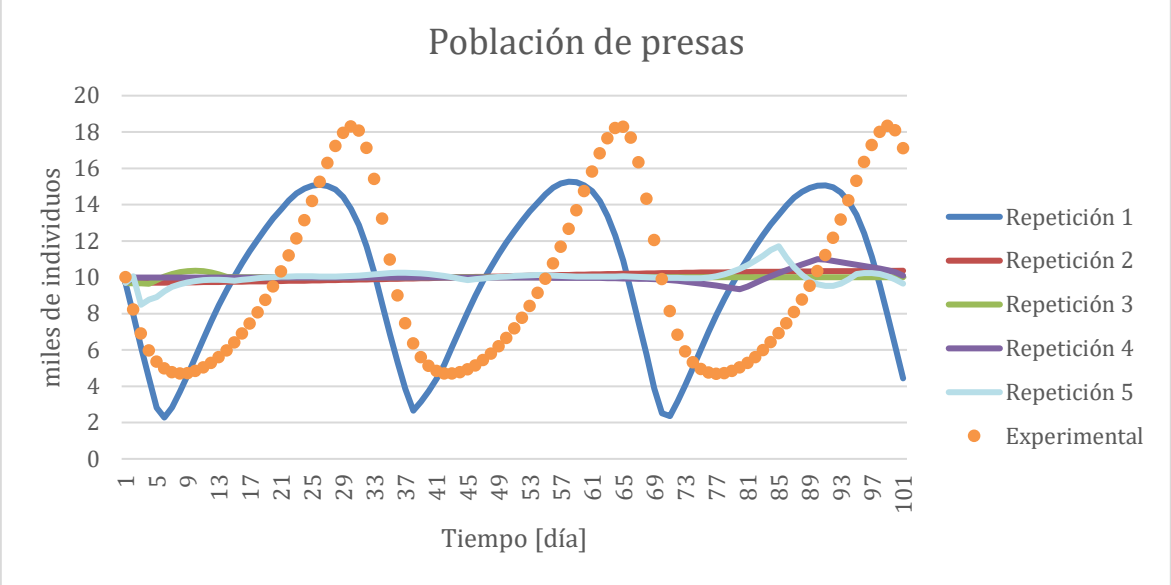

Figura 31 Población de presas al simular modelos obtenidos por el algoritmo.

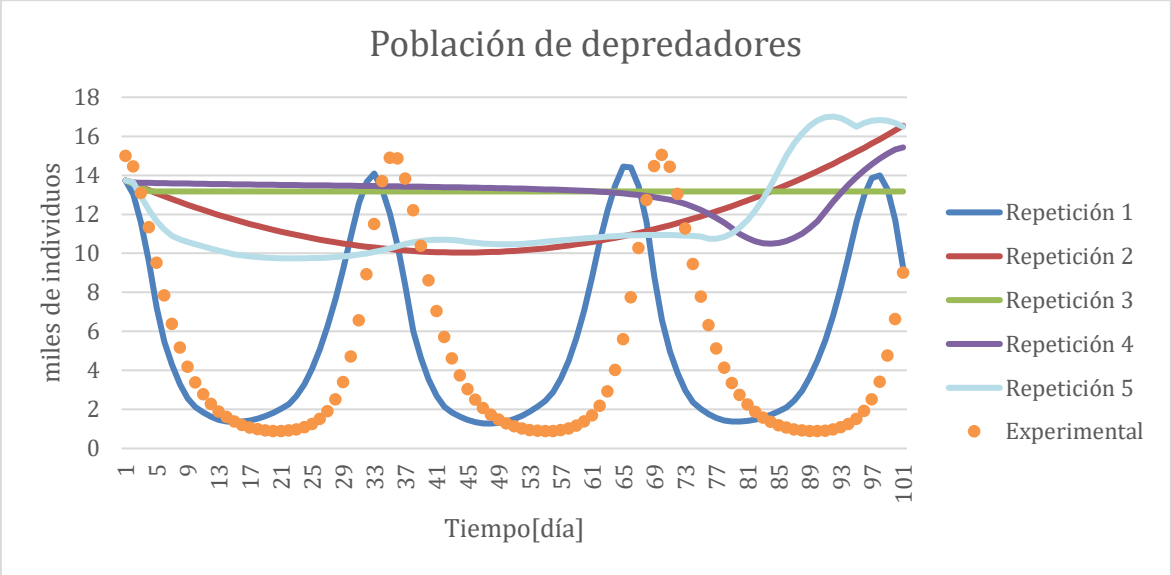

Figura 32 Población de depredadores al simular modelos obtenidos por el algoritmo.

En ambas figuras se puede observar que la única simulación que se asemeja al comportamiento presentado en los datos corresponde al modelo obtenido de la Repetición 1.

De este modelo resulta, en primer lugar, que la variación de presas a través del tiempo depende tanto de sí misma como del número de depredadores. En segundo lugar, se pueden apreciar tres términos en el modelo: los dos primeros muestran el incremento de las presas, el cual depende del número de estas; mientras que el último término representa una disminución que depende de la población de las especies presentes. Finalmente, a partir de las discusiones realizadas, para mejorar los resultados en posteriores iteraciones se debe limitar el uso de las funciones trigonométricas, exigir que las soluciones presenten ambas variables, además de aumentar el número de individuos y generaciones.

4.2.2 Resultado Final

En este apartado se presenta el mejor modelo obtenido tras una segunda iteración de la metodología para los datos con y sin error. Primero, se probó asumiendo conocido el término que representa el aumento generado por la reproducción de las presas. A partir de ello, se obtuvo el siguiente sistema de ecuaciones, donde f(x,y) es el término que debe proponer el algoritmo:

$$\frac{dx}{dt} = 0.1 * x - f(x,y) \quad (36)$$

$$\frac{dy}{dt} = 0.04 * x * y - 0.4 * y \quad (37)$$

A continuación, se muestra el mejor modelo obtenido de 5 repeticiones para cada set de datos. En la Tabla 13 se detalla el término propuesto por el algoritmo junto con el fitness calculado como error cuadrático medio, mientras que la tabla Tabla 14 muestra el coeficiente de correlación para las distintas variables para ambos casos:

Tabla 13 Modelos Obtenidos al buscar parte de la expresión de las presas.

	Sin error	Con error
Ecuación	$-0.0183 * x * (y - 0{,}1)$	$-0.009 * y * (x * (1 + 1.0092^{y}) - 0.55)$
Fitness	3.59	7.97

Tabla 14 coeficiente de correlación obtenidos al buscar parte de la expresión de las presas.

Variable	Set de datos	Coeficiente de correlación
X	Sin error	0.932
Y	Sin error	0.913
X	Con error	0.857
y	Con error	0.837

Las Figura 33 y Figura 34 muestran el resultado obtenido al simular el modelo representado por las ecuaciones (36) y (37), reemplazando el término f(x) por el mostrado en la tabla anterior para los casos con y sin error, respectivamente, donde las líneas continuas representan la simulación de los modelos mientras que los puntos son los valores experimentales.

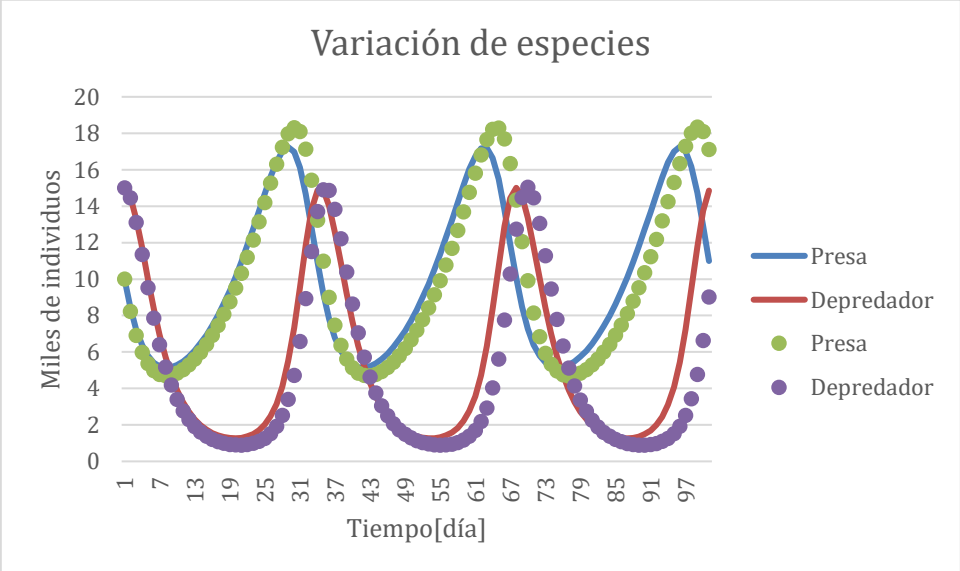

Figura 33 Mejor Modelo Obtenido a Partir de los Datos sin Error para el Caso Lotka-Volterra.

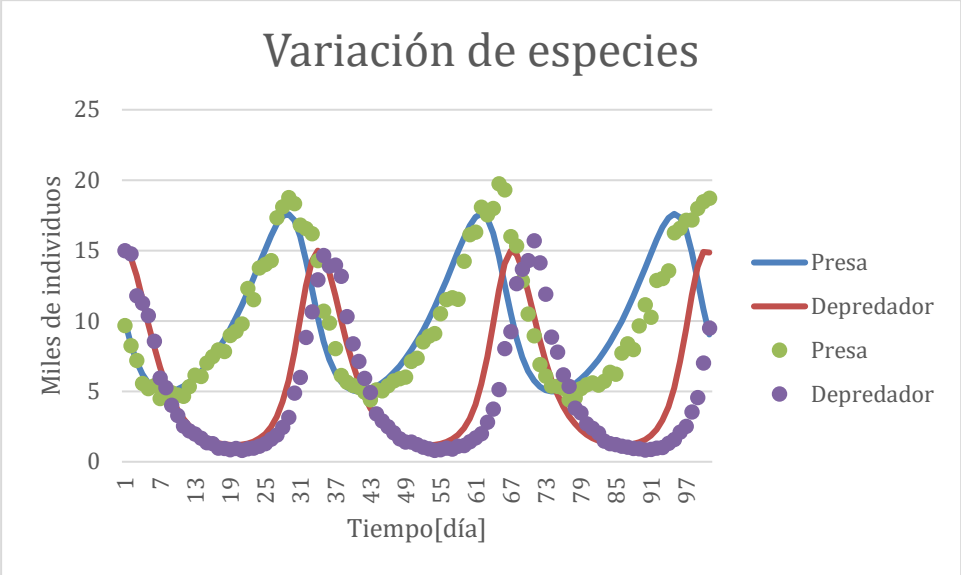

Figura 34 Mejor Modelo Obtenido a Partir de los Datos con Error para el Caso Lotka-Volterra.

El término faltante de la ecuación de las presas corresponde al término que representa la disminución de estas, debido al consumo provocado por parte de los depredadores. A continuación, se analizan los resultados del algoritmo que corresponden a propuestas para el término. Primero, para el caso de datos sin error se tiene el siguiente modelo de Lotka -Volterra

$$\frac{dx}{dt} = 0.1 * x - 0.0183 * x * y + 0.00183 \quad (38)$$

$$\frac{dy}{dt} = 0.04 * x * y - 0.4 * y \quad (39)$$

A partir de la Figura 33 es posible identificar que la solución propuesta posee el comportamiento oscilatorio de los datos, representándolos de manera cercana, donde el último término de la ecuación que define el cambio de presas a través del tiempo refleja que existe un aumento constante del número de las presas. Este incremento es insignificante con respecto al provocado por el primer término, que muestra la reproducción ocurrida en un intervalo de tiempo, ya que porcentualmente este término aporta entre el 99.999% y 99.99% del aumento de presas. El cambio que genera es tan mínimo que, si se vuelve a simular el modelo ignorando esta constante, la diferencia calculada

como error cuadrático media obtenida es de 0.0112 que es un valor bajo cuando las variables utilizados son del orden de 10. Por lo anterior, el diseño del modelo queda como:

$$\frac{dx}{dt} = 0.1 * x - 0.0183 * x * y \quad (40)$$

$$\frac{dy}{dt} = 0.04 * x * y - 0.4 * y \quad (41)$$

Sumado a esto, se observa que la simulación del modelo propuesto tiene dos diferencias con respecto a los datos originales: el máximo número de depredadores y presas que se puede alcanzar disminuye. Lo anterior se debe a que el coeficiente del término de disminución de presas es menor, lo que provoca que el máximo de depredadores se alcance más temprano, igualando el aumento provocado por la reproducción. La otra diferencia se encuentra en el tiempo del periodo, el cual es menor por la misma razón.

El segundo caso entrega el término para diseñar el modelo a partir de los datos con errores el cual se representa de la siguiente manera:

$$\frac{dx}{dt} = 0.1 * x - 0{,}009 * y * x * (1 + 1{,}0092^{y}) + 0.00495 * y \quad (42)$$

$$\frac{dy}{dt} = 0.04 * x * y - 0.4 * y \quad (43)$$

Este modelo gráficamente logra representar de forma cercana los datos y entrega los comportamientos deseados, lo cual se puede observar en Figura 34. Se identifican dos diferencias en el número máximo de individuos que puede alcanzar cada especie y el periodo de las oscilaciones, lo cual se debe a los coeficientes que poseen distintos valores. Tras un análisis matemático de la ecuación, se puede extraer que: 1) la constante elevada a la variable de los depredadores varía entre 1.01 y 1.16 con un promedio de 1.074 durante la simulación. Si se simula el modelo reemplazando la constante por su promedio, se tiene que el error cuadrático medio entre ambos es de 0.3, por esto se puede simplificar el modelo como:

$$\frac{dx}{dt} = 0.1 * x - 0.0187 * y * x + 0.00495 * y \quad (44)$$

$$\frac{dy}{dt} = 0.04 * x * y - 0.4 * y \quad (45)$$

El tercer término de la ecuación que representa el cambio de presas muestra un aumento de estas por la presencia de depredadores, a diferencia del modelo obtenido a partir de los datos sin error este término no puede ser ignorado. Esta variación explicita que existe una ganancia para las presas por la presencia de la otra especie, esto puede significar que existe un nivel de simbiosis recíproca de algún nivel desconocido (comensalismo, proto-cooperación o mutualismo), además de la depredación que es la común entre ambas especies [61].

Lo anterior sería posible si solo se consideran los intervalos de tiempo de datos entregados, ya que, si se proyecta la simulación en un intervalo mayor, esta última se comporta de manera distinta a la que se observaría en la realidad. Este comportamiento se puede apreciar en las Figura 35 y Figura 36 donde se muestran los resultados de simular el número de especies durante 1000 días, tanto para el modelo propuesto por el algoritmo como el original.

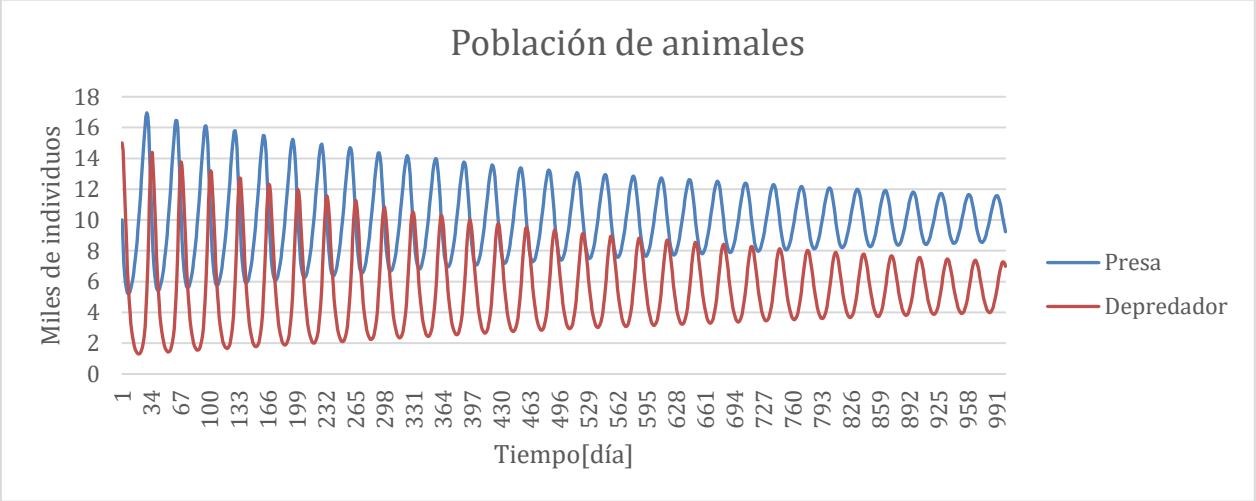

Figura 35 Simulacion del modelo diseñado para Lotka-Volterra a partir de los datos con error por 1000 dias.

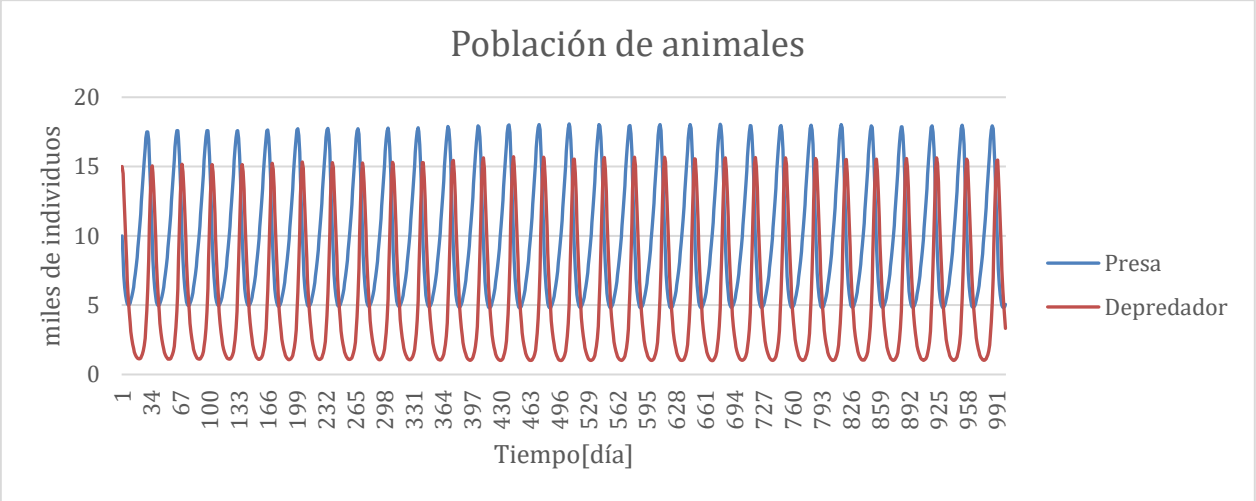

Figura 36 Simulacion del modelo original de Lotka-Volterra por 1000 dias.

En la Figura 35 se observa que a medida que pasa el tiempo las oscilaciones de ambas especies mantienen su periodo, pero disminuyen su amplitud. Esta disminución seguirá ocurriendo hasta que ambas poblaciones alcancen un estado estacionario. Por el contrario, la segunda mantiene una amplitud más o menos constante al punto en que la diferencia puede deberse al método de integración utilizado. Por lo anterior se debe ignorar el término que aumenta el número de presas basados en la presencia de depredadores, obteniendo un modelo similar al diseñado con los datos sin error, el cual mantiene el comportamiento oscilatorio a través del tiempo.

Sumado a lo anterior se observa que el modelo propuesto tiene un comportamiento oscilatorio amortiguado, donde la amplitud seguirá disminuyendo hasta alcanzar el equilibrio, 10 miles de individuos para las presas y 5.49 miles de individuos para los depredadores. Además los máximos locales siguen un comportamiento exponencial lo que se observa en el siguiente gráfico:

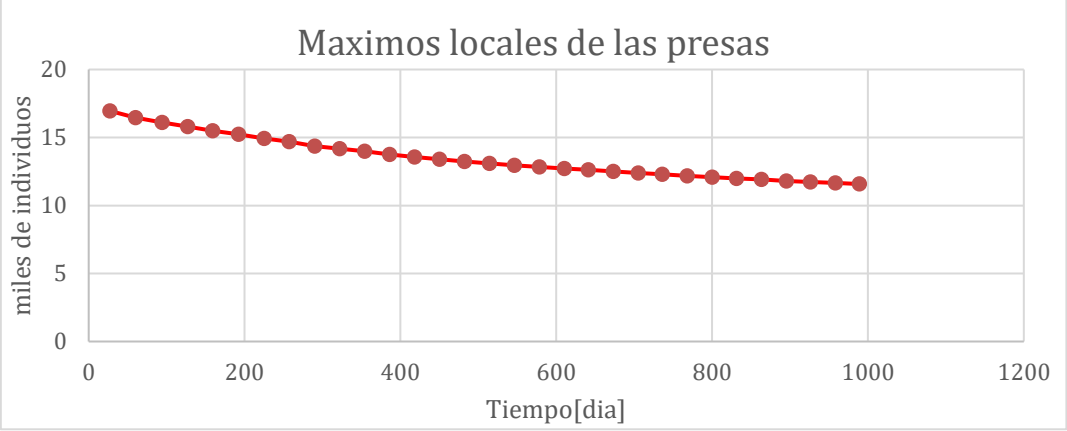

Figura 37 Máximos locales de

Este comportamiento se produce por el tercer término del modelo propuesto para los valores con error, ya que aumenta el mínimo que puede alcanzar la presa lo que en consecuencia provoca que las presas tengan un compuesto similar. Seguido de esto el máximo de población de las presas disminuye porque el numero de depredadores es mayor al que tendría en una situación normal, esto último provoca una disminución del máximo de población de la otra especie. Esto se sigue produciendo hasta alcanzar el estado estacionario.

El segundo experimento realizado para este apartado es similar al primero, con la diferencia que ahora el algoritmo debe proponer la ecuación que representa el cambio de presas a través del tiempo en su totalidad. Los datos utilizados son los mismos a los del caso anterior; de igual forma se realizaron cinco iteraciones del algoritmo. En la Tabla 15, se presenta el mejor resultado obtenido para cada set de datos y la Tabla 16 muestra el coeficiente de correlación para las distintas variables para ambos casos. La simulación de estos modelos se puede observar en las Figura 38 y Figura 39, donde las líneas continuas representan la simulación de los modelos mientras que los puntos son los valores experimentales.

Tabla 15 Mejores Modelos Obtenidos al Buscar el Comportamiento de las Presas en su Totalidad.

	Sin error	Con error
Ecuación	0.12*(x-0.11)-0.056*y*x	$0.118 * x - y * x * 0.02 - 0.05$
Fitness	104	18

Tabla 16 Coeficiente de correlación de los modelos que representan el comportamiento de las presas.

Variable	Set de datos	Coeficiente de correlación
Presas	Sin error	-0.311
Depredadores	Sin error	-0.027
Presas	Con error	0.680
Depredadores	Con error	0.624

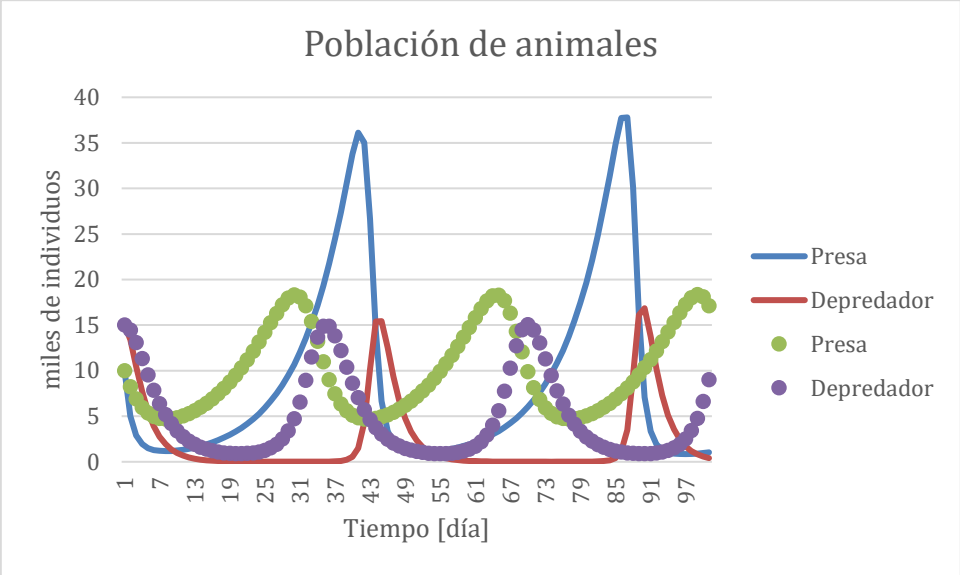

Figura 38 Modelo Obtenido para los Datos sin Error.

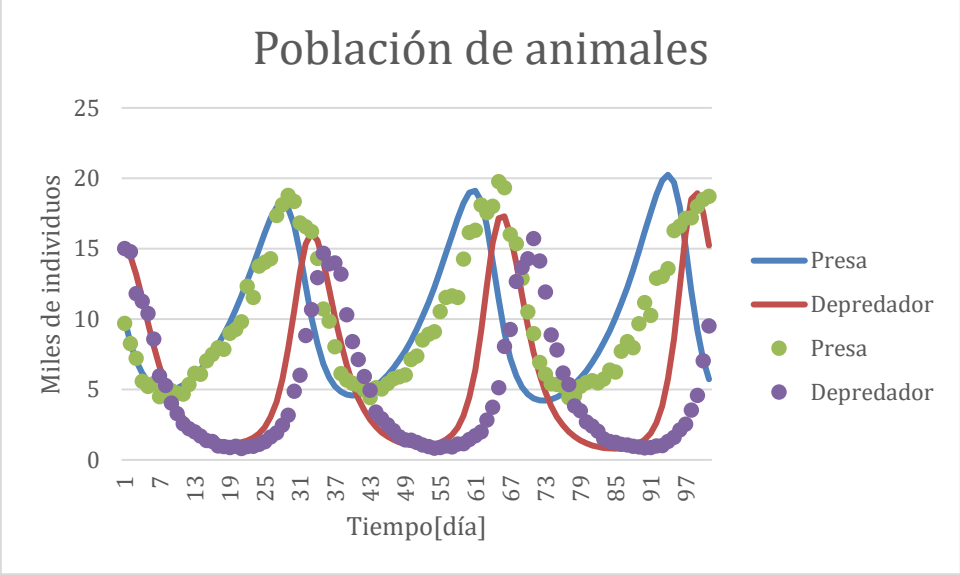

Figura 39 Modelo Obtenido para los Datos con Error.

De la tablas y figuras se observa que la calidad de los modelos obtenido disminuye con respecto al experimento anterior, lo cual es lo esperado ya que ahora el algoritmo esta trabajando con menos información. En específico para el resultado obtenido del set de datos sin error, el valor de coeficiente de

correlación es negativo para la variable de las presas (x) y un valor cercano a 0 para los depredadores (y), esto indica que para la primera variable el modelo obtenido entrega valores en la dirección opuesta y para el segundo que no existe correlación entre los datos originales y los obtenidos al simular el resultado. Sin embargo, un análisis visual muestra que el modelo logra simular el comportamiento oscilatorio y la amplitud de los depredadores se asemejan, más aún la función obtenida poseen los mismos términos con diferencia solo en los coeficientes alcanzado. Finalmente, de la Tabla 15 se observa que los datos con error son mejor con un *fitness* 10 veces menor.

Las diferencias mostradas anteriormente se deben a que la población inicial influye en gran medida en los resultados. En este experimento, al analizar el *fitness* de la población inicial de las iteraciones que entregaron los mejores resultados para cada set de datos, se observa que para los datos con error, un 15% de los individuos poseen un *fitness* menor a 100, mientras que la otra población no tiene un individuo con ese *fitness*, lo que en términos del algoritmo significa que desde el inicio la población con error tiene mejores soluciones que el mejor resultado obtenido a partir de los datos sin error. Esta diferencia significa que cuando se utilizaron los datos sin error el algoritmo debió gastar un mayor número de generaciones en crear soluciones factibles, mientras que en el otro set se utiliza más generaciones en ajustar parámetros. Por esto podría ser una buena idea, si es que se conoce qué valor de *fitness* es aceptable para la solución, crear set de generaciones iniciales hasta que alguna tenga un porcentaje mínimo de individuos con un *fitness* aceptable.

Por otro lado, la Figura 40 muestra la simulación del modelo que se obtiene si se le permite al algoritmo utilizar 10 generaciones más con el set de datos sin error, donde se obtuvo un *fitness* de 9.3, una correlación de 0.925 y 0.861 para las variables de las presas y depredadores respectivamente, el modelo de las presas representado por la ecuación (46).

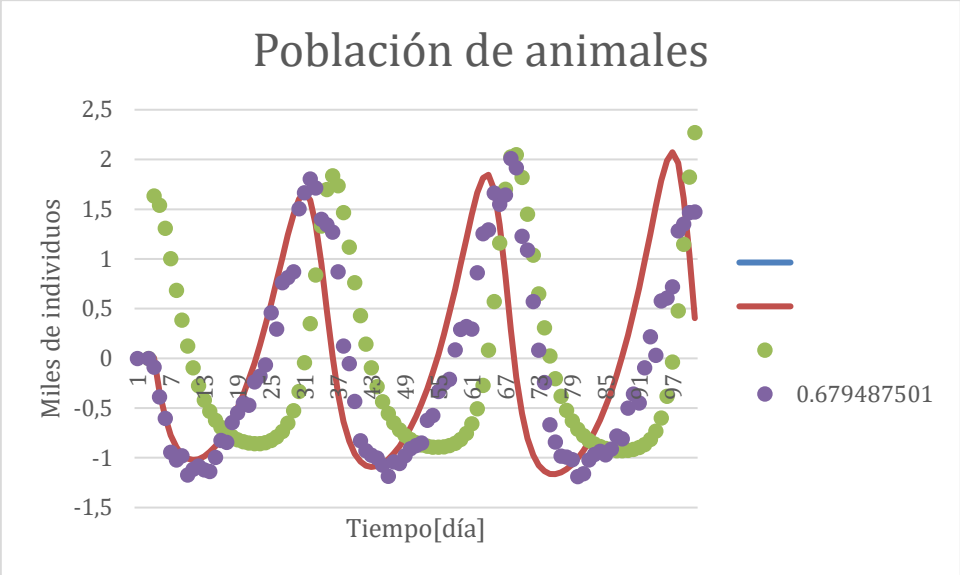

Figura 40 Resultado al Otorgarle 10 Generaciones Más.

$$\frac{dx}{dt} = 0{,}131 * x - 0{,}035 * y * x - 0{,}009 \quad (46)$$

Ambos modelos son similares en cuanto a las funciones que utilizan, diferenciándose en los parámetros escogidos, por lo que el análisis realizado para uno de ellos es similar para el otro. En primer lugar, el algoritmo logra proponer tanto el término que aumenta el número de presas a través de la reproducción, como el de la disminución a través del consumo por parte de los depredadores. Acto seguido, una expresión que muestra una disminución constante por parte de las presas que puede deberse a muerte por edad o el medio es hostil. Este último término al ser eliminado genera un cambio en la simulación menor a 1 al calcular el error cuadrático medio por lo que puede ser eliminado.

Finalmente se puede resumir que, aunque se logran obtener buenas aproximaciones del modelo de Lotka-Volterra tanto para los datos con y sin error, el algoritmo utilizado posee ciertas desventajas, siendo la más importante el fenómeno en estudio, el cual, si está definido por un sistema de ecuaciones, la PG solo puede proponer una de ellas, por lo que se debe poseer la información necesaria para conocer el resto del sistema.

Debido a que el trabajo de memoria no está enfocado en el ámbito computacional existen varias funciones y características de la suite de Gplab que no fueron estudiadas y/o implementadas. En los experimentos realizados se utiliza la función "gplab", la cual al entregarle los sets de parámetros y los datos experimentales realiza, de forma automática, una variedad de pasos, siendo los más importantes para la realización de la memoria: diseñar los

individuos, calcular su *fitness*, crear las nuevas generaciones y escoger el mejor resultado. Una persona es capaz de realizar los pasos mencionados anteriormente de manera manual, dando mayor control al usuario de cómo se relacionan los individuos. De esta manera, podría ser posible crear dos sets de poblaciones, donde cada una representa una ecuación de un sistema de ecuaciones que se quiere estudiar, donde los individuos son evaluados en pares.

4.3 Modelo 3: Velocidad de Reacción Oxidación de Monóxido de carbono

A continuación, se muestran los resultados obtenidos al simular la concentración de gases en la superficie de una placa de platino y la presión parcial de dióxido de carbono dentro del reactor, mientras se mantiene una presión constante de oxígeno y monóxido de carbono durante 100 segundos. El sistema descrito anteriormente está representado por el siguiente modelo:

$$\frac{d\theta_{Co}}{dt} = k_1 * S_{CO}(\theta_{Co}, \theta_O, T) - k_2 * \theta_{CO} * e^{k_3 * \theta_{CO}} - k_4 * \theta_{CO} * \theta_O \quad (47)$$

$$\frac{d\theta_O}{dt} = k_5 * (1 - \theta_{Co} - \theta_O)^2 - k_6 * \theta_O - k_6 * \theta_O - k_4 * \theta_{CO} * \theta_O \quad (48)$$

$$\frac{dp_{CO_2}}{dt} = -k_8 * p_{CO_2} + k_7 * k_4 * \theta_{CO} * \theta_O \quad (49)$$

La Figura 41 muestra la simulación en el intervalo de tiempo desde 0 a 100 segundos. Este fue simulado utilizando los parámetros detallados en la

Tabla 5, mediante una implementación del método de Runge Kutta orden 45 a través del software de Matlab.

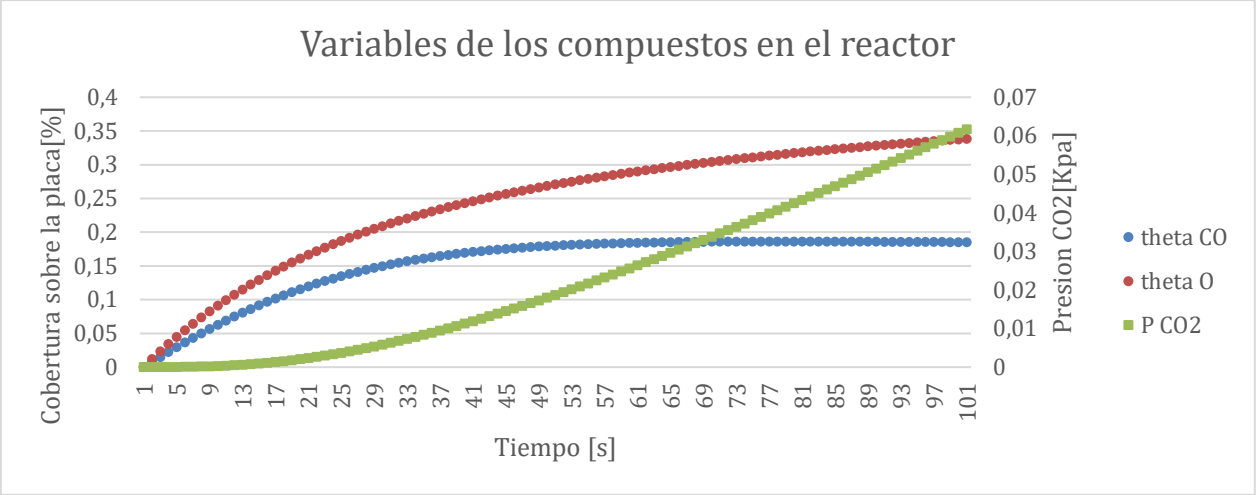

Figura 41 Reacción de CO2 en Placa de Platino-

En la figura se observa que tanto el cubrimiento de oxígeno como el de monóxido de carbono sobre la placa, en el inicio poseen un incremento de tipo exponencial que converge a 0.4 y 0.18 respectivamente. Por su parte, la presión parcial del dióxido de carbono tiene un aumento mínimo al inicio, el cual se incrementa hasta alcanzar un cambio constante de presión parcial a través del tiempo, punto desde el que se comporta como una función lineal.

La implementación computacional del algoritmo al momento de realizar el trabajo de memoria no permite la obtención del sistema de ecuaciones completo, sino que la PG puede diseñar una ecuación. Para el estudio realizado, en este apartado se considera lo siguiente:

1) Es conocido que, tanto la cobertura de oxígeno como la del monóxido, poseen términos correspondientes al comportamiento con el catalizador, en este caso una placa de platino.
2) Se conoce que la expresión de la velocidad de reacción está presente en las 3 ecuaciones del sistema.

A partir de lo anterior, se estudian 2 experimentos donde la información que se posee es distinta. En el primero de ellos, se asume conocida toda la información con excepción de la velocidad de reacción, la que ocurre en el catalizador. Si se define BD como el término de la reacción, el cual es propuesto por el algoritmo, a partir de lo anterior se posee el siguiente modelo:

$$\frac{d\theta_{Co}}{dt} = k_1 * S_{CO}(\theta_{Co}, \theta_O, T) - k_2 * \theta_{CO} * e^{k_3 * \theta_{CO}} - BD \quad (50)$$

$$\frac{d\theta_O}{dt} = k_5 * (1 - \theta_{Co} - \theta_O)^2 - k_6 * \theta_O - BD \quad (51)$$

$$\frac{dp_{CO_2}}{dt} = -k_8 * p_{CO_2} + k_7 * BD \quad (52)$$

En el segundo experimento, el término desconocido corresponde a la interacción entre el oxígeno en forma de gas y el catalizador, donde para este caso se comienza con el siguiente modelo:

$$\frac{d\theta_{Co}}{dt} = BD - k_4 * \theta_{CO} * \theta_O \quad (53)$$

$$\frac{d\theta_O}{dt} = k_5 * (1 - \theta_{Co} - \theta_O)^2 - k_6 * \theta_O - k_4 * \theta_{CO} * \theta_O \quad (54)$$

$$\frac{dp_{CO_2}}{dt} = -k_8 * p_{CO_2} + k_7 * k_4 * \theta_{CO} * \theta_O \quad (55)$$

4.3.1 Primera Iteración

A continuación, se muestran los resultados obtenidos en la primera iteración de diseño del modelo, donde se realizaron cinco repeticiones del algoritmo al primer experimento, donde el algoritmo debe proponer el término de la velocidad de la reacción. La simulación de los modelos obtenidos se observa en la Figura 42, donde todos los valores obtenidos son iguales por lo que se representan a través de la línea de color morado.

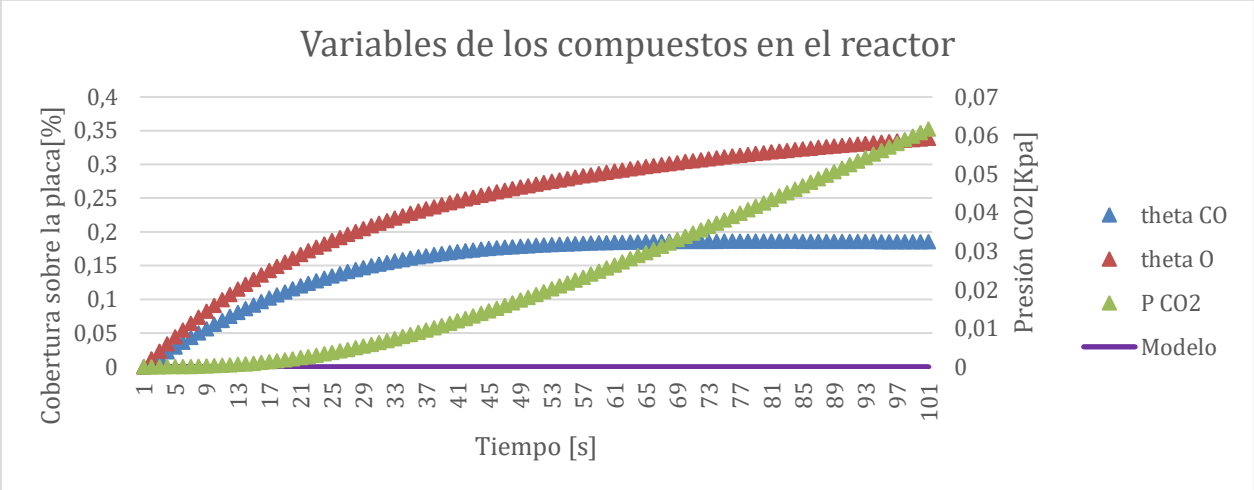

Figura 42 Resultado que no pueden ser simulados obtenidos por el algoritmo.

Al observar la figura anterior, es notorio que el algoritmo no logra proponer un término adecuado, por lo que un análisis matemático de estos no generaría más información relevante ya que el valor obtenido de la simulación no es correcto. La razón de esta situación, entendida como la obtención de un valor de 0 fijo, se debe a cómo se diseñó la modificación que se encarga de tratar

las ecuaciones diferenciales. Esta modificación limita el tiempo de integración permitido durante el paso del *fitness,* el cual durante la realización del trabajo de memoria se fijó en 10 segundos.

Esto se realizó con el objetivo de evitar un incremento desmedido en el tiempo esperado, debido a las funciones *stiff* que se podrían crear durante la ejecución del algoritmo, en el caso que se sobrepase este valor el algoritmo termina la integración del modelo y fija los resultados faltantes de la simulación el valor 0.

En cambio, al analizar el resto de los individuos se observa que existen algunos que pueden ser evaluados en el tiempo permitido. Sin embargo, al calcular el *fitness* de estos individuos, se obtiene un resultado peor que el que entregan aquellos donde se sobrepasa el tiempo de integración. Esto muestra que, la modificación diseñada originalmente en modelos complejos genera un efecto negativo, por esto se vuelve a diseñar obteniendo el Algoritmo 7.

Algoritmo 7: *Fitness* con integración

```
1. Fitness(tree,X){
2.   F=treetofuncion(tree)
3.   Time=tic
4.   While Time<t
5.           Y=ode(F,X)
6.           Fitness_ind=f(Y,X)
7.           Return [Y Fitness_ind]
8.   Y=zeros(X)
9.   Fitness_ind=inf
10.  Return [Y Fitness_ind]
11.  }
```

Este algoritmo representa cómo se calcula el *fitness* desde este punto en adelante. La diferencia con respecto al diseño original consiste en que, si el tiempo es sobrepasado, aparte de fijar los resultados faltantes de la simulación en 0, también se fija el valor del *fitness* del individuo como infinito. Esto provoca que el algoritmo ignore esta posible solución al momento de crear la nueva generación.

Es necesario notar que, dependiendo del fenómeno con el que se esté trabajando, los modelos pueden ser más o menos complejos que los mostrados en este estudio, requiriendo diferentes tiempos de integración. Esto

provocaría que fuera necesario escoger un buen valor del tiempo; si el valor es demasiado bajo, el algoritmo ignora la mayoría de las soluciones. En cambio, un valor elevado generaría un aumento del tiempo a procesar ecuaciones *stiff*. En el peor de los casos, todos los individuos simulados serían funciones *stiff*, lo que provocaría que el tiempo de ejecución mínimo del algoritmo consistiera por la multiplicación de los siguientes valores: el número de individuos, el número de generaciones y el tiempo límite de integración.

4.3.2 Segunda Iteración

En este apartado se presenta el mejor modelo obtenido tras una segunda iteración para ambos experimentos. En el primero de ellos, el algoritmo debe proponer el término de la velocidad de reacción, donde el mejor resultado obtenido lo representa como una constante, con un valor de 0.0018 que es similar al valor obtenido cuando la reacción llega a su estado estacionario. Al simular el modelo de la reacción de oxidación del monóxido de carbono sobre la placa de platino, utilizando esta constante como la velocidad de reacción, los resultados obtenidos se aprecian en la Figura 43, con la Tabla 17 los valores obtenidos para el coeficiente de correlación para cada variable.

Tabla 17 coeficiente de correlación Modelo con Reacción Orden 0.

Variable	Coeficiente de correlación
θ_{Co}	0.982
θ_{O}	0.971
p_{CO_2}	0.914

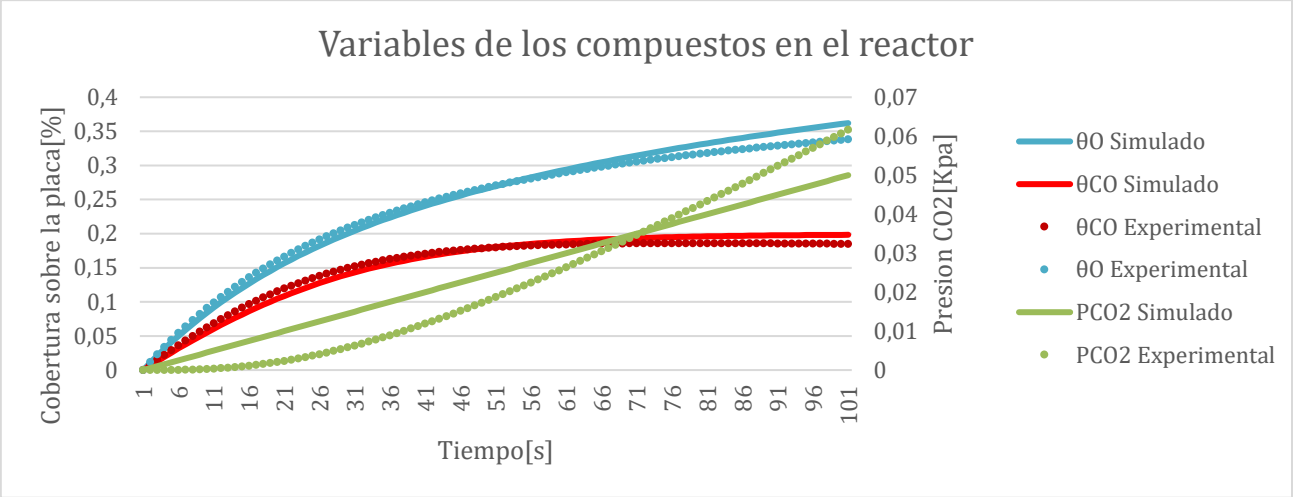

Figura 43 Comparación Modelo con Reacción Orden 0.

Tomando como base la figura y tabla anterior, se puede suponer que el modelo es adecuado para representar los datos. Sumado a lo anterior, desde un punto de vista matemático, el modelo no presenta irregularidades. Sin embargo, una reacción química cuya velocidad está representada solo por una constante es conocida como una reacción de orden 0. Estas son independientes de la concentración de productos y reactantes.

El modelo estudiado en esta sección representa una reacción que no sigue este comportamiento, lo que se ha demostrado en estudios previos[62]. Por este motivo, se debería realizar una nueva iteración del experimento, con la diferencia de exigir el uso de las variables de cubrimiento de los gases sobre la placa.

En el caso del segundo experimento, se espera que el algoritmo diseñe la ecuación de adsorción y desorción para el cambio de cubrimiento de oxígeno sobre la placa a través del tiempo. Se realizan cinco repeticiones, obteniéndose el mejor término mostrado en la Tabla 18, mientras que la Tabla 19 presenta los coeficiente de correlación para cada variable, donde la simulación de este se puede observar en la Figura 44.

Tabla 18 Modelo Obtenido al Buscar los Términos de Desorción y Adsorción.

modelo	*Fitness*
$\frac{\theta_O}{\theta_{CO}} - log(log(\theta_{CO} + \theta_O) + sin(\theta_O) + 9.2)$	0.028

Tabla 19 coeficiente de correlación obtenido al Buscar los Términos de Desorción y Adsorción.

Variable	Coeficiente de correlación
θ_{Co}	0.996
θ_O	0.999

p_{CO_2}	0.984

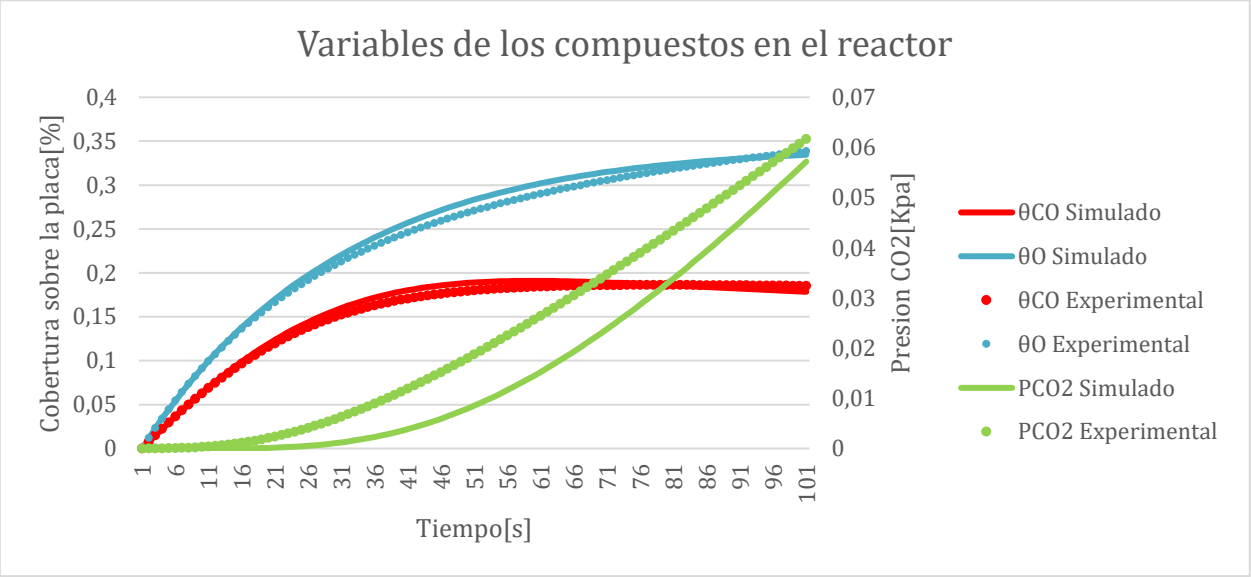

Figura 44 Simulación del Modelo Obtenido para la Adsorción y Desorción del CO.

Desde un punto de vista gráfico sumado a los valores obtenido para los coeficiente de correlacion, la solución logra representar el fenómeno desconocido relacionado con la adsorción y desorción del oxígeno, donde se reflejan todos los comportamientos de los datos. En primer lugar, se posee un aumento exponencial del cubrimiento de los gases en la placa hasta llegar a un estado estacionario. En segundo lugar, la presión del CO2 aumenta de manera constante.

Se puede observar que el algoritmo logró deducir que la parte faltante del modelo corresponde a una diferencia entre dos términos, ambos dependientes de la concentración de los gases presente en el catalizador. Más aún, el primero de ellos es similar al efecto de absorción donde, mientras mayor sea la presencia de monóxido de carbono en la placa, menos espacios tendrá el oxígeno para acoplarse, reduciendo el aumento de este último. En cambio, el segundo término refleja un efecto similar a la desorción donde, mientras mayor sea la concentración de gases en la placa, mayor será la posibilidad de que el oxígeno se libere sin reaccionar aumentando la disminución de este.

Aun cuando el análisis gráfico y fenomenológico entregan razones por la que el término propuesto podría ser considerado de buena calidad, un estudio matemático muestra que el modelo se indefine al inicio, donde la presencia de gases en la placa es 0, por lo que se debe evaluar tanto un cociente con valor de 0 en el denominador y numerador, además de un logaritmo con valor 0.

Por último, es necesario destacar que los valores de *fitness* obtenidos para los mejores modelos son de orden de magnitud menor que los obtenidos para los modelos anteriores. Esto se debe a que este valor se basa en el cálculo del error cuadrático medio, donde los resultados dependen del orden de magnitud de los datos empleados. Por lo anterior, no es posible especificar si un resultado es aceptable basándose solo en el valor del *fitness*.

4.4 Resumen resultados

A continuación, se presenta una síntesis de los resultados más importante y las discusiones más relevantes del Capítulo 4. Primero se muestra un cuadro de síntesis de los mejores diseños obtenidos para cada modelo, seguido de las discusiones pertinentes, para finalizar con las recomendaciones que se deben tener en cuenta al utilizar la metodología propuesta.

En la Tabla 20, la primera columna corresponde al modelo original a partir del cual se generaron los datos, seguido de la información que se desea obtener del algoritmo. Finalmente, el diseño propuesto para el conocimiento deseado, tras la utilización de la metodología a partir de los datos. Para facilitar la lectura se resaltan de color rojo los términos del modelo que se consideran desconocidos.

Tabla 20 Resumen de los mejores resultados obtenidos para cada modelo

Modelo original	Término diseñado por PG	Término propuesto
$\frac{dh}{dt} = 0.2 - 0.08 * h$	$\frac{dh}{dt} = 0.3 - 0.13 * h$	Cambio de altura del líquido a través del tiempo
$\frac{dx}{dt} = 0.1 * x - 0.02 * x * y$ $\frac{dy}{dt} = 0.04 * x * y - 0.4 * y$	$\frac{dx}{dt} = 0.118 * x - 0.02 * y * x$	Cambio de presas a través del tiempo

$\frac{d\theta_{Co}}{dt} = 0.04 * S_{CO}(\theta_{Co}, \theta_O, T) - 0.015 * \theta_{CO} * e^{3.97*\theta_{CO}} - 0.03 * \theta_{CO} * \theta_O$ $\frac{d\theta_O}{dt} = 0.012 * (1 - \theta_{Co} - \theta_O)^2 - 0.03 * \theta_{CO} * \theta_O$ $\frac{dp_{CO_2}}{dt} = 0.5 * 0.03 * \theta_{CO} * \theta_O$	$\upsilon = 0.0022$	Velocidad de reacción
$\frac{d\theta_{Co}}{dt} = 0.04 * S_{CO}(\theta_{Co}, \theta_O, T) - 0.015 * \theta_{CO} * e^{3.97*\theta_{CO}} - 0.03 * \theta_{CO} * \theta_O$ $\frac{d\theta_O}{dt} = 0.012 * (1 - \theta_{Co} - \theta_O)^2 - 0.03 * \theta_{CO} * \theta_O$ $\frac{dp_{CO_2}}{dt} = 0.5 * 0.03 * \theta_{CO} * \theta_O$	$\frac{\theta_O}{\theta_{CO}}$ $-log(log(\theta_{CO} + \theta_O) + \theta_O + 9.2)$	Adsorción y desorción del monóxido-

Los resultados observados fueron obtenidos tras realizar dos iteraciones de la metodología, donde cada una consiste en los siguientes pasos:

1) Análisis de los datos e información recolectada.

2) Realizar 5 repeticiones del algoritmo de programación genética.

3) Analizar los resultados, donde aparte de observar si los resultados son aceptables se espera extraer información para mejorar futuras iteraciones.

Se puede observar de la tabla, que, con excepción de los últimos dos casos, los modelos obtenidos representan la forma característica del fenómeno en estudio, además entregan parámetros que se acercan a los reales. En cambio, para el caso del tercer modelo que corresponde a la reacción de oxidación del monóxido de carbono, se realizaron dos estudios donde los términos a descubrir son distintos, primero se busca el término de la velocidad de reacción y en el segundo se desea proponer el término de la adsorción y desorción del CO. En ambos casos mencionados, se puede observar que los términos propuestos se alejan de los deseados y más aún, para el segundo caso dado que las variables parten con un valor de 0 es matemáticamente incorrecto.

A pesar de lo anterior los términos propuestos sirven para mejorar la siguiente interacción, en el primer caso se obtiene una reacción de orden 0, cuando en realidad depende de dos variables. A pesar de esto, el valor para la velocidad de reacción entregado por el algoritmo es valor similar al original, una vez se alcanza el estado estacionario, por lo que se debe explicitar que los resultados utilicen estas variables.

Del resultado obtenido en el segundo caso se puede extraer que la parte faltante del modelo corresponde a una diferencia entre dos términos que tienen comportamiento similar a la adsorción y desorción, donde el primero de ellos aumenta mientras menor sea la concentración de los gases y el segundo es lo opuesto.

Finalmente es necesario mencionar que los resultados mostrados se obtuvieron asumiendo que el conocimiento es mínimo, demostrando la capacidad de diseñar modelos complejos, por otra parte, la cantidad de datos utilizados supera la que normalmente se logra conseguir a partir de experimentos.

Por último, a continuación, se enumeran las consideraciones que se deben tener en cuenta al utilizar la metodología propuesta:

1) Se debe ajustar el tiempo máximo permitido para el proceso de integrar analíticamente la solución entregada por los individuos, de manera contrario se posee un mayor coste en el tiempo de ejecución o las mejores soluciones no alcanzan a ser evaluadas.

2) Se aconseja que en la primera iteración no se agreguen condiciones extras, para observar el comportamiento del algoritmo con los datos, como, asegurarse que el algoritmo proponga soluciones que represente todos los estados del fenómeno.

3) No basarse solo en el valor del *fitness* como medida de la calidad del resultado, ya que dependiendo del método en que se calcule y los datos utilizados el rango de este valor puede variar en orden de magnitud. De forma similar el coeficiente de correlación tampoco como único criterio de elección

Capítulo 5

5 Conclusiones

La obtención de modelos a partir de datos experimentales utilizando técnicas de *Machine Learning*, no es solo un tema de interés en los últimos años, sino que se ha aplicado de diversas formas en la resolución de problemas. A pesar de esto, no existe una guía universal que permita la utilización de estas herramientas a personas con un conocimiento básico del área de programación, provocando un gasto de tiempo para aprender las habilidades mínimas para utilizarlas. Por lo que poseer una metodología que oriente a los usuarios interesados en aplicar estas herramientas en sus respectivas áreas, pero que no posean amplias habilidades para programar se transforma en una necesidad para facilitar su uso, aumentar la efectividad y reducir los tiempos.

El objetivo del trabajo de memoria fue diseñar una metodología capaz de proponer modelos químicos, ya sea en su totalidad o solo una parte de estos, a través del algoritmo de programación genética. La elección del algoritmo se realizó con base en un estudio previo acerca de los diferentes enfoques en que se clasifica el *Machine Learning,* donde el aprendizaje supervisado, en específico el modelo de regresión simbólica, es el escogido por el tipo de datos que se utiliza normalmente en los sistemas químicos. El algoritmo, por su parte, se escoge a partir de una revisión bibliográfica, donde se encontraron diversas aplicaciones en el área de interés, entre las que se seleccionó la programación genética. Finalmente, para ampliar el rango de modelos que pueden ser resueltos se adapta el algoritmo para trabajar con modelos diferenciales.

A partir de la literatura especializada, fue posible obtener una amplia gama de información a revisar, obteniendo los diferentes parámetros que utiliza el algoritmo. Con base en esto, se llegó a la conclusión que se deben destacar los siguientes parámetros: el número de individuos y generaciones, los terminales, la extensión de la solución, las funciones a utilizar y la función de *fitness.* Lo anterior se fundamenta en la importancia de estos parámetros al momento de implementar el algoritmo. Sumado a esto, también se determinó que los principales problemas en este tipo de algoritmo, corresponde al bloat y la sobreestimación.

La estrategia de trabajo utilizada en este trabajo de memoria consistió en diseñar la metodología, la cual puede ser simplificada en tres partes. La primera de ellas se asocia con la recolección de información y estudio de los datos relacionados con el fenómeno de interés. La segunda parte de la

metodología corresponde a la implementación del algoritmo para obtener propuestas para el modelo. Finalmente, la metodología propuesta se enfocó en la realización de un análisis de los resultados obtenido a partir del algoritmo con el objetivo de mejorar la calidad de la siguiente iteración.

Producto de estos pasos, se diseñaron nuevas características que fueron agregadas al algoritmo para obtener una mejora en la calidad del resultado y su ejecución. Las características seleccionadas fueron el *snipping*, el límite en el número de nodos, el ajuste de parámetros, el uso de bloques conocidos del modelo y la utilización del criterio de información de Akaike como medida del *fitness*.

Para evaluar la utilidad de la metodología, se aplicó en la recuperación de tres modelos predefinidos de diferente complejidad: Altura de líquido en un estanque, modelo de Lokta-Volterra y la reacción de oxidación del monóxido de carbono sobre una placa de platino. Esto se realizó a partir de datos que se obtuvieron de la simulación de dichos modelos y se asumió un conocimiento previo mínimo de los fenómenos.

A partir de los resultados obtenidos y las discusiones realizadas, se pudo concluir de manera general que, en primer lugar, es necesario realizar una elección adecuada de los parámetros para evitar un consumo de recursos mayor al necesario; más aún puede orientar de manera errada la solución. Por ejemplo, los nodos internos están compuestos por funciones protegidas, las cuales representan expresiones matemáticas. Las funciones protegidas tienen la cualidad de que deben devolver un valor en todo momento, lo que exige un análisis matemático del resultado, ya que puede haber casos en que el modelo matemático obtenido sea indeterminado si se simula con los datos experimentales, lo que requiere un análisis sobre el comportamiento que representa, causando un análisis sobre el efecto que se desea representar.

En segundo lugar, se concluye de manera general que no es posible definir un valor por el cual las soluciones entregadas por el algoritmo sean aceptables, ya que el valor del fitness se calcula con base en medidores estadísticos, cuya magnitud depende de la magnitud de los valores de las variables. Esta conclusión es importante, puesto que el algoritmo por sí solo utiliza el fitness para comparar propuestas de modelos y entregar la mejor de ellas, sin comprobar si se logran representar las cualidades del fenómeno de interés.

Finalmente, se concluye que las características diseñadas afectaron el resultado obtenido en tres casos específicos:

1. El fitness de la solución, donde la característica del ajuste de parámetros y el uso de bloque conocidos del modelo son las modificaciones que entregan la principal mejora.

2. La complejidad de la solución, donde los mejores resultados fueron obtenidos al utilizar las características del *snipping*, el criterio de información de Akaike (ACII) y la limitación del número de nodos.

3. El tiempo de ejecución se vio afectado de manera positiva por el *snipping*, el uso de bloques conocidos del modelo y la limitación del número de nodos. Paralelamente, se descubrió que el ajuste de parámetros causó un aumento desmedido del tiempo, llegando a incrementarse hasta un 300%.

En relación con los modelos diseñados se obtuvo que, para el primer modelo presentado en este trabajo de memoria, al utilizar 100 individuos con 10 generaciones y dos iteraciones, se logró diseñar un modelo similar al original con diferencia solo en los valores de los parámetros. Para el segundo modelo, el trabajo siguió un efecto similar con la diferencia que para este caso se utilizaron 200 individuos con 20 generaciones. Para el último modelo, no se logró obtener un diseño similar al original, sin embargo, se obtuvo información importante del comportamiento del fenómeno. Sumado a lo anterior, uno de los hallazgos en esta investigación corresponde a que, dada la naturaleza estocástica del algoritmo, es posible que se terminen integrando modelos con ecuaciones *stiff*, las cuales requieren un tiempo de integración excesivamente mayor a lo normal además de no entregar información relevante en la mayoría de los casos. Por esto es necesario limitar el tiempo que se emplea al momento de integrar el modelo propuesto en el algoritmo.

Con base en los resultados y sus discusiones fue posible demostrar que el algoritmo sirve para proponer modelos que cuentan con elementos desconocidos parcialmente o en su totalidad, tanto para datos con error como sin error. Junto con esto, los pasos de los análisis propuestos en la metodología permiten diseñar un modelo cercano a lo deseado u obtener información a partir de los resultados de manera tal que en la siguiente iteración se obtengan resultados de mayor calidad.

Una de las principales ventajas que presenta la metodología propuesta es que no se requiere conocimiento previo de los términos del modelo que se quiere diseñar. Al ser estocástico el algoritmo se asegura que, si se entregan suficientes individuos y generaciones, se logrará encontrar la solución indicada.

Las principales limitaciones en el desarrollo de la memoria recaen en que la implementación computacional utilizada solo puede proponer una ecuación a la vez. Lo anterior significa que, si el fenómeno a investigar corresponde a un sistema de ecuaciones de dos o más, es necesario poseer la información necesaria de manera que el algoritmo deba proponer solo una de las ecuaciones a la vez. Junto con esto, también se encuentran problemas ligados

al *bloat*, el cual se vio reducido en este trabajo al implementar un *snipping* simple que une parámetros que se encuentran cercanos entre ellos en el árbol binario que representa el modelo propuesto.

Finalmente, es importante destacar que, aunque la metodología y modificaciones realizadas están enfocadas para ser utilizadas en el ámbito químico, estas pueden ser aplicadas en otras áreas, incluso en otros algoritmos. En futuras investigaciones podría complementarse el trabajo realizado en esta memoria a partir de modificaciones en la implementación computacional, de manera que sea posible que el algoritmo proponga modelos por dos o incluso más ecuaciones.

6 Bibliografía

[1] G. Allaire and A. Craig, *Numerical Analysis and Optimization: An Introduction to Mathematical Modelling and Numerical Simulation*. 2007.

[2] G. Marion, S. Scotland, D. Lawson, and G. Marion, "An Introduction to Mathematical Modelling," 2008.

[3] P. Frejd, *Modes of Mathematical Modelling : An analysis of how modelling is used and interpreted in and out of school settings*, no. 181. 2014.

[4] G. Franceschini and S. Macchietto, "Model-based design of experiments for parameter precision: State of the art," *Chem. Eng. Sci.*, vol. 63, no. 19, pp. 4846–4872, 2008.

[5] A. goleman, daniel; boyatzis, Richard; Mckee, *Hands-on Machine Learning*, vol. 53, no. 9. 2019.

[6] G. Yang, X. Li, J. Wang, L. Lian, and T. Ma, "Modeling oil production based on symbolic regression," *Energy Policy*, vol. 82, no. 1, pp. 48–61, 2015.

[7] A. Colins, Z. P. Gerdtzen, M. T. Nuñez, and J. C. Salgado, "Mathematical modeling of intestinal iron absorption using genetic programming," *PLoS One*, vol. 12, no. 1, pp. 1–24, 2017.

[8] M. Paulinas and A. Ušinskas, "A survey of genetic algorithms applications for image enhancement and segmentation," *Inf. Technol. Control*, vol. 36, no. 3, pp. 278–284, 2007.

[9] H. V. Hultmann Ayala and L. Dos Santos Coelho, "Tuning of PID controller based on a multiobjective genetic algorithm applied to a robotic manipulator," *Expert Syst. Appl.*, vol. 39, no. 10, pp. 8968–8974, 2012.

[10] E. A. Bender, *An Introduction to Mathematical Modeling*. 2000.

[11] D. R. Tobergte and S. Curtis, *Theory of Heat Transfer with Forced convection Film flows*, vol. 53, no. 9. 2013.

[12] I. Tw, "Free convection boundary-layer flow o v e r a v e r t i c a l flat p l a t e," 1961.

[13] R. Frigg and S. Hartmann, "Models in Science (Stanford Encyclopedia of Philosophy/Spring 2020 Edition)," *Metaphysics Research Lab, Stanford University*, 2020. [Online]. Available: https://plato.stanford.edu/archives/spr2020/entries/models-science/. [Accessed: 25-Oct-2020].

[14] J. Valadi and P. Siarry, *Applications of metaheuristics in process engineering*, vol. 9783319065. 2014.

[15] K. C. Furman and N. V. Sahinidis, "A critical review and annotated bibliography for heat exchanger network synthesis in the 20th century," *Ind. Eng. Chem. Res.*, vol. 41, no. 10, pp. 2335–2370, 2002.

[16] N. Z. W. Verheyen, "Design of flexible heat exchanger network for multi-period operation," *Chem. Eng. Sci.*, vol. 61, no. 23, pp. 7730–7753, 2006.

[17] O. Pérez, L. Zumalacárregui, and O. Gozá, "Simplificaciones en el Cálculo de Columnas de Destilación Alcohólica," *Inf. Tecnol.*, vol. 21, no. 6, pp. 103–112, 2010.

[18] L. A. Rodríguez Umaña, "Modelo del control de nivel y caudal de líquido en un depósito cilíndrico usando la herramienta Simulink de Matlab," *Dialnet*, vol. 9, no. 1, pp. 7–8, 2013.

[19] E. Systems, “NEW PROBLEMS OF INSTRUMENTATION DESIGN AND Institute of Electronic Systems , 2 Institute of Radioelectronics,” no. September 2000, 2014.

[20] N. Bacaër, *A short history of mathematical population dynamics*. 2011.

[21] I. R. Epstein, J. A. Pojman, and G. Nicolis, *An Introduction to Nonlinear Chemical Dynamics: Oscillations, Waves, Patterns, and Chaos*, vol. 52, no. 11. 1999.

[22] W. F. Ramirez, *Computational Methods in Process Simulation*, 2nd Editio., vol. c. 1997.

[23] T. Engel and G. Ertl, “Elementary Steps in the Catalytic Oxidation of Carbon Monoxide on Platinum Metals,” *Adv. Catal.*, vol. 28, no. C, pp. 1–78, 1979.

[24] J. R. Crelghton, F.-H. Tseng, J. M. White, and J. S. Turner, “Numerical Modeling of Steady-State Carbon Monoxide Oxidation on Pt and Pd,” *Encycl. Earth Sci. Ser.*, vol. 14, no. 9, 1980.

[25] J. P. Reyes Vásquez, I. Altamirano Zanipatin, D. S. Aldás Salazar, L. A. Morales Perrazo, and C. R. Reyes Vásquez, “Modelo de planeación y programación de la producción para el troquelado de cuero en la industria de calzado,” *Rev. Ingeneria Ind.*, vol. 16, no. 3, pp. 233–249, 2017.

[26] L. M. Martínez Parra, “ASIGNACION DE HORARIOS DE TRABAJO, RESOLUCIÓN MEDIANTE ALGORITMO DE BUSQUEDA TABÚ,” 2013.

[27] Y. Ding, L. Chen, and K. Hao, *Nature-Inspired Optimization Algorithms*, vol. 118. 2018.

[28] D. H. Wolpert and W. G. Macready, “No free lunch theorems for optimization,” *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, 1997.

[29] E. M. Golafshani, “Introduction of Biogeography-Based Programming as a new algorithm for solving problems,” *Appl. Math. Comput.*, vol. 270, pp. 1–12, 2015.

[30] V. Nasteski, “An overview of the supervised machine learning methods,” *Horizons.B*, vol. 4, no. December 2017, pp. 51–62, 2017.

[31] M. Khanum, T. Mahboob, W. Imtiaz, H. Abdul Ghafoor, and R. Sehar, “A Survey on Unsupervised Machine Learning Algorithms for Automation, Classification and Maintenance,” *Int. J. Comput. Appl.*, vol. 119, no. 13, pp. 34–39, 2015.

[32] “Introducing Voyage Deepdrive - Voyage.” [Online]. Available: https://news.voyage.auto/introducing-voyage-deepdrive-69b3cf0f0be6. [Accessed: 07-May-2020].

[33] O. Vinyals *et al.*, “Grandmaster level in StarCraft II using multi-agent reinforcement learning,” *Nature*, vol. 575, no. 7782, pp. 350–354, Nov. 2019.

[34] D. Silver *et al.*, “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[35] “Waymo’s first external fundraising round just grew to $3 billion - The Verge.” [Online]. Available: https://www.theverge.com/2020/5/12/21256082/waymo-external-investment-extension-3-billion-self-driving. [Accessed: 12-May-2020].

[36] V. Veloso de Melo and W. Banzhaf, “Automatic feature engineering for regression models with machine learning: An evolutionary computation and statistics hybrid,” *Inf. Sci. (Ny).*, vol. 430–431, no. November, pp. 287–313, 2018.

[37] M. Quade, M. Abel, K. Shafi, R. K. Niven, and B. R. Noack, “Prediction of dynamical systems by symbolic regression,” *Phys. Rev. E*, vol. 94, no. 1, pp. 1–15, 2016.

[38] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, 1989.

[39] M. Z. M. Kamali, N. Kumaresan, and K. Ratnavelu, “Solving differential equations with ant colony programming,” *Appl. Math. Model.*, vol. 39, no. 10–11, pp. 3150–3163, 2015.

[40] D. Karaboga, C. Ozturk, N. Karaboga, and B. Gorkemli, “Artificial bee colony programming for symbolic regression,” *Inf. Sci. (Ny).*, vol. 209, pp. 1–15, 2012.

[41] H. Vaddireddy and O. San, “Equation Discovery Using Fast Function Extraction: a Deterministic Symbolic Regression Approach,” *Fluids*, vol. 4, no. 2, p. 111, 2019.

[42] R. Poli, W. B. Langdon, N. F. McPhee, and J. R. Koza, *Field Guide to Genetic Programing*, no. March. 2008.

[43] J. T. Alander, “On optimal population size of genetic algorithms,” *Proceedings - Computer Systems and Software Engineering: 6th Annual European Computer Conference, CompEuro 1992*. pp. 65–70, 1992.

[44] D. Vrajitoru, “Large Population or Many Generations for Genetic Algorithms? Implications in Information Retrieval,” no. September, pp. 199–222, 2000.

[45] J. Sadeghi and S. T. A. Niaki, “Two parameter tuned multi-objective evolutionary algorithms for a bi-objective vendor managed inventory model with trapezoidal fuzzy demand,” *Appl. Soft Comput. J.*, vol. 30, pp. 567–576, 2015.

[46] D. Gupta and S. Ghafir, “An Overview of methods maintaining Diversity in Genetic Algorithms,” *Int. J. Emerg. Technol. Adv. Eng.*, vol. 2, no. 5, pp. 56–60, 2012.

[47] E. Nicoară, “Mechanisms to avoid the premature convergence of genetic algorithms,” *Bul. Univ. Pet. Gaze din Ploiesti, Mat.*, vol. LXI, no. 1, pp. 87–96, 2009.

[48] A. Piszcz and T. Soule, “Genetic programming: Optimal population sizes for varying complexity problems,” *GECCO 2006 - Genet. Evol. Comput. Conf.*, vol. 1, pp. 953–954, 2006.

[49] P. J. Angeline, *Genetic programming: On the programming of computers by means of natural selection,* vol. 33, no. 1. 1994.

[50] C. R. S. Banerji, T. Mansour, and S. Severini, “A notion of graph likelihood and an infinite monkey theorem,” *J. Phys. A Math. Theor.*, vol. 47, no. 3, 2014.

[51] J. Bongard and H. Lipson, “Automated reverse engineering of nonlinear dynamical systems,” *Proc. Natl. Acad. Sci.*, vol. 104, no. 24, pp. 9943–9948, 2007.

[52] D. Jakubovitz, R. Giryes, and M. R. D. Rodrigues, “Generalization Error in Deep Learning,” *Appl. Numer. Harmon. Anal.*, pp. 153–193, 2019.

[53] J. Bacardit and J. M. Garrel, “Bloat control and generalization pressure using the minimum description length principle for a Pittsburgh approach Learning Classifier System,” vol. 4399/2007, no. January 2005, 2007.

[54] A. Purohit and N. S. Choudhari, “Code Bloat Problem in Genetic Programming,” vol. 3, no. 4, pp. 1–5, 2013.

[55] S. Silva and J. Almeida, “GPLAB: A Genetic Programming Toolbox for MATLAB,” *ECOS - Evol. Complex Syst. Gr.*, vol. Version 3, no. April, pp. 1–73, 2007.

[56] “Laboratorio de Computación Distribuida y Modelamiento de Procesos (PMDC Lab) - Facultad de Ciencias Físicas y Matemáticas - Universidad de Chile.” [Online]. Available: http://ingenieria.uchile.cl/investigacion/presentacion/laboratorios/departamento-de-ingenieria-quimica-y-biotecnologia/90972/laboratorio-de-computacion-distribuida-y-modelamiento-de-procesos. [Accessed: 14-Apr-2020].

[57] A. Colins, “DETERMINACIÓN EXPERIMENTAL Y MODELACIÓN MATEMÁTICA DE LOS FLUJOS DE TRANSPORTE DE HIERRO EN CÉLULAS CACO-2,” Universidad de Chile, 2015.

[58] B. H., “Akaikes Information Criterion and Recent Developments in Information Complexity,” *J. Math. Psychol.*, vol. 44, no. 1, pp. 62–91, 2000.

[59] C.-L. Hurvich, Clifford M.; Tsai, “Biometrika Trust Regression and Time Series Model Selection in Small Samples Author (s): Clifford M . Hurvich and Chih-Ling Tsai Published by : Oxford University Press on behalf of Biometrika Trust Stable URL : http://www.jstor.org/stable/2336663 REFERE,” *Biometrika*, vol. 76, no. 2, pp. 297–307, 1989.

[60] W. Banzhaf, *Artificial Intelligence: Genetic Programming*, Second Edi., vol. 2. Elsevier, 2015.

[61] L. Hilje, “Simbiosis: Consideraciones terminológicas y evolutivas,” *Uniciencia*, vol. 1, no. 1, pp. 57–60, 2016.

[62] R. K. Herz and S. P. Marin, “Surface chemistry models of carbon monoxide oxidation on supported platinum catalysts,” *J. Catal.*, vol. 65, no. 2, pp. 281–296, 1980.