UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN

Uso de Algoritmos Genéticos y Redes Neuronales para Control Predictivo en un Reactor Tipo CSTR

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO QUÍMICO

P R E S E N T A :

HÉCTOR CARLOS ARANDA MARTÍNEZ

DIRECTOR DE TESIS:
DR. RICARDO PARAMONT HERNÁNDEZ GARCÍA

CUAUTITLÁN IZCALLI, EDO. DE MÉXICO
2016

VNIVERSIDAD NACIONAL AVTONOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN
UNIDAD DE ADMINISTRACIÓN ESCOLAR
DEPARTAMENTO DE EXÁMENES PROFESIONALES

ASUNTO: VOTO APROBATORIO

M. en C. JORGE ALFREDO CUÉLLAR ORDAZ
DIRECTOR DE LA FES CUAUTITLAN
PRESENTE

ATN: M. EN A. ISMAEL HERNÁNDEZ MAURICIO
Jefe del Departamento de Exámenes Profesionales
de la FES Cuautitlán.

Con base en el Reglamento General de Exámenes, y la Dirección de la Facultad, nos permitimos comunicar a usted que revisamos el: **Trabajo de Tesis**

Uso de Algoritmos Genéticos y Redes Neuronales para Control Predictivo en un Reactor Tipo CSTR.

Que presenta el pasante: **Héctor Carlos Aranda Martínez**
Con número de cuenta: **412001882** para obtener el Título de la carrera: **Ingeniería Química**

Considerando que dicho trabajo reúne los requisitos necesarios para ser discutido en el **EXAMEN PROFESIONAL** correspondiente, otorgamos nuestro **VOTO APROBATORIO**.

ATENTAMENTE
"POR MI RAZA HABLARÁ EL ESPÍRITU"
Cuautitlán Izcalli, Méx. a 13 de Abril de 2016.

PROFESORES QUE INTEGRAN EL JURADO

	NOMBRE	FIRMA
PRESIDENTE	Dr. Ricardo Paramont Hernández García	
VOCAL	M. en C. Gilberto Atilano Amaya Ventura	
SECRETARIO	Dra. Abigail Martínez Estrada	
1er. SUPLENTE	M. en E. María Teresa Ylizaliturri Gómez Palacio	
2do. SUPLENTE	Dr. Martín Rogelio Cruz Díaz	

NOTA: los sinodales suplentes están obligados a presentarse el día y hora del Examen Profesional (art. 127).

IHM/cga

Dedicatorias

A mamá y papá, quienes han sabido guiarme e impulsar mis más grandes ambiciones.
Me han dado todo para ser quien soy y estar donde estoy.

Agradecimientos

A Dios, porque desde donde quiera que se encuentre no me ha dejado y ha guiado cada uno de mis pasos.

A mi madre, por ser uno de los pilares de mis logros. Gracias por siempre estar a mi lado en cada momento, por jalarme las orejas cuando es necesario, por tantas enseñanzas y amor incondicional. Te estaré eternamente agradecido y siempre estaré orgulloso de ser tu hijo. Éste trabajo es para ti.

A mi padre, por ser mi mayor ejemplo a seguir. Me lleno de orgullo cuando dicen que me parezco a ti. Gracias por ser otro gran pilar en mi vida y nunca dejarme a la deriva. Todo el esfuerzo que has hecho por mi ahora ha dado un fruto más. Este trabajo también es para ti.

A mis hermanos, que han sido para mi una gran fuente de amor, apoyo y aprendizaje. Gracias por creer en mí y animarme a seguir mis sueños. Mis dos mejores amigos, gracias.

A Itza, por convertirse en la fuerza que impulsa mis acciones. Gracias por tu apoyo, tu amor, tus consejos, tus regaños y tu sonrisa; por estar a mi lado en todo momento. Gracias por hacer de estos años los mejores de mi vida.

A mis abuelos, mis tíos y mis primos. Gracias por todo su apoyo y su cariño incondicional, porque hemos sido una familia muy unida y eso ha enriquecido mi vida completa.

A mis amigos de carrera, en especial a Abraham, Alexys, Pepe, Ottmar, Fredy, Rosario, Andrea, Rocío. Gracias por compartir este tiempo conmigo, he aprendido tanto con ustedes y me he divertido tanto que, cuando me dí cuenta, ya habíamos terminado la carrera. Un exitoso futuro les espera.

A mis profesores, porque cumplieron con su labor de la manera más humilde y sincera. Gracias por compartir su conocimiento y experiencia.

A mi asesor, por mostrarme el maravilloso camino de la "Inteligencia Artificial" y así haber despertado en mí el deseo de aprender, que culminó en este trabajo.

A mi *alma mater*, que me otorgó las herramientas y el espacio para desarrollarme profesionalmente.

"La vida es muy simple, pero insistimos en hacerla complicada"
Confucio

Índice general

Objetivos

General

Diseñar un método que utilice una red neuronal artificial y un algoritmo genético para predecir los valores de los parámetros que controlarán de manera eficiente la operación de un reactor continuo perfectamente agitado (CSRT por sus siglas en inglés) ante una perturbación.

Particulares

- Demostrar que un algoritmo genético es capaz de encontrar parámetros de control de un CSTR que hagan que se comporte satisfactoriamente ante una perturbación dada.
- Entrenar una red neuronal para poder predecir los valores de los parámetros del controlador en dependencia con el tipo de perturbación.
- Demostrar que la unión de estos dos algoritmos puede controlar la operación de un CSTR pese a ser sometido a diferentes perturbaciones.

Capítulo 1

Introducción

En la actualidad las nuevas tecnologías están viendo la luz del mundo: celulares, automóviles, casas, electrodomésticos, por mencionar algunos. Lo que llama mas la atención es que todos tienen un nuevo adjetivo común: "inteligentes". Esto nos puede hacer referencia a que tal vez piensen por sí mismos y aprendan como lo hacemos nosotros los humanos, aunque la realidad sea otra. Entonces podemos plantear las siguientes preguntas: ¿Son estas tecnologías realmente inteligentes? ¿Realmente adquieren conocimientos que pueden comprender para usarlos posteriormente? ¿Pueden identificar relaciones entre las "cosas"? Aunque el adjetivo le brinda un valor agregado a las tecnologías, realmente no son inteligentes. Todavía no se ha encontrado un método para que las computadoras tengan esa inteligencia de la que hablamos. Según Roger Penrose [1] debe haber algo no computable en las neuronas que no permite que sean simuladas con las computadoras actuales y tiene que ver con fenómenos cuánticos dentro de la misma estructura de éstas.

Entonces el concepto de inteligencia que nos quieren vender no es el que pensamos; sin embargo ¿Qué es entonces eso por lo que se les llama "inteligentes"? A estos artículos se les programa de manera que parezcan inteligentes, por ejemplo: Cuando recibes una llamada en tu "smartphone" suena un timbre personalizado para la persona que te habla, aparece su imagen e inclusive le puede mandar un mensaje del por qué no puedes contestar con sólo apretar un lugar en la pantalla. Además te recuerda tus citas importantes y los cumpleaños de los demás. Te da el reporte meteorológico en cualquier lugar donde te encuentres y hasta los lugares donde podrías ir a comer, cenar, divertirte, y muchos otros más. Esto parece ser muy inteligente, pero el celular no hubiera podido hacer nada si no se le hubiera

dicho específicamente, paso por paso, lo que debía hacer. Lo mismo pasa con los automóviles que se conducen solos y las casas que se controlan por sí mismas. Cualquier error en la programación es suficiente para causar un pequeño error en la operación o un desastre.

Las redes neuronales artificiales (*Artificial Neural Networks* o ANN) son algoritmos clasificadores y ajustadores de funciones que se usan ampliamente para la detección de patrones y ajuste de funciones desconocidas. Estos algoritmos son capaces, teóricamente, de ajustar cualquier tipo de función [2]. Es por ello que se utilizará una de ellas en conjunto con un algoritmo genético para poder predecir los parámetros de los controladores que mantendrán estable un sistema dado.

Los algoritmos genéticos (*Genetic Algorithms* o GA) son algoritmos de búsqueda y optimización estocástica basados en la evolución por selección natural. Fueron desarrollados por John Holland, quien se basó en los sistemas naturales para resolver problemas de optimización [3]. Pueden encontrar los valores óptimos que minimizan o maximizan una función dentro de un campo de búsqueda delimitado. Estos algoritmos pueden manejar funciones de N variables, además de poder trabajar con funciones desconocidas, es decir que no son explícitas o son producto de un sistema muy complejo, como las simulaciones de equipos usados en Ingeniería Química.

Cambiando un poco el tema, todos dependemos de la industria química. Sin ella no podríamos vestirnos, comer, viajar, estudiar, divertirnos y hacer todo lo que hacemos en la actualidad. Desde la fabricación de plásticos hasta la de fármacos, todo esto es la industria química. Éstas utilizan equipos tan pequeños como una televisión o tan grandes como un edificio que se pueden controlar casi con un sólo dedo. Esto ha sido posible gracias a los avances en adquisición de datos e instrumentos que pueden ser controlados, ya sea hidráulica, neumática o eléctricamente. De esta manera procesos de alto riesgo, difíciles de controlar, de largos tiempos de operación pueden ser ahora operados desde un cuarto de control de donde se reciben los datos y con base en estos datos mandar las señales que permitan controlar el proceso. Esto, por supuesto, lo hacen los operadores, sin embargo ¿Podrá una máquina hacer el trabajo del operador? Si se le programa tal como piensa el operador entonces tendría una posibilidad, pero es casi imposible e impredecible conocer todas las acciones que el operador podría tomar ante ciertas circunstancias. Sin embargo una máquina podría hacer algunas tareas básicas como controlar la temperatura de un reactor o la presión en una línea de hidrógeno.

Para poder encontrar la manera ideal de controlar un proceso se pueden hacer experimentos en plantas piloto o equipos en operación, lo cual

genera gastos de mantenimiento, servicios y mano de obra, entre otros. Sin embargo, algo muy factible es simular los experimentos "*in silico*", es decir, en una computadora digital, haciendo gran cantidad de experimentos a un pequeño costo para después corroborar con datos experimentales generados en la operación del equipo. En la actualidad el poder de cómputo se ha incrementado tanto que podemos hacer complejas simulaciones de sistemas en estado transitorio, simulación dinámica, que nos dan más información de cómo se comporta el sistema y es una herramienta muy poderosa para la optimización de una planta [4]. Todos los modelos que se tratarán en este trabajo han sido desarrollados e implementados desde cero, lo que permite que se comprenda completamente lo que está pasando en el sistema y ayuda a identificar diferentes tipos de errores.

En la literatura podemos encontrar sistemas de control que utilizan los algoritmos genéticos y/o redes neuronales para el control de procesos. En este trabajo se diseñará un método que utilice un algoritmo genético para la búsqueda de los parámetros de control de un CSTR, parámetros que luego serán usados para entrenar una red neuronal que podrá predecir estos mismos ante una perturbación dada.

Capítulo 2

Modelación matemática

El hombre ha buscado siempre la explicación de todo fenómeno que ha observado, lo cual ha llevado a diferentes resultados. Sin embargo, de acuerdo con Ingham [4], la naturaleza tiene todo tipo de modelos cualitativos que no tienen una expresión matemática precisa. Estos son algo complejos de describir, aunque parece que nosotros los hemos comprendido muy bien y los hemos utilizado desde entonces. Por ejemplo podríamos hablar del amor, tal como lo hizo Hanna Fray en su libro "The mathematics of love" [5]. En él menciona que sería muy difícil definir al amor en una ecuación, pues se basa en diferentes tipos de comportamientos y conductas, los cuales podrían ser ajustados por un modelo *desconocido*. Esto tal vez es algo difícil de deducir, puesto que es algo a lo que estamos muy acostumbrados y no nos hemos detenido a pensar. Así como este ejemplo hay muchos, lo importante es que, en algunos casos, hemos encontrado una representación a ciertos modelos que no requiere más que sustituir algunos números y unos simples cálculos aritméticos.

En ingeniería química los modelos matemáticos son el pan de cada día. Desde las materias de primer semestre, hasta las del último, estudiamos diferentes modelos, cada uno con un propósito diferente y , algunos de entre ellos, aterradores a primera vista. Tal vez le faltó mencionar a algún profesor que debíamos encontrarle el gusto a trabajar con todo tipo de modelos, o simplemente no le tomamos la suficiente importancia.

Desde el simple modelo de la ecuación de la recta (que tal vez sea el más simple de todos), hasta algún modelo que prediga el comportamiento de un reactor de flujo pistón en donde exista convección, difusión y reacción (que se modela con maravillosas ecuaciones diferenciales parciales), todos son útiles para resolver problemas de ingeniería. Es por ello que en este trabajo se le

dará gran importancia al desarrollo del modelo a estudiar.

Los reactores químicos son uno de los motivos principales de la existencia de la Ingeniería Química. En toda industria de transformación química podemos encontrar un reactor. Existen de diferentes tipos y tamaños de ellos, con base en la función que tienen. Todos ellos son importantes y tienen sus ventajas y desventajas, pero en esta sección trataremos sólo con el modelado de un reactor tipo CSTR.

2.1. Problema

El problema será muy similar al del ejemplo 8-8 de *"Elements of Chemical Reactions"* 4ª Ed [6], además que ya se ha estudiado este sistema anteriormente por Furusawa *et al* [7].

Se tiene la siguiente reacción:

$$CH_3-CH-CH_2-O + H_2O \xrightarrow{H_2SO_4} CH_2-CH-CH_3-(OH)_2$$

Se desea llevar a cabo la reacción de hidrólisis del óxido de propileno, el cual es un compuesto muy volátil (su punto de ebullición en estado puro es de 34°C), para producir propilenglicol. La reacción es exotérmica de pseudo-primer órden, ya que el agua se encuentra en exceso. Para llevar a cabo la reacción, se mezcla la corriente de óxido de propileno (F_1) con la corriente de metanol (F_2) en proporciones iguales ($F_1 = F_2$). El metanol permite que la mezcla entre óxido de propileno y agua sea homogénea, de lo contrario se formarían 2 fases. En seguida se une esta nueva corriente con la corriente de agua que contiene 0.1 % en masa (W %) de ácido sulfúrico (F_3) para catalizar la reacción. La mezcla genera calor debido a los calores de mezclado de los componentes, el cual es ΔH_{dil}. Se hace ingresar la nueva mezcla directamente al reactor que cuenta con un sistema de enfriamiento por chaqueta. En la chaqueta entra agua de enfriamiento (F_A) a una temperatura $T_{A,e}$ y sale a $T_{A,s}$. El reactor cuenta con una superficie extendida para el intercambio de calor con la chaqueta. La corriente de agua de enfriamiento está controlada por un controlador de temperatura que mide la temperatura dentro del reactor. La temperatura de la reacción no debe de exceder nunca los 50°C, pues si sobrepasa esa temperatura se perderá demasiado óxido de propileno.

Se puede apreciar en la figura 2.1 un esquema del equipo. Se desea obtener la mayor cantidad de óxido de propileno sin sobrepasar la temperatura de 50°C. Es por esto que se utilizará un controlador de temperatura, el cual mantendrá las condiciones adecuadas para el funcionamiento correcto del equipo.

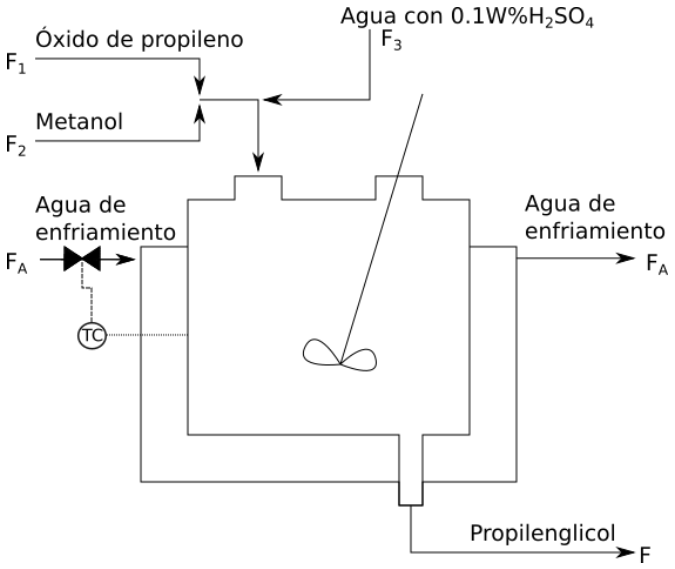

Figura 2.1: Esquema del reactor a estudiar

2.2. Balance de materia

Con base en la figura 2.1 se formula primero el balance del óxido de propileno, ya que es el reactivo que rige la cinética de la reacción. No es necesario hacer el balance global de materia, ya que que el volumen, la densidad y el flujo volumétrico de las entradas serán considerados constantes. Entonces podemos considerar:

$$F_1 + F_2 + F_3 \equiv F \tag{2.1}$$

Ya que se tienen ecuaciones diferenciales, es necesario establecer condiciones iniciales. Se propone entonces la siguiente condición inicial:

$$C_{OP,0} = C_{OP}(t = 0) = C_{OP,e} \tag{2.2}$$

En donde $C_{OP,e}$ es la concentración de óxido de propileno a la entrada.

Se desarrolla el balance de materia:

$$\begin{aligned}
\frac{d}{dt}(VC_{OP}) &= FC_{OP,0} - FC_{OP} - kVC_{OP} \\
V\frac{dC_{OP}}{dt} &= FC_{OP,0} - C_{OP}(F + kV) \qquad (2.3) \\
\text{Sustituyendo } C_{OP} &= C_{OP,0}(1 - \bar{x}_{OP}) \\
VC_{OP,0}\frac{d}{dt}(1 - \bar{x}_{OP}) &= FC_{OP,0} - C_{OP,0}(1 - \bar{x}_{OP})(F + kV) \\
-\frac{d\bar{x}_{OP}}{dt} &= \frac{F}{V} - (1 - \bar{x}_{OP})\left(\frac{F}{V} + k\right) \\
\frac{d\bar{x}_{OP}}{dt} &= (1 - \bar{x}_{OP})\left(\tau^{-1} + k\right) - \tau^{-1} \\
\frac{d\bar{x}_{OP}}{dt} &= k(1 - \bar{x}_{OP}) - \frac{\bar{x}_{OP}}{\tau} \qquad (2.4)
\end{aligned}$$

En donde C_{OP}, $\bar{x}_{OP}$, ρ y V son la concentración del óxido de propileno en el reactor, la concentración molar de óxido de propileno en el reactor, la densidad promedio de la mezcla reaccionante y el volumen del reactor, respectivamente. Se utilizará la ecuación de Arrhenius, la cual tiene la siguiente forma:

$$k = k_0 e^{-\frac{E_A}{RT}} \qquad (2.5)$$

En donde

k es la constante cinética de reacción

k_0 es el factor preexponencial de Arrhenius

E_A es la energía de activación de la reacción

R es la constante de los gases ideales

También se utilizará el tiempo de residencia, es cual es obtenido con la siguiente ecuación:

$$\tau = \frac{V}{F} \qquad (2.6)$$

Además de la conversión del óxido de propileno, nos interesa saber cuánto propilenglicol se está generando. El balance dinámico del propilenglicol (PG) es:

$$\begin{aligned}
V\frac{dC_{PG}}{dt} &= -FC_{PG} + VkC_{OP} \\
\frac{dC_{PG}}{dt} &= kC_{OP} - \frac{F}{V}C_{PG} \qquad (2.7)
\end{aligned}$$

Las ecuaciones del balance de materia escritas en notación funcional, son:

$$\frac{d\bar{x}}{dt} = f_1(T, \bar{x}) \tag{2.8}$$

$$\frac{dC_{PG}}{dt} = f_2(T, \bar{x}, C_{PG}) \tag{2.9}$$

2.3. Balance de energía

Para el balance de energía se considerará que la temperatura de referencia es $T_{ref} = 0^oC$ y que el agua de enfriamiento de la chaqueta se encuentra perfectamente agitada, por lo que tiene una sola temperatura. El balance para el reactor es el siguiente:

$$\begin{aligned}\frac{d}{dt}(V\rho CpT) = F\rho Cp(T_e - T) - \Delta H_R kVC_{OP} - \Delta H_{dil} F\rho \\ + UA(T_A - T)\end{aligned}$$

$$\frac{dT}{dt} = \frac{(T_e - T)}{\tau} - \frac{\Delta H_{dil}}{\tau Cp} + \frac{UA(T_A - T) - \Delta H_R kVC_{OP}}{V\rho Cp} \tag{2.10}$$

En donde

Cp es la capacidad calorífica promedio de la mezcla reaccionante

ΔH_R es la entalpía de reacción

ΔH_{dil} es la entalpía de dilución

U es el coeficiente global de transferencia de calor

A es el área de transferencia entre el reactor y la chaqueta

T_A es la temperatura del agua de enfriamiento

T_e es la temperatura de entrada de los reactivos

Para el lado de la chaqueta tenemos lo siguiente:

$$V_A\rho_A Cp_A \frac{d}{dt}T_A = F_A\rho_A Cp_A(T_{A,e} - T_A) - UA(T_A - T)$$

$$\frac{dT_A}{dt} = \frac{F_A}{V_A}(T_{A,e} - T_A) - \frac{UA}{V_A\rho_A Cp_A}(T_A - T) \tag{2.11}$$

En donde

F_A es el flujo de agua de enfriamiento

Cp_A es la capacidad calorífica del agua

V_A es el volumen de agua de enfriamiento en la chaqueta

ρ_A es la densidad del agua

$T_{A,e}$ es la temperatura de entrada del agua de enfriamiento

El valor del coeficiente global de transferencia de calor para el reactor propuesto es de $400\frac{W}{m^2K}$, que está dentro del rango que establece Winkler [8]. Él dice que para un recipiente enchaquetado con solución acuosa en el interior y agua en la chaqueta, el coeficiente global de transferencia de calor se encuentra entre 200-500$\frac{W}{m^2K}$.

Las ecuaciones del balance de energía escritas en notación funcional son:

$$\frac{dT}{dt} = f_3(T, T_A, \bar{x}) \tag{2.12}$$

$$\frac{dT_A}{dt} = f_4(T, T_A, \int_0^t T dT, \frac{d}{dt}T) \tag{2.13}$$

La ecuación (2.13) depende de la integral y la derivada de la temperatura porque el término F_A es la variable manipulada por el controlador, como se menciona en la siguiente sección. Podemos ver que el balance de materia (2.8 y 2.9), el de energía del reactor (2.12) y el de energía para la chaqueta (2.13) están acoplados, por lo que se deberá resolver el sistema de ecuaciones diferenciales.

2.4. Control PID

Se controlará el flujo de entrada de agua de enfriamiento (F_A), ya que la conversión de la reacción depende directamente de la temperatura. Además se debe de cuidar la temperatura dentro del reactor porque uno de los reactivos (el óxido de propileno) es muy volátil. La ecuación es la siguiente:

$$F_A = F_{A,sesgo} - Kc\left((T_{sp} - T) + \tau_D\frac{d}{dt}(T_{sp} - T) + \frac{1}{\tau_I}\int_0^t (T_{sp} - T)dt\right)$$

$$F_A = F_{A,sesgo} - Kc\left((T_{sp} - T) - \tau_D\frac{d}{dt}T + \frac{1}{\tau_I}\int_0^t (T_{sp} - T)dt\right) \tag{2.14}$$

En donde

$F_{A,sesgo}$ es el valor del flujo de agua de enfriamiento cuando el sistema se encuentra en estado estacionario.

T_{sp} es la temperatura del *set point*, es decir, la temperatura a la cual el controlador deberá mantener en el sistema.

Kc, τ_D, τ_I son los parámetros afinables del controlador

2.5. Datos

Se tienen los siguientes datos (algunos de ellos obtenidos del ejemplo 8-8 de *"Elements of Chemical Reactions"* 4ª Ed [6], del *Manual del Ingeniero Químico* y del artículo de Furusawa):

$$F_1 = F_2 = 4{,}72x10^{-3}\frac{m^3}{s}$$
$$\frac{F_3}{F_1 + F_2} = 2{,}5$$
$$T_0 = 15^oC$$
$$V = 2{,}3m^3$$
$$\Delta H_R = -36967{,}4\frac{kcal}{kmol}$$
$$\Delta H_{dil} = -8{,}5\frac{kcal}{kgmezcla}$$
$$k_0 = 4{,}71x10^9\frac{1}{s}$$
$$E_A = 18000\frac{cal}{mol}$$
$$R = 1{,}9859\frac{cal}{molK}$$
$$F_A = 11{,}54x10^{-3}\frac{m^3}{s}$$
$$Cp_A = 1\frac{kcal}{kg^oC}$$

$$\rho_A = 1000\frac{kg}{m^3}$$
$$U = 2500\frac{J}{sm^2K} = 0{,}5971\frac{kcal}{sm^2K}$$

Se propone que el área total de contacto entre el reactor y la chaqueta sea de $25{,}12m^2$. Los datos de las substancias son:

Substancia	$\bar{Cp}_i(kcal/(mol^oC))$	$\rho_i(kg/m^3)$	$M_i(g/mol)$
Óxido de Propileno (OP)	35	830	58
Metanol (Me)	19.5	791.8	32
Agua (H_2O)	18	1000	18
Propilenglicol (PG)	46	1040	76

Cuadro 2.1: Propiedades fisicoquímicas de las substancias

Debido a que no hay cambios significativos de temperatura, los valores anteriores se supondrán constantes. por tanto, se calcularán valores promedios y se utilizarán en los cálculos. Estos valores se obtienen de la siguiente manera:

Corrientes de entrada

$$W_{i,e} = F_{i,e}\rho_{i,e}$$
$$W_{OP,e} = F_{OP,e}\rho_{OP} = 0{,}3717\frac{kg}{s}$$
$$W_{Me,e} = F_{Me,e}\rho_{Me} = 0{,}3737\frac{kg}{s}$$
$$W_{H_2O,e} = F_{H_2O,e}\rho_{H_2O} = 2{,}3597\frac{kg}{s}$$
$$W_{PG,e} = F_{PG,e}\rho_{PG} = 0\frac{kg}{s}$$
$$W_e = \sum_{i=1}^{n} W_{i,e}$$
$$W_e = W_{OP,e} + W_{Me,e} + W_{H_2O,e} + W_{PG,e} = 3{,}1251\frac{kg}{s}$$

Donde

$W_{i,e}$ es el flujo másico de la substancia i a la entrada

$F_{i,e}$ es el flujo volumétrico de la substancia i a la entrada

$\rho_{i,e}$ es la densidad de la substancia i a la entrada

W_e es el flujo másico total a la entrada

A su vez, los flujos molares de entrada son

$$\bar{W}_{i,e} = W_{i,e} M_i^{-1}$$

$$\bar{W}_{OP,e} = W_{OP,e} M_{OP}^{-1} = 6{,}75e^{-3} \frac{mol}{s}$$

$$\bar{W}_{Me,e} = W_{Me,e} M_{Me}^{-1} = 11{,}68e^{-3} \frac{mol}{s}$$

$$\bar{W}_{H_2O,e} = W_{H_2O,e} M_{H_2O}^{-1} = 131{,}10e^{-3} \frac{mol}{s}$$

$$\bar{W}_{PG,e} = W_{PG,e} M_{PG}^{-1} = 0 \frac{mol}{s}$$

$$\bar{W}_e = \sum_{i=1}^{n} \bar{W}_{i,e}$$

$$\bar{W}_e = \bar{W}_{OP,e} + \bar{W}_{Me,e} + \bar{W}_{H_2O,e} + \bar{W}_{PG,e} = 149{,}53e^{-3} \frac{mol}{s}$$

Donde

$\bar{W}_{i,e}$ es el flujo molar de la substancia i a la entrada

$W_{i,e}$ es el flujo másico de la substancia i a la entrada

$M_{i,e}$ es la masa molar de la substancia i

$\bar{W}_e$ es el flujo molar total a la entrada

Por lo tanto las fracciones másicas (x_i) y molares ($\bar{x}_i$) serán:

$$x_{i,e} = \frac{W_{i,e}}{W_e} \qquad \bar{x}_{i,e} = \frac{\bar{W}_{i,e}}{\bar{W}_e}$$

$$x_{OP,e} = \frac{W_{OP,e}}{W_e} = 0{,}1253 \qquad \bar{x}_{OP,e} = \frac{\bar{W}_{OP,e}}{\bar{W}_e} = 0{,}0452$$

$$x_{Me,e} = \frac{W_{Me,e}}{W_e} = 0{,}1196 \qquad \bar{x}_{Me,e} = \frac{\bar{W}_{Me,e}}{\bar{W}_e} = 0{,}0781$$

$$x_{H_2O,e} = \frac{W_{H_2O,e}}{W_e} = 0{,}7551 \qquad \bar{x}_{H_2O,e} = \frac{\bar{W}_{H_2O,e}}{\bar{W}_e} = 0{,}8767$$

$$x_{PG,e} = \frac{W_{PG,e}}{W_e} = 0 \qquad \bar{x}_{PG,e} = \frac{\bar{W}_{PG,e}}{\bar{W}_e} = 0$$

Donde

$x_{i,e}$ es la fracción másica de la substancia i a la entrada

$\bar{x}_{i,e}$ es la fracción molar de la substancia i a la entrada

Corrientes de salida

Para calcular los siguientes datos, se utiliza la tabla estequiométrica 2.2.

	C_3H_5OH	$+$	H_2O	$\rightarrow$	C_3H_7OOH
inicio)	$F_{OP}\rho_{OP}M_{OP}^{-1}$		$F_{H_2O}\rho_{H_2O}M_{H_2O}^{-}1$		
reaccion)	$F_{OP}\rho_{OP}M_{OP}^{-1}$		$F_{OP}\rho_{OP}M_{OP}^{-1}$		
final)	0		$F_{H_2O}\rho_{H_2O}M_{H_2O}^{-}1 - F_{OP}\rho_{OP}M_{OP}^{-1}$		$F_{OP}\rho_{OP}M_{OP}^{-1}$

Cuadro 2.2: Tabla estequimétroca de la reacción de óxido de propileno

Con estos datos podemos calcular los flujos másicos y molares de salida, los cuales son los siguientes:

$$\bar{W}_{OP} = 0\frac{mol}{s}$$

$$\bar{W}_{Me} = W_{Me,e}M_{Me}^{-1} = 11{,}68e^{-3}\frac{mol}{s}$$

$$\bar{W}_{H_2O} = F_{H_2O,e}\rho_{H_2O}M_{H_2O}^{-}1 - F_{OP,e}\rho_{OP}M_{OP}^{-1} = 124{,}34e^{-3}\frac{mol}{s}$$

$$\bar{W}_{PG} = W_{PG,e}M_{PG}^{-1} = 6{,}75e^{,3}\frac{mol}{s}$$

$$\bar{W} = \bar{W}_{OP} + \bar{W}_{Me} + \bar{W}_{H_2O} + \bar{W}_{PG} = 142{,}77e^{-3}\frac{mol}{s}$$

$$W_{OP} = \bar{W}_{OP}M_{OP} = 0\frac{kg}{s}$$

$$W_{Me} = \bar{W}_{Me}M_{Me} = 0{,}3737\frac{kg}{s}$$

$$W_{H_2O} = \bar{W}_{H_2O}M_{H_2O} = 2{,}2382\frac{kg}{s}$$

$$W_{PG} = \bar{W}_{PG}M_{PG} = 0{,}5133\frac{kg}{s}$$

$$W = W_{OP} + W_{Me} + W_{H_2O} + W_{PG} = 3{,}1251\frac{kg}{s}$$

Y sus respectivas fracciones:

$$x_{OP} = \frac{W_{OP}}{W} = 0 \qquad \bar{x}_{OP} = \frac{\bar{W}_{OP}}{\bar{W}} = 0$$
$$x_{Me} = \frac{W_{Me}}{W} = 0{,}1196 \qquad \bar{x}_{Me} = \frac{\bar{W}_{Me}}{\bar{W}} = 0{,}0818$$
$$x_{H_2O} = \frac{W_{H_2O}}{W} = 0{,}8701 \qquad \bar{x}_{H_2O} = \frac{\bar{W}_{H_2O}}{\bar{W}} = 0{,}8701$$
$$x_{PG} = \frac{W_{PG}}{W} = 0{,}1642 \qquad \bar{x}_{PG} = \frac{\bar{W}_{PG}}{\bar{W}} = 0{,}0473$$

Usando los datos generados anteriormente con los de la tabla 2.1 y suponiendo soluciones ideales:

$$\bar{\Lambda}_m = \sum \bar{\Lambda}_i * \bar{x}_i$$
$$\Lambda_m = \sum \Lambda_i x_i$$

Donde

$\bar{\Lambda}_j$ es cualquier propiedad molar de la mezcla o el componente

Λ_j es cualquier propiedad másica de la mezcla o el componente

Se generó la siguiente tabla:

	$\rho(\frac{kg}{m^3})$	$\bar{Cp}(\frac{kcal}{kmol^oC})$	$Cp(\frac{kcal}{kg^oC})$	$M(\frac{kg}{kmol})$
Entrada	953.8	18.88	0.9033	20.9
Salida	981.67	19.45	0.8885	21.89
Promedio	967.74	19.17	0.8962	21.39

Cuadro 2.3: Propiedades medias del sistema

Fácilmente se puede calcular el flujo volumétrico a la salida, el cual es

$$F_s = \frac{W}{\rho} = \frac{3{,}1251\frac{kg}{s}}{967{,}74\frac{kg}{m^4}} = 3{,}230x10^{-3}\frac{m^3}{s} \tag{2.15}$$

Y el flujo promedio que será utilizado es:

$$F = \frac{F_1 + F_2 + F_3 + F_s}{2} = \frac{4{,}72 + 4{,}72 + 4{,}72 * 5 + 32{,}3}{2} x10^{-4}\frac{m^3}{s}$$
$$F = 3{,}267x10^{-3}\frac{m^3}{s}$$

Estas serán las propiedades promedio las que se utilizarán en este trabajo.

Capítulo 3

Simulación en estado transitorio

La simulación de diferentes sistemas se lleva a cabo usando modelos matemáticos dependientes del tiempo. Para poder llevar a cabo estas simulaciones necesitamos de herramientas matemáticas que nos permitan resolver las ecuaciones de los modelos. La manera más exacta y la más confiable es resolver las ecuaciones analíticamente, aunque en la mayoría casos puede ser algo muy complejo, cuando no imposible. Otra manera es resolviendo las ecuaciones con métodos numéricos, utilizando el poder de cómputo disponible hoy en día. Así podemos obtener resultados muy cercanos a los obtenidos analíticamente. En esta sección se comenzará por definir algunos métodos que se usarán en la solución de ecuaciones que modelan al reactor. Los algoritmos aquí mencionados han sido obtenidos del libro de Chapra *et al* "*Métodos numéricos para ingenieros*" [20].

3.1. Métodos Runge-Kutta

Los métodos de Runge-Kutta (RK) cuentan con gran cantidad de variantes. Todos ellos cambian en complejidad de acuerdo a la orden del método, que está dado por la cantidad de evaluaciones que se le hace a la ecuación diferencial. Sin embargo, solamente se explicará el método de cuarto orden, que es el que será usado.

Un método de 4º orden (*RK4*) está definido por las siguientes ecuaciones:

$$y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2_3 + k4) \tag{3.1}$$

donde:

$$k_1 = f(x_i, y_i) \tag{3.2}$$

$$k_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_1\right) \tag{3.3}$$

$$k_3 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_2\right) \tag{3.4}$$

$$k_4 = f\left(x_i + h, y_i + hk_3\right) \tag{3.5}$$

Lo que hace este método es evaluar diferentes pendientes en cada una de las k, por lo que la ecuación (3.1) es una ponderación de todas las pendientes evaluadas. Aquí h representa el paso por iteración del método numérico. Hay que mencionar que mientras más pequeño sea el paso, mayor cantidad de iteraciones se tienen que hacer, pero la precisión es mayor que con un paso más grande.

Podemos hacer la comparación entre el método RK4 y el método de Euler, para ello se propone la siguiente ecuación:

$$\frac{dy}{dx} = 2cos(2x) \tag{3.6}$$

La solución analítica será $f(x) = sin(2x)$, sin embargo, resolviendo la ecuación (3.6) numéricamente, obtenemos el resultado de la figura 3.1.Se observa claramente que, con un mismo paso, h, el método RK4 es mucho más preciso que el Euler.

3.2. Simulación de un CSTR

Como se mencionó en el capítulo 2, la manera de resolver las ecuaciones del modelo es resolviendo el sistema de ecuaciones. En este caso tenemos 4 ecuaciones diferenciales ordinarias, donde 3 de ellas están acopladas. Se

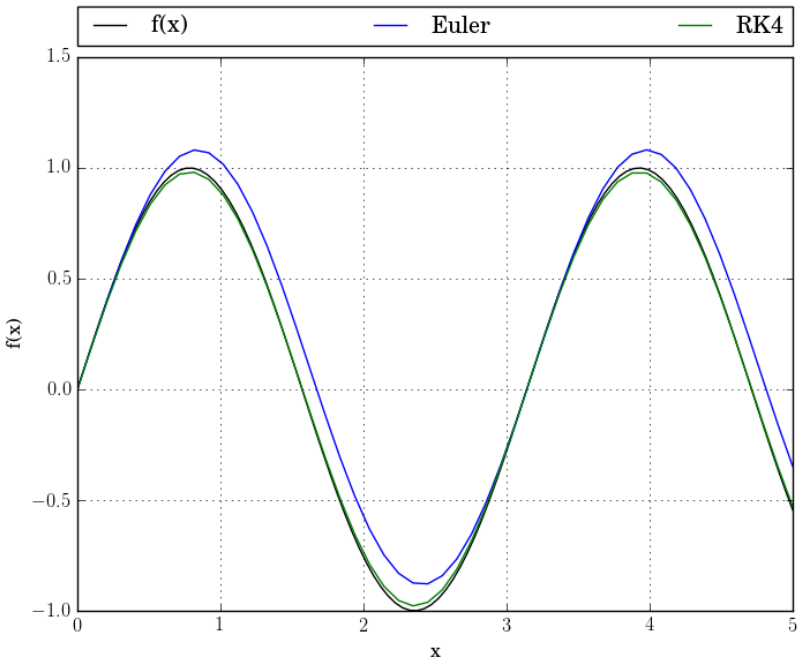

Figura 3.1: Comparación entre el método Euler y RK4, con $h = 0{,}1$

representará a las ecuaciones de la siguiente manera:

$$\begin{aligned}
\frac{d}{dt}\bar{x} &= f_1(T, \bar{x}) \\
\frac{d}{dt}C_{PG} &= f_2(T, \bar{x}, C_{PG}) \\
\frac{d}{dt}T &= f_3(T, T_A, \bar{x}) \\
\frac{d}{dt}T_A &= f_4(T, T_A, \int_0^t T dT, \frac{d}{dt}T)
\end{aligned}$$

Sin embargo, f_4 es una ecuación integro-diferencial, esto quiere decir que es diferencial y tiene un término integral. Dada la complejidad del problema y por la facilidad de implementación, se resolverá el sistema de ecuaciones numéricamente, usando el método RK4 hibridizado con el método de los trapecios. Para la simulación del CSTR se utilizará un valor de $h = 0{,}1$. El método de resolución del sistema de ecuaciones es entonces el siguiente:

Cálculo de todas las $k_{1,i}$

$k_{1,1} = f_1(T, \bar{x})$

$k_{1,2} = f_2(T, \bar{x}, C_{PG})$

$k_{1,3} = f_3(T, T_A, \bar{x})$

$k_{1,4} = f_4(T, T_A, 0, k_{1,3})$

Se calcula el valor del trapecio cuando avanza $h/2$

$$T_{1/2} = T + \frac{h}{2}k_{12}$$

$$Trap_{1/2} = (2T_{sp} - (T + T_{1/2}))\frac{h}{4}$$

Cálculo de todas las $k_{2,i}$

$$k_{2,1} = f_1(T + \frac{h}{2}k_{1,3}, x + \frac{h}{2}k_{1,1})$$

$$k_{2,2} = f_2(T + \frac{h}{2}k_{1,3}, x + \frac{h}{2}k_{1,1}, C_{PG} + \frac{h}{2}k_{1,2})$$

$$k_{2,3} = f_3(T + \frac{h}{2}k_{1,3}, T_A + \frac{h}{2}k_{1,4}, x + \frac{h}{2}k_{1,1})$$

$$k_{2,4} = f_4(T + \frac{h}{2}k_{1,3}, T_A + \frac{h}{2}k_{1,4}, Trap_{1/2}, k_{2,3})$$

Cálculo de todas las $k_{3,i}$

$$k_{3,1} = f_1(T + \frac{h}{2}k_{2,3}, x + \frac{h}{2}k_{2,1})$$

$$k_{3,2} = f_2(T + \frac{h}{2}k_{2,3}, x + \frac{h}{2}k_{2,1}, C_{PG} + \frac{h}{2}k_{2,2})$$

$$k_{3,3} = f_3(T + \frac{h}{2}k_{2,3}, T_A + \frac{h}{2}k_{2,4}, x + \frac{h}{2}k_{2,1})$$

$$k_{3,4} = f_4(T + \frac{h}{2}k_{2,3}, T_A + \frac{h}{2}k_{2,4}, Trap_{1/2}, k_{3,3})$$

Se calcula el valor del trapecio cuando avanza h

$$T_1 = T_{1/2} + \frac{h}{2}k_{1,2}$$

$$Trap_1 = (2T_{sp} - (T_1 + T_{1/2}))\frac{h}{4}$$

$$Trap = Trap_{1/2} + Trap_1$$

Cálculo de todas las $k_{4,i}$

$$k_{4,1} = f_1(T + hk_{3,3}, x + hk_{3,1})$$

$$k_{4,2} = f_2(T + hk_{3,3}, x + hk_{3,1}, C_{PG} + hk_{3,2})$$

$$k_{4,3} = f_3(T + hk_{3,3}, T_A + hk_{3,4}, x + hk_{3,1})$$
$$k_{4,4} = f_4(T + hk_{3,3}, T_A + hk_{3,4}, Trap, k_{4,3})$$

Actualización de las variables

$$\bar{x} = \bar{x} + \frac{h}{6}(k_{1,1} + 2k_{2,1} + 2k_{3,1} + k4, 1)$$
$$C_{PG} = C_{PG} + \frac{h}{6}(k_{1,2} + 2k_{2,2} + 2k_{3,2} + k4, 2)$$
$$T = T + \frac{h}{6}(k_{1,3} + 2k_{2,3} + 2k_{3,3} + k4, 3)$$
$$T_A = T_A + \frac{h}{6}(k_{1,4} + 2k_{2,4} + 2k_{3,4} + k4, 4)$$

Para entender mejor el método hibridizado, se utilizará la figura 3.2, donde también se puede apreciar las forma que pudiesen tomar las evaluaciones de las pendientes del método RK4.

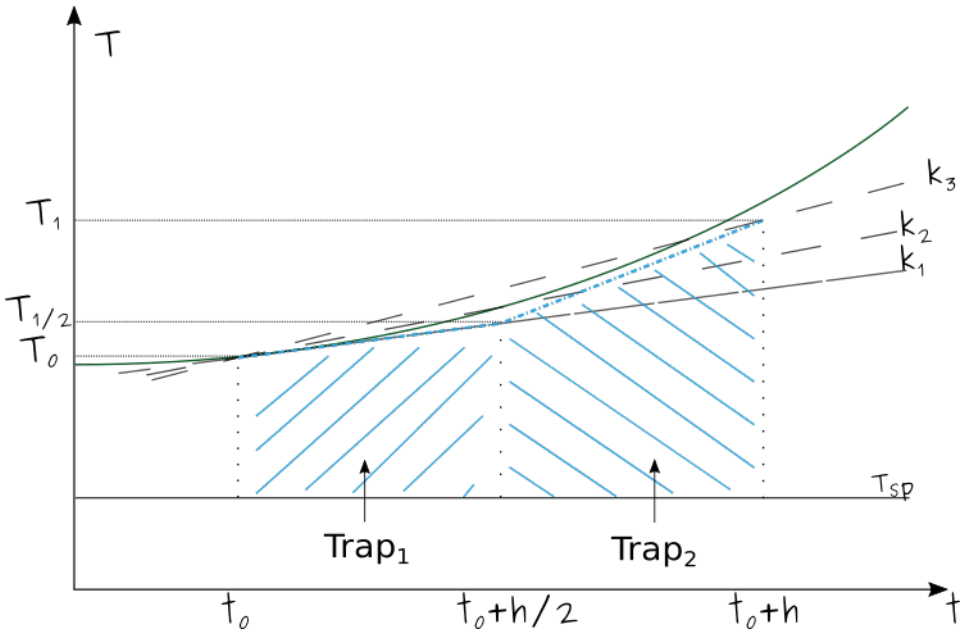

Figura 3.2: Cálculo de los trapecios en el método híbrido

En éste método híbrido, se procede a calcular dos trapecios, uno para la sección entre x_o y $x_o + \frac{h}{2}$ y el otro para la sección entre $x_o + \frac{h}{2}$ y $x_o + h$. Para ello se calculan los valores de $T_{1/2}$ y T_1, que corresponden a las temperaturas en $t_0 + \frac{h}{2}$ y $t_0 + h$. Después de eso simplemente se calcula el valor del trapecio

correspondiente, donde:

$$Trap_1 = \frac{(T_{sp} - T_0 + T_{sp} - T_{1/2})\frac{h}{2}}{2}$$
$$Trap_2 = \frac{(T_{sp} - T_{1/2} + T_{sp} - T_1)\frac{h}{2}}{2}$$

Podemos ver que, con base en la figura, ambos trapecios deben tener un signo negativo, debido a que se definió que el término integral en el controlador PID es $\int_{t_0}^{t}(T_{sp} - T)dT$.

Capítulo 4

Algoritmos genéticos

Los algoritmos genéticos (AG) son estrategias de optimización desarrollados por John Holland [9], quien se basó en la teoría de la evolución de Darwin para optimizar una función objetivo. La naturaleza es un sistema optimizador por excelencia, pues con base en diferentes tipos de ecosistemas genera individuos cuyas características se van adecuando para vivir en ese tipo de ambiente. Hay tanta diversidad de ellos que al mezclarse generan nuevos individuos con características de ambos padres y algunas que son únicas en su especie, de esta forma se genera una población variada de la que se seleccionan los individuos más aptos. Esto hace que a través de los años existan nuevos seres cada vez más adaptados a su medio ambiente. Darwin se dio cuenta de esto y escribió una obra (1859) que cambiaría la manera de ver a la naturaleza. Los algoritmos genéticos que podemos llamar tradicionales, usan codificaciones de las posibles soluciones del problema de optimización que se está atacando para aplicar un grupo de operadores (operadores genéticos) sobre ellas; es mediante la aplicación de esos operadores que se van transformando esas codificaciones (y por tanto las soluciones que representan), con el objetivo de obtener cada vez mejores soluciones.

Las codificaciones que usan los algoritmos genéticos, AGs, tradicionales son cadenas binarias. Un ejemplo de cadena binaria es: 0110100101101. Los algoritmos genéticos pertenecen al tipo de algoritmos que van explorando el dominio de la función (o *espacio de búsqueda*, como se le llama en inteligencia artificial) en busca de óptimo de ésta. Los algoritmos tradicionales de este tipo, hacen esta búsqueda punto a punto, es decir en cada paso en búsqueda del óptimo, pasan de un solo punto en el dominio de la función a otro. Los AGs hacen esta búsqueda usando varios puntos en el dominio simultáneamente, es decir, en un paso se mueven de una lista de puntos en

el dominio a otra. A esta lista de puntos se le llama *población*; a cada punto en la población se le llama *individuo.*

Los principales operadores genéticos que usa un AG son la selección, el cruzamiento y la mutación. Los dos últimos operadores, el cruzamiento y la mutación actúan sobre una población para producir otra población que por lo general contiene al menos algunos individuos que no contenía la población de la que se generó; es decir, son operadores que generan variedad en las poblaciones. El operador de selección se encarga de favorecer la persistencia y proliferación a lo largo de las generaciones, de los individuos más aptos.

Un algoritmo genético debe hacer uso de un subprograma que se encargue de asignar una aptitud a cada individuo obtenido. Ese subprograma hace uso de la función objetivo, que es la función que se desea optimizar. Para evaluar la función objetivo, es necesario descodificar las cadenas binarias para obtener los valores de las variables de las que depende.

En la gran mayoría de los casos, los operadores de selección son no deterministas, esto es, si deben seleccionar entre dos individuos, no toman con una probabilidad del cien por ciento al mejor de ellos, aunque sea mucho más apto que el otro, sino que pueden, aunque sea con una probabilidad muy baja, seleccionar al menos apto. La razón de diseñar operadores de selección con este comportamiento es el permitir que, en el proceso de búsqueda del óptimo, se pueda abandonar el que puede ser un óptimo local, en búsqueda de otro mejor óptimo local o el óptimo global. De eso se desprende que la aptitud de cualquier individuo debe ser siempre mayor que cero.

4.1. El Algoritmo Genético Tradicional

Una forma de representar un algoritmo genético tradicional se muestra en el algoritmo 1.

En este algoritmo, las variables w_i (línea 1), representan cadenas binarias en las que se codifica una posible solución, éstos son los individuos. f_i (líneas 2 y 8) es la aptitud asignada al *i-ésimo* individuo, se calcula con una composición de funciones:$f \circ f_{obj} \circ d$, donde $d : [0, 1]^l \to \Re^n$; y si se está optimizando una función objetivo en lo reales, de parámetros continuos, f_{obj}, $f : \Re^n \to \Re$, es decir, $f_i = f(f_{obj}(d(w_i)))$. El criterio de terminación (línea 4), por lo general, consiste en alcanzar un número predeterminado (por el usuario) de iteraciones (generaciones). P(1/2) (líneas 5, 6 y 7) se refiere a una población intermedia, ésta es una población donde se van almacenando las copias de los individuos seleccionados. Para aplicar el operador de cruzamiento (línea 6), se debe antes formar, al azar y sin reemplazo, una pareja

Algoritmo 1 Algoritmo Genético Tradicional

1: Obtener una población inicial: P(0) = (w_1,w_2,...,w_μ)
2: Calcular las aptitudes: f_1,f_2,...,f_μ
3: $i = 0$
4: **Mientras** no se cumpla criterio de terminación **Hacer**
5: P(1/2) = Selección(P(i))
6: P(1/2) = Cruzamiento(P(1/2),p_c)
7: P($i+1$) = Mutación (P(1/2), p_m)
8: Calcular las aptitudes: f_1,f_2,...,f_μ
9: $i = i + 1$
10: **Terminar Mientras**

de individuos de la población intermedia; p_c, es la probabilidad de cruzamiento, esta es la probabilidad de que se someta a cruzamiento (ver sección 4.3) la pareja que se ha formado. Por lo general, a p_c se le asignan valores mayores a 0.65. Para determinar si la pareja formada se cruzara o no, se obtiene un número aleatorio, r, de distribución uniforme en el intervalo [0, 1], si $r \leq p_c$, la pareja se cruza. El operador de mutación (línea 7), consiste en lo siguiente, para cada bit de la población se obtiene un número aleatorio, r, de distribución uniforme, si $r \leq p_m$, el bit en cuestión se cambia por su complemento (i.e., si es 1 se cambia por 0 y viceversa). La probabilidad de mutación, p_m, se acostumbra fijar en el intervalo [0.001, 0.01].

La naturaleza ha decidido que la manera más efectiva de almacenar la información es en código cuaternario, utilizando 4 bases nitrogenadas llamadas: Adenina, Citosina, Timina y Guanina. Se podría utilizar una representación de 0, 1, 2 y 3 para cada base y obtendríamos el mismo resultado. En los algoritmos genéticos, que se pueden llamar clásicos, se codifica el resultado en código binario, aunque el código usado puede ser uno que se adecue al problema en cuestión. Esto es porque cada cadena binaria muestrea una mayor proporción del espacio de búsqueda, a comparación si se utilizara otro sistema numérico [10, pp.66–74]. Un ejemplo del genoma codificado en una cadena binaria es el de la siguiente figura:

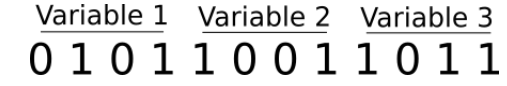

Figura 4.1: Representación del código genético como una cadena binaria

En donde hay 3 variables que conforman a la cadena, todas codificadas

en código binario de 4 bits (se le llama bit a cada dígito binario). Cuando se juntan, forman una cadena de 12 bits que representa a un individuo. Sin embargo, la representación de las variables puede adaptarse al problema utilizando diferentes estructuras de datos, en lugar de solamente utilizar cadenas binarias. Hay variaciones que utilizan matrices, listas, grafos, entre otros, para representar al individuo.

Cuando los algoritmos genéticos utilizan codificaciones de los valores de las variables continuas con cadenas binarias, requieren que el número de esas cadenas sea finito y que la longitud de cada una de ellas también lo sea. Estos dos hechos implican que el dominio de la variable continua se discretiza. Por tanto, se puede numerar la cantidad de valores de la variable en el intervalo fijado para ella con un número entero y cada uno de esos valores se representará con una cantidad fijada de dígitos. Por ejemplo, sea la variable $x \in [a, b]$, supongamos que dividimos ese intervalo en s subintervalos, el tamaño de cada subintervalo se puede calcular con la ecuación:

$$\Delta x = \frac{b - a}{s} \tag{4.1}$$

La cantidad de números que habrá en el intervalo $[a, b]$, será $s + 1$. Cada número se puede expresar así:

$$x_i = i\Delta x + a \text{ ; para } i = 0, 1, ..., s \tag{4.2}$$

Obsérvese que i hace las veces de un índice de los números en el intervalo. Sea, por ejemplo, $s = 100$, es decir, que el intervalo se divida en 100 subintervalos; entonces el intervalo se representará con 101 números. Para codificar cada número se puede usar el índice i, es decir, para hacer referencia a uno de los 101 números del intervalo, basta con indicar simplemente su índice, así, se puede hacer referencia al límite inferior del intervalo, a, con el índice $i = 0$, y al límite superior del intervalo, b, con el índice $i = 100$. Si deseamos asignar una cadena binaria para codificar a cada número, simplemente se puede expresar el valor numérico del índice i en sistema binario. Por ejemplo, para $i = 100$ el número en sistema binario es: 1100100. Es necesario que todos los números binarios que se usan para representar el intervalo tengan la misma longitud, así el límite inferior del intervalo se codificará con el número binario 0000000, el siguiente número binario 0000001 y así sucesivamente. Es muy conveniente que todos los números binarios de una longitud dada, en este caso 7, representen (codifiquen) algún número en el intervalo, esto implica que es conveniente que el límite inferior del intervalo, a, se codifique con el número binario 0000000 y el superior, b, con el número

1111111. Para logra esto, debemos cambiar el número de subintervalos en el que se debe dividir el intervalo, en este caso ese número es $2^7 - 1 = 127$ subintervalos. Tomando en cuenta lo anterior, para decodificar una cadena binaria, w, dada, de longitud l, se debe seguir el algoritmo 2.

Algoritmo 2 Decodificación de una cadena binaria en un número de punto flotante. Caso en el que se tiene como datos w, a, b y l.

Entrada: w, a, b, l

1: Calcular el tamaño del subintervalo, ecuación (4.1)
2: Convertir el entero binario, w, en el entero decimal correspondiente, d: $d = Bin\_a\_Dec(w)$
3: Calcular el valor del número codificado, ecuación (4.2)

La función $Bin\_a\_Dec()$ (línea 2) convierte enteros binarios en enteros decimales.

Muchas veces no se conoce la longitud del número binario o cadena binaria, con la que se representarán los números del intervalo, pero sí se tiene una idea de la magnitud de Δx_{max}, la distancia máxima que debe haber entre dos números consecutivos de la representación discretizada: $\Delta x_{max} = x_{i+1} - x_i$, en ese caso se puede aplicar el algoritmo 3 para decodificar una cadena binaria w dada.

Algoritmo 3 Decodificación de una cadena binaria en un número de punto flotante. Caso en el que se tiene como datos w, a, b y Δx_{max}.

Entrada: w, a, b, Δx_{max}

1: Calcular la longitud de la cadena

$$l = \lceil log_2 \left(\frac{b-a}{\Delta x_{max}} + 1 \right) \rceil \tag{4.3}$$

2: Calcular la magnitud del subintervalo corregido (Δx_{corr}):

$$\Delta x_{corr} = \frac{b-a}{2^l - 1} \tag{4.4}$$

3: Convertir el entero binario, w, en el entero decimal correspondiente, d: $d = Bin\_a\_Dec(w)$
4: Calcular el valor del número codificado: $x = d\Delta x_{corr} + a$

Si el resultado del argumento de la función techo en la ecuación (4.3) no es un entero, entonces esa función hace que a la longitud, l, se le asigne el

número entero inmediato superior, esto hace que disminuya el valor de la Δx_{max} propuesta, ese valor corresponde a la Δx_{corr} (línea 3).

Para codificar una variable continua en una cadena binaria, simplemente se tiene que utilizar la ecuación (4.2), despejando la variable d, la cual corresponde a un entero decimal que se encuentra en el intervalo $[0,2^l]$. Finalmente se usa una función que convierte el entero decimal a un número binario, de la siguiente manera:

$$w = Dec\_a\_Bin(d) \tag{4.5}$$

Como se mencionó anteriormente, la función objetivo puede depender de una o más variables. En caso de que dependa de más de una variable, los números binarios correspondientes a la codificación de cada variable deben concatenarse, es decir unirse para formar una sola cadena binaria. Es importante tener esto en mente al momento de decodificarlas.

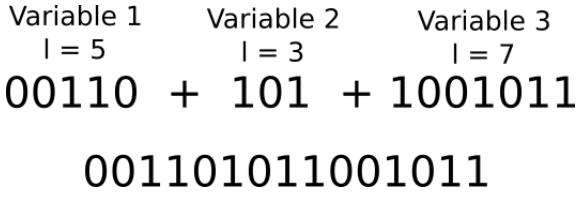

Figura 4.2: Concatenación de variables de diferente longitud

En la figura 4.2 podemos apreciar la concatenación de 3 variables codificadas como cadenas binarias de diferente longitud. La primera de longitud 5, la segunda de longitud 3 y la cuarta de longitud 7. El resultado final es una cadena de 15 bits de longitud.

Los algoritmos genéticos son poderosos algoritmos de optimización y han sido muy populares en las últimas décadas por muchas razones, algunas de ellas son:

- Hace una búsqueda paralela explicita del óptimo de la función. Esto lo hace al probar diferentes posibles soluciones (individuos) por iteración.
- Realiza una búsqueda por paralelismo implícito al muestrear una gran proporción del espacio de búsqueda. Esta proporción es mayor cuando se codifica en cadenas binarias [10, pp.66–74].
- No utiliza direcciones de búsqueda basadas en el gradiente, por lo que no es necesario tener información sobre las derivadas.

- Tiene menor probabilidad de quedarse en un máximo local, como lo haría un algoritmo escalador.
- El número de variables de las que depende la función a optimizar no representa ningún problema en la operación de un AG.
- Puede optimizar funciones ruidosas, rugosas y discontinuas.
- Puede optimizar problemas en donde no se tenga de manera explícita la función objetivo, pero sí se tenga la capacidad de comparar cada pareja de posibles soluciones que se presenten y decir cual es la mejor.

4.2. Operadores de selección

La selección es un operador muy importante, ya que es el que orienta la búsqueda. Los otros dos operadores "clásicos" de los algoritmos genéticos, el cruzamiento y la mutación, tienen por objeto formar nuevos individuos, esto es proporcionar variedad que es sometida al proceso de selección. El operador de selección puede proporcionar una selección a la que podríamos llamar "laxa" o una selección "rigurosa"; en el primer caso nos referimos a una selección poco estricta; esto traerá por resultado que el proceso de búsqueda tienda a ser lento, pero tenderá a encontrar buenos valores (cercanos al o a los óptimos). En el otro extremo, la selección muy rigurosa, tenderá a llevar al proceso hacia los óptimos locales rápidamente, haciendo que el salir de un óptimo local sea muy difícil o casi imposible. Se utilizan los operadores de selección para obtener una población intermedia, generalmente de tamaño igual o mayor a la población inicial (μ), que contiene, en teoría, a los mejores individuos de cada población, a los que posteriormente se les someterá a los demás operadores. Algunos operadores de selección, mencionados por Melanie Mitchell [12, pp.166–171], son los siguientes:

- **Selección por torneo**. Consiste en lo siguiente:
  1. Todos los miembros de la población, μ individuos, se dividen al azar en grupos de igual tamaño, el número de individuos por grupo lo define el usuario.
  2. De cada grupo, se selecciona al azar, favoreciendo a los más aptos, un individuo; los individuos seleccionados se almacenan en una población intermedia
  3. El proceso se repite desde el paso (1), tantas veces haga falta, hasta completar el número fijado por el usuario de individuos en la población intermedia, λ.

Por ejemplo, si cada grupo se hace de dos individuos, se tendrá como resultado de la primera selección, una población intermedia con $\mu/2$ individuos; después de repetir el proceso se habrán seleccionado otros $\mu/2$ individuos. Si el usuario fijó que el tamaño de la población intermedia fuera igual al de la población, es decir $\mu = \lambda$, bastarán estas dos aplicaciones del proceso. Otro ejemplo, si el número de individuos por grupo es de cuatro, entonces en cada proceso se seleccionarán m/4 individuos, suponiendo que el usuario sigue fijando $\mu = \lambda$, se necesitará repetir el proceso de selección cuatro veces. Cuando se escogen dos individuos por grupo, a la hora de hacer la selección en un grupo se puede asignar, por ejemplo, una aptitud de 0.75 al mejor de ellos y de 0.25 al peor, así tendrá tres veces más oportunidad de ser escogido el mejor individuo sobre el peor. En este último caso el mejor individuo de la población, será escogido cuando mucho dos veces. En cambio, si se fija un tamaño de grupo de cuatro, el mejor individuo de la población puede ser seleccionado hasta cuatro veces. Se puede ver entonces que la dureza de la selección en este operador se puede regular con el tamaño de los grupos, a mayor tamaño de los grupos más "rondas" selectivas habrá y se escogerán más veces los individuos más aptos, haciendo la selección más dura (se favorece a los mejores individuos); a menor tamaño de los grupos ocurrirá lo contrario.

- **Selección por ruleta**. Se asigna a cada individuo un pedazo de una ruleta, como se observa en la figura 4.3. El tamaño del arco que le corresponde a cada individuo será proporcional a su aptitud. Esto quiere decir que, para un individuo con aptitud A le corresponde un pedazo de arco S y para un individuo con aptitud $2A$ le corresponde un pedazo de arco $2S$. De esta manera individuos con la mejor aptitud tendrán un mayor pedazo de la ruleta. La ruleta cuenta con un solo apuntador que, al girarse la ruleta, señala al individuo que formará parte de la población intermedia. La ruleta debe de girarse λ veces para completar la población intermedia. Este tipo de selección (proporcional a la aptitud), puede favorecer que algún superindividuo se apodere de la población, debido que el mejor individuo tiene mayor probabilidad de ser elegido por cada vuelta que se le da a la ruleta. Esta puede ser considerada como una selección "rigurosa".

- **Selección por jerarquización**. Para evitar la formación de super individuos, se propone asignar un valor al desempeño del peor individuo y otro valor, mayor al anterior, al desempeño del mejor individuo.

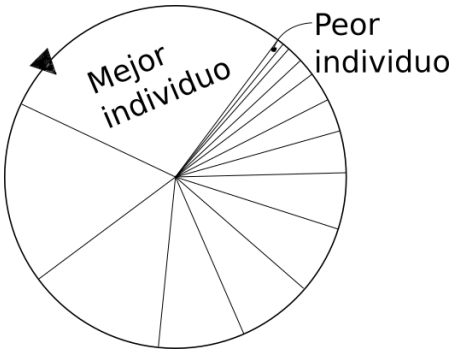

Figura 4.3: Ruleta de individuos

Para ello se acomodan los individuos de peor a mejor desempeño y se enumeran, siendo 1 el peor individuo. Se utiliza una función lineal (ecuación (4.6), Baker (1985)) para asignar el valor de desempeño al resto de individuos. Es esta ecuación min y max son los valores supuestos para el peor y el mejor individuo respectivamente, pos es la enumeración del individuo y μ es el número de individuos en la población. Una vez realizada la jerarquización se procede a utilizar la selección por ruleta, de esta manera se impide que un individuo se apodere de la mayor parte de la ruleta y la selección se vuelve más laxa.

$$V_{ind} = min + (max - min)\frac{pos - 1}{\mu - 1} \tag{4.6}$$

4.3. Operadores de cruzamiento

El cruzamiento es la operación donde los padres heredan su información genética a sus hijos. En la naturaleza es un proceso aleatorio y define las características que se pasarán a la siguiente generación. Podría decirse que los operadores de cruzamiento son una característica que distingue a los algoritmos genéticos del resto de los algoritmos evolutivos. Estos aprovechan la gran cantidad de información genética en su población, intercambiando información entre ellos y generando variedad genética en la población; pues sin el cruzamiento, la única fuente de variedad en la población se generaría a través de la mutación. La mayoría de las veces se suele utilizar una probabilidad de cruzamiento, aunque algunos diseñadores pueden forzar a los individuos a cruzarse siempre.

Hay diferentes operadores de cruzamiento, pero, a final de cuentas todos consisten en escoger de manera aleatoria un conjunto de subcadenas en los individuos que se intercambian entre sí. Por ejemplo, sean la cadena $a = a_1a_2...a_n$, y la cadena $b = b_1b_2...b_n$ las que se cruzarán. Se escogen al azar m subcadenas para intercambiar, por ejemplo si $m = 2$, serán dos subcadenas las que se intercambiarán entre sí. Se obtienen al azar cuatro enteros i_1, i_i, j_1, j_2 tales que $i_k < j_k$, $1 \leq i_k, j_k \leq n-1$, para k = 1,2, entonces se intercambia la subcadena $a_{i_1}a_{i_1+1}...a_{j_11}$ con la subcadena $b_{i_11}b_{i_1+1}...b_{j_1}$ y la subcadena $a_{i_2}a_{i_2+1}...a_{j_2}$ con la subcadena $b_{i_2}b_{i_2+1}...b_{j_2}$.

- **Cruzamiento en un punto**. Se ejemplifica en la figura 4.4. Se genera un número aleatorio entero de distribución uniforme, p, en [1,l-1]. Ese punto será la posición a partir del cual se hará el intercambio de subcadenas. En el ejemplo $p = 3$ y se hará el intercambio de la información de padres a partir de la posición 3 en adelante.

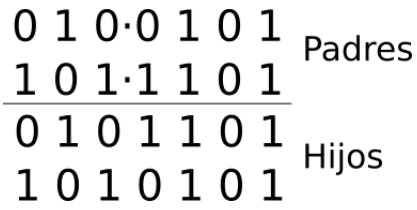

Figura 4.4: Cruzamiento en un punto

- **Cruzamiento en dos puntos**. Se ejemplifica en la figura 4.5. Es muy parecido al cruzamiento en un punto, sin embargo, aquí se generan dos números al azar en el intervalo [1,l-1], donde $p_1 \leq p_2$ y p_i, estos números definen la posición de inicio y fin de la subcadena que se intercambiará entre ambos individuos. En la figura 4.5 se obtivieron $p_1 = 2$ y $p_2 = 5$, por lo que la subcadena 001 del primer padre se cambiará por la subcadena 111 del segundo padre, respetando la posición de donde se obtuvieron.

- **Cruzamiento uniforme**. Ejemplificado en la figura 4.6. En él se recorre todo el código genético de un individuo, al mismo tiempo se van generando números aleatorios. Si el número generado es mayor al dado por la probabilidad de cruzamiento, se genera un intercambio de ese bit entre los individuos. En la imagen podemos ver que hay un intercambio de información en los bits 1,3 y 6.

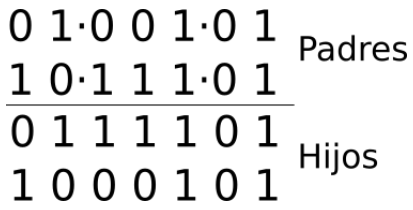

Figura 4.5: Cruzamiento en un dos puntos

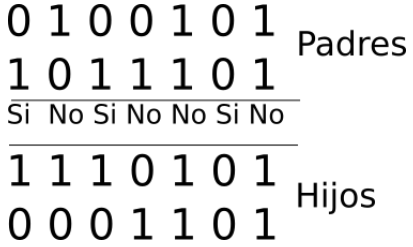

Figura 4.6: Cruzamiento uniforme

- **Cruzamiento anular**. Se ejemplifica en la figura 4.7. Cada genoma es considerado como un anillo y se numera cada bit del genoma. Se generan dos números aleatorios en el rango de la longitud de la cadena y se intercambian ambas partes. En el ejemplo se tienen que intercambiar los bits 1,6 y 7.

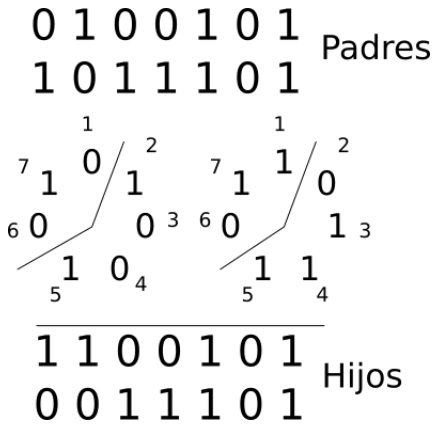

Figura 4.7: Cruzamiento anular

4.4. Mutación

La mutación es una parte esencial en un algoritmo evolutivo (como lo son los algoritmos genéticos). Los algoritmos genéticos lo usa para dar variedad a la población que no puede ser dada por el cruzamiento. Sin este operador el algoritmo podría quedar estancado, en caso de operadores de selección rigurosos, y nunca llegar al óptimo. Gracias a la mutación pueden existir individuos con características únicas que ninguno de sus padres tenía. En caso de utilizar codificación binaria, la mutación se refiere a cambiar el valor del bit por su complemento; es decir, si el bit tiene un valor de 0 se cambia por 1 y viceversa.

La mutación puede ser uniforme, es decir que todos los bits de una cadena tengan la misma probabilidad de mutar; o puede favorecer a partes de la cadena a mutar más que otras. En el caso de la mutación uniforme, el usuario define la probabilidad de mutación de cada bit. Podemos ver en la figura 4.8 que se mutan los bits 1 y 6 de la cadena.

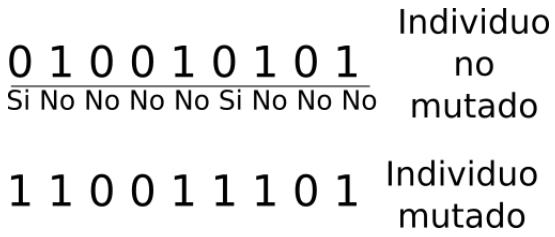

Figura 4.8: Mutación uniforme

4.5. Elitismo

El elitismo es una buena práctica que lleva al AG a converger al óptimo global, esto nos lo dice el teorema de Rudolph, el cual mencionan Reeves y Rowe [13, pp.124]. El elitismo asegura que en la población P_{i+1} se tendrá un individuo igual o mejor que en la población P_i.

4.6. Vasconcelos

La cantidad de individuos creados en la población de hijos puede ser diferente en cada adaptación del algoritmo genético. El número de individuos en la población inicial está representado por μ y el número de individuos de la población intermedia por λ. Cuando la población de hijos substituye

a la de padres, es decir, los mejores μ individuos de la población intermedia formarán la población $i+1$; se dice que es un algoritmo tipo (μ, λ). En cambio, cuando entre la población intermedia y al población i se escogen a los mejores μ individuos para formar la pobación $i+1$, el algoritmo es de tipo $(\mu + \lambda)$.

Figura 4.9: Algoritmo tipo (μ, λ)

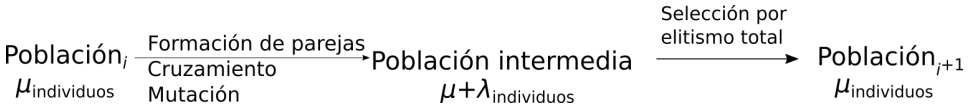

Figura 4.10: Algoritmo tipo $(\mu + \lambda)$

Kuri [14, pp.209–2011] utiliza un algoritmo genético tipo $(\mu + \lambda)$. En este tipo de algoritmos el tamaño de la población intermedia tendrá λ más individuos que la población inicial; se suele utilizar $\lambda = \mu$. Además, Kuri propone una formación determinista de parejas. En ella se acomodan todos los individuos por desempeño y se selecciona al mejor y el peor individuo, de manera que se empareja al individuo i con el individuo $\mu - i + 1$. A esta formación determinista de parejas, Kuri le llama la estrategia Vasconcelos.

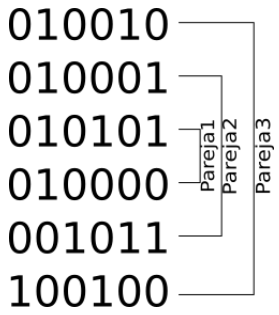

Figura 4.11: Formación de parejas determinista por desempeño

Esta es la manera en la que selecciona las parejas. Puede usarse cualquier operador de cruzamiento y de mutación. Sin embargo, para que el algoritmo funcione correctamente, tiene que utilizarse elitismo total para seleccionar la siguiente población. Este tipo de cruzamiento da gran variedad genética

a la población, sin embargo también puede estar generando individuos con bajo desempeño. Para asegurar que la población $i+1$ sea mejor o igual a la población i, se utiliza elitismo total. A la estrategia Vasconcelos unida con el elitismo total, Kuri le llama el modelo Vasconcelos. El algoritmo utilizado en este trabajo estará basado en el modelo Vasconcelos.

4.7. Implementación de un algoritmo genético

Es esta última parte del capítulo se explicará por medio de un pseudocódigo el algoritmo genético que se implementó para usarse en el presente trabajo. Fue codificado en el lenguaje de programación Python. Es muy utilizado por principiantes ya que tiene una estructura muy limpia y su sintaxis es muy intuitiva, sin embargo, es muy potente a la hora de crear códigos más complejos. Gracias a la gran comunidad activa en todo el mundo hay un sinfín de bibliotecas externas que pueden utilizarse. Existe una biblioteca llamada "*Distributed Evolutionary Algorithms in Python*" (*https://pypi.python.org/pypi/deap*), que incluye un módulo para usar un algoritmo genético. No se usó esta biblioteca, ya que no es el objetivo del trabajo. Todo se diseño y programó desde cero. Los algoritmos son los siguientes:

El primer algoritmo nos muestra cómo se genera la población inicial (algoritmo 4). Primero se calcula la longitud de la cadena binaria con la que se codificará cada variable, l. En seguida, por cada individuo se creará una cadena vacía, es decir una cadena sin números. Por cada variable de la que depende la función objetivo se obtendrá un número aleatorio de distribución uniforme entre 0 y el número máximo de subintervalos, que es $2^l - 1$. Ésto garantizará que la cadena con el mayor valor sea $1_1 1_2 1_3 .. 1_l$. Se convierte el entero a binario y, en caso de ser necesario, se completa con ceros a la izquierda hasta tener la longitud requerida. Por ejemplo, si el número binario obtenido de la conversión es 100101 y $l = 10$, entonces se tiene que completar con 4 ceros a la izquierda, al final obtenemos la cadena binaria de 10 bits de longitud 0000100101. Finalmente la cadena binaria obtenida se concatena con el individuo. Por ejemplo, si el individuo es una cadena vacía y se tienen 2 variables codificadas en números binarios de longitud $l = 10$, las cuales son 0000100101 y 1001011011; después de concatenar todas las cadenas, el individuo será una cadena binaria de 20 bits de longitud: 00001001011001011011. Es esta cadena la que se agrega a la lista de la población. Finalmente se calcula la aptitud de cada individuo de la nueva población por medio del algoritmo 8 (más adelante se detallará en el

Algoritmo 4 NuevaPoblación

Entrada: Número de individuos, Número de variables, a, b, Δx
Salida: Población

```
1: Población ← Lista vacía
2: Calcular la longitud de la cadena, l, ecuación (4.3).
3: Para i=1 hasta Número de Individuos Hacer
4:    Individuo ← Cadena vacía
5:    Para j=1 hasta Número de variables Hacer
6:       rand ← Número aleatorio de distribución uniforme en [0,2^l − 1]
7:       Bin ← Dec_a_Bin(rand)
8:       Si la longitud de Bin es diferente a l Entonces
9:          Completar con ceros a la izquiera para que la longitud de Bin sea
            igual a l
10:      Terminar Si
11:      Individuo=Individuo + Bin
12:   Terminar Para
13:   Agregar Individuo a Población
14: Terminar Para
15: Aptitud Población ← Calcular Aptitud(Población)
16: Regresar Población, Aptitud Población
```

funcionamiento de este algoritmo).

El siguiente algoritmo (algoritmo 5) formará las parejas para cruzamiento y creará la población intermedia. Primero se ordena la población con base en su aptitud, ésto para poder utilizar la estrategia Vasconcelos. Después se empiezan a formar las parejas, de modo que el individuo i se empareje con el individuo $\mu - i + 1$. Una vez creada la pareja se selecciona las posiciones de los puntos que delimitarán las subcadenas a intercambiar (ver el operador de cruzamiento en dos puntos, sección 4.3). Es importante recalcar que las posiciones en las que se intercambian las subcadenas deben de ser las mismas, es decir que si, por ejemplo, $p_1 = 5$, no debe colocarse el principio de la subcadena en otra posición. Finalmente se intercambian las subcadenas en su respectiva posición, ésto creará dos nuevos individuos que formarán parte de la población intermedia.

Una vez obtenida la población intermedia, se someterá a mutación cada individuo. Cada bit del individuo tendrá una probabilidad p_m de mutar, es decir, de cambiar su valor por su complemento. Para ello se toma la cadena binaria del individuo y se recorre cada bit, generando un número aleatorio de distribución uniforme en el intervalo [0,1] por cada bit. Si el número generado es menor a p_m, se cambia el valor del bit por su complemento.

Como se mencionó anteriormente, se utilizará un algoritmo basado en el modelo Vasconcelos. El modelo Vasconcelos utiliza elitismo total para elegir a los μ mejores individuos entre la población i y la población intermedia. Aquí se utilizará una forma de elitismo determinista, donde se ordena la población i y a la población intermedia de acuerdo a su aptitud. Después se compara al padre m con el hijo m, el mejor de ellos formará parte de la nueva población. Esto asegura que el mejor individuo de ambas poblaciones pase a la nueva población, además que otorga mayor variedad genética a la nueva población.

En este trabajo se utilizará el algoritmo genético con dos propósitos: encontrar los parámetros que controlen de manera satisfactoria el reactor CSTR, modelado anteriormente (ver sección 2.1), ante una perturbación; y entrenar a una red neuronal. Para la obtención de los parámetros se diseñó una función objetivo, algoritmo 8, a la cual se le alimentan 2 datos: la perturbación que se someterá al sistema y los parámetros de control. Las perturbaciones, de la concentración de entrada del óxido de propileno, serán de tipo rampa, escalón y pulso. La función objetivo ejecutará la simulación del sistema en un intervalo de 1000 segundos, de los cuales los primeros 10 segundos corresponden a la perturbación y el resto del tiempo es para conocer cómo se comporta el sistema dada la perturbación. Una vez finalizada la simulación se evalúa una función de desempeño, ecuación (4.7). Ésta fun-

Algoritmo 5 Formación de parejas y cruzamiento

Entrada: Población, Función Objetivo
Salida: Población Intermedia

1: Población Intermedia ← Lista vacía
2: Población ← Ordenar por aptitud(Población)
3: **Para** i=1 **hasta** $\mu/2$ **Hacer**
4: Intividuo1 ← Población(i)
5: Individuo2 ← Población($\mu - i + 1$)
6: p_1 ← Número aleatorio de distribución uniforme en [1,l-1]
7: p_2 ← Número aleatorio de distribución uniforme en [1,l-1] que sea mayor a p_1
8: subcadena1 ← Subcadena del Individuo1 que comience en la posición p_1 y termine en p_2
9: subcadena2 ← Subcadena del Individuo2 que comience en la posición p_1 y termine en p_2
10: Hijo1 ← Individuo1 intercambiando la subcadena1 por la subcadena2
11: Hijo2 ← Individuo2 intercambiando la subcadena2 por la subcadena1
12: Agregar Hijo1 a Población Intermedia
13: Agregar Hijo2 a Población Intermedia
14: **Terminar Para**
15: **Regresar** Población Intermedia

Algoritmo 6 Mutación

Entrada: Población Intermedia, p_m
Salida: Población Intermedia

1: **Para** cada Individuo **en la** Población Intermedia **Hacer**
2: **Para** bit **en el** Individuo **Hacer**
3: r ← Número aleatorio de distribución uniforme en [0,1]
4: **Si** $r < p_m$ **Entonces**
5: Cambiar el valor del bit por su complemento
6: **Terminar Si**
7: **Terminar Para**
8: **Terminar Para**
9: **Regresar** Población Intermedia

Algoritmo 7 Elitismo

Entrada: Población, Población Intermedia, Aptitud Población
Salida: Nueva Población

```
1: Nueva Población ← Lista vacía
2: Aptitud Población Intermedia ← Calcular Aptitud(Población Intermedia)
3: Población ← Ordenar por aptitud(Población)
4: Población Intermedia ← Ordenar por aptitud(Población Intermedia)
5: Para i=1 hasta Número de individuos Hacer
6:    Si Aptitud Población(i)>Aptitud Población Intermedia(i) Entonces
7:       Agregar Población(i) a Nueva Población
8:    Si no
9:       Agregar Población Intermedia(i) a Nueva Población
10:   Terminar Si
11: Terminar Para
12: Regresar Nueva Población
```

ción fue diseñada considerando que el sistema debe alcanzar lo más rápido el estado estacionario sin tener cambios bruscos en sus variables y modificando lo menos posible las condiciones de operación, en este caso el flujo de agua de enfriamiento. El término integral nos da información de qué tan rápido llega al estado estacionario, mientras que los términos diferenciales nos dicen si hubo cambios bruscos en las variables (en este caso el flujo de agua de enfriamiento y la temperatura del interior del reactor). Finalmente el término exponencial impide que los parámetros del controlador tomen valores muy grandes, esto es porque se requiere llegar al estado estacionario sin hacer grandes cambios en el sistema.

Algoritmo 8 $f_{Objetivo}$

Entrada: Perturbación, parámetros

```
1: Ejecutar la simulación del reactor CSTR, utilizando la perturbación y lo parámetros.
2: Aptitud ← Evaluar ecuación (4.7).
3: Regresar Aptitud
```

$$f = \int_0^t (T - T_{sp})dt + \frac{d}{dt}F_A + \frac{d}{dt}T + 10^{(K_C+\tau_D+\tau_I)\frac{1}{4}} \tag{4.7}$$

El algoritmo genético que se utilizará para encontrar los parámetros de

control del CSTR es el algoritmo 9. En él, cada individuo será una cadena binaria compuesta de 3 variables codificadas, las cuales corresponden a los parámetros de control: k_c, τ_D y τ_I^{-1}. Las características de operación del algoritmo genético utilizado son las siguientes:

Número de individuos = 30
Número de generaciones = 100
Límite mímino de búsqueda = 0
Límite máximo de búsqueda = 10
Δx_{max} = 0,01

Éste algoritmo nos regresará los parámetros encontrados que hacen que la ecuación (4.7) arroje valores muy pequeños. No se puede asegurar que son los valores óptimos, sin embargo, son valores que se acercan a los óptimos.

Algoritmo 9 Búsqueda de los mejores parámetros de control por medio de un AG

Entrada: Número de variables, Límites de búsqueda, Número de individuos, Número de generaciones, Δx, Función objetivo, Perturbación
Salida: Pesos

1: Obtener de manera aleatoria la población inicial, $P_0 = (p_1, p_2, ..., p_\mu)$.
2: **Para** cada p_i **en** P_0 **Hacer**
3: $p_{i,decod} \leftarrow$ decodificación del individuo p_i, algoritmo 3.
4: Calcular $f_{Objetivo}$(Perturbación, $p_{i,decod}$).
5: **Terminar Para**
6: $i = 0$
7: **Mientras** no se cumpla la condición de terminación **Hacer**
8: $P_{1/2} \leftarrow$ Cruzar p_1 con p_μ, p_2 con $p_{\mu-1}$, ... (Algoritmo 5)
9: $P_{1/2} \leftarrow$ Mutación($P_{1/2}$) (Algoritmo 6)
10: **Para** cada p_i **en** $P_{1/2}$ **Hacer**
11: $p_{i,decod} \leftarrow$ decodificación del individuo p_i, algoritmo 3.
12: Calcular $f_{Objetivo}$(Perturbación, $p_{i,decod}$).
13: **Terminar Para**
14: $P_{i+1} \leftarrow$ Elitismo(P_i,$P_{1/2}$), algoritmo 7.
15: $i = i + 1$
16: **Terminar Mientras**

Después de haber encontrado los valores de K_C, τ_D y τ_I^-1 éstos son almacenados junto con la perturbación dada. Posteriormente se utilizarán

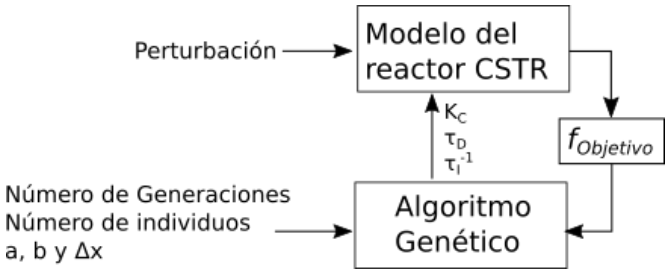

Figura 4.12: Diagrama de flujo para encontrar los parámetros del controlador

todos estos datos para entrenar a la red neuronal. Éste proceso puede ser apreciado en el diagrama de flujo en la figura 4.12.

Capítulo 5

Redes multicapa de perceptrones

Desde siempre nos ha maravillado el funcionamiento del cuerpo humano y han habido grandes avances tecnológicos al imitar lo que la naturaleza ha logrado. El cerebro humano es una de las máquinas (por llamarlo así) más complejas que existen. Sin embargo está compuesto de elementos más simples que han sido objeto de gran cantidad de estudios intentando descifrar su funcionamiento. Estos elementos son las neuronas. En 1943 salió a la luz un artículo escrito por Warren McCulloch y por Walter Pitts [15], un neurocirujano y un matemático, quienes propusieron un modelo matemático de cómo las neuronas trabajan. Mas tarde este modelo podría ser entrenado y adquiriría el nombre de perceptrón. Con el avance en la fabricación de computadoras, en los años 50 se pudo simular por primera vez una red neuronal. Antes de esto todos los trabajos sobre neuronas se simulaban con circuitos eléctricos. Desde entonces se han desarrollado diferentes modelos como ADALINE o MADALINE desarrollados por Widrow y Hoff en 1962.

Una célula neuronal recibe información por las dentritas, la procesa mediante el cuerpo celular y transmite nueva información a través de su axón. La neurona artificial, o neurodo, puede ser representado gráficamente en la figura 5.1. Al igual que una célula neuronal, el neurodo puede recibir señales, procesarlas y emitir una señal propia. En donde $x_1, x_2, ..., x_m$ son las señales de entrada, que pueden ser los datos de entrenamiento o prueba, o las salidas de neurodos de capas inferiores; $w_1, w_2, ..., w_m$ son los factores de eficiencia sináptica, también llamados pesos, de la neurona i; σ es una función de ponderación (ecuación 5.1) y y es una función de activación o transferencia (por

ejemplo, la ecuación 5.2). Para que un neurodo se active, la ponderación de las entradas deberá ser igual o mayor que un umbral, representado por Θ. En redes multicapa de perceptrones, éste umbral es arrojado por un neurodo de polarización (o conocido como *bias unit*, en inglés), que forma parte de las entradas de la neurona artificial. A este neurodo *bias* generalmente se le alimenta un valor de -1.

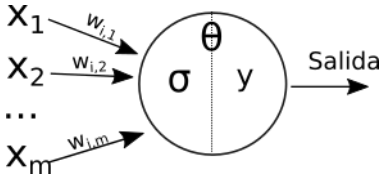

Figura 5.1: Representación gráfica de una neurona artificial

$$\sigma_i = \sum_{j=0}^{m} w_{i,j} x_j \tag{5.1}$$

$$y_i = \frac{1}{1 + e^{-\sigma}} \tag{5.2}$$

En la red neuronal, la importancia de la conexión entre neuronas y la cantidad de información transferida está representada por su peso. Por ejemplo, si el peso entre el neurodo i y el neurodo j, $w_{i,j}$, tiene un valor muy grande respecto al resto de pesos en la misma capa, el neurodo j podría ser el que, con base en la señal que envía, determina si el neurodo i se activará o no. Podemos decir entonces que una red neuronal artificial es función de los pesos sinápticos de la misma.

Las redes de dos capas solo pueden resolver problemas linealmente separables, esto es, aquellos en los que los puntos de una clase pueden ser separados de los otros mediante una linea recta (o un plano o hiperplano dependiendo de la dimensionalidad). Para resolver problemas más complejos hace falta una red de tres capas. En 1969 Marvin Minsky y Seymour Papert publicaron el libro *Perceptrons* [16], en el que recalcan esta limitación.

No fue hasta 1986, cuando Rumelhart [17] diera a conocer un nuevo algoritmo para entrenar una red neuronal multicapa de perceptrones. El nuevo algoritmo fue llamado "*Backpropagation*". Este método usa un descenso por la dirección negativa del gradiente para disminuir la función error, que se define en la ecuación (5.6). Son muy utilizadas para reconocer patrones que, hasta la fecha, sólo un ser vivo podría reconocer, además que pueden ajustar un número indeterminado de funciones al mismo tiempo.

5.1. Propagación hacia adelante

La arquitectura de una red nos dice cómo estarán relacionados los neurodos, es decir, la posición de éstos y sus interacciones. En el tipo de arquitectura que se manejará, cada perceptrón de una capa está conectado con todos los de la inmediata siguiente. En cada conexión hay una transferencia de información entre ambos perceptrones en una sola dirección. Se tiene que aclarar que la capa de entrada está constituida por simples distribuidores, que se encargan de repartir los datos de entrada entre los perceptrones de la segunda capa. Las capas posteriores están constituidas por perceptrones con una función de activación sigmoidal.

La propagación hacia adelante (o *Forward Propagation*) es el proceso en donde se alimenta a la red una serie de datos en la capa entrada de la red multicapa de perceptrones (RMP) y se hace el cómputo de la función de transferencia de cada neurodo de la red. Para ilustrar este proceso, se utilizará la figura 5.2, utilizando la notación de Henseler [19]. Los pesos se pueden distinguir entre sí de la siguiente forma: Un primer subíndice nos indica el neurodo al que entra la información; un segundo subíndice que indica el neurodo del que sale; un tercer índice, que se colocará como superíndice, que indica la capa del neurodo al que entra la información. Para el ejemplo, el peso $w^2_{1,1}$ pertenece a la conexión entre el primer neurodo de la capa 2 y el primero de la capa anterior; $w^3_{3,2}$ pertenece a la conexión entre el tercer neurodo de la capa 3 y el segundo de la capa anterior.

Para calcular la función activación de cada neurodo se calcula primero la suma ponderada con la ecuación (5.4). Una vez calculada σ^p_i, se obtiene y^p_i; esto se repite por cada neurodo de cada capa.

$$y^p_i = \frac{1}{1 + e^{-\sigma^p_i}} \tag{5.3}$$

$$\sigma^p_i = \sum_{j=1}^{m_{p-1}} y^{p-1}_j w^p_{i,j} \tag{5.4}$$

Donde

σ^p_i es la suma ponderada de las entradas del neurodo i de la capa p.

y^p_i es función de activación del neurodo i de la capa p

$w^p_{i,j}$ es el peso que va del neurodo j al neurodo i de la capa p.

m_k es el número de neurodos de la capa m

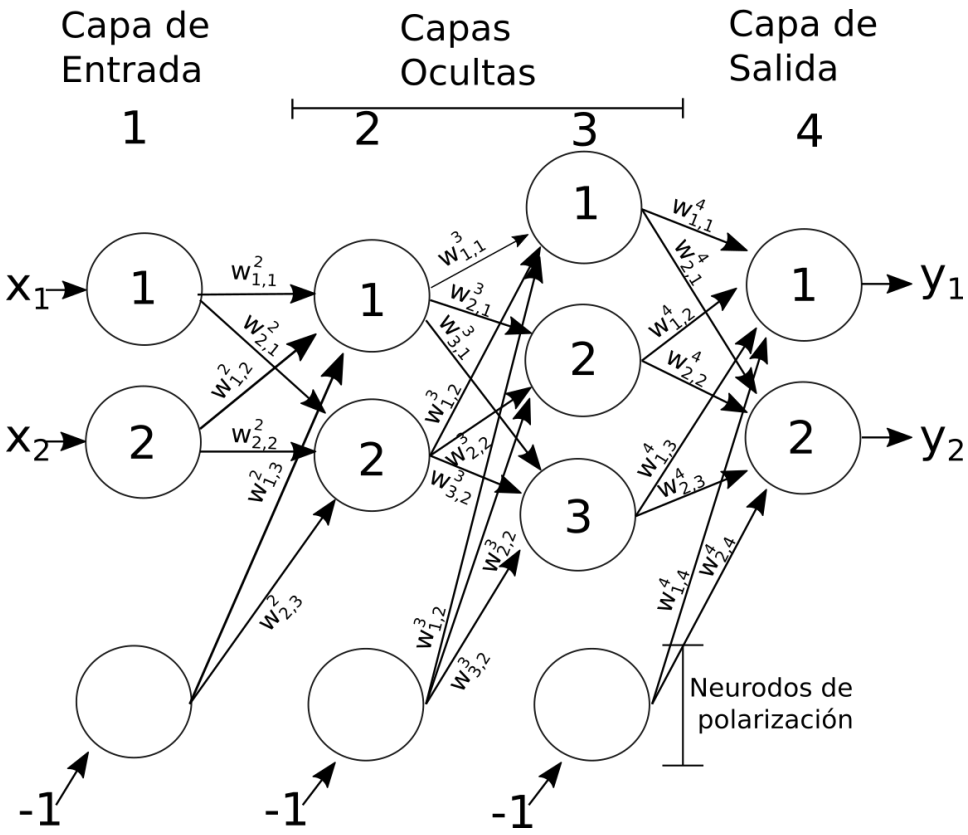

Figura 5.2: Red neuronal a entrenar

Para la ecuación (5.4), cuando se trata de la segunda capa, el valor de y_i^p será el valor de x_i, ya que la primera capa, como se dijo anteriormente, es sólo una capa distribuidora que no modifica el valor de las entradas. Ésto quiere decir que $y_i^1 = x_i$. Sin embargo, esto no pasa para nos neurodos de polarización, que evalúan la función de activación donde $\sigma = -1$.

Es importante mencionar que los valores x alimentados a la red y los valores Y deben estar normalizados, es decir, deben tener valores entre 0 y 1. Para ello se utiliza la ecuación (5.5). En donde h es el valor mínimo que x puede tomar y g es el valor máximo. Para el caso de los parámetros del controlador, es sencillo: no pueden ser menores a 0 ni mayores a 10, criterio fijado en el capítulo anterior. Sin embargo, en el caso de las perturbaciones, el mínimo es 0 (ya que no puede haber concentración menor a 0) y el mayor puede ser un número mayor a 2 (que es el valor en estado estacionario),

sin embargo no está fijado. Se recomienda utilizar todos los datos de las perturbaciones para encontrar el valor con la mayor magnitud, a ese valor se le podría sumar una pequeña cantidad como 0.5, así se puede asumir que la mayoría, por no decir todas las perturbaciones normalizadas serán menores que 1.

$$\kappa(x) = \frac{x - h}{g - h} \tag{5.5}$$

5.2. Entrenamiento

El aprendizaje de una red neuronal está dado por la modificación de sus pesos. Esto quiere decir que una red aprende cuando los valores de los pesos hacen que las salidas de la red sean muy parecidas o iguales a las que se esperan. Para ello una red debe de ser entrenada. El entrenamiento puede ser de dos maneras: supervisado y no supervisado. En el primero se entrena a la red comparándola con el resultado esperado utilizando una función error, como la presentada en la ecuación (5.6), y modificando sus parámetros (pesos) hasta minimizar la función error. En el entrenamiento no supervisado la red utiliza otro tipo de herramientas estadísticas y es ideal para el agrupamiento de datos (o *clustering* en inglés) por su tendencia a auto-organizarse. En este trabajo se utilizará el entrenamiento supervisado, así que se utilizará un algoritmo optimizador para minimizar la función error modificando los valores de los pesos.

$$E = \frac{1}{n} \sum_{i=1}^{m_N} (y_i^N - Y_i)^2 \tag{5.6}$$

Donde:

- m_N es el número de perceptrones en la capa de salida
- N es el número de la capa de salida
- y_i^N es la salida del *i-ésimo* perceptrón en la capa N
- n es el número de datos de entrenamiento de la red
- E es la medida del error. Básicamente mide la diferencia entre el vector obtenido de la red y el deseado, Y
- Y_i es el vector de entrenamiento, contiene los valores deseados de y^N

El algoritmo de entrenamiento usado es un algoritmo genético, en donde cada individuo es una cadena binaria con la codificación de cada peso de

la red. En el presente trabajo, se entrenará una red de 100 neurodos de entrada, tres capas ocultas: la primera con 20 neurodos, la segunda con 10 y la tercera con 5; y 3 neurodos de salida, uno por cada parámetro del controlador PID. Con esta arquitectura se deberán ajustar 2285 pesos para minimizar la función error. La longitud de la cadena codificada de cada peso, considerando un $\Delta x = 0{,}01$ y un espacio de búsqueda entre -5 y 5, está dada por la ecuación (4.3):

$$l = \lceil log_2 \left(\frac{b-a}{\Delta x} + 1 \right) \rceil = \lceil log_2 \left(\frac{5-(-5)}{0{,}01} + 1 \right) \rceil = \lceil 9{,}97 \rceil = 10$$

Esto quiere decir que cada variable, en este caso cada peso, estará representada por un número binario de 10 bits, lo cual genera un genoma de 22850 bits y 2^{22850} posibles combinaciones, un espacio de búsqueda inmenso al que puede enfrentarse un algoritmo genético.

El algoritmo de entrenamiento está representado en el algoritmo 10. Cada individuo de la población está constituido por una lista, W_i, que contiene todos los pesos con los que cuenta la red. E_i es la aptitud del individuo W_i, en este caso es la evaluación de la función error, dados los pesos en $W_{i,decod}$.

Algoritmo 10 Entrenamiento de una red multicapa de perceptrones por un AG

1: Obtener de manera aleatoria la población inicial, $P_0 = (W_1, W_2, ..., W_\mu)$
2: **Para** cada W_i **en** P_0 **Hacer**
3: $W_{i,decod} \leftarrow$ Decodificación del individuo W_i, algoritmo 3.
4: Obtener $E_i(W_{i,decod})$, algoritmo 11.
5: **Terminar Para**
6: $i = 0$
7: **Mientras** no se cumpla la condición de terminación **Hacer**
8: $P_{1/2} \leftarrow$ Cruzar W_1 con W_μ, W_2 con $W_{\mu-1}$, ...
9: $P_{1/2} \leftarrow$ Mutación($P_{1/2}$)
10: **Para** cada W_i **en** $P_{1/2}$ **Hacer**
11: $W_{i,decod}$= Decodificación del individuo W_i, algoritmo 3.
12: Obtener $E_i(W_{i,decod})$, algoritmo 11.
13: **Terminar Para**
14: $P_{i+1} \leftarrow$ Elitismo(P_i,$P_{1/2}$), algoritmo 7.
15: $i = i + 1$
16: **Terminar Mientras**

En la figura 5.3 se puede observar el flujo del proceso de entrenamiento.

Los datos de entrenamiento de la red serán tomados de los resultados obtenidos por el algoritmo genético en la búsqueda de los mejores parámetros

Algoritmo 11 Cálculo de E

Entrada: x, Y, W_{decod}

1: $y^N \leftarrow$ Ejecutar la propagación hacia adelante con los pesos W_{decod} y los datos de entrenamiento x, Y.
2: $E \leftarrow$ Calcular el error, ecuación (5.6).
3: **Regresar** E

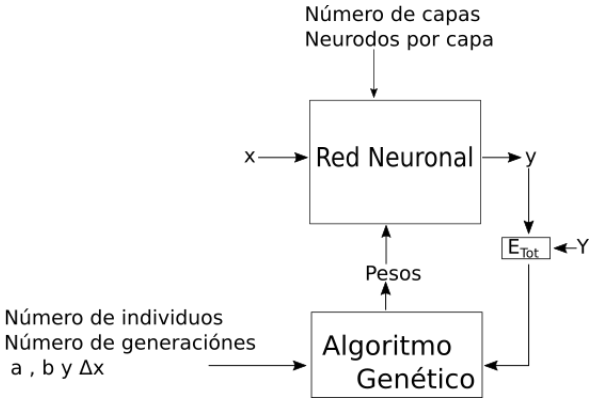

Figura 5.3: Diagrama de flujo del proceso de entrenamiento

del controlador PID que compensen una perturbación dada. La perturbación n consta de 100 valores de concentración de entrada al reactor CSTR; el intervalo de tiempo de la simulación es de 0.1 segundos por lo que el tiempo que representan esos 100 valores es de 10 segundos. Para el entrenamiento se alimentan los 100 valores en la capa de entrada de la RNA, es por esto que la capa de entrada consta de 100 neurodos. Luego de haber hecho la propagación hacia adelante se calculó la función error, utilizando los valores de los parámetros correspondientes a esa perturbación: $Y^{(n)} = [K_C^{(n)}, \tau_D^{(n)}, \tau_I^{(n)}]$. Utilizando este método, se hace que la red aprenda los valores de K_C, τ_D y τ_I que corresponden a esa perturbación. De esta manera cuando se le alimenten valores que se asemejen a una perturbación que se entrenó, la RNA arrojará valores parecidos a los requeridos por el sistema para ese tipo de perturbación.

5.3. Predicción de parámetros del controlador

Para la predicción de los parámetros, se utilizará una lista de 100 valores obtenidos, de la concentración de entrada del óxido de propileno, a lo largo

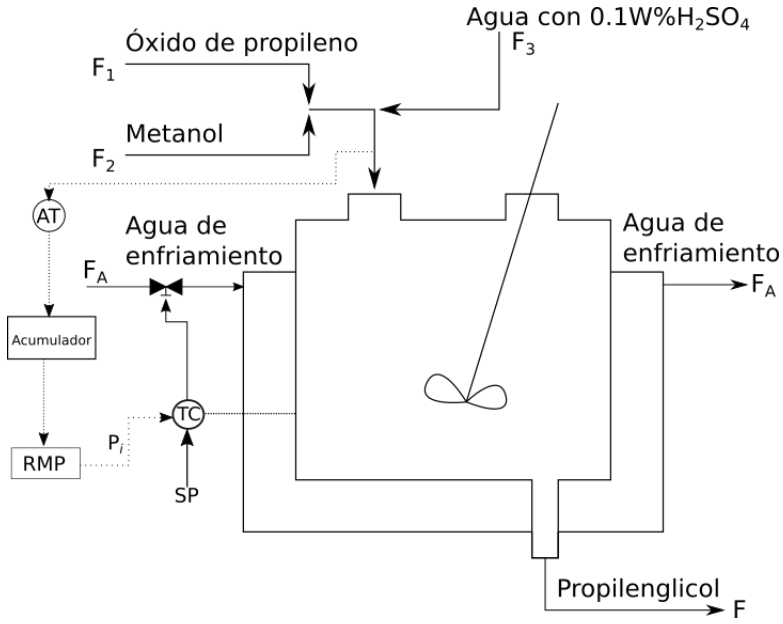

Figura 5.4: Esquema de la utilización de la red neuronal.

de la simulación. Podemos ver en la figura 5.4 que hay un transmisor de concentración (AT) que manda la señal a un acumulador. Éste acumulador almacena solamente 100 datos, una vez que tiene 100 registros, manda los datos a la Red Multicapa de Perceptrones (RMP) que genera el vector $P_i = [K_C, \tau_D, \tau_I]$ para ser ingresado en el control de temperatura a la vez que el *set point* (SP), el cual ya había sido establecido. El transmisor de concentración lee la concentración cada 0,1s y el acumulador, si ya tiene 100 registros, elimina el más antiguo y agrega el nuevo. Así, la RMP va modificando los parámetros en dependencia de la perturbación que es leída cada 0,1s.

Capítulo 6

Resultados

6.1. Curvas de calor generado y calor retirado

En estado estacionario, puede plantearse el balance de energía en dos partes: el calor que es generado en el reactor y el calor que es retirado del reactor. Puede darse el caso que, aunque reciba la misma corriente de alimentación y tenga el mismo volumen, opere en uno o dos estados estacionarios; cuál de los dos, dependerá de las condiciones iniciales del reactor antes de que alcance dicho estado estacionario. Esto puede ocurrir cuando la curva de calor retirado intersecta la curva de calor generado en más de un punto. Así, generando ambas curvas (la del calor generado y la del calor retirado), podemos determinar dónde se encuentran esos puntos. Las curvas serán generadas por las siguientes ecuaciones.

$$\dot{Q}_{gen} = -\Delta H_R V k C_{OP} = -C_{OP,e} \Delta H_R V k (1 - \bar{x}_{OP}) \tag{6.1}$$

$$\dot{Q}_{ret} = F\rho Cp(T_0 - T) + UA(T_A - T) + \Delta H_{dil} F\rho \tag{6.2}$$

$$\dot{Q}_{max} = -FC_{OP,e}\Delta H_R \tag{6.3}$$

En donde:

$\dot{Q}_{gen}$ es el calor generado por la reacción

$\dot{Q}_{ret}$ es el calor que se retira del reactor por medio de las corrientes de alimentación y salida y el retirado por el sistema de enfriamiento

$\dot{Q}_{max}$ es el calor que se generaría si reacciona todo el reactivo que entra al reactor

Se puede adimensionalizar el calor de reacción y el retirado dividiendo ambos entre $\dot{Q}_{max}$, de esta manera nos quedarán las fracciones de calor retirado y calor generado. La fracción de calor generado depende directamente de la conversión (x). Esto quiere decir que cuando $x = 0$ (no hay reacción), el calor generado será igual a $0 kcal/s$. Sin embargo, cuando $x = 1$ (la reacción se llevó al 100 %), el calor generado será $\dot{Q}_{max}$. Es por esto que

$$x = x_{Qgen} \tag{6.4}$$

Entonces la fracción de calor generado es

$$\begin{aligned} x_{Qgen} &= \frac{\dot{Q}_{gen}}{\dot{Q}_{max}} \\ x_{Qgen} &= \frac{-VkC_{OP}\Delta H_R}{-FC_{OP,e}\Delta H_R} \\ x_{Qgen} &= \frac{\tau k C_{OP,e}(1 - x_{Qgen})}{C_{OP,e}} \\ x_{Qgen} &= \tau k - \tau k x_{Qgen} \\ x_{Qgen} &= \frac{\tau k}{1 + \tau k} \end{aligned} \tag{6.5}$$

Y la fracción de calor retirado será

$$x_{Qret} = \frac{\dot{Q}_{ret}}{\dot{Q}_{max}} \tag{6.6}$$

Para el problema se obtiene la figura 6.1. Como se puede observar hay un solo punto de intersección y éste es estable para las condiciones que se mencionan en el planteamiento del problema. Si el volumen y los flujos de entrada del reactor permanecen constantes, al igual que el área de transferencia y las temperaturas de entrada de agua de enfriamiento y de reactivos, este punto puede moverse al cambiar la temperatura de salida del enchaquetado. Si llegase a subir la temperatura del reactor, el agua de enfriamiento absorberá mayor calor y cambiará su temperatura de salida, lo que produciría que la curva de calor retirado cambie su posición, intersectando la curva de calor generado en otro punto. Esto afectará al proceso si la temperatura dentro del reactor aumenta a más de 50°C. Ya que la cantidad de calor generado depende de la concentración a la entrada del óxido de propileno, una perturbación en esta condición generará cambios en la temperatura interna del reactor.Ésta fue una de las razones por las que se decidió utilizar perturbaciones en la concentración de entrada del óxido de propileno.

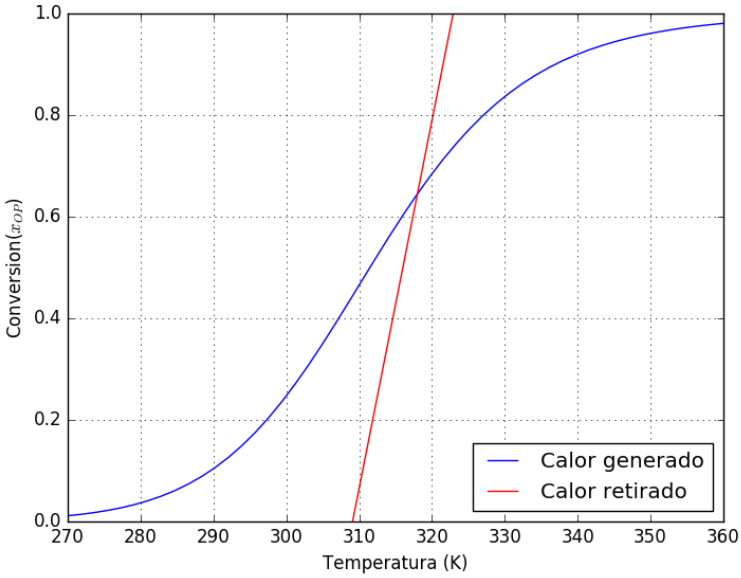

Figura 6.1: Calor generado y calor retirado contra la temperatura

6.2. Diagramas espacio-fase

Los diagramas espacio-fase nos dan información de cómo se comporta el sistema a partir de sus condiciones iniciales. Para ello se tiene que dar las condiciones iniciales de composición y temperatura, luego hacer la simulación dinámica para ver cómo se comporta el sistema. Se resuelven las ecuaciones (2.3) y (2.10) suponiendo una sola condición inicial de conversión y diferentes temperaturas. En las figuras 6.2, 6.3 y 6.4 se muestran tres ejemplos para tres condiciones iniciales de conversión distintas.

6.3. Obtención de parámetros de control

Como se indicó anteriormente, el sistema tiene un punto de operación estable. Las condiciones iniciales de conversión y temperatura son:

$$T = 318K$$
$$\bar{x}_{OP} = 0{,}6449$$

Si se inicia la simulación con estas condiciones iniciales y se añade una perturbación tipo escalón aumentando la concentración de entrada del óxido

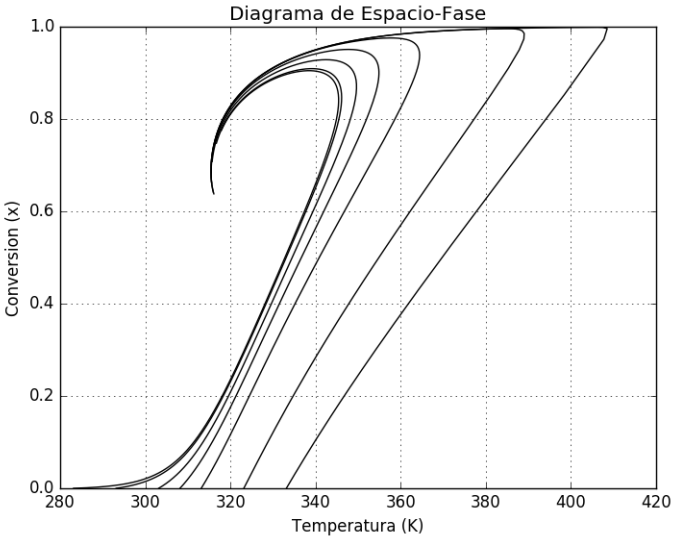

Figura 6.2: Diagrama espacio-fase con $\bar{x}_{OP,0} = 0$

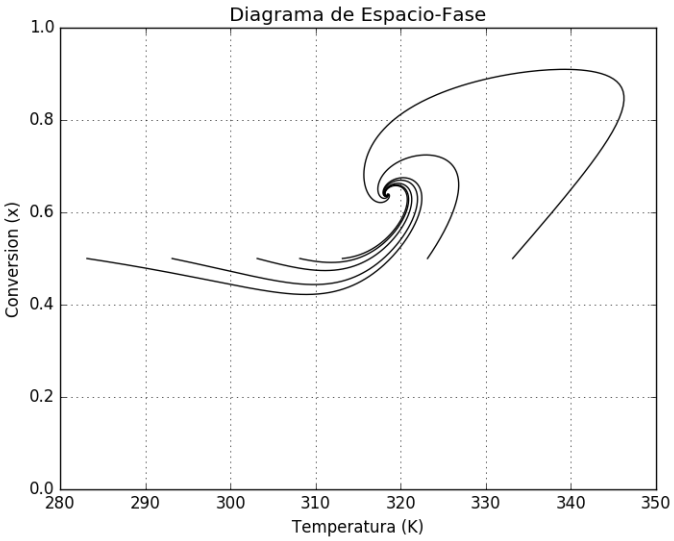

Figura 6.3: Diagrama espacio-fase con $\bar{x}_{OP,0} = 0{,}5$

de propileno, el sistema sin controlador se sale de su estado estable y nos

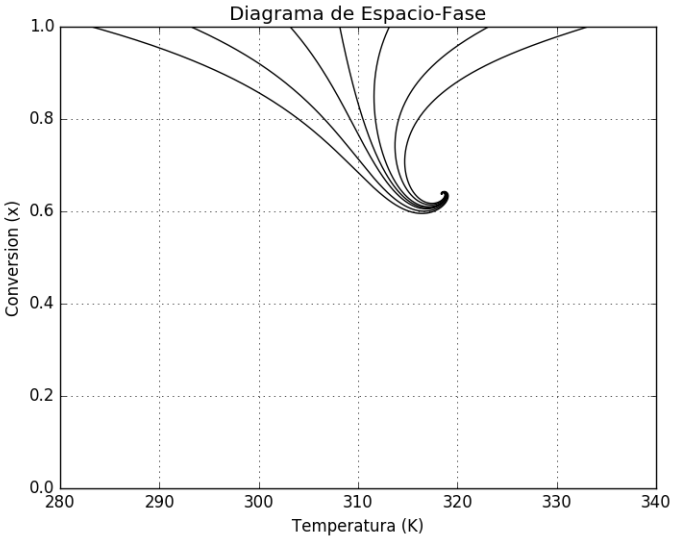

Figura 6.4: Diagrama espacio-fase con $\bar{x}_{OP,0} = 1$

lleva a temperaturas mayores de 50°C (223.15K), como se puede observar en la figura 6.5. Es por ello que se propone usar control predictivo para que el sistema regrese rápidamente al punto estable.

En la búsqueda de los valores de los parámetros se simularon 100 perturbaciones diferentes en la alimentación del óxido de propileno; todas ellas de tipo escalón, rampa o pulso, cambiando diferentes parámetros como la magnitud de la perturbación y su pendiente (en el caso de la perturbación tipo rampa). Ejemplos de ellas se pueden apreciar en las figuras 6.6, 6.7, 6.8. Además se agregó un poco de ruido a las perturbaciones, esperando simular el error producido por los instrumentos de medición. Es importante mencionar que las perturbaciones fueron creadas aleatoriamente, cambiando parámetros como la pendiente y el ancho de la rampa, la magnitud del escalón y del pulso; asimismo también se consideraron las magnitudes negativas, generando escalones y pulsos hacia abajo.

Es entonces en este momento que se procedió a utilizar el algoritmo 9, definido en la sección 4.7. Hay que recordar que el trabajo del AG será minimizar la función objetivo, esto lo hará modificando los parámetros del controlador. Se alimentó cada una de las perturbaciones y se obtuvo un vector de parámetros "*ideales*" para esa perturbación. Finalmente se almacenaron tanto la perturbación como los resultados obtenidos por el AG.

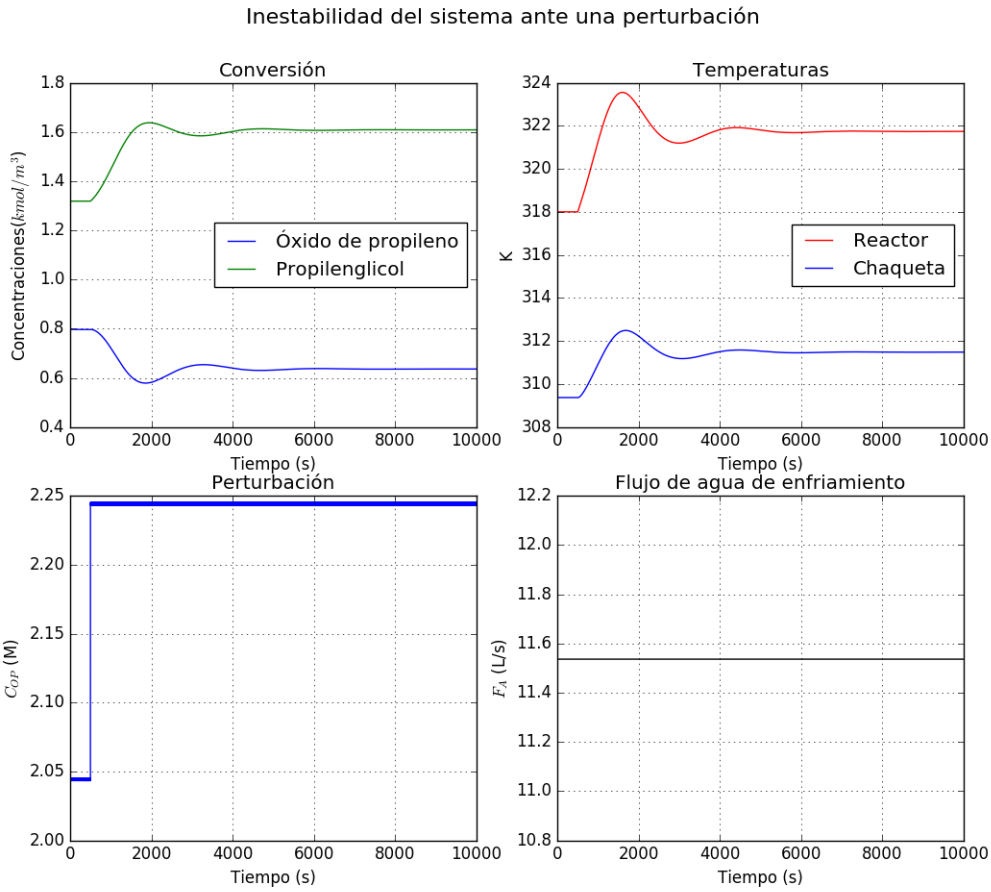

Figura 6.5: Inestabilidad del sistema de reacción ante una perturbación tipo escalón en la concentración de entrada del óxido de propileno de 0.2M

6.4. Entrenamiento de la red neuronal artificial

Con los datos obtenidos anteriormente por el AG, se procede a entrenar la RMP. Las características de la red son las siguientes:

Número de neurodos en la capa de entrada $= 100$

Número de capas ocultas $= 3$

Número de neurodos en la capa de salida $= 3$

Número de neurodos por capa oculta $= [20, 10, 5]$

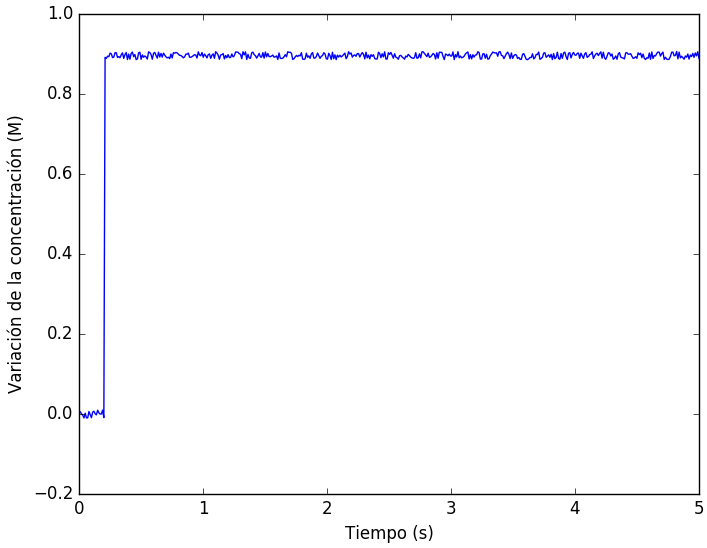

Figura 6.6: Perturbación tipo escalón

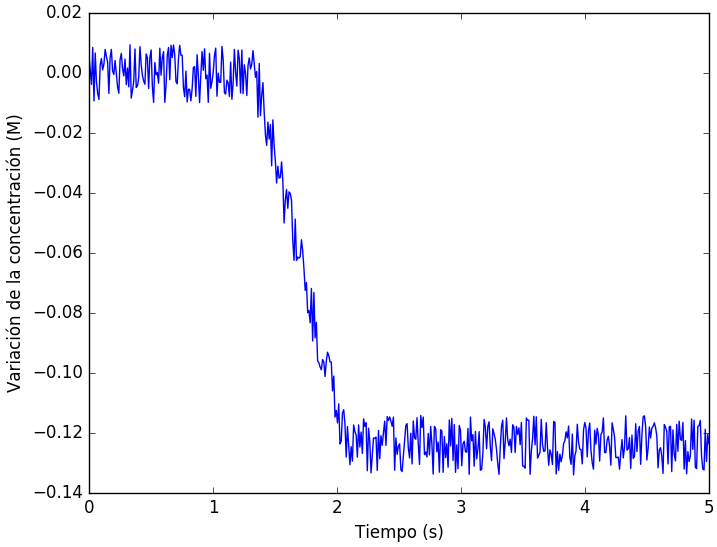

Figura 6.7: Perturbación tipo rampa

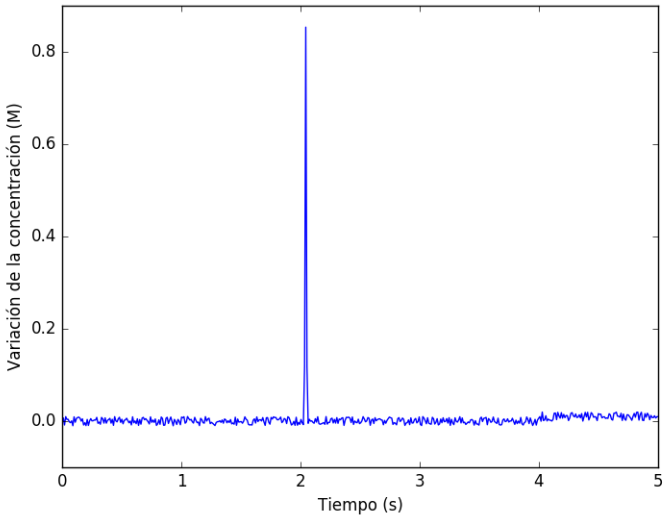

Figura 6.8: Perturbación tipo pulso

El algoritmo genético que se utilizó para entrenar la red, contó con los siguientes parámetros:

Número de individuos: 50
Número de generaciones: 1000
Límite mímino de búsqueda: -5
Límite máximo de búsqueda: 5
Δx_{max}: 0,01

6.5. Uso del control predictivo

Una vez alimentados los datos y entrenada la red, se procedió a poner a prueba los resultados obtenidos. Eso se hizo de la siguiente manera:

1. Se crea una serie de perturbaciones, de manera aleatoria, a la que será sometido el sistema. Ésta está compuesta por más de un tipo de perturbación a lo largo de 1000 segundos. Esto permite observar mejor el comportamiento del controlador, al compensar un conjunto de perturbaciones una tras de otra.

2. Se ejecuta la simulación, en donde la RMP va generar cada 0,1s los parámetros más adecuados para el controlador con base al tipo de perturbación que se le alimenta. Ver sección 5.3.

3. Una vez terminada la simulación, (1000 segundos) se calcula el valor de la función objetivo.

4. Se toma la misma serie de perturbaciones y se somete a la simulación sin control predictivo. Para esto se encontró que, en promedio, los valores para Kc, τ_D y τ_I que mejores resultados obtuvieron, fueron los siguientes

$$Kc = 3$$
$$\tau_D = 3$$
$$\tau_I = 2$$

   Es importante recalcar que estos parámetros se mantuvieron fijos a lo largo de la simulación, al contrario de los valores obtenidos por el control predictivo.

5. Se calcula el valor de la función objetivo de la simulación sin control.

6. Comparación de ambos resultados.

Después de someter al sistema a diferentes perturbaciones de distintos tipos y mezclas entre ellas, se han seleccionado 3 para ser presentadas como las más relevantes. Junto con cada explicación del tipo de perturbación y su efecto en el sistema, existen dos figuras que describen su comportamiento gráficamente. Una de ellas es la que se simuló con el control predictivo y la otra es la que carece de éste. A continuación se describen las perturbaciones.

- **Perturbación 1.** La simulación está representada en las figuras 6.9 y 6.10. Podemos apreciar que se tienen dos perturbaciones tipo escalón en aproximadamente 600 segundos de separación.La función objetivo evaluada con el control predictivo arrojó un valor de 776.1, mientras que sin control predictivo fue de 1247.3. Esto es importante mencionarlo, pues demuestra que el algoritmo genético hizo su trabajo al encontrar parámetros que minimizaran la función objetivo. La diferencia entre el sistema con control predictivo y el que carece de éste, fue el tiempo de respuesta. Además que el primero utiliza un flujo de agua mayor para compensar rápidamente la perturbación.

- **Perturbación 2.** Esta simulación está representada en las figuras 6.11 y 6.12. Podemos ver que se tienen dos rampas, una seguida de la otra y de pendiente contraria. Algo interesante es ver cómo el sistema tiene que recuperarse de una perturbación después de que otra ocurra. En este caso la función objetivo tiene un valor de 1431.8 para el sistema con control predictivo y un valor de 2741.9 para el otro. Al igual que en la perturbación anterior, el control predictivo actúa un poco más rápido que el otro, permitiendo compensar la perturbación con más rapidez.

- **Perturbación 3.** Una de las perturbaciones que puede parecer menos importante es la de pulso, ya que solamente cambia las condiciones en un intervalo de tiempo muy pequeño. Sin embargo esto puede ser suficiente para sacar al sistema de su estado estacionario. En esta serie de perturbaciones podemos ver perturbaciones tipo pulso hacia arriba y hacia abajo. Ambos controladores intentan compensar rápidamente las perturbaciones, evitando que ambos sistemas sufrieran grandes cambios. En este caso el valor de la función objetivo es de 64.8 para el sistema con control predictivo y 66.3 para el otro. Podemos ver que son muy próximos, lo que podría indicar que el control predictivo no es tan necesario para perturbaciones como éstas. También es importante mencionar que el control predictivo parece desempeñarse mejor cuando la perturbación es tipo rampa o escalón, pues el cambio brusco o constante requieren una selección más precisa de los parámetros de control.

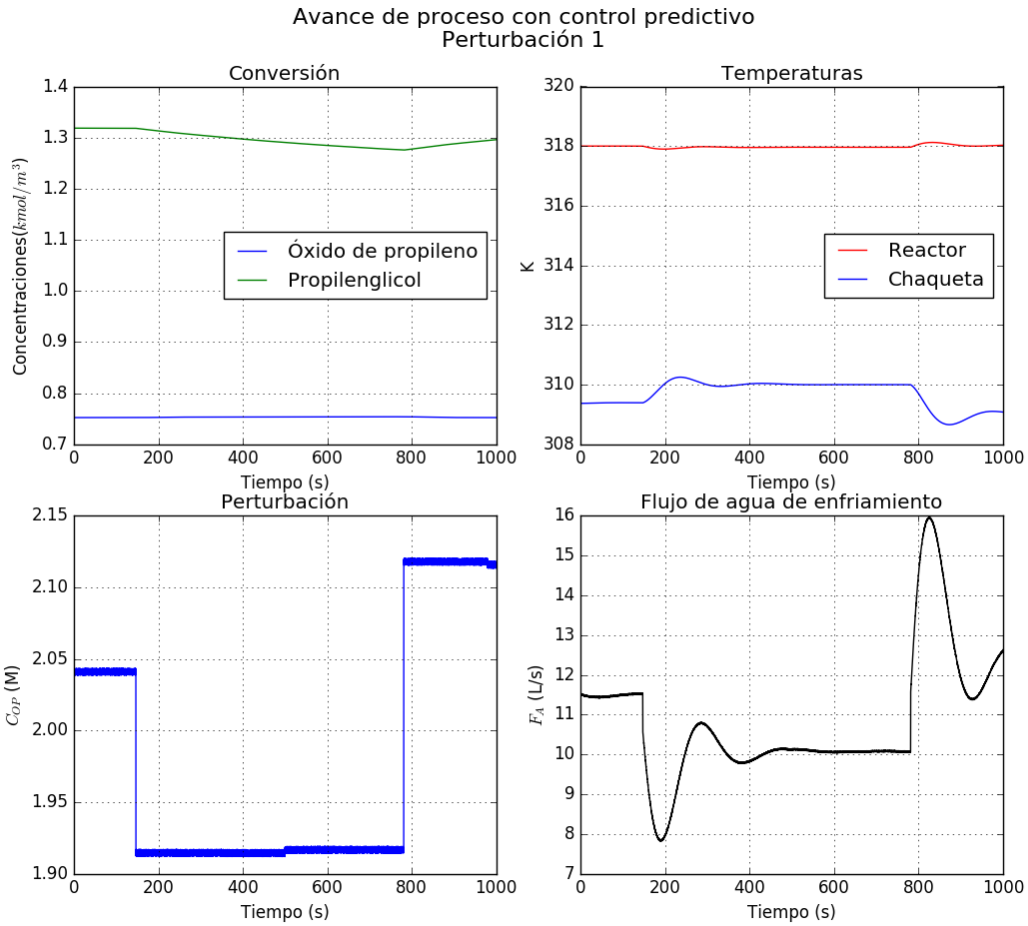

Figura 6.9: Perturbación 1 – $f_{Objetivo} = 776{,}1$

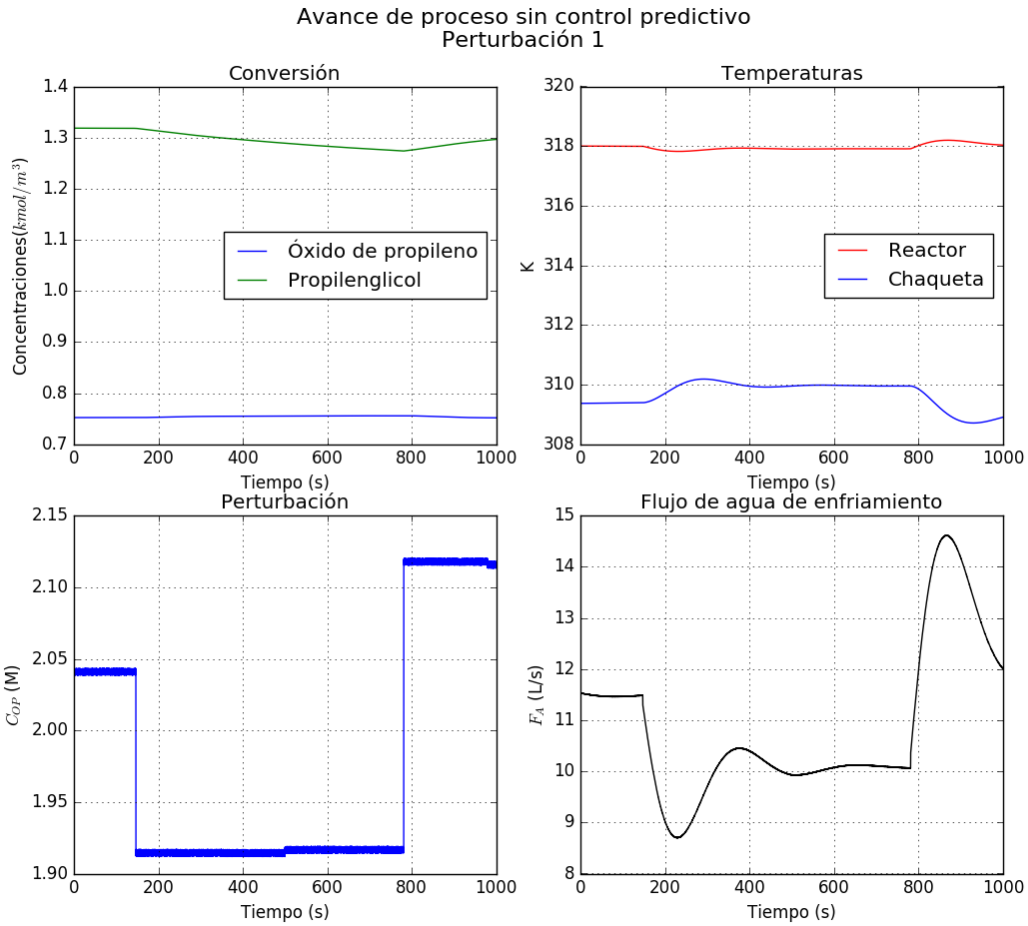

Figura 6.10: Perturbación 1 – $f_{Objetivo} = 1247{,}3$

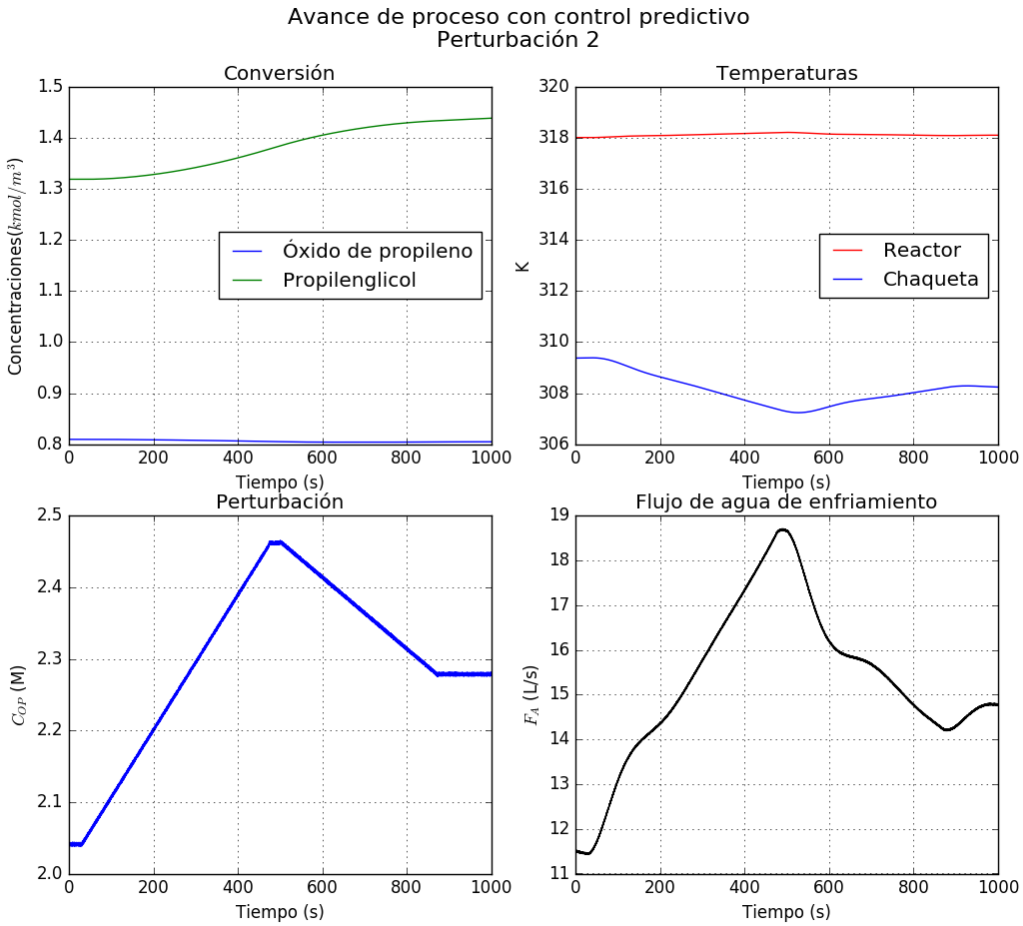

Figura 6.11: Perturbación 2 – $f_{Objetivo} = 1431{,}8$

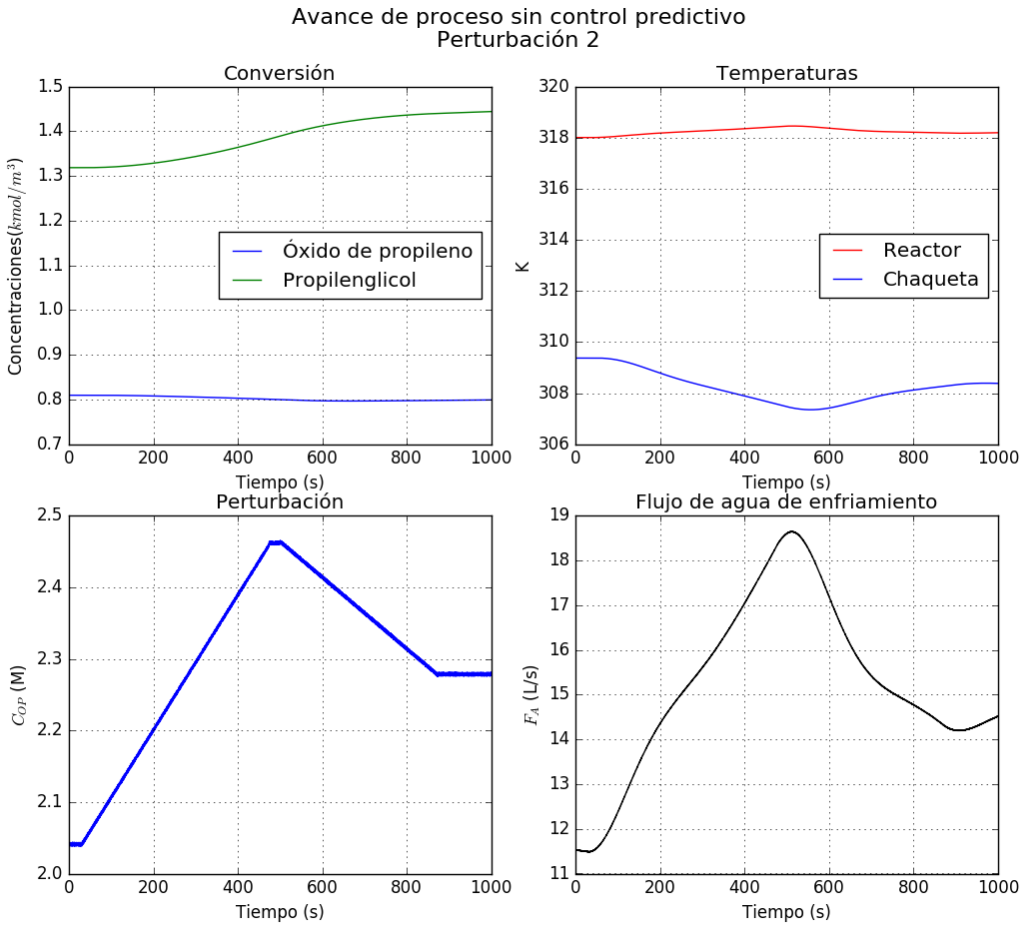

Figura 6.12: Perturbación 2 – $f_{Objetivo} = 2741{,}9$

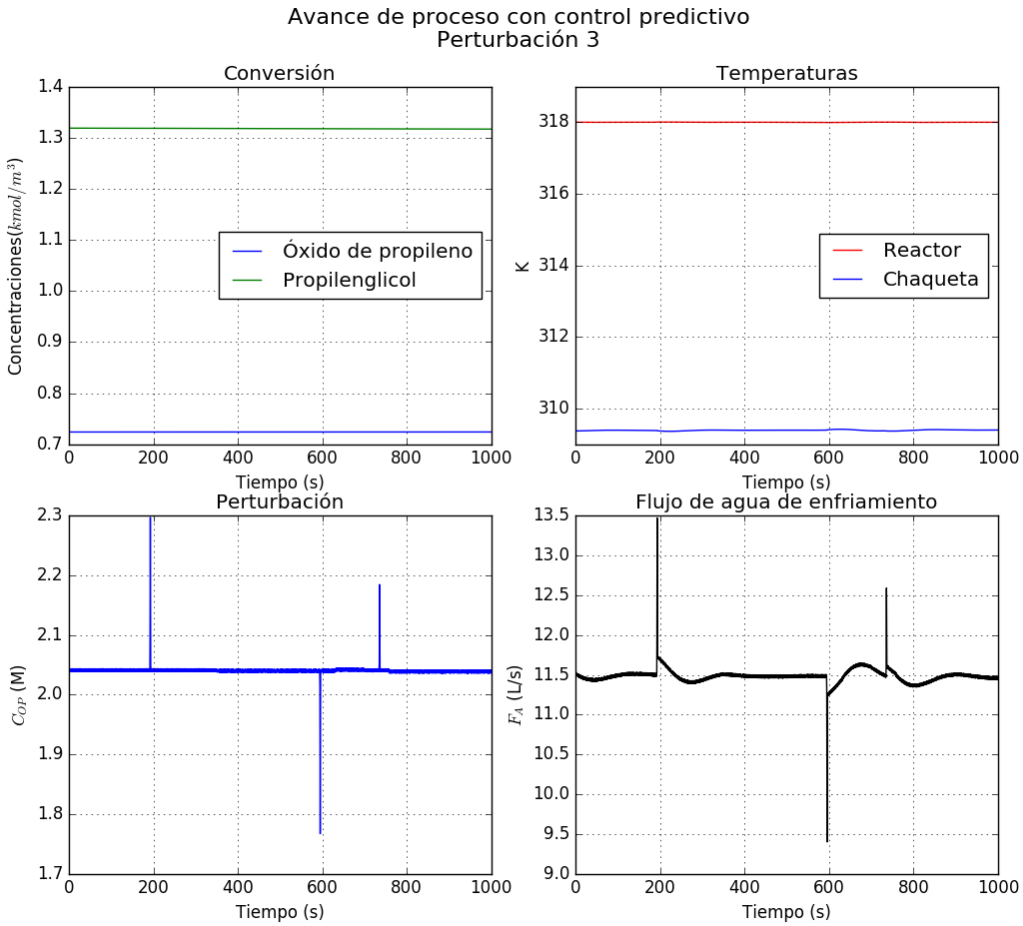

Figura 6.13: Perturbación 3 – $f_{Objetivo} = 64{,}8$

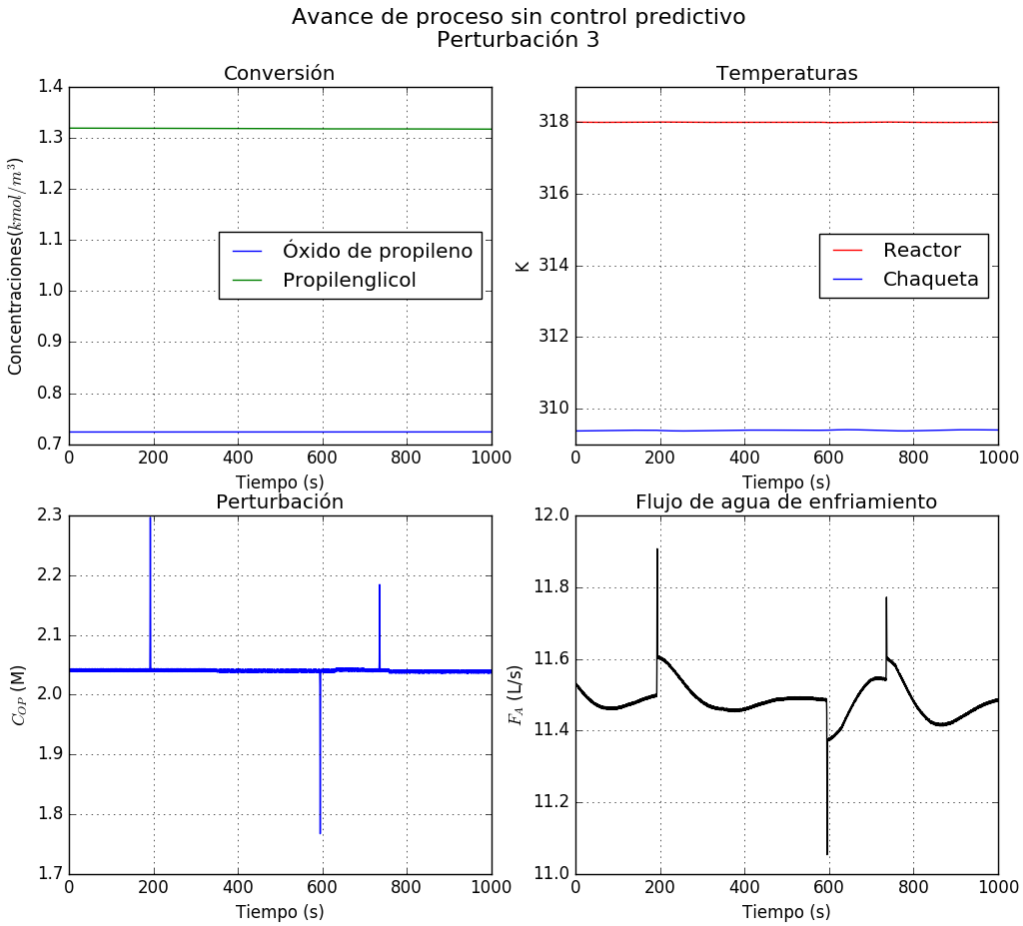

Figura 6.14: Perturbación 3 – $f_{Objetivo} = 66{,}3$

Capítulo 7

Conclusiones

En la industria es indispensable, por cuestiones de calidad y seguridad, el control de procesos. Un buen sistema se control de procesos es el mejor aliado de los ingenieros de producción. Este trabajo ha demostrado que un algoritmo genético, en combinación con una red neuronal, permiten controlar un CSTR mediante control predictivo. Demostrando que el control predictivo es más eficiente que un sistema en el que sus parámetros de control permanecen constantes en el tiempo. Sin embargo este método no se limita solamente al uso en un reactor químico, puede ser usado en cualquier otro proceso teniendo un modelo matemático que simule fielmente al sistema que se desea controlar.

Bibliografía

[1] Penrose Roger. *Shadows of the Mind*. Oxford University Press, 1994.

[2] Kurt Hornik; Maxwell Stinchcombe; Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2, 1989.

[3] D. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, 1989.

[4] Ingham John. *Chemical Engineering Dynamics*. Wiley-VCH, Weinheim, 2000.

[5] H. Fry. *The mathematics of love : patterns, proofs and the search for the ultimate equation*. TED Books/Simon & Schuster, 2015.

[6] H. Fogler. *Elementos de ingeniería de las reacciones químicas*. Pearson Education, third edition, 2001.

[7] T. Furusawa, H. Nishimura, and T. Miyauchi. Experimental study of a bistable continuous stirred-tank reactor. *Journal of Chemical Engineering of Japan*, 2(1):95–100, 1969.

[8] Michael A. Winkler. *Chemical Engineering Problems in Biotechnology*. Elsevier Science Publishers, Ireland, 1990.

[9] S. Chapra and R. Canale. *Métodos Numéricos para Ingenieros*. McGraw-Hill, fifth edition, 2007.

[10] L. B. Booker, D. E. Goldberg, and J. H Holland. Classifier systems and genetic algorithms. *Artificial Intelligence*, 40, 1989.

[11] John H Holland. *Adaptation in natural and artificial systems*. MIT Press, 1992.

[12] Melanie Mitchell. *An introduction to genetic algorithms*. MIT Press, 1996.

[13] C. Reeves and J. Rowe. *Genetic Algorithms - Principles and Perspectives*. Kluwer Academic, 2002.

[14] A. Kuri. *Comprehensive Approach to Genetic Algorithms in Optimization and Learning*. Colección de Ciencia de la Computación, IPN, 1999.

[15] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[16] M Minsky and Papert S. *Perceptrons: An Introduction to Computational Geometry*. The MIT Press, 1969.

[17] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323, oct 1986.

[18] J. Henseler. *Artificial Neural Networks: An Introduction to ANN Theory and Practice*, chapter Back Propagation, pages 37–66. Springer Berlin Heidelberg, Berlin, Heidelberg, 1995.