UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Identificación de Sistemas mediante Neuroevolución de Topologías Aumentadas

TESIS

Que para obtener el título de

Ingeniero Mecatrónico

P R E S E N T A

Rodrigo Romero Mondragón

DIRECTOR DE TESIS

M.I. Serafín Castañeda Cedeño

Ciudad Universitaria, Cd. Mx., 2021

Agradecimientos

Doy gracias a todas y cada una de las personas que han pasado por mi vida y la han hecho tal y como es. En especial a mis papás, por su esfuerzo para darme una juventud plena; a mi hermano, por enseñarme sobre la vida; a Frida, por siempre estar a mi lado y ayudarme cuando tropiezo; a mis amigos, como el Edgar Miguel, el Manu, el Neto, el Carlos, el Serch, el Chava, el otro Chava, el Vic, el Puebla, el Uriel, el Matus, el Rubén y el Robert por hacer la carrera más divertida; a mis profesores por transmitir a mi y a mis compañeros tanto conocimiento; a Hilaria, Ramen, Cuco y Kira, por mover sus colas al verme; y a toda mi demás familia, por siempre contar con ella.

Educad a los niños y no será necesario castigar a los hombres.

- Pitagoras

Índice general

Índice de figuras

Resumen

El objetivo de este trabajo es probar el uso del algoritmo de Neuroevolución de Topologías Aumentadas (NEAT) para identificar sistemas dinámicos. Se presentan conceptos utilizados, como que es un sistema, algunos tipos de sistema, métodos de identificación de sistemas, medidores de éxito de la identificación, que son las redes neuronales y algunos ejemplos de identificación mediante las mismas, que son algoritmos genéticos, y finalmente se da una breve explicación de como funciona el algoritmo NEAT. Después, se explica la metodología a seguir y las herramientas utilizadas para cada una de las siguientes etapas: Recolección de datos, evolución de redes y validación de las redes. Luego, se usa la metodología para identificar un motor de corriente directa utilizando distintas señales de excitación. Se presentan los esquemas y fotos del experimento realizado, así como las condiciones en que se realizaron la obtención de datos, la evolución y la validación de los resultados. Finalmente se presentan los resultados obtenidos y se comparan con un modelo clásico del motor. En varias ocasiones, la metodología logra igualar o mejorar el desempeño del modelo clásico, aunque no se logra relacionar la efectividad de la metodología con el tipo de señales utilizadas.

Capítulo 1

Introducción

El proceso de identificación de sistemas es de suma importancia en aplicaciones ingenieriles, ya sea para la predicción de fenómenos naturales o para control de maquinaria y procesos.

Para realizar una identificación adecuada, además de conocer el comportamiento del sistema al ser utilizado, es conveniente tener más conocimiento previo del sistema. Este conocimiento incluye el saber los subsistemas que componen el sistema a identificar, mediciones físicas de estos subsistemas y propiedades del comportamiento del sistema, como su linealidad o si es un sistema de primero o tercer orden. Esto simplifica los pasos de selección de modelo (por que ya se puede modelar uno), la determinación de los parámetros de este (por que las medidas físicas forman parte de los parámetros) y su validación. Sin embargo, todo este conocimiento previo no siempre esta disponible, ya sea por falta de permisos, es difícil aplicar las entradas deseadas o es un sistema poco estudiado.

Una forma de lograr abarcar todas las dinámicas del sistema es utilizar un modelo con mayor complejidad (de alto orden y no lineal), pero se terminaría necesitando un mayor poder de computo que no siempre esta disponible.

Otra forma de resolver el problema es utilizando un modelo simple y solo identificar una región de operación de interés. El problema con esto es que, si se requiriera analizar otra región, una nueva identificación seria necesaria.

Este trabajo busca realizar la identificación de un sistema con un conocimiento mínimo de este, que de como resultado un modelo con la menor complejidad posible y que no se limite a regiones especificas de operación. Para lograr este cometido, la metodología a seguir se apoya principalmente del algoritmo de Neuroevolucion de Topologías Aumentadas o NEAT, por sus siglas en ingles. El algoritmo busca la red neuronal más apta para realizar una tarea en especifico. La búsqueda se realiza cambiando la estructura y pesos de varias redes mediante operaciones de mutación y cruza, similares a las de un algoritmo genético. Este algoritmo ya se ha utilizado para identificar sistemas estáticos y crear controladores de manera exitosa [20].

Para comprobar la efectividad de esta metodología, se realizara la identificación de un motor de corriente directa con distintos conjuntos de datos. Finalmente, se compararan los desempeños de las redes resultantes con el de un modelo obtenido matemáticamente con los parámetros reportados por su fabricante.

1.1. Objetivo general

Probar el uso del algoritmo de Neuroevolución de Topologías Aumentadas (NEAT) para identificar sistemas dinámicos mediante la identificación de un sistema lineal de primer orden.

1.1.1. Objetivos específicos

- Instrumentar y obtener los datos de salida de un sistema electromecánico mientras es excitado con distintas señales de entrada.
- Adaptar una implementación del algoritmo NEAT para identificar el sistema.
- Realizar la identificación del sistema variando las señales de entrada.
- Comparar el comportamiento de las redes neuronales resultantes, el del sistema físico y un modelo clásico.

Capítulo 2

Antecedentes

2.1. Sistemas

Un sistema puede ser definido como una sección del universo conformada de varias partes o componentes que interactúan entre si y con su entorno [18] [17]. Las interacciones pueden ser representadas como variables medibles y pueden ser clasificadas de 4 maneras [13]:

- Entradas: La causa de estas variables es ajena al sistema. Las entradas pueden ser manipuladas directamente por un usuario del sistema.
- Salidas: Son las variables del sistema medibles de interés para un usuario
- Perturbaciones: Son el efecto de sucesos inesperados en el entorno del sistema.
- Ruido: Es la incertidumbre inherente del sistema o del los instrumentos de medición utilizados.

Una clasificación importante de los sistemas es la siguiente [17]:

- Sistemas estáticos: Son sistemas que no dependen del tiempo. Esto significa que la salida solo dependerá de las variables de entrada aplicadas.
- Sistemas dinámicos: Son sistemas que si dependen del tiempo. Esto significa que la salida depende de la entrada y de la misma salida del sistema.

Para poder conocer el comportamiento de los sistemas es necesario concebir un modelo que pueda predecir las variables de salida conociendo las variables de entrada. El modelado de sistemas esta basado normalmente en leyes físicas [17] las cuales están representadas en ecuaciones matemáticas. Dependiendo la representación matemática del sistema, es posible identificar otras dos clasificaciones de sistemas:

- Sistemas lineales: estos sistemas pueden ser representados por ecuaciones lineales. Este tipo de sistemas siguen el principio de superposición.
- Sistemas no lineales: El principio de superposición no es aplicable para estos.

El principio de superposición significa que para cualquier entrada expresada como una combinación lineal de ciertas señales, la respuesta del sistema es la misma combinación lineal de las respuestas a las señales individuales [14].

Finalmente una ultima clasificación de sistemas útil para este trabajo es la siguiente [17]:

- Sistemas en tiempo continuo: los sistemas en tiempo continuo son aquellos con variables que se van modificando instantáneamente mientras avanza el tiempo. Estos sistemas pueden ser representados por ecuaciones diferenciales.
- Sistemas en tiempo discreto: son sistemas con variables que cambian solo en momentos instantáneos en intervalos que pueden ser constantes o no. Estos sistemas pueden ser representados por ecuaciones en diferencias.

Es importante mencionar que un sistema no forzosamente es físico. También existen sistemas sociales, biológicos o económicos.

2.2. Identificación de Sistemas

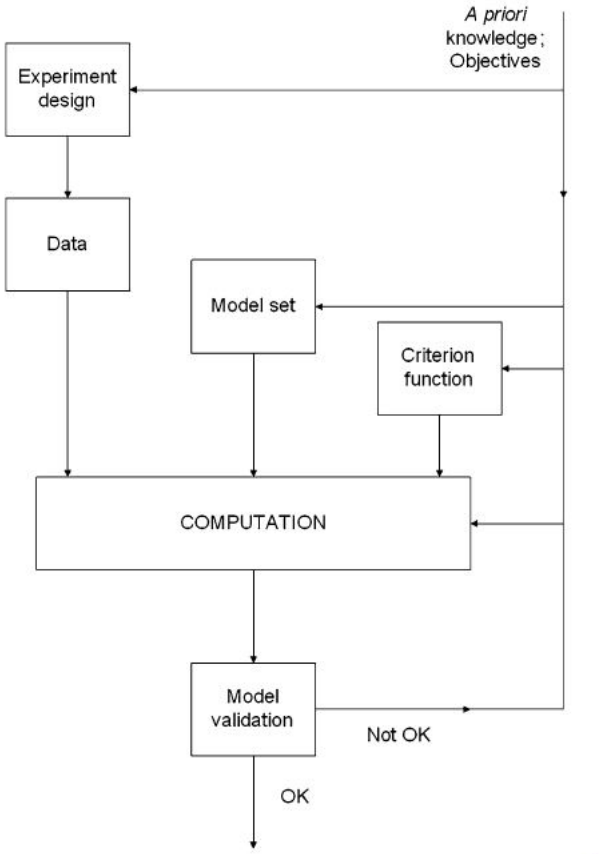
La identificación de sistemas consiste en lograr adecuar lo mejor posible los modelos matemáticos que los describen a su comportamiento real [18].

Hay cuatro pasos principales en la identificación de sistemas [18]:

- Obtención de información del sistema:
  Este paso es posible de realizar con solo observar el comportamiento común del sistema, como en las situaciones en las que normalmente se ocuparía el sistema. Sin embargo, resulta más eficiente crear experimentos dedicados que aplique entradas especificas al sistema para optimizar el proceso de obtención de datos. De esta etapa depende mucho el resultado final de la identificación.
- Selección de la estructura de un modelo para representar el sistema:
  Existen varios tipos de modelos con los que pueden representar los sistemas como los siguientes:
  - Modelos paramétricos contra no paramétricos:
    Los modelos paramétricos describen al sistema usando un número pequeño de valores característicos llamados parámetros (como una función de transferencia). Un modelo no paramétrico utiliza varias mediciones de la respuesta del sistema a distintas entradas para describirlo (como las tablas de saturación).
  - Modelos de caja blanca contra caja negra:
    Los modelos de caja blanca son los que se obtienen al modelar el sistema mediante leyes físicas. En este tipo de modelos se puede observar el funcionamiento interno del sistema claramente. Un modelo de caja negra no necesita del conocimiento interno del sistema y solo necesita de la suficiente información para poder representar el comportamiento de la salida del sistema con respecto a su entrada. Este se puede obtener con solo los datos obtenidos en el paso anterior.
  - Modelos lineales contra no lineales:
    En la vida real casi todos los sistemas tienen comportamientos no lineales. Sin embargo, es posible representar un comportamiento no lineal como lineal si se elige una región apropiada. Normalmente esta región es donde el sistema se encuentra comúnmente o donde se desea que sea operado.
- Elegir los parámetros del modelo para que el modelo se ajuste a las observaciones:
  De usarse un modelo paramétrico en el paso anterior, es necesario ahora obtener los valores de estos parámetros tal que el modelo se apegue lo mejor posible a la información obtenida en el primer paso. Normalmente, esto se logra minimizando un criterio, como una función de costo, que mida el ajuste del modelo con la información. Usualmente la función de costo define la distancia entre los datos y el modelo.

- Validar el modelo seleccionado:
  Finalmente, es necesario revisar si el modelo al que se llegó describe la información recabada de manera adecuada y, si no, conocer porque no la describe. El modelo puede llegar a no apegarse a la información por error en la identificación, pero también por dinámicas no modeladas o ruido en la obtención de los datos.

Figura 2.1: Algoritmo clásico para la identificación de sistemas [13]



En la Figura 2.1 se observa un diagrama de flujo con el algoritmo básico de la identificación de sistemas.

2.2.1. Modelos de Sistemas

A continuación se presentan algunos modelos matemáticos utilizados para representar sistemas lineales, invariantes en el tiempo:

- Ecuaciones diferenciales y en diferencias.
- Función de transferencia.
- Representación de respuesta al impulso.
- Espacio de estados.

Todos estos modelos tienen una versión en tiempo continuo y una en tiempo discreto. A continuación, se presentan algunas definiciones de estos modelos.
La función de transferencia es definida como la razón de la transformada de Laplace de la salida entre la transformada de Laplace de la entrada, asumiendo condiciones iniciales nulas [17].

$$Y(s) = G(s)U(s) \implies G(s) = \frac{Y(s)}{G(s)} \tag{2.1}$$

La representación en espacio de estados utiliza el concepto de estado, que es la cantidad mínima de variables del sistema que se debe conocer, junto con la entrada, para poder determinar el comportamiento del sistema en cualquier momento. Se plantean una serie de ecuaciones que describen la dinámica de cada estado en función de los otros estados, y sus entradas. En general, la representación de cualquier sistema en espacio de estados es la siguiente:

$$\begin{aligned} \dot{x} &= f(x, u) \\ y &= h(x, u) \end{aligned} \tag{2.2}$$

donde:
x es el vector de variables de estado,
u es el vector de señales de entrada,
f es la función vectorial que describe la dinámica del sistema,
y es el vector de salidas, y
h es la función vectorial que describe la salida del sistema.
Sin embargo para sistemas lineales existe la siguiente representación:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{2.3}$$

donde:
A es la matriz de estados,
B es la matriz de entradas,
C es la matriz de salidas, y
D es la matriz de transición directa [17].

Usualmente, al identificar sistemas se utilizan modelos en tiempo discreto, ya que puede usarse con una cantidad de datos limitada y es más sencillo de implementar en código para la realización de simulaciones. Resultado de esto también es conveniente utilizar modelos simples ya que si el modelo fuera demasiado complejo, este podría no ser útil para la aplicación deseada. La identificación de sistemas es considerado un modelado aproximado para una aplicación en especifico.

Estos no son los únicos modelos existentes. más adelante se presentaran a otras formas de representación.

2.2.2. Métodos de identificación

Existen varios métodos de identificación. La elección de que método elegir depende del tipo de modelo que se quiere obtener. A continuación se presentan algunas categorías y con ejemplos [13]:

- Identificación no paramétrica o basada en datos.
  - Identificación directa por respuesta al impulso
  - Identificación directa por respuesta al escalón

    - Identificación por respuesta a una señal senoidal
- Identificación paramétrica
    - Regresiones lineales y no lineales
        - Estimación por mínimos cuadrados
        - Estimación por máxima verosimilitud

2.2.3. Señales de excitación

Siguiendo los pasos de la identificación, primero es necesario recabar datos del comportamiento del sistema y para ello es necesario diseñar una señal apropiada para que sea aplicada al sistema. Es importante seleccionar estas señales dependiendo del sistema y de para qué se utilizará el modelo obteniendo de la identificación. A continuación se presentan algunas señales de propósito general comúnmente utilizadas [18]:

- Barrido senoidal:
  También llamada chirp (chirrido)periódica, consiste de una señal senoidal que cambia su frecuencia de arriba a abajo y viceversa periódicamente. Se puede describir con la siguiente ecuación

$$u(t) = A\sin((at+b)t) \quad 0 \leq t \leq T_0 \tag{2.4}$$

  donde:
  T_0 es el periodo,
  $a = \pi(k_2 - k_1)f_0^2$,
  $b = 2\pi k_1 f_0$,
  $f_0 = 1/T_0$,
  $k_2 > k_1 \in \mathbb{N}$, y
  $k_2 f_0, k_1 f_0$ son las frecuencias más alta y más baja respectivamente.

- Multisenoidal de Schroeder:
  Es una suma de señales senoidales relacionadas por sus harmónicas

$$u(t) = \sum_{k=1}^{F} A\cos(1\pi f_k t + \phi_k) \tag{2.5}$$

  donde:
  las fases de Schroeder $\phi_k = -k(k-1)$, y
  $f_k = l_k f_0 \quad l_k \in \mathbb{N}$.

- Secuencia binaria pseudoaleatoria:
  Es una secuencia periódica de longitud N que cambia la señal entre 2 valores (por ejemplo +1 y -1). Los cambios ocurren en tiempos discretos que son múltiplos del tiempo de muestreo del sistema.

$$u(t) = A \times sign(w(t) - p_0) \tag{2.6}$$

  donde:
  A es la amplitud,
  sign es una función que devuelve 1 si el numero es positivo, -1 si es negativo y cero si es cero,
  $w(t)$ es una función pseudoaleatoria generada por una computadora y,
  p_0 es la probabilidad de que el valor final sea negativo, y va de 0 a 1.

- Ruido aleatorio:
  Una secuencia de ruido con frecuencias que pueden ser seleccionadas mediante filtros.
- Ráfaga aleatoria:
  Es una secuencia de ruido que se aplica al inicio de la secuencia de muestreo y después se aplica una señal nula.

$$u(t) = w(t)r(t)$$

$$w(t) = \begin{cases} 1 & 0 \leq t \leq T_1 \\ 0 & T_1 \leq t \leq T \end{cases}$$

  donde:
  r(t) es una variable aleatoria.
- Impulso o impacto: Es una señal que excita a la planta durante un pulso corto.

2.2.4. Métodos de validación de modelos

El último paso de la identificación consiste en validar el modelo obtenido. Existen algunas formas de validar que un modelo resultante es apropiado [13]:

- Conocimiento previo:

  Una primera prueba para validar el modelo es revisar los parámetros resultantes del modelo y confirmar que tengan sentido con el sistema real. Por ejemplo, si se tuviera un modelo obtenido por conocimiento físico del sistema y se obtuvieran parámetros relacionados con sus características con valores negativos, resultaría incoherente ya que esto podría implicar que el sistema tiene una longitud o resistencia negativa.
- Inspección gráfica:

  Pueden ser graficados los resultados del modelo obtenido a la par con los datos de salida del sistema y de esta manera observar que tanto se parecen entre si. Ademas, puede graficarse el error entre la salida real y la del modelo y observar si no es muy grande durante ciertos intervalos o si existen cambios periódicos en el error. Este tipo de comportamientos del error suelen ser resultado de dinámicas no modeladas, por lo que es necesario hacer nuevas consideraciones en el modelo.

2.2.5. Criterio de Error de Predicción Final

Dado que muchos modelos pueden ajustarse el mismo conjunto de datos de manera efectiva, es necesario tener un criterio para elegir cual es modelo a utilizar. Normalmente, es deseable utilizar el modelo más sencillo. Para poder medir la complejidad de un modelo, además de que tan bueno es el ajuste del mismo se utiliza el criterio de Error de Predicción Final (FPE por sus siglas en ingles) de Akaike [13]. Este criterio indica que el modelo con el menor FPE es el más exacto. La formula para calcular el FPE es la siguiente:

$$J_{FPE}(M) := \frac{1 + d_M/N}{1 - d_M/N} \frac{1}{N} \sum_{t=1}^{N} \frac{1}{2} error^2 \tag{2.7}$$

donde:
d_M es el número de parámetros en el modelo, y
N es el número de datos utilizados en la identificación.

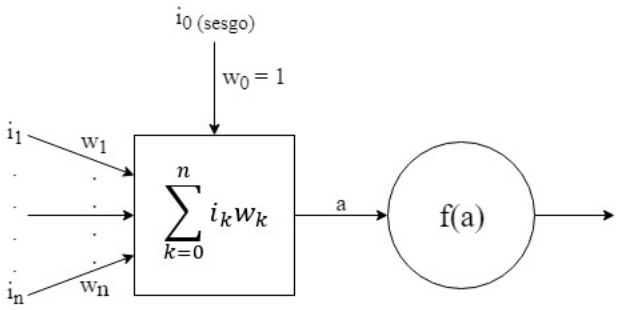


Figura 2.2: Representación gráfica de una neurona artificial.

2.3. Redes Neuronales Artificiales

Las redes neuronales artificiales (RNA) son modelos matemáticos que pretenden representar el funcionamiento interno de un cerebro. Están compuestas de neuronas, que son funciones no lineales, parametrizadas y acotadas. Las variables de las neuronas son llamadas entradas de la neurona y el resultado es llamada salida. En la Figura 2.2 se puede observar una representación gráfica de una neurona. Los parámetros de la función son asignados a cada entrada. La función f, llamada función de activación, recibe la combinación lineal de las entradas (x_i) ponderadas por los parámetros (w_i). Comúnmente a los parámetros se les llama pesos por la traducción de ponderado en ingles (*weighted*). Adicionalmente, a la combinación se le agrega un termino constante llamado sesgo (w_0) La función de activación puede ser de muchos tipos aunque es recomendable que sea una función sigmoidea, como la tangente hiperbólica o la tangente inversa[8].

Estas neuronas pueden ser combinadas de tal manera que la salida de una neurona puede ser la entrada de otra. A esta combinación se le llama una red de neuronas.

2.3.1. Tipos de Redes Neuronales

Existen varios tipos de RNA. A continuación se presentan algunos de estos tipos:

- RNA de avance (*feedforward*):
  Una red neuronal de avance puede ser representada gráficamente como un conjunto de neuronas conectadas mutuamente, en el que el flujo de información fluye en una sola dirección (hacia adelante), de las entradas a las salidas. A las neuronas que entregan la salida final de la red se les llama neuronas de salida. A todas las otras neuronas de las red se les conoce como neuronas ocultas. A este tipo de redes se les conoce como redes estáticas, ya que, como un sistema estático, la salida de la red siempre depende solo de la entrada de la red [8].
- RNA recurrente o de retroceso (*reccurent/feedbackward*):
  En la representación gráficas de estas redes se pueden observar ciclos. Esto significa que cada neurona puede estar conectada a si misma o a una neurona "más adelante"de ella en la red. por lo tanto la salida de la red no solo depende de la entrada, sino también de los valores pasados de la misma red. Por lo tanto a cada conexión de este tipo de red tiene un retraso además de tener asignado un parámetro. Es posible ver la analogía de estas redes con los sistemas dinámicos, en especifico en tiempo discreto. Una propiedad importante de estas redes es que no importa que tan

compleja sea la red, es posible encontrar una forma canónica de esta. Esto significa que la red esta hecha de una red de avance y algunas de las salidas de algunas neuronas (conocidas como salidas de estado) realimentan a la red después de un retraso de tiempo [8].

En la figura 2.3 se presentan representaciones gráficas de cada tipo de red.

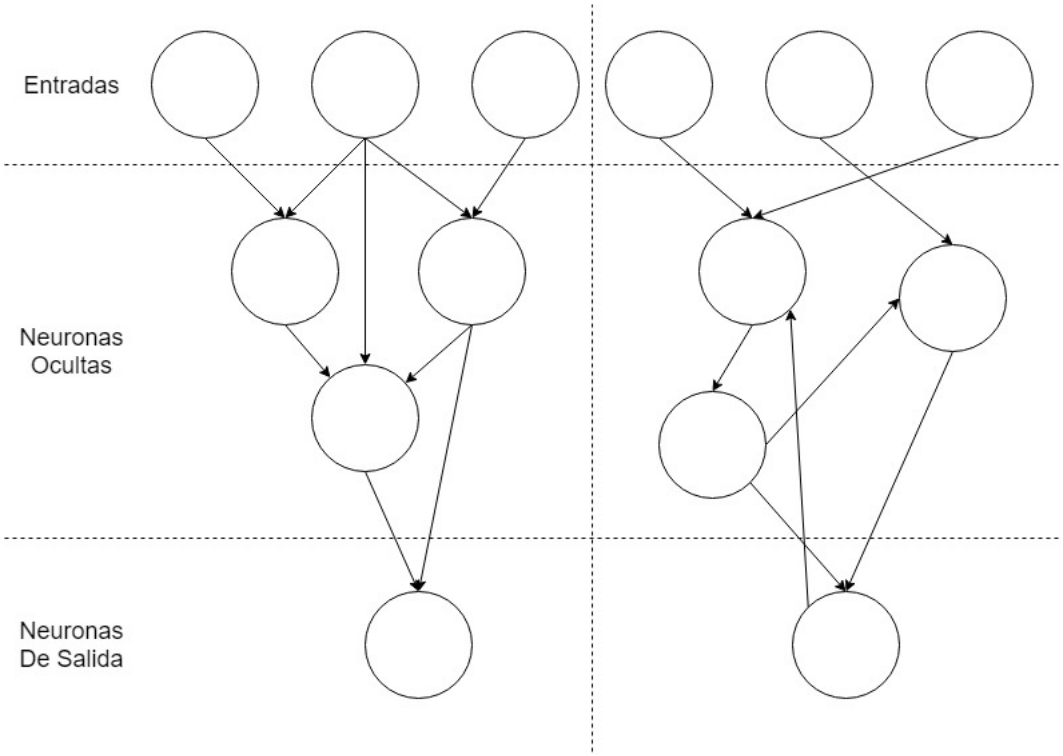


Figura 2.3: De izquierda a derecha: Red neuronal de adelanto, red neuronal recurrente. Es fácil observar que, a pesar de tener la misma cantidad de neuronas, las redes son diferentes por sus conexiones. En la red recurrente pueden verse ciclos en las conexiones de nodos.

2.3.2. Entrenamiento

Para lograr que una RNA tenga un comportamiento deseado, es necesario asignar los pesos de las conexiones de manera adecuada. Dado que la cantidad de parámetros en redes muy grandes suelen ser muchos, existen algoritmos que sirven para poder encontrar los pesos más adecuados. A este proceso se le llama entrenamiento de la red. El método más común de entrenamiento de redes es el de mínimos cuadrados o LMS (*Least-mean square*) [8]. Este método recibe su nombre por que busca minimizar la función de costo, la cual es el error cuadrado entre la salida de la red y la salida deseada. Esto se expresa en la siguiente ecuación:

$$J = \sum_{k=1}^{N} (y_{d,k} - y_{r,k})^2 \tag{2.8}$$

donde:
$y_{d,k}$ es la salida k-ésima deseada,

$y_{r,k}$ es la salida k-ésima de la red, y
N es el número de datos utilizados en el entrenamiento.

Para lograr minimizar esta función es necesario encontrar su gradiente mediante un algoritmo de retropropagación (*backpropagation*).

2.4. Identificación de sistemas con redes neuronales

Como se menciono previamente, las redes neuronales pueden ser similares a sistemas estáticos y dinámicos. Esta similitud permite que se utilicen redes neuronales como modelos en el proceso de identificación. Además, ya ha sido demostrado que es posible aproximarse al comportamiento de cualquier sistema no lineal mediante una red neuronal [6][15]. La identificación puede ser realizada mediante LMS u otros métodos de entrenamiento mediante gradiente [12].

En [22] se entrena una red de una sola capa mediante mínimos cuadrados para identificar un sistema discreto de segundo orden con una entrada y una salida. Dado que la red entrenada no es recursiva por si sola, el autor agrega entradas a la red tal que el número de entradas es igual al orden del sistema más el número de salidas retrasadas del sistema. Como entrada para obtener los datos previos del sistema utiliza una secuencia binaria pseudoaleatoria. Esta metodología resulto exitosa para identificar el sistema mencionado, sin embargo, se apoya del conocimiento previo del orden del sistema, lo cual no siempre es posible conocer.

En [9] se presenta una topología de redes con una parte estática y una dinámica y con funciones de activación difusas. La metodología para entrenar estas redes también se basa en el método del gradiente, aunque para la determinación de los parámetros iniciales de la red se utiliza un método llamado ANFIS. Al igual que el ejemplo anterior, la estructura de la red e determinada por conocimiento previo del sistema a identificar. En especifico, para seleccionar la cantidad de estados internos de la red se necesita saber el orden del sistema. En una de las aplicaciones practicas de la metodología, el sistema a identificar se excita mediante una señal aleatoria discreta.

2.5. Algoritmos Genéticos

Un algoritmo genético es un programa o un conjunto de reglas que utiliza una computadora para resolver un problema complejo simulando la teoría de la evolución mediante selección natural [7].

El algoritmo maneja una población de posibles soluciones. Cada solución es representada con un genoma, que es una representación abstracta. También se define un conjunto de operaciones de reproducción. Estas operaciones se aplican directamente a los genomas y se utilizan para realizar mutaciones y recombinaciones de las soluciones del problema. La representación de las soluciones y las operaciones de reproducción son de mucha importancia, ya que el comportamiento del algoritmo genético depende mucho de estos.

La selección debe comparar a cada uno de las soluciones en la población. Para comparar se utiliza una función de aptitud (*fitness function*). Cada genoma tiene un valor asociado que corresponde con la aptitud de la solución que representa y este valor debe corresponder con la evaluación de que tan buen candidato de solución es. La solución óptima es la que maximiza la función de aptitud.

Una vez definidos las funciones de reproducción y la función de adaptabilidad se comienza la evolución. Se inicia generando una población inicial de genomas. Esta población debe ofrecer una diversidad amplia de material genético. Generalmente la población inicial se genera aleatoriamente.

Después, se hace un proceso iterativo que evoluciona a la población. Cada iteración consiste en los siguientes pasos [19]:

- Selección:
  Se seleccionan individuos de la población para reproducirse. Esta selección se hace aleatoriamente con una probabilidad proporcional a la aptitud de cada individuo, tal que los de mejores aptitud sean elegidos más seguido que los que tienen poca aptitud.
- Reproducción:
  Los individuos seleccionados del paso anterior producen "hijos"mediante las funciones de reproducción, ya sea cruza (recombinación) o mutación.
- Evaluación:
  Se evalúan los nuevos individuos para obtener su aptitud.
- Remplazo:
  Finalmente la población vieja se remplaza por la nueva.

En la figura 2.4 se presenta un diagrama de flujo del algoritmo genético.

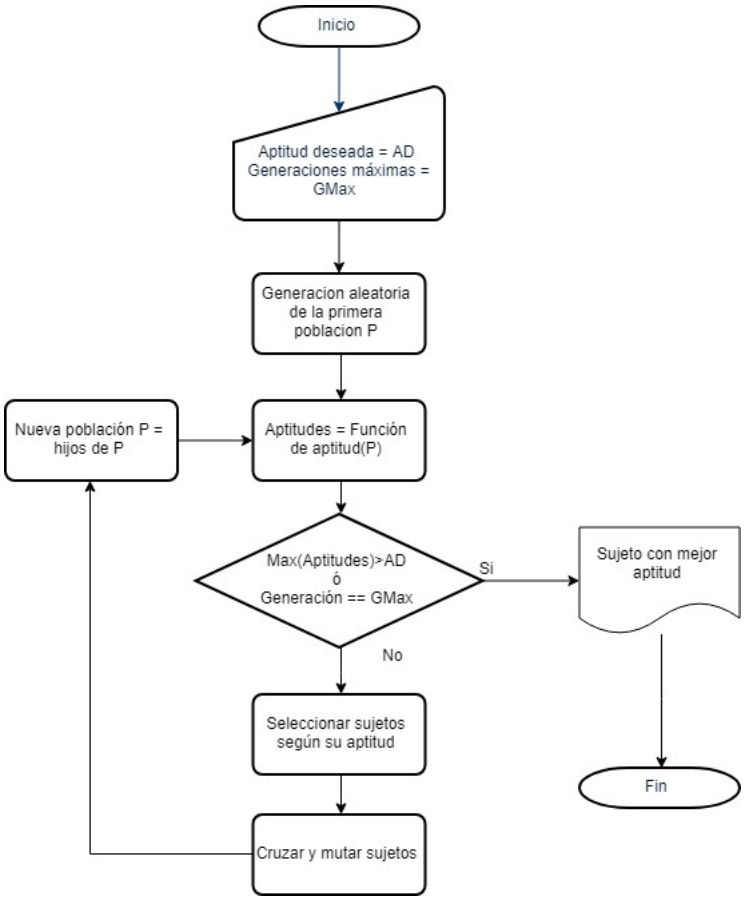


Figura 2.4: Diagrama de flujo de un algoritmo genético.

2.6. Neuroevolución de Topologías Aumentadas

La neuroevolución es la evolución artificial de redes neuronales artificiales utilizando algoritmos genéticos. Esto significa que el algoritmo genético utiliza genomas que representan redes neuronales para resolver un problema. El método de Neuroevolución de Topologías Aumentadas (NEAT por sus siglas en ingles) puede modificar los pesos de las neuronas así como la topología de la red [20]. A continuación se explicaran las propiedades más importantes de este método para su implementación adecuada.

2.6.1. Codificación genética

Los genomas utilizados por NEAT están divididos en 2 partes. La parte de los nodos, y la parte de las conexiones.

El genoma de nodos se refiere a las neuronas de la red y sus entradas. En este genoma, cada gen da información de una sola neurona o nodo. La información de cada nodo incluye un número que identifica al nodo y su tipo (entrada, neurona oculta o salida).

El genoma de conexiones contiene la información de todas las conexiones entre nodos. La información de las conexiones incluye que nodos conecta, en que dirección, su peso, si se encuentra habilitada y su número de innovación. Este ultimo número se utiliza en los procesos de selección y reproducción del algoritmo y sera explicado más a fondo en las secciones siguientes.

En la figura 2.5 se presenta un ejemplo de genoma con su fenotipo.

Genoma de nodos

ID	1	2	3	4	5	6	7
Tipo	Entrada	Entrada	Entrada	Oculta	Oculta	Oculta	Salida

Genoma de conexiones

Nodos	1>4	2>5	3>4	4>6	5>7	6>5	6>7	7>4
Peso	5	2	3	1	5	6	7	8
Habilitada	1	1	1	1	1	0	1	1
Inovacion	1	2	3	6	7	8	10	12

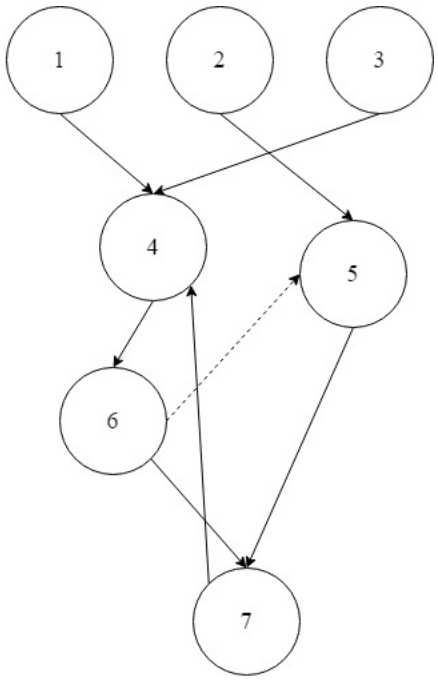


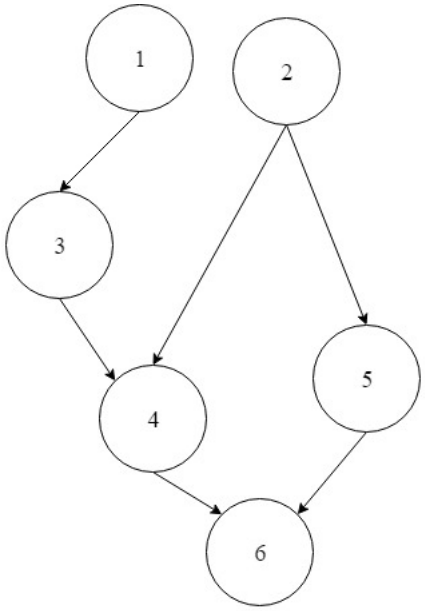
Figura 2.5: Ejemplo de un genoma de NEAT y la red que representa.

2.6.2. Operaciones de reproducción

Por ser un algoritmo genético, NEAT también cuenta con operaciones mutación y cruza para generar nuevas poblaciones.

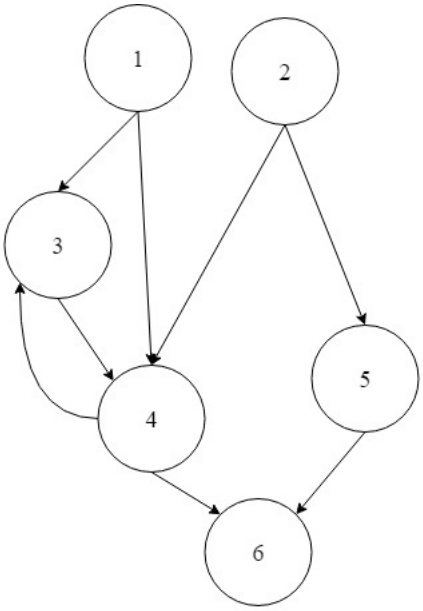
La operación de mutación puede modificar tanto pesos como estructuras de la red. Los pesos pueden ser modificados de manera convencional, como suma o multiplicación con un número aleatorio o inclusive sustitución por otro número aleatorio. Existen 2 mutación estructurales. Ambas aumentan la longitud del genoma agregando nodos o conexiones. Las nuevas conexiones contienen pesos generados aleatoriamente. Un ejemplo de esto se encuentra en la figura 2.6. Los nuevos nodos dividen conexiones ya existentes. Esto significa que la conexión antigua se deshabilita y el nodo nuevo esta conectado a los 2 nodos que se conectaban con la conexión vieja. La conexión que llega al nuevo nodo tiene un peso de valor unitario mientras que la conexión que sale del nuevo nodo tiene el mismo peso que la conexión antigua. Este método se utiliza para minimizar el efecto en el comportamiento de la red al agregar nuevo nodos. Esto se muestra en el ejemplo de la figura 2.7.

La operación de cruza consiste en combinar los pesos de las conexiones. Sin embargo, dado que una red puede tener más conexiones que otras, también es posible que los hijos hereden conexiones de los padres. Para esto es necesario utilizar el número de innovación mencionado anteriormente. Gracias al número de innovación no es necesario buscar que conexiones comparten las redes que se cruzan ya que al surgir nuevas conexiones idénticas el proceso de mutación, estas tienen el mismo número de innovación. Por lo tanto la operación consistirá en revisar los números de innovación, elegir aleatoriamente el peso de las conexiones con el mismo número y agregar las conexiones extras del padre con mejor aptitud. Un ejemplo de cruza se muestra en la figura 2.8.



Genoma de conexiones

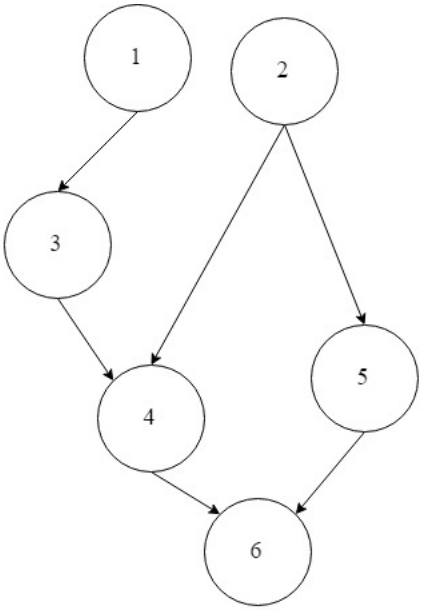
Nodos	1>3	2>4	2>5	3>4	4>6	5>6
Peso	5	2	3	1	5	6
Habilitada	1	1	1	1	1	1
Inovacion	1	2	3	6	7	8



Genoma de conexiones

Nodos	1>3	2>4	2>5	3>4	4>6	5>6	1>4	4>3
Peso	5	2	3	1	5	6	8	10
Habilitada	1	1	1	1	1	1	1	1
Inovacion	1	2	3	6	7	8	9	10

Figura 2.6: Mutación de conexiones

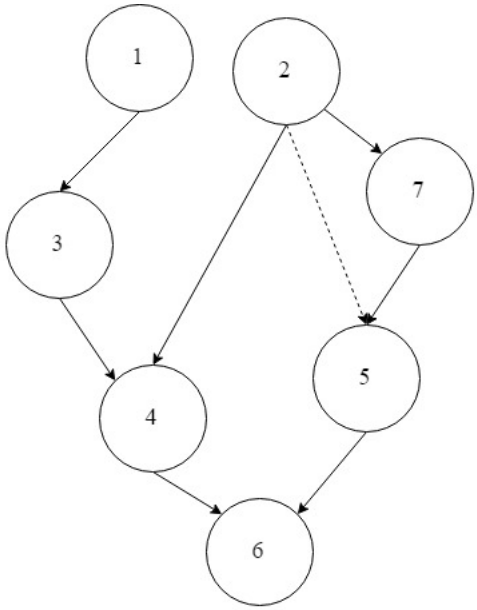


Genoma de nodos

ID	1	2	3	4	5	6
Tipo	Entrada	Entrada	Entrada	Oculta	Oculta	Salida

Genoma de conexiones

Nodos	1>3	2>4	2>5	3>4	4>6	5>6
Peso	5	2	3	1	5	6
Habilitada	1	1	1	1	1	1
Inovacion	1	2	3	6	7	8



Genoma de nodos

ID	1	2	3	4	5	6	7
Tipo	Entrada	Entrada	Entrada	Oculta	Oculta	Salida	Oculta

Genoma de conexiones

Nodos	1>3	2>4	2>5	3>4	4>6	5>6	2>7	7>5
Peso	5	2	3	1	5	6	1	3
Habilitada	1	1	0	1	1	1	1	1
Inovacion	1	2	3	6	7	8	?	?

Figura 2.7: Mutación de nodos

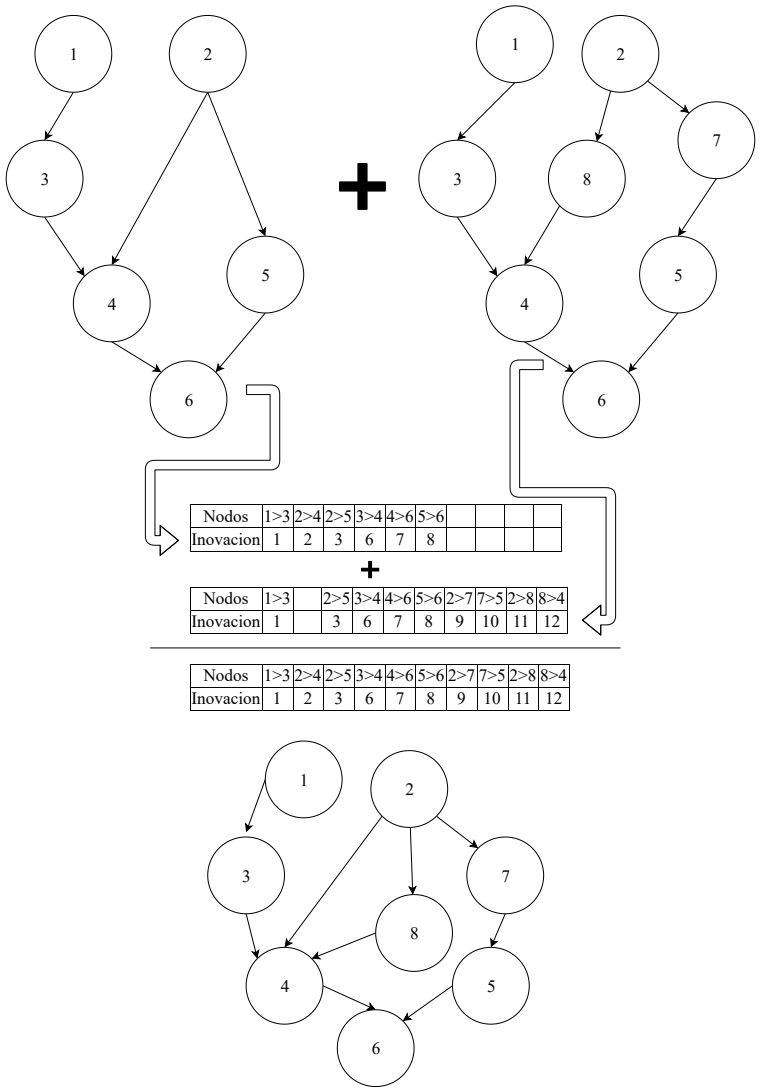


Figura 2.8: Cruza

2.6.3. Especiación

Dado que es posible que una red con aptitud alta disminuya esta con la adición de conexiones o nodos, aunque estas adiciones puedan significar una mayor aptitud después de varias generaciones, es necesario proteger estas mutaciones estructurales y no descartar las redes que las contengan por tener baja aptitud de manera inmediata. Para esto se utiliza la especulación. Para dividir a la población de redes en especies se utiliza de nuevo el número de innovación. El número de genes distintos entre genomas de conexiones se clasifican en conexiones disjuntas y excedentes. Cuando se comparan 2 genomas, si el número de innovación de una conexión que no existe en otro genoma es menor que el número máximo de innovación del genoma con el menor número máximo de innovación se dice que es disjunto. De lo contrario es excedente. Para conocer la especie de una red es necesario determinar la distancia de

compatibilidad con otras redes. Esta distancia esta definida de la siguiente manera:

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \bar{W} \tag{2.9}$$

donde: δ es la distancia de compatibilidad,
E es el número de genes excedentes,
D es el número de genes disjuntos,
$\bar{W}$ es el promedio de diferencias entre genes coincidentes,
N es el número de genes en el genoma más grande para normalizar los términos, y
c_1, c_2, c_3 son coeficientes que ponderan el efecto de cada termino.

Para determinar si una red es de una especie en particular, se obtiene la distancia de compatibilidad de esta con el representante de esa especie y se compara con un margen de compatibilidad (δ_t). El representante de la especie se obtiene aleatoriamente de los miembros de esta especie de la generación anterior. Si no se encuentra una especie compatible con la red se genera una especie nueva.

Para seleccionar las redes de las que surgirá la población de la siguiente generación se utiliza una aptitud modificada que puede obtenerse mediante la siguiente ecuación:

$$f'_i = \frac{f_i}{\sum_{j=1}^{n} sh(d(i,j))} \tag{2.10}$$

donde:
f_i es la aptitud de la red i-ésima,
n es el número de redes en la generación,
$\delta(i,j)$ es la distancia de compatibilidad entre la red i-ésima y la red j-ésima,
$sh(d(i,j))$ es la función de pertenencia, la cual se hace 0 si $\delta(i,j)$ es mayor que δ_t o 1 si es menor.

Esta ecuación implica que la aptitud modificada de una red se vera disminuida entre mayor sera el número de miembros en su especie. De esta manera se asegura que haya diversidad en la población y no se pierdan estructuras que puedan mejorar la aptitud de la población.

Finalmente, si la aptitud de la población disminuyera a causa de una especie durante varias generaciones, esta especie se elimina por completo y se mantienen las especies con mejores aptitudes.

Capítulo 3

Metodología

Para poder comprobar que es posible realizar una identificación de un sistema mediante el algoritmo NEAT se seguirán los siguientes pasos:

- Recolección de datos
- Evolucionar redes hasta obtener una que se comporte según los datos
- Validación de la red

Aunque, como se menciono en la sección 2.4, es posible identificar sistemas más complejos con redes neuronales, esta metodología se enfocara en la identificación de un motor de corriente directa. A pesar de esto, la metodología es lo suficientemente general como para ser utilizada con una variedad de sistemas electromecánicos. Se hablara más a profundidad del sistema a identificar en la sección 4.1.

3.1. Recolección de datos

Siguiendo los pasos de la identificación presentados anteriormente, el primer paso para hacer una identificación con NEAT es obtener datos del sistema. Para la recolección de datos se utiliza una tarjeta de adquisición de datos y un ordenador para controlarla y almacenar lo datos al realizar los experimentos. La tarjeta provee la señal de entrada al sistema y recolecta la señal de entrada y la señal de salida. El esquema general de los experimentos se puede apreciar en la figura 3.1.

La tarjeta utilizada para realizar los experimentos es un dispositivo de adquisición de datos (DAQ) de National Instruments, USB-6255. Este DAQ cuenta con 80 entradas analógicas, 2 salidas analógicas y 24 puertos de entrada/salida digitales, además de una conexión USB para la comunicación con una PC [16]. La resolución de las entradas analógicas es de 16 bits al igual que el de salidas analógicas. La cantidad máxima de muestras que las entradas analógicas pueden tomar es de 1.25 millones por segundo con un rango máximo de $\pm 10[V]$. Las salidas analógicas pueden entregar un rango de $\pm 10[V]$ con $\pm 5[mA]$ con actualización de hasta 2.86 veces por segundo. Dadas sus características este DAQ es ideal para su uso durante la adquisición de datos.

Dado que no hay un consenso en la señales utilizadas para la identificación mediante RNA, se utilizan 2 funciones distintas. En la primera se utilizara una señal chirp según la ecuación 2.4 y en la segunda una secuencia binaria pseudoaleatoria según la ecuación 2.6. Los parámetros de las señales son los siguientes:

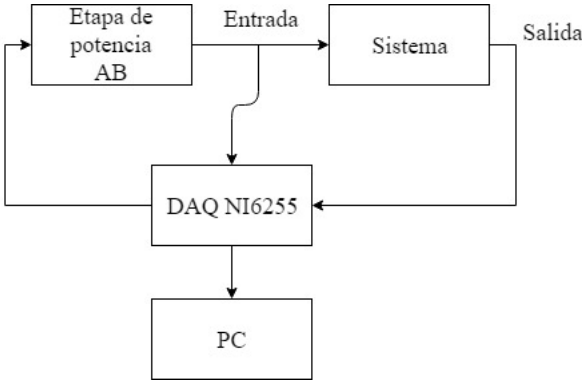


Figura 3.1: Diagrama general de los experimentos

para la señal chirp $T_0 = 10[s]$, $k_1 = 0{,}1$, $k_2 = 10$ y $A = 10[V]$; y para la señal pseudoaleatoria $p_0 = 0{,}5$ y $A = 10[V]$.

La amplitud de ambas no sobrepasaran el rango de $\pm 10[V]$ para poder utilizar las entradas analógicas de la DAQ directamente. Para aumentar la corriente disponible en las señales de entrada al sistema se utiliza una etapa de potencia tipo AB.

3.1.1. Interfaz

La interfaz para la comunicación con el DAQ en la PC se realiza con Visual Studio y el software NI-DAQmx 19.1 [1].

La interfaz utilizada fue generada a partir de ejemplos proporcionados en la pagina de National Instruments y se presenta en la figura 3.2.

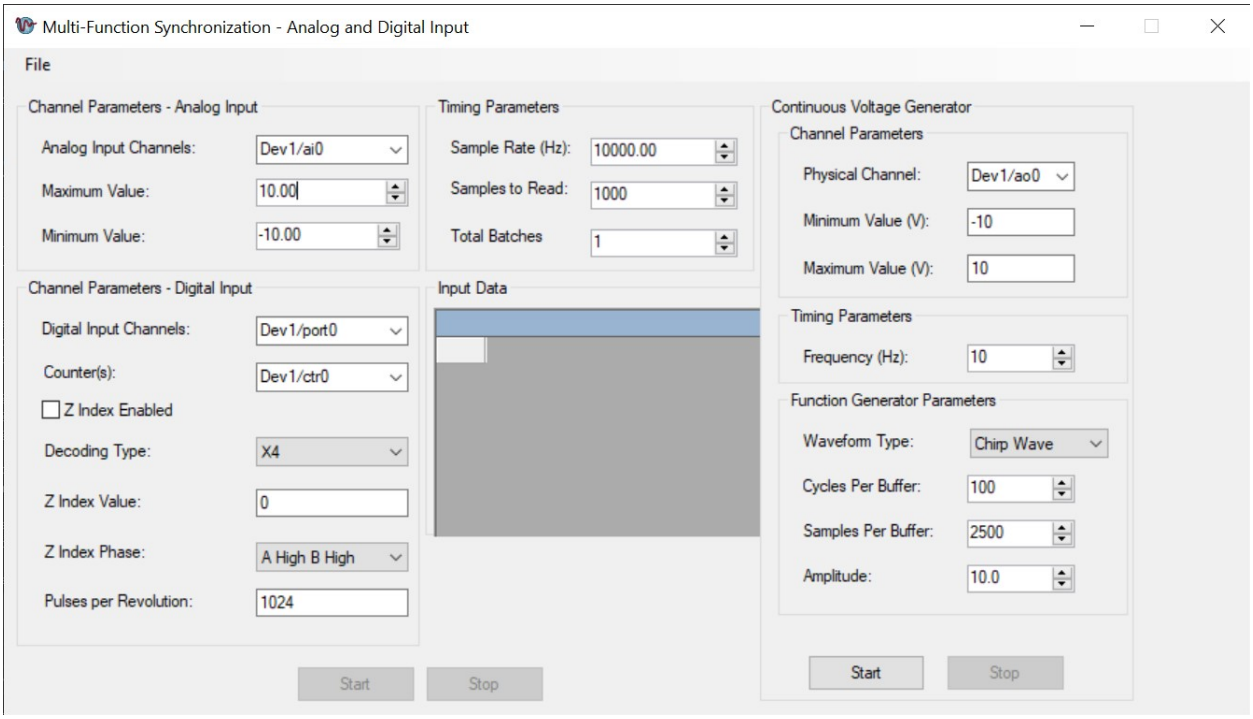


Figura 3.2: Pantalla principal de la interfaz para control de la DAQ

3.2. Evolución

Dado que el algoritmo NEAT es relativamente famoso, muchas implementaciones se han hecho en varios lenguajes de programación [4]. Se probaron las implementaciones [2] y [3], sin embargo la seleccionada es SharpNEAT [11] ya que es la mejor documentada, más cómoda de utilizar, más completa, tiene una interfaz gráfica hecha en Windows Forms y contiene otros métodos de neuroevolución como HyperNEAT.

Otra característica que tiene SharpNEAT y que es importante mencionar es la inclusión de una característica adicional agregada al algoritmo NEAT. Esta modificación consiste en realizar una búsqueda en el espacio de soluciones con solo operaciones de reproducción que disminuyen la complejidad de las redes en la población, como la remoción de conexiones o nodos sin más de 2 conexiones. Esta búsqueda de simplificación se intercala con la búsqueda normal de NEAT, a la que se le conoce como complejificación. Los procesos intercalados permite realizar una búsqueda más rápida y evita estancamiento con soluciones muy complejas pero con pocas probabilidades de mejora. [10].

Para utilizar a SharpNEAT se crea un nuevo experimento con una función de aptitud apropiada. La función utilizada en este trabajo es la siguiente:

$$Apt = \Sigma_{i=0}^{N}(|D_i| - (K_p|D_i - S_i| + K_v|\dot{D}_i - \dot{S}_i|)) \tag{3.1}$$

donde:
Apt es la aptitud de la red,
D_i es el dato a la salida i-ésimo recolectado,
S_i es la respuesta de la red al dato de entrada i-ésimo recolectado,
K_p y K_v son constantes para ponderan el efecto de los errores sobre la aptitud,

y los puntos sobre las letras indican que los datos son derivados con respecto al tiempo. El método numérico utilizado para derivar es el de diferencias finitas.

Esta ecuación esta basada en la función de consto de mínimos cuadrados. Sin embargo, dado que SharpNEAT solicita solo valores positivos y busca maximizar la aptitud, es necesario restar el costo a un valor positivo. El valor positivo es la suma de todas la magnitudes de los datos recabados ya que es ese el error que obtendría la red si siempre devolviera resultados nulos. Cualquier aptitud negativa es considerada cero. Esto permiten que la diversidad de la población sea lo suficientemente grande para que existan estructuras que lleven a resultados favorables más adelante en la evolución pero que la búsqueda no sea demasiado azarosa.

El motivo de añadir el error entre las derivadas de los datos deseados y los producidos por la red es el de darle ventaja a los individuos que producen una salida sin cambios bruscos. Esta ventaja puede ser controlada mediante los valores de K_p y K_v.

A pesar de que la estructura de la red es mayormente determinada mediante el algoritmo, aún es necesario indicar el número de entradas y de salidas deseadas. Estas son el número de entradas y salidas del sistema a identificar. También se agregan una cantidad extra de entradas igual al número de salidas del sistema, esto con el propósito de poder indicar las condiciones iniciales del sistema. Estas salidas adicionales reciben la salida anterior de la red. Además se agrega una entrada extra que recibe la diferencia de tiempo que ha pasado entre la recepción del dato anterior y el actual. Finalmente, la función de activación que se utiliza en todas las neuronas de la red es una función lineal pura.

SharpNEAT incluye dos aplicaciones para realizar la evolución. La primera incluye una interfaz gráfica (GUI) en la que se puede configurar los parámetros de la evolución y monitorear la evolución observando la aptitud de las poblaciones, la cantidad de especies, la complejidad, entre otras. Además, muestra gráficamente a la red campeona de cada generación y permite elegir donde guardar los datos producidos o inicializar poblaciones en base a genomas preexistentes. Imágenes de la interfaz se presentan en las figuras 3.3, 3.4, y 3.5.

La otra aplicación es la más mínima y se corre en la consola de Windows. Solo imprime la mejor aptitud de cada generación en consola pero puede ser modificada para mostrar otras variables y para guardar información importante durante la evolución. La configuración de la evolución se realiza a través de un archivo xml.

Ambas aplicaciones se ocupan para realizar la identificación. En particular, la GUI se utiliza para implementar el experimento de identificación y verificar su funcionamiento de manera rápida. La aplicación en consola se utiliza para realizar muchos entrenamientos consecutivos con distintos conjuntos de datos.

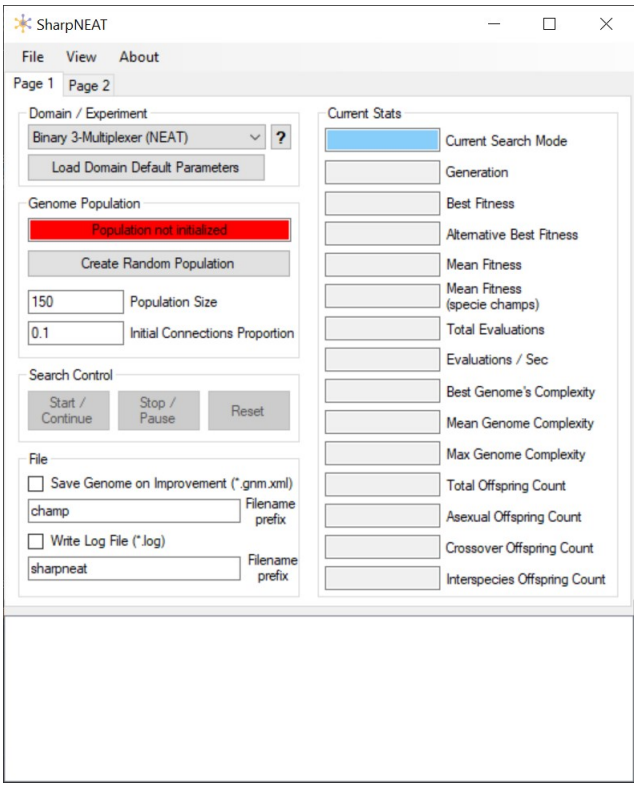


Figura 3.3: Pantalla principal de la interfaz de SharpNEAT. En esta pantalla se elige el experimento en el que utilizar NEAT. Además, indica el estado de la población en evolución y donde se guardan los genomas ganadores y la bitácora de la evolución (*log*).

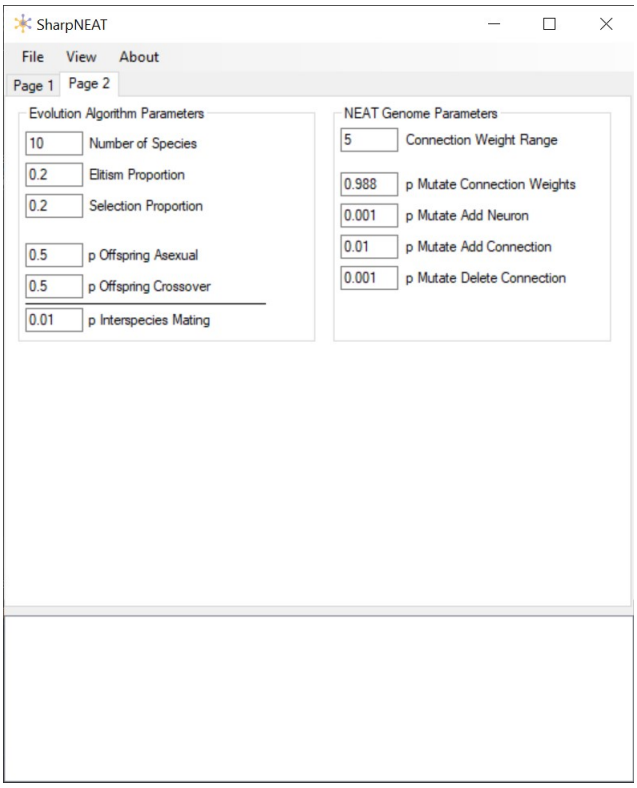


Figura 3.4: Segunda pantalla de la interfaz de SharpNEAT. En esta se indican los parámetros del algoritmo.

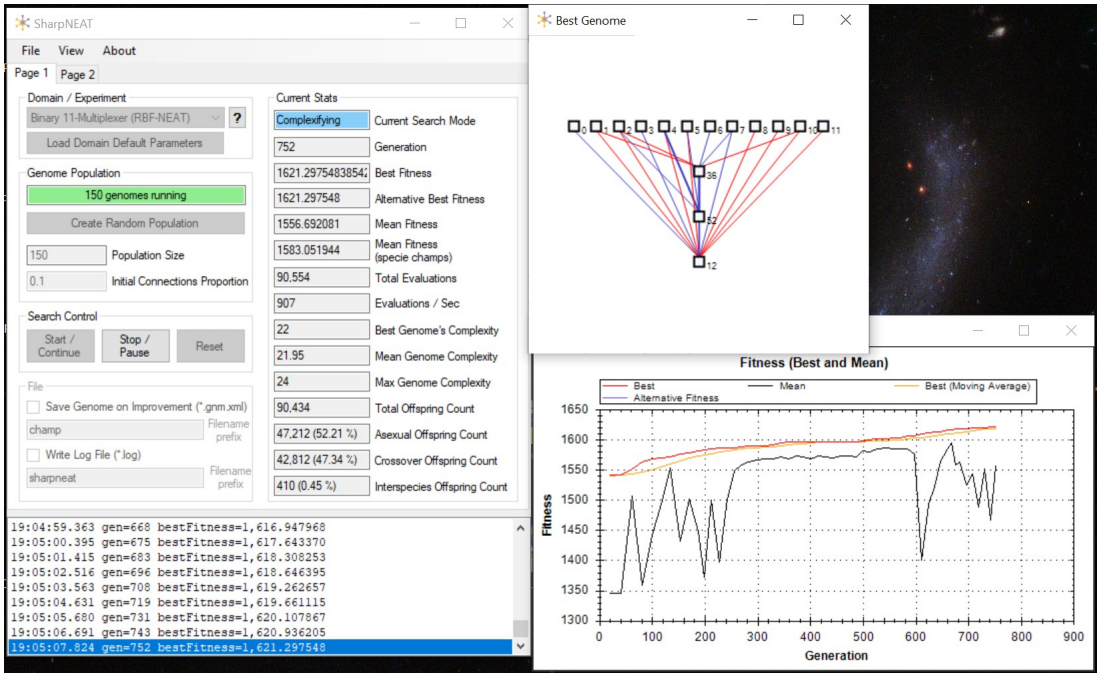


Figura 3.5: Interfaz de SharpNEAT mientras el algoritmo corre. Es posible visualizar al mejor individuo de la generación (ventana superior derecha) y las estadísticas de las generaciones durante el proceso (ventana inferior derecha).

3.3. Validación

Para validar el comportamiento de la red ganadora se propone un lazo de control proporcional el cual controla al sistema real así como a la red ganadora y se comparar ambos comportamientos. Este control también se utiliza con un modelo teórico del sistema para igual comparar su eficacia contra la de la red. El lazo de control propuesto se puede ver en la figura 3.6. Para la implementación del control en el sistema real se utiliza igualmente la DAQ.

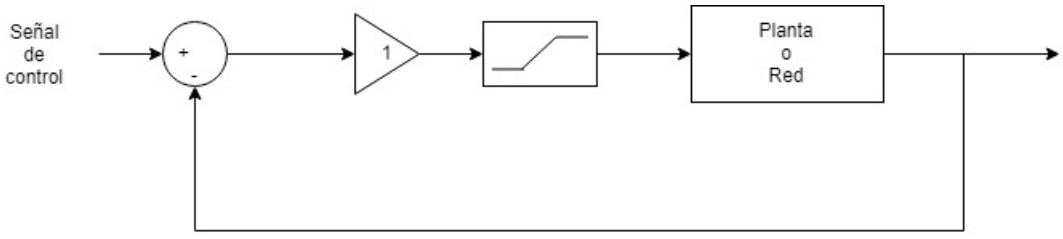


Figura 3.6: Este es el lazo de control que se utiliza para la validación. Se puede observar entre la ganancia unitaria y la planta un bloque de saturación que previene que la DAQ exceda los sus limites de voltaje.

Capítulo 4

Experimento

4.1. Motor de Corriente Directa

El primer experimento consiste en identificar un motor de corriente directa. El motor que se utiliza es un motor Pittman de la serie 14204 a 12.0[V]. Este motor se puede observar en la figura 4.1. Las características del motor proporcionadas por el fabricante se pueden observar en el Cuadro 4.1. Estos datos se obtienen de [5].

Figura 4.1: Motor Pittman 14204 12.0 V con codificador

Característica	Unidades	Valor
Voltaje máximo	V	12.0
Inercia del rotor J	Nm^2	$2{,}56041 \times 10^{-4}$
Constante del par K	$\frac{Nm}{A}$	0.031
Constante de amortiguamiento D	Nms	$1{,}21 \times 10^{-5}$
Resistencia de la armadura R_a	Ω	0.27
Inductancia de la armadura L_a	H	$0{,}4 \times 10^{-3}$

Cuadro 4.1: Características del motor Pittman 14204 12.0 V

El motor además tiene integrado un codificador rotativo incremental de cuadratura 4 con resolución de 1024 pulsos por revolución del cual se obtiene la información de la posición angular del motor.

4.1.1. Modelo del Sistema

El modelo que se utiliza en la etapa de validación es el siguiente [21]:

$$K\omega_p(t) = -R_a i_a(t) - L_a \frac{di_a(t)}{dt} + V_t(t) \tag{4.1}$$

$$Ki_a(t) = J\frac{d\omega_p(t)}{dt} + D\omega_p(t) + T_L(t) \tag{4.2}$$

donde:
$\omega_p(t)$ es la velocidad del rotor,
$V_t(t)$ es el voltaje de las terminales del motor,
$i_a(t)$ es la corriente de la armadura del motor,
$T_L(t)$ es el par del motor,
J es la inercia del rotor,
K es la constante del par,
D es la constante de amortiguamiento,
R_a es la resistencia de la armadura, y
L_a es la inductancia de la armadura,

Los parámetros de este modelo son conocidas gracias a los datos proporcionados anteriormente y, dado que durante el experimento no existe ningún par actuando sobre la flecha del motor, se puede suponer que $T_L(t) = 0$. Para un fácil manejo del modelo se utilizará su representación en espacio de estados considerando los siguientes estados, entradas y salidas:

$$x = \begin{pmatrix} i_a(\mathrm{t}) \\ \omega_p(\mathrm{t}) \\ \alpha_p(\mathrm{t}) \end{pmatrix},\ u = V_t(t),\ y = \alpha_p(t),$$

donde $\omega_p(t) = \frac{d\alpha_p(t)}{dt}$ para obtener a la posición angular como salida del sistema.

El modelo resultante es el siguiente:

$$\dot{x} = \begin{pmatrix} -675. & -77{,}5 & 0 \\ 121{,}074 & -0{,}0472581 & 0 \\ 0 & 1 & 0 \end{pmatrix} x + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} u$$

$$y = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} x$$

Finalmente, dado que para validación se obtuvieron en promedio cada 0.0153726[s] se discretiza el modelo mediante un reten de orden cero. El modelo a utilizar para la para la validación es el siguiente:

$$x(k+1) = \begin{pmatrix} -0{,}709293 & -0{,}216028 & -5{,}58613 \\ 0{,}270437 & 0{,}798309 & -5{,}19663 \\ 0{,}00207865 & 0{,}0138223 & 0{,}960057 \end{pmatrix} x(k) + \begin{pmatrix} 363{,}383 \\ 338{,}046 \\ 2{,}59831 \end{pmatrix} u(k)$$

$$y(k+1) = \begin{pmatrix} 0{,}0000159771 & 0{,}000106242 & 0{,}0150655 \end{pmatrix} x(k) + 0{,}0199713u(k)$$

4.1.2. Circuito para la obtención de datos y validación

Como se menciono anteriormente se utiliza la DAQ junto con una etapa de potencia AB para la obtención de los datos y su validación. El circuito utilizado se puede observar en las figuras 4.2 y 4.3.

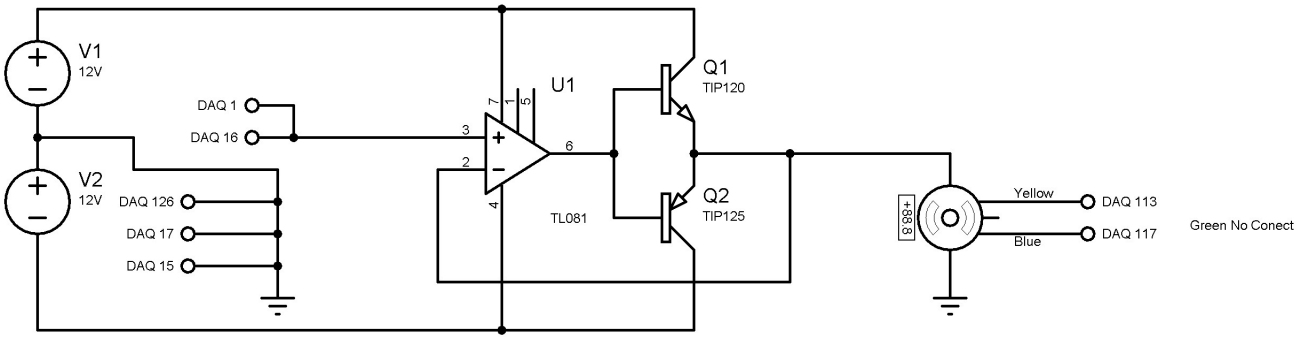


Figura 4.2: Diagrama del circuito utilizado.

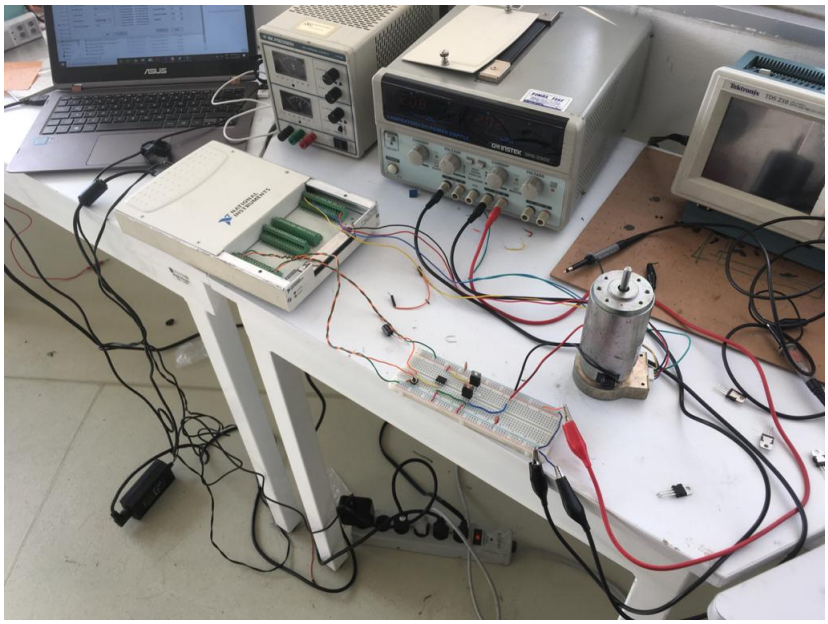


Figura 4.3: Disposición para la obtención de datos y validación.

Procedimiento de evolución

Tomando en cuenta los mencionado anteriormente, la estructura básica de las redes a evolucionar es de 3 entradas y 1 salida: una entrada del voltaje en las terminales del motor, otra para la diferencia de tiempo, una más para la posición actual del motor y una salida que indica la posición resultante del motor. Las unidades utilizadas para voltaje, posición y tiempo son volts, radianes y segundos, respectivamente.

Los parámetros de NEAT que se utilizan se presentan en la figura 4.4. Para evitar experimentar con estos parámetros se utilizan unos que ya hayan sido probados. En este caso, se eligieron los parámetros indicados por [20] para el experimento XOR. La única excepción es el número de especies. El motivo de esto es permitir más diversidad durante la evolución. El tamaño de población para cada generación es de 1000 individuos. Las constantes K_p y K_v utilizadas en la ecuación de aptitud 3.1 son 1 y 0.8 respectivamente. Estos valores fueron elegidos para evitar que la evolución se enfoque demasiado en lograr la velocidad del motor y no la posición.

4.2. Validación

Para la validación de este experimento se utilizaron también 2 aplicaciones en conjunto. Otra interfaz gráfica basada en la utilizada para la obtención de datos y se modifico la aplicación de consola de sharpNEAT para realizar la validación inmediatamente después de la evolución. La pantalla de la interfaz se presenta en la figura 4.5.

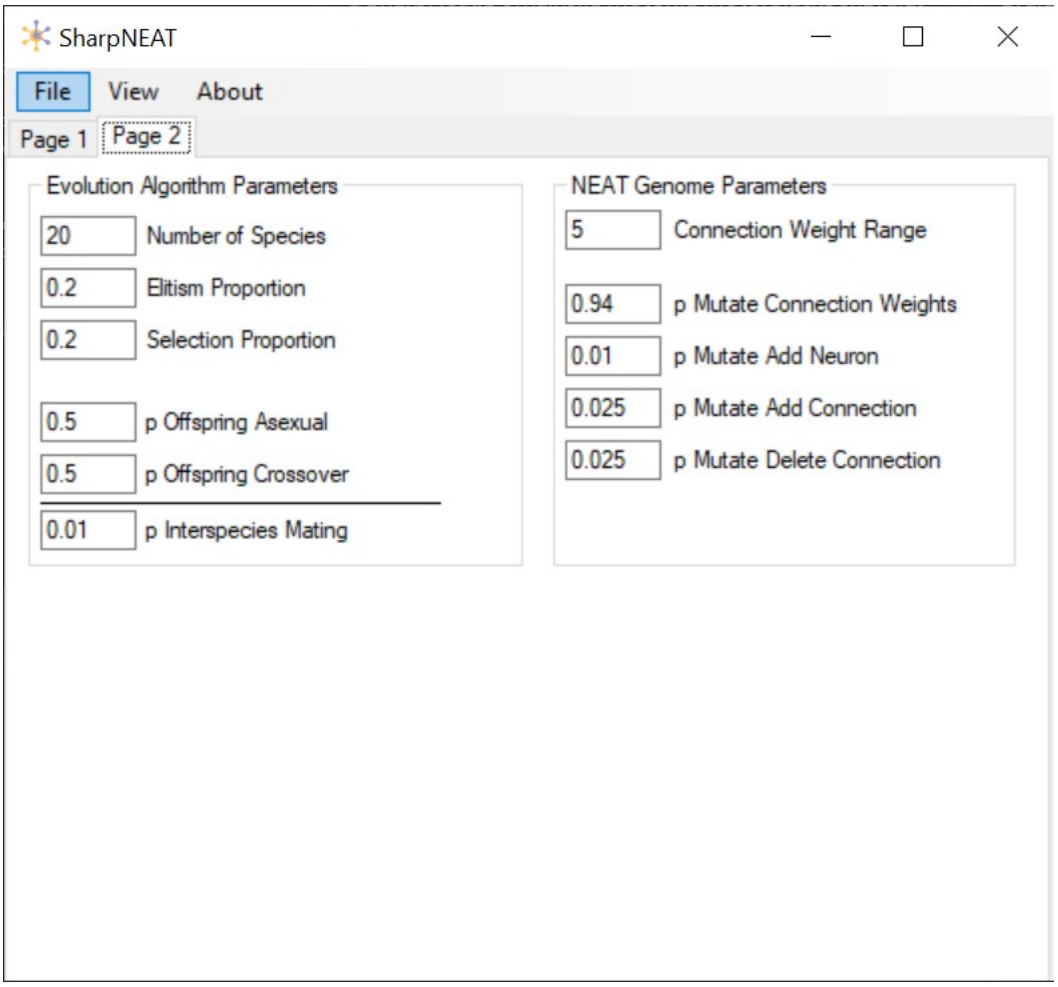


Figura 4.4: Parámetros del algoritmo NEAT utilizados durante la evolución

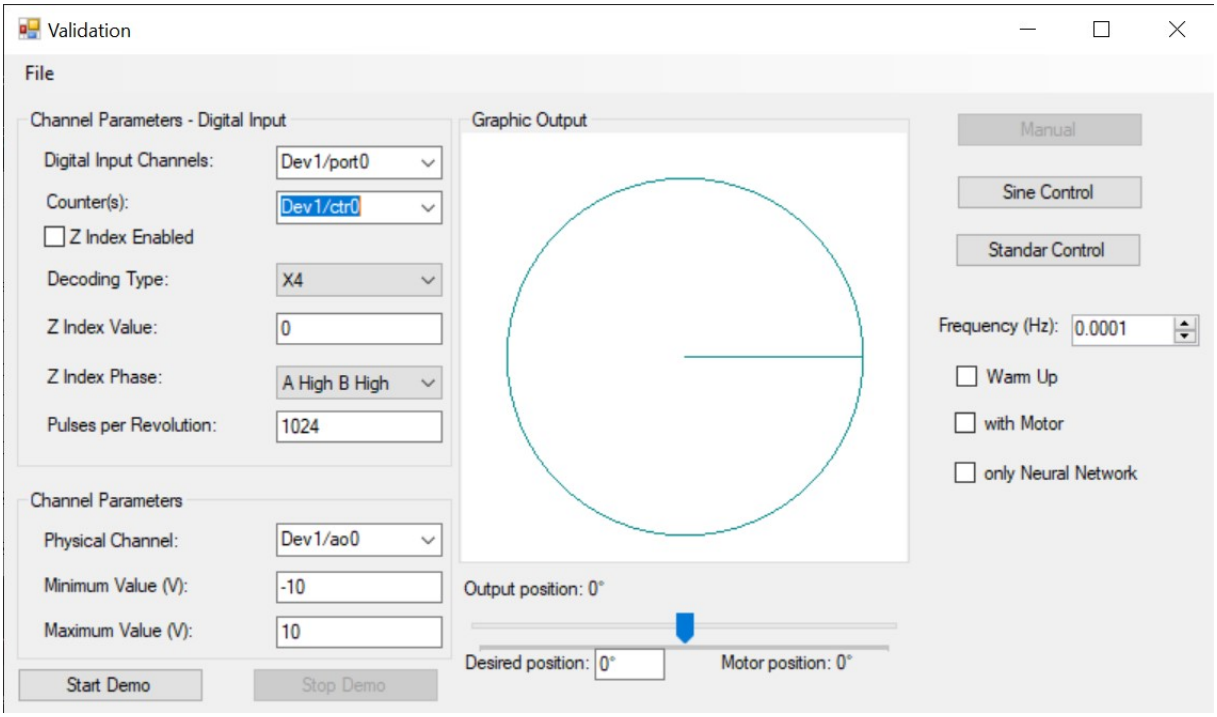


Figura 4.5: Ventana utilizada para la validación

En la interfaz la señal de control puede ser dictada manualmente o mediante una señal senoidal, a la cual se le puede cambiar su frecuencia. Para mantener consistente la validación entre varias redes se utiliza una señal de control estándar, igual a la que se utiliza con el motor real. Esa señal de control se puede apreciar en la figura 4.6.

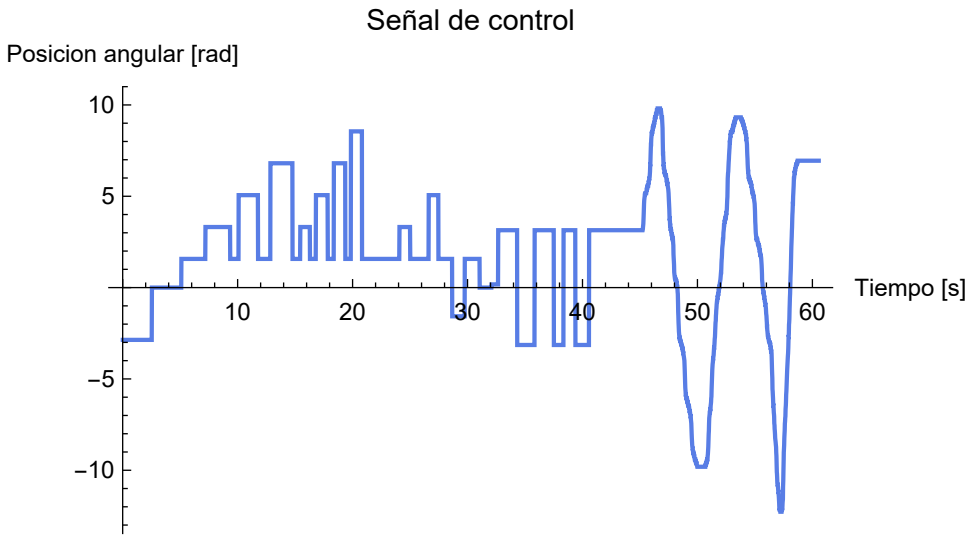


Figura 4.6: Señal de control producida manualmente utilizada para la validación.

A pesar de que la interfaz también tiene la capacidad de realizar la validación de redes resultado de la evolución, se prefirió utilizar la validación integrada en la aplicación de consola para automatizar el procedimiento.

Para poder medir la calidad de la identificación y poder comparar a las redes entre si y con el modelado clásico, se obtiene el error absoluto y relativo que existe entre la posición real del motor y la salida del modelo y las redes. De los datos de error se obtiene el error máximo, el promedio y el FPE con la ecuación 2.7. El número de parámetros de las redes se considera como el número de conexiones entre neuronas.

Capítulo 5

Resultados

A continuación se presentan algunas de gráficas representativas de los datos obtenidos durante cada etapa de la identificación. El resto de las gráficas, así como los códigos para generarlas y el resto de las aplicaciones utilizadas se encuentran en el siguiente repositorio: `https://github.com/Rodrigo2118/IDMotor.git`.

5.0.1. Datos obtenidos

En la obtención de datos se adquirieron varios conjuntos de datos en sesiones consecutivas. La frecuencia a la que se recabaron los datos fue de 1000[Hz]. Para observar si existe alguna relación entre el tiempo de muestreo y la calidad de la identificación también se obtuvieron algunos conjuntos de datos con una frecuencia de 100[Hz].

Algunos de los datos recabados se muestran en la figura 5.1. En la parte superior de las gráficas se muestra el nombre del archivo en el que se guardaron. Los archivos con la partícula "mr" son datos recabados con menor resolución. Finalmente, los archivos con la partícula "tj" son conjuntos de datos iguales a sus homónimos sin la partícula pero con la diferencia de que le faltan aproximadamente la mitad de los datos. Estos conjuntos se hicieron para dar más variedad a los conjuntos de entrenamiento en cuanto a diferencias en el tiempo de muestreo.

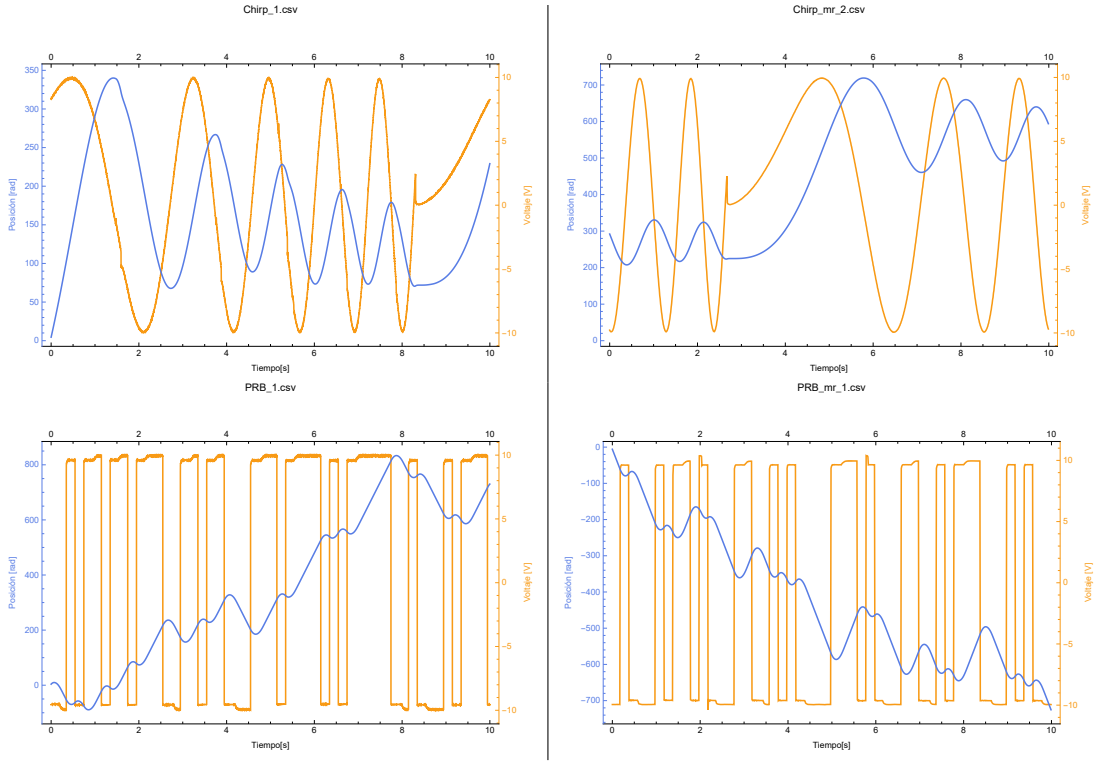


Figura 5.1: Algunos datos obtenidos. En linea naranja se muestran los datos del voltaje en las terminales (entrada) y en linea azul los datos de posición angular del eje del motor (salida).

5.0.2. Entrenamiento

Inicialmente, todos los entrenamientos realizados tuvieron una duración de 20 minutos. El desarrollo de estos entrenamientos observado desde los valores de aptitud y complejidad de las redes de algunos conjuntos de entrenamiento se pueden observar en la figura 5.2.

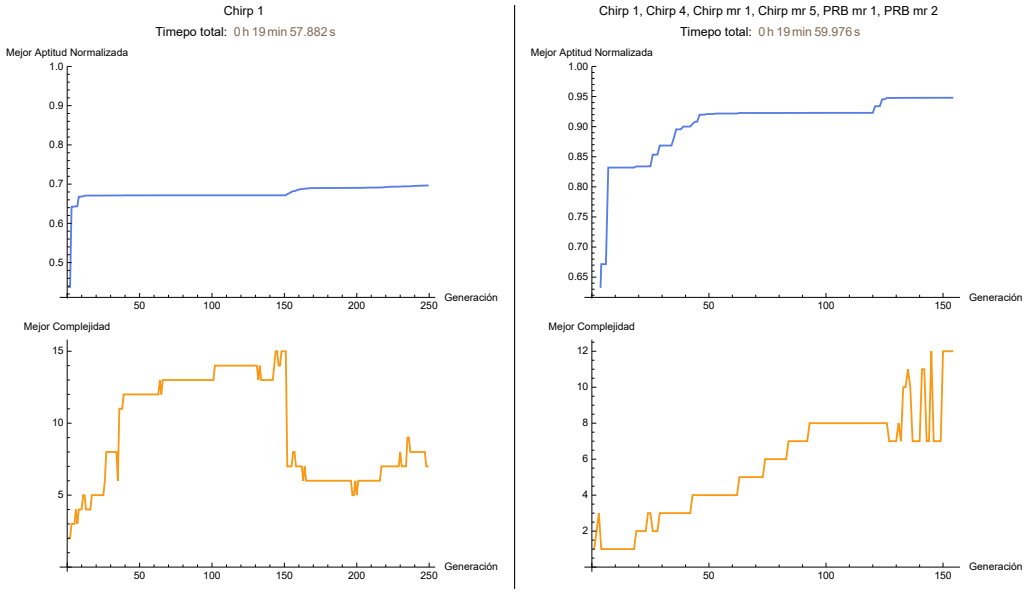


Figura 5.2: Algunas gráficas de aptitud y complejidad contra el número de generaciones en el entrenamiento. Es posible apreciar que mayor complejidad no siempre significa mayor aptitud.

Algunas redes y sus desempeños se observan en las figuras 5.3 y 5.4.

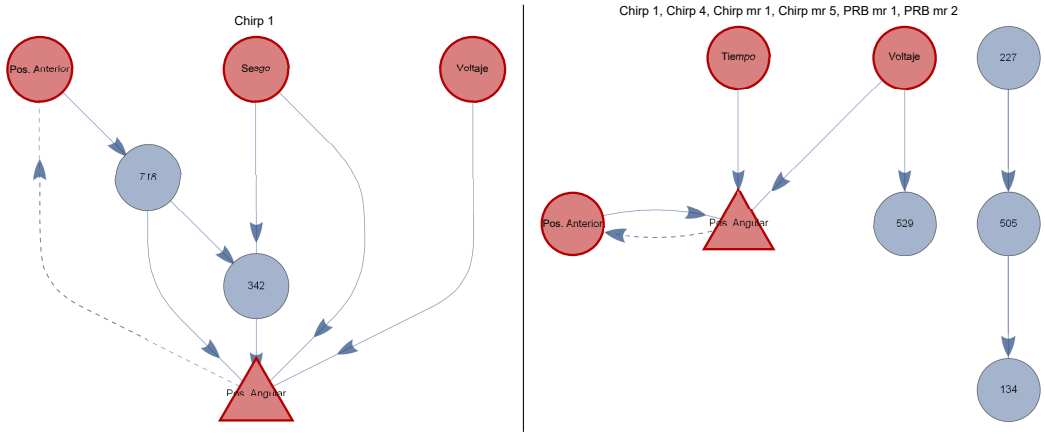


Figura 5.3: Topología de las mejores redes resultado de los entrenamientos mostrados en la figura 5.2. La conexión en linea punteada no existe en el genoma de las redes en realidad, pero indica la realimentación a la red con su respuesta anterior. En la red derecha se puede observar una subred que no interviene en la red principal. Estas se hubiera eliminado en un proceso de simplificación posterior. También se pueden eliminar del genoma manualmente.

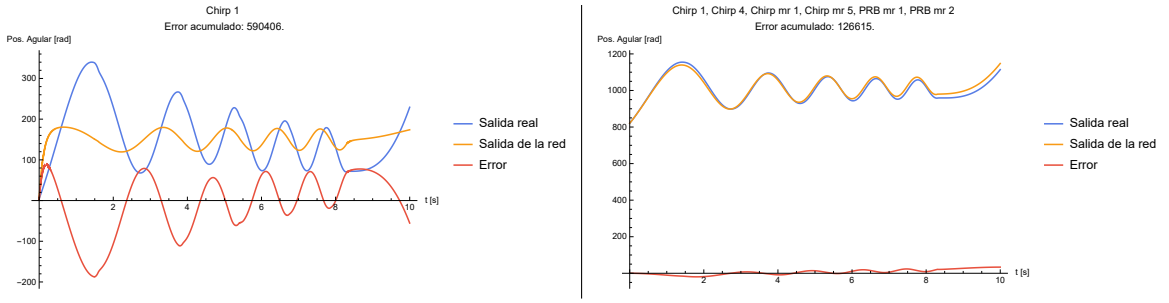


Figura 5.4: Desempeño de las redes mostradas en la figura 5.3. Por inspección visual, se puede verificar que la aptitud de la red de la derecha es mejor que la de la izquierda.

5.1. Validación

En la figura 5.5 se observa el comportamiento del modelo clásico comparado con el motor real. Como se puede observar, a pesar de seguir de manera adecuada el comportamiento del motor y tener un FPE muy pequeño, el modelo comete algunos errores de hasta casi 4 [rad]. Esto puede ser causado por la imprecisión de los datos proveídos por el fabricante, además de la simplificación de la planta al hacer su modelado. El resto de los datos utilizados para su comparación con las redes se muestran en el cuadro 5.1.

Error Absoluto Máximo	3.76758 [rad]
Error Absoluto Promedio	0.0816399
Desviación Estándar	0.462756
Error Relativo Máximo	70.6824 %
Error Relativo Promedio	0.02116019 %

Cuadro 5.1: Datos resultado de la validación del modelo clásico.

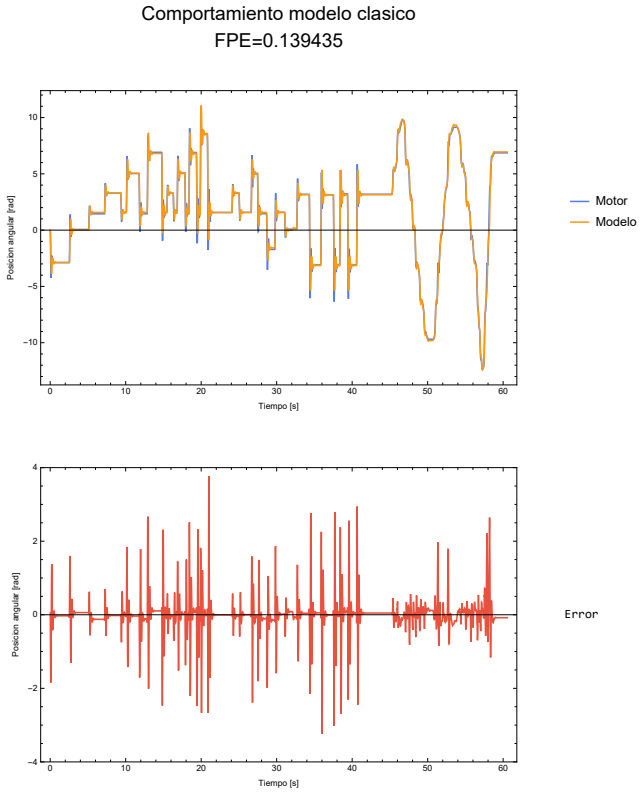


Figura 5.5: Comparación entre el comportamiento del modelo clásico del motor y el motor real. En general, el error del modelo es muy bajo pero aumenta considerablemente en los cambios bruscos de posición.

En la figura 5.6 se observa la misma comparación pero con 2 redes en vez del modelo clásico. Con una simple inspección visual se observa que las redes no logran seguir el comportamiento del motor adecuadamente. Esto también se comprueba con los valores tan altos de FPE.

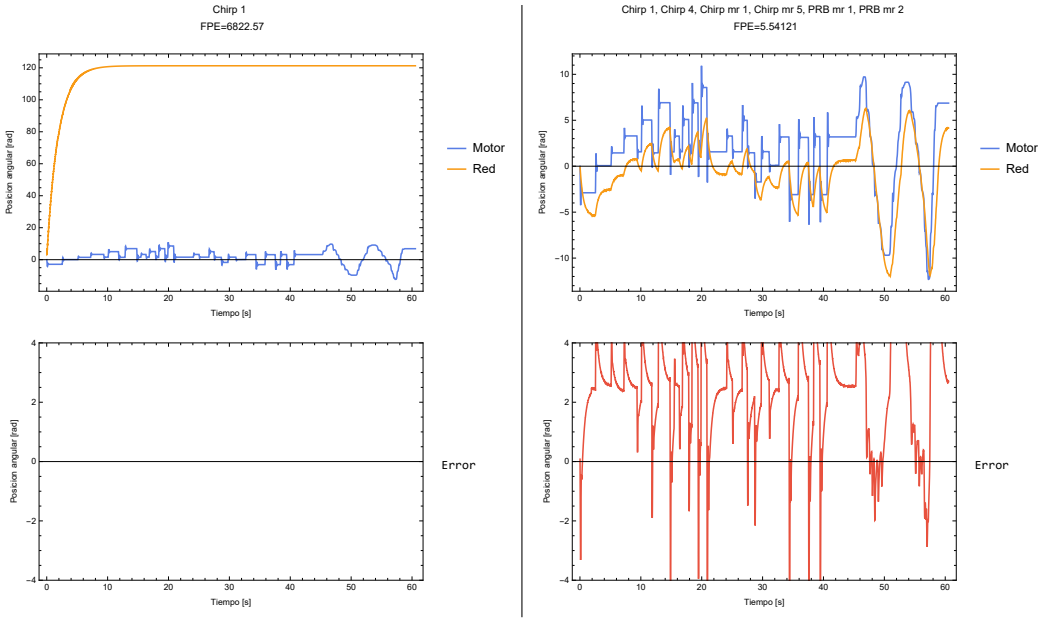


Figura 5.6: Comportamiento de las redes a la señal de control en la figura 4.6 y el controlador proporcional. Las gráficas inferiores son el error entre el comportamiento del motor y el de la red. En la gráfica de error de la izquierda, no se observa la linea del error por que es siempre mayor a 4 [rad]. La red de la derecha, a pesar de tener mejor desempeño que la de la izquierda, sigue estando muy alejada del comportamiento real.

Sin embargo, algunas de las redes resultantes del entrenamiento resultaron muy efectivas al seguir el comportamiento del motor. Estas se muestran en la figura 5.7. A pesar de no tener un FPE igual o menor que el modelo clásico, estas redes logran predecir el comportamiento del motor con gran precisión.

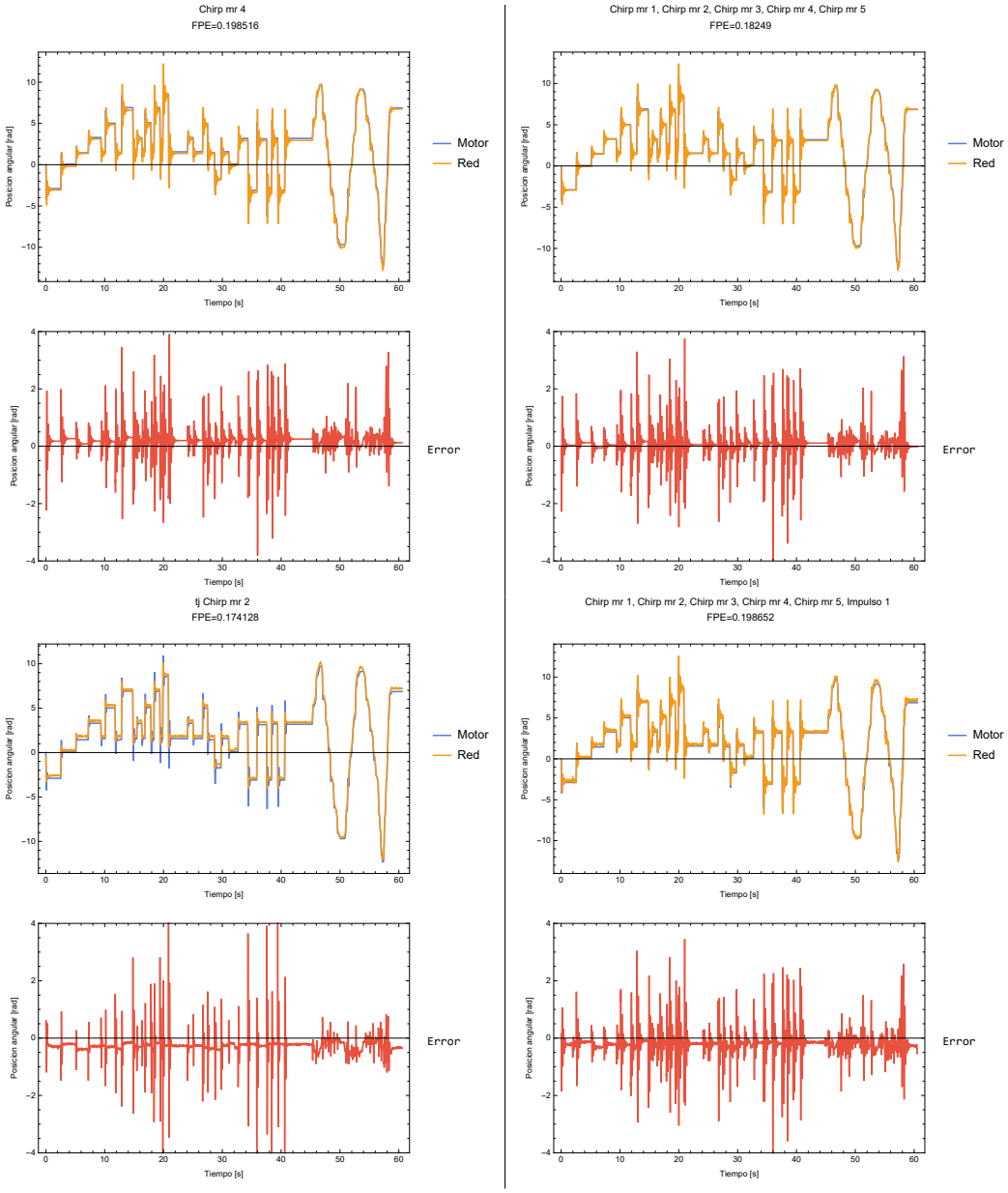


Figura 5.7: Validación de las redes con FPEs más bajos. Las redes logran acercarse al comportamiento del motor a excepción de los cambios bruscos en posición, muy similar al modelo clásico en la figura 5.5. Sin embargo, todas tienen un error constante cuando el motor esta estático.

5.2. Entrenamientos y validaciones posteriores

Para comprobar que los resultados de los entrenamientos son reproducibles y verificar si la identificación mejoraba con respecto al tiempo de evolución, se repitieron los entrenamientos con los datos utilizados para entrenar las redes de la figura 5.7, además de otras combinaciones que no resultaron tan

prometedoras como control. La evolución de estos entrenamientos es de 40 minutos.

La validación de los entrenamientos repetidos se muestra en la figura 5.9.

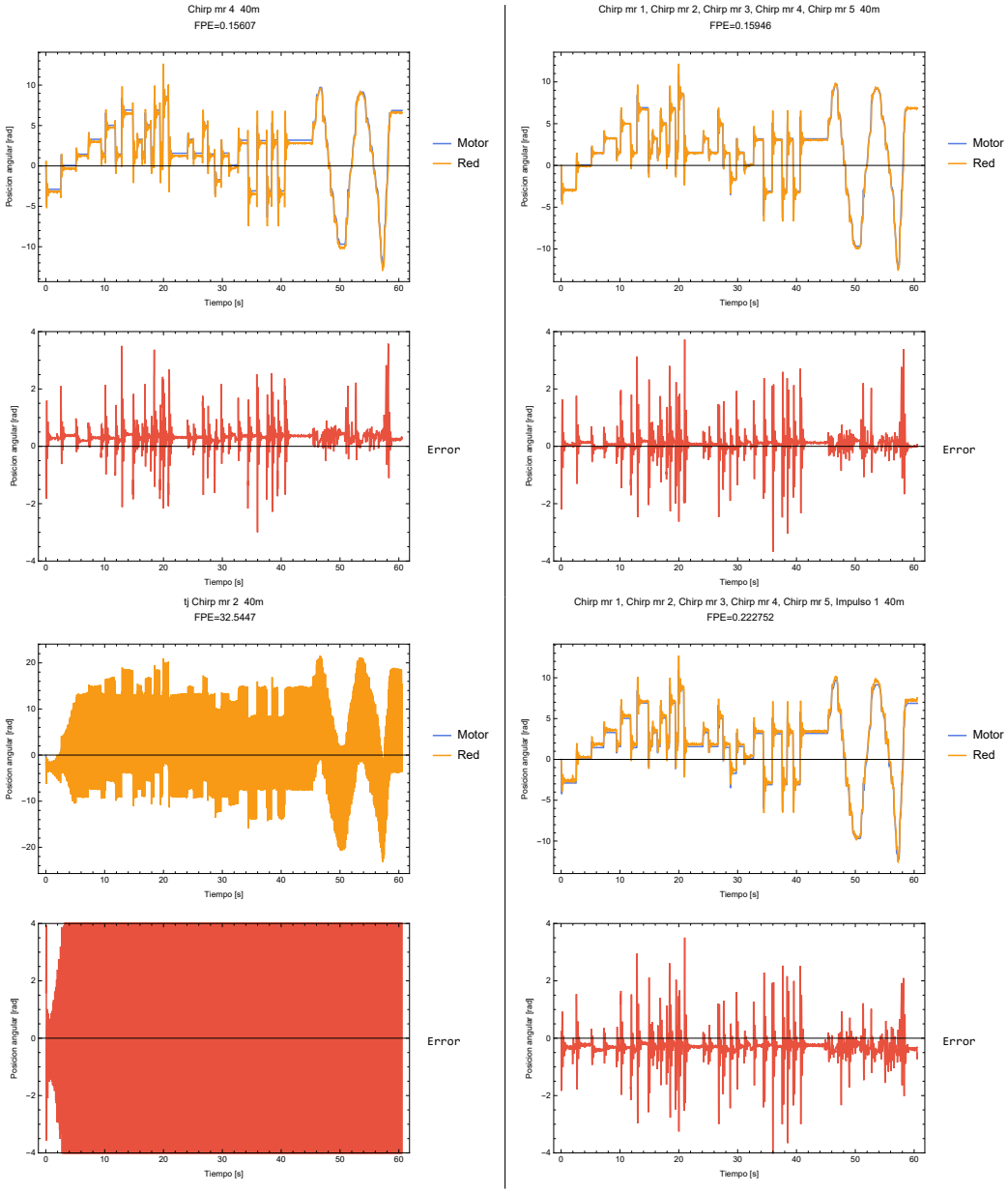


Figura 5.8: Validación de las redes resultantes de los entrenamientos repetidos. En la mayoría de los casos, los nuevos entrenamientos mejoraron el desempeño de las redes disminuyendo el error constante, aunque, en el caso de la red inferior izquierda, el comportamiento empeoro.

Los entrenamientos con los datos Chirp mr 4 y los Chirp mr 1 a 5 redujeron su FPE en comparación de sus contra partes de 20 minutos. Lo contrario sucedió con los otros 2 entrenamientos, aunque en mayor medida particularmente con los datos tj Chirp mr 2.

Finalmente, en busca de mejorar el FPE del modelo clásico con las redes resultantes se volvieron a repetir los entrenamientos de los mejores 2. Sin embargo, a diferencia de los entrenamientos de 40 minutos, los entrenamientos finales solo duran 30 minutos pero se utiliza la red resultante de los entrenamientos anteriores como parte de la población inicial de la evolución (gen semilla o *seed*).

La validación de estos 2 entrenamientos se muestra en la figura 5.9. Al igual que los entrenamientos anteriores, el aumento de tiempo no resulto en una mejor identificación para ambos conjuntos de datos, solo para el Chirp mr 4, con un FPE ligeramente mejor al de el modelo clásico.

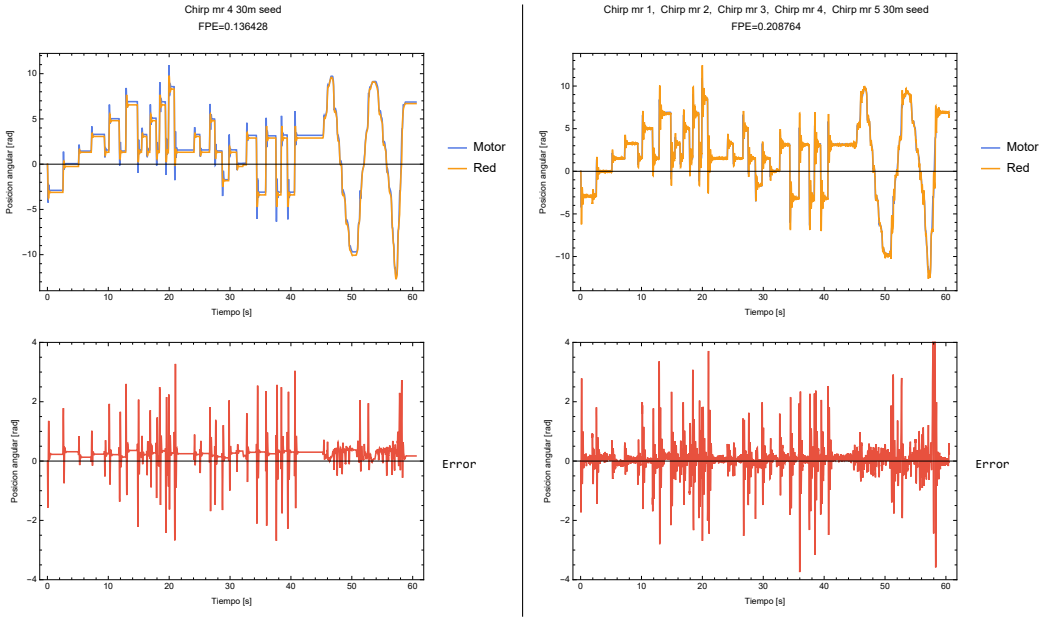


Figura 5.9: Validación de las redes resultantes de los entrenamientos con genes semilla. Al igual que en los entrenamientos repetidos, el error constante disminuyó.

Los datos generales de todos los entrenamientos realizados se encuentran en la figura 5.10.

Entrenamiento	Datos totales	Generaciones	Aptitud máxima posible	Aptitud	Aptitud (%)	EAA (Entrenamiento)	Complejidad	Error absoluto máximo	Error absoluto promedio	Desviación Estandar	Error relativo máximo	Error relativo promedio	FPE
Chirp 1	10000	249	3.13651×10^6	2.18474×10^6	69.655	590406.	7	133.594	118.09	14.0411	15813.2	37.0466	6822.57
Chirp 1, Chirp 2, Chirp 3, Chirp 4, Chirp 5	50000	83	7.20478×10^7	5.15794×10^7	71.5905	2.77559×10^6	16	714.917	700.859	19.7124	91605.8	219.735	247421.
Chirp 1, Chirp 2, Chirp 3, Chirp 4, Chirp 5 40m	50000	731	7.20478×10^7	5.35353×10^7	74.3052	2.91889×10^6	18	522.073	414.414	35.0928	54009.	131.956	92326.4
Chirp 1, Chirp 4, Chirp mr 1, Chirp mr 5	22000	120	2.63332×10^7	2.58463×10^7	98.1512	111157.	5	23.6715	7.38595	3.39947	1122.62	2.34701	42.4649
Chirp 1, Chirp 4, Chirp mr 1, Chirp mr 5, PRB mr 1, PRB mr 2	24000	154	2.9302×10^7	2.77782×10^7	94.7995	126615.	6	12.0432	2.58127	1.79445	192.502	0.806325	5.54121
Chirp 1, Chirp 4, Chirp mr 3, Impulso 1, PRB 1, PRB 4, PRB mr 2	43000	102	8.57378×10^7	8.28352×10^7	96.6145	172565.	8	10.5929	1.86613	1.52536	170.562	0.596279	3.51991
Chirp 1, Chirp mr 1, Chirp 2, PRB mr 1	22000	149	1.27135×10^7	8.54131×10^6	67.1828	1.10478×10^6	21	609.608	338.038	47.84	49321.7	107.493	61288.7
Chirp 1 40m	10000	3462	3.13651×10^6	3.06156×10^6	97.6104	48594.3	27	16.1442	0.842612	2.05452	880.92	0.287534	3.64692
Chirp 2	10000	457	8.37867×10^6	8.30778×10^6	99.154	41928.4	18	37.5286	15.6626	6.32054	1691.81	5.18504	159.283
Chirp 2, Chirp 4, Chirp mr 5, Chirp mr 1, PRB 1, PRB 2, PRB 3, PRB 4, PRB 5, PRB 6, PRB mr 1	83000	88	2.26049×10^8	1.80989×10^8	80.0662	2.43534×10^6	14	573.547	557.969	28.7896	73266.6	175.686	153297.
Chirp 3	10000	459	1.4289×10^7	1.33448×10^7	93.3921	525451.	11	690.772	675.311	88.6368	88463.4	211.855	217225.
Chirp 4	10000	418	2.01819×10^7	1.92477×10^7	95.3707	496631.	10	906.119	883.811	183.758	115540.	278.988	334108.
Chirp 5	10000	442	2.60616×10^7	2.50154×10^7	95.9854	555760.	19	1269.04	1253.39	177.571	163836.	393.222	741323.
Chirp 5, Chirp mr 1, PRB 5	21000	186	9.15934×10^7	7.04711×10^7	76.939	1.81655×10^7	21	1591.56	631.031	127.457	77261.1	200.249	212798.
Chirp 5 40m	10000	2124	2.60616×10^7	2.57049×10^7	98.631	260040.	21	46630.8	38064.9	12185.1	5.03559×10^6	12515.7	6.42327×10^8
Chirp mr 1	1000	3263	355963.	353418.	99.285	1514.3	28	68.0623	43.8034	17.6789	6674.22	11.8235	855.286
Chirp mr 1, Chirp mr 2, Chirp mr 3, Chirp mr 4, Chirp mr 5	5000	720	7.45002×10^6	7.43183×10^6	99.7558	3150.68	20	3.98154	0.126692	0.50282	69.0146	0.0421141	0.18249
Chirp mr 1, Chirp mr 2, Chirp mr 3, Chirp mr 4, Chirp mr 5, Impulso 1	6000	491	7.49353×10^6	7.37737×10^6	98.4498	3100.71	18	4.20431	0.231471	0.483629	99.6661	0.0831256	0.198652
Chirp mr 1, Chirp mr 2, Chirp mr 3, Chirp mr 4, Chirp mr 5, Impulso 1, PRB mr 1, PRB mr 2, tj Chirp 2, tj Chirp mr 1, tj Chirp mr 2, tj Chirp mr 3, tj PRB 2	42385	194	6.04492×10^7	2.47687×10^7	40.9744	50343.2	18	11.5554	3.91087	1.14073	654.173	1.22064	8.64573
Chirp mr 1, Chirp mr 2, Chirp mr 3, Chirp mr 4, Chirp mr 5, Impulso 1 40m	6000	2944	7.49353×10^6	7.38913×10^6	98.6068	2672.	20	4.31169	0.318818	0.474256	114.2	0.105741	0.222752
Chirp mr 1, Chirp mr 2, Chirp mr 3, Chirp mr 4, Chirp mr 5, PRB mr 1, PRB mr 2	7000	203	1.04189×10^7	1.01524×10^7	97.4425	104944.	26	5.46353	0.328169	0.655834	84.4624	0.108249	0.360136
Chirp mr 1, Chirp mr 2, Chirp mr 3, Chirp mr 4, Chirp mr 5 40m	5000	6363	7.45002×10^6	7.43267×10^6	99.7671	2594.25	26	3.72296	0.130325	0.473206	65.546	0.0429127	0.15946
Chirp mr 1, Chirp mr 2, PRB mr 2	3000	1451	3.36735×10^6	3.29929×10^6	97.979	28342.7	25	3.88161	0.255014	0.523069	118.206	0.0869043	0.230173
Chirp mr 2	1000	2427	884907.	883143.	99.8006	1087.51	23	5.60088	0.371633	0.783724	62.0579	0.137107	0.544928
Chirp mr 3	1000	2824	2.07068×10^6	2.06721×10^6	99.8323	2883.42	22	3.46427	0.617127	0.351719	29.2982	0.207268	0.291813
Chirp mr 4	1000	2818	1.47972×10^6	1.47812×10^6	99.8921	1062.87	11	3.90759	0.249852	0.486856	51.4218	0.0785037	0.198516
Chirp mr 4 40m	1000	20360	1.47972×10^6	1.47818×10^6	99.8963	993.841	34	3.57757	0.333948	0.365572	40.9228	0.113887	0.15607
Chirp mr 5	1000	2392	2.65875×10^6	2.65607×10^6	99.8991	2009.3	27	3.37898	1.05187	0.255681	116.499	0.343296	0.600232
Impulso 1	1000	2827	43509.8	36544.6	83.9916	1295.36	32	77.4175	9.65812	26.4959	7880.32	4.58461	682.385
PRB 10	20000	385	6.48443×10^7	5.35358×10^7	82.5606	2.80479×10^6	12	3058.74	3049.39	273.923	397376.	957.02	4.5635×10^6
PRB 1	10000	351	7.44651×10^6	6.61925×10^6	88.8906	530002.	13	221.235	96.3697	62.426	13130.2	33.0234	6981.
PRB 2	10000	380	2.17608×10^7	1.87851×10^7	86.3252	2.38914×10^6	35	1053.75	1037.78	42.4572	135616.	325.86	546701.
PRB 3	10000	397	3.62688×10^7	3.40711×10^7	93.9404	1.63076×10^6	25	3782.89	2599.91	1057.48	347018.	880.701	3.36721×10^6
PRB 4	10000	318	5.07322×10^7	4.7446×10^7	93.5225	2.37722×10^6	24	2507.4	2491.9	79.3919	325309.	782.876	3.13771×10^6
PRB 5	10000	274	6.51758×10^7	6.45651×10^7	99.063	290807.	5	2197.08	1055.54	626.115	143009.	370.048	767351.
PRB 6	10000	382	1.22473×10^7	8.54466×10^6	69.7676	3.10226×10^6	7	775.48	766.249	64.1838	99680.9	240.851	288609.
PRB 7	10000	468	2.39132×10^7	2.33782×10^7	97.7627	411292.	25	225.822	128.331	54.2695	16967.3	44.6024	9589.35
PRB 7 40m	10000	3354	2.39132×10^7	2.33431×10^7	97.6161	324842.	21	244390.	123839.	70128.9	1.67815×10^7	43127.1	1.01374×10^{10}
PRB 8	10000	614	3.5056×10^7	3.23396×10^7	92.2513	1.98749×10^6	34	1989.85	1702.9	516.658	224392.	558.584	1.26849×10^6
PRB 8 40m	10000	3653	3.5056×10^7	3.41467×10^7	97.4061	548576.	26	953.835	944.556	31.4266	122925.	297.201	450716.
PRB 9	10000	464	4.6211×10^7	4.5682×10^7	98.8553	297038.	11	10.3266	1.62976	1.55035	165.617	0.511537	3.31681
PRB 9, PRB 10, PRB 1, PRB 2, PRB 3, PRB 4, PRB 5, PRB 6	90000	95	3.04687×10^8	1.51211×10^8	49.6283	1.27781×10^7	22	693.773	677.561	19.4065	88654.8	212.719	232209.
PRB mr 1	1000	3764	842396.	823827.	97.7957	10936.4	25	15.1772	2.29779	2.29191	266.498	0.741407	6.70729
PRB mr 1, PRB mr 2	2000	2156	2.96887×10^6	2.92569×10^6	98.5457	13430.7	24	13.6658	1.95223	2.06745	443.478	0.631138	5.29552
PRB mr 2	1000	3531	2.12647×10^6	2.09271×10^6	98.4122	15045.6	40	725.934	12.2655	114.722	10602.	3.36616	7459.6
tj Chirp 2	14984	1156	1.25622×10^7	5.28296×10^6	42.0545	2.06857×10^6	1	16.9706	3.08966	3.21456	759.536	1.41405	12.5973
tj Chirp 2, tj Chirp mr 1, tj Chirp mr 2, tj Chirp mr 3	19447	577	1.74872×10^7	4.92679×10^6	28.1737	340264.	15	383.83	368.326	35.9888	48438.9	115.549	66256.2
tj Chirp mr 1	1491	8814	534013.	287229.	53.7869	89028.5	1	12.3033	3.19642	2.61803	1.707	1.00398	10.1713
tj Chirp mr 2	1493	8909	1.32733×10^6	808250.	60.8929	3085.12	36	5.39298	0.273057	0.439453	67.3909	0.0883417	0.174128
tj Chirp mr 2 40m	1493	32310	1.32733×10^6	433206.	32.6374	2485.26	50	14.2222	7.8332	3.61941	881.318	2.17531	32.5447
Chirp mr 1, Chirp mr 2, Chirp mr 3, Chirp mr 4, Chirp mr 5 30m seed	5000	4797	7.45002×10^6	7.43886×10^6	99.8501	1486.07	29	5.26745	0.146785	0.537647	70.8157	0.0491924	0.208764
Chirp mr 4 30m seed	1000	13715	1.47972×10^6	1.47824×10^6	99.9	981.727	13	3.2751	0.274019	0.375056	30.0898	0.093907	0.136428

Figura 5.10: Comparación de todos los entrenamientos realizados.

Capítulo 6

Conclusiones

Observando los resultados anteriores, no se destaco ninguna combinación de parámetros de las señales de entrada que permita realizar identificaciones exitosas de manera consistente. Encontrar una combinación confiable pudo haber sido logrado repitiendo señales (o combinaciones de señales) para varios entrenamientos y no solo hacer un entrenamiento con cada señal o, también, permitiendo entrenamientos de mayor duración.

A pesar de esto, es posible afirmar que el algoritmo NEAT es capaz de identificar un motor de corriente directa. La datos del comportamiento del motor recabados con ayuda del DAQ y el circuito de potencia fueron suficientes para encontrar modelos que logran identificar al sistema y su validación. El error de estos modelos es similar al del modelo clásico en varias ocasiones e, incluso, se logro obtener un modelo con un FPE mejor que el del modelo clásico.

Además, dado que el comportamiento del motor de corriente directa es similar al de un sistema lineal de primer orden, es muy probable que estos resultados sean reproducibles al identificar sistemas con un comportamiento parecido.

Con todo esto, más las propiedades de no linealidad de las redes neuronales y la complejificación del algoritmo NEAT, es posible inferir que esta metodología puede realizar identificaciones más generales, como de sistemas no lineales o de mayor orden. Para esto sera necesario más pruebas e investigación, como lo que se presenta en la siguiente sección.

6.1. Trabajo a futuro

Algunas de las cosas que se pueden realizar a continuación son las siguientes:

- Aumentar el tiempo de entrenamiento con mejores recursos computacionales.
- Variar los parámetros de NEAT para observar sus efectos en la identificación.
- Probar otros algoritmos de entrenamiento de redes neuronales.
- Investigar y probar otras funciones de aptitud para observar sus efectos.
- Obtener el comportamiento del motor a otras señales de control para comprobar las validaciones realizadas.
- Realizar la identificación de otros sistemas de mayor complejidad.

Bibliografía

[1] Ni-daqmx. https://www.ni.com/en-us/support/downloads/drivers/download.ni-daqmx.html#311818, 2019. [En linea; ultimo acceso 5-enero-2020].

[2] http://multineat.com/, 2020. [En linea; ultimo acceso 5-enero-2020].

[3] https://github.com/neat-python/neat-python, 2020. [En linea; ultimo acceso 5-enero-2020].

[4] Find the right version of neat for your needs. http://eplex.cs.ucf.edu/neat_software/, 2020. [En linea; ultimo acceso 5-enero-2020].

[5] Ametek. *Pittman Brush Commutated DC Servo Motors.* [Obtenido en linea:https://www.motiontech.com.au/wp-content/uploads/2016/07/MT-Pittman-Brush-Commutated-DC-Servo-Motors-200319.pdf].

[6] Filippo Maria Bianchi, Enrico Maiorino, Michael C. Kampffmeyer, Antonello Rizzi, and Robert Jenssen. *Recurrent neural networks for short-term load forecasting : an overview and comparative analysis.* SpringerBriefs in Computer Science. Springer, 2017.

[7] Tyler Biscontini. Genetic algorithm (ga). *Salem Press Encyclopedia of Health*, 2017.

[8] Gérard Dreyfus. *Neural networks : methodology and applications.* Springer, 2005.

[9] Marcos Ángel González Olvera and Yu Tang Xu. *Identificación de sistemas dinámicos mediante una red neurodifusa recurrente.* 2005.

[10] Colin Green. Phased searching with neat. https://sharpneat.sourceforge.io/phasedsearch.html, 2004. [En linea; ultimo acceso 5-enero-2020].

[11] Colin Green. Sharpneat neuroevolution framework. https://sharpneat.sourceforge.io/, 2020. [En linea; ultimo acceso 5-enero-2020].

[12] Andrzej Janczak. *Identification of nonlinear systems using neural networks and polynomial models : a block-oriented approach.* Lecture notes in control and information sciences: 310. Springer, 2005.

[13] K.J. Keesman. *System Identification: An Introduction.* Advanced Textbooks in Control and Signal Processing. Springer London, 2011.

[14] David C. Lay, Ana Elizabeth García Hernández, and David C. Lay. *Álgebra lineal y sus aplicaciones.* Pearson Educación, 2016.

[15] Wolfgang Maass, Prashant Joshi, and Eduardo D Sontag. Computational aspects of feedback in neural circuits. *PLOS Computational Biology*, 3(1):1–20, 01 2007.

[16] National Instruments. *NI 6255 DEVICE SPECIFICATIONS*, 6 2016.

[17] Katsuhiko Ogata. *System dynamics.* Pearson/Prentice Hall, 2004.

[18] R. Pintelon and J. Schoukens. *System Identification: A Frequency Domain Approach.* Wiley, 2012.

[19] S. N. Sivanandam and S. N. Deepa. *Introduction to genetic algorithms.* Springer, 2008.

[20] Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

[21] S. Weerasooriya and M. A. El-Sharkawi. Identification and control of a dc motor using back-propagation neural networks. *IEEE Transactions on Energy Conversion*, 6(4):663–669, Dec 1991.

[22] Wenle Zhang. System identification of linear systems with a linear neural network: — on relationship of lms learning and least squares estimation. *2018 Chinese Control And Decision Conference (CCDC), Chinese Control And Decision Conference (CCDC), 2018*, page 4615, 2018.