EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de La Laguna

**DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN**

# “Control descentralizado para manipuladores robóticos aplicando redes neuronales con entrenamiento UKF”

POR

**Ing. Juan Felipe Guerra Cano**

# T E S I S

**PRESENTADO COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA**

**DIRECTOR DE TESIS**
**Dr. Miguel Ángel Llama Leal**

**CODIRECTOR DE TESIS**
**Dr. Ramón García Hernández**

**ISSN: 0188-9060**


RIITEC: (19)-TMCIE- 2019
Torreón, Coahuila de Zaragoza. México
Noviembre 2019

"2019, Año del Caudillo del Sur, Emiliano Zapata"

Torreón, Coah., 08/Noviembre/2019
Dependencia: DEPI/CPCIE
Oficio: DEPI/CPCIE/015/2019
Asunto: Autorización de impresión de tesis.

**C. Ing. Juan Felipe Guerra Cano**
**CANDIDATO AL GRADO DE MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA.**
**PRESENTE**

Después de haber sometido a revisión su trabajo de tesis titulado:

**"Control descentralizado para manipuladores robóticos aplicando redes neuronales con entrenamiento UKF"**

Habiendo cumplido con todas las indicaciones que el jurado revisor de tesis hizo, se le comunica que se le concede la autorización con número de registro **RIITEC: (19)-TMCIE-2019**, para que proceda a la impresión del mismo.

**ATENTAMENTE**
*EDUCACIÓN TECNOLÓGICA FUENTE DE INNOVACIÓN*

**DR. JOSÉ IRVING HERNÁNDEZ JACQUEZ**
**Jefe de la División de Estudios de Posgrado e Investigación**
**del Instituto Tecnológico de la Laguna**

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLOGICO DE LA LAGUNA
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de La Laguna

"2019, Año del Caudillo del Sur, Emiliano Zapata"

Torreón, Coah., 21/Octubre/2019

**DR. JOSÉ IRVING HERNÁNDEZ JACQUEZ**
**JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN**

Por medio de la presente, hacemos de su conocimiento que después de haber sometido a revisión el trabajo de tesis titulado:

**"Control descentralizado para manipuladores robóticos aplicando redes neuronales con entrenamiento UKF"**

Desarrollado por el **C. Juan Felipe Guerra Cano**, con número de control **M1713037** y habiendo cumplido con todas las correcciones que se le indicaron, estamos de acuerdo que se le conceda la autorización de la fecha de examen de grado para que proceda a la impresión de la misma.

**ATENTAMENTE**
*EDUCACIÓN TECNOLÓGICA FUENTE DE INNOVACIÓN*

**Dr.Miguel Ángel Llama Leal**
**Asesor/Director de Tesis**

**Dr. Ramón García Hernández**
**Coasesor**

**Dr. Francisco Jurado Zamarripa**
**Comité Tutorial**

**Dr. Víctor Adrian Santibañez Dávila**
**Comité Tutorial**

*Dedicado a mis padres,*
*hermanos, amigos y profesores*
*que me apoyaron para*
*terminar este trabajo.*

*¡Muchas gracias!*

# Agradecimientos

A CONACYT y a la División de Estudios de Posgrado e Investigación del Instituto Tecnológico de la Laguna por darme la oportunidad de hacer los estudios de maestría.

A mi comité de tesis por sus enseñanzas y consejos, especialmente a mis asesores el Dr. Miguel Ángel Llama Leal y el Dr. Ramón García Hernández por su orientación y apoyo.

A todos mis compañeros y amigos con los que compartí muchos momentos durante mis estudios de maestría.

A mis amigos de generación, Andrés, Busquets, Edgar y Soto, por siempre apoyarnos unos a los otros, por compartir nuestras metas.

Finalmente agradezco enormemente a mis padres Margarita y Felipe, por darme la vida y todo el apoyo que me dan en todas mis decisiones.

# Resumen

La complejidad del modelado matemático de sistemas dinámicos, es un problema que existe debido al deseo inconmensurable de construir y controlar sistemas más grandes y sofisticados. Debido a esto nuevas teorías han surgido como alternativas al concepto ortodoxo de control de sistemas de alto orden, tales como parámetros distribuidos, redes neuronales y la teoría de control descentralizado. Los robóts manipuladores son ejemplos de esta clase de sistemas, debido a sus dinámicas no lineales complejas y la dificultad en la medición de sus parámetros. Aún si, solo se consideran los términos más importantes, el modelo matemático resultante requiere algoritmos de control con una gran cantidad de operaciones matemáticas, por este motivo es posible diseñar un control descentralizado, que consista en controladores independientes, considerando únicamente las variables locales para cada subsistema.

En este trabajo se presenta un controlador descentralizado utilizando redes neuronales recurrentes de alto orden (RHONNs) discretas para aproximar las dinámicas de los sistemas, por medio de un esquema de identificación y control neuronal. El entrenamiento de las RHONNs se realiza utilizando el filtro de Kalman *unscented* (UKF) y se compara con trabajos previos realizados con filtro de Kalman extendido (EKF). Los sistemas empleados son el manipulador robótico de accionamiento directo vertical de dos grados de libertad (g.d.l.) y el robot industrial Mitsubishi PA10-7CE de siete g.d.l., los cuales se encuentran en el Laboratorio de Mecatrónica y Control del Instituto Tecnológico de la Laguna.

# Abstract

The complexity of dynamic systems modeling is a problem that exists because of our intensive and limitless desire to build and control ever larger and more sophisticated systems. Due to this, new theories have arisen as alternatives to the orthodox concept of control of high order systems, such as distributed parameters, neural networks, and decentralized control theory. Robot manipulators are examples of this class of systems, due to their nonlinear complex dynamics, with strong interconnections, parameters difficult to measure and dynamics difficult to model. Considering only the most important terms, the mathematical model obtained requires control algorithms with a great number of mathematical operations, for which it is possible to design decentralized control, that consist of independent controllers, considering only local variables to each subsystem.

This work presents a decentralized controller using a discrete-time recurrent high order neural networks (RHONNs) to approximate the dynamics of systems, through an identification scheme and neural control strategy. The neural network learning is performed on-line by unscented Kalman filter (UKF) and it is compared with previous works done with extended Kalman filter (EKF). The systems used are the vertical direct-drive robotic manipulator with two degrees of freedom (DOF) and the industrial robot Mitsubishi PA10-7CE of seven DOF, which are located in the Mechatronics and Control Laboratory of the Instituto Tecnológico de la Laguna.

# Índice general

# Índice de figuras

# Índice de tablas

# Capítulo 1

# Introducción

## 1.1. Antecedentes

Definir *control de robots* puede resultar fácil en primera instancia; en términos generales consiste en hacer que un robot lleve a cabo una determinada tarea. Sin embargo, el cómo realizar dicha tarea, ha hecho que áreas interdisciplinarias como la ingeniería de control, la robótica y la mecatrónica interactúen entre sí para solucionar este problema.

Principalmente, un sistema robótico resultaría inútil si no se puede controlar en la forma en que se posiciona en su espacio de trabajo; además, es necesario considerar la interacción con el entorno; para ello la comunidad científica se ha dado a la tarea de abordar este problema tanto de manera teórica como práctica. Por ejemplo, la ingeniería de control con el fin de producir un comportamiento deseado en el robot se basa en el modelado matemático y en el análisis de su dinámica; la robótica por su parte estudia los robots, su diseño, manufactura y aplicación, mientras que la mecatrónica abarca la combinación sinérgica de las ingenierías mecánica, eléctrica, electrónica de control y de sistemas computacionales que se extiende y particulariza para todo tipo de robots; por ejemplo: robots manipuladores, robots móviles, robots paralelos, robots flexibles, robots de servicio, teleoperación o sistemas multirobot.

En el caso particular para manipuladores robóticos que se desplazan libremente dentro de su espacio de trabajo sin que tenga una interacción con el entorno, la posición y velocidad articular son las salidas del robot y el objetivo de control asociado al seguimiento de trayectorias; éste recibe el nombre de *control de movimiento*.

De acuerdo a datos del Consorcio Nacional de Recursos de Información Científica y Tecnológica (CONRICYT) en la ultima década, la cantidad de trabajos que abordan el tema del control de movimiento de manipuladores robóticos ha superado la cifra de 19,000 trabajos publicados, en donde se analizan diversos tipos de controladores cinemáticos y dinámicos; todo debido a la demanda tecnológica de la sociedad actual que ha requerido nuevos enfoques a los problemas de control surgidos por la necesidad de crear sistemas cada vez más complejos y sofisticados. Las redes neuronales (*Neural Networks* o NN por sus siglas en inglés), debido a su habilidad de aprendizaje y su capacidad de adaptarse, son una herramienta significativa en el modelado y control de este tipo de sistemas.

El uso de las redes neuronales en la robótica se aplica en la aproximación de la dinámica del sistema, donde es necesario trabajar con sistemas no lineales de alto orden con un escaso conocimiento de la planta y de su medio ambiente, y a partir de esta aproximación del modelo proponer una ley de control.

Los robots manipuladores son ejemplos de esta clase de sistemas debido a sus dinámicas no lineales complejas y la dificultad en la medición de sus parámetros. Aún si solo se consideran los términos más importantes, el modelo matemático resultante requiere algoritmos de control con una gran cantidad de operaciones matemáticas; por este motivo es posible diseñar un control descentralizado que consista en controladores independientes, considerando únicamente las variables locales para cada subsistema [Siljak, 2011].

Algunos autores [Nguyen and Widrow, 1990] definen una red neuronal como un sistema con entradas y salidas que está compuesta de muchos elementos de procesamiento similares. Cada elemento de procesamiento tiene un número de pesos sinápticos o simplemente pesos. Ajustando los pesos de un elemento se puede cambiar el comportamiento del mismo y, por lo tanto, puede también alterar el comportamiento total de la red para alcanzar la relación de entrada-salida deseada. Este último proceso es conocido como entrenamiento de la red.

El filtro de Kalman (KF) [Kalman, 1960] es una herramienta popular y efectiva como estimador de estados para sistemas lineales, el cual puede ser simplemente implementado en dos pasos: predicción y actualización. Las ecuaciones de predicción determinan las estimaciones a priori de los estados y la covarianza del error; en el siguiente paso, se actualizan las estimaciones para calcular la matriz de ganancias de Kalman, dando como resultado las estimaciones siguientes [Simon, 2006]. En el caso de sistemas no lineales se requieren aproximaciones para tratar la no linealidad de las ecuaciones de estado y/o salida del sistema. Existen bastantes variantes del KF para sistemas no lineales, el filtro de Kalman extendido (EKF) [Kalman and Bucy, 1961] y el filtro de Kalman *Unscented* (UKF) [Julier and Uhlmann, 1997] son los más utilizados. Ambos filtros mantienen la misma metodología que el filtro de Kalman original, pero utilizan diferentes enfoques para tratar la no linealidad. El EKF involucra matrices Jacobianas en un enfoque de linealización analítica mientras que el UKF utiliza un enfoque estadístico denominado transformación *Unscented* (UT) [Julier and Uhlmann, 1997].

En algunos trabajos y artículos, se presenta el análisis, el estudio y la comparación entre los filtros EKF y UKF; en [Rhudy et al., 2013], los autores presentan un análisis del desempeño de distintos algoritmos para la estimación de la posición de una aeronave usando el EKF y el UKF. La información detallada para la implementación de los algoritmos EKF y UKF se presenta en [Rhudy and Gu, 2013].

Los autores de [Jin, 1998] y [Liu, 1999] muestran la aplicación de un controlador descentralizado en manipuladores robóticos en donde cada articulación y su respectivo eslabón se considera un subsistema con el fin de desarrollar controladores locales utilizando únicamente las mediciones de posición y de velocidad angulares, que además, compensan los efectos de interconexión considerándolos como perturbaciones.

En [Safaric and Rodic, 2000], se reporta un esquema de control descentralizado para manipuladores robóticos, donde se utilizan redes neuronales “feedforward” para aproximar las dinámicas de

cada articulación de manera independiente. Una estrategia de control cinemático descentralizado para múltiples manipuladores redundantes se propone en [Li et al., 2012] para resolver el problema de ejecución de la tarea cooperativa por medio de redes neuronales recurrentes.

En [García-Hernández et al., 2009], los autores presentan un esquema de identificación y control neuronal descentralizado discreto aplicando redes neuronales recurrentes de alto orden (RHONN) entrenadas con un algoritmo EKF. La ley de control en tiempo discreto propuesta se basa en técnicas de modos deslizantes y control por bloques. Este algoritmo de control se implementa en tiempo real para un robot planar de 2 grados de libertad (g.d.l.) mostrado en la Figura 1.1.

En [García-Hernández et al., 2011] se propone una estrategia de control descentralizado similar, usando un esquema para la identificación y el control del brazo robótico Mitsubishi PA10-7CE de 7 g.d.l. mostrado en la Figura 1.2.

En [Flores-Ávila, 2014] se propuso un esquema de control descentralizado difuso adaptable para el brazo robótico Mitsubishi PA10-7CE de 7 g.d.l. mostrado en la Figura 1.2.

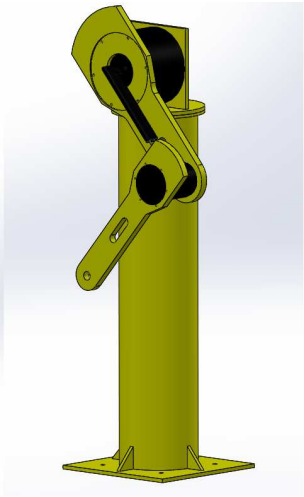

Figura 1.1: Robot planar de 2 g.d.l

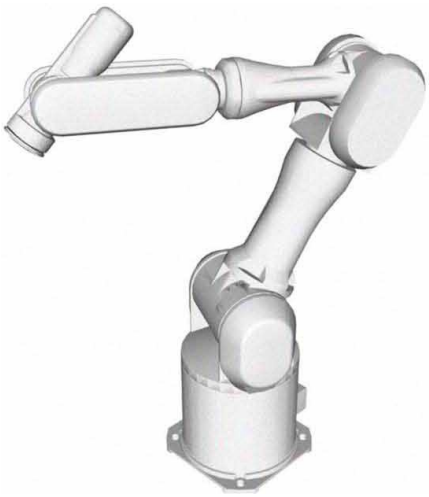

Figura 1.2: Brazo robótico Mitsubishi PA10-7CE

## 1.2. Objetivos de la tesis

### 1.2.1. Objetivo general

Desarrollar algoritmos de control basados en redes neuronales entrenadas por medio de filtros de Kalman no lineales, con el fin de analizar su desempeño frente a otros algoritmos existentes en la actualidad. Posteriormente se considera su implementación en tiempo real en un robot manipulador de los disponibles en el laboratorio de posgrado de Mecatrónica y Control del Instituto Tecnológico de La Laguna (ITL).

### 1.2.2. Objetivos específicos

- Investigar el estado del arte acerca de la aplicación de estimadores de estados para el entrenamiento de una red neuronal recurrente de alto orden (*Recurrent High Order Neural Network* o

RHONN por sus siglas en inglés).

- Diseñar algoritmos de control basados en redes neuronales recurrentes de alto orden con entrenamiento UKF utilizando *Matlab*/*Simulink*.
- Realizar simulaciones sujetas a diferentes condiciones iniciales con el fin de analizar su desempeño frente al entrenamiento con el filtro de Kalman Extendido de la literatura.
- Implementar la metodología propuesta en tiempo real en un robot manipulador de 2 g.d.l.

## 1.3. Motivación

La teoría de control no lineal y sus aplicaciones en la estimación de estados para modelos complejos se ha incrementado por el desarrollo tecnológico de la computación que mediante el cálculo numérico facilita su análisis e implementación. Un ejemplo de las aplicaciones está relacionado con los sistemas mecatrónicos que permiten aplicar el filtro de Kalman para la estimación de estados y en casos específicos el entrenamiento de redes neuronales. La motivación de este trabajo es desarrollar e implementar un algoritmo de aprendizaje basado en el filtro de Kalman *Unscented* para el entrenamiento de redes neuronales y aplicarlo posteriormente a manipuladores robóticos a nivel simulación y tiempo real, permitiendo así la comparación de esta metodología propuesta con otras reportadas en la literatura.

## 1.4. Estructura de la tesis

La estructura del presente trabajo de tesis se divide de la siguiente forma:

En el capítulo 2 se introducen los conceptos básicos de redes neuronales; además de la metodología de aplicación de los filtros de Kalman se incluyen también algunas definiciones de estabilidad. En el capítulo 3 se presenta el esquema de control descentralizado neuro-adaptable indirecto para la identificación y seguimiento de trayectorias. En el capítulo 4 se implementan los esquemas de identificación y control para el robot manipulador de dos grados de libertad y el brazo robótico Mitsubishi PA10-7CE, respectivamente. Además se muestran los resultados de los experimentos en tiempo real de los controladores descritos en el capítulo 3 para el caso del robot manipulador de 2 g.d.l.

Finalmente, en el capítulo 5 se presentan las conclusiones generales y trabajo futuro.

# Capítulo 2

# Fundamentos Teóricos

## 2.1. Definiciones de estabilidad

Considérese el sistema no lineal con múltiples entradas-múltiples salidas (MIMO por sus siglas en inglés)

$$x(k+1) = F(x(k), u(k)) \tag{2.1}$$

$$y(k) = h(x(k)) \tag{2.2}$$

donde $x(k) \in \mathbb{R}^n$ son los estados del sistema, $u(k) \in \mathbb{R}^m$ describe las entradas al sistema, $y(k) \in \mathbb{R}^p$ son las salidas del sistema; $F$ y $h$ son funciones no lineales.

Además, se considera un tipo especial de sistemas dinámicos no lineales en tiempo discreto donde el ruido en el proceso $Q(k)$ y el ruido en la medición $R(k)$ se suponen aditivos y son descritos por

$$\begin{aligned} x(k+1) &= F(x(k), u(k)) + Q(k) \\ y(k) &= H(x(k)) + R(k) \end{aligned} \tag{2.3}$$

**Definición 2.1** [Ge et al., 2004] Se dice que el sistema (2.1) es forzado o que tiene entrada(s). En contraste, el sistema descrito por una ecuación sin presencia explícita de una entrada $u(k)$, es decir

$$x(k+1) = F(x(k))$$

se dice que es no forzado. Puede obtenerse después de seleccionar la entrada $u(k)$ como una función de realimentación del estado

$$u(k) = \vartheta(x(k)) \tag{2.4}$$

que dicha sustitución elimina $u(k)$:

$$x(k+1) = F(x(k), \vartheta(x(k))) \tag{2.5}$$

lo cual produce un sistema no forzado (2.5).

**Definición 2.2** [Khalil, 2002] Las soluciones de (2.1) son

- *Uniformemente acotadas* si existe una constante positiva $c$, independiente de $k_0 \geq 0$, y para cada $a \in (0, c)$, se encuentra $\beta = \beta(a) > 0$, independiente de $k_0$, tal que

$$||x(k_0)|| \leq a \Rightarrow ||x(k)|| \leq \beta, \quad \forall k \geq k_0 \tag{2.6}$$

- *Uniformemente acotadas de manera global* si (2.6) se mantienen para una $a$ arbitrariamente grande.
- *Última y uniformemente acotadas* con $b$ como última cota si existen constantes positivas $b$ y $c$, independientes de $k_0 \geq 0$, y para cada $a \in (0, c)$ se encuentra $K = K(a, b) \geq 0$ independiente de $k_0$, tal que

$$||x(k_0)|| \leq a \Rightarrow ||x(k)|| \leq b, \quad \forall k \geq k_0 + K \tag{2.7}$$

- *Última y uniformemente acotadas de manera semiglobal* (**semiglobally uniformly ultimately bounded** o SGUUB por sus siglas en inglés) si (2.7) se mantienen para una $a$ arbitrariamente pequeña

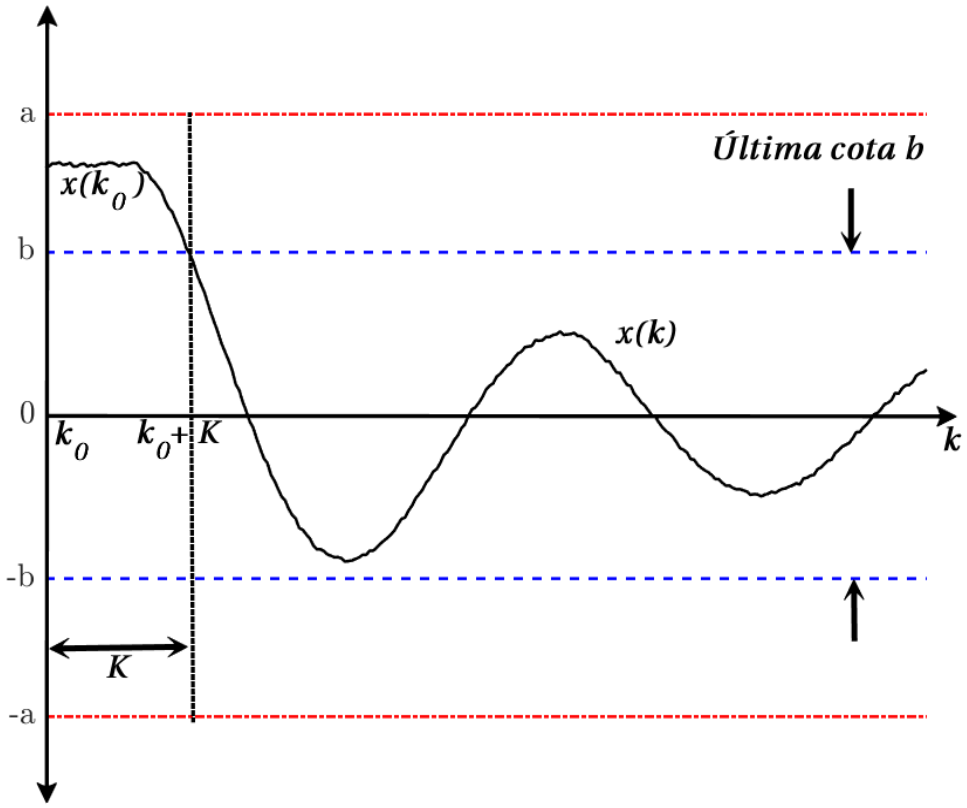


Figura 2.1: Representación de SGUUB

**Definición 2.3** [Ge et al., 2004] La solución para (2.1)-(2.4) es SGUUB, si para cualquier $\Omega \subset \mathbb{R}^n$ y toda $x(0) \in \Omega$, existe un $\epsilon > 0$ y un número $\mathbf{K}(\epsilon, x(k_0))$ tal que $||x(k)|| < \epsilon \quad \forall k \geq k_0 + \mathbf{K}$.

En otras palabras, la solución de (2.1) es SGUUB si para cualquier conjunto acotado $\Omega$ (arbitrariamente grande) y cualquier conjunto $\Omega_0$ (arbitrariamente pequeño) que contiene $(0,0)$ como un punto interior, existe una ley de control (2.4) tal que cada trayectoria del sistema en lazo cerrado que comenzando en $\Omega$ entre en el conjunto $\Omega_0 = \{x(k) \mid ||x(k)|| < \epsilon\}$ en un tiempo finito y permanezca dentro a partir de entonces [Zhang and Ioannou, 1996].

**Teorema 2.4** [Ge et al., 2004] Sea $V(x(k))$ una función de Lyapunov para el sistema en tiempo discreto (2.1), que satisface las siguientes propiedades:

$$\begin{aligned} \gamma_1(||x(k)||) \leq V(x(k)) &\leq \gamma_2(||x(k)||) \\ V(x(k+1)) - V(x(k)) &= \Delta V(x(k)) \\ &\leq -\gamma_3(||x(k)||) + \gamma_3(\zeta) \end{aligned}$$

donde $\zeta$ es una constante positiva, $\gamma_1(\cdot)$ y $\gamma_2(\cdot)$ son funciones estrictamente crecientes, y $\gamma_3(\cdot)$ es una función continua, no decreciente. Por lo que, si $\Delta V(x(k)) < 0$ para $||x(k)|| > \zeta$, entonces $x(k)$ es última y uniformemente acotada, i.e., existe un instante de tiempo $k_T$, tal que $||x(k)|| < \zeta, \ \forall k < k_T$.

**Definición 2.5** [Khalil, 2002] Un subconjunto $S \in \mathbb{R}^n$ se dice acotado si existe $r > 0$ tal que $||x|| \leq r$ para toda $x \in S$.

**Definición 2.6** [Khalil, 2002] Se dice que el sistema (2.8) es BIBO (*bounded input bounded output*, entrada acotada salida acotada) estable si para una entrada acotada $u(k)$, el sistema produce una salida acotada $y(k)$ para $0 < k < \infty$.

**Lema 2.7** [Rugh and Rugh, 1996] Considérese el sistema lineal discreto variante en el tiempo dado por

$$\begin{aligned} x(k+1) &= A(k)x(k) + B(k)u(k) \\ y(k) &= C(k)x(k) \end{aligned} \tag{2.8}$$

donde, $A(k)$, $B$ y $C$ son matrices de dimensiones apropiadas, $x(k) \in \mathbb{R}^n$, $y(k) \in \mathbb{R}^p$ y $u(k) \in \mathbb{R}^m$. Sea $\Phi(k_1, k_0)$ la matriz de transición de estado correspondiente a $A(k)$ para el sistema (2.8), $\Phi(k_1, k_0) = \prod_{k=k_0}^{k=k_1-1} A(k)$. Si $\Phi(k_1, k_0) < 1 \ \forall k_1 > k_0 > 0$ entonces el sistema (2.8) es

1. Exponencialmente estable de manera global para el sistema no forzado y
2. BIBO estable.

**Teorema 2.8** (Principio de separación) [Lin and Byrnes, 1994] El problema de estabilización asintótica del sistema (2.1)-(2.2) a través de la realimentación del estado estimado

$$\begin{aligned} u(k) &= \vartheta(\hat{x}(k)) \\ \hat{x}(k+1) &= F(\hat{x}(k), u(k), y(k)) \end{aligned} \tag{2.9}$$

tiene solución si y solo si el sistema (2.1)-(2.2) es asintóticamente estable y exponencialmente detectable.

**Corolario 2.9** [Lin and Byrnes, 1994] Existe un observador exponencial para el sistema en tiempo discreto no lineal estable de Lyapunov (2.1)-(2.2) con $u(k) = 0$ si y solo si la aproximación lineal

$$\begin{aligned} x(k+1) &= A(k)x(k) + B(k)u(k) \\ y(k) &= C(k)x(k) \\ A(k) = \frac{\partial F}{\partial x}|_{x=0},\ B(k) &= \frac{\partial F}{\partial u}|_{x=0},\ C(k) = \frac{\partial h}{\partial x}|_{x=0} \end{aligned} \tag{2.10}$$

del sistema (2.1)-(2.2) es detectable.

**Definición 2.9a** [Hespanha, 2018] El par $(A, C)$ del sistema (2.1)-(2.2) se dice que es *detectable*, si existe una matriz de realimentación de salida $L$ tal que el sistema

$$\dot{x} = [A(t) - L(t)C(t)]x$$

es asintóticamente estable.

**Teorema 2.10** [Karvonen et al., 2014] Considérese el sistema no lineal (2.3) y el algoritmo UKF descrito en el apéndice B. Además, teniendo en cuenta que las siguientes suposiciones son ciertas

(1) Existen números reales $f_{min}$, $h_{min} \neq 0$, y $f_{max}$, $h_{max} \neq 0$ de tal manera que se cumplan para cualquier $k \geq 0$

$$\begin{aligned} f_{min}^2 I \leq F(k)F(k)^T \leq f_{max}^2 I, \\ h_{min}^2 I \leq H(k)H(k)^T \leq h_{max}^2 I. \end{aligned}$$

(2) Existen números reales $p_{min}$, $p_{max}$, $r_{min}$ , $r_{max}$, $q_{min}$ , $q_{max} > 0$ tales que cumplen

$$\begin{aligned} p_{min} I \leq P(k) \leq p_{max} I, \\ q_{min} I \leq Q(k) \leq q_{max} I, \\ r_{min} I \leq R(k) \leq r_{max} I, \end{aligned}$$

entonces el error de estimación $\tilde{x}(k) = x(k) - \hat{x}(k)$ se encuentra acotado de manera exponencial según [Dimirovski et al., 2012].

## 2.2. Redes neuronales

Una de las metas de la ingeniería en control es implementar sistemas automáticos capaces de operar con un mayor grado de independencia de las acciones del ser humano, dentro de un ambiente

no estructurado y con incertidumbres [Gupta and Rao, 1994].

Un sistema de control inteligente tiene la habilidad de aprender adquiriendo información durante la operación de los comportamientos desconocidos de la planta y su entorno, de manera que su ejecución es mejorada. Con este concepto de aprendizaje es posible extender la región de operación y finalmente lograr la implementación de sistemas autónomos.

Una metodología de control que tiene el potencial de implementar este aprendizaje, son las redes neuronales artificiales (Artificial Neural Networks o ANNs por sus siglas en inglés). Ciertamente, la morfología neuronal del sistema nervioso es mucho más compleja. No obstante, una analogía simplificada fue desarrollada en la década de los sesentas, la cual ha sido utilizada en múltiples aplicaciones de ingeniería.

Las redes neuronales artificiales debido a su habilidad de aprendizaje y su capacidad de adaptarse, son una herramienta significativa en el modelado y control de sistemas autónomos.

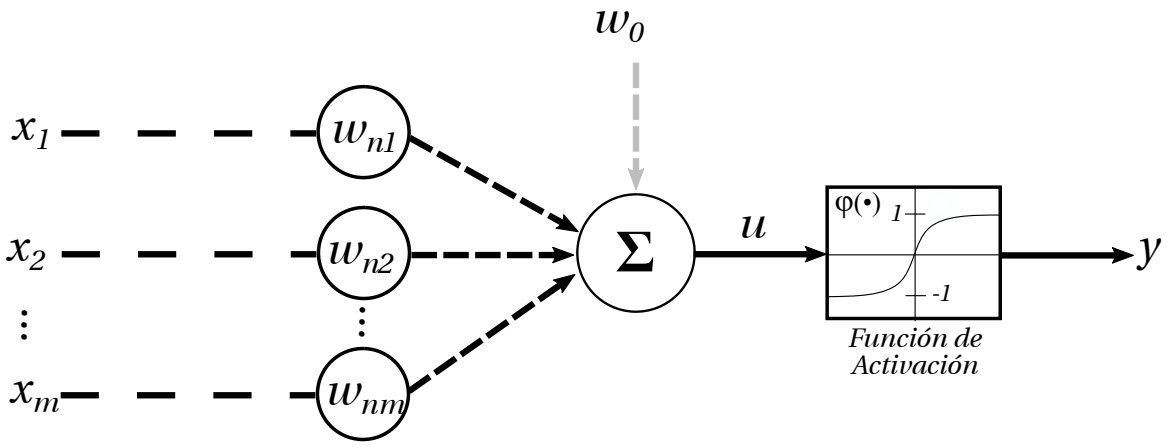


Figura 2.2: Modelo de una neurona artificial

Las ANNs son modelos simplificados de sus homónimas biológicas como se muestra en la Figura 2.2, ya que tratan de emular las capacidades del cerebro para resolver problemas complejos por medio del aprendizaje, que es el proceso por el cual los parámetros de la red se adaptan por una interacción continua con el medio ambiente [Sánchez and Alanís, 2006]. Este proceso implica la siguiente secuencia de eventos.

1. La estimulación es provocada por el medio ambiente.
2. La red ajusta sus parámetros.
3. Se genera una nueva respuesta

Se pueden identificar tres elementos básicos del modelo [Haykin, 2004]:

- Un conjunto de sinapsis o enlaces de conexión, cada uno de ellos parametrizado por un peso propio llamado peso sináptico, denotado por $w_{nj}$. Es importante aclarar que el primer subíndice corresponde a la neurona en cuestión (receptora), mientras que el segundo subíndice se refiere a la señal de entrada $x_j$ de la sinapsis $j$ (neurona emisora). A diferencia de la sinapsis en el cerebro los pesos sinápticos en neuronas artificiales pueden encontrarse dentro de un rango que incluye valores tanto negativos (señal inhibidora) como positivos (señal excitadora).

- Un sumador para las señales de entrada ponderadas por los respectivos pesos sinápticos de cada neurona.

- Una función de activación para limitar la amplitud de la salida de la neurona. A la función de activación también se le llama función de acotamiento que consiste en una transformación no lineal que limita el rango permisible de la señal de salida a un valor finito.

El modelo mostrado en la Figura 2.2 también incluye un umbral, denotado por $w_0$ , el cual polariza la entrada (disminuye o aumenta). En términos matemáticos, se puede describir el comportamiento de la neurona $n$ que se muestra en la Figura 2.2 de la siguiente manera

$$u = \sum_{j=1}^{m} w_{nj} x_j, \tag{2.11}$$

$$y = \varphi\left(u + w_0\right), \tag{2.12}$$

donde $x_1, \cdots, x_m$ son las señales de entrada; $w_{n1}, \cdots, w_{nm}$ son los pesos sinápticos correspondientes a la neurona $n$; $u$ es la combinación lineal de las entradas ponderadas por los pesos sinápticos; $w_0$ es el umbral; $\varphi(\cdot)$ es la función de activación; por último, $y$ es la salida de la neurona.

Existen múltiples funciones de activación $\varphi(\cdot)$ para definir la salida de la neurona en función del potencial de activación denotado por $v = (u + w_0)$. En [Haykin, 2004] se incluyen los siguientes tipos de funciones de activación las cuales toman valores en el intervalo cerrado $[0,\ 1]$.

1. *Función escalón*. Esta función de activación, presentada en la Figura 2.3, también se le conoce como función *Heaviside* o escalón unitario, la cual se describe por

$$\varphi(v) = \begin{cases} 1 & \text{para} \quad v \geq 0, \\ 0 & \text{para} \quad v < 0, \end{cases} \tag{2.13}$$

que produce la salida de la neurona $n$ como:

$$y = \begin{cases} 1 & \text{para} \quad v \geq 0, \\ 0 & \text{para} \quad v < 0. \end{cases} \tag{2.14}$$

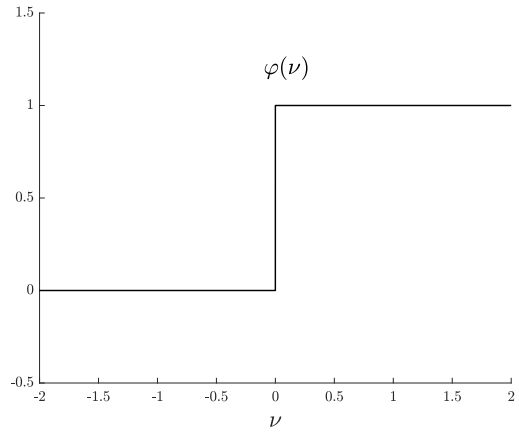


Figura 2.3: Función escalón

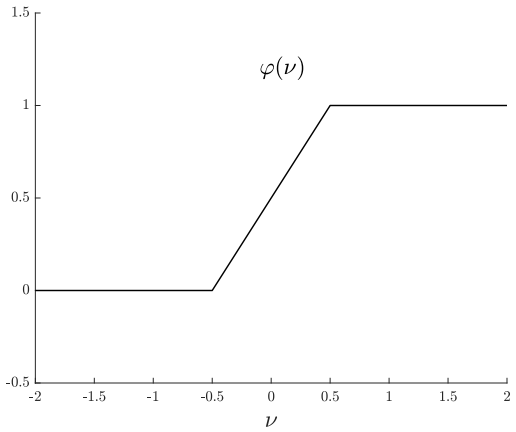


Figura 2.4: Función lineal a tramos

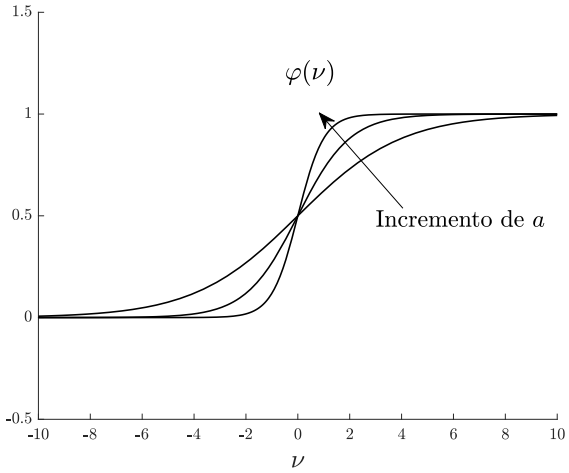


Figura 2.5: Función logística

En la literatura el utilizar la función escalón en el modelo neuronal mostrado en la Figura 2.2 como función de activación se le conoce como modelo *McCulloch-Pitts*. En este modelo la salida de la neurona $n$ toma el valor de uno si el campo local inducido de esa neurona no es negativo y cero en caso contrario; tal declaración describe la propiedad de *todo o nada* del Teorema 1 de [McCulloch and Pitts, 1943].

2. *Función lineal a tramos*. Para este tipo de función ilustrada en la Figura 2.4, se tiene

$$\varphi(v) = \begin{cases} 1 & \text{para} \quad v \geq a, \\ v + a & \text{para} \quad a > v > -a, \\ 0 & \text{para} \quad v \leq -a; \end{cases} \tag{2.15}$$

donde el factor de amplificación de la región de operación se supone unitario.

3. *Función sigmoidal*. Esta función cuya gráfica se presenta en la Figura 2.5 y tiene forma de "s", es sin lugar a dudas la forma más común de función de activación utilizada en el diseño de redes neuronales. Es una función estrictamente creciente que presenta un excelente balance entre comportamiento lineal y no lineal, lo que brinda un desempeño asitótico. Un ejemplo es la función logística definida:

$$\varphi(v) = \frac{1}{1 + e^{av}} \tag{2.16}$$

donde $a$ determina la pendiente de la función sigmoidal. Una característica de este tipo de función es que cuenta con un rango continuo de valores entre cero y uno; lo que permite obtener su derivada, a diferencia de la función escalón.

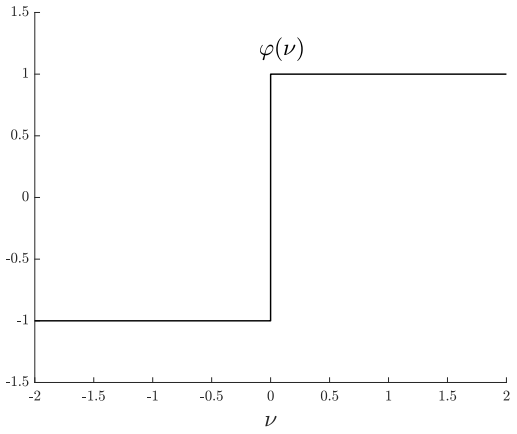


Figura 2.6: Función signo

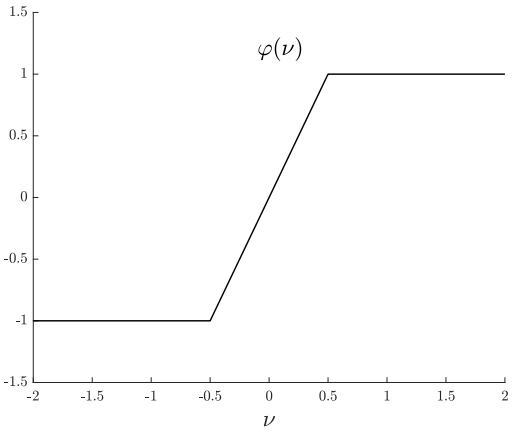


Figura 2.7: Función saturación

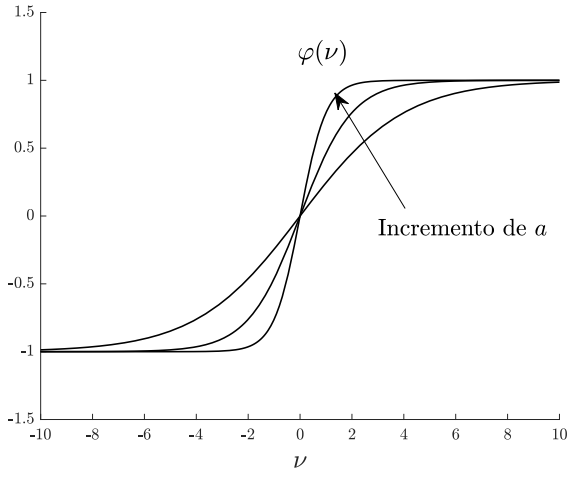


Figura 2.8: Función tangente hiperbólica

Actualmente, las funciones de activación pueden tomar valores en el intervalo cerrado $[-1,\ 1]$, las

cuales corresponden a la función *signo*, mostrada en la Figura 2.6, para el caso de la función escalón; la función saturación, mostrada en la Figura 2.7, en caso de la función lineal a tramos; y para la función sigmoidal se utiliza la función tangente hiperbólica, mostrada en la Figura 2.8, definida como:

$$\varphi(v) = tanh(av). \tag{2.17}$$

Desde el punto de vista de control, la habilidad de las ANNs para tratar con sistemas no lineales es un punto muy significativo. Respecto a este tema, métodos para el diseño de controladores no lineales que incluyen los basados en la teoría de control adaptable combinado con el uso de redes neuronales juegan un rol importante para tratar con sistemas de control no lineales variantes en el tiempo.

Para este trabajo se define a una red neuronal como un aproximador de funciones no lineales compuesta por elementos (neuronas) organizadas en una topología específica que utiliza una ley de aprendizaje para mejorar su desempeño.

### 2.2.1. Topología de redes neuronales

La manera en que las neuronas de una red neuronal son estructuradas esta estrechamente relacionado con el algoritmo de aprendizaje (entrenamiento) utilizado para entrenar la red. Dicho de otra forma, los algoritmos de entrenamiento son reglas utilizadas para el diseño de la arquitectura de las redes neuronales.

Las estructuras neuronales más utilizadas para desarrollar algoritmos de control más robustos son las redes *feedforward* y las redes recurrentes [García-Hernández et al., 2009]. El último tipo ofrece una herramienta más adecuada para modelar y controlar sistemas no lineales [Sanchez et al., 2008]. En la mayor parte de ellas, la metodología de diseño se basa en el enfoque de Lyapunov. Sin embargo, la mayoría de esos trabajos se desarrollan para sistemas de tiempo continuo. Por otro lado, mientras que está disponible la extensa literatura para sistemas de control lineal en tiempo discreto, las técnicas de diseño de control no lineal en tiempo discreto no se han discutido con el mismo grado. Para sistemas no lineales en tiempo discreto, el problema de control es más complejo debido a los acoplamientos entre subsistemas, entradas y salidas. Además, Las redes neuronales discretas son las más adecuadas para implementaciones en tiempo real [Sanchez et al., 2008]. En este trabajo de tesis se utilizaran redes neuronales recurrentes de alto orden discretas.

### 2.2.2. Redes neuronales recurrentes de alto orden discretas

Una red neuronal recurrente posee uno o más lazos de retroalimentación, ésta puede ser local, retroalimentación de una neurona a si misma, o global cuando la neurona realimenta a otras neuronas de la misma capa o de capas anteriores. Por lo cual existe una amplia gama de arquitecturas en el diseño de redes neuronales recurrentes, que las habilita a un mayor número de aplicaciones como son: modelado, control y representación en espacio de estados de sistemas no lineales.

Considérese la siguiente red neuronal recurrente de alto orden (RHONN), mostrada en la Figura 2.9

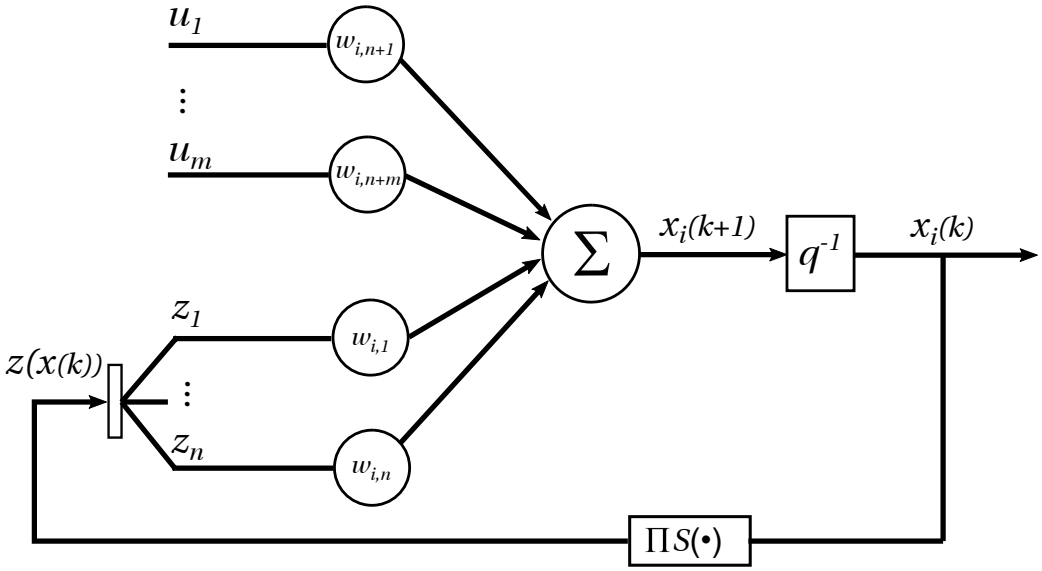


Figura 2.9: Esquema de una RHONN discreta

$$x_i(k+1) = w_i^T z_i\left(x(k), u(k)\right), \quad i = 1, ..., n \tag{2.18}$$

donde $x_i(k)$ es el estado de la $i$-ésima neurona en la iteración $k$ con $i = 1, ..., n$, siendo $n$ la dimensión del estado: $w_i$ es el vector de pesos adaptables en línea; $d_j(k)$ son enteros no negativos, y $z_i(x(k), u(k))$ está definida como:

$$z_I(x,u) = \begin{bmatrix} z_{I1} \\ z_{I2} \\ \vdots \\ z_{IL} \end{bmatrix} = \begin{bmatrix} \Pi_{j\in I_1} y_j^{d_j(1)} \\ \Pi_{j\in I_2} y_j^{d_j(2)} \\ \vdots \\ \Pi_{j\in I_L} y_j^{d_j(L)} \end{bmatrix} \tag{2.19}$$

donde $L$ es el número de conexiones de alto orden; $\{I_1, I_2, ..., I_L\}$ es una colección de $L$ subconjuntos no ordenados $\{1, 2, ..., m+n\}$; mientras que $y$ es un vector construido por las entradas a cada neurona, definido como:

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \\ y_{n+1} \\ \vdots \\ y_{n+m} \end{bmatrix} = \begin{bmatrix} S(x_1) \\ \vdots \\ S(x_n) \\ S(u_1) \\ \vdots \\ S(u_m) \end{bmatrix} \tag{2.20}$$

con $u = [u_1 \ u_2 \ \ldots \ u_m]^T$ como el vector de entradas a la red neuronal. La función $S(\bullet)$ es una sigmoide

suave, monótonamente creciente, de la forma

$$S(x) = \frac{\mu}{1 + e^{-\beta x}} + \epsilon \tag{2.21}$$

donde $\beta$ y $\mu$ son constantes positivas y $\epsilon$ es un número real positivo pequeño. Obviamente $S(x) \in [\epsilon, \epsilon + 1]$.

Considérese el problema de aproximar el sistema no lineal en tiempo discreto (2.1) por medio de la siguiente modificación a la RHONN discreta (2.18) [García-Hernández et al., 2017]:

$$x_i(k+1) = w_i^{*T} z_i\left(x(k), u(k)\right) + \epsilon_\psi \tag{2.22}$$

donde $w_i$ son las matrices de pesos ajustables, $\epsilon_\psi$ es un error de aproximación acotado, que se puede reducir aumentando el número de pesos ajustables. Suponiendo que existe un vector de pesos ideales $w_i^*$ tal que minimiza $||\epsilon_\psi||$ dentro de un conjunto compacto $\Omega \subset \mathbb{R}^n$. En general, se supone que este vector existe y es constante pero desconocido, lo que permite definir el error de estimación de los pesos como

$$\tilde{w}_i(k) = w_i(k) - w_i^* \tag{2.23}$$

Debido a que $w_i^*$ es constante, entonces $\tilde{w}_i(k+1) - \tilde{w}_i(k) = w_i(k+1) - w_i(k)$. Partiendo de (2.18) se pueden derivar tres posibles modelos de RHONN:

- Modelo Paralelo. En esta configuración, las conexiones de realimentación de la NN provienen de las salidas de la propia NN.
- Modelo *Feedforward* (HONN). En esta configuración, las conexiones de la NN provienen de las señales de entrada.
- Modelo Serie-Paralelo. En esta configuración, la realimentación de la NN proviene directamente de la planta real.

Las redes neuronales recurrentes de alto orden discretas presentan las siguientes características

- Permiten el modelado eficiente de sistemas dinámicos complejos
- Implementación sencilla
- Capacidad de ajustar sus parámetros en línea

## 2.3. Filtro de Kalman

El algoritmo de Kalman fue formulado en 1960 por Rudolph E. Kalman [Kalman, 1960], quien además, entre otras cosas, contribuyó a la teoría de control moderna (R. Kalman aportó los conceptos

de observabilidad y controlabilidad, así como el principio de dualidad). Su filtro significó un gran aporte al control moderno, siendo comparado al trabajo de Wiener realizado por el control clásico [Ogata, 2003].

De manera simple, el filtro de Kalman (KF por sus siglas en inglés) es un procedimiento matemático que por medio de un mecanismo de predicción y corrección, pronostica los nuevos estados a partir de su estimación previa, añadiendo un término proporcional (ganancia de Kalman) al error de predicción, de forma que este último se minimiza estadísticamente.

Dentro de la notación de ecuaciones en espacio de estados, el KF se describe como un sistema dinámico lineal que provee una solución recursiva de forma que cada actualización del estado estimado se calcula a partir de la estimación anterior y de la nueva entrada; de esta manera sólo se requiere almacenar la estimación anterior, sin la necesidad de almacenar todos los datos pasados.

En el entrenamiento de redes neuronales, los pesos sinápticos de la red neuronal son los estados que el KF estima, por medio de la medición de la salida de la red, prediciendo el comportamiento futuro de los estados. Es por ello que surge la alternativa de visualizar el entrenamiento de redes neuronales como un problema de filtrado óptimo [Haykin, 2004].

El KF ha sido modificado y extendido para su uso en sistemas no lineales, el Filtro de Kalman Extendido (EKF) y el Filtro de Kalman *Unscented* (UKF) [Julier and Uhlmann, 1997]. El EKF utiliza matrices Jacobianas para realizar la aproximación lineal, mientras que el UKF realiza una aproximación estadística, llamada transformación *Unscented* (UT). En el entrenamiento de redes neuronales los pesos sinápticos de la red neuronal son los estados a estimar y la salida de la red es la medición usada por el filtro de Kalman.

### 2.3.1. Filtro de Kalman lineal

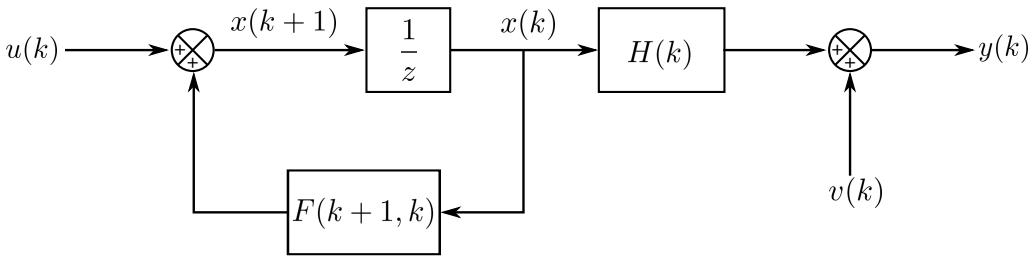


Figura 2.10: Sistema dinámico lineal en tiempo discreto

La aplicación del filtro de Kalman ofrece una solución óptima para sistemas lineales con un criterio de error cuadrático y perturbaciones por ruido blanco Gaussiano aditivo (*Additive white Gaussian noise* o AWGN por sus siglas en inglés). Además, debido a que la solución es recursiva, sólo requiere

almacenar el estado estimado anterior, lo que le proporciona una gran eficiencia computacional.

Se considera el sistema lineal en tiempo discreto mostrado en la Figura 2.10 y que puede ser descrito por el siguiente par de ecuaciones

**Ecuación del proceso**

$$x(k+1) = F(k+1,k)x(k) + u(k) \tag{2.24}$$

donde $k$ denota el tiempo de muestreo, $x(k) \in \mathbb{R}^n$ representa los estados del sistema, $F(k+1,k) \in \mathbb{R}^{n\times n}$ es la matriz de transición de estados desde la iteración $k$ a $k+1$, y $u(k) \in \mathbb{R}^n$ es el vector de perturbación del proceso (AWGN).

**Ecuación de salida**

$$y(k) = H(k)u(k) + v(k) \tag{2.25}$$

donde $y(k) \in \mathbb{R}^m$ es el vector de salida (medible en $k$); $H(k) \in \mathbb{R}^{m\times n}$ es la matriz de medición y $v(k) \in \mathbb{R}^n$ representa el ruido en la medición (AWGN).

El vector de estados $x(k)$ generalmente desconocido, es definido como el conjunto mínimo de datos para describir el comportamiento del sistema. Por tanto, $x(k)$ descrito por (2.24) y (2.25) se puede estimar con el filtro de Kalman, el cual se aplica por medio de las siguientes ecuaciones:

1. Inicialización para $k = 0$

$$\begin{aligned} \hat{x}(0) &= E\{x(0)\} \\ P(0) &= E\left\{(x(0) - E\{x(0)\},)\,(x(0) - E\{x(0)\})^T\right\}, \end{aligned} \tag{2.26}$$

2. Propagación del estado estimado

$$\hat{x}^-(k) = F(k+1,k)\hat{x}^-(k-1) \tag{2.27}$$

3. Propagación de la covarianza del error

$$P^-(k) = F(k+1,k)P(k-1)F^T(k,k-1) + Q(k-1) \tag{2.28}$$

4. Matriz de ganancia de Kalman

$$K(k) = P^-(k)H^T(k)\left[H(k)P^-(k)H^T(k) + R(k)^T\right]^{-1} \tag{2.29}$$

5. Actualización del estado estimado

$$\hat{x}(k) = \hat{x}^-(k) + K(k)\left(y(k) - H(k)\hat{x}^-(k)\right) \tag{2.30}$$

6. Actualización de la covarianza del error

$$P(k) = (I - K(k)H(k))\,P^{-}(k) \tag{2.31}$$

donde la matriz de covarianza del ruido del proceso $Q(k)$ se define como:

$$E\left\{u(n)u^{T}(k)\right\} = \begin{cases} Q(k) & \text{para} \quad n = k, \\ 0 & \text{para} \quad n \neq k, \end{cases} \tag{2.32}$$

y la matriz de covarianza del ruido de la medición $R(k)$ se define como:

$$E\left\{v(n)v^{T}(k)\right\} = \begin{cases} R(k) & \text{para} \quad n = k, \\ 0 & \text{para} \quad n \neq k, \end{cases} \tag{2.33}$$

siendo $E(\cdot)$ la esperanza matemática [Walpole et al., 1999]. Además, es importante señalar que no existe una relación entre el ruido de medición y el ruido del proceso.

### 2.3.2. Filtro de Kalman extendido EKF

Normalmente, la mayoría de los modelos de sistemas reales son no lineales, por este motivo se extendió el uso del filtro de Kalman a través de un procedimiento de linealización. Este filtro resultante es conocido como Filtro de Kalman Extendido (EKF) descrito en [Kalman and Bucy, 1961] para sistemas dinámicos no lineales discretos cuyas ecuaciones en espacio de estados son:

$$x(k+1) = f(k,x) + u(k), \tag{2.34}$$

$$y(k) = h(k,x) + v(k), \tag{2.35}$$

La idea principal del EKF es linealizar el modelo en espacio de estados de (2.34) y (2.35) a cada instante de tiempo alrededor del estado estimado más reciente, y una vez obtenido el modelo lineal se aplica el algoritmo de Kalman [Alanis et al., 2007]. De manera explícita, la aproximación se realiza en dos pasos:

**Paso 1.** Se calculan las matrices:

$$F_{k+1,k} = \left.\frac{\partial f(k,x(k))}{\partial x}\right|_{x=\hat{x}(k)}, \tag{2.36}$$

$$H_{k} = \left.\frac{\partial h(k,x(k))}{\partial x}\right|_{x=\hat{x}^{-}(k)} \tag{2.37}$$

**Paso 2.** Con $F(\bullet)$ y $H(\bullet)$ evaluadas, se emplean las series de Taylor en una aproximación de primer orden para las funciones $f(k,x)$ y $h(k,x)$ como sigue:

$$f(k,x) \approx F(x,\hat{x}(k)) + F_{k+1,k}(x,\hat{x}(k)), \tag{2.38}$$

$$h(k,x) \approx H(x,\hat{x}^{-}(k)) + H_{k+1,k}(x,\hat{x}^{-}(k)), \tag{2.39}$$

ahora con estas expresiones, se aproxima el sistema dado por (2.34) y (2.35) como se muestra:

$$x(k+1) \approx F_{k+1,k}x(k) + u(k) + d(k), \tag{2.40}$$

$$\bar{y}(k) \approx H_k x(k) + v(k), \tag{2.41}$$

donde se tienen dos nuevas variables descritas por:

$$\bar{y}(k) = y(k) - \{h(x, \hat{x}^-(k)) - H_k(\hat{x}^-(k)\}, \tag{2.42}$$

$$d(k) = f(x, \hat{x}^-(k)) - F_{k+1,k}(\hat{x}(k)). \tag{2.43}$$

Dadas las ecuaciones en espacio de estados linealizadas, se procede a realizar el mismo procedimiento que en el caso del filtro de Kalman [Haykin, 2004].

A pesar de que el EKF es una de las herramientas de filtrado más populares para la estimación de sistemas no lineales, cuenta con dos inconvenientes que dificultan su aplicación. En primer lugar, al realizar la linealización no se toma en cuenta las incertidumbres ni el ruido en las variables del sistema. En segundo lugar, la precisión de la propagación está limitada por el empleo de series de Taylor truncadas de primer orden para la linealización [Castrejon-Lozano, 2008].

### 2.3.3. Filtro de Kalman *Unscented* UKF

Con el propósito de superar las limitaciones previas, el filtro de Kalman *Unscented* fue desarrollado usando una aproximación de muestreo determinístico para propagar la información a través de una transformación no lineal [Julier and Uhlmann, 1997]. Adicionalmente, este tipo de muestreo cuenta con una precisión por arriba del tercer orden y no se necesitan calcular matrices Jacobianas de manera explícita, por lo que es más sencillo de implementar.

El UKF aditivo, es el miembro más popular de una familia de filtros de Kalman, conocida como de puntos sigma (*Sigma Point Kalman Filters*) [Van Der Merwe et al., 2004]. Al igual que el EKF, se utiliza para estimación en sistemas dinámicos no lineales. Estas variantes que se han desarrollado para el UKF tienen el propósito de mejorar su desempeño en diversas direcciones. El UKF aditivo fue propuesto con el fin de disminuir el número de cálculos realizados en cada iteración al no utilizar los estados aumentados que emplea el UKF tradicional. Esto reduce significativamente las operaciones empleadas en el cálculo y propagación de los *puntos sigma*, lo que lo hace más apropiado para ejecutarse en sistemas complejos por su bajo volumen de procesamiento. Por este motivo, en este trabajo se utilizará el UKF aditivo para comparar su desempeño con el EKF.

#### Transformación *Unscented*

La transformación *unscented* (Unscented transform o UT por sus siglas en inglés) es un método para calcular las estadísticas de una variable aleatoria que sufre una transformación no lineal, basándose en el principio de que es más fácil aproximar una distribución de probabilidad que una función

no lineal aleatoria [Julier et al., 1996]. La razón por la cual se le llama transformación *unscented* es incierta [Van Der Merwe et al., 2004]. El filtro fue desarrollado por Julier en el verano de 1994, en el *Robotics Research Group* (RRG) de Oxford, Reino Unido; y fue publicado en 1995 simplemente con el nombre de Nuevo Filtro [Julier et al., 1995]. Con el propósito de brindarle un nombre específico, se hicieron distintas propuestas y por votación se eligió el término *unscented* [Castrejon-Lozano, 2008].

Principalmente, la aplicación del algoritmo UKF se basa en que es más sencillo y preciso aproximar la naturaleza del ruido implicado en el sistema a través de una transformación no lineal por medio de un propagación de una serie de muestras ponderadas que propagar la misma función no lineal. A este planteamiento se le llama aproximación de *puntos sigma*, la cual se describe como sigue Considérese $x \in \mathbb{R}^n$ una variable aleatoria que se propaga a través de $y = g(x)$, una función no lineal arbitraria. Suponga que $x$ tiene media $\bar{x}$ y covarianza $P_x$. Para calcular la media y covarianza de $y$ se forma un conjunto de $2n+1$ *puntos sigma* $\{\chi_i : \ i = 0,...,2n\}$, donde $\chi_i \in \mathbb{R}^n$. Los *puntos sigma* se calculan de la siguiente manera:

$$\begin{aligned}
\chi_0 &= \bar{x} \\
\chi_i &= \bar{x} + \zeta(\sqrt{P_x})_i \quad i = 1,...,n \\
\chi_i &= \bar{x} - \zeta(\sqrt{P_x})_i \quad i = n+1,...,2n
\end{aligned}$$

donde $\zeta$ es un factor de escala que determina la dispersión de los *puntos sigma* alrededor de $\bar{x}$ y $\sqrt{P_x}$ indica la $i$-ésima columna de la raíz cuadrada de la matriz de covarianza $P_x$. Usualmente, debido a su eficiencia, se utiliza la factorización de *Cholesky* para calcular la raíz cuadrada de la matriz $P_x$ [Rhudy et al., 2011]. Como la raíz cuadrada de una matriz definida positiva no es única, cualquier rotación ortonormal del conjunto de puntos sigma es válido. Una vez que los puntos sigma se han calculado, es necesario propagarlos a través de la función no lineal

$$\mathcal{Y} = g(\chi_i) \ \ i = 0,...,2n.$$

La media y la covarianza de $y$ se aproximan utilizando una media y covarianza de muestras ponderadas de los *puntos sigma* transformados:

$$\begin{aligned}
\bar{y} &\approx \sum_{i=0}^{2n} w_i^m \mathcal{Y}_i \\
P_y &\approx \sum_{i=0}^{2n} \sum_{j=0}^{2n} w_{ij}^c \mathcal{Y}_i \mathcal{Y}_j^T \\
P_{xy} &\approx \sum_{i=0}^{2n} \sum_{j=0}^{2n} w_{ij}^c \chi_i \mathcal{Y}_j^T
\end{aligned}$$

donde $w_i^m$ y $w_{ij}^c$ son pesos escalares, cuyos superíndices $m$ y $c$ indican si son pesos de la media o de la covarianza, respectivamente. Los valores específicos de los pesos $w$ y factores de escala $\zeta$ dependen del tipo de aproximación de puntos sigma que se utilicen.

En el Apéndice B se muestran los pasos para la implementación de los filtros de Kalman no lineales haciendo una comparación entre el EKF y el UKF [Rhudy and Gu, 2013]. Resumiendo, ambos filtros son discretos, recursivos y cuentan con una complejidad de cálculo computacional similar, lo que los vuelve herramientas factibles en el problema de estimación de estados de sistemas no lineales. La Figura 2.11 muestra la comparación entre la propagación de la media y la covarianza entre el KF, EKF y UKF

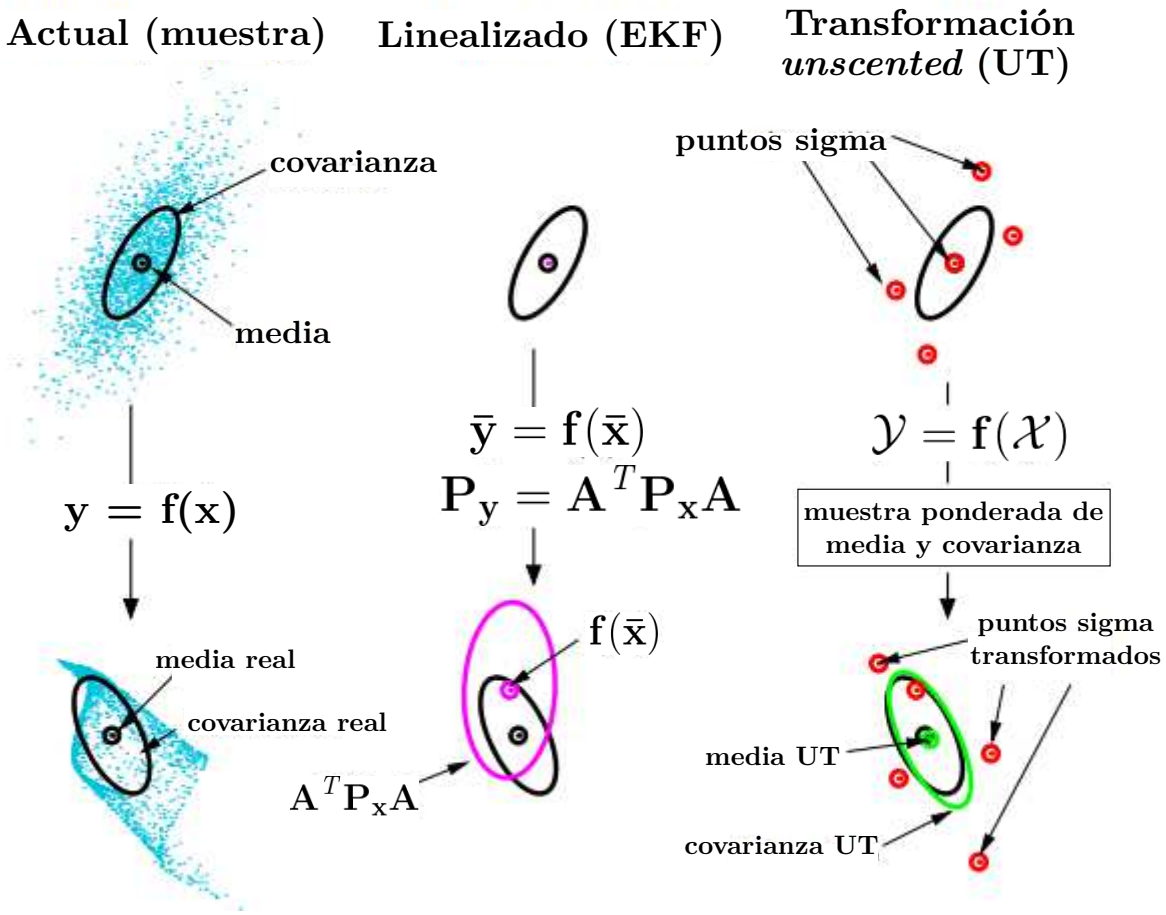


Figura 2.11: Ejemplo de la transformación *unscented*

Algunas diferencias son que el desempeño del EKF depende de dos factores: En primer lugar, las aproximaciones de "primer orden" a los términos óptimos que son afectadas directamente por la no linealidad del sistema. En segundo lugar se requiere del cálculo correcto de las matrices Jacobianas que en la mayoría de aplicaciones es no trivial. Estas discrepancias pueden generar un pobre desempeño o incluso divergencia. Por otro lado, el UKF no requiere del cálculo de matrices Jacobianas y genera estimaciones de mayor orden que el EKF, que se deben al número de operaciones realizadas por iteración $\mathcal{O}(L)^3$, donde $L$ es la dimensión del vector de estado aumentado; mientras que el EKF utiliza $\mathcal{O}(N)^3$ operaciones por iteración, siendo $N$ la dimensión del vector de estado. En el caso par-

ticular del UKF aditivo (ruidos de proceso y medida aditivos), los filtros presentan el mismo orden de operaciones necesarias por iteración [Pascual, 2004].

Además, para este trabajo es importante hacer los siguientes comentarios

**Comentario 2.11** Los algoritmos EKF y UKF son utilizados únicamente en el entrenamiento de las redes neuronales cuyos pesos representan los estados a ser estimados por los filtros EKF y UKF.

**Comentario 2.12** Una de las propiedades más reconocidas de las NN es que el vector del error de aproximación, denotado como $\epsilon_{z_i}$ se encuentra acotado.

# Capítulo 3

# Identificación y control neuronal

El desarrollo y avance tecnológico de la computación ha impulsado a la simulación y ejecución en tiempo real de modelos dinámicos no lineales cada vez más complejos, que mediante el cálculo numérico facilita el análisis e implementación de la teoría de control no lineal y sus aplicaciones en la estimación de estados.

En este trabajo se pretende desarrollar e implementar un algoritmo de control basado en UKF para aplicarlo a manipuladores robóticos, cuyo esquema es representado en la Figura 3.1.

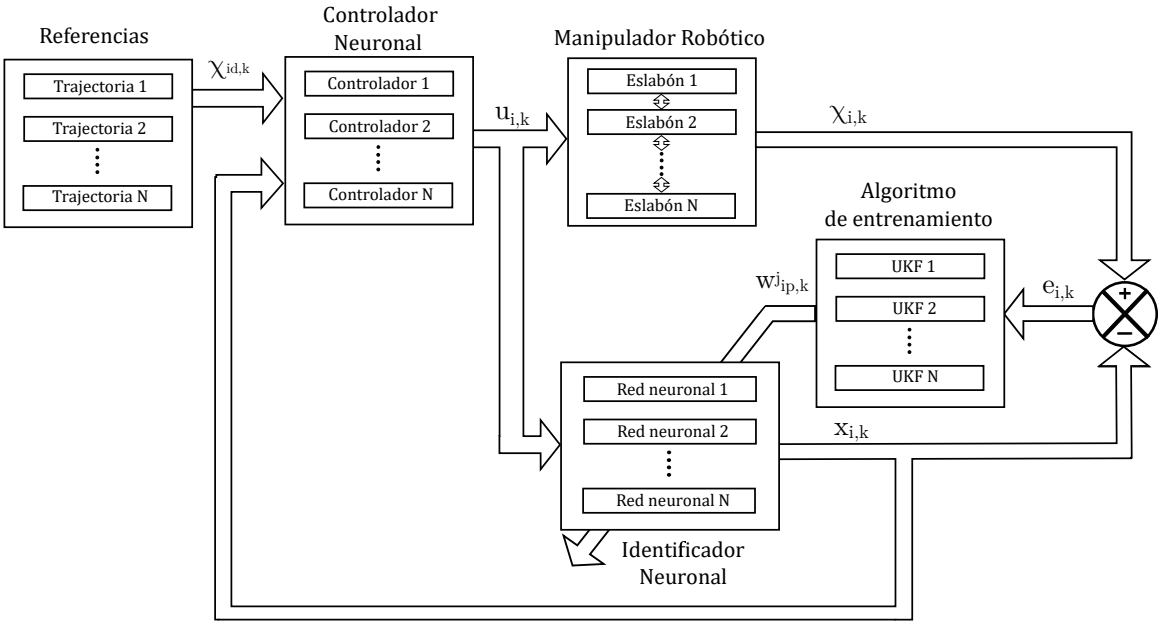


Figura 3.1: Esquema de identificación y control neuronal con entrenamiento UKF

## 3.1. Sistema descentralizado discreto

Considérese la siguiente clase de sistema, no lineal y discreto, el cual se representa en la forma no lineal controlable por bloques (NBCF por sus siglas en inglés *Nonlinear Block Controllable Form*)

conformada por $r$ bloques.

$$\begin{aligned}
\chi_{i,k+1}^{1} &= f_i^1\left(\chi_i^1\right) + B_i^1\left(\chi_i^1\right)\chi_i^2 + \Gamma_{i\ell}^1 \\
\chi_{i,k+1}^{2} &= f_i^2\left(\chi_i^1, \chi_i^2\right) + B_i^2\left(\chi_i^1, \chi_i^2\right)\chi_i^3 + \Gamma_{i\ell}^2 \\
&\vdots \\
\chi_{i,k+1}^{j} &= f_i^j\left(\chi_i^1, \cdots, \chi_i^j\right) + B_i^j\left(\chi_i^1, \cdots, \chi_i^j\right)\chi_i^{j+1} + \Gamma_{i\ell}^j \\
&\vdots \\
\chi_{i,k+1}^{r} &= f_i^r\left(\chi_i\right) + B_i^r\left(\chi_i\right)u_i + \Gamma_{i\ell}^r
\end{aligned} \tag{3.1}$$

donde $\chi_i \in \mathbb{R}^{n_i}$, $\chi_i^j \in \mathbb{R}^{n_{ij}\times 1}$, $i = 1,\ldots,N$, $j = 1,\ldots,r-1$, $l = 1,\ldots,n_{ij}$. $N$ es el número de subsistemas y $u_i \in \mathbb{R}^{m_i}$ es el vector de entradas. Además se supone que $f_i^j$, $B_i^j$ y $\Gamma_i^j$ son funciones *suaves* y acotadas, además $f_i^j(0) = 0$ y $B_i^j(0) = 0$, mientras que $n_{i1} \le n_{i2} \le \cdots \le n_{ij} \le m_i$ define las diferentes estructuras de los subsistemas. Los términos de interconexión están dados por

$$\begin{aligned}
\Gamma_{i\ell}^1 &= \sum_{\ell=1,\ell\neq i}^{N} \gamma_{i\ell}^1\left(\chi_\ell^1\right) \\
&\vdots \\
\Gamma_{i\ell}^j &= \sum_{\ell=1,\ell\neq i}^{N} \gamma_{i\ell}^j\left(\chi_\ell^1, \cdots, \chi_\ell^j\right) \\
&\vdots \\
\Gamma_{i\ell}^r &= \sum_{\ell=1,\ell\neq i}^{N} \gamma_{i\ell}^r\left(\chi_\ell\right)
\end{aligned} \tag{3.2}$$

donde $\chi_\ell = \left[\chi_\ell^1, \chi_\ell^2, \cdots, \chi_\ell^j\right]^T$ representa el vector de estado del $\ell$-ésimo subsistema con $1 \le \ell \le N$ y $\ell \neq i$.

## 3.2. Identificador neuronal descentralizado

La siguiente estructura de red neuronal recurrente de alto orden (RHONN) descentralizada discreta se propone para identificar al sistema representado por (3.1):

$$\begin{aligned}
x_{i,k+1}^{1} &= w_{i,k}^1 S\left(\chi_{i,k}^1\right) + w_i'^1 \chi_{i,k}^2 \\
x_{i,k+1}^{2} &= w_{i,k}^2 S\left(\chi_{i,k}^1, \chi_{i,k}^2\right) + w_i'^2 \chi_{i,k}^3 \\
&\vdots \\
x_{i,k+1}^{j} &= w_{i,k}^j S\left(\chi_{i,k}^1, \chi_{i,k}^2, \cdots, \chi_{i,k}^j\right) + w_i'^j \chi_i^{j+1}(k) \\
&\vdots \\
x_{i,k+1}^{r} &= w_{i,k}^r S\left(\chi_{i,k}^1, \cdots, \chi_{i,k}^r\right) + w_i'^r u_{i,k}
\end{aligned} \tag{3.3}$$

donde $x^j_{i,k+1} = \begin{bmatrix} x^1_i & x^2_i & \cdots & x^r_i \end{bmatrix}^T$ es el $j$-ésimo estado de la red neuronal, con $i = 1,\ldots,N$ y $j = 1,\ldots,r-1$; $w^j_{i,k}$ son parámetros constantes, $rank(w^{'j}_i) = n_{ij}$, $S(\bullet)$ es la función de activación y $u_{i,k}$ representa el vector de entradas.

Es importante señalar que la ecuación (3.3) representa un identificador serie-paralelo y no considera explícitamente los términos de interconexión (3.2) cuyos efectos son compensados por la actualización de pesos neuronales de la red [Zhang and Ioannou, 1996, García-Hernández et al., 2017].

**Proposición 3.1** El seguimiento de trayectoria deseada $x^j_{i\mathrm{d}}$ definida en términos del estado $\chi^j_i$ formulado en la ecuación (3.1) puede establecerse como la siguiente desigualdad [Felix et al., 2005]

$$||x^j_{i\mathrm{d}} - \chi^j_i|| \leq ||x^j_i - \chi^j_i|| + ||x^j_{i\mathrm{d}} - x^j_i|| \tag{3.4}$$

donde $||\cdot||$ representa la norma Euclidiana, $i = 1,\ldots,N$, $j = 1,\ldots,r$; $x^j_{i\mathrm{d}} - \chi^j_i$ es el error de seguimiento del sistema, $x^j_i - \chi^j_i$ es el error de identificación y $x^j_{i\mathrm{d}} - x^j_i$ es el error de seguimiento de la RHONN.

Antes de continuar, es necesario establecer los siguientes requisitos para el seguimiento y control de la red neuronal.

**Requerimiento 3.2**

$$\lim_{t\to\infty} ||x^j_i - \chi^j_i|| \leq \zeta^j_i \tag{3.5}$$

con $\zeta^j_i$ como una constante pequeña positiva

**Requerimiento 3.3**

$$\lim_{t\to\infty} ||x^j_{i\mathrm{d}} - x^j_i|| = 0 \tag{3.6}$$

Un identificador descentralizado neuronal basado en la ecuación (3.3) asegura que (3.5) se cumpla, mientras que (3.6) se pretende obtener por un controlador en tiempo discreto diseñado con control por bloques y técnicas de modos deslizantes.

Es posible establecer la Proposición 3.1 debido al principio de separación (mostrado en el Teorema 4.3 de [Lin and Byrnes, 1994]) para sistemas no lineales en tiempo discreto.

## 3.3. Ley de aprendizaje en línea

Se propone un algoritmo de entrenamiento basado en UKF descrito para cada $i$-ésima neurona por [Julier and Uhlmann, 1997, Wan and Van Der Merwe, 2000, Haykin, 2004, Rhudy and Gu, 2013].

$$\begin{aligned} K^j_{i,k} &= P^{jxy}_{i,k}(P^{jyy}_{i,k})^{-1} \\ w^j_{i,k+1} &= w^j_{i,k} + K^j_{i,k}e^j_{i,k} \\ P^j_{i,k+1} &= P^j_{i,k} - K^j_{i,k}P^{jyy}_{i,k}K^{jT}_{i,k} \end{aligned} \tag{3.7}$$

con

$$P_{i,k}^{jxy} = \sum_{i=0}^{2L} \eta_i^c [\mathcal{X}_{i,k|k-1}^j - \hat{x}_{i,k}^{j-}][\mathcal{Y}_{i,k|k-1}^j - \hat{y}_{i,k}^{j-}]^T \tag{3.8a}$$

$$P_{i,k}^{jyy} = R_{i,k}^j + \sum_{i=0}^{2L} \eta_i^{jc} [\mathcal{Y}_{i,k|k-1}^j - \hat{y}_{i,k}^{j-}][\mathcal{Y}_{i,k|k-1}^j - \hat{y}_{i,k}^{j-}]^T \tag{3.8b}$$

$$P_{i,k}^j = Q_{i,k}^j + \sum_{i=0}^{2L} \eta_i^{jc} \left[\mathcal{X}_{i,k|k-1}^j - \hat{x}_{i,k}^{j-}\right] \left[\mathcal{X}_{i,k|k-1}^j - \hat{x}_{i,k}^{j-}\right]^T \tag{3.8c}$$

$$e_{i,k}^j = \left[\chi_{i,k}^j - x_{i,k}^j\right] \tag{3.8d}$$

donde $e_{i,k}^j$ es el error de identificación, $P_{i,k}^j$ es la matriz de covarianza del error de predicción, $P_{i,k}^{jyy}$ es la matriz de covarianza de predicción de la salida, $P_{i,k}^{jxy}$ es la matriz de covarianza cruzada entre el estado y la salida; $w_{i,k}^j$ es el $j$-ésimo peso (estado) del $i$-ésimo subsistema, $\eta_i^{jc}$ es un parámetro de diseño, $\chi_{i,k}^j$ es el $j$-ésimo estado de la planta y $x_{i,k}^j$ el $j$-ésimo estado de la red neuronal. $L$ es el número de estados, $K_{i,k}^j$ es la matriz de ganancias de Kalman, $Q_{i,k}^j$ es la matriz de covarianza de ruido en la medición, $R_{i,k}^j$ es la matriz de covarianza de ruido en los estados, $\hat{x}_{i,k}^{j-}$ es la media de la predicción del estado, $\hat{y}_{i,k}^{j-}$ es la media de la predicción de salida, $\mathcal{X}_{i,k|k-1}^j$ y $\mathcal{Y}_{i,k|k-1}^j$ son la propagación de cada *punto sigma* a través de la predicción y la observación respectivamente. Es importante remarcar que $K_i^j$, $P_i^j$, $R_i^j$ y $Q_i^j$ se encuentran acotados para el UKF (Teorema 2.9).

La técnica central del algoritmo UKF es la generación de los *puntos sigma*, que es la principal característica de la UT

$$\mathcal{X}_{i,k-1}^j = [w_{i,k-1}^j \quad w_{i,k-1}^j \pm \sqrt{L+\lambda}\sqrt{P_{i,k-1}^j}] \tag{3.9}$$

donde $\mathcal{X}_{i,k-1}^j \in \mathbb{R}^{L\times 2L+1}$ es una matriz de puntos sigma, en la cual cada columna representa un *punto sigma*. Una vez que se han generado los *puntos sigma*, cada punto pasa a través de funciones no lineales y son usados para calcular

$$\begin{aligned}
\mathcal{X}_{i,k|k-1}^j &= F(\mathcal{X}_{i,k-1}^j, u_{i,k-1}),\\
\mathcal{Y}_{i,k|k-1}^j &= H(\mathcal{X}_{i,k-1}^j, u_{i,k-1}),\\
\hat{x}_{i,k}^{j-} &= \sum_{i=0}^{2L} \eta_i^{jm} \mathcal{X}_{i,k|k-1}^j,\\
\hat{y}_{i,k}^{j-} &= \sum_{i=0}^{2L} \eta_i^{jm} \mathcal{Y}_{i,k|k-1}^j,
\end{aligned} \tag{3.10}$$

donde $F$ es una función no lineal de predicción del estado, $H$ es una función no lineal de predicción de la salida; $\lambda$ es un parámetro de escalamiento y los vectores de peso $\eta_i^{jm}$ (promedio) y $\eta_i^{jc}$ (covarianza) se definen por

$$\begin{aligned}
\lambda &= \alpha^2(L+\kappa) - L,\\
\eta_0^{jm} &= \lambda/(L+\lambda),\\
\eta_0^{jc} &= \lambda/(L+\lambda) + 1 - \alpha^2 + \beta,\\
\eta_i^{jm} &= \eta_i^{jc} = \lambda/[2(L+\lambda)],\\
i &= 1,\ldots,2L.
\end{aligned}$$

El parámetro $\alpha$ determina la propagación de los *puntos sigma* ($10^{-4} \leq \alpha \leq 1$). $\beta$ incluye la información de la distribución previa (para distribuciones Gaussianas $\beta = 2$) y $\kappa = 3 - L$ es un parámetro de ajuste para reducir los errores de predicción globales.

Cabe destacar que el subíndice $k|k$-1 (que se lee como "$k$ *dado* $k$-1") significa que este es el valor predicho del *punto sigma* en base en la información del valor anterior. Una representación alternativa de $k|k$-1 es el superíndice $k^-$ [Rhudy and Gu, 2013].

Entonces la dinámica del error (3.8d) puede expresarse como

$$e_{i,k+1}^j = \tilde{w}_{i,k}^j z_{i,k}^j + \epsilon_{z_i^j}. \tag{3.11}$$

Por otro lado, el error de estimación del peso $\tilde{w}_{i,k}^j$ se obtiene con

$$\tilde{w}_{i,k+1}^j = \tilde{w}_{i,k}^j - K_{i,k}^j e_{i,k}^j. \tag{3.12}$$

El siguiente teorema se define para realizar el análisis de estabilidad para el $i$-ésimo subsistema de la RHONN (3.3) y para identificar el $i$-ésimo subsistema del sistema no lineal (3.1).

**Teorema 3.4** El $i$-ésimo subsistema de la RHONN (3.3) entrenada con el algoritmo basado en UKF (3.7) para identificar el $i$-ésimo subsistema de la planta no lineal representado como (3.1) pero en ausencia de interconexiones, asegura que el error de identificación (3.8d) se encuentre SGUUB; además, los pesos de la RHONN se mantienen acotados.

***Prueba.*** Se considera la siguiente función candidata de Lyapunov

$$V_{i,k}^j = \tilde{w}_{i,k}^{j\mathrm{T}} P_{i,k}^j \tilde{w}_{i,k}^j + e_{i,k}^{j2} \tag{3.13}$$

con la diferencia de Lyapunov definida como

$$\begin{aligned}
\Delta V_{i,k}^j &= V_{i,k+1}^j - V_{i,k}^j,\\
&= \tilde{w}_{i,k+1}^{jT} P_{i,k+1}^j \tilde{w}_{i,k+1}^j + e_{i,k+1}^{j2}\\
&\quad - \tilde{w}_{i,k}^{jT} P_{i,k}^j \tilde{w}_{i,k}^j - e_{i,k}^{j2}
\end{aligned} \tag{3.14}$$

Usando (3.11) y (3.12) en (3.13), (3.14) cambia a

$$\begin{aligned}
\Delta V_{i,k}^j = {} & [\tilde{w}_{i,k}^j - K_{i,k}^j e_{i,k}^j]^T [P_{i,k}^j - A_{i,k}^j][\tilde{w}_{i,k}^j - K_{i,k}^j e_{i,k}^j]\\
& + [\tilde{w}_{i,k}^j z_{i,k}^j + \epsilon_{z_i^j}]^2 - \tilde{w}_{i,k}^{jT} P_{i,k}^j \tilde{w}_{i,k}^j - e_{i,k}^{j2}
\end{aligned} \tag{3.15}$$

con $A^j_{i,k} = K^j_{i,k} P^{jyy}_{i,k} K^{jT}_{i,k}$; entonces, (3.15) puede expresarse como

$$\begin{aligned}\Delta V^j_{i,k} = {} & \tilde{w}^{jT}_{i,k} P^j_{i,k} \tilde{w}^j_{i,k} - e^j_{i,k} \tilde{w}^{jT}_{i,k} P^j_{i,k} K^j_{i,k} - \tilde{w}^{jT}_{i,k} A^j_{i,k} \tilde{w}^j_{i,k} + e^j_{i,k} \tilde{w}^{jT}_{i,k} A^j_{i,k} K^j_{i,k} \\ & - e^j_{i,k} K^{jT}_{i,k} P^j_{i,k} \tilde{w}^j_{i,k} + e^{j2}_{i,k} K^{jT}_{i,k} P^j_{i,k} K^j_{i,k} + e^j_{i,k} K^{jT}_{i,k} A^j_{i,k} \tilde{w}^j_{i,k} - e^{j2}_{i,k} K^{jT}_{i,k} A^j_{i,k} K^j_{i,k} \\ & + (\tilde{w}^j_{i,k} z^j_{i,k})^2 + 2\epsilon_{z^j_i} \tilde{w}^j_{i,k} z^j_{i,k} + \epsilon^2_{z^j_i} - \tilde{w}^{jT}_{i,k} P^j_{i,k} \tilde{w}^j_{i,k} - e^{j2}_{i,k}\end{aligned} \tag{3.16}$$

usando las siguientes desigualdades

$$\begin{aligned} & X^T X + Y^T Y \geq 2X^T Y \\ & X^T X + Y^T Y \geq -2X^T Y \\ & -\lambda_{\min}(P) X^2 \geq -X^T P X \geq -\lambda_{\max}(P) X^2 \end{aligned}$$

que son válidas $\forall\ X, Y \in \mathbb{R}^n$, $\forall\ P \in \mathbb{R}^{n\times n}$, $P = P^T > 0$, (3.16) se puede reescribir como

$$\begin{aligned}\Delta V^j_{i,k} \leq {} & -||\tilde{w}^j_{i,k}||^2 \lambda_{\min}(A^j_{i,k}) \\ & - e^{j2}_{i,k} ||K^j_{i,k}||^2 \lambda_{\min}(A^j_{i,k}) \\ & + e^{j2}_{i,k} ||K^j_{i,k}||^2 \lambda^2_{\text{máx}}(P^j_{i,k}) \\ & + ||\tilde{w}^j_{i,k}||^2 \lambda^2_{\text{máx}}(A^j_{i,k}) ||K^j_{i,k}||^2 \\ & + e^{j2}_{i,k} ||K^j_{i,k}||^2 \lambda_{\text{máx}}(P^j_{i,k}) \\ & + ||\tilde{w}^j_{i,k}||^2 + 2||\tilde{w}^j_{i,k}||^2 ||z^j_{i,k}||^2 + 2\epsilon^2_{z^j_i}\end{aligned} \tag{3.17}$$

Definiendo

$$\begin{aligned} E^j_{i,k} &= \lambda_{\min}(A^j_{i,k}) - \lambda^2_{\text{máx}}(A^j_{i,k}) ||K^j_{i,k}||^2 - 2||z^j_{i,k}||^2 - 1 \\ F^j_{i,k} &= ||K^j_{i,k}||^2 \lambda_{\min}(A^j_{i,k}) - ||K^j_{i,k}||^2 \lambda^2_{\text{máx}}(P^j_{i,k}) - ||K^j_{i,k}||^2 \lambda_{\text{máx}}(P^j_{i,k}) \end{aligned}$$

y seleccionando $Q^j_{i,k}$ y $R^j_{i,k}$, tales que $E^j_{i,k} > 0$ y $F^j_{i,k} > 0$, $\forall\ k$, (3.17) se expresa como

$$\Delta V^j_{i,k} \leq -||\tilde{w}^j_{i,k}||^2 E^j_{i,k} - |e^j_{i,k}|^2 F^j_{i,k} + 2\epsilon^2_{z^j_i}$$

Por lo tanto $\Delta V^j_{i,k} \leq 0$ cuando

$$||\tilde{w}^j_{i,k}|| > \frac{\sqrt{2}|\epsilon_{z^j_i}|}{\sqrt{E^j_{i,k}}} \equiv \kappa^j_{i,1}$$

y

$$|e^j_{i,k}| > \frac{\sqrt{2}|\epsilon_{z^j_i}|}{\sqrt{F^j_{i,k}}} \equiv \kappa^j_{i,2}$$

mientras, el error de estimación de peso $\tilde{w}^j_{i,k}$ y el error de identificación $e^j_{i,k}$ son últimamente acotados por

$$||\tilde{w}_{i,k}^{j}|| \leq b_1 \quad \forall\ k > k_1$$

y

$$|e_{i,k}^{j}| \leq b_2 \quad \forall\ k > k_2$$

donde

$$b_1 = \sqrt{\frac{\lambda_{\text{máx}}(P_{i,k}^{j})}{\lambda_{\min}(P_{i,k}^{j})}}\kappa_{i,1}^{j}$$

y

$$b_2 = \kappa_{i,2}^{j}$$

Por lo que, de acuerdo al Teorema 2.3, las soluciones de (3.11) y (3.12) son SGUUB (ultima y uniformemente acotadas de manera semiglobal).

## 3.4. Diseño del controlador

Para el seguimiento de la trayectoria del primer bloque en (3.1) es necesario definir el error de seguimiento como

$$\mathbf{z}_{i,k}^{1} = x_{i,k}^{1} - x_{i\text{d},k}^{1} \tag{3.18}$$

donde $x_{i\text{d},k}^{1}$ es la $i$-ésima señal de la trayectoria deseada y $x_{i,k}^{1}$ es el $i$-ésimo estado de la red neuronal, con $i = 1,\ldots,N$ subsistemas.

A partir de (3.18) se determina el siguiente valor como

$$\mathbf{z}_{i,k+1}^{1} = w_{i,k}^{1} S(\chi_{i,k}^{1}) + w_{i}^{'1}\chi_{i,k}^{2} - x_{i\text{d},k+1}^{1} \tag{3.19}$$

La ecuación (3.19) es vista como un bloque con el estado $\mathbf{z}_{i,k}^{1}$ y el estado $\chi_{i,k}^{2}$ es considerado como una entrada de pseudo-control, donde es posible imponer una dinámica deseada. Esta ecuación se puede resolver con la anticipación de la dinámica deseada para este bloque como

$$\begin{aligned}\mathbf{z}_{i,k+1}^{1} &= w_{i,k}^{1} S(\chi_{i,k}^{1}) + w_{i}^{'1}\chi_{i,k}^{2} - x_{i\text{d},k+1}^{1} \\ &= k_{i}^{1}\mathbf{z}_{i,k}^{1}\end{aligned} \tag{3.20}$$

donde $|k_i^1| < 1$ con el fin de garantizar la estabilidad de (3.20). $x_{i\text{d},k}^{2}$ se calcula como

$$x_{i\text{d},k}^{2} = \frac{1}{w_{i}^{'1}}[-w_{i,k}^{1} S(\chi_{i,k}^{1}) + \chi_{i\text{d},k+1}^{1} + k_{i}^{1}\mathbf{z}_{i,k}^{1}]. \tag{3.21}$$

Teniendo en cuenta que el valor calculado de estado $x^2_{i\mathrm{d},k}$ en (3.21) no es su valor real, en lugar de ello, representa el comportamiento deseado para $\chi^2_{i,k}$. Entonces, para evitar confusiones, se refiere al valor deseado de $\chi^2_{i,k}$ como $x^2_{i\mathrm{d},k}$ en (3.21).

De la misma manera que para el primer bloque, se puede definir una segunda variable en las nuevas coordenadas como:

$$\mathbf{z}^2_{i,k} = x^2_{i,k} - x^2_{i\mathrm{d},k}. \tag{3.22}$$

El siguiente valor para $\mathbf{z}^2_{i,k}$ produce

$$\mathbf{z}^2_{i,k+1} = x^2_{i,k+1} - x^2_{i\mathrm{d},k+1} \tag{3.23}$$

donde se impone la dinámica deseada para este bloque como

$$\begin{aligned} \mathbf{z}^2_{i,k+1} &= w^2_{i,k} S(\chi^1_{i,k}, \chi^2_{i,k}) + w'^2_i \chi^3_{i,k} - x^2_{i\mathrm{d},k+1} \\ &= k^2_i \mathbf{z}^2_{i,k} \end{aligned} \tag{3.24}$$

donde $|k^2_i| < 1$. De forma iterativa, se toman estos pasos, en donde en el último paso la variable deseada conocida es $x^r_{i\mathrm{d},k}$, y la última nueva variable se define como

$$\mathbf{z}^r_{i,k} = x^r_{i,k} - x^r_{i\mathrm{d},k}.$$

El nuevo valor se obtiene como

$$\mathbf{z}^r_{i,k+1} = w^r_{i,k} S(\chi^1_{i,k}, \ldots, \chi^r_{i,k}) + w'^r_i u_{i,k} - x^r_{i\mathrm{d},k+1} \tag{3.25}$$

El sistema (3.3) se puede representar en las nuevas variables $\mathbf{z} = \begin{bmatrix} z^{1T}_i & z^{2T}_i & \cdots & z^{rT}_i \end{bmatrix}$ como

$$\begin{aligned} \mathbf{z}^{r-1}_{i,k+1} &= k^{r-1}_i \mathbf{z}^{r-1}_{i,k} + w'^{r-1}_i \mathbf{z}^r_{i,k}, \\ \mathbf{z}^r_{i,k+1} &= w^r_{i,k} S(\chi^1_{i,k}, \ldots, \chi^r_{i,k}) + w'^2_i u_{i,k} - x^r_{i\mathrm{d},k+1}. \end{aligned} \tag{3.26}$$

Implementando un control por modos deslizantes, se tiene que cuando la entrada de control está limitada por $u_{0i}$

$$|u_{i,k}| \leq u_{0i} \tag{3.27}$$

se selecciona una superficie de deslizamiento como $S_{D_{i,k}} = \mathbf{z}^r_{i,k} = 0$, entonces el sistema (3.26) se reescribe de la siguiente manera:

$$S_{D_{i,k+1}} = w^r_{i,k} S(\chi^1_{i,k}, \ldots, \chi^r_{i,k}) + w'^2_i u_{i,k} - x^r_{i\mathrm{d},k+1} \tag{3.28}$$

Con la superficie de deslizamiento definida [Utkin et al., 2009], el siguiente paso es encontrar una ley de control que tome a consideración el acotamiento de (3.28), por lo tanto, el control $u_{i,k}$ se selecciona como

$$u_{i,k} = \begin{cases} u_{eq_{i,k}} & \text{for } \left\| u_{eq_{i,k}} \right\| \leq u_{0i}, \\ u_{0i} \dfrac{u_{eq_{i,k}}}{\left\| u_{eq_{i,k}} \right\|} & \text{for } \left\| u_{eq_{i,k}} \right\| > u_{0i}, \end{cases} \tag{3.29}$$

donde el control equivalente $u_{eq_{i,k}}$ se obtiene a partir de $S_{D_{i,k+1}} = 0$ como

$$u_{eq_{i,k}} = \frac{1}{w_i'^r}[-w_{i,k}^r S(\chi_{i,k}^1, \ldots, \chi_{i,k}^r) + x_{i\mathrm{d},k+1}^r] \tag{3.30}$$

Tanto el esquema de identificación como el esquema de control se ilustra en la Figura 3.1. Además es importante resaltar que los manipuladores robóticos utilizados en este trabajo se encuentran expresados en la forma NBCF.

## 3.5. Análisis de estabilidad

Para demostrar que las trayectorias del sistema en lazo cerrado sobre la superficie $S_{D_{i,k}}$ son estables se realiza el análisis de estabilidad con el siguiente teorema.

**Teorema 3.1** La ley de control (3.29) asegura que la superficie de deslizamiento $S_{D_{i,k}} = \mathbf{z}_{i,k}^r = 0$ es estable para el sistema (3.3).

***Prueba.***: Se reescribe $S_{D_{i,k+1}}$ como

$$S_{D_{i,k+1}} = S_{D_{i,k}} - x_{i,k}^r + x_{i\mathrm{d},k}^r + w_{i,k}^r S_i \left(\chi_{i,k}^1, ..., \chi_{i,k}^r\right) + w_i'^r u_{i,k} - x_{i\mathrm{d},k+1}^r$$

teniendo en cuenta que cuando $||u_{eq_{i,k}}|| \leq u_{0_i}$, el control equivalente se aplica y se produce el seguimiento de la superficie de deslizamiento $S_{D_{i,k}} = 0$. Por otro lado, cuando $||u_{eq_{i,k}}|| > u_{0_i}$, la estrategia de control cambia a $u_{0_i} \frac{u_{eq_{i,k}}}{||u_{eq_{i,k}}||}$, lo cual convierte al sistema en lazo cerrado en

$$\begin{aligned} S_{D_{i,k+1}} &= S_{D_{i,k}} - x_{i,k}^r + x_{i\mathrm{d},k}^r + w_{i,k}^r S_i \left(\chi_{i,k}^1, ..., \chi_{i,k}^r\right) + w_i'^r u_{0_i} \frac{u_{eq_{i,k}}}{||u_{eq_{i,k}}||} - x_{i\mathrm{d},k+1}^r, \\ &= \left(S_{D_{i,k}} - x_{i,k}^r + x_{i\mathrm{d},k}^r + w_{i,k}^r S_i \left(\chi_{i,k}^1, ..., \chi_{i,k}^r\right) - x_{i\mathrm{d},k+1}^r\right)\left(1 - \frac{u_{0_i}}{||u_{eq_{i,k}}||}\right), \\ &= \left(S_{D_{i,k}} + f_{s_{i,k}}\right)\left(1 - \frac{u_{0_i}}{||u_{eq_{i,k}}||}\right), \end{aligned}$$

donde $f_{s_{i,k}} = -x_{i,k}^r + x_{i\mathrm{d},k}^r + w_{i,k}^r S_i \left(\chi_{i,k}^1, ..., \chi_{i,k}^r\right) - x_{i\mathrm{d},k+1}^r$. A lo largo de cualquier solución del sistema, la diferencia de Lyapunov (3.14) con $V_{i,k} = ||S_{D_{i,k}}||$ se tiene que

$$
\begin{aligned}
\Delta V_{i,k} &= ||S_{D_{i,k+1}}|| - ||S_{D_{i,k}}||, \\
&= ||S_{D_{i,k}} + f_{s_{i,k}}|| \left(1 - \frac{u_{0_i}}{||u_{eq_{i,k}}||}\right) - ||S_{D_{i,k}}||, \\
&= ||S_{D_{i,k}} + f_{s_{i,k}}|| \left(1 - \frac{u_{0_i}}{||\frac{1}{w_i'^r}||\,||S_{D_{i,k}} + f_{s_{i,k}}||}\right) - ||S_{D_{i,k}}||, \\
&= ||S_{D_{i,k}} + f_{s_{i,k}}|| - \frac{u_{0_i}}{||\frac{1}{w_i'^r}||} - ||S_{D_{i,k}}||, \\
&\leq ||S_{D_{i,k}}|| + ||f_{s_{i,k}}|| - \frac{u_{0_i}}{||\frac{1}{w_i'^r}||} - ||S_{D_{i,k}}||, \\
&\leq ||f_{s_{i,k}}|| - \frac{u_{0_i}}{||\frac{1}{w_i'^r}||}.
\end{aligned}
$$

En [Utkin et al., 2009] se muestra que bajo la condición $u_{0_i} > ||\frac{1}{w_i'^r}||\,||f_{s_{i,k}}||$ en lazo cerrado, el vector de estados alcanza la superficie de deslizamiento $S_{D_{i,k}}$ en tiempo finito. Entonces $S_{D_{i,k}} = 0$ se rige por el siguiente sistema de orden reducido (Ecuación de modos deslizantes, SME por sus siglas en inglés *Sliding Mode Equation*)

$$
\begin{aligned}
z^1_{i,k+1} &= k^1_i z^1_{i,k} + w'^1_i z^2_{i,k}, \\
z^2_{i,k+1} &= k^2_i z^2_{i,k} + w'^2_i z^3_{i,k}, \\
z^{r-1}_{i,k+1} &= k^{r-1}_i z^{r-1}_{i,k},
\end{aligned}
\tag{3.31}
$$

y si $|k^j_i| < 1$, el sistema (3.31) es asintóticamente estable.

**Lema 3.2** Si $lim_{t\to\infty}||x^j_{i\mathrm{d}} - x^j_i|| = 0$ y se cumple que $u_{0_i} > ||\frac{1}{w_i'^r}||\,||f_{s_{i,k}}||$, entonces la función candidata de Lyapunov para todo el sistema definida como

$$
V_k = \sum_{i=1}^{N} V_{i,k}
$$

cuya $\Delta V_k < 0$ garantiza la estabilidad asintótica para el sistema interconectado.

**Prueba.** Sea $V_k = \sum_{i=1}^{N} V_{i,k}$ entonces

$$
\begin{aligned}
\Delta V_k &= \sum_{i=1}^{N} \left(||S_{D_{i,k+1}}|| - ||S_{D_{i,k}}||\right) \\
&= \sum_{i=1}^{N} \left(||S_{D_{i,k}} + f_{s_{i,k}}|| \left(1 - \frac{u_{0_i}}{||u_{eq_{i,k}}||}\right) - ||S_{D_{i,k}}||\right) \\
&= \sum_{i=1}^{N} \left(||S_{D_{i,k}} + f_{s_{i,k}}|| \left(1 - \frac{u_{0_i}}{||\frac{1}{w_i'^r}||\,||S_{D_{i,k}} + f_{s_{i,k}}||}\right) - ||S_{D_{i,k}}||\right) \\
&\leq \sum_{i=1}^{N} \left(||f_{s_{i,k}}|| - \frac{u_{0_i}}{||\frac{1}{w_i'^r}||}\right)
\end{aligned}
$$

y como se cumple con $u_{0_i} > ||\frac{1}{w_i'^r}||\,||f_{s_{i,k}}||$ entonces $\Delta V_k < 0$.

# Capítulo 4

# Aplicación a robots manipuladores

## 4.1. Prototipos experimentales

### 4.1.1. Robot manipulador de 2 g.d.l.

El objetivo de control es obtener un buen desempeño en el seguimiento de trayectorias para el manipulador robótico de accionamiento directo vertical de 2 g.d.l., el cual se encuentra en el Laboratorio de Mecatrónica y Control del Instituto Tecnológico de la Laguna.

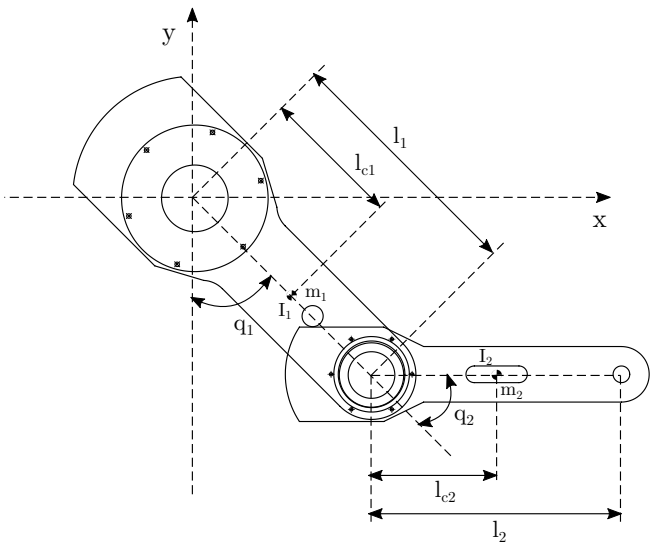


Figura 4.1: Robot manipulador de 2 g.d.l.

El manipulador se ilustra en la Figura 4.1, el cual consiste en dos eslabones rígidos que son articulados por los servomotores DM 1200-A y DM 1200-B, para el hombro y el codo respectivamente, de accionamiento directo sin escobillas de alto par sin reducción de engranajes, que presentan un juego (backlash) reducido y menor fricción en comparación con los actuadores con engranajes [Reyes and Kelly, 1997, Reyes and Kelly, 2001, García-Hernández et al., 2009].

En la Tabla 4.1 se muestran los valores numéricos de los parámetros ilustrados en la Figura 4.1.

Tabla 4.1: Parámetros del robot de 2 g.d.l.

| Descrición | Notación | Valor | Unidad |
|---|---|---|---|
| Masa del eslabón 1 | $m_1$ | 23.902 | $[kg]$ |
| Masa del eslabón 2 | $m_2$ | 3.880 | $[kg]$ |
| Longitud del eslabón 1 | $l_1$ | 0.450 | $[m]$ |
| Longitud del eslabón 2 | $l_2$ | 0.450 | $[m]$ |
| Inercia del eslabón 1 | $I_1$ | 1.266 | $[kg\ m^2]$ |
| Inercia del eslabón 2 | $I_2$ | 0.093 | $[kg\ m^2]$ |
| Centro de masa del eslabón 1 | $l_{c1}$ | 0.091 | $[m]$ |
| Centro de masa del eslabón 2 | $l_{c2}$ | 0.048 | $[m]$ |

El modelo dinámico no lineal de un robot manipulador se escribe de manera compacta:

$$\ddot{q} = M(q)^{-1}\left[\tau - C(q,\dot{q}) - g(q) - \tau_f(\dot{q})\right] \tag{4.1}$$

De acuerdo con [Kelly and Santibáñez, 2003] las matrices y vectores $M(q)$, $C(q,\dot{q})$ y $g(q)$ a utilizar en la simulación fueron las siguientes

$$M(q) = \begin{bmatrix} 0.351 + 0.168\cos(q(2)) & 0.102 + 0.084\cos(q_2) \\ 0.102 + 0.084\cos(q_2) & 0.102 \end{bmatrix}$$
$$C(q,\dot{q}) = \begin{bmatrix} -0.168\,\mathrm{sen}(q_2)\dot{q}_2 & -0.084\,\mathrm{sen}(q_2)\dot{q}_2 \\ 0.084\,\mathrm{sen}(q(2))\dot{q}_1 & 0 \end{bmatrix}$$
$$g(q) = \begin{bmatrix} 9.81(3.921\,\mathrm{sen}(q_1)) + 0.186\,\mathrm{sen}(q_1+q_2)) \\ 9.81(0.186\,\mathrm{sen}(q_1+q_2)) \end{bmatrix}$$
$$\tau_f = \begin{bmatrix} f_{c1}sgn(\dot{q}_1) + f_{v1}\dot{q}_1 \\ f_{c2}sgn(\dot{q}_2) + f_{v2}\dot{q}_2 \end{bmatrix}$$

donde los coeficientes de fricción viscosa y de Coulomb se proporcionan a través de la Tabla 4.2

Tabla 4.2: Parámetros de fricción cinética para el robot de 2 g.d.l.

| Fricción de Coulomb | Velocidad + | Velocidad - |
|---|---|---|
| $f_{c1}$ | 7.170 [Nm] | 8.049 [Nm] |
| $f_{c2}$ | 1.734 [Nm] | 1.734 [Nm] |
| Fricción Viscosa | Valor | Unidad |
| $f_{v1}$ | 2.288 | [Nm/rad] |
| $f_{v2}$ | 0.175 | [Nm/rad] |

Se propone la siguiente configuración de una red neuronal descentralizada (serie-paralelo) para identificar el modelo del manipulador robótico

$$\begin{aligned} x^1_{i,k+1} &= w^1_{i1,k} S\left(\chi^1_{i,k}\right) + w'^1_i \chi^2_{i,k}, \\ x^2_{i,k+1} &= w^2_{i1,k} S\left(\chi^1_{i,k}\right) + w^2_{i2,k} S\left(\chi^2_{i,k}\right) + w'^2_i u_{i,k} \end{aligned} \tag{4.2}$$

donde $x^1_{i,k}$ y $x^2_{i,k}$ identifican los estados $\chi^1_{i,k}$ y $\chi^2_{i,k}$; $i = 1,2$, respectivamente. Los pesos ajustables son denotados por $w^j_{ip,k}$; en donde $p$ es el número de pesos ajustables con $p = 1$ y $j = 1$ para el primer estado de la NN. Para el segundo estado tenemos que $p = 1,2$ y $j = 1,2$. $w'^1_i$ y $w'^2_i$ son parámetros constantes.

Debido a la variación en el tiempo de los pesos de la RHONN, es necesario garantizar la controlabilidad del sistema asegurando que los pesos $w'^1_i$ y $w'^2_i$ no sean cero. Para actualizar los pesos $w^j_{ip}$ se implementa un algoritmo de entrenamiento basado en UKF.

El objetivo es seguir una señal de referencia deseada, diseñando una ley de control basada en la técnica de modos deslizantes. El error de seguimiento se define como

$$\mathbf{z}^1_{i,k} = x^1_{i,k} - x^1_{i\mathrm{d},k} \tag{4.3}$$

donde $x^1_{i\mathrm{d},k}$ es la señal de la trayectoria deseada. Usando (4.2) e imponiendo la dinámica deseada para $\mathbf{z}^1_{i,k}$ se tiene

$$\begin{aligned} \mathbf{z}^1_{i,k+1} &= w^1_{i1,k} S(\chi^1_{i,k}) + w'^2_i \chi^2_{i,k} - x^1_{i\mathrm{d},k+1} \\ &= k^1_i \mathbf{z}^1_{i,k}. \end{aligned} \tag{4.4}$$

El valor deseado $x^1_{i\mathrm{d},k+1}$ para $\chi^2_{i,k}$ se calcula a partir de (4.4) como

$$x^2_{i\mathrm{d},k+1} = \frac{1}{w'^1_i}[-w^1_{i1,k} S(\chi^1_{i,k}) + \chi^1_{i\mathrm{d},k+1} + k^1_i \mathbf{z}^1_{i,k}]. \tag{4.5}$$

En el siguiente paso, se necesita definir una nueva variable como

$$\mathbf{z}^2_{i,k} = x^2_{i,k} - x^2_{i\mathrm{d},k}. \tag{4.6}$$

El siguiente valor se obtiene como

$$\begin{aligned} \mathbf{z}^2_{i,k+1} &= w^2_{i1,k} S\left(\chi^1_{i,k}\right) + w^2_{i2,k} S\left(\chi^2_{i,k}\right) + w'^2_i \chi^3_{i,k} - x^2_{i\mathrm{d},k+1} \\ &= k^2_i \mathbf{z}^2_{i,k} \end{aligned} \tag{4.7}$$

la superficie deslizante se elige como $S_{D_i,k} = z^2_{i,k} = 0$. La ley de control está dada por

$$u_{i,k} = \begin{cases} u_{eq_{i,k}} & \text{for} \left\| u_{eq_{i,k}} \right\| \leq \tau^{\max}_i, \\ u_{0i} \dfrac{u_{eq_{i,k}}}{\left\| u_{eq_{i,k}} \right\|} & \text{for} \left\| u_{eq_{i,k}} \right\| > \tau^{\max}_i, \end{cases} \tag{4.8}$$

donde el $u_{eq_{i,k}}$ se calcula a partir de $S_{D_{i,k+1}} = 0$ como

$$u_{eq_{i,k}} = \frac{1}{w'^2_i}[-(w^2_{i1,k} S(\chi^1_{i,k}) + w^2_{i2,k} S(\chi^2_{i,k})) + x^2_{i\mathrm{d},k+1}] \tag{4.9}$$

y las entradas de control están acotadas por $\tau^{\max}_i$.

### 4.1.2. Robot industrial Mitsubishi PA10-7CE

El robot manipulador industrial Mistubishi PA10-7CE es una de las versiones del Brazo Inteligente Portátil de Propósito General (Portable General-Purpose Intelligent Arm PA10) de arquitectura abierta desarrollado por Mitsubishi Heavy Industries (MHI). Esta versión se encuentra disponible en el Laboratorio de Mecatrónica y Control del Instituto Tecnológico de La Laguna, y está compuesto por siete articulaciones conectadas a través de eslabones tal y como se ilustra en la Figura 4.2.

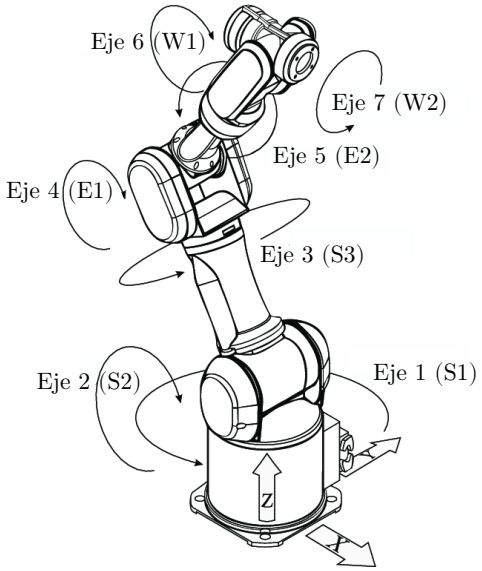


Figura 4.2: Estructura del manipulador Mitsubishi PA10-7CE

El manipulador Mitsubishi PA10-7CE es un robot de arquitectura abierta que cuenta con las siguientes características

- Estructura jerárquica con cuatro niveles de control
- Comunicación entre niveles por medio de interfaces estandarizadas
- Interfaz abierta en el último nivel de control

Existen dos versiones del robot Mitsubishi PA10: el PA10-6C y el PA10-7C, donde el dígito del sufijo indica el número de grados de libertad del brazo. Los modelos con el sufijo “CE” son la versión mejorada de los “C”. Existen las dos versiones del robot Mitsubishi CE, la versión de 6 grados de libertad (PA10-6CE), y otra de siete grados de libertad (PA10-7CE). La diferencia radica en que el PA10-7CE tiene dos eslabones en lugar del eslabón 2 del PA10-6CE, el resto de los eslabones es idéntico para uno y otro modelo.

El sistema PA10 consiste en cuatro niveles de control los cuales fueron modificados para controlar el robot en modo par [Ramírez-Woo, 2008, Flores-Ávila, 2014], los cuales son los siguientes y se muestran en la Figura 4.3

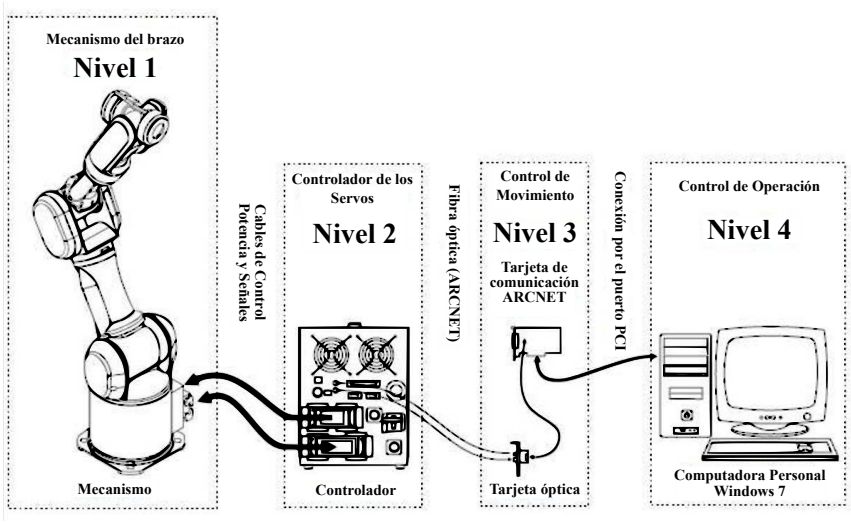


Figura 4.3: Componentes del sistema PA10-7CE

1. Brazo robótico
2. Controlador de Servomotores
3. Sección de Control de Movimiento
4. Sección de Control de Operación

Estas características proporcionan al usuario la facilidad de trabajar y programar tareas (Nivel 4) dentro de una interfaz gráfica de usuario (GUI por sus siglas en inglés *Graphics User Interface*), desarrollado por [Flores-Ávila, 2014] dentro de MATLAB/Simulink, sin preocuparse por los niveles más bajos.

En la Tabla 4.3 se muestran las consignas de par máximo que se pueden aplicar a las articulaciones

Tabla 4.3: Pares máximos de las articulaciones para el brazo robótico Mitsubishi PA10-7CE

| **Articulación** | **Par Máximo** |
|---|---|
| S1 | $\pm$ 232 $Nm$ |
| S2 | $\pm$ 232 $Nm$ |
| S3 | $\pm$ 100 $Nm$ |
| E1 | $\pm$ 100 $Nm$ |
| E2 | $\pm$ 14.5 $Nm$ |
| W1 | $\pm$ 14.5 $Nm$ |
| W2 | $\pm$ 14.5 $Nm$ |

En trabajos anteriores [Ramírez-Woo, 2008, Sánchez-Mazuca, 2013, Flores-Ávila, 2014] ya se ha tratado el modelado del manipulador PA10-7CE; en base a estos trabajos, y con la ayuda de las herramientas SYMORO+ [Khalil et al., 1989] y HEMERO [Maza and Ollero, 2001] se obtuvo un modelo de una manera numérica y semi-simbólica (ver Apéndice D). Si se desea conocer más a fondo sobre la

dinámica y cinemática del manipulador industrial se recomienda apoyarse en textos especializados como [Pholsiri, 2004].

Además, como el objetivo de esta tesis no consiste en la obtención del modelo matemático se utilizaron los parámetros de fricción y las consignas del robot PA10 reportados en [Ramírez-Woo, 2008, Sánchez-Mazuca, 2013].

El objetivo es seguir una señal de referencia, diseñando una ley de control que combina control por bloques y técnicas de modos deslizantes. El error de seguimiento se define como

$$\mathbf{z}_{i,k}^{1} = x_{i,k}^{1} - x_{i\mathrm{d},k}^{1} \tag{4.10}$$

donde $x_{i\mathrm{d},k}^{1}$ es la señal de la trayectoria deseada. Usando (3.3) e imponiendo la dinámica deseada para $\mathbf{z}_{i,k}^{1}$ se tiene que

$$\begin{aligned} \mathbf{z}_{i,k+1}^{1} &= w_{i1,k}^{1} S(\chi_{i,k}^{1}) + w_{i}^{\prime 2} \chi_{i,k}^{2} - x_{i\mathrm{d},k+1}^{1} \\ &= k_{i}^{1} \mathbf{z}_{i,k+1}^{1} \end{aligned} \tag{4.11}$$

donde el valor deseado $x_{i\mathrm{d},k+1}^{1}$ para $\chi_{i,k}^{2}$ se calcula a partir de (4.12) como

$$x_{i\mathrm{d},k+1}^{2} = \frac{1}{w_{i}^{\prime 1}} [-w_{i1,k}^{1} S(\chi_{i,k}^{1}) + \chi_{i\mathrm{d},k+1}^{1} + k_{i}^{1} \mathbf{z}_{i,k}^{1}]. \tag{4.12}$$

En el siguiente paso, se necesita definir una nueva variable como

$$\mathbf{z}_{i,k}^{2} = x_{i,k}^{2} - x_{i\mathrm{d},k}^{2} \tag{4.13}$$

El siguiente valor se obtiene como

$$\begin{aligned} \mathbf{z}_{i,k+1}^{2} &= w_{i1,k}^{2} S\left(\chi_{i,k}^{1}\right) + w_{i2,k}^{2} S\left(\chi_{i,k}^{2}\right) + w_{i}^{\prime 2} \chi_{i,k}^{3} - x_{i\mathrm{d},k+1}^{2} \\ &= k_{i}^{2} \mathbf{z}_{i,k}^{2} \end{aligned} \tag{4.14}$$

La superficie de deslizamiento se elige como $S_{D_i,k} = z_{i,k}^{2} = 0$, mientras que la ley de control está dada por

$$u_{i,k} = \begin{cases} u_{eq_{i,k}} & \text{for} \left\| u_{eq_{i,k}} \right\| \leq \tau_{i}^{\max} \\ u_{0i} \dfrac{u_{eq_{i,k}}}{\left\| u_{eq_{i,k}} \right\|} & \text{for} \left\| u_{eq_{i,k}} \right\| > \tau_{i}^{\max} \end{cases} \tag{4.15}$$

donde el $u_{eq_{i,k}}$ se calcula a partir de $S_{D_{i,k+1}} = 0$ como

$$u_{eq_{i,k}} = \frac{1}{w_{i}^{\prime 2}} [-(w_{i1,k}^{2} S(\chi_{i,k}^{1}) + w_{i2,k}^{2} S(\chi_{i,k}^{2})) + x_{i\mathrm{d},k+1}^{2}] \tag{4.16}$$

y las entradas de control están acotadas por $\pm\tau_{i}^{\max}$.

## 4.2. Resultados en simulación

### 4.2.1. Robot manipulador de 2 g.d.l.

Para la simulación del controlador descentralizado neuronal por bloques (DNBC), se eligieron las trayectorias discretas propuestas en [García-Hernández et al., 2009]

$$x^1_{1\mathrm{d},k} = b_1(1-e^{d_1 kT^3}) + c_1(1-e^{d_1 kT^3})\,\mathrm{sen}(\omega_1 kT)[^\circ]$$
$$x^1_{2\mathrm{d},k} = b_2(1-e^{d_2 kT^3}) + c_2(1-e^{d_2 kT^3})\,\mathrm{sen}(\omega_2 kT)[^\circ]$$

donde $b_1 = 45^\circ$, $c_1 = 10^\circ$, $d_1 = -2.0$, y $\omega_1 = 286.4789$ [°/s] son los parámetros de la trayectoria deseada de la primera articulación, mientras que $b_2 = 60^\circ$, $c_2 = 125^\circ$, $d_2 = -1.8$, y $\omega_2 = 57.2958$ [°/s] son los parámetros de la trayectoria deseada de la segunda articulación; con un periodo de muestreo $T = 2.5\ ms$.

Las trayectorias seleccionadas incorporan un término sinusoidal para evaluar el rendimiento en presencia de señales periódicas relativamente rápidas, para las cuales las no linealidades de la dinámica del robot son realmente importantes y presentan un término que crece sin problemas para mantener el robot en un estado de operación sin saturar los actuadores, cuyos límites son $\pm 150$ [$Nm$] y $\pm 15$ [$Nm$], respectivamente [Reyes and Kelly, 1996].

Las Figuras 4.4 y 4.5 muestran los resultados de la simulación para la identificación y el seguimiento de las trayectorias utilizando el algoritmo de control (DNBC) con entrenamiento UKF.

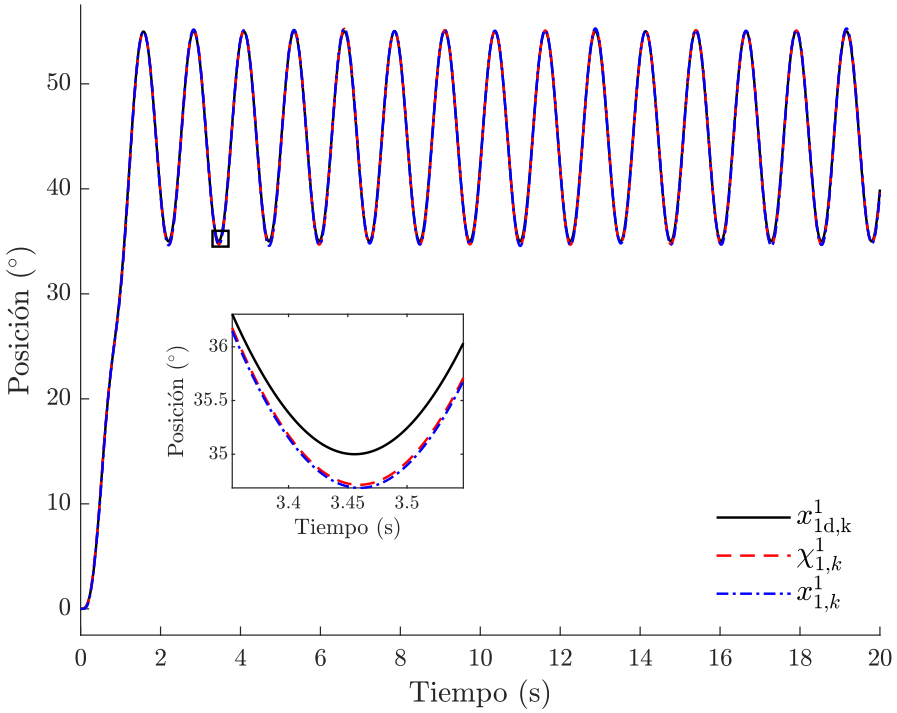


Figura 4.4: Identificación y seguimiento del eslabón 1 con aprendizaje UKF

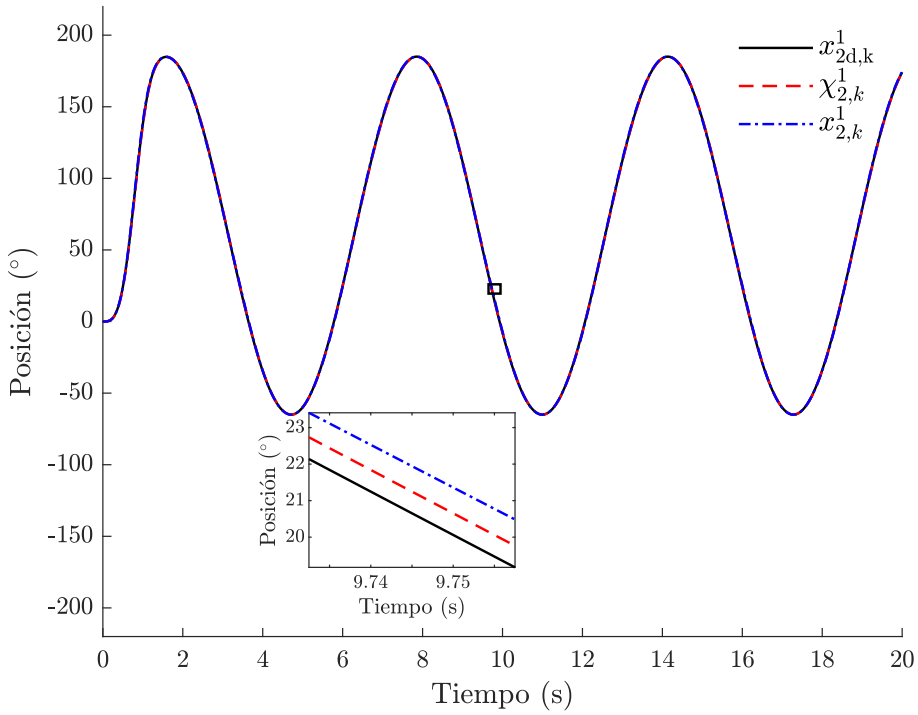


Figura 4.5: Identificación y seguimiento del eslabón 2 con aprendizaje UKF

Las condiciones iniciales para la planta son las mismas que las del identificador neuronal, que son iguales a cero para ambas articulaciones. De acuerdo con estas figuras, los errores de identificación para todas las articulaciones presentan un buen comportamiento y permanecen limitados como se muestra en la Figura 4.6. Los errores de seguimiento se presentan en la Figura 4.7. Los pares aplicados se muestran en la Figura 4.8.

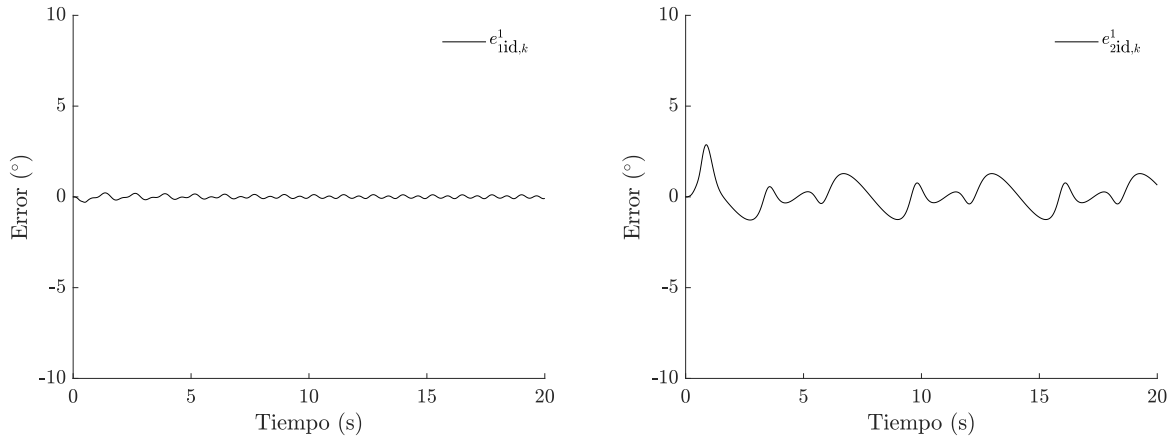


Figura 4.6: Errores de identificación de los eslabones 1 y 2 con aprendizaje UKF

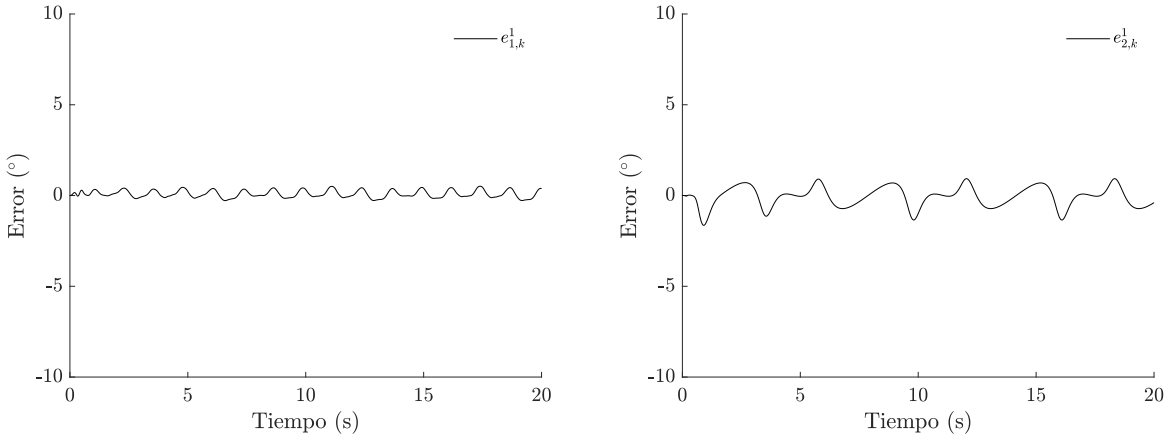


Figura 4.7: Errores de seguimiento de los eslabones 1 y 2 con aprendizaje UKF

Las oscilaciones de las señales de control, que se presentan en algunos instantes se deben a las ganancias y los parámetros constantes elegidos para el controlador. Se observa fácilmente que, ambas señales de control siempre se encuentran dentro de los límites de los actuadores $\pm\tau_1^{máx}$ y $\pm\tau_2^{máx}$, respectivamente. En las Figuras 4.9 a la 4.13 se muestran los resultados del algoritmo DNBC con entrenamiento utilizando el filtro de Kalman Extendido con el objetivo de realizar una comparación con la metodología propuesta.

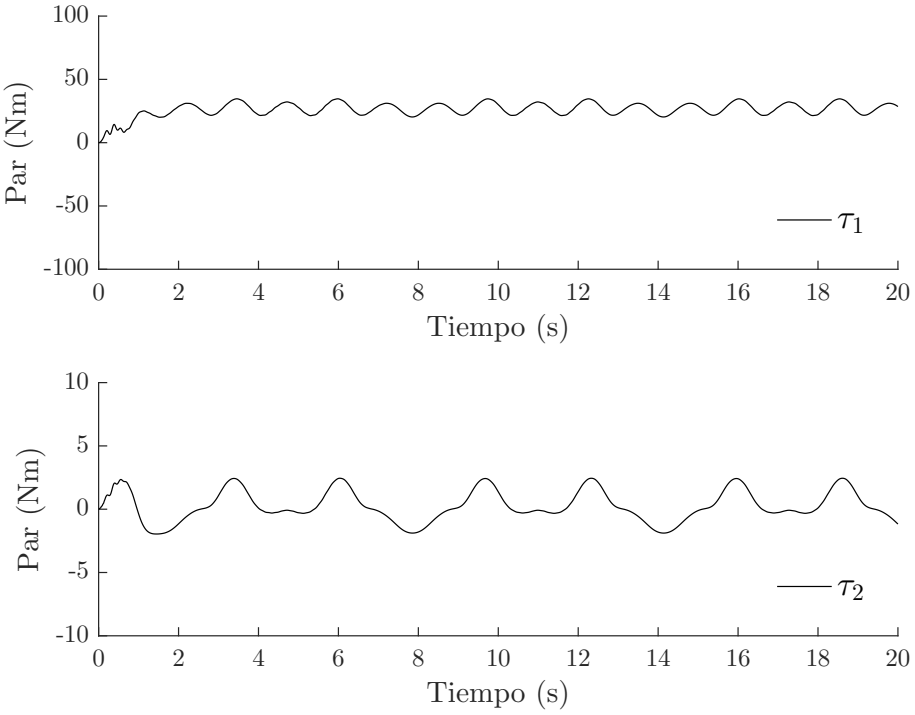


Figura 4.8: Pares aplicados a los eslabones 1 y 2 con aprendizaje UKF

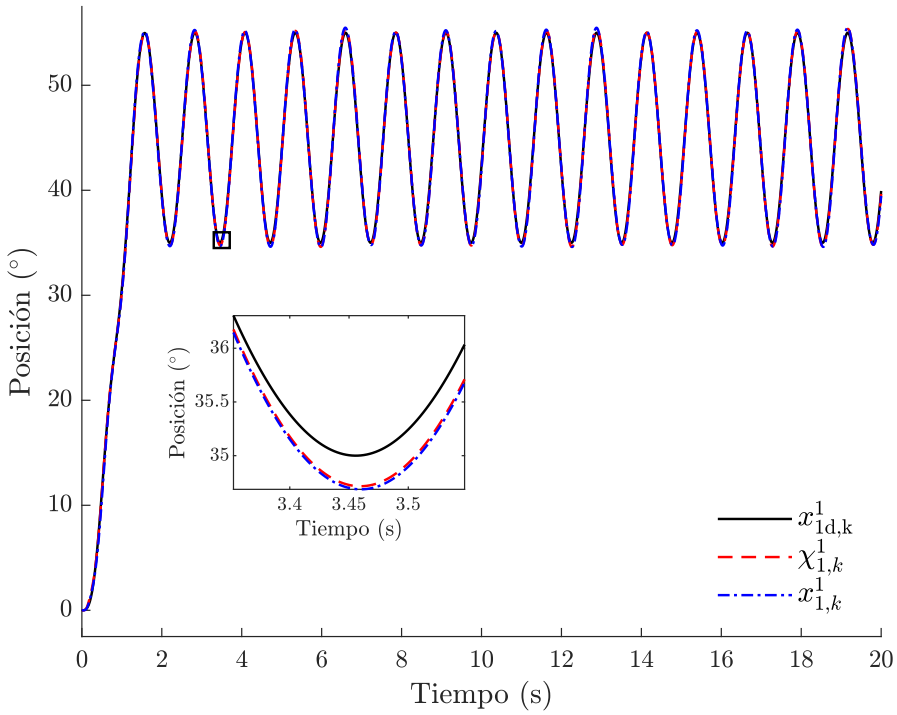


Figura 4.9: Identificación y seguimiento del eslabón 1 con aprendizaje EKF

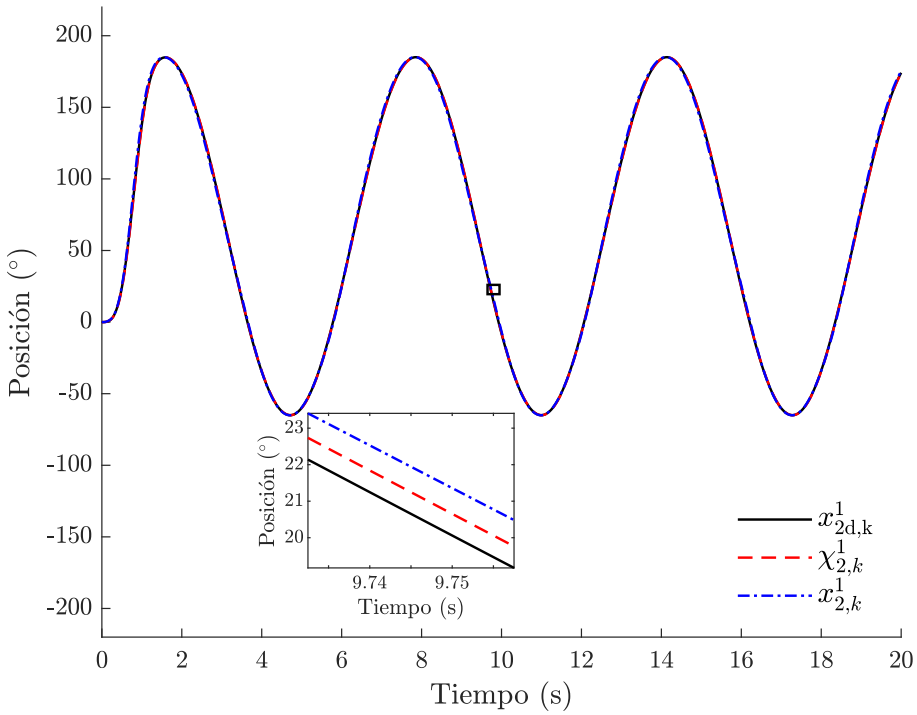


Figura 4.10: Identificación y seguimiento del eslabón 2 con aprendizaje EKF

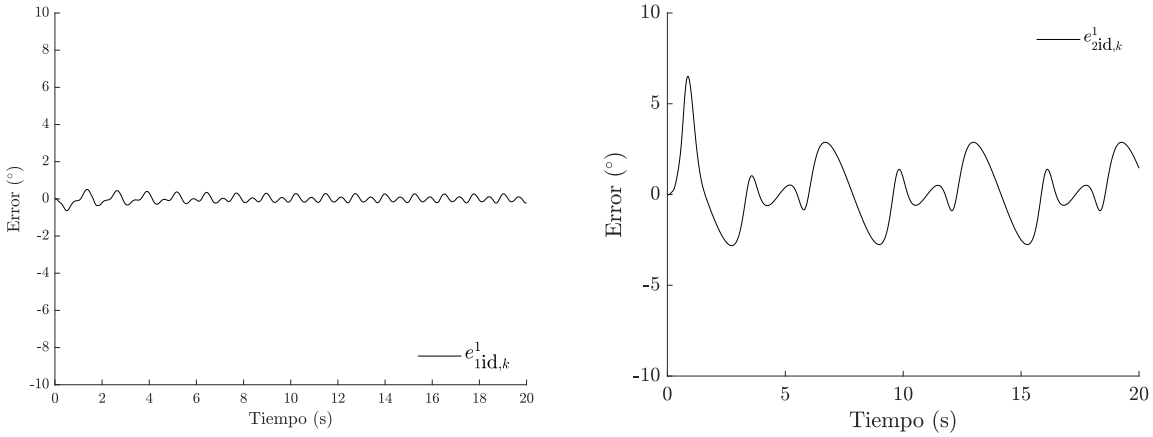


Figura 4.11: Errores de identificación de los eslabones 1 y 2 con aprendizaje EKF

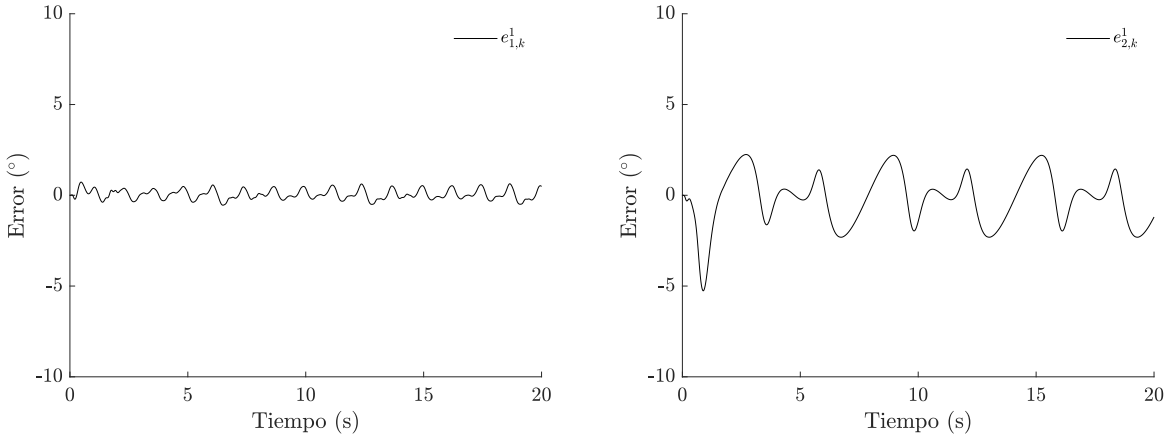


Figura 4.12: Errores de seguimiento de los eslabones 1 y 2 con aprendizaje EKF

Los filtros de Kalman brindan una manera de estimar el vector de estado utilizando una ganancia de observador óptima para minimizar la media del error cuadrático (MSE) esperado de la señal estimada. Este hecho permite analizar y comparar el desempeño de los dos algoritmos de entrenamiento, por medio del MSE del error de seguimiento $e^1_{i_{Track},k}$, calculado como

$$MSE[e^1_{i_{Track},k}] = \sqrt{\frac{1}{t}\sum_{k=0}^{n} ||e^1_{i_{Track},k}||^2\, T} \tag{4.17}$$

donde $e^1_{i_{Track},k} = x^1_{i\mathrm{d},k} - \chi^1_{i,k}$,$x^1_{i\mathrm{d},k}$ es la trayectoria deseada, $\chi^1_{i,k}$ es la salida del sistema, $T$ es el periodo de muestreo y $t$ es el tiempo de simulación. Los resultados se muestran en la Tabla 4.4

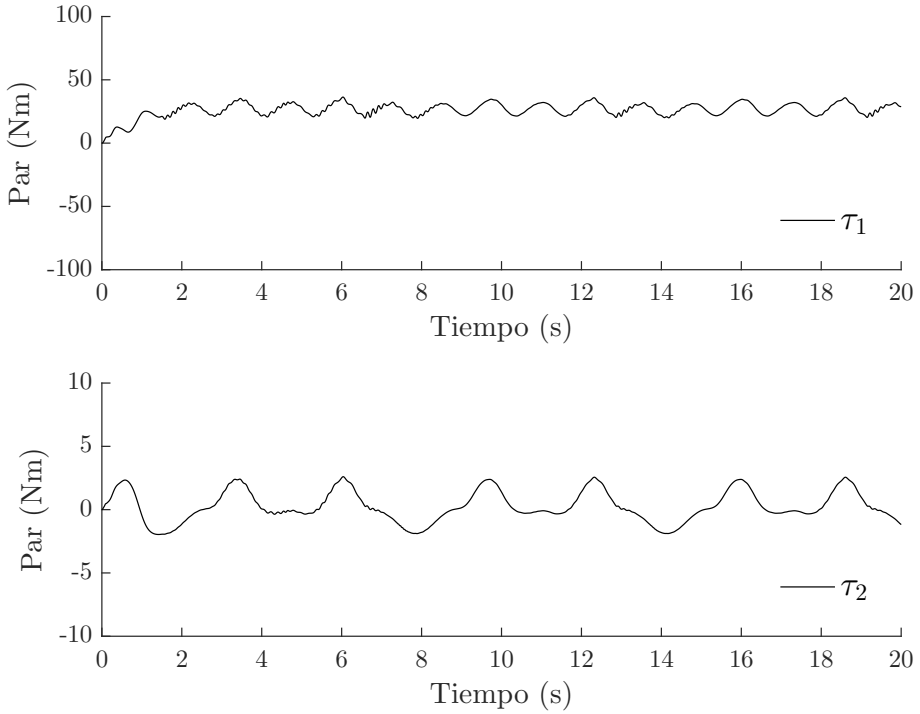


Figura 4.13: Pares aplicados a los eslabones 1 y 2 con aprendizaje EKF

Tabla 4.4: MSE del error de seguimiento en simulación para el manipulador robótico de 2 g.d.l.

| | Algoritmo | |
|---|---|---|
| Eslabón | UKF | EKF |
| 1 | 0.1653° | 0.1899° |
| 2 | 0.3283° | 0.7087° |

Los resultados de simulación obtenidos son alentadores. Además, como se muestra en la comparación entre los dos algoritmos de entrenamiento, el entrenamiento UKF muestra un mejor rendimiento y es más fácil de implementar en los manipuladores robóticos. Es importante destacar que el esquema de control propuesto no requiere conocer los parámetros de la planta ni las perturbaciones externas.

Todas las simulaciones fueron realizadas en el paquete Simulink de MATLAB en la versión R2015a; además fueron utilizadas para elaborar el artículo mostrado en el Apéndice A.

### 4.2.2. Robot industrial Mitsubishi PA10-7CE

Para la simulación del controlador descentralizado neuronal por bloques (por sus siglas en inglés DNBC), se eligieron las trayectorias discretas propuestas en [García-Hernández et al., 2011]

$$\begin{aligned}
x^1_{1\mathrm{d},k} &= c_1(1-e^{d_1 kT^3})\,\mathrm{sen}(\omega_1 kT)[^\circ] \\
x^1_{2\mathrm{d},k} &= c_2(1-e^{d_2 kT^3})\,\mathrm{sen}(\omega_2 kT)[^\circ] \\
x^1_{3\mathrm{d},k} &= c_3(1-e^{d_3 kT^3})\,\mathrm{sen}(\omega_3 kT)[^\circ] \\
x^1_{4\mathrm{d},k} &= c_4(1-e^{d_4 kT^3})\,\mathrm{sen}(\omega_4 kT)[^\circ] \\
x^1_{5\mathrm{d},k} &= c_5(1-e^{d_5 kT^3})\,\mathrm{sen}(\omega_5 kT)[^\circ] \\
x^1_{6\mathrm{d},k} &= c_6(1-e^{d_6 kT^3})\,\mathrm{sen}(\omega_6 kT)[^\circ] \\
x^1_{7\mathrm{d},k} &= c_7(1-e^{d_7 kT^3})\,\mathrm{sen}(\omega_7 kT)[^\circ]
\end{aligned} \tag{4.18}$$

donde los parámetros $c_i$, $d_i$ y $\omega_i$ para las trayectorias deseadas de cada articulación se encuentran en la Tabla 4.5. El periodo de muestreo es de $T = 1\ ms$.

Tabla 4.5: Parámetros de trayectorias deseadas para el brazo robótico Mitsubishi PA10-7CE

| Eslabón | $c_i$ | $d_i$ | $\omega_i$ |
|---|---|---|---|
| 1 | 90 | -0.001 | 16.3292°/s |
| 2 | 60 | -0.001 | 24.9236°/s |
| 3 | 90 | -0.01 | 31.7991°/s |
| 4 | 60 | -0.01 | 36.9557°/s |
| 5 | 90 | -0.01 | 19.7670°/s |
| 6 | 60 | -0.01 | 35.2369°/s |
| 7 | 90 | -0.01 | 26.6425°/s |

Las trayectorias seleccionadas en (4.18) permiten evaluar el rendimiento en presencia de señales periódicas relativamente rápidas, para las cuales las no linealidades de la dinámica del robot son realmente importantes y presentan un término que crece sin problemas para mantener el robot en un estado de operación sin saturar los actuadores.

Las condiciones iniciales para la planta son iguales a cero para todas las articulaciones; en el caso del identificador neuronal las condiciones iniciales son iguales a 10° para todas las articulaciones. Los resultados de la simulación tanto para el filtro UKF como para el EKF con motivo de comparación, se muestran en las Figuras 4.14 a 4.69. De acuerdo con estas figuras los errores de seguimiento y de identificación para todas las articulaciones presentan un buen comportamiento y permanecen limitados como se muestra en las Figuras 4.28 a 4.55. Los pares aplicados se muestran en las Figuras 4.56 a 4.69. Se observa fácilmente, que las señales de control siempre se encuentran dentro de los límites de los actuadores $\pm\tau_1^{máx}$ a $\pm\tau_7^{máx}$, respectivamente.

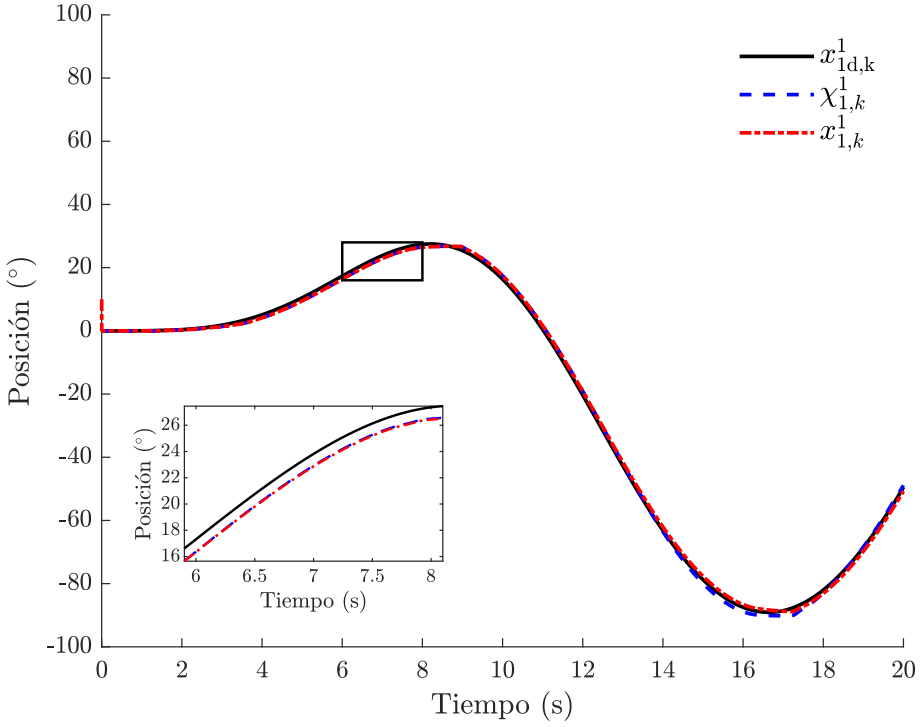


Figura 4.14: Identificación y seguimiento del eslabón 1 UKF

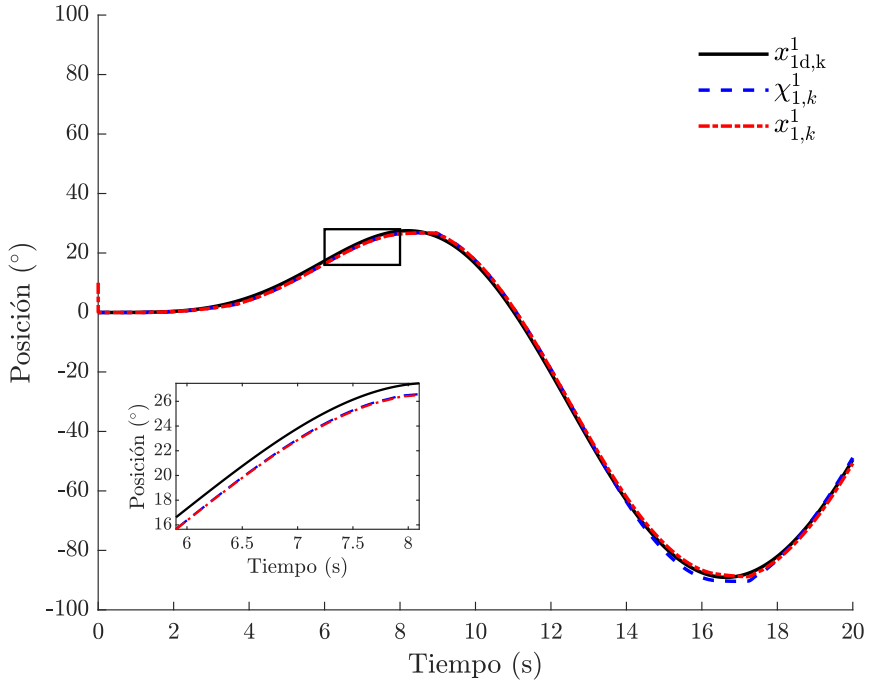


Figura 4.15: Identificación y seguimiento del eslabón 1 EKF

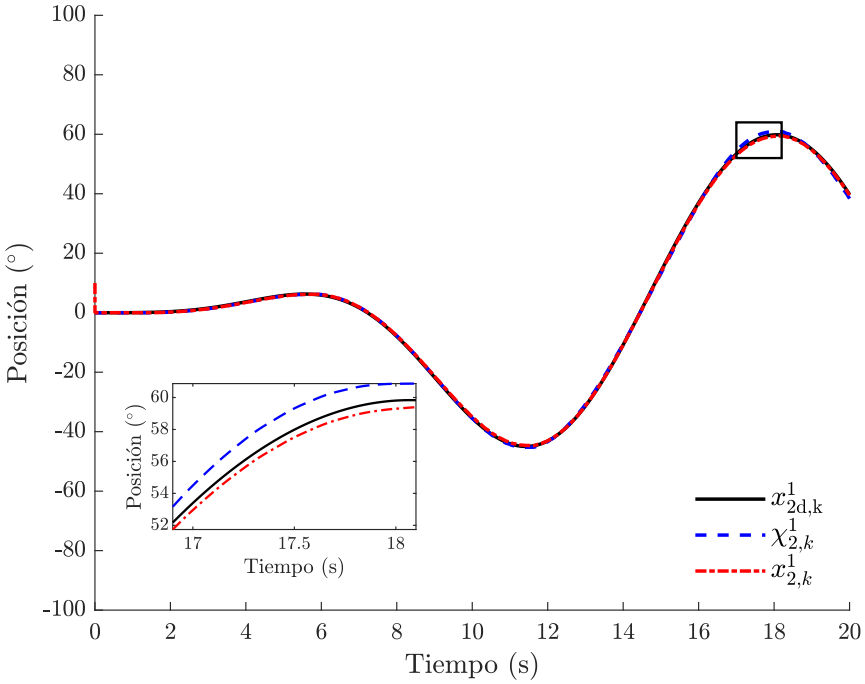


Figura 4.16: Identificación y seguimiento del eslabón 2 UKF

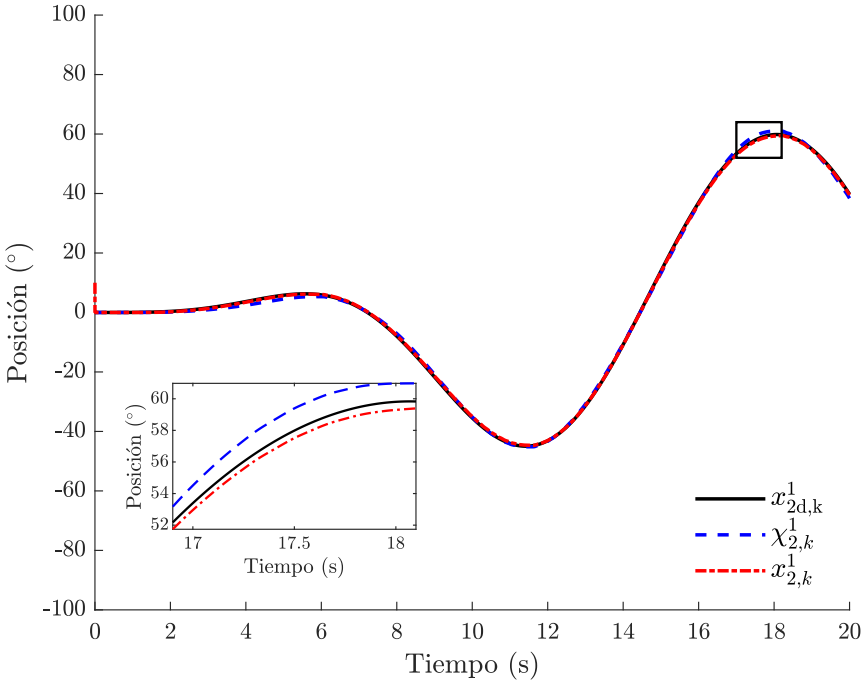


Figura 4.17: Identificación y seguimiento del eslabón 2 EKF

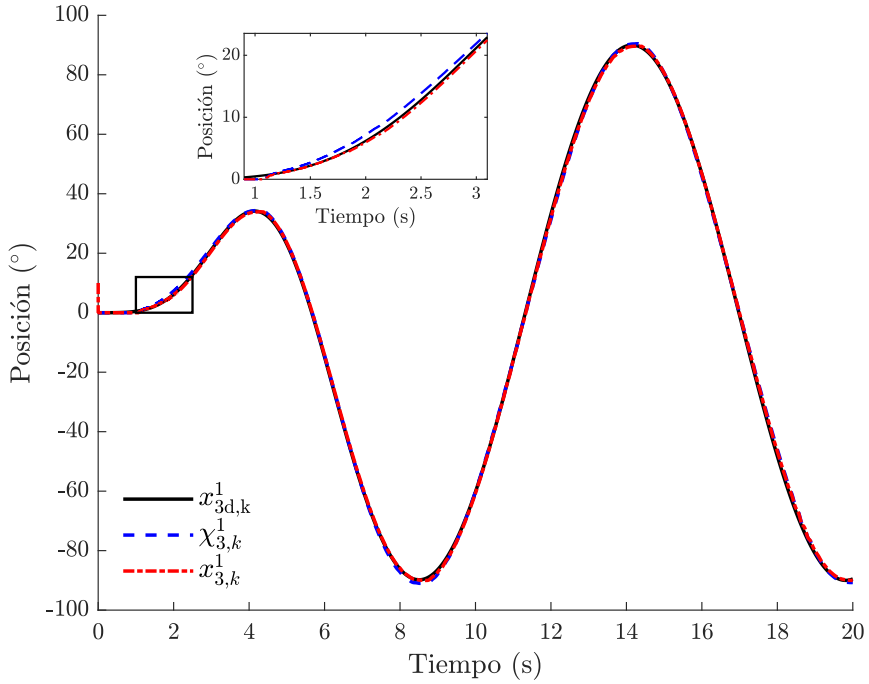


Figura 4.18: Identificación y seguimiento del eslabón 3 UKF

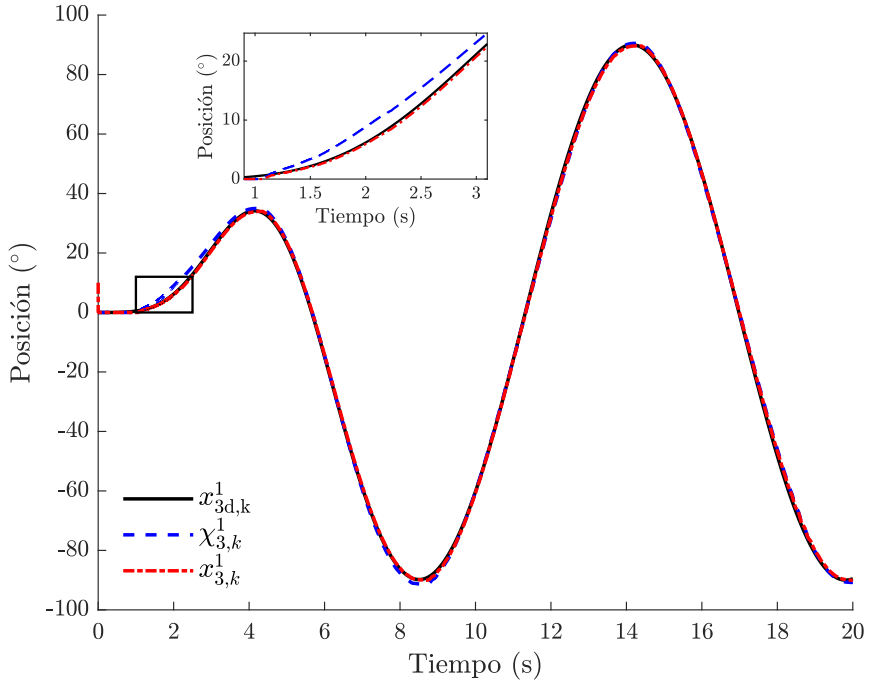


Figura 4.19: Identificación y seguimiento del eslabón 3 EKF

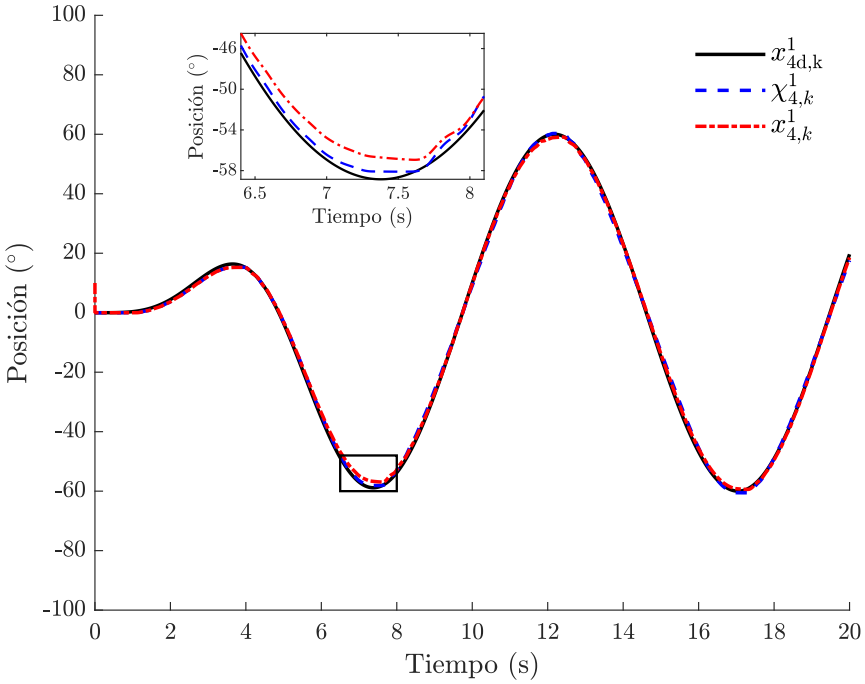


Figura 4.20: Identificación y seguimiento del eslabón 4 UKF

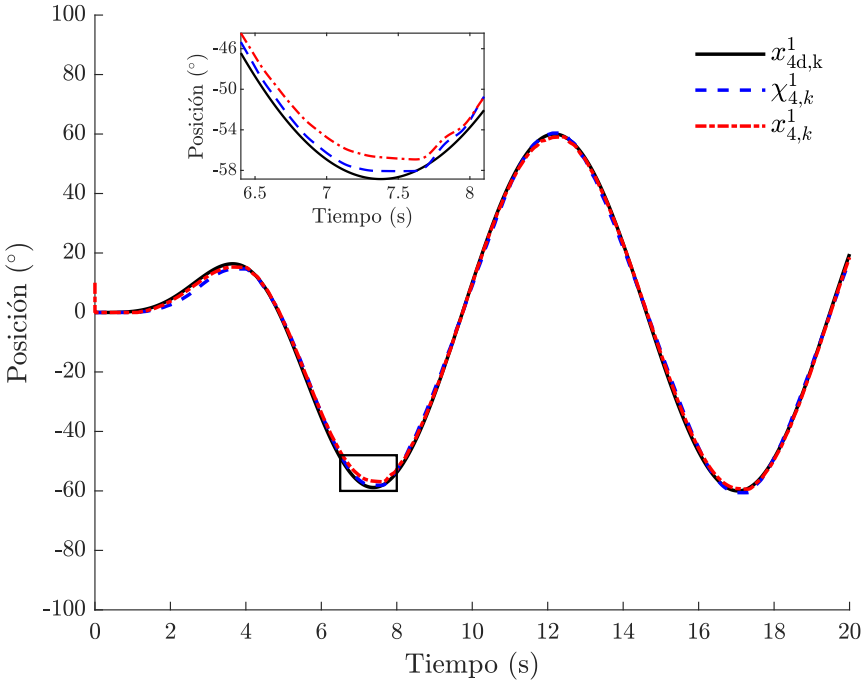


Figura 4.21: Identificación y seguimiento del eslabón 4 EKF

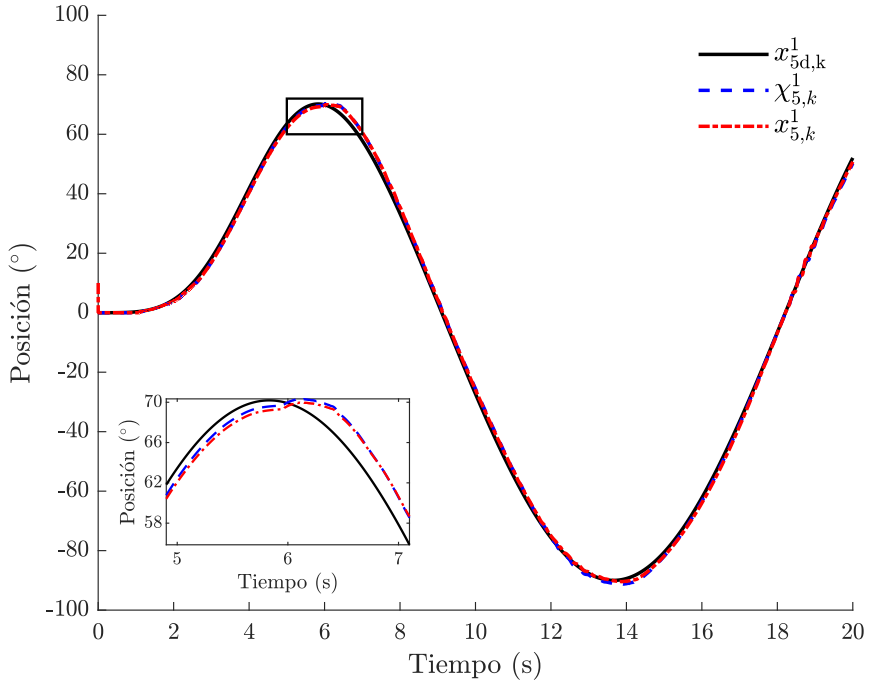


Figura 4.22: Identificación y seguimiento del eslabón 5 UKF

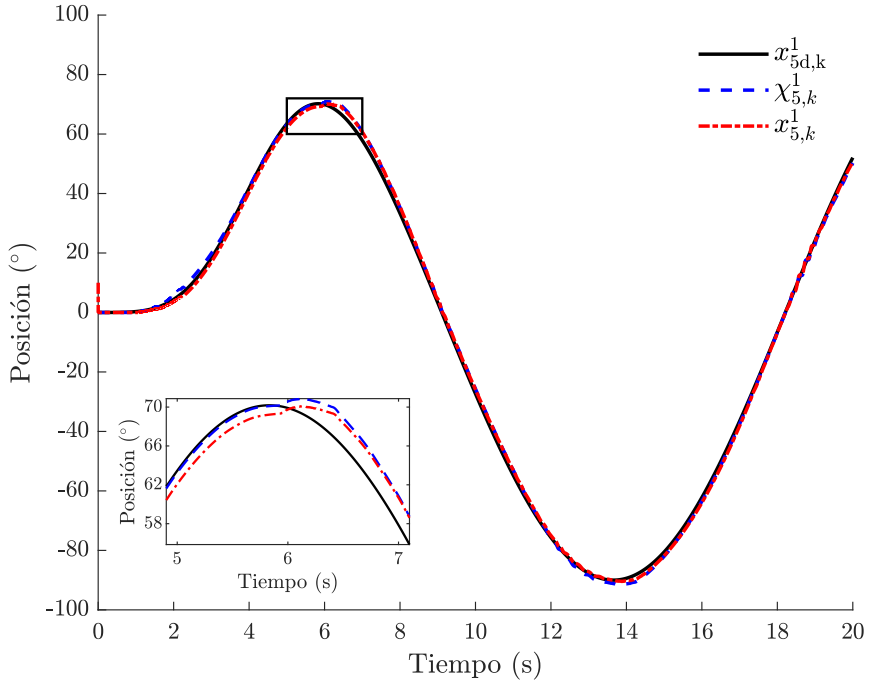


Figura 4.23: Identificación y seguimiento del eslabón 5 EKF

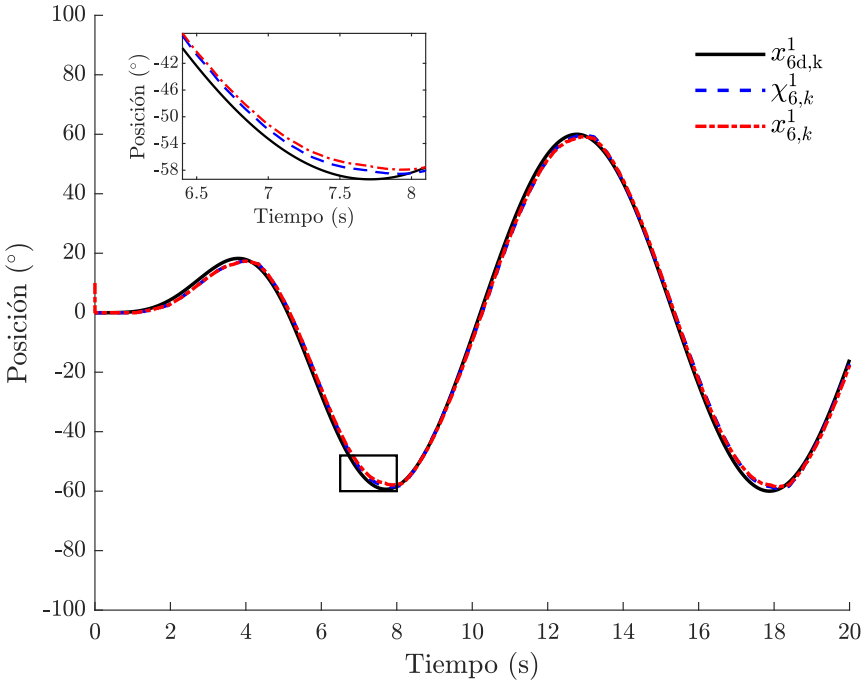


Figura 4.24: Identificación y seguimiento del eslabón 6 UKF

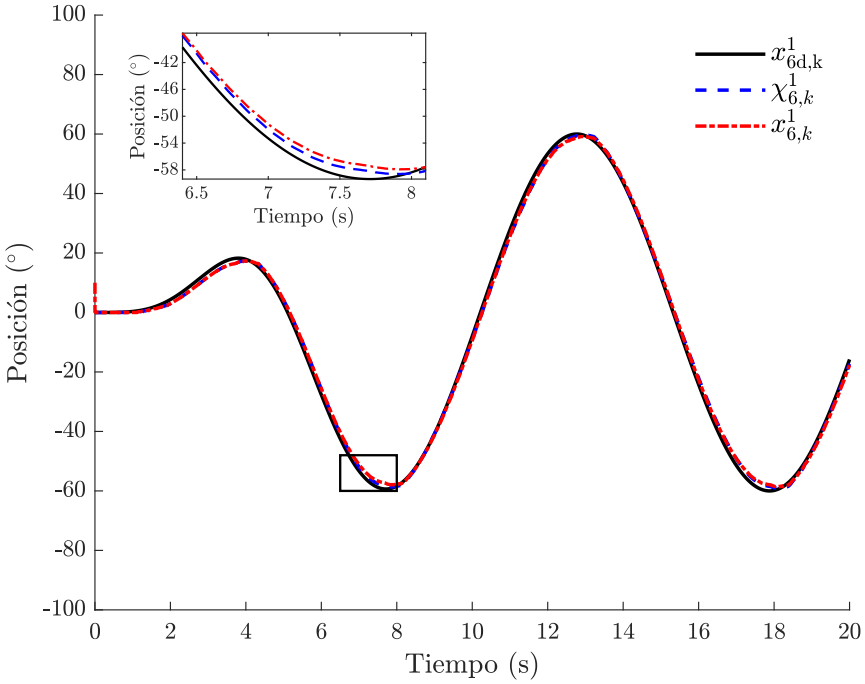


Figura 4.25: Identificación y seguimiento del eslabón 6 EKF

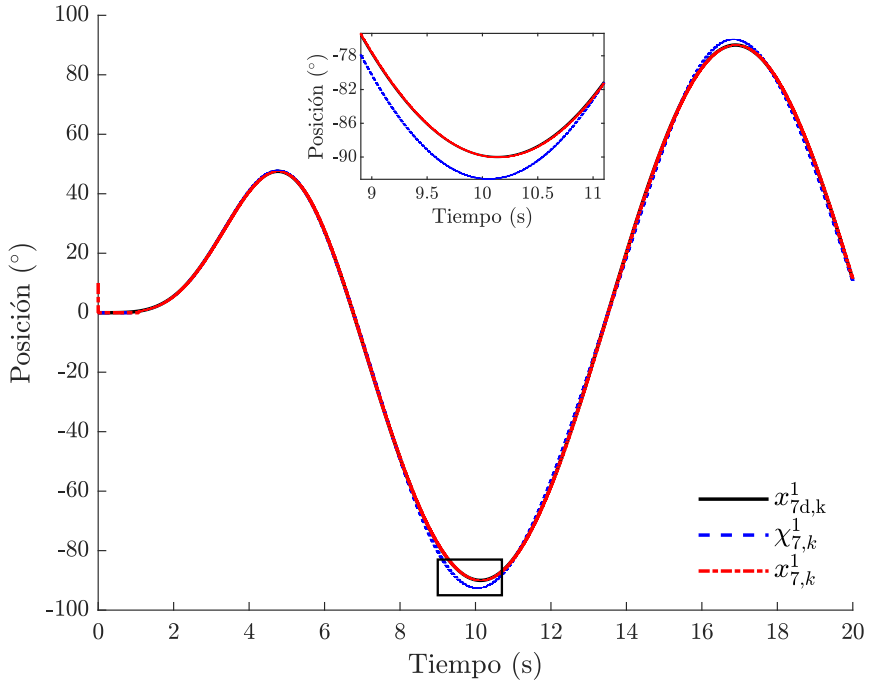


Figura 4.26: Identificación y seguimiento del eslabón 7 UKF

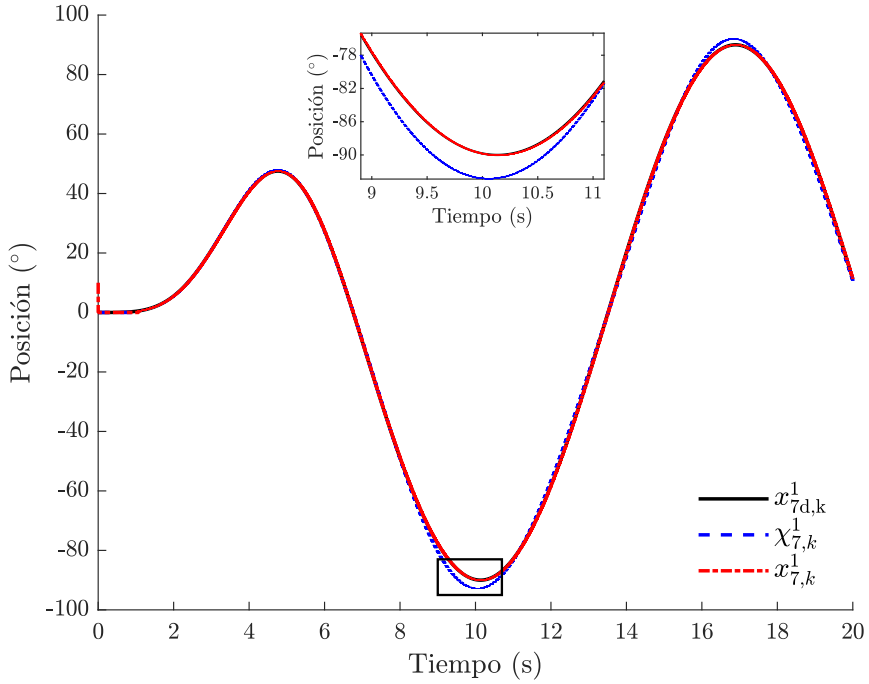


Figura 4.27: Identificación y seguimiento del eslabón 7 EKF

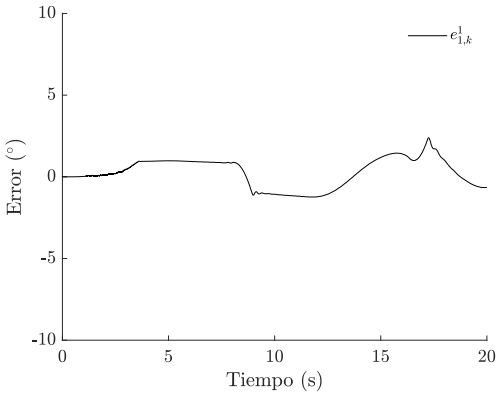


Figura 4.28: Error de seguimiento del eslabón 1 UKF

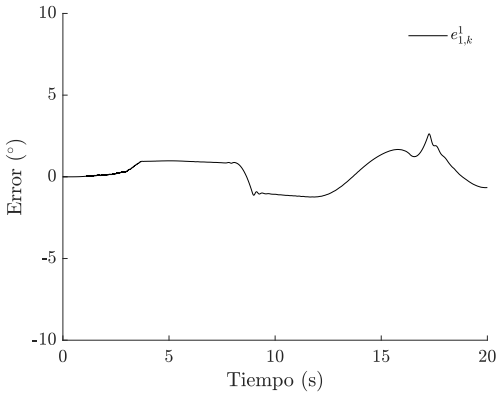


Figura 4.29: Error de seguimiento del eslabón 1 EKF

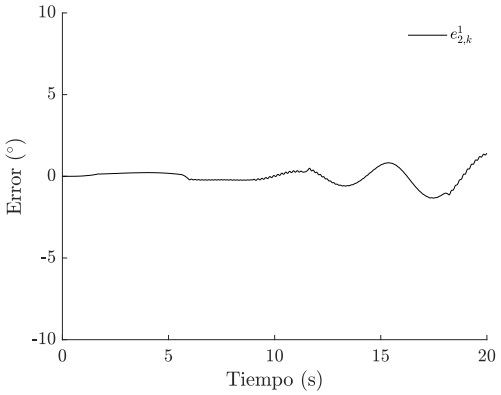


Figura 4.30: Error de seguimiento del eslabón 2 UKF

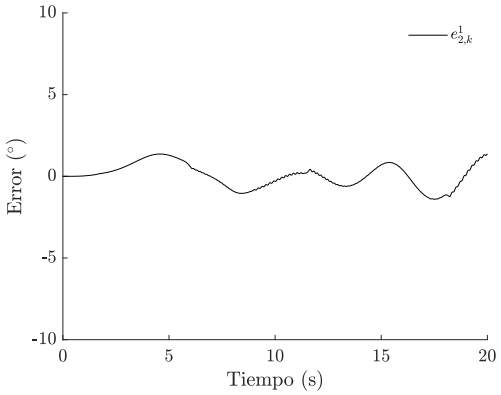


Figura 4.31: Error de seguimiento del eslabón 2 EKF

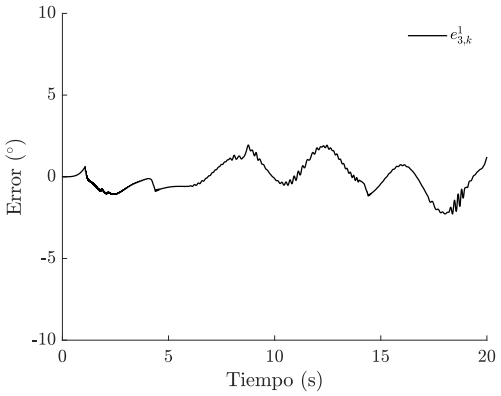


Figura 4.32: Error de seguimiento del eslabón 3 UKF

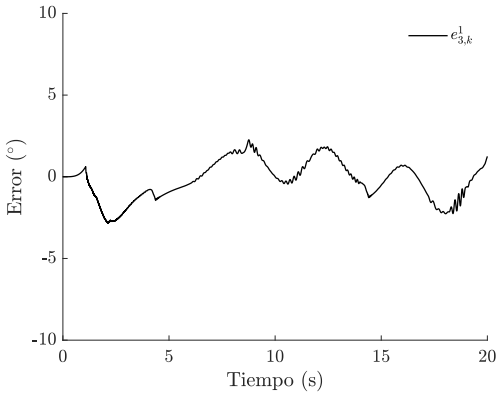


Figura 4.33: Error de seguimiento del eslabón 3 EKF

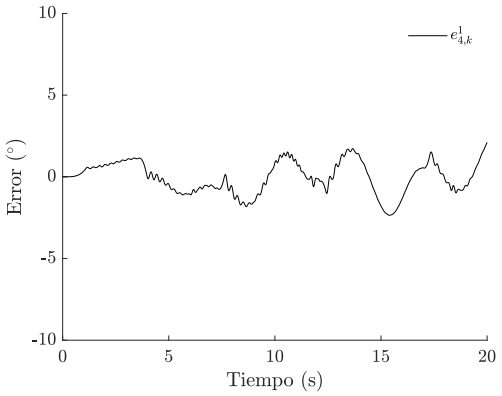


Figura 4.34: Error de seguimiento del eslabón 4 UKF

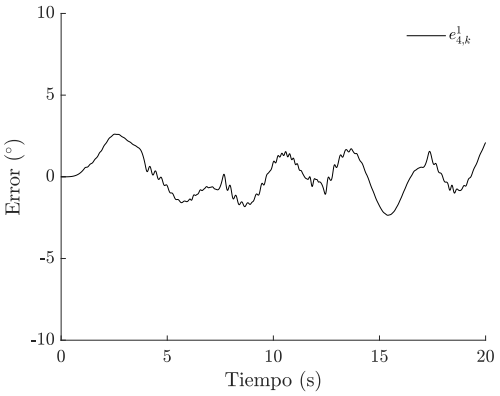


Figura 4.35: Error de seguimiento del eslabón 4 EKF

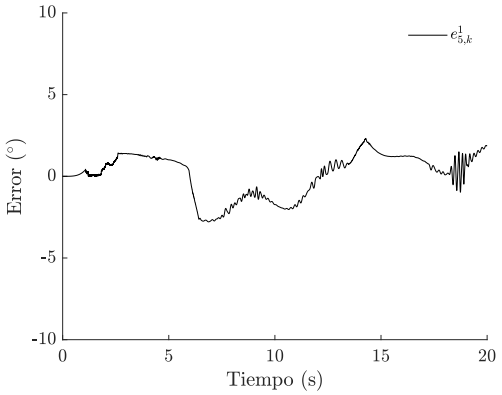


Figura 4.36: Error de seguimiento del eslabón 5 UKF

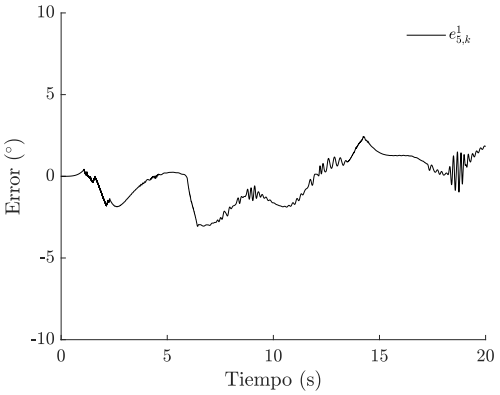


Figura 4.37: Error de seguimiento del eslabón 5 EKF

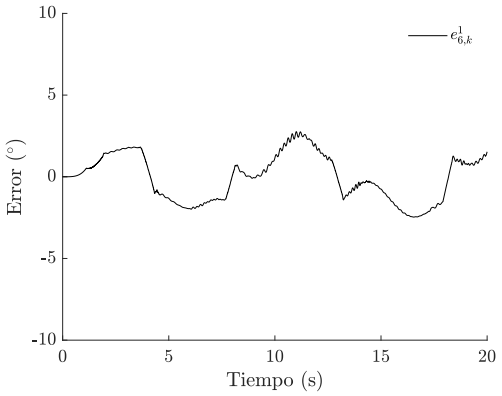


Figura 4.38: Error de seguimiento del eslabón 6 UKF

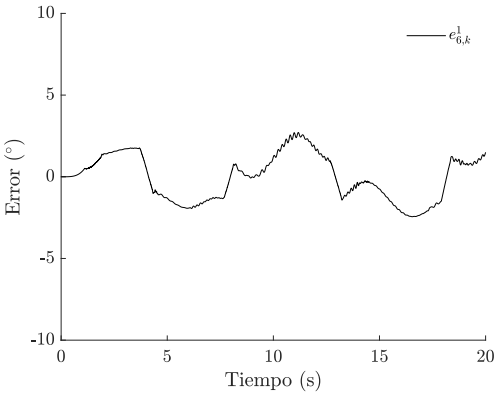


Figura 4.39: Error de seguimiento del eslabón 6 EKF

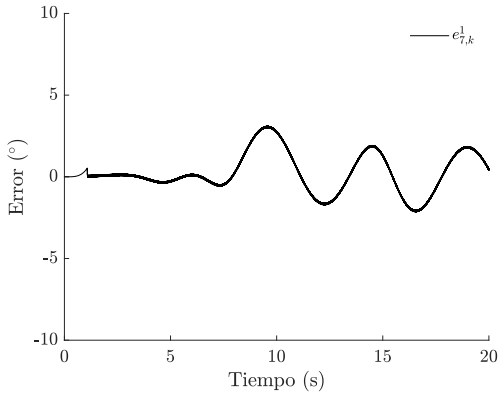


Figura 4.40: Error de seguimiento del eslabón 7 UKF

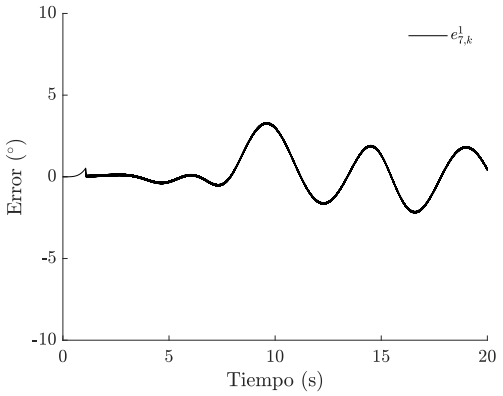


Figura 4.41: Error de seguimiento del eslabón 7 EKF

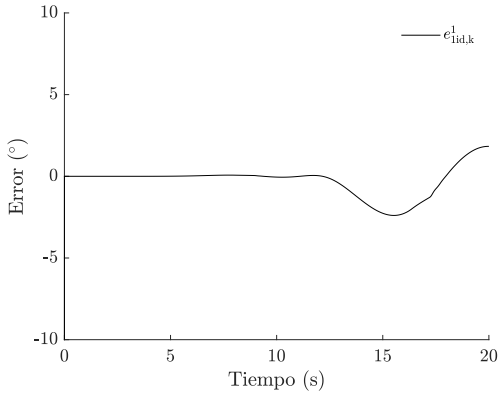


Figura 4.42: Error de identificación del eslabón 1 UKF

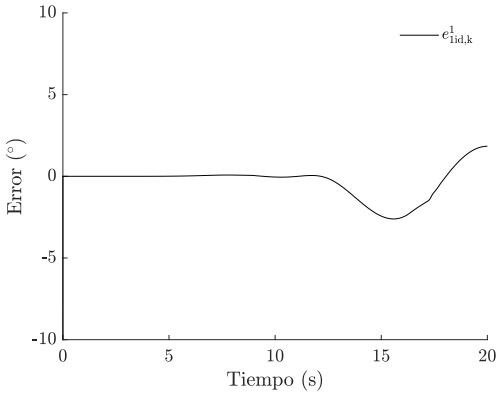


Figura 4.43: Error de identificación del eslabón 1 EKF

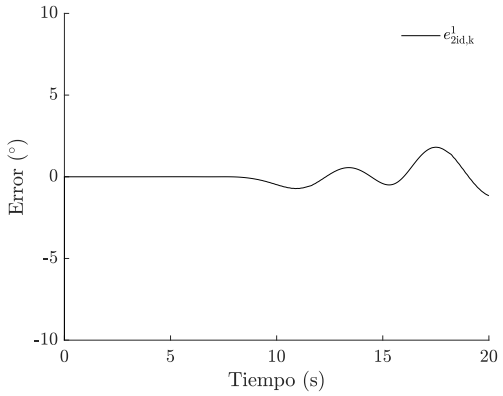


Figura 4.44: Error de identificación del eslabón 2 UKF

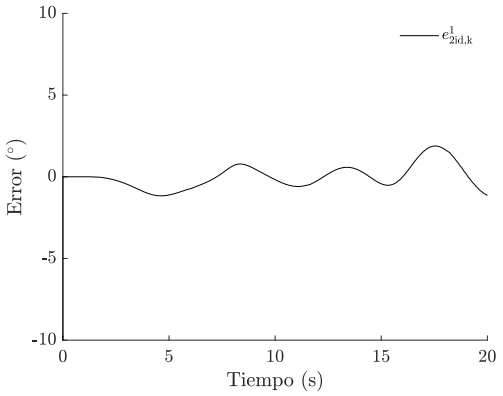


Figura 4.45: Error de identificación del eslabón 2 EKF

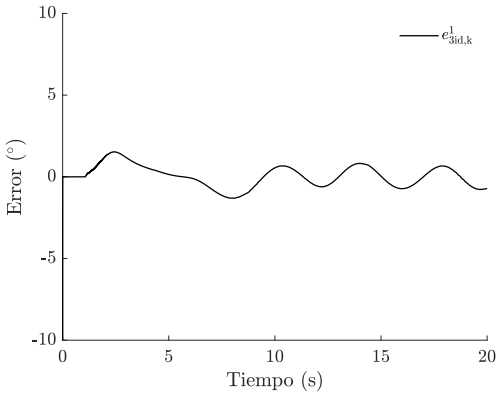


Figura 4.46: Error de identificación del eslabón 3 UKF

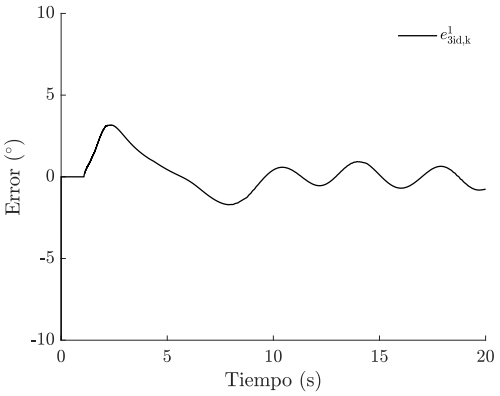


Figura 4.47: Error de identificación del eslabón 3 EKF

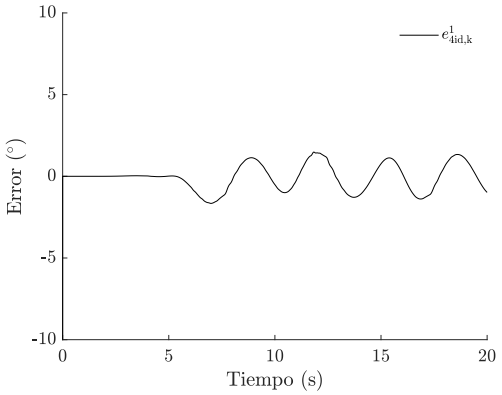


Figura 4.48: Error de identificación del eslabón 4 UKF

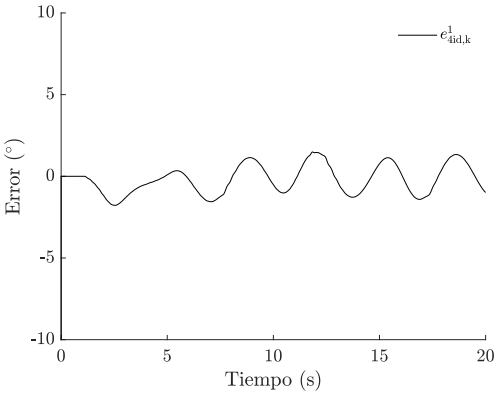


Figura 4.49: Error de identificación del eslabón 4 EKF

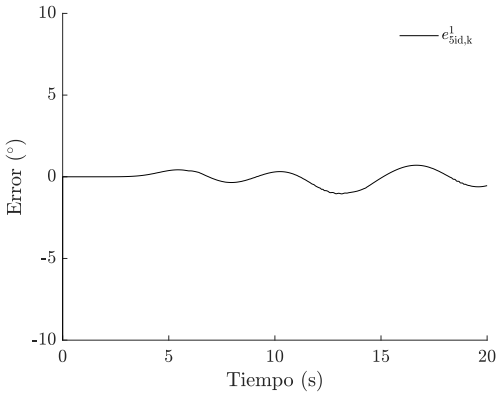


Figura 4.50: Error de identificación del eslabón 5 UKF

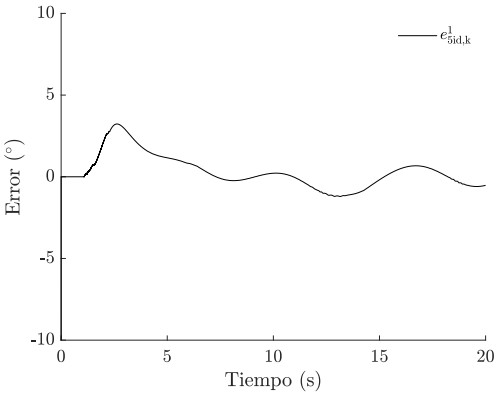


Figura 4.51: Error de identificación del eslabón 5 EKF

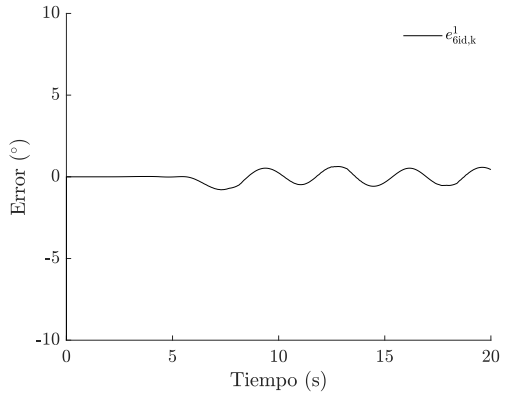


Figura 4.52: Error de identificación del eslabón 6 UKF

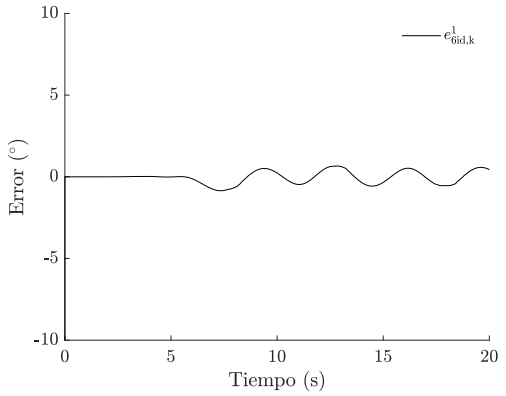


Figura 4.53: Error de identificación del eslabón 6 EKF

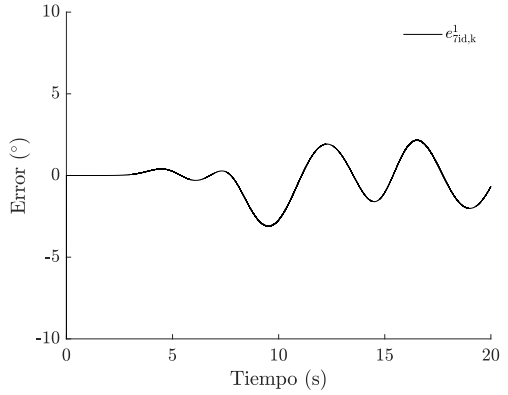


Figura 4.54: Error de identificación del eslabón 7 UKF

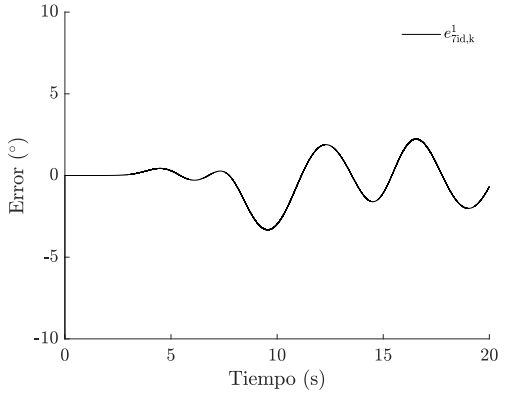


Figura 4.55: Error de identificación del eslabón 7 EKF

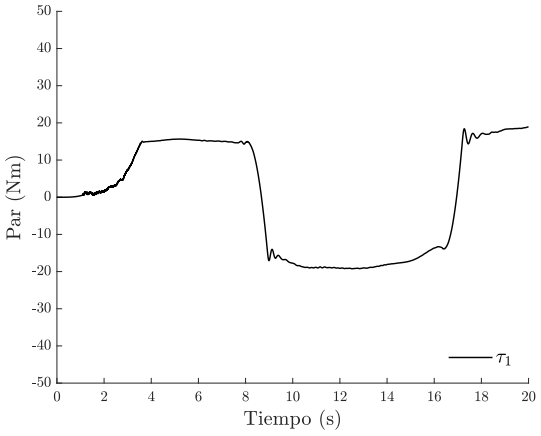


Figura 4.56: Pares aplicados al eslabón 1 UKF

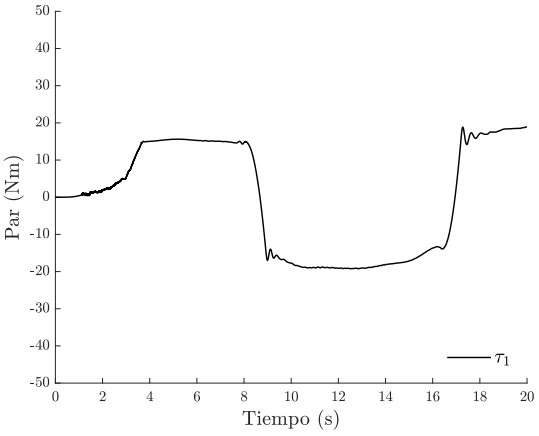


Figura 4.57: Pares aplicados al eslabón 1 EKF

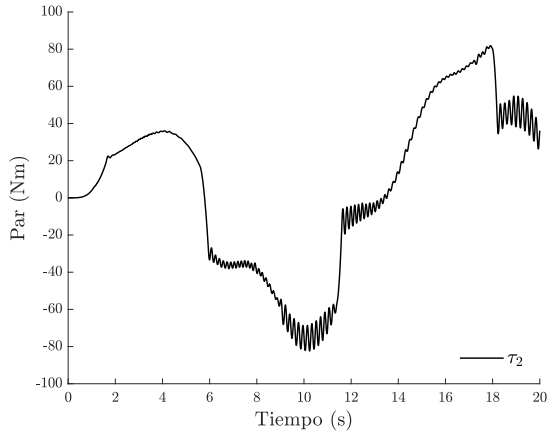


Figura 4.58: Pares aplicados al eslabón 2 UKF

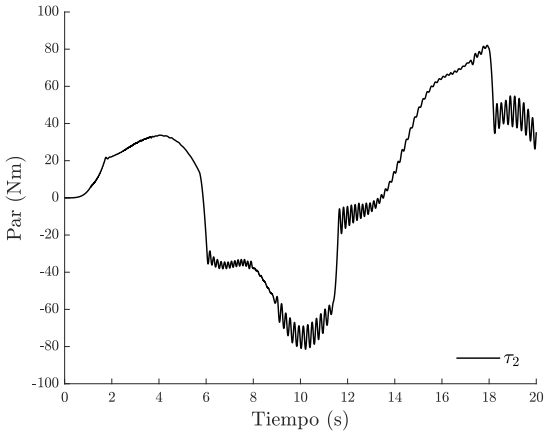


Figura 4.59: Pares aplicados al eslabón 2 EKF

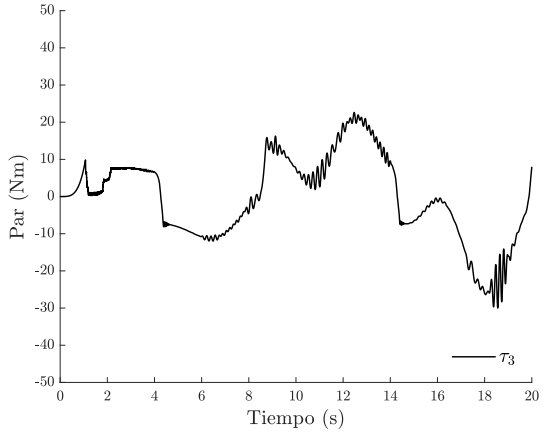


Figura 4.60: Pares aplicados al eslabón 3 UKF

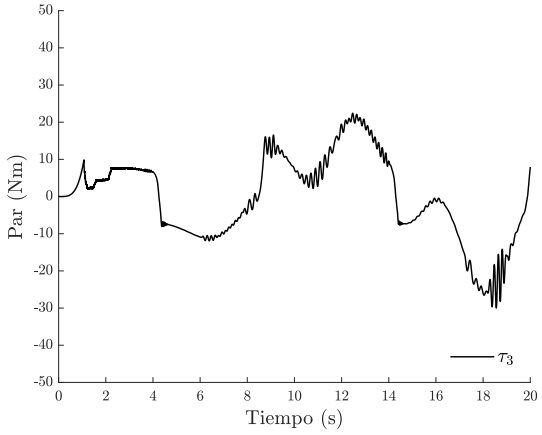


Figura 4.61: Pares aplicados al eslabón 3 EKF

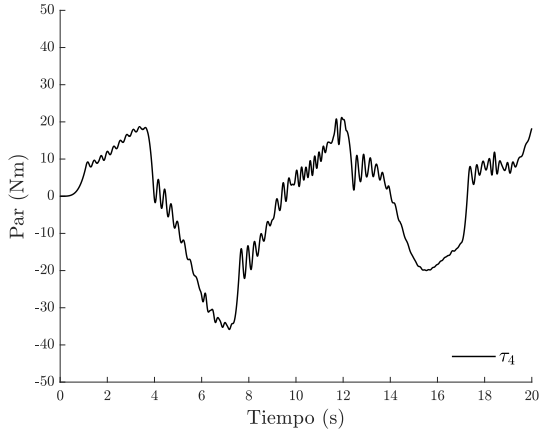


Figura 4.62: Pares aplicados al eslabón 4 UKF

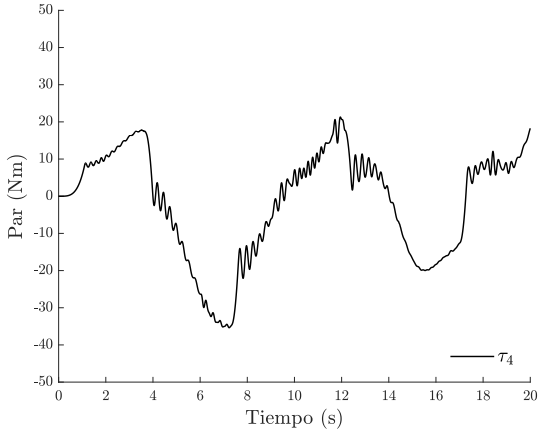


Figura 4.63: Pares aplicados al eslabón 4 EKF

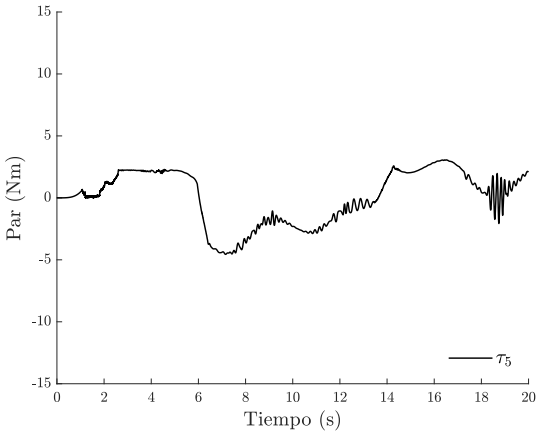


Figura 4.64: Pares aplicados al eslabón 5 UKF

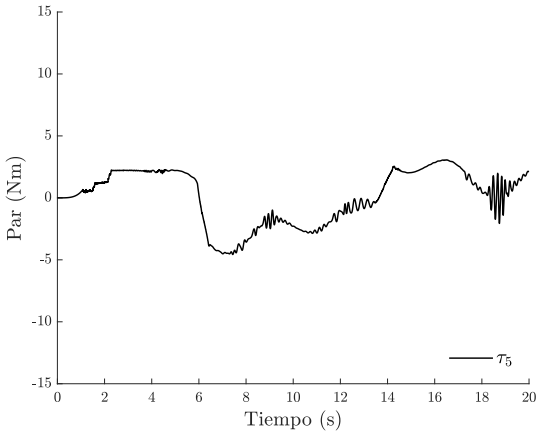


Figura 4.65: Pares aplicados al eslabón 5 EKF

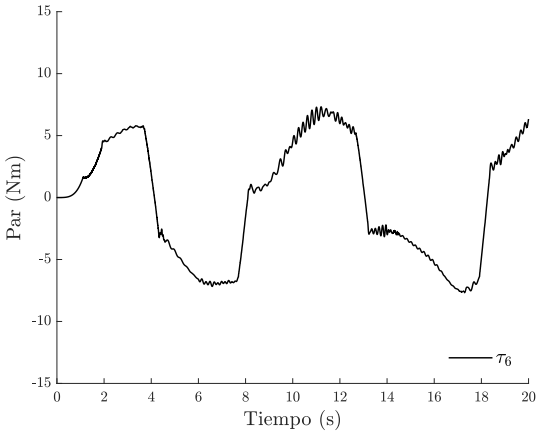


Figura 4.66: Pares aplicados al eslabón 6 UKF

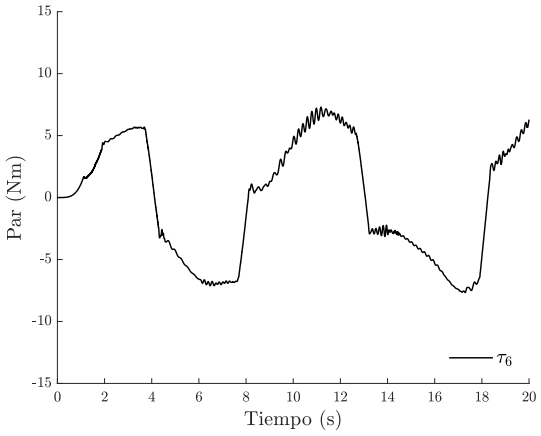


Figura 4.67: Pares aplicados al eslabón 6 EKF

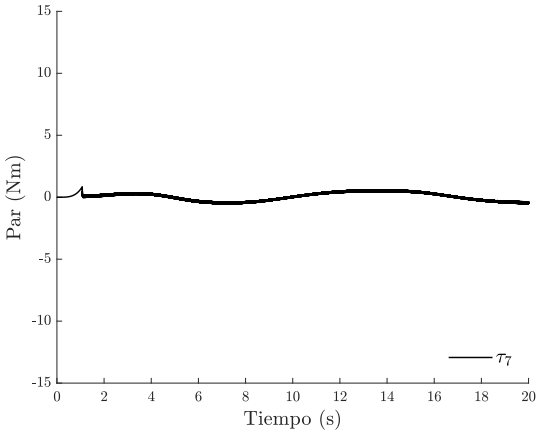


Figura 4.68: Pares aplicados al eslabón 7 UKF

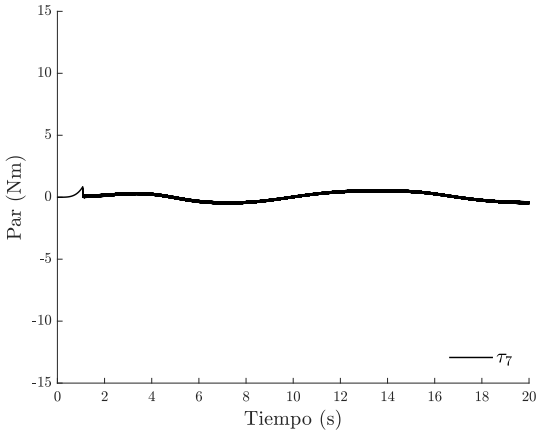


Figura 4.69: Pares aplicados al eslabón 7 EKF

Como se observa en las Figuras 4.28 a 4.43, los errores de identificación y seguimiento son relativamente pequeños para la mayoría de los eslabones, esto es debido a el buen desempeño de las RHONN para identificar el modelo no lineal que describe el comportamiento del brazo robótico industrial.

También como en el caso del robot de 2 g.d.l., se analizará y comparará el desempeño de los algoritmos propuestos por medio del MSE (4.17). Los resultados se muestran en la Tabla 4.6.

Tabla 4.6: MSE del error de seguimiento para el brazo robótico Mitsubishi PA10-7CE

| | Algoritmo | |
|---|---|---|
| Eslabón | UKF | EKF |
| 1 | 0.9239° | 0.9787° |
| 2 | 0.5246° | 0.7290° |
| 3 | 0.9143° | 1.1950° |
| 4 | 0.9327° | 1.1690° |
| 5 | 1.3090° | 1.3590° |
| 6 | 1.3730° | 1.3590° |
| 7 | 1.2130° | 1.2630° |

Al igual que para el robot manipulador de 2 g.d.l., los resultados de simulación obtenidos muestran que el entrenamiento UKF tiene un rendimiento mejor para la mayoría de las articulaciones; además es más fácil de implementar. Es importante aclarar que los parámetros de inicialización $Q_0$, $R_0$, $P_0$, $w_{ji}(0)$, y los valores de los pesos fijos $w_i^{'r}$ son los mismos para ambos esquemas de control.

## 4.3. Resultados experimentales

La evaluación experimental de los identificadores y controladores neuronales, previamente descritos en el capítulo 3, se realizaron en el robot manipulador de 2 g.d.l. que se encuentra en el laboratorio de Mecatrónica y Control perteneciente a la División de Estudios de Posgrado e Investigación del Instituto Tecnológico de La Laguna.

Los algoritmos de identificación y de control se programaron en el entorno de desarrollo integrado (*Integrated Development Environment* IDE) WinMechLab 2.0 desarrollado por [Campa and Kelly, 2008]. Previo a la evaluación experimental, se realizaron nuevamente simulaciones, por medio de esta interfaz de los controladores neuronales (ver Apéndice C), que ya habían sido simulados en Simulink en la sección 4.2.

Cabe señalar que todas las pruebas experimentales se realizaron con un periodo de muestreo de 2.5 ms; antes de realizar los experimentos fue necesario alinear los eslabones del robot manipulador y cargar el archivo de *demo2.rob* para comprobar el estado de los actuadores y la comunicación por medio de un controlador par calculado para seguimiento propuesto en [Kelly and Santibáñez, 2003].

El tiempo total de ejecución para cada algoritmo fue de 60 segundos y las trayectorias discretas seleccionas fueron las mismas que se utilizaron en la simulación con los mismos parámetros. Como ya se mencionó en este capítulo, se eligieron estas trayectorias, porque permiten mantener el robot en un estado de operación exigente sin saturar los actuadores (consignas de velocidades y pares mostradas en [Márquez, 1995]).

### 4.3.1. Control descentralizado neuronal con entrenamiento UKF

A continuación se muestran los resultados del control neuronal descentralizado con entrenamiento UKF, para lo cual fue necesario convertir, compilar y simular, la simulación previa hecha en Simulink, en el entorno de WinMechLab para su ejecución en tiempo real; el programa, así como sus parámetros se encuentran en el Apéndice C.2.

En las Figuras 4.70 y 4.71 ilustran los resultados de la identificación y el seguimiento de las trayectorias; los errores de seguimiento se presentan en las Figuras 4.72 y 4.73; en las Figuras 4.74 y 4.75 se muestran los errores de identificación; y los pares aplicados en las Figuras 4.76 y 4.77.

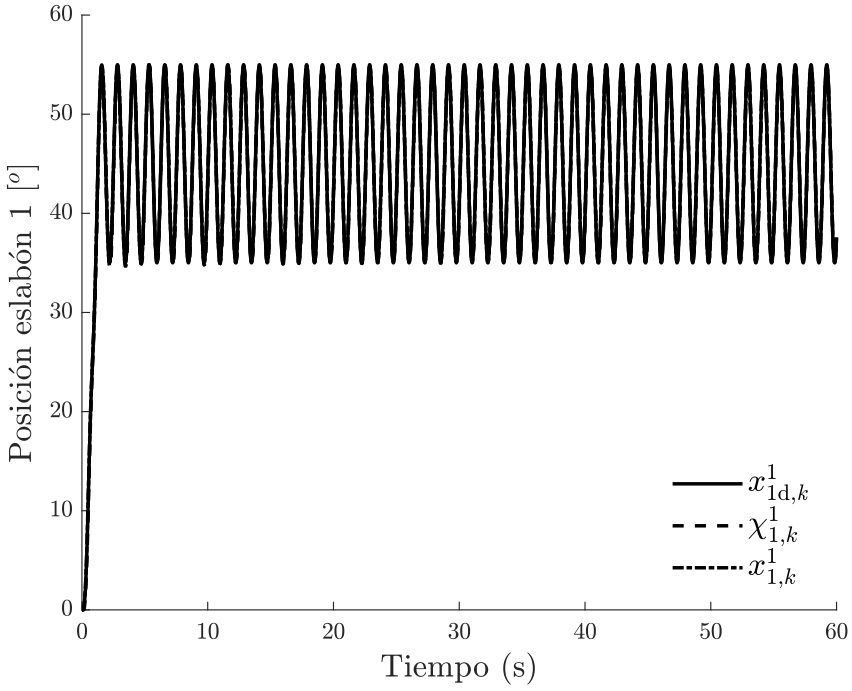


Figura 4.70: Identificación y seguimiento del eslabón 1 con aprendizaje UKF

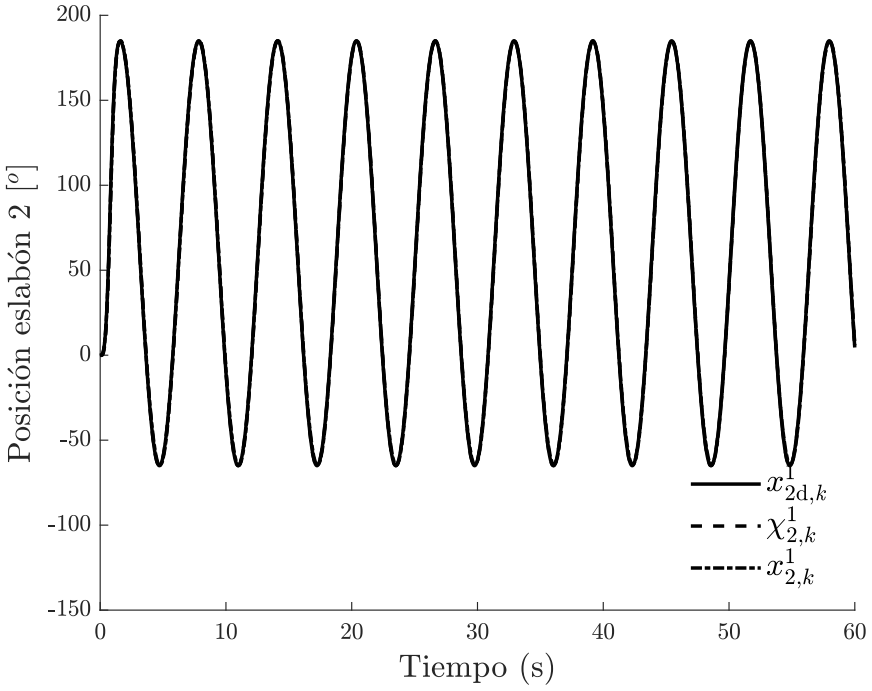


Figura 4.71: Identificación y seguimiento del eslabón 2 con aprendizaje UKF

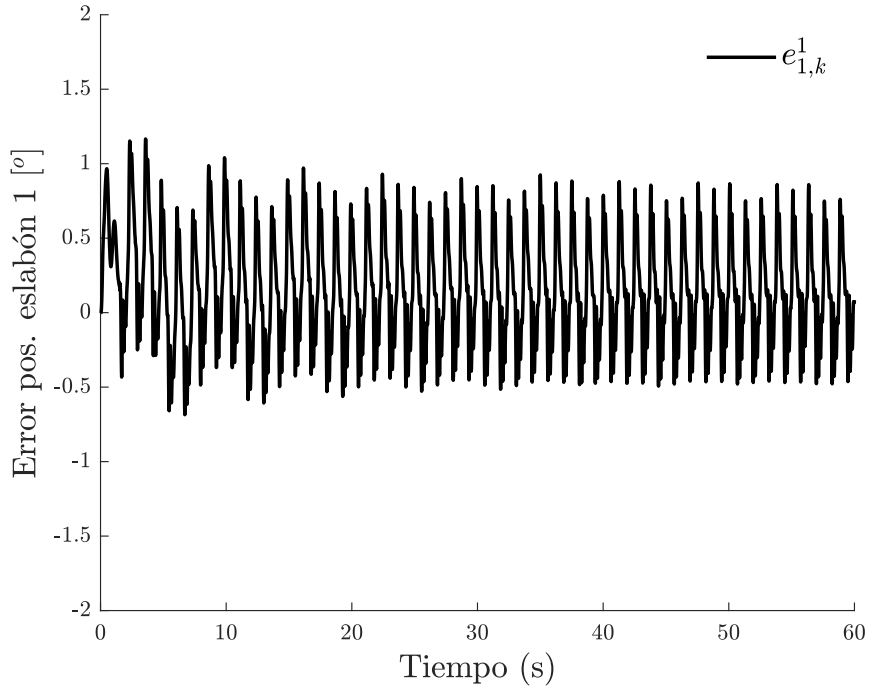


Figura 4.72: Error de seguimiento del eslabón 1 con aprendizaje UKF

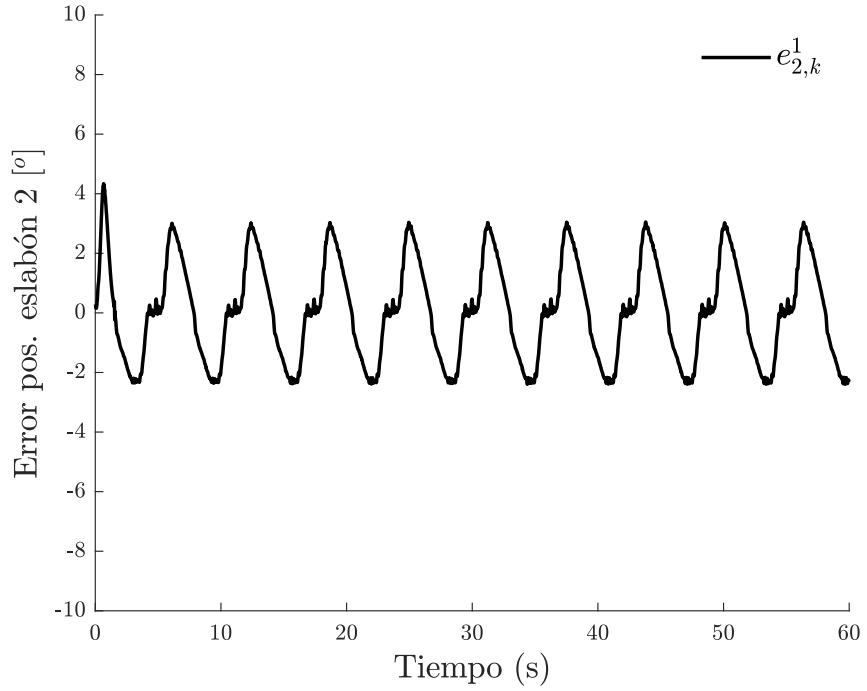


Figura 4.73: Error de seguimiento del eslabón 2 con aprendizaje UKF

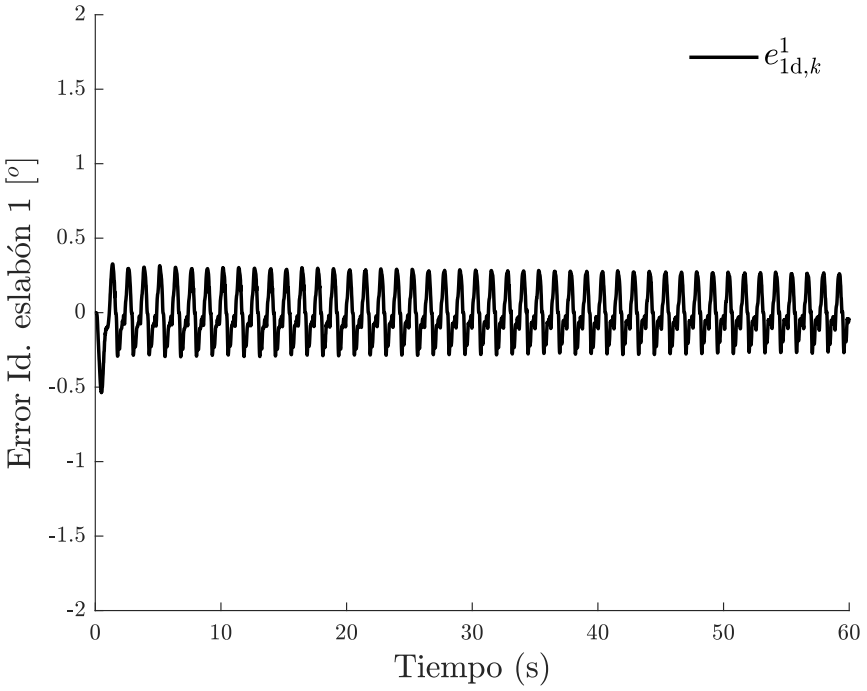


Figura 4.74: Error de identificación del eslabón 1 con aprendizaje UKF

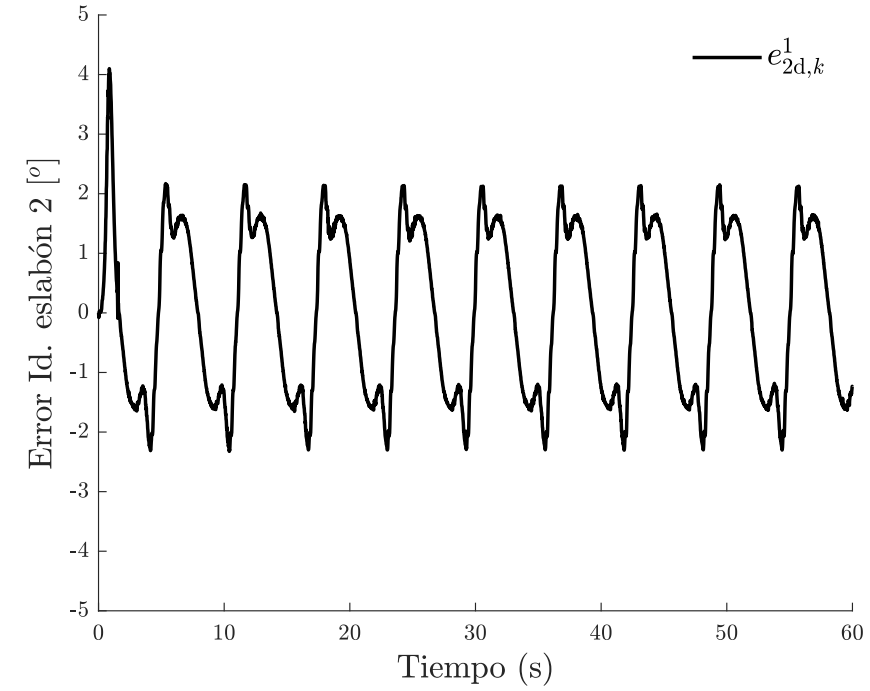


Figura 4.75: Error de identificación del eslabón 2 con aprendizaje UKF

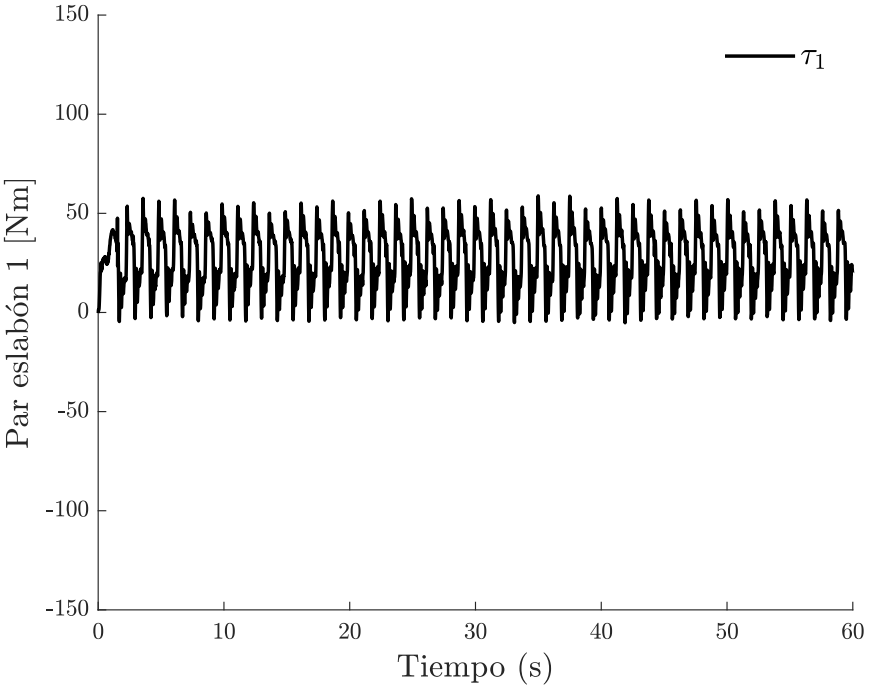


Figura 4.76: Par aplicado al eslabón 1 con aprendizaje UKF

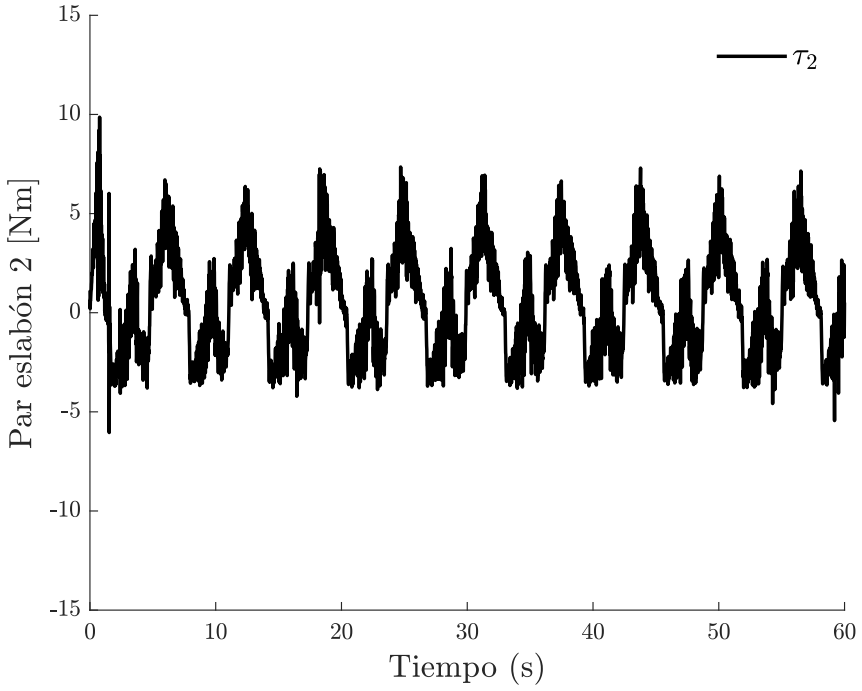


Figura 4.77: Par aplicado al eslabón 2 con aprendizaje UKF

### 4.3.2. Control descentralizado neuronal con entrenamiento EKF

De igual manera que en la sección anterior, a continuación se muestran los resultados del control descentralizado neuronal con entrenamiento EKF, además, el programa para su ejecución en tiempo real se incluye en el Apéndice C.3.

Las Figuras 4.78 y 4.79 ilustran los resultados de la identificación y el seguimiento de las trayectorias. Los errores de seguimiento que se presentan en las Figuras 4.80 y 4.81 son mayores que en el entrenamiento UKF de las Figuras 4.72 y 4.73.

Por otro lado, los errores de identificación que se encuentran en las Figuras 4.82 y 4.83 son menores a los mostrados en las Figuras 4.74 y 4.75 para el entrenamiento UKF, esto se a que los valores que se utilizaron en ambos filtros fueron los que se obtuvieron en la sintonización del filtro EKF. Los pares aplicados están representados en las Figuras 4.84 y 4.85, y de la misma manera que para el entrenamiento UKF en las Figuras 4.76 y 4.77 no saturan, en ningún momento, los actuadores.

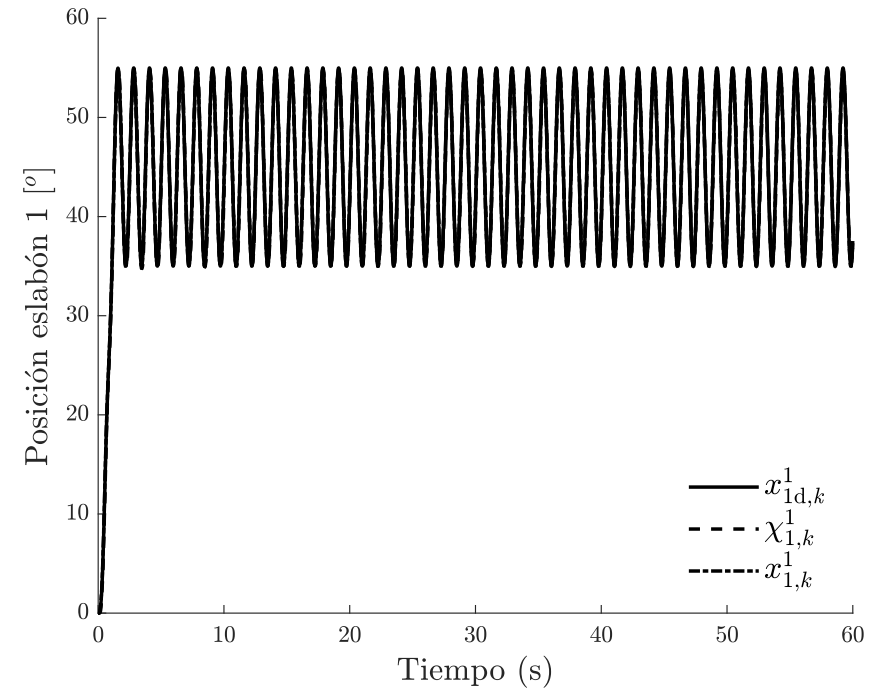


Figura 4.78: Identificación y seguimiento del eslabón 1 con aprendizaje EKF

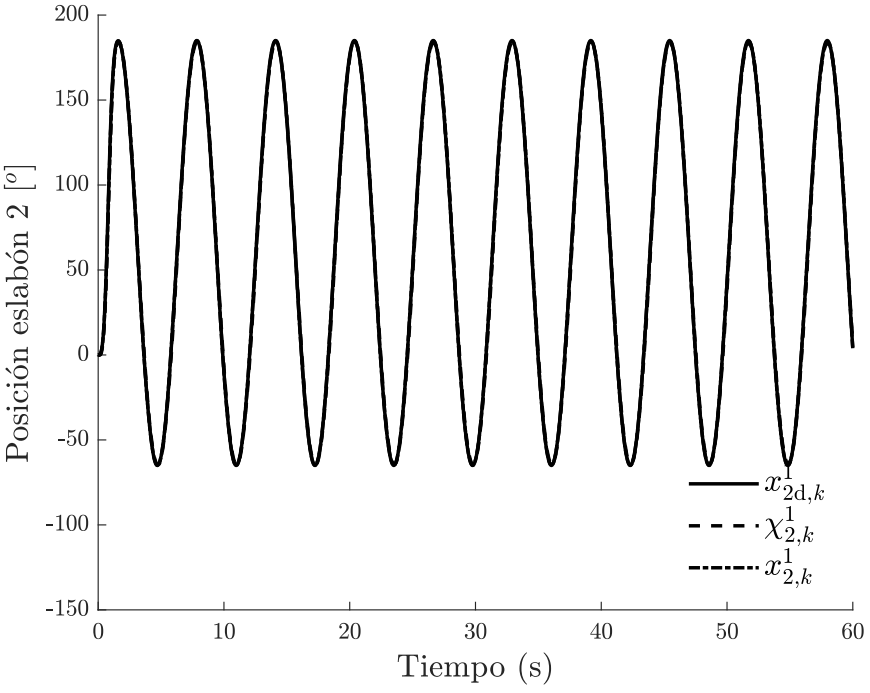


Figura 4.79: Identificación y seguimiento del eslabón 2 con aprendizaje EKF

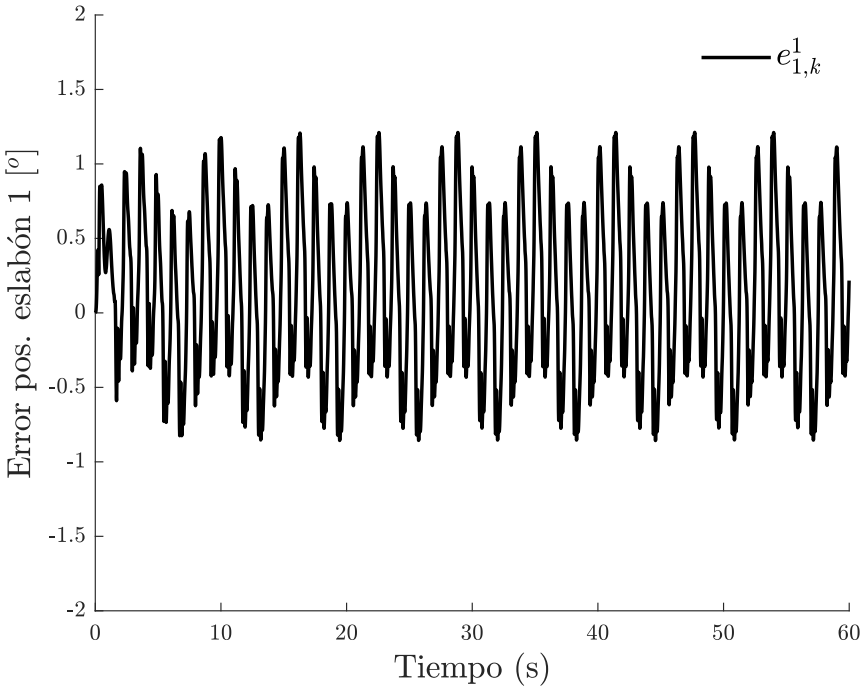


Figura 4.80: Error de seguimiento del eslabón 1 con aprendizaje EKF

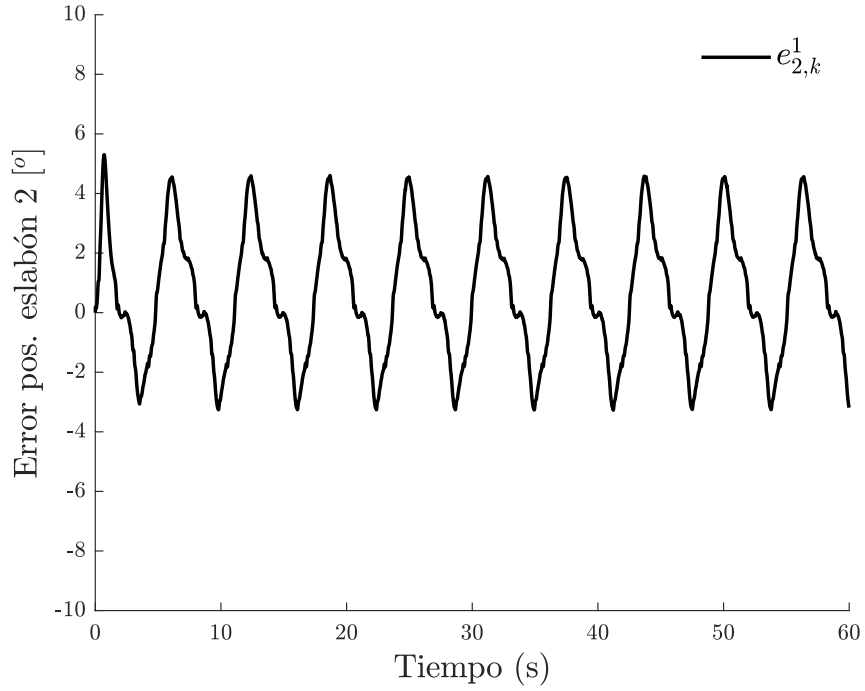


Figura 4.81: Error de seguimiento del eslabón 2 con aprendizaje EKF

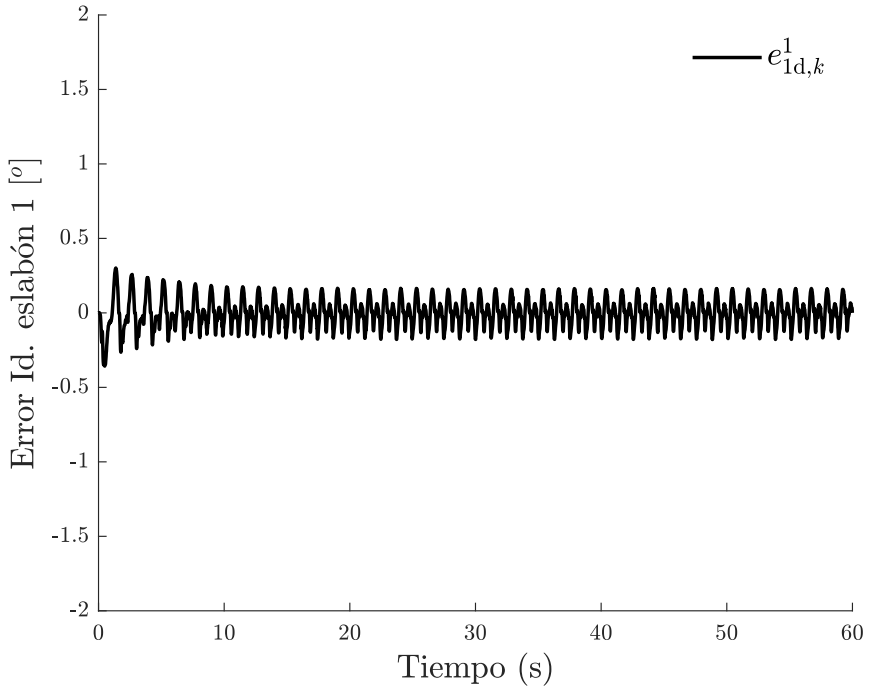


Figura 4.82: Error de identificación del eslabón 1 con aprendizaje EKF

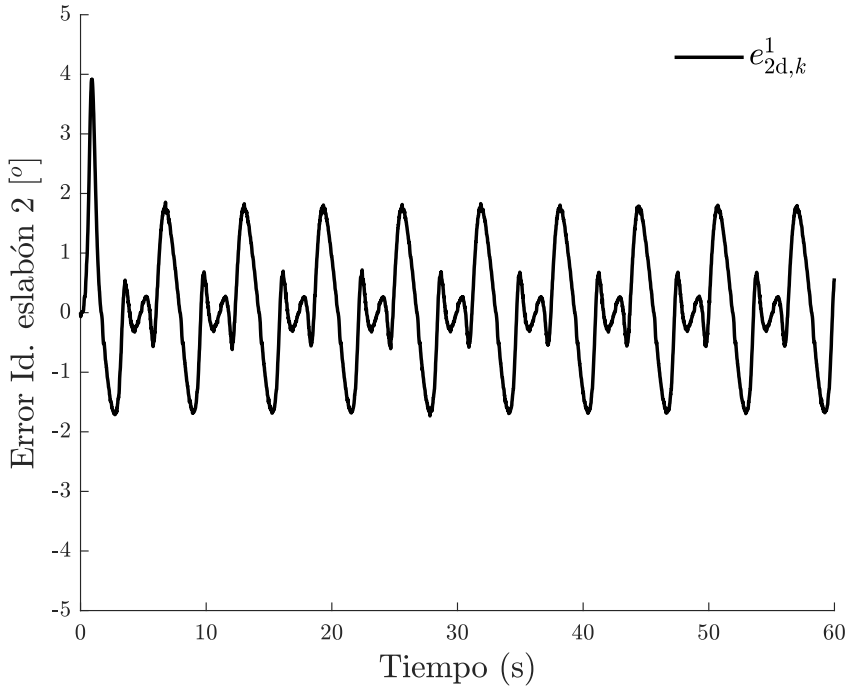


Figura 4.83: Error de identificación del eslabón 2 con aprendizaje EKF

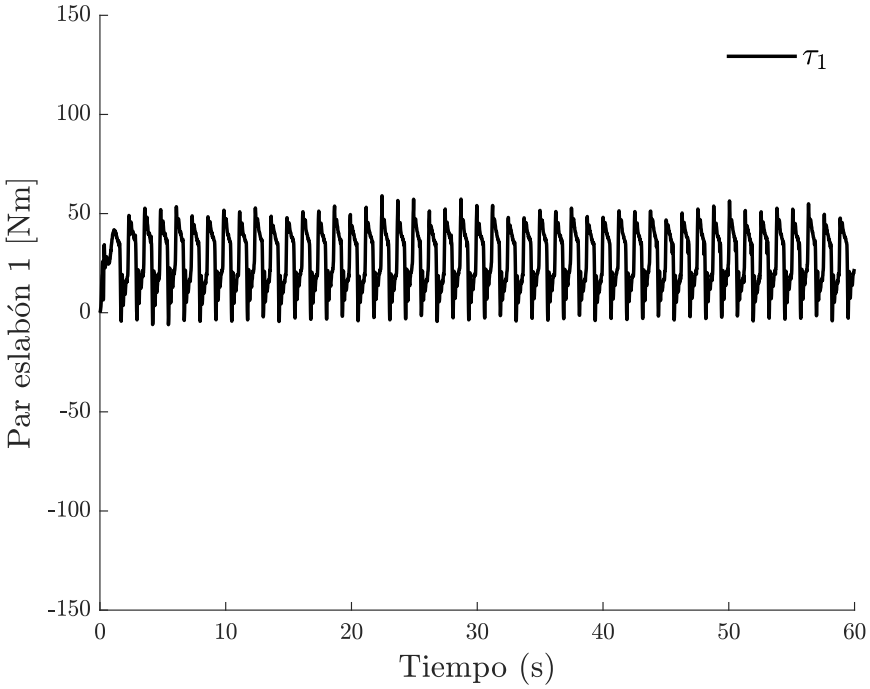


Figura 4.84: Par aplicado la eslabón 1 con aprendizaje EKF

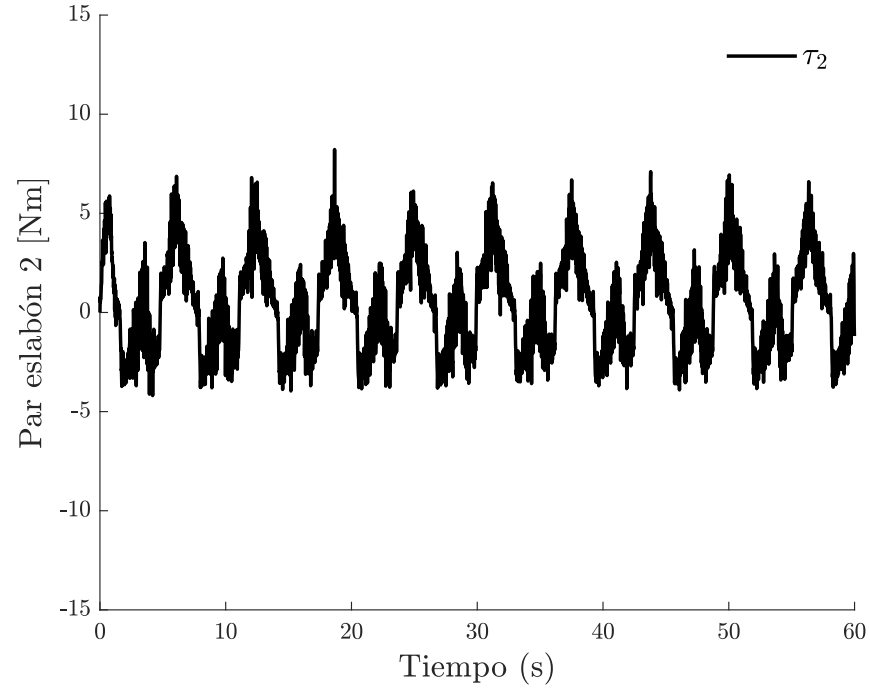


Figura 4.85: Par aplicado al eslabón 2 con aprendizaje EKF

Por último, en la Tabla 4.7 aparecen los resultados del desempeño de los controladores por medio del MSE (4.17).

Tabla 4.7: MSE del error de seguimiento para el manipulador robótico de 2 g.d.l.

| | Algoritmo | |
|---|---|---|
| Eslabón | UKF | EKF |
| 1 | 0.4555° | 0.5539° |
| 2 | 1.7332° | 2.3388° |

# Capítulo 5

# Conclusiones y Trabajo futuro

Las extensiones del filtro de Kalman no lineales EKF y UKF han sido abordadas en este trabajo de tesis desde un marco teórico común. Ambos han sido aplicados en el entrenamiento de RHONNs para la identificación de dos de los manipuladores robóticos que se encuentran en el laboratorio de Mecatrónica y Control de la División de Posgrado e Investigación del ITL: el robot de dos g.d.l. y el brazo robótico Mitsubishi PA10-7CE.

En la mayoría de los experimentos y simulaciones realizadas se encontró que el desempeño era superior al utilizar el algoritmo de entrenamiento UKF que cuando se utilizaba el algoritmo EKF. Esto se pudo apreciar con el hecho de que con el algoritmo UKF se obtuvo un error cuadrático medio menor al obtenido por el algoritmo EKF.

Uno de los aspectos criticables de este trabajo es la sintonización poco práctica de los parámetros de diseño; específicamente, las matrices de covarianza $Q$ y $R$ que indican a los filtros las características del ruido en la medición en el proceso y la salida, respectivamente. Aunque se considera que los resultados obtenidos son suficientes, para justificar que el desempeño del algoritmo UKF es superior al del algoritmo EKF, al menos para los problema planteados y sus soluciones particulares. Cabe destacar que seguramente hubiese sido posible alcanzar un mejor desempeño sintonizando los filtros, sin recurrir a métodos heurísticos de prueba-error, sobre todo en los experimentos en tiempo real.

Otro aspecto importante fue el cambio de plataforma para realizar las pruebas en tiempo real en el robot de 2 g.d.l. Como se mencionó en el capítulo 4, las pruebas experimentales fueron realizadas en WinMechLab, lo cual requirió una gran cantidad de tiempo y esfuerzo, principalmente para entender la lógica de programación y funcionamiento de la interfaz.

## 5.1. Conclusiones generales

La principal aportación de esta tesis fue el control descentralizado neuronal con entrenamiento UKF propuesto que muestra un mejor desempeño que el basado en entrenamiento EKF. Como se menciona en el capítulo 2, la linealización realizada por el algoritmo EKF no toma en cuenta las incertidumbres ni el ruido en las variable del sistema, lo que requiere de una mayor atención a la sintonización de la matriz $Q$ para alcanzar un desempeño aceptable, por lo que el entrenamiento UKF

es más fácil de implementar.

De los objetivos planteados al inicio de la tesis (ver sección 1.2) se puede decir que se cumplieron en su mayoría. En resumen, las aportaciones de este trabajo son:

- Estudio de las propiedades y aplicaciones del filtro de Kalman para sistemas no lineales.
- Propuesta de un algoritmo de control descentralizado neuro-adaptable con entrenamiento UKF para comparación con el basado en EKF reportado en la literatura.
- Implementación en tiempo real del controlador propuesto en el robot manipulador de dos grados de libertad.
- Implementación a nivel simulación del controlador propuesto en el brazo robótico PA10-7CE.
- Publicación de un artículo en un congreso internacional con los resultados parciales de este trabajo de tesis.
- Artículo aceptado en un congreso nacional con los resultados parciales del robot Mitsubishi PA10-7CE, obtenidos en este trabajo de tesis.

## 5.2. Trabajo futuro

Por distintos factores, pero sobretodo por la falta de tiempo, como trabajo futuro se tienen los puntos pendientes siguientes:

- En primer lugar, aún es necesario encontrar la causa de las discrepancias entre los resultados de las simulaciones y las pruebas en tiempo real para el robot manipulador de 2 g.d.l., si bien, los resultados de las simulaciones (tanto en Simulink como en WinMechLab) no tuvieron grandes diferencias entre sí, los resultados en tiempo real no presentaron gran similitud. Por lo cual, se propone monitorear los pesos de la red al realizar los experimentos para evaluar su evolución y compararlos con los obtenidos en la simulación como una posible solución.
- Por otro lado, los experimentos en tiempo real para el manipulador PA10-7CE no pudieron ser realizados debido a un problema con la articulación 4. El problema se presentó al momento de ejecutar el programa de calentamiento “PIDseg” en la interfaz *PA10Control* de MATLAB desarrollada en [Flores-Ávila, 2014]; transcurridos 10 segundos del experimento la articulación se detenía mientras el resto seguía en movimiento; esto causaba que el experimento tuviera que ser detenido. Ya que la interfaz no muestra advertencias por rebasar las consignas articulares (pares y velocidades) es necesario revisar la documentación de la tarjeta de adquisición de datos, el paquete QUARC y la interfaz realizada en MATLAB para encontrar la posible causa del bloqueo de la articulación.

- Por último, también hace falta realizar la prueba de estabilidad del filtro UKF con ruido aditivo, la cual no se encuentra reportada en la literatura para el entrenamiento de redes neuronales. También sería conveniente implementar un algoritmo de control descentralizado neuroadaptable con entrenamiento basado en los filtros EKF y UKF, en el que los parámetros de diseño (pesos fijos) cuenten con una sintonización automática, es decir, no heurística.

# Apéndice A

## Publicaciones

Parte del trabajo de esta tesis se presentó como el artículo titulado “Decentralized Neural Block Control for a Robot Manipulator based in UKF training” en el 2019 International Conference on Electronics, Communications and Computers (CONIELECOMP), 2019, Puebla, México.

Además fue aceptado el artículo “Control Descentralizado Neuro-Adaptable Indirecto con entrenamiento UKF aplicado a Robótica” al Congreso Nacional de Control Automático 2019, Puebla, México.

# Decentralized Neural Block Control for a Robot Manipulator based in UKF training

Guerra J.F., Garcia-Hernandez R., Llama M.A.
*División de Estudios de Posgrado e Investigación, Instituto Tecnológico de la Laguna*
Torreón, México
[m.jfguerrac,rgarciah,mllama]@correo.itlalaguna.edu.mx

***Abstract*—This work presents an alternative solution of the trajectory tracking problem for a two degrees of freedom (DOF) robot manipulator using a discrete-time decentralized control strategy. The local controller for each joint uses only local angular position and velocity measurements. A recurrent high order neural network (RHONN) structure is used to aproximate the manipulator dynamics, and based on this model, a discrete-time control law is computed, which combines block control and the sliding mode techniques. The neural network learning is performed on-line by Unscented Kalman filter (UKF) algorithm, a variation of nonlinear Kalman Filter.**

***Index Terms*—Robot control, Sliding mode control, Trajectory tracking, Decentralized control, Recurrent neural networks, Unscented Kalman filter.**

## I. INTRODUCTION

The complexity of dynamic systems modeling is a problem that exists because of our intensive and limitless desire to build and control ever larger and more sophisticated systems. Due to this, new theories have arisen as alternatives to the orthodox concept of control of high order systems, such as distributed parameters, neural networks, and decentralized control theory.

Robot manipulators are examples of this class of systems, due to their nonlinear complex dynamics, with strong interconnections, parameters difficult to measure and dynamics difficult to model. Considering only the most important terms, the mathematical model obtained requires control algorithms with a great number of mathematical operations, for which it is possible to design decentralized control, that consist in independent controllers, considering only local variables to each subsystem [1].

The Kalman filter [2] is a popular and effective state estimator for linear systems, which is implemented in two steps: prediction and update. The prediction equations determine the a priori estimates of the state and error covariance; in the update step, the a priori estimates are updated using the computed Kalman gain matrix, resulting in a posteriori estimates [3]. In case of nonlinear systems, is required approximations in order to deal the nonlinearity in the state dynamic equations and/or the output equations of the system. While there are many variations on the Kalman filter for nonlinear systems, Extended Kalman Filter (EKF) [4] and the Unscented Kalman Filter (UKF) [5] are most common.

These filters operate within the existing Kalman filter framework, but use different approaches to deal the nonlinearity. The EKF uses an analytical linearization approach involving Jacobian matrices, while the UKF uses a statistical approach called the Unscented Transformation (UT) [5].

Some works and simulation studies have shown a comparison of EKF and UKF; in [6], authors present an analysis relative to performance of two different Global Positioning System/Inertial Navigation System (GPS/INS) sensor fusion algorithms for estimating aircraft attitude angle using the EKF and UKF. Detailed information about implementation of EKF and UKF algorithms is presented in [7].

Decentralized control has been applied in robot manipulators [8], [9] wherein each joint and the respective link is considered as a subsystem in order to develop local controllers, which just consider local angular position and angular velocity measurements, and compensate the interconnection effects, usually assumed as disturbances.

In [10], a decentralized control for robot manipulators is reported; it is based on the estimation of independent dynamics for each of the joints, using feedforward neural networks. A decentralized kinematic control of multiple redundant manipulators is proposed in [11], in order to solve the cooperative task execution problem, using a recurrent neural network with independent modules.

In [12], the authors present a discrete-time decentralized neural identification and control scheme for large-scale uncertain nonlinear systems, which is developed using recurrent high order neural networks (RHONN); the neural network learning algorithm uses an EKF. The discrete-time control law proposed is based on block control and sliding mode techniques. The control algorithm is first simulated and then implemented in real-time for a two degree of freedom (DOF) planar robot. In [13], the authors proposed a similar decentralized control strategy using a recurrent neural identifier and block control structure for both identification and control. This approach was tested only using simulations with a Mitsubishi PA10-7CE robot arm with seven DOF.

In present work, it is proposed an UKF-based training algorithm for a decentralized RHONN structure in order to identify a robot manipulator model. With this model, a discrete-time control is designed, which combines discrete-time block control and sliding mode techniques. The block control approach is used to design a nonlinear sliding surface such that the resulting sliding mode dynamics is described by a desired linear system [14]. Additionally, simulations for the proposed control scheme using a two DOF robot manipulator,

978-1-7281-1145-2/19/$31.00 ©2019 IEEE 

is presented with good performance.

## II. DISCRETE-TIME DECENTRALIZED SYSTEMS

Let consider a class of discrete-time nonlinear and interconnected system which can be presented in the nonlinear block-controllable (NBC) form [15] consisting of $r$ blocks

$$\begin{aligned}
\chi_{i,k+1}^1 &= f_i^1\left(\chi_i^1\right) + B_i^1\left(\chi_i^1\right)\chi_i^2 + \Gamma_{i\ell}^1 \\
\chi_{i,k+1}^2 &= f_i^2\left(\chi_i^1,\chi_i^2\right) + B_i^2\left(\chi_i^1,\chi_i^2\right)\chi_i^3 + \Gamma_{i\ell}^2 \\
&\vdots \\
\chi_{i,k+1}^j &= f_i^j\left(\chi_i^1,\chi_i^2,\cdots,\chi_i^j\right) \\
&\quad + B_i^j\left(\chi_i^1,\chi_i^2,\cdots,\chi_i^j\right)\chi_i^{j+1} + \Gamma_{i\ell}^j \\
&\vdots \\
\chi_{i,k+1}^r &= f_i^r\left(\chi_i\right) + B_i^r\left(\chi_i\right)u_i + \Gamma_{i\ell}^r
\end{aligned} \tag{1}$$

where $\chi_i \in \mathbb{R}^{n_i}$, $\chi_i = \begin{bmatrix}\chi_i^{1T} & \chi_i^{2T} & \cdots & \chi_i^{rT}\end{bmatrix}^T$ and $\chi_i^j \in \mathbb{R}^{n_{ij}\times 1}$, $\chi_i^j = \begin{bmatrix}\chi_{i1}^j & \chi_{i2}^j & \cdots & \chi_{il}^j\end{bmatrix}^T$, $i = 1,\ldots,N$; $j = 1,\ldots,r-1$; $l = 1,\ldots,n_{ij}$; $N$ is the number of subsystems, $u_i \in \mathbb{R}^{m_i}$ is the input vector, the rank of $B_i^j = n_{ij}$, $\sum_{j=1}^r n_{ij} = n_i$, $\forall \chi_i^j \in D_{\chi_i^j} \subset \mathbb{R}^{n_{ij}}$. It is assumed that $f_i^j$, $B_i^j$ and $\Gamma_i^j$ are smooth and bounded functions, $f_i^j(0) = 0$ and $B_i^j(0) = 0$. The integers $n_{i1} \le n_{i2} \le \cdots \le n_{ij} \le m_i$ define the different subsystem structures. The interconnection terms are given by

$$\begin{aligned}
\Gamma_{i\ell}^1 &= \sum_{\ell=1,\ell\neq i}^N \gamma_{i\ell}^1\left(\chi_\ell^1\right) \\
\Gamma_{i\ell}^2 &= \sum_{\ell=1,\ell\neq i}^N \gamma_{i\ell}^2\left(\chi_\ell^1,\chi_\ell^2\right) \\
&\vdots \\
\Gamma_{i\ell}^j &= \sum_{\ell=1,\ell\neq i}^N \gamma_{i\ell}^j\left(\chi_\ell^1,\chi_\ell^2,\cdots,\chi_\ell^j\right) \\
&\vdots \\
\Gamma_{i\ell}^r &= \sum_{\ell=1,\ell\neq i}^N \gamma_{i\ell}^r\left(\chi_\ell\right)
\end{aligned} \tag{2}$$

where $\chi_\ell = \left[\chi_\ell^1,\chi_\ell^2,\cdots,\chi_\ell^j\right]^T$ represents the state vector of the $\ell$-th subsystem with $1 \le \ell \le N$ and $\ell \neq i$.

Terms in (2) reflect the interaction between the $i$-th subsystem and the other ones.

### A. Decentralized Neural Network Identifier

The following decentralized RHONN structure is proposed to identify (1):

$$\begin{aligned}
x_{i,k+1}^1 &= w_{i,k}^1 S\left(\chi_{i,k}^1\right) + w_i^{'1}\chi_{i,k}^2 \\
x_{i,k+1}^2 &= w_{i,k}^2 S\left(\chi_{i,k}^1,\chi_{i,k}^2\right) + w_i^{'2}\chi_{i,k}^3 \\
&\vdots \\
x_{i,k+1}^j &= w_{i,k}^j S\left(\chi_{i,k}^1,\chi_{i,k}^2,\cdots,\chi_{i,k}^j\right) + w_i^{'j}\chi_i^{j+1}(k) \\
&\vdots \\
x_{i,k+1}^r &= w_{i,k}^r S\left(\chi_{i,k}^1,\cdots,\chi_{i,k}^r\right) + w_i^{'r}u_{i,k}
\end{aligned} \tag{3}$$

where $x_{i,k+1}^j = \begin{bmatrix}x_i^1 & x_i^2 & \cdots & x_i^r\end{bmatrix}^T$ is the $j$-th block neuron state with $i = 1,\ldots,N$ and $j = 1,\ldots,r-1$, $w_{i,k}^j$ are fixed parameters with $rank(w_i^{'j}) = n_{ij}$, $S(\bullet)$ is the activation function, and $u_{i,k}$ represents the input vector.

It is worth to note that (3), constitutes a series-parallel identifier configuration [16] and does not consider explicitly the interconnection terms, whose effects are compensated by the neural network weights update.

### B. On-Line Learning Law

There are different training algorithms for neural networks (NN), including those based on Kalman filter (KF) [13]–[15] and [17]. Due to the fact that NN mapping is nonlinear, Unscented Kalman Filter (UKF) based algorithm is used instead linear KF [18].

The training goal is to find the optimal weight values $w_{i,k}^j$ which minimize the prediction error. We use an UKF-based training algorithm described for each $i$-th neuron by [5], [7], [18] and [19]

$$\begin{aligned}
K_{i,k}^j &= P_{i,k}^{jxy}(P_{i,k}^{jyy})^{-1} \\
w_{i,k+1}^j &= w_{i,k}^j + K_{i,k}^j e_{i,k}^j \\
P_{i,k+1}^j &= P_{i,k}^j - K_{i,k}^j P_{i,k}^{jyy} K_{i,k}^{jT}
\end{aligned} \tag{4}$$

with

$$\begin{aligned}
P_{i,k}^{jxy} &= \sum_{i=0}^{2L} \eta_i^c[\mathcal{X}_{i,k|k-1}^j - \hat{x}_{i,k}^{j-}][\mathcal{Y}_{i,k|k-1}^j - \hat{y}_{i,k}^{j-}]^T \\
P_{i,k}^{jyy} &= R_{i,k}^j + \sum_{i=0}^{2L} \eta_i^{jc}[\mathcal{Y}_{i,k|k-1}^j - \hat{y}_{i,k}^{j-}][\mathcal{Y}_{i,k|k-1}^j - \hat{y}_{i,k}^{j-}]^T \\
P_{i,k}^j &= Q_{i,k}^j + \sum_{i=0}^{2L} \eta_i^{jc}[\mathcal{X}_{i,k|k-1}^j - \hat{x}_{i,k}^{j-}][\mathcal{X}_{i,k|k-1}^j - \hat{x}_{i,k}^{j-}]^T \\
e_{i,k}^j &= [\chi_{i,k}^j - x_{i,k}^j]
\end{aligned} \tag{5}$$

where $e_{i,k}^j$ is the identification error, $P_{i,k}^j$ is the prediction error covariance matrix, $P_{i,k}^{jyy}$ is the covariance of predicted output matrix, $P_{i,k}^{jxy}$ is the cross-covariance of state and output matrix; $w_{i,k}^j$ is the $j$-th weight (state) of the $i$-th subsystem, $\eta_i^{jc}$ is a design parameter, $\chi_{i,k}^j$ is the $j$-th plant state and $x_{i,k}^j$ is the $j$-th neural network state. $L$ is the number of states, $K_{i,k}^j$ is the Kalman gain matrix, $Q_{i,k}^j$ is the measurement noise covariance matrix, $R_{i,k}^j$ is the state noise covariance matrix, $\hat{x}_{i,k}^{j-}$ is the mean of predicted state, $\hat{y}_{i,k}^{j-}$ is the mean of predicted output, $\mathcal{X}_{i,k|k-1}^j$ and $\mathcal{Y}_{i,k|k-1}^j$ are propagation each sigma-point through prediction and observation respectively.

The UT is the central technique of the UKF algorithm whose principal characteristic is generate sigma points as follows

$$\mathcal{X}^{j}_{i,k-1} = [w^{j}_{i,k-1} \quad w^{j}_{i,k-1} \pm \sqrt{L+\lambda}\sqrt{P^{j}_{i,k-1}}] \tag{6}$$

where $\mathcal{X}^{j}_{i,k-1} \in \mathbb{R}^{L\times 2L+1}$ is a matrix of sigma points, where each column of this matrix represents a sigma point. Once the sigma points have been generated, each point is passed through the nonlinear functions and then used to calculate

$$\begin{aligned}
\mathcal{X}^{j}_{i,k|k-1} &= F(\mathcal{X}^{j}_{i,k-1}, u_{i,k-1}) \\
\mathcal{Y}^{j}_{i,k|k-1} &= H(\mathcal{X}^{j}_{i,k-1}, u_{i,k-1}) \\
\hat{x}^{j-}_{i,k} &= \sum_{i=0}^{2L} \eta^{jm}_{i} \mathcal{X}^{j}_{i,k|k-1} \\
\hat{y}^{j-}_{i,k} &= \sum_{i=0}^{2L} \eta^{jm}_{i} \mathcal{Y}^{j}_{i,k|k-1}
\end{aligned} \tag{7}$$

where $F$ is a nonlinear state prediction function, $H$ is a nonlinear output prediction function, $\lambda$ is a scaling parameter, and weight vectors $\eta^{jm}_{i}$ (mean) and $\eta^{jc}_{i}$ (covariance) are defined by

$$\begin{aligned}
\lambda &= \alpha^2(L+\kappa) - L \\
\eta^{jm}_{0} &= \lambda/(L+\lambda) \\
\eta^{jc}_{0} &= \lambda/(L+\lambda) + 1 - \alpha^2 + \beta \\
\eta^{jm}_{i} &= \eta^{jc}_{i} = \lambda/[2(L+\lambda)] \\
i &= 1, \ldots, 2L
\end{aligned}$$

The primary scaling parameter $\alpha$ determines the spread of the sigma points such that $10^{-4} \leq \alpha \leq 1$. Secondary scaling parameter $\beta$ is used to include information about the prior distribution (for Gaussian distributions $\beta = 2$), and $\kappa = 3 - L$ provides an extra degree of freedom to "fine tune" the higher order moments of the approximation, and can be used to reduce the overall prediction errors [5], [19].

Note that use of the subscript $k|k$-1, or "$k$ given $k$-1", means that this is the predicted value of the sigma point based on the information from the previous time step. An alternative representation that is sometimes used instead of $k|k$-1 is a subscript of $k^-$ [7].

### *C. Controller Design*

For trajectory tracking of the first block in (1), let us define tracking error as

$$\mathbf{z}^{1}_{i,k} = x^{1}_{i,k} - x^{1}_{\mathrm{id},k} \tag{8}$$

where $x^{1}_{\mathrm{id},k}$ is the $i$-th desired trajectory signal and $x^{1}_{i,k}$ is the $i$-th neural network state, with $i = 1, \ldots, N$ subsystems.

Once the first new variable (8) is determined, one step ahead is taken

$$\mathbf{z}^{1}_{i,k+1} = w^{1}_{i,k} S(\chi^{1}_{i,k}) + w_i^{'1} \chi^{2}_{i,k} - x^{1}_{\mathrm{id},k+1} \tag{9}$$

Equation (9) is viewed as a block with state $\mathbf{z}^{1}_{i,k}$ and the state $\chi^{2}_{i,k}$ is considered as a pseudo-control input, where desired dynamics can be imposed. This equation can be solved with the anticipation of the desired dynamics for this block as

$$\begin{aligned}
\mathbf{z}^{1}_{i,k+1} &= w^{1}_{i,k} S(\chi^{1}_{i,k}) + w_i^{'1} \chi^{2}_{i,k} - x^{1}_{\mathrm{id},k+1} \\
&= k^{1}_{i} \mathbf{z}^{1}_{i,k}
\end{aligned} \tag{10}$$

where $|k^{1}_{i}| < 1$, in order to ensure stability of (10). $x^{2}_{\mathrm{id},k}$ is computed as

$$x^{2}_{\mathrm{id},k} = \frac{1}{w_i^{'1}}[-w^{1}_{i,k} S(\chi^{1}_{i,k}) + \chi^{1}_{\mathrm{id},k+1} + k^{1}_{i} \mathbf{z}^{1}_{i,k}] \tag{11}$$

Note that the calculated value of state $x^{2}_{\mathrm{id},k}$ in (11) is not its real value; instead of it, represents the desired behavior for $\chi^{2}_{i,k}$. Then, to avoid confusion, this desired value of $\chi^{2}_{i,k}$ is referred as $x^{2}_{\mathrm{id},k}$ in (11).

Proceeding in the same way as for the first block, a second variable in the new coordinates is defined as:

$$\mathbf{z}^{2}_{i,k} = x^{2}_{i,k} - x^{2}_{\mathrm{id},k} \tag{12}$$

Taking one step ahead for $\mathbf{z}^{2}_{i,k}$ yields

$$\mathbf{z}^{1}_{i,k+1} = x^{2}_{i,k+1} - x^{2}_{\mathrm{id},k+1} \tag{13}$$

The desired dynamics for this block is imposed as

$$\begin{aligned}
\mathbf{z}^{2}_{i,k+1} &= w^{2}_{i,k} S(\chi^{1}_{i,k}, \chi^{2}_{i,k}) + w_i^{'2} \chi^{3}_{i,k} - x^{2}_{\mathrm{id},k+1} \\
&= k^{2}_{i} \mathbf{z}^{2}_{i,k}
\end{aligned} \tag{14}$$

where $|k^{2}_{i}| < 1$.

These steps are taken iteratively. At the last step, the known desired variable is $x^{r}_{\mathrm{id},k}$, and the last new variable is defined as

$$\mathbf{z}^{r}_{i,k} = x^{r}_{i,k} - x^{r}_{\mathrm{id},k}$$

As usually, taking one step ahead yields

$$\mathbf{z}^{r}_{i,k+1} = w^{r}_{i,k} S(\chi^{1}_{i,k}, \ldots, \chi^{r}_{i,k}) + w_i^{'r} u_{i,k} - x^{r}_{\mathrm{id},k+1} \tag{15}$$

System (3) can be represented in the new variables $\mathbf{z} = [z_i^{1T} \quad z_i^{2T} \quad \cdots \quad z_i^{rT}]$ as

$$\begin{aligned}
\mathbf{z}^{r-1}_{i,k+1} &= k^{r-1}_{i} \mathbf{z}^{r-1}_{i,k} + w_i^{'r-1} \mathbf{z}^{r}_{i,k} \\
\mathbf{z}^{r}_{i,k+1} &= w^{r}_{i,k} S(\chi^{1}_{i,k}, \ldots, \chi^{r}_{i,k}) + w_i^{'2} u_{i,k} - x^{r}_{\mathrm{id},k+1}
\end{aligned} \tag{16}$$

For a sliding mode control implementation, when the control resources are bounded by $u_{0i}$

$$|u_{i,k}| \leq u_{0i} \tag{17}$$

a sliding manifold and a control law, which drives the states toward such manifold, must be designed. The sliding manifold is selected as $S_{D_{i,k}} = \mathbf{z}^{r}_{i,k} = 0$; then, system (16) is rewritten as follows:

$$S_{D_{i,k+1}} = w^{r}_{i,k} S(\chi^{1}_{i,k}, \ldots, \chi^{r}_{i,k}) + w_i^{'2} u_{i,k} - x^{r}_{\mathrm{id},k+1} \tag{18}$$

Once the sliding manifold is defined, the next step is to find a control law which takes into consideration the bound (18), therefore, the control $u_{i,k}$ is selected as [20]

$$u_{i,k} = \begin{cases} u_{eq_{i,k}} & \text{for } \left\| u_{eq_{i,k}} \right\| \leq u_{0i} \\ u_{0i} \dfrac{u_{eq_{i,k}}}{\left\| u_{eq_{i,k}} \right\|} & \text{for } \left\| u_{eq_{i,k}} \right\| > u_{0i} \end{cases} \tag{19}$$

where the equivalent control $u_{eq_{i,k}}$ is calculated from $S_{D_{i,k+1}} = 0$ as

$$u_{eq_{i,k}} = \frac{1}{w_i'^r}[-w_{i,k}^r S(\chi_{i,k}^1, \ldots, \chi_{i,k}^r) + x_{\mathrm{id},k+1}^r] \tag{20}$$

The whole proposed identification and control scheme for the system is displayed in Fig.1.

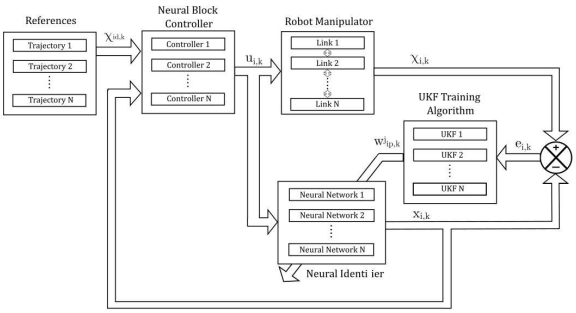


Fig. 1. Decentralized neural block control scheme

## III. APPLICATION TO A TWO DOF ROBOT MANIPULATOR

### A. Robot Description

The main goal of the controller is to obtain good performance in trajectory tracking tasks for a two DOF, vertical direct-drive robot, which is a prototype robot for research purposes. This robot is at the Mechatronics and Control Laboratory of the Instituto Tecnológico de la Laguna.

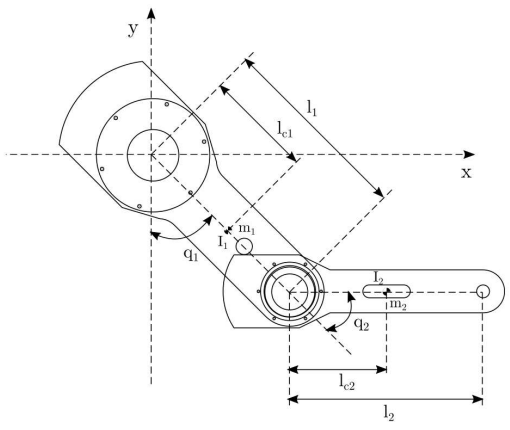


Fig. 2. Two DOF robot manipulator

The robot manipulator, illustrated in Fig. 2, consists of two rigid links. High-torque brushless direct-drive servos are used to drive the joints without gear reduction. This kind of joints present reduced backlash and significantly lower joint friction as compared to the actuators with gear drives [12].

In Table I are included the numerical values of the different parameters shown in Fig. 2.

TABLE I
2-DOF ROBOT MANIPULATOR PARAMETERS

| Description | Notation | Value | Unit |
|---|---|---|---|
| Mass link 1 | $m_1$ | 23.902 | $[kg]$ |
| Mass link 2 | $m_2$ | 3.880 | $[kg]$ |
| Length link 1 | $l_1$ | 0.450 | $[m]$ |
| Length link 2 | $l_2$ | 0.450 | $[m]$ |
| Inertia link 1 | $I_1$ | 1.266 | $[kg\ m^2]$ |
| Inertia link 2 | $I_2$ | 0.093 | $[kg\ m^2]$ |
| Center of mass link 1 | $l_{c1}$ | 0.091 | $[m]$ |
| Center of mass link 2 | $l_{c2}$ | 0.048 | $[m]$ |

### B. Control Objective

To identify the robot manipulator model, from (1) and (3) it is proposed the following decentralized series-parallel neural network configuration

$$\begin{aligned} x_{i,k+1}^1 &= w_{i1,k}^1 S\left(\chi_{i,k}^1\right) + w_i'^1 \chi_{i,k}^2 \\ x_{i,k+1}^2 &= w_{i1,k}^2 S\left(\chi_{i,k}^1\right) + w_{i2,k}^2 S\left(\chi_{i,k}^2\right) + w_i'^2 u_{i,k} \end{aligned} \tag{21}$$

where $x_{i,k}^1$ and $x_{i,k}^2$ identify $\chi_{i,k}^1$ and $\chi_{i,k}^2$; $i = 1, 2$, respectively; $w_{ip}^j$ are the adjustable weights, $p$ is the number of the adjustable weights with $p = 1$; $j = 1$ for the first NN state and $p = 1, 2$; $j = 1, 2$ for the second one; $w_i'^1$ and $w_i'^2$ are fixed parameters.

Due to the time varying of RHONN weights, we need to guarantee the controllability of the system by assuring the weights $w_i'^1$ and $w_i'^2$ are not zero. To update the weights $w_{ip}^j$ an UKF-based training algorithm is implemented.

The goal is to track a desired reference signal, which is achieved by designing a control law based on the sliding mode technique. The tracking error is defined as

$$\mathbf{z}_{i,k}^1 = x_{i,k}^1 - x_{\mathrm{id},k}^1 \tag{22}$$

where $x_{\mathrm{id},k}^1$ is the desired trajectory signal. Using (21) and introducing the desired dynamics for $\mathbf{z}_{i,k}^1$ we have

$$\begin{aligned} \mathbf{z}_{i,k+1}^1 &= w_{i1,k}^1 S(\chi_{i,k}^1) + w_i'^2 \chi_{i,k}^2 - x_{\mathrm{id},k+1}^1 \\ &= k_i^1 \mathbf{z}_{i,k+1}^1 \end{aligned} \tag{23}$$

desired value $x_{\mathrm{id},k+1}^1$ for $\chi_{i,k}^2$ is computed from (23) as

$$x_{\mathrm{id},k+1}^2 = \frac{1}{w_i'^1}[-w_{i1,k}^1 S(\chi_{i,k}^1) + \chi_{\mathrm{id},k+1}^1 + k_i^1 \mathbf{z}_{i,k}^1] \tag{24}$$

At the next step, let define a new variable as

$$\mathbf{z}_{i,k}^2 = x_{i,k}^2 - x_{\mathrm{id},k}^2 \tag{25}$$

Taking one step ahead, we have

$$\begin{aligned} \mathbf{z}_{i,k+1}^2 &= w_{i1,k}^2 S\left(\chi_{i,k}^1\right) + w_{i2,k}^2 S\left(\chi_{i,k}^2\right) + w_i'^2 \chi_{i,k}^3 - x_{\mathrm{id},k+1}^2 \\ &= k_i^2 \mathbf{z}_{i,k}^2 \end{aligned} \tag{26}$$

manifold for the sliding mode is chosen as $S_{D_i,k} = z_{i,k}^2 = 0$. The control law is given by

$$u_{i,k} = \begin{cases} u_{eq_{i,k}} & \text{for } \left\| u_{eq_{i,k}} \right\| \leq \tau_i^{\max} \\ u_{0i} \frac{u_{eq_{i,k}}}{\left\| u_{eq_{i,k}} \right\|} & \text{for } \left\| u_{eq_{i,k}} \right\| > \tau_i^{\max} \end{cases} \quad (27)$$

where the $u_{eq_{i,k}}$ is computed from $S_{D_{i,k+1}} = 0$ as

$$u_{eq_{i,k}} = \frac{1}{w_i'^2}[-(w_{i1,k}^2 S(\chi_{i,k}^1) + w_{i2,k}^2 S(\chi_{i,k}^2)) + x_{\text{id},k+1}^2] \quad (28)$$

and the control resources are bounded by $\tau_i^{\max}$

## IV. SIMULATION RESULTS

For simulation, the discrete-time trajectories proposed in [12] were chosen as

$$x_{1\text{d},k}^1 = b_1(1 - e^{d_1 kT^3}) + c_1(1 - e^{d_1 kT^3}) \sin(\omega_1 kT)[\text{rad}]$$
$$x_{2\text{d},k}^1 = b_2(1 - e^{d_2 kT^3}) + c_2(1 - e^{d_2 kT^3}) \sin(\omega_2 kT)[\text{rad}]$$

where $b_1 = \pi/4$, $c_1 = \pi/18$, $d_1 = -2.0$, and $\omega_1 = 5.0$ [rad/s] are parameters of the desired position trajectory for the first joint, whereas $b_2 = \pi/3$, $c_2 = 25\pi/36$, $d_2 = -1.8$, and $\omega_2 = 1.0$ [rad/s] are parameters of the desired position trajectory for the second joint with a sampling period $T = 2.5$ milliseconds.

These selected trajectories incorporate a sinusoidal term to evaluate the performance in presence of relatively fast periodic signals, for which the non-linearities of the robot dynamics are really important and present a term that smoothly grows for maintaining the robot in an operation state without saturating actuators whose limit are in 150 $[Nm]$ and 15 $[Nm]$, respectively [12].

Simulation results of the proposed approach for identification and trajectory tracking using the decentralized neural block control (DNBC) with UFK training scheme are shown in Fig. 3 and Fig. 4. The initial conditions for the plant are the same as those of the neural identifier, which are equal to zero for both joints. According to these figures, the identification errors for all joints present a good behavior and remain bounded as shown in Fig. 5. The tracking errors for each joint are presented in Fig. 6. The applied torques to each joint are shown in Fig. 7. The control signals present oscillations at some instants of time due to the gains and fixed parameters chosen for each controller. It is easy to see that both control signals are always inside of the prescribed limits given by the actuators manufacturer, that is, their absolute values are smaller than the bounds $\tau_1^{max}$ and $\tau_2^{max}$, respectively. In Fig. 8 to Fig. 12 are illustrated the simulations result of the DNBC with EKF training.

Kalman filters provides a way to estimate the state vector using an optimal observer gain to minimize the expected Mean Square Error (MSE) of the estimated signal. Due this fact, Table II includes a comparative analysis of the two learning algorithm schemes, whose performance criterion considered is the MSE value of the tracking error $e_{i_{Track},k}^1$ calculated as

$$MSE[e_{i_{Track},k}^1] = \sqrt{\frac{1}{t}\sum_{k=0}^{n} ||e_{i_{Track},k}^1||^2 T} \quad (29)$$

where $e_{i_{Track},k}^1 = x_{\text{id},k}^1 - \chi_{i,k}^1$, $\chi_{i,k}^1$ is the system output, $T$ is the sampling rate and $t$ is the simulation time.

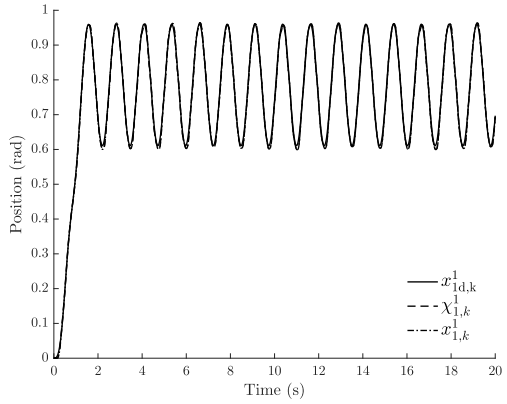


Fig. 3. Identification and tracking for joint 1 with UKF training

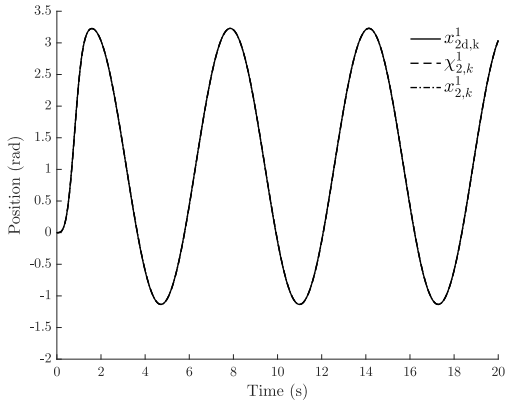


Fig. 4. Identification and tracking for joint 2 with UKF training

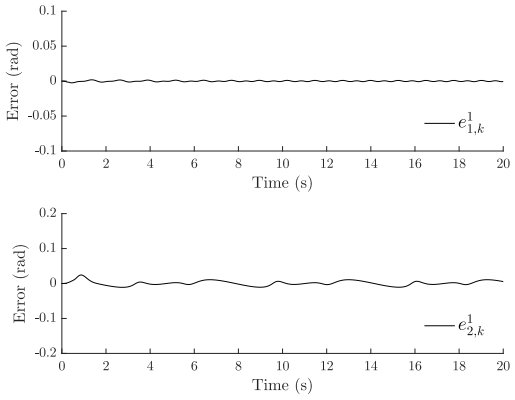


Fig. 5. Identification errors for joints 1 and 2 with UKF training

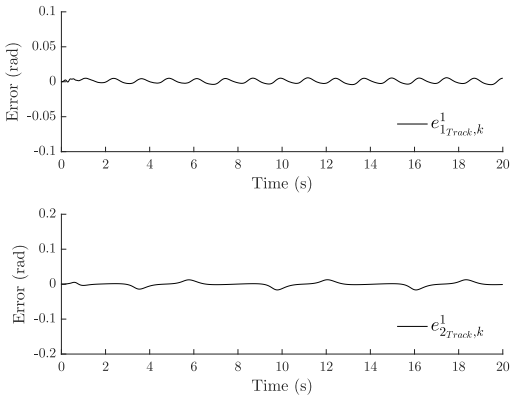


Fig. 6. Tracking errors for joints 1 and 2 with UKF training

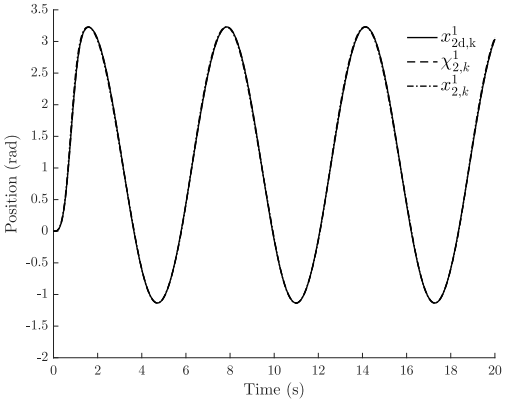


Fig. 9. Identification and tracking for joint 2 with EKF training

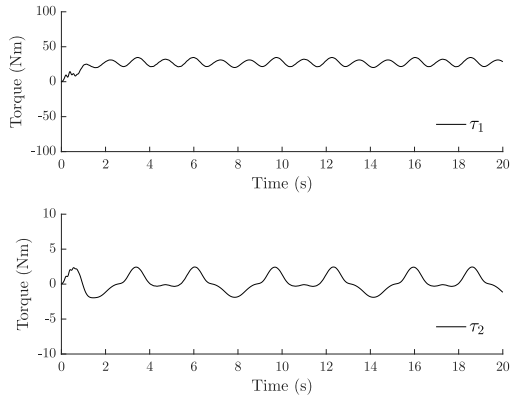


Fig. 7. Applied torques to joint 1 and 2 with UKF training

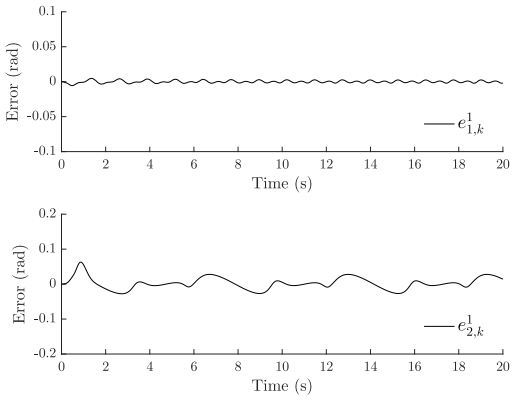


Fig. 10. Identification errors for joints 1 and 2 with EKF training

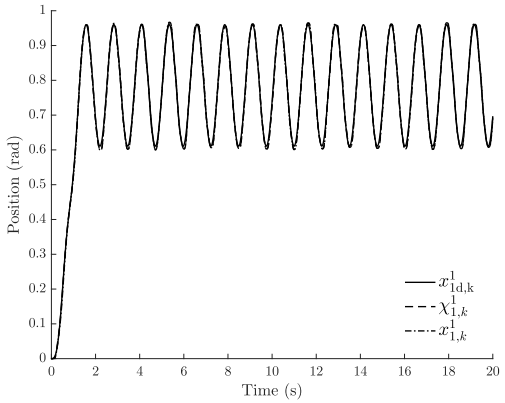


Fig. 8. Identification and tracking for joint 1 with EKF training

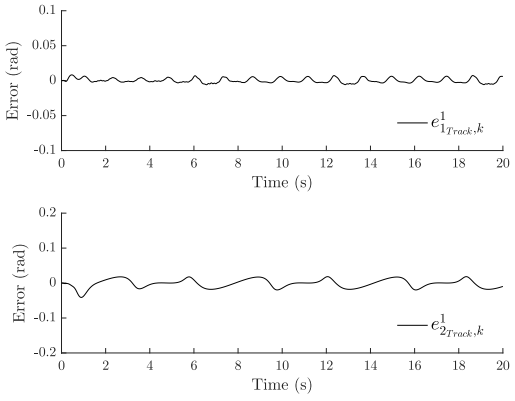


Fig. 11. Tracking errors for joints 1 and 2 with EKF training

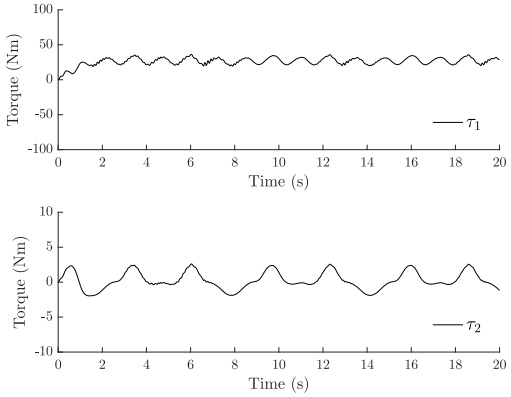


Fig. 12. Applied torques to joint 1 and 2 with EKF training

TABLE II
COMPARISON OF THE MSE FOR THE TRACKING ERROR 2-DOF ROBOT

| Algorithm | $e^1_{1_{Track},k}$ | $e^1_{2_{Track},k}$ |
|---|---|---|
| UKF | 2.8854e-3 | 5.73e-3 |
| EKF | 3.3145e-3 | 12.37e-3 |

## V. CONCLUSIONS

A decentralized neural identification and control scheme for trajectory tracking is applied to a discrete-time two DOF robot manipulator model. The training of the each recurrent high-order neural network is performed online using an Unscented Kalman filter in a series-parallel configuration. It is important to remark that the proposed approach control scheme does not require to know the plant parameters nor the external disturbances; the obtained simulation results are very encouraging. In addition, as shown in the comparison between the two training algorithms, UKF training shows better performance and is easier to implement in robot manipulators.

As future work, research will proceed to obtain real-time experiments in several robots.

## REFERENCES

[1] D. D. Siljak, *Decentralized control of complex systems*. Courier Corporation, 2011.
[2] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.
[3] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.
[4] R. E. Kalman and R. S. Bucy, "New results in linear filtering and prediction theory," *Journal of Basic Engineering*, vol. 83, no. 1, pp. 95–108, 1961.
[5] S. J. Julier and J. K. Uhlmann, "A new extension of the Kalman Filter to nonlinear systems," *Signal processing, sensor fusion, and target recognition VI*, vol. 3068, no. International Society for Optics and Photonics., pp. 182–194, 1997.
[6] M. Rhudy, Y. Gu, J. Gross, S. Gururajan, and M. Napolitano, "Sensitivity and robustness analysis of EKF and UKF design parameters for GPS/INS sensor fusion," *AIAA Journal of Aerospace Information Systems*, vol. 10, no. 3, pp. 131–143, 2013.
[7] M. Rhudy and Y. Gu, "Understanding nonlinear Kalman filters, part II: An implementation guide," *Interactive Robotics Letters*, 2013, http://www2.statler.wvu.edu/%7Eirl/page13.html.
[8] Y. Jin, "Decentralized adaptive fuzzy control of robot manipulators," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 28, no. 1, pp. 47–57, 1998.
[9] M. Liu, "Decentralized control of robot manipulators: nonlinear and adaptive approaches," *IEEE Transactions on Automatic control*, vol. 44, no. 2, pp. 357–363, 1999.
[10] R. Safaric and J. Rodic, "Decentralized neural-network sliding-mode robot controller," in *Industrial Electronics Society, 2000. IECON 2000. 26th Annual Conference of the IEEE*, vol. 2. IEEE, 2000, pp. 906–911.
[11] S. Li, S. Chen, B. Liu, Y. Li, and Y. Liang, "Decentralized kinematic control of a class of collaborative redundant manipulators via recurrent neural networks," *Neurocomputing*, vol. 91, pp. 1–10, 2012.
[12] R. García-Hernández, E. N. Sanchez, V. Santibáñez, M. A. Llama, and E. Bayro-Corrochano, "Real-time decentralized neural block controller for a robot manipulator," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 906–911.
[13] R. García-Hernández, E. N. Sanchez, V. Santibáñez, and J. Ruz-Hernandez, "Decentralized neural block control for an industrial PA10-7CE robot arm," IEEE, pp. 2787–2794, 2011.
[14] R. García-Hernández, M. Lopez-Franco, E. N. Sanchez, A. Y. Alanis, and J. Ruz-Hernandez, *Decentralized Neural Control: Application to Robotics*, ser. Studies in Systems, Decision and Control. Switzerland: Springer, 2016, vol. 96.
[15] E. N. Sanchez, A. Y. Alans, and A. G. Loukianov, *Discrete-time high order neural control: trained with Kalman filtering*. Springer Science, Business Media, 2008, vol. 112.
[16] Y. Zhang and P. A. Ioannou, "A new class of nonlinear robust adaptive controllers," *International Journal of Control*, vol. 65, no. 5, pp. 745–769, 1996.
[17] A. Y. Alanis, E. N. Sanchez, and A. G. Loukianov, "Discrete-time adaptive backstepping nonlinear control via high-order neural networks," *IEEE Transactions on Neural Networks*, vol. 18, no. 4, pp. 1185–1195, 2007.
[18] S. Haykin, *Kalman filtering and neural networks*. John Wiley and Sons, 2004, vol. 47.
[19] E. A. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*. IEEE, 2000, Conference Proceedings, pp. 153–158.
[20] V. Utkin, J. Guldner, and J. Shi, *Sliding mode control in electro-mechanical systems*. CRC press, 2009.

# Apéndice B

## Filtros de Kalman no lineales

| **Filtro de Kalman Extendido (EKF)** | **Filtro de Kalman Unscented (UKF)** |
|---|---|
| Parámetros de inicialización de los Filtros | |
| Matrices jacobianas analíticas<br>$F_{k-1} = \frac{\partial f}{\partial x_{k-1}}\big\vert_{\hat{x}_{k-1}},\ H_k = \frac{\partial h}{\partial x_k}\big\vert_{\hat{x}_{k\vert k-1}}$ | Parámetros de escalamiento<br>$\alpha = 1,\ \beta = 2,\ \kappa = 3 - L$<br>$\lambda = \alpha(L+\kappa) - L$<br>$\eta_0^m = \lambda/(L+\lambda)$<br>$\eta_0^c = \lambda/(L+\lambda) + 1 - \alpha^2 + \beta$<br>$\eta_i^m = \eta_i^c = 1/[2(L+\lambda)],\ i = 1,\ldots,2L$ |
| Inicialización de los Filtros | |
| $Q_k = E\left[w_k w_k^T\right],\quad R_k = \left[v_k v_k^T\right]\quad \hat{x}_0 = E[x_0],\quad P_0 = \left[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T\right]$ | |
| Predicción | |
| $\hat{x}_{k\vert k-1} = f\left(\hat{x}_{k-1}, u_{k-1}\right)$<br>$P_{k\vert k-1} = F_{k-1} P_{k-1} F_{k-1}^T + Q_{k-1}$ | $\sqrt{P_{k-1}} = chol(P_{k-1})$<br>$\chi_{k-1} = \left[\hat{x}_{k-1}\ \ \hat{x}_{k-1} + \sqrt{L+\lambda}\sqrt{P_{k-1}}\ \ \hat{x}_{k-1} - \sqrt{L+\lambda}\sqrt{P_{k-1}}\right]$<br>$\chi_{k\vert k-1}^{(i)} = f\left(\chi_{k-1}^{(i)}, u_{k-1}\right),\ i = 0,1,\ldots,2L$<br>$\hat{x}_{k\vert k-1} = \sum_{i=0}^{2L} \eta_i^m \chi_{k\vert k-1}^{(i)}$<br>$P_{k\vert k-1} = Q_{k-1} + \sum_{i=0}^{2L} \eta_i^m \left(\chi_{k\vert k-1}^{(i)} - \hat{x}_{k\vert k-1}\right) \left(\chi_{k\vert k-1}^{(i)} - \hat{x}_{k\vert k-1}\right)^T$ |
| Observación | |
| $\hat{y}_{k\vert k-1} = h\left(\hat{x}_{k\vert k-1}\right)$<br>$P_k^{yy} = H_k P_{k\vert k-1} H_k^T + R_k$<br>$P_k^{xy} = P_{k\vert k-1} H_k^T$ | $\psi_{k\vert k-1}^{(i)} = h\left(\chi_{k-1}^{(i)}, u_{k-1}\right),\ i = 0,1,\ldots,2L$<br>$\hat{y}_{k\vert k-1} = \sum_{i=0}^{2L} \eta_i^m \psi_{k\vert k-1}^{(i)}$<br>$P_k^{xy} = \sum_{i=0}^{2L} \eta_i^m \left(\chi_{k\vert k-1}^{(i)} - \hat{x}_{k\vert k-1}\right) \left(\psi_{k\vert k-1}^{(i)} - \hat{y}_{k\vert k-1}\right)^T$<br>$P_k^{yy} = R_k + \sum_{i=0}^{2L} \eta_i^m \left(\psi_{k\vert k-1}^{(i)} - \hat{y}_{k\vert k-1}\right) \left(\psi_{k\vert k-1}^{(i)} - \hat{y}_{k\vert k-1}\right)^T$ |
| Actualización | |
| $K_k = P_k^{xy}\left(P_k^{yy}\right)^{-1}$<br>$\hat{x}_k = \hat{x}_{k\vert k-1} + K_k\left(y_k - \hat{y}_{k\vert k-1}\right)$<br>$P_k = P_{k\vert k-1} - K_k P_k^{yy} K_k^T$ | |

# Apéndice C

## WinMechLab

WinMechLab acrónimo de *Windows Mechatronics Laboratory* [Campa and Kelly, 2008] es un sistema multipropósito para el control en tiempo real de mecanismos, ejecutándose bajo MS– Windows, en adición con RTX. WinMechLab permite la edición, compilación, simulación y ejecución de algoritmos de control, escritos usando una sintaxis simple, muy similar al lenguaje C.

Una de las características principales de WinMechLab es su versatilidad en el manejo de dispositivos de hardware. WinMechLab 2.0 incluye bibliotecas de interfaz para algunas de las tarjetas de adquisición comúnmente usadas como: MultiQ y MultiQ-PCI de Quanser Consulting, ServoToGo de ServoToGo Inc., MFIO-3A de Precision MicroDynamics y Sensoray626.

Otra característica interesante de WinMechLab es su capacidad para simular mecanismos con solo proporcionar el modelo correspondiente en la descripción de estado-variable.

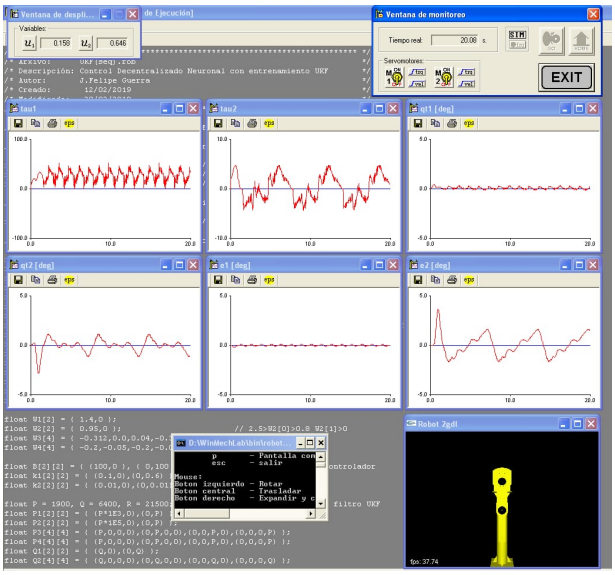


Figura C.1: Interfaz del simulador WinMechLab

## C.1. Estructura de programa en WinMechLab

```
/* ************************************************************************ */
/* Archivo:      .rob                                                       */
/* Descripción:                                                             */
/* Autor:                                                                   */
/* Creado:                                                                  */
/* Modificado:                                                              */
/* ************************************************************************ */

/* ***************** DECLARACION DE VARIABLES Y PARAMETROS *************** */

/* ------------------------------ Variables principales ------------------------------ */

float pos1, pos2;                       /* posiciones articulares [deg]    */
float vel1, vel2;                       /* velocidades articulares [deg/s] */
float tau1, tau2;                       /* pares articulares [N-m]         */

/* ------------------------------ Variables auxiliares ------------------------------- */

float t;                                /* Tiempo de ejecución [s]         */

/* ---------------------------------- Parámetros ------------------------------------- */


/* ************************** CUERPO DEL PROGRAMA ************************** */


/* ---------------------------------- Conversiones ----------------------------------- */


/* ------------------------------ Cálculo de referencias ----------------------------- */

t = Tiempo_real;

/* ------------------------- Evaluación de modelo dinámico ----------------------- */


/* ---------------------------------- Ley de control --------------------------------- */

tau1 = 0.0;
tau2 = 0.0;

/* ************************** FIN DEL PROGRAMA ***************************** */
```

## C.2. Control descentralizado neuronal con entrenamiento UKF (WinMechLab)

```
/* ************************************************************************ */
/* Archivo:       UKF(seq).rob                                              */
/* Descripción:   Control Decentralizado Neuronal con entrenamiento UKF     */
/* Autor:         J.Felipe Guerra                                           */
/* Creado:        12/02/2019                                                */
/* Modificado:    02/04/2019                                                */
/* ************************************************************************ */

/* ***************** DECLARACION DE VARIABLES Y PARAMETROS ************** */

/* ---------------------------- Variables principales ---------------------- */

float pos1, pos2;                       /* posiciones articulares [deg]    */
float vel1, vel2;                       /* velocidades articulares [deg/s] */
float tau1, tau2;                       /* pares articulares [N-m]         */

/* ---------------------------- Variables auxiliares ----------------------- */

float t;                                /* Tiempo de ejecución [s]         */

/* ------------------------------- Parámetros ------------------------------ */

float q1,q2,q3,q4,qd1,qd2;                          // Estados y trayectorias deseadas
float qt1,qt2,qd1deg,qd2deg;                        // Errores y conversiones
float kT1 = 0.0025;                                 // Delay (kT+1)
float kT2 = 0.005;                                  // Delay (kT+2)
float e1, e2, e3, e4;                               // Errores de identificación [rad]
float e1deg, e2deg, e3deg, e4deg;                   // Errores de identificación [deg]
float ve1, ve2, se1, se2, mse1, mse2;               // MSE
float x1, x2, x3, x4;                               // Condiciones inicales del Identificador
float b1 = Pi / 4, b2 = Pi / 3;                     // Parámetros de las trayectorias
float c1 = Pi / 18, c2 = (25 * Pi) / 36;
float d1 = -2, d2 = -1.8, o1 = 5, o2 = 1;
float w11 = 1/75, w12 = 0, w21 = 1/75, w22 = 0;     // Pesos neuronales iniciales
float w31 = 1, w35 = 1/30, w36 = 0;
float w41 = 1, w45 = 0, w46 = 1/30;
float u01 = 150;                                    // Par max. 1er eslabon
float u02 = 15;                                     // Par max. 2do eslabon

float W1[2] = { 1.4,0.0 };
float W2[2] = { 0.65,0.0 };                           // 2.5>W2[0]>0.8 W2[1]>0
float W3[4] = { -0.312,0.0,0.04,-0.3 };
float W4[4] = { 0.3,0.3,0.5,0.09 };

float B[2][2] = { {75,0 }, { 0,30 } };          // Parámetros del controlador
float k1[2][2] = { {0.3,0},{0,0.03} };
float k2[2][2] = { {0.2,0},{0,0.6} };

float P = 81000, Q = 1.2*18467000, R = 221690000;               // Parámetros del filtro UKF

float P1[2][2] = { {P,0},{0,P} };
float P2[2][2] = { {P,0},{0,P} };
```

```
float P3[4][4] = { {P,0,0,0},{0,P,0,0},{0,0,P,0},{0,0,0,P} };
float P4[4][4] = { {P,0,0,0},{0,P,0,0},{0,0,P,0},{0,0,0,P} };
float Q1[2][2] = { {Q,0},{0,Q} };
float Q2[4][4] = { {Q,0,0,0},{0,Q,0,0},{0,0,Q,0},{0,0,0,Q} };
float L1 = 2, L2 = 4;
float alpha = 0.01, beta = 2, kappa1 = 3 - L1, kappa2 = 3 - L2;
float lambda1 = alpha * alpha*(L1 + kappa1) - L1;
float lambda2 = alpha * alpha*(L2 + kappa2) - L2;
float et = 1 / (2 * (L1 + lambda1));
float etm1[5] = { (lambda1/(lambda1+L1)),et,et,et,et };
float etc1[5] = { (lambda1/(lambda1+L1)) + (1 - alpha*alpha + beta),et,et,et,et };
float etm2[9] = { (lambda2/(lambda2+L2)),et,et,et,et,et,et,et,et };
float etc2[9] = { (lambda2/(lambda2+L2)) + (1 - alpha*alpha + beta),et,et,et,et,et,et,et,et };
float Rk[2], Rk1[2], Rk2[2], Z1[2], Z2[2], Z3[4], Z4[4];
float sP1[2][2], sP2[2][2], sP3[4][4], sP4[4][4];
float xim1[2][5], xim2[2][5], xim3[4][9], xim4[4][9];
float xm1[2], xm2[2], xm3[4], xm4[4], K1[2], K2[2], K3[4], K4[4];
float xix1[2][5], xix2[2][5], xix3[4][9], xix4[4][9];
float xixt1[5][2], xixt2[5][2], xixt3[9][4], xixt4[9][4];
float Xix1[2][2], Xix2[2][2], Xix3[4][4], Xix4[4][4];
float M[2][2], M2[2][2], M3[4][4], M4[4][4];
float Pm1[2][2], Pm2[2][2], Pm3[4][4], Pm4[4][4];
float xiy1[5], xiy2[5], xiy3[9], xiy4[9];
float psm1[5], psm2[5], psm3[9], psm4[9];
float ym1, ym2, ym3, ym4, psx1, psx2, psx3, psx4;
float Py1, Py2, Py3, Py4;
float Px1[2], Px2[2], Px3[4], Px4[4];
float P1k[2][2], P2k[2][2], P3k[4][4], P4k[4][4], W1k[2], W2k[2], W3k[4], W4k[4];
float f11, f12, f21, f22, x1k, x2k, x3k, x4k, wq11, wq21;
float Flk[2], F2k[2], WQ1[2], X1k[2], X1[2], X2[2];
float Z1k[2], Z1k1[2], X2dk[2], X2dk1[2], Z2k[2], Ueq[2];
float psy1[5], psy2[5], psy3[9], psy4[9];
float sm1, sm2, sm3, sm4, srl = sqrt(L1+lambda1);

/* ************************ CUERPO DEL PROGRAMA ************************ */

/* ------------------------------ Conversiones ------------------------------ */

int i,j,k,row,col;
q1 = (pos1*Pi) / 180;
q2 = (pos2*Pi) / 180;
q3 = (vel1*Pi) / 180;
q4 = (vel2*Pi) / 180;
e1deg = (e1*180) / Pi;
e2deg = (e2*180) / Pi;
e3deg = (e3*180) / Pi;
e4deg = (e4*180) / Pi;

/* ------------------------- Cálculo de referencias ------------------------- */

t = Tiempo_real;

qd1 = b1 * (1 - exp(d1*t*t*t)) + c1 * (1 - exp(d1*t*t*t))*sin(o1*t);
qd2 = b2 * (1 - exp(d2*t*t*t)) + c2 * (1 - exp(d2*t*t*t))*sin(o2*t);
qd1deg = (qd1 * 180) / Pi;
qd2deg = (qd2 * 180) / Pi;
```

```
qt1 = qd1deg - pos1;
qt2 = qd2deg - pos2;

Rk2[0] = qd1;
Rk2[1] = qd2;
if (t > kT1)
{
  Rk1[0]=b1*(1-exp(d1*(t-kT1)*(t-kT1)*(t-kT1)))+c1*(1-exp(d1*(t-kT1)*(t-kT1)*(t-kT1)))*sin(o1*t);
  Rk1[1]=b2*(1-exp(d2*(t-kT1)*(t-kT1)*(t-kT1)))+c2*(1-exp(d2*(t-kT1)*(t-kT1)*(t-kT1)))*sin(o2*t);
  Rk[0]=b1*(1-exp(d1*(t-kT2)*(t-kT2)*(t-kT2)))+c1*(1-exp(d1*(t-kT2)*(t-kT2)*(t-kT2)))*sin(o1*t);
  Rk[1]=b2*(1-exp(d2*(t-kT2)*(t-kT2)*(t-kT2)))+c2*(1-exp(d2*(t-kT2)*(t-kT2)*(t-kT2)))*sin(o2*t);
}

Z1[0] = tanh(q1);
Z1[1] = 1;
Z2[0] = tanh(q2);
Z2[1] = 1;

Z3[0] = tanh(q1);     Z3[1] = tanh(q2);
Z3[2] = tanh(q3);     Z3[3] = tanh(q4);
Z4[0] = tanh(q1);     Z4[1] = tanh(q2);
Z4[2] = tanh(q3);     Z4[3] = tanh(q4);

Py1 = R;
Py2 = R;
Py3 = R;
Py4 = R*1E-3; // Cambio 02/04/2019 Py4 = R;

ym1 = 0;
ym2 = 0;
ym3 = 0;
ym4 = 0;

for (row = 0; row < 2; row++)
{
        for (col = 0; col < 2; col++)
        {
                Pm1[row][col] = Q1[row][col];
                Pm2[row][col] = Q1[row][col];
        }
}

for (row = 0; row < 4; row++)
{
        for (col = 0; col < 4; col++)
        {
                Pm3[row][col] = Q2[row][col];
                Pm4[row][col] = Q2[row][col];
        }
}



/* ---------------------- Evaluación de modelo dinámico ---------------------- */

/* ---------------------- Factorización de Cholesky ---------------------- */
```

```
for (i = 0; i < 2; i++)
{
        for (j = 0; j <= i; j++)
        {
                sm1 = 0;
                sm2 = 0;
                if (j == i)
                {
                        for (k = 0; k < j; k++)
                        {
                                sm1 += pow(sP1[j][k], 2);
                                sm2 += pow(sP2[j][k], 2);
                        }
                        sP1[j][j] = sqrt(P1[j][j] - sm1);
                        sP2[j][j] = sqrt(P2[j][j] - sm2);
                }
                else
                {
                        for (int k = 0; k < j; k++)
                        {
                                sm1 += (sP1[i][k] * sP1[j][k]);
                                sm2 += (sP2[i][k] * sP2[j][k]);
                        }
                        sP1[i][j] = (P1[i][j] - sm1) / sP1[j][j];
                        sP2[i][j] = (P2[i][j] - sm2) / sP2[j][j];
                }
        }
}

for (i = 0; i < 4; i++)
{
        for (j = 0; j <= i; j++)
        {
                sm3 = 0;
                sm4 = 0;
                if (j == i)
                {
                        for (k = 0; k < j; k++)
                        {
                                sm3 += pow(sP3[j][k], 2);
                                sm4 += pow(sP4[j][k], 2);
                        }
                        sP3[j][j] = sqrt(P3[j][j] - sm3);
                        sP4[j][j] = sqrt(P4[j][j] - sm4);
                }
                else
                {
                        for (int k = 0; k < j; k++)
                        {
                                sm3 += (sP3[i][k] * sP3[j][k]);
                                sm4 += (sP4[i][k] * sP4[j][k]);
                        }
                        sP3[i][j] = (P3[i][j] - sm3) / sP3[j][j];
                        sP4[i][j] = (P4[i][j] - sm4) / sP4[j][j];
                }
```

```
        }
}

/* ———————————————————— Red neuronal 1 y 2 ———————————————————— */

/* ———————————————————— Generación de puntos sigma ———————————————————— */


for (i = 0; i < 2; i++)
{
        for (j = 0; j < 5; j++)
        {
                if (j %2 == 0)
                        k = 1;
                else
                        k = 0;
                if (j < 3)
                {
                        xim1[i][j] = W1[i] + srl * sP1[i][k];
                        xim2[i][j] = W2[i] + srl * sP2[i][k];
                }
                else
                {
                        xim1[i][j] = W1[i] - srl * sP1[i][k];
                        xim2[i][j] = W2[i] - srl * sP2[i][k];
                }
                if (j == 0)
                {
                        xim1[i][j] = W1[i];
                        xim2[i][j] = W2[i];
                }
        }
}

for (row = 0; row < 2; row++)
{
        xm1[row] = 0;
        for (i = 0; i < 5; i++)
        {
                xm1[row] += xim1[row][i]*etm1[i];
        }
}
xm2[0] = xim2[0][0]*etm1[0] + xim2[0][1]*etm1[1] + xim2[0][2]*etm1[2]
+ xim2[0][3]*etm1[3] + xim2[0][4]*etm1[4];
xm2[1] = xim2[1][0]*etm1[0] + xim2[1][1]*etm1[1] + xim2[1][2]*etm1[2]
+ xim2[1][3]*etm1[3] + xim2[1][4]*etm1[4];

/* ———————————————————— Propagación de puntos sigma ———————————————————— */

for (row = 0; row < 2; row++)
{
        for (i = 0; i < 5; i++)
        {
                xix1[row][i] = xim1[row][i] - xm1[row];
                xix2[row][i] = xim2[row][i] - xm2[row];
                xixt1[i][row] = xix1[row][i];
```

```
                xixt2[i][row] = xix2[row][i];
        }
}

for (row = 0; row < 2; row++)
{
        for (col = 0; col < 2; col++)
        {
                M[row][col] = 0;
                M2[row][col] = 0;
                for (i = 0; i < 5; i++)
                {
                        M[row][col] += xix1[row][i] * xixt1[i][col];
                        M2[row][col] += xix2[row][i] * xixt2[i][col];
                }
        }
}

/* ---------------------------- Predicción del estado ---------------------------- */

for (row = 0; row < 2; row++)
{
        for (col = 0; col < 2; col++)
        {
                for (i = 0; i < 5; i++)
                {
                        Xix1[row][col] += etc1[i] * M[row][col];
                        Xix2[row][col] += etc1[i] * M2[row][col];
                }
        }
}

/* ---------------------------- Predicción de covarianza ---------------------------- */

for (row = 0; row < 2; row++)
{
        for (col = 0; col < 2; col++)
        {
                Pm1[row][col] += Xix1[row][col];
                Pm2[row][col] += Xix2[row][col];
        }
}

/* ---------------------------- Propagación de puntos sigma ---------------------------- */

for (i = 0; i < 5; i++)
{
        xiy1[i] = (xim1[0][i] * Z1[0] + xim1[1][i] * Z1[1]) + w11 * q3;
        xiy2[i] = (xim2[0][i] * Z2[0] + xim2[1][i] * Z2[1]) + w21 * q4;
        psm1[i] = xiy1[i] - x1;
        psm2[i] = xiy2[i] - x2;
}

/* ---------------------------- Predicción de salida ---------------------------- */

for (i = 0; i < 5; i++)
```

```
{
        ym1 += etm1[i] * psm1[i];
        ym2 += etm1[i] * psm2[i];
}

for (i = 0; i < 5; i++)
{
        psy1[i] = psm1[i] - ym1;
        psy2[i] = psm2[i] - ym2;
}

/* ------------------------ Predicción de covarianza ------------------------ */

for (i = 0; i < 5; i++)
{
        Py1 += etc1[i] * (psy1[i]*psy1[i]);
        Py2 += etc1[i] * (psy2[i]*psy2[i]);
}

for (row = 0; row < 2; row++)
{
          for (i = 0; i < 5; i++)
          {
                  Px1[row] += (etc1[i] * xix1[row][i])*psy1[i];
                  Px2[row] += (etc1[i] * xix2[row][i])*psy2[i];
          }
}

/* ------------------------------ Actualización ------------------------------ */

for (row = 0; row < 2; row++)
{
        K1[row] = Px1[row]/Py1;
        K2[row] = Px2[row]/Py2;
        W1k[row] = W1[row] + K1[row] * e1;
        W2k[row] = W2[row] + K2[row] * e2;
}

for(i = 0;i < 2; i++)
{
        for(j = 0;j < 2; j++)
        {
            P1k[i][j] = Pm1[i][j] - (K1[i] * Py1 * K1[j]);
            P2k[i][j] = Pm2[i][j] - (K2[i] * Py2 * K2[j]);
        }
}

/* ------------------------------ Red neuronal 3 y 4 ------------------------------ */

/* ------------------------ Generación de puntos sigma ------------------------ */

for (i = 0; i < 4; i++)
{
        for (j = 0; j < 9; j++)
        {
                k = 0;
```

```
                if (j > 0 && j < 5)
                        k = j - 1;
                else
                        k = j - 5;
                if (j < 5)
                {
                        xim3[i][j] = W3[i] + srl * sP3[i][k];
                        xim4[i][j] = W4[i] + srl * sP4[i][k];
                }
                else
                {
                        xim3[i][j] = W3[i] - srl * sP3[i][k];
                        xim4[i][j] = W4[i] - srl * sP4[i][k];
                }
                if (j == 0)
                {
                        xim3[i][j] = W3[i];
                        xim4[i][j] = W4[i];
                }
        }
}


for (i = 0; i < 9; i++)
{
        xm3[0] += etm2[i] * xim3[0][i];
        xm3[1] += etm2[i] * xim3[1][i];
        xm3[2] += etm2[i] * xim3[2][i];
        xm3[3] += etm2[i] * xim3[3][i];
        xm4[0] += etm2[i] * xim4[0][i];
        xm4[1] += etm2[i] * xim4[1][i];
        xm4[2] += etm2[i] * xim4[2][i];
        xm4[3] += etm2[i] * xim4[3][i];
}

/* ---------------------- Propagación de puntos sigma ---------------------- */

for (row = 0; row < 4; row++)
{
        for (i = 0; i < 9; i++)
        {
                xix3[row][i] = xim3[row][i] - xm3[row];
                xix4[row][i] = xim4[row][i] - xm4[row];
                xixt3[i][row] = xix3[row][i];
                xixt4[i][row] = xix4[row][i];
        }
}

for (row = 0; row < 4; row++)
{
        for (col = 0; col < 4; col++)
        {
                M3[row][col] = 0;
                M4[row][col] = 0;
                for (i = 0; i < 9; i++)
                {
```

```
                        M3[row][col] += xix3[row][i] * xixt3[i][col];
                        M4[row][col] += xix4[row][i] * xixt4[i][col];
                }
        }
}

/* ————————————————————— Predicción del estado ————————————————————— */

for (row = 0; row < 4; row++)
{
        for (col = 0; col < 4; col++)
        {
                for (i = 0; i < 9; i++)
                {
                        Xix3[row][col] += etc2[i] * M3[row][col];
                        Xix4[row][col] += etc2[i] * M4[row][col];
        }
        }
}

/* ————————————————————— Predicción de covarianza ————————————————————— */



for (row = 0; row < 4; row++)
{
        for (col = 0; col < 4; col++)
        {
                Pm3[row][col] += Xix3[row][col];
                Pm4[row][col] += Xix4[row][col];
        }
}

/* ————————————————————— Propagación de puntos sigma ————————————————————— */

for (i = 0; i < 9; i++)
{
   xiy3[i] = (xim3[0][i] * Z3[0] + xim3[1][i] * Z3[1] + xim3[2][i] * Z3[2] + xim3[3][i] * Z3[3]) +
   +0.01 * (q3 + tau1);
   xiy4[i] = (xim4[0][i] * Z4[0] + xim4[1][i] * Z4[1] + xim4[2][i] * Z4[2] + xim4[3][i] * Z4[3]) +
   +0.01 * (q4 + tau2);
   psm3[i] = xiy3[i] - x3;
   psm4[i] = xiy4[i] - x4;
}

/* ————————————————————— Predicción de salida ————————————————————— */

for (i = 0; i < 9; i++)
{
        ym3 += etm2[i] * psm3[i];
        ym4 += etm2[i] * psm4[i];
}

for (i = 0; i < 9; i++)
{
        psy3[i] = psm3[i] - ym3;
```

```
        psy4[i] = psm4[i] - ym4;
}

/* ---------------------- Predicción de covarianza ---------------------- */

for (i = 0; i < 9; i++)
{
        Py3 += etc2[i] * (psy3[i] * psy3[i]);
        Py4 += etc2[i] * (psy4[i] * psy4[i]);
}

for (row = 0; row < 4; row++)
{
        for (i = 0; i < 9; i++)
        {
                Px3[row] += etc2[i] * xix3[row][i] * psy3[i];
                Px4[row] += etc2[i] * xix4[row][i] * psy4[i];
        }
}

/* ------------------------------ Actualización ------------------------------ */

for (row = 0; row < 4; row++)
{
        K3[row] = Px3[row] / Py3;
        K4[row] = Px4[row] / Py4;
        W3k[row] = W3[row] + (K3[row] * e3);
        W4k[row] = W4[row] + (K4[row] * e4);
}

for(i = 0;i < 4; i++)
{
        for(j = 0;j < 4; j++)
        {
        P3k[i][j] = Pm3[i][j] - (K3[i] * Py3 * K3[j]);
        P4k[i][j] = Pm4[i][j] - (K4[i] * Py4 * K4[j]);
        }
}

/* ------------------------------ Ley de control ------------------------------ */

f11 = W1[0] * Z1[0] + W1[1] * Z1[1];
f12 = W2[0] * Z2[0] + W2[1] * Z2[1];
f21 = (W3[0] * Z3[0] + W3[1] * Z3[1] + W3[2] * Z3[2] + W3[3] * Z3[3]) + w31 * q3;
f22 = (W4[0] * Z4[0] + W4[1] * Z4[1] + W4[2] * Z4[2] + W4[3] * Z4[3]) + w41 * q4;
x1k = w11 * q3 + f11;
x2k = w21 * q4 + f12;
x3k = f21 + w35 * tau1;
x4k = f22 + w46 * tau2;
wq11 = tanh(q1 + q3 * kT1)*W1k[0] + W1k[1];
wq21 = tanh(q2 + q4 * kT1)*W2k[0] + W2k[1];

Flk[0] = f11;
Flk[1] = f12;
X1k[0] = x1k;
X1k[1] = x2k;
```

```
WQ1[0] = wq11;
WQ1[1] = wq21;
F2k[0] = f21;
F2k[1] = f22;
X1[0] = x1;
X1[1] = x2;
X2[0] = x3;
X2[1] = x4;
Z1k[0] = X1[0] - Rk[0];
Z1k[1] = X1[1] - Rk[1];
Z1k1[0] = X1k[0] - Rk1[0];
Z1k1[1] = X1k[1] - Rk1[1];
X2dk[0] = B[0][0] * (-F1k[0] + (k1[0][0] * Z1k[0]) + Rk1[0]);
X2dk[1] = B[1][1] * (-F1k[1] + (k1[1][1] * Z1k[1]) + Rk1[1]);
Z2k[0] = X2[0] - X2dk[0];
Z2k[1] = X2[1] - X2dk[1];
X2dk1[0] = B[0][0] * (-WQ1[0] + (k1[0][0] * Z1k1[0]) + Rk2[0]);
X2dk1[1] = B[1][1] * (-WQ1[1] + (k1[1][1] * Z1k1[1]) + Rk2[1]);

Ueq[0] = B[0][0] * (-F2k[0] + (k2[0][0] * Z2k[0]) + X2dk1[0]);
Ueq[1] = B[1][1] * (-F2k[1] + (k2[1][1] * Z2k[1]) + X2dk1[1]);

if (abs(Ueq[0]) > u01)
{
        tau1 = u01 * (Ueq[0] / (abs(Ueq[0])));
}
else
{
        tau1 = Ueq[0];
}
if (abs(Ueq[1]) > u02)
{
        tau2 = u02 * (Ueq[1] / (abs(Ueq[1])));
}
else
{
        tau2 = Ueq[1];
}

if (t > kT1)
{
    x1 = x1k;
    x2 = x2k;
    x3 = x3k;
    x4 = x4k;

    e1 = q1 - x1;
    e2 = q2 - x2;
    e3 = q3 - x3;
    e4 = q4 - x4;

    ve1 += qt1*qt1;
    ve2 += qt2*qt2;

    se1 = (ve1*kT1)/t;
    se2 = (ve2*kT1)/t;
```

```
    mse1 = sqrt(se1);
    mse2 = sqrt(se2);

    for (i = 0; i < 2; ++i)
    {
        W1[i] = W1k[i];
        W2[i] = W2k[i];
        Px1[i] = 0;
        Px2[i] = 0;
    }

    for (i = 0; i < 4; ++i)
    {
        W3[i] = W3k[i];
        W4[i] = W4k[i];
        xm3[i] = 0;
        xm4[i] = 0;
        Px3[i] = 0;
        Px4[i] = 0;
    }

   for (row = 0; row < 2; row++)
   {
        for (col = 0; col < 2; col++)
        {
                Xix1[row][col] = 0;
                Xix2[row][col] = 0;
                P1[row][col] = P1k[row][col];
                P2[row][col] = P2k[row][col];
        }
   }
   for (row = 0; row < 4; row++)
   {
        for (col = 0; col < 4; col++)
        {
                Xix3[row][col] = 0;
                Xix4[row][col] = 0;
                P3[i][j] = P3k[i][j];
                P4[i][j] = P4k[i][j];
        }
   }
}
/* ************************** FIN DEL PROGRAMA ************************** */
```

## C.3. Control descentralizado neuronal con entrenamiento EKF (WinMechLab)

```
/* ********************************************************************** */
/* Archivo:       DNBC_EKF.rob                                            */
/* Descripción:   Control descentralizado neuronal con entrenamiento EKF  */
/* Autor:         Juan F. Guerra Cano                                     */
/* Creado:        27/11/2018                                              */
/* Modificado:    02/04/2019                                              */
/* ********************************************************************** */

/* ***************** DECLARACION DE VARIABLES Y PARAMETROS ************** */

/* ------------------------- Variables principales ------------------------- */

float pos1, pos2;                       /* posiciones articulares [deg]    */
float vel1, vel2;                       /* velocidades articulares [deg/s] */
float tau1, tau2;                       /* pares articulares [N-m]         */

/* ------------------------- Variables auxiliares -------------------------- */

float t;                                /* Tiempo de ejecución [s]         */

float q1,q2,q3,q4,qd1,qd2;              /* Variables de estado             */
float qd1k,qd1deg,qd2deg,qt1,qt2;       /* Conversiones                    */
float delay1 = 0.0025 , delay2 = 0.005; /* Retardos [kT+1] [kT+2]          */
float e1, e2, e3, e4;                   /* Errores de identificación [rad] */
float e1deg, e2deg, e3deg, e4deg;       /* Errores de identificación [deg] */
float x1, x2, x3, x4;                   /* C.I del Identificador neuronal  */

float ve1, ve2, se1, se2, mse1, mse2;   // MSE

float b1 = Pi / 4, b2 = Pi / 3;         /* Parámetros de las trayectorias  */
float c1 = Pi / 18, c2 = (25 * Pi) / 36;
float d1 = -2, d2 = -1.8;
float o1 = 5, o2 = 1;

float u01 = 150.0;                      /* Límite de par eslabón 1         */
float u02 = 15.0;                       /* Límite de par eslabón 2         */

float w11 = 0.01, w12 = 0;              /* C.I. de los pesos neuronales    */
float w21 = 0.01, w22 = 0;
float w31 = 1, w35 = 0.01, w36 = 0;
float w41 = 1, w45 = 0, w46 = 0.01;

float W1[2] = { 1.4,0.0 };
float W2[2] = { 0.65,0.0 };                     // 2.5>W2[0]>0.8 W2[1]>0
float W3[4] = { -0.312,0.0,0.04,-0.3 };
float W4[4] = { 0.3,0.3,0.5,0.09 };

/* ---------------------- Parámetros del filtro EKF ---------------------- */

float eta1 = 0.5, eta2 = 0.5;
float eta3 = 0.9, eta4 = 0.9;
```

```
float B[2][2] = { {75,0 }, { 0,30 } }; //50,25
float k1[2][2] = { {0.3,0},{0,0.03} };
float k2[2][2] = { {0.2,0},{0,0.6} };

float P = 81000, Q = 1.2*18467000, R = 221690000;
float Rp = R*10;                                    // R3 y R4  >> R1 y R2  10:1 aprox.

float P1[2][2] = { {P,0},{0,P} };
float P2[2][2] = { {P,0},{0,P} };
float P3[4][4] = { {P,0,0,0},{0,P,0,0},{0,0,P,0},{0,0,0,P} };
float P4[4][4] = { {P,0,0,0},{0,P,0,0},{0,0,P,0},{0,0,0,P} };

float Q1[2][2] = { {Q,0},{0,Q} };
float Q2[4][4] = { {Q,0,0,0},{0,Q,0,0},{0,0,Q,0},{0,0,0,Q} };

float R1[2][2] = { {R,0},{0,R} };
float R2[4][4] = { {Rp,0,0,0},{0,Rp,0,0},{0,0,Rp,0},{0,0,0,Rp} };

float Rk[2], Rk1[2], Rk2[2];
float Z1[2], Z2[2], Z3[4], Z4[4];
float tM1[2][2], M1[2][2], tM2[2][2], M2[2][2], K1[2], K2[2];
float tM3[4][4], tM4[4][4], M3[4][4], M4[4][4], K3[4], K4[4];
float det1, det2, det3, det4, det1k = 1, det2k = 1, det3k = 1, det4k = 1;
float P1k[2][2], P2k[2][2], P3k[4][4], P4k[4][4];
float W1k[2], W2k[2], W3k[4], W4k[4];
float adj3[4][4], adj4[4][4], tK3[4], tK4[4];
float f11, f12, f21, f22, x1k, x2k, x3k, x4k, wq11, wq21;
float F1k[2], F2k[2], WQ1[2], X1k[2], X1[2], X2[2];
float Z1k[2], Z1k1[2], X2desk[2], X2desk1[2], Z2k[2], Ueq[2];


/* ************************* CUERPO DEL PROGRAMA ************************* */


/* ------------------------------ Conversiones ------------------------------ */

int i,j,k,row,col;

q1 = (pos1*Pi)/180;
q2 = (pos2*Pi)/180;
q3 = (vel1*Pi)/180;
q4 = (vel2*Pi)/180;

e1deg = (e1*180)/Pi;
e2deg = (e2*180)/Pi;
e3deg = (e3*180)/Pi;
e4deg = (e4*180)/Pi;

/* ------------------------- Cálculo de referencias ------------------------- */

t = Tiempo_real;

qd1 = b1 * (1 - exp(d1*t*t*t)) + c1 * (1 - exp(d1*t*t*t))*sin(o1*t);
qd2 = b2 * (1 - exp(d2*t*t*t)) + c2 * (1 - exp(d2*t*t*t))*sin(o2*t);

qd1deg = (qd1*180)/Pi;
```

```
qd2deg = (qd2*180)/Pi;

qt1 = qd1deg - pos1;
qt2 = qd2deg - pos2;

Rk2[0] = qd1;
Rk2[1] = qd2;

if (t > delay1)
{
        Rk1[0] = b1*(1-exp(d1*(t-delay1)*(t-delay1)*(t-delay1)))
        + c1*(1-exp(d1*(t-delay1)*(t-delay1)*(t-delay1)))*sin(o1*t);

        Rk1[1] = b2*(1-exp(d2*(t-delay1)*(t-delay1)*(t-delay1)))
        + c2*(1-exp(d2*(t-delay1)*(t-delay1)*(t-delay1)))*sin(o2*t);

        Rk[0] = b1*(1-exp(d1*(t-delay2)*(t-delay2)*(t-delay2)))
        + c1*(1-exp(d1*(t-delay2)*(t-delay2)*(t-delay2)))*sin(o1*t);

        Rk[1] = b2*(1-exp(d2*(t-delay2)*(t-delay2)*(t-delay2)))
        + c2*(1-exp(d2*(t-delay2)*(t-delay2)*(t-delay2)))*sin(o2*t);
}

Z1[0] = tanh(q1); Z1[1] = 1;
Z2[0] = tanh(q2); Z2[1] = 1;
Z3[0] = tanh(q1); Z3[1] = tanh(q2);
Z3[2] = tanh(q3); Z3[3] = tanh(q4);
Z4[0] = tanh(q1); Z4[1] = tanh(q2);
Z4[2] = tanh(q3); Z4[3] = tanh(q4);

/* ---------- Identificación de modelo dinámico ---------- */

/*  Ecuaciones de Kalman Neuronas 1 y 2 */

tM1[0][0] = ((Z1[0]*P1[0][0] + Z1[1]*P1[1][0])*Z1[0]
+ (Z1[0]*P1[0][1] + Z1[1]*P1[1][1])*Z1[1]) + R1[0][0];

tM2[0][0] = ((Z2[0]*P2[0][0] + Z2[1]*P2[1][0])*Z2[0]
+ (Z2[0]*P2[0][1] + Z2[1]*P2[1][1])*Z2[1]) + R1[0][0];

tM1[0][1] = ((Z1[0]*P1[0][0] + Z1[1]*P1[1][0])*Z1[0]
+ (Z1[0]*P1[0][1] + Z1[1]*P1[1][1])*Z1[1]) + R1[0][1];

tM2[0][1] = ((Z2[0]*P2[0][0] + Z2[1]*P2[1][0])*Z2[0]
+ (Z2[0]*P2[0][1] + Z2[1]*P2[1][1])*Z2[1]) + R1[0][1];

tM1[1][0] = ((Z1[0]*P1[0][0] + Z1[1]*P1[1][0])*Z1[0]
+ (Z1[0]*P1[0][1] + Z1[1]*P1[1][1])*Z1[1]) + R1[1][0];

tM2[1][0] = ((Z2[0]*P2[0][0] + Z2[1] * P2[1][0])*Z2[0]
+ (Z2[0]*P2[0][1] + Z2[1]*P2[1][1])*Z2[1]) + R1[1][0];

tM1[1][1] = ((Z1[0] * P1[0][0] + Z1[1]*P1[1][0])*Z1[0]
+ (Z1[0]*P1[0][1] + Z1[1]*P1[1][1])*Z1[1]) + R1[1][1];

tM2[1][1] = ((Z2[0]*P2[0][0] + Z2[1]*P2[1][0])*Z2[0]
```

```
+ (Z2[0]*P2[0][1] + Z2[1]*P2[1][1])*Z2[1]) + R1[1][1];

det1 = (tM1[0][0]*tM1[1][1]) - (tM1[0][1]*tM1[1][0]);
det2 = (tM2[0][0]*tM2[1][1]) - (tM2[0][1]*tM2[1][0]);

if (det1==0)
{
    det1=det1k;
}

if (det2==0)
{
    det2=det2k;
}

M1[0][0] =  tM1[1][1]/det1;
M1[0][1] = -tM1[0][1]/det1;
M1[1][0] = -tM1[1][0]/det1;
M1[1][1] =  tM1[0][0]/det1;

M2[0][0] =  tM2[1][1]/det2;
M2[0][1] = -tM2[0][1]/det2;
M2[1][0] = -tM2[1][0]/det2;
M2[1][1] =  tM2[0][0]/det2;

K1[0] = M1[0][0]*(P1[0][0]*Z1[0] + P1[0][1]*Z1[1])
+ M1[0][1]*(P1[1][0]*Z1[0] + P1[1][1]*Z1[1]);

K1[1] = M1[1][0]*(P1[0][0]*Z1[0] + P1[0][1]*Z1[1])
+ M1[1][1]*(P1[1][0]*Z1[0] + P1[1][1]*Z1[1]);

K2[0] = M2[0][0]*(P2[0][0]*Z2[0] + P2[0][1]*Z2[1])
+ M2[0][1]*(P2[1][0]*Z2[0] + P2[1][1]*Z2[1]);

K2[1] = M2[1][0]*(P2[0][0]*Z2[0] + P2[0][1]*Z2[1])
+ M2[1][1]*(P2[1][0]*Z2[0] + P2[1][1]*Z2[1]);

P1k[0][0] = P1[0][0] - (K1[0]*Z1[0]*P1[0][0] + K1[0]*Z1[1]*P1[1][0])
+ Q1[0][0];

P1k[0][1] = P1[0][1] - (K1[0]*Z1[0]*P1[0][1] + K1[0]*Z1[1]*P1[1][1])
+ Q1[0][1];

P1k[1][0] = P1[1][0] - (K1[1]*Z1[0]*P1[0][0] + K1[1]*Z1[1]*P1[1][0])
+ Q1[1][0];

P1k[1][1] = P1[1][1] - (K1[1]*Z1[0]*P1[0][1] + K1[1]*Z1[1]*P1[1][1])
+ Q1[1][1];

P2k[0][0] = P2[0][0] - (K2[0]*Z2[0]*P2[0][0] + K2[0]*Z2[1]*P2[1][0])
+ Q1[0][0];

P2k[0][1] = P2[0][1] - (K2[0]*Z2[0]*P2[0][1] + K2[0]*Z2[1]*P2[1][1])
+ Q1[0][1];

P2k[1][0] = P2[1][0] - (K2[1]*Z2[0]*P2[0][0] + K2[1]*Z2[1]*P2[1][0])
```

```
+ Q1[1][0];

P2k[1][1] = P2[1][1] - (K2[1]*Z2[0]*P2[0][1] + K2[1]*Z2[1]*P2[1][1])
+ Q1[1][1];

W1k[0] = (eta1 * K1[0] * e1) + W1[0];
W2k[0] = (eta2 * K2[0] * e2) + W2[0];
W1k[1] = (eta1 * K1[1] * e1) + W1[1];
W2k[1] = (eta2 * K2[1] * e2) + W2[1];

/*  Ecuaciones de Kalman Neurona 3 */

for (i = 0; i < 4; i++)
{
        for (j = 0; j < 4; j++)
        {
                tM3[i][j] = ((Z3[0]*P3[0][0] + Z3[1]*P3[1][0] + Z3[2]*P3[2][0]
                + Z3[3]*P3[3][0])*Z3[0] + (Z3[0]*P3[0][1] + Z3[1]*P3[1][1]
                + Z3[2]*P3[2][1] + Z3[3]*P3[3][1])*Z3[1] + (Z3[0]*P3[0][2]
                + Z3[1]*P3[1][2] + Z3[2]*P3[2][2] + Z3[3]*P3[3][2])*Z3[2]
                + (Z3[0]*P3[0][3] + Z3[1]*P3[1][3] + Z3[2]*P3[2][3]
                + Z3[3]*P3[3][3])*Z3[3]) + R2[i][j];
        }
}

adj3[0][0] = tM3[1][1]*tM3[2][2]*tM3[3][3] - tM3[1][1]*tM3[2][3]*tM3[3][2]
- tM3[2][1]*tM3[1][2]*tM3[3][3] + tM3[2][1]*tM3[1][3]*tM3[3][2]
+ tM3[3][1]*tM3[1][2]*tM3[2][3] - tM3[3][1]*tM3[1][3]*tM3[2][2];

adj3[1][0] = -tM3[1][0]*tM3[2][2]*tM3[3][3] + tM3[1][0]*tM3[2][3]*tM3[3][2]
+ tM3[2][0]*tM3[1][2]*tM3[3][3] - tM3[2][0]*tM3[1][3]*tM3[3][2]
- tM3[3][0]*tM3[1][2]*tM3[2][3] + tM3[3][0]*tM3[1][3]*tM3[2][2];

adj3[2][0] = tM3[1][0]*tM3[2][1]*tM3[3][3] - tM3[1][0]*tM3[2][3]*tM3[3][1]
- tM3[2][0]*tM3[1][1]*tM3[3][3] + tM3[2][0]*tM3[1][3]*tM3[3][1]
+ tM3[3][0]*tM3[1][1]*tM3[2][3] - tM3[3][0]*tM3[1][3]*tM3[2][1];

adj3[3][0] = -tM3[1][0]*tM3[2][1]*tM3[3][2] + tM3[1][0]*tM3[2][2]*tM3[3][1]
+ tM3[2][0]*tM3[1][1]*tM3[3][2] - tM3[2][0]*tM3[1][2]*tM3[3][1]
- tM3[3][0]*tM3[1][1]*tM3[2][2] + tM3[3][0]*tM3[1][2]*tM3[2][1];

adj3[0][1] = -tM3[0][1]*tM3[2][2]*tM3[3][3] + tM3[0][1]*tM3[2][3]*tM3[3][2]
+ tM3[2][1]*tM3[0][2]*tM3[3][3] - tM3[2][1]*tM3[0][3]*tM3[3][2]
- tM3[3][1]*tM3[0][2]*tM3[2][3] + tM3[3][1]*tM3[0][3]*tM3[2][2];

adj3[1][1] = tM3[0][0]*tM3[2][2]*tM3[3][3] - tM3[0][0]*tM3[2][3]*tM3[3][2]
- tM3[2][0]*tM3[0][2]*tM3[3][3] + tM3[2][0]*tM3[0][3]*tM3[3][2]
+ tM3[3][0]*tM3[0][2]*tM3[2][3] - tM3[3][0]*tM3[0][3]*tM3[2][2];

adj3[2][1] = -tM3[0][0]*tM3[2][1]*tM3[3][3] + tM3[0][0]*tM3[2][3]*tM3[3][1]
+ tM3[2][0]*tM3[0][1]*tM3[3][3] - tM3[2][0]*tM3[0][3]*tM3[3][1]
- tM3[3][0]*tM3[0][1]*tM3[2][3] + tM3[3][0]*tM3[0][3]*tM3[2][1];

adj3[3][1] = tM3[0][0]*tM3[2][1]*tM3[3][2] - tM3[0][0]*tM3[2][2]*tM3[3][1]
- tM3[2][0]*tM3[0][1]*tM3[3][2] + tM3[2][0]*tM3[0][2]*tM3[3][1]
+ tM3[3][0]*tM3[0][1]*tM3[2][2] - tM3[3][0]*tM3[0][2]*tM3[2][1];
```

```
adj3[0][2] = tM3[0][1]*tM3[1][2]*tM3[3][3] - tM3[0][1]*tM3[1][3]*tM3[3][2]
- tM3[1][1]*tM3[0][2]*tM3[3][3] + tM3[1][1]*tM3[0][3]*tM3[3][2]
+ tM3[3][1]*tM3[0][2]*tM3[1][3] - tM3[3][1]*tM3[0][3]*tM3[1][2];

adj3[1][2] = -tM3[0][0]*tM3[1][2]*tM3[3][3] + tM3[0][0]*tM3[1][3]*tM3[3][2]
+ tM3[1][0]*tM3[0][2]*tM3[3][3] - tM3[1][0]*tM3[0][3]*tM3[3][2]
- tM3[3][0]*tM3[0][2]*tM3[1][3] + tM3[3][0]*tM3[0][3]*tM3[1][2];

adj3[2][2] = tM3[0][0]*tM3[1][1]*tM3[3][3] - tM3[0][0]*tM3[1][3]*tM3[3][1]
- tM3[1][0]*tM3[0][1]*tM3[3][3] + tM3[1][0]*tM3[0][3]*tM3[3][1]
+ tM3[3][0]*tM3[0][1]*tM3[1][3] - tM3[3][0]*tM3[0][3]*tM3[1][1];

adj3[3][2] = -tM3[0][0]*tM3[1][1]*tM3[3][2] + tM3[0][0]*tM3[1][2]*tM3[3][1]
+ tM3[1][0]*tM3[0][1]*tM3[3][2] - tM3[1][0]*tM3[0][2]*tM3[3][1]
- tM3[3][0]*tM3[0][1]*tM3[1][2] + tM3[3][0]*tM3[0][2]*tM3[1][1];

adj3[0][3] = -tM3[0][1]*tM3[1][2]*tM3[2][3] + tM3[0][1]*tM3[1][3]*tM3[2][2]
+ tM3[1][1]*tM3[0][2]*tM3[2][3] - tM3[1][1]*tM3[0][3]*tM3[2][2]
- tM3[2][1]*tM3[0][2]*tM3[1][3] + tM3[2][1]*tM3[0][3]*tM3[1][2];

adj3[1][3] = tM3[0][0]*tM3[1][2]*tM3[2][3] - tM3[0][0]*tM3[1][3]*tM3[2][2]
- tM3[1][0]*tM3[0][2]*tM3[2][3] + tM3[1][0]*tM3[0][3]*tM3[2][2]
+ tM3[2][0]*tM3[0][2]*tM3[1][3] - tM3[2][0]*tM3[0][3]*tM3[1][2];

adj3[2][3] = -tM3[0][0]*tM3[1][1]*tM3[2][3] + tM3[0][0]*tM3[1][3]*tM3[2][1]
+ tM3[1][0]*tM3[0][1]*tM3[2][3] - tM3[1][0]*tM3[0][3]*tM3[2][1]
- tM3[2][0]*tM3[0][1]*tM3[1][3] + tM3[2][0]*tM3[0][3]*tM3[1][1];

adj3[3][3] = tM3[0][0]*tM3[1][1]*tM3[2][2] - tM3[0][0]*tM3[1][2]*tM3[2][1]
- tM3[1][0]*tM3[0][1]*tM3[2][2] + tM3[1][0]*tM3[0][2]*tM3[2][1]
+ tM3[2][0]*tM3[0][1]*tM3[1][2] - tM3[2][0]*tM3[0][2]*tM3[1][1];

det3 = tM3[0][0]*tM3[1][1]*tM3[2][2]*tM3[3][3] + tM3[0][1]*tM3[1][2]*tM3[2][3]*tM3[3][0]
+ tM3[0][2]*tM3[1][3]*tM3[2][0]*tM3[3][1] + tM3[0][3]*tM3[1][0]*tM3[2][1]*tM3[3][2]
- tM3[3][0]*tM3[2][1]*tM3[1][2]*tM3[0][3] - tM3[3][1]*tM3[2][2]*tM3[1][3]*tM3[0][0]
- tM3[3][2]*tM3[2][3]*tM3[1][0]*tM3[0][1] - tM3[3][3]*tM3[2][0]*tM3[1][1]*tM3[0][2];

if (det3==0)
{
    det3=det3k;
}

for (row = 0; row < 4; ++row)
{
        for (col = 0; col < 4; ++col)
        {
                M3[row][col] = adj3[row][col] / det3;
        }
}

for (row = 0; row < 4; row++)
{
        tK3[row] = 0;
        for (col = 0; col < 4; col++)
        {
```

```
                tK3[row] += P3[row][col]*Z3[col];
        }
}

for (i = 0; i < 4; i++)
{
        K3[i] = 0;
        for (j = 0; j < 4; j++)
        {
                K3[i] += M3[i][j]*tK3[j];
        }
}

for (i = 0; i < 4; i++)
{
        for (j = 0; j < 4; j++)
        {
                P3k[i][j] = P3[i][j] - (K3[i]*Z3[0]*P3[0][j]
                + K3[0]*Z3[1]*P3[1][j] + K3[0]*Z3[2]*P3[2][j]
                + K3[0]*Z3[3]*P3[3][j]) + Q2[i][j];
        }
}

for (i = 0; i < 4; i++)
{
        W3k[i] = eta3*K3[i]*e3 + W3[i];
}

/* ———————————— Identificación de modelo dinámico ———————————— */

/*  Ecuaciones de Kalman Neurona 4 */

for (i = 0; i < 4; i++)
{
        for (j = 0; j < 4; j++)
        {
                tM4[i][j] = ((Z4[0]*P4[0][0] + Z4[1]*P4[1][0] + Z4[2]*P4[2][0]
                + Z4[3]*P4[3][0])*Z4[0] + (Z4[0]*P4[0][1] + Z4[1]*P4[1][1]
                + Z4[2]*P4[2][1] + Z4[3]*P4[3][1])*Z4[1] + (Z4[0]*P4[0][2]
                + Z4[1]*P4[1][2] + Z4[2]*P4[2][2] + Z4[3]*P4[3][2])*Z4[2]
                + (Z4[0]*P4[0][3] + Z4[1]*P4[1][3] + Z4[2]*P4[2][3]
                + Z4[3]*P4[3][3])*Z4[3]) + R2[i][j];
        }
}

adj4[0][0] = tM4[1][1]*tM4[2][2]*tM4[3][3] - tM4[1][1]*tM4[2][3]*tM4[3][2]
- tM4[2][1]*tM4[1][2]*tM4[3][3] + tM4[2][1]*tM4[1][3]*tM4[3][2]
+ tM4[3][1]*tM4[1][2]*tM4[2][3] - tM4[3][1]*tM4[1][3]*tM4[2][2];

adj4[1][0] = -tM4[1][0]*tM4[2][2]*tM4[3][3] + tM4[1][0]*tM4[2][3]*tM4[3][2]
+ tM4[2][0]*tM4[1][2]*tM4[3][3] - tM4[2][0]*tM4[1][3]*tM4[3][2]
- tM4[3][0]*tM4[1][2]*tM4[2][3] + tM4[3][0]*tM4[1][3]*tM4[2][2];

adj4[2][0] = tM4[1][0]*tM4[2][1]*tM4[3][3] - tM4[1][0]*tM4[2][3]*tM4[3][1]
- tM4[2][0]*tM4[1][1]*tM4[3][3] + tM4[2][0]*tM4[1][3]*tM4[3][1]
+ tM4[3][0]*tM4[1][1]*tM4[2][3] - tM4[3][0]*tM4[1][3]*tM4[2][1];
```

```
adj4[3][0] = -tM4[1][0]*tM4[2][1]*tM4[3][2] + tM4[1][0]*tM4[2][2]*tM4[3][1]
+ tM4[2][0]*tM4[1][1]*tM4[3][2] - tM4[2][0]*tM4[1][2]*tM4[3][1]
- tM4[3][0]*tM4[1][1]*tM4[2][2] + tM4[3][0]*tM4[1][2]*tM4[2][1];

adj4[0][1] = -tM4[0][1]*tM4[2][2]*tM4[3][3] + tM4[0][1]*tM4[2][3]*tM4[3][2]
+ tM4[2][1]*tM4[0][2]*tM4[3][3] - tM4[2][1]*tM4[0][3]*tM4[3][2]
- tM4[3][1]*tM4[0][2]*tM4[2][3] + tM4[3][1]*tM4[0][3]*tM4[2][2];

adj4[1][1] = tM4[0][0]*tM4[2][2]*tM4[3][3] - tM4[0][0]*tM4[2][3]*tM4[3][2]
- tM4[2][0]*tM4[0][2]*tM4[3][3] + tM4[2][0]*tM4[0][3]*tM4[3][2]
+ tM4[3][0]*tM4[0][2]*tM4[2][3] - tM4[3][0]*tM4[0][3]*tM4[2][2];

adj4[2][1] = -tM4[0][0]*tM4[2][1]*tM4[3][3] + tM4[0][0]*tM4[2][3]*tM4[3][1]
+ tM4[2][0]*tM4[0][1]*tM4[3][3] - tM4[2][0]*tM4[0][3]*tM4[3][1]
- tM4[3][0]*tM4[0][1]*tM4[2][3] + tM4[3][0]*tM4[0][3]*tM4[2][1];

adj4[3][1] = tM4[0][0]*tM4[2][1]*tM4[3][2] - tM4[0][0]*tM4[2][2]*tM4[3][1]
- tM4[2][0]*tM4[0][1]*tM4[3][2] + tM4[2][0]*tM4[0][2]*tM4[3][1]
+ tM4[3][0]*tM4[0][1]*tM4[2][2] - tM4[3][0]*tM4[0][2]*tM4[2][1];

adj4[0][2] = tM4[0][1]*tM4[1][2]*tM4[3][3] - tM4[0][1]*tM4[1][3]*tM4[3][2]
- tM4[1][1]*tM4[0][2]*tM4[3][3] + tM4[1][1]*tM4[0][3]*tM4[3][2]
+ tM4[3][1]*tM4[0][2]*tM4[1][3] - tM4[3][1]*tM4[0][3]*tM4[1][2];

adj4[1][2] = -tM4[0][0]*tM4[1][2]*tM4[3][3] + tM4[0][0]*tM4[1][3]*tM4[3][2]
+ tM4[1][0]*tM4[0][2]*tM4[3][3] - tM4[1][0]*tM4[0][3]*tM4[3][2]
- tM4[3][0]*tM4[0][2]*tM4[1][3] + tM4[3][0]*tM4[0][3]*tM4[1][2];

adj4[2][2] = tM4[0][0]*tM4[1][1]*tM4[3][3] - tM4[0][0]*tM4[1][3]*tM4[3][1]
- tM4[1][0]*tM4[0][1]*tM4[3][3] + tM4[1][0]*tM4[0][3]*tM4[3][1]
+ tM4[3][0]*tM4[0][1]*tM4[1][3] - tM4[3][0]*tM4[0][3]*tM4[1][1];

adj4[3][2] = -tM4[0][0]*tM4[1][1]*tM4[3][2] + tM4[0][0]*tM4[1][2]*tM4[3][1]
+ tM4[1][0]*tM4[0][1]*tM4[3][2] - tM4[1][0]*tM4[0][2]*tM4[3][1]
- tM4[3][0]*tM4[0][1]*tM4[1][2] + tM4[3][0]*tM4[0][2]*tM4[1][1];

adj4[0][3] = -tM4[0][1]*tM4[1][2]*tM4[2][3] + tM4[0][1]*tM4[1][3]*tM4[2][2]
+ tM4[1][1]*tM4[0][2]*tM4[2][3] - tM4[1][1]*tM4[0][3]*tM4[2][2]
- tM4[2][1]*tM4[0][2]*tM4[1][3] + tM4[2][1]*tM4[0][3]*tM4[1][2];

adj4[1][3] = tM4[0][0]*tM4[1][2]*tM4[2][3] - tM4[0][0]*tM4[1][3]*tM4[2][2]
- tM4[1][0]*tM4[0][2]*tM4[2][3] + tM4[1][0]*tM4[0][3]*tM4[2][2]
+ tM4[2][0]*tM4[0][2]*tM4[1][3] - tM4[2][0]*tM4[0][3]*tM4[1][2];

adj4[2][3] = -tM4[0][0]*tM4[1][1]*tM4[2][3] + tM4[0][0]*tM4[1][3]*tM4[2][1]
+ tM4[1][0]*tM4[0][1]*tM4[2][3] - tM4[1][0]*tM4[0][3]*tM4[2][1]
- tM4[2][0]*tM4[0][1]*tM4[1][3] + tM4[2][0]*tM4[0][3]*tM4[1][1];

adj4[3][3] = tM4[0][0]*tM4[1][1]*tM4[2][2] - tM4[0][0]*tM4[1][2]*tM4[2][1]
- tM4[1][0]*tM4[0][1]*tM4[2][2] + tM4[1][0]*tM4[0][2]*tM4[2][1]
+ tM4[2][0]*tM4[0][1]*tM4[1][2] - tM4[2][0]*tM4[0][2]*tM4[1][1];

det4 = tM4[0][0]*tM4[1][1]*tM4[2][2]*tM4[3][3] + tM4[0][1]*tM4[1][2]*tM4[2][3]*tM4[3][0]
+ tM4[0][2]*tM4[1][3]*tM4[2][0]*tM4[3][1] + tM4[0][3]*tM4[1][0]*tM4[2][1]*tM4[3][2]
- tM4[3][0]*tM4[2][1]*tM4[1][2]*tM4[0][3] - tM4[3][1]*tM4[2][2]*tM4[1][3]*tM4[0][0]
```

```
- tM4[3][2]*tM4[2][3]*tM4[1][0]*tM4[0][1] - tM4[3][3]*tM4[2][0]*tM4[1][1]*tM4[0][2];

if (det4==0)
{
    det4=det4k;
}

for (row = 0; row < 4; ++row)
{
        for (col = 0; col < 4; ++col)
        {
                M4[row][col] = adj4[row][col] / det4;
        }
}

for (row = 0; row < 4; row++)
{
        tK4[row] = 0;
        for (int col = 0; col < 4; col++)
        {
                tK4[row] += P4[row][col]*Z4[col];
        }
}

for (i = 0; i < 4; i++)
{
        K4[i] = 0;
        for (j = 0; j < 4; j++)
        {
                K4[i] += M4[i][j]*tK4[j];
        }
}

for (i = 0; i < 4; i++)
{
        for (j = 0; j < 4; j++)
        {
                P4k[i][j] = P4[i][j] - (K4[i]*Z4[0]*P4[0][j]
                + K4[0]*Z4[1]*P4[1][j] + K4[0]*Z4[2]*P4[2][j]
                + K4[0]*Z4[3]*P4[3][j]) + Q2[i][j];
        }
}

for (i = 0; i < 4; i++)
{
        W4k[i] = eta4*K4[i]*e4 + W4[i];
}

/* ---------------------- Controlador Neuronal ---------------------- */

f11 = W1[0]*Z1[0] + W1[1]*Z1[1];
f12 = W2[0]*Z2[0] + W2[1]*Z2[1];
f21 = (W3[0]*Z3[0] + W3[1]*Z3[1] + W3[2]*Z3[2] + W3[3]*Z3[3]) + w31*q3;
f22 = (W4[0]*Z4[0] + W4[1]*Z4[1] + W4[2]*Z4[2] + W4[3]*Z4[3]) + w41*q4;

x1k = w11*q3 + f11;
```

```
x2k = w21*q4 + f12;
x3k = f21 + w35*tau1;
x4k = f22 + w46*tau2;

wq11 = tanh(q1 + q3*delay1)*W1k[0] + W1k[1];
wq21 = tanh(q2 + q4*delay1)*W2k[0] + W2k[1];

Flk[0] = f11;            Flk[1] = f12;
X1k[0] = x1k;            X1k[1] = x2k;
WQ1[0] = wq11;           WQ1[1] = wq21;
F2k[0] = f21;            F2k[1] = f22;
X1[0] = x1;              X1[1] = x2;
X2[0] = x3;              X2[1] = x4;

Z1k[0] = X1[0] - Rk[0];
Z1k[1] = X1[1] - Rk[1];

Z1k1[0] = X1k[0] - Rk1[0];
Z1k1[1] = X1k[1] - Rk1[1];

X2desk[0] = B[0][0]*(-Flk[0] + (k1[0][0]*Z1k[0]) + Rk1[0]);
X2desk[1] = B[1][1]*(-Flk[1] + (k1[1][1]*Z1k[1]) + Rk1[1]);

Z2k[0] = X2[0] - X2desk[0];
Z2k[1] = X2[1] - X2desk[1];

X2desk1[0] = B[0][0]*(-WQ1[0] + (k1[0][0]*Z1k1[0]) + Rk2[0]);
X2desk1[1] = B[1][1]*(-WQ1[1] + (k1[1][1]*Z1k1[1]) + Rk2[1]);

/* ------------------------------ Ley de control ------------------------------ */

Ueq[0] = B[0][0]*(-F2k[0] + (k2[0][0]*Z2k[0]) + X2desk1[0]);
Ueq[1] = B[1][1]*(-F2k[1] + (k2[1][1]*Z2k[1]) + X2desk1[1]);

if (abs(Ueq[0]) < u01)
{
        tau1 = Ueq[0];
}
else
        tau1 = u01*(Ueq[0] / (abs(Ueq[0])));

if (abs(Ueq[1]) < u02)
{
        tau2 = Ueq[1];
}
else
        tau2 = u02*(Ueq[1] / (abs(Ueq[1])));


/* Actualización de estados y errores */

if (t > delay1)
{
   x1 = x1k;
   x2 = x2k;
   x3 = x3k;
```

```
    x4 = x4k;

    e1 = q1 - x1;
    e2 = q2 - x2;
    e3 = q3 - x3;
    e4 = q4 - x4;

    ve1 += qt1*qt1;
    ve2 += qt2*qt2;

    se1 = (ve1*delay1)/t;
    se2 = (ve2*delay1)/t;
    mse1 = sqrt(se1);
    mse2 = sqrt(se2);

    for (i = 0; i < 2; ++i)
    {
         W1[i] = W1k[i];
         W2[i] = W2k[i];
    }

    for (i = 0; i < 4; ++i)
    {
         W3[i] = W3k[i];
         W4[i] = W4k[i];
    }

    for (i = 0; i < 2; i++)
    {
         for (j = 0; j < 2; j++)
         {
             P1[i][j] = P1k[i][j];
             P2[i][j] = P2k[i][j];
         }
    }

    for (i = 0; i < 4; i++)
    {
         for (j = 0; j < 4; j++)
         {
             P3[i][j] = P3k[i][j];
             P4[i][j] = P4k[i][j];
             tM3[i][j] = 0;
             tM4[i][j] = 0;
         }
    }
}
/* ************************** FIN DEL PROGRAMA ************************ */
```

# Apéndice D

# Modelo dinámico del robot PA10-7CE

## D.1. Modelo dinámico (método de Lagrange) generado por SYMORO+

```
%(*****************************************)%
%(** SYMORO+ : SYmbolic MOdelling of RObots **)%
%(**=====================================**)%
%(**       IRCCyN-ECN - 1, rue de la Noe     **)%
%(**       B.P.92101                         **)%
%(**       44321 Nantes cedex 3, FRANCE      **)%
%(**       www.irccyn.ec-nantes.fr           **)%
%(*****************************************)%

S1=Sin(q1);
C1=Cos(q1);
S2=Sin(q2);
C2=Cos(q2);
S3=Sin(q3);
C3=Cos(q3);
S4=Sin(q4);
C4=Cos(q4);
S5=Sin(q5);
C5=Cos(q5);
S6=Sin(q6);
C6=Cos(q6);
S7=Sin(q7);
C7=Cos(q7);
AS17=0.001*C7 - 0.001*S7;
AS37=0.001*C7 + 0.001*S7;
AJ117=0.01*C7;
AJ317=0.01*S7;
AJA117=AJ117*C7;
AJA317=AJ317*C7;
AJA337=AJ317*S7;
PAS127=-0.07*AS17;
PAS327=-0.07*AS37;
XXP6=0.1 + AJA117;
ZZP6=0.1 + AJA337;
MXP6=0.001 + AS17;
AS16=C6*MXP6 - 0.08*S6;
AS36=-0.08*C6 - MXP6*S6;
AJ116=PAS127*S6 + C6*XXP6;
```

```
AJ126=-(C6*PAS127) - 0.01*S6;
AJ136=AJA317*C6 + PAS327*S6;
AJ316=C6*PAS127 - S6*XXP6;
AJ326=-0.01*C6 + PAS127*S6;
AJ336=C6*PAS327 - AJA317*S6;
AJA116=AJ116*C6 - AJ126*S6;
AJA216=AJA317*C6 + PAS327*S6;
AJA316=AJ316*C6 - AJ326*S6;
AJA336=-(AJ326*C6) - AJ316*S6;
XXP5=0.05 + AJA116;
ZZP5=0.5 + AJA336;
MXP5=0.001 + AS16;
MYP5=0.2 + AS37;
AS15=C5*MXP5 - MYP5*S5;
AS35=C5*MYP5 + MXP5*S5;
AJ115=-(AJA216*S5) + C5*XXP5;
AJ125=AJA216*C5 - S5*ZZP6;
AJ135=AJA316*C5 - AJ336*S5;
AJ315=AJA216*C5 + S5*XXP5;
AJ325=AJA216*S5 + C5*ZZP6;
AJ335=AJ336*C5 + AJA316*S5;
AJA115=AJ115*C5 - AJ125*S5;
AJA215=-(AJA316*C5) + AJ336*S5;
AJA315=AJ315*C5 - AJ325*S5;
AJA335=AJ325*C5 + AJ315*S5;
PAS115=-0.48*AS36;
PAS125=-0.48*AS15;
PAS325=-0.48*AS35;
PAS335=-0.48*AS36;
XXP4=1 + AJA115 - 2*PAS115;
XYP4=AJA215 - PAS125;
YZP4=-AJ335 - PAS325;
ZZP4=1 + AJA335 - 2*PAS335;
MXP4=0.001 + AS15;
MYP4=0.7 - AS36;
AS14=C4*MXP4 - MYP4*S4;
AS34=-(C4*MYP4) - MXP4*S4;
AJ114=-((AJA215 - PAS125)*S4) + C4*XXP4;
AJ124=C4*XYP4 - S4*ZZP5;
AJ134=AJA315*C4 - S4*YZP4;
AJ224=-AJ335 - PAS325;
AJ314=-(C4*(AJA215 - PAS125)) - S4*XXP4;
AJ324=-(S4*XYP4) - C4*ZZP5;
AJ334=-(AJA315*S4) - C4*YZP4;
AJA114=AJ114*C4 - AJ124*S4;
AJA214=AJA315*C4 - AJ224*S4;
AJA314=AJ314*C4 - AJ324*S4;
AJA334=-(AJ324*C4) - AJ314*S4;
XXP3=1.5 + AJA114;
ZZP3=1.5 + AJA334;
MXP3=0.001 + AS14;
MYP3=1.2 + AS35;
AS13=C3*MXP3 - MYP3*S3;
AS33=C3*MYP3 + MXP3*S3;
AJ113=-(AJA214*S3) + C3*XXP3;
AJ123=AJA214*C3 - S3*ZZP4;
```

```
AJ133=AJA314*C3 - AJ334*S3;
AJ313=AJA214*C3 + S3*XXP3;
AJ323=AJA214*S3 + C3*ZZP4;
AJ333=AJ334*C3 + AJA314*S3;
AJA113=AJ113*C3 - AJ123*S3;
AJA213=-(AJA314*C3) + AJ334*S3;
AJA313=AJ313*C3 - AJ323*S3;
AJA333=AJ323*C3 + AJ313*S3;
PAS113=-0.45*AS34;
PAS123=-0.45*AS13;
PAS323=-0.45*AS33;
PAS333=-0.45*AS34;
XXP2=2 + AJA113 - 2*PAS113;
XYP2=AJA213 - PAS123;
YZP2=-AJ333 - PAS323;
ZZP2=2 + AJA333 - 2*PAS333;
MXP2=0.001 + AS13;
MYP2=1.5 - AS34;
AS12=C2*MXP2 - MYP2*S2;
AS32=-(C2*MYP2) - MXP2*S2;
AJ112=-((AJA213 - PAS123)*S2) + C2*XXP2;
AJ122=C2*XYP2 - S2*ZZP3;
AJ132=AJA313*C2 - S2*YZP2;
AJ222=-AJ333 - PAS323;
AJ312=-(C2*(AJA213 - PAS123)) - S2*XXP2;
AJ322=-(S2*XYP2) - C2*ZZP3;
AJ332=-(AJA313*S2) - C2*YZP2;
AJA112=AJ112*C2 - AJ122*S2;
AJA212=AJA313*C2 - AJ222*S2;
AJA312=AJ312*C2 - AJ322*S2;
AJA332=-(AJ322*C2) - AJ312*S2;
ZZP1=3 + AJA332;
VZ131=-(C3*S2);
VZ231=S2*S3;
PRV131=0.45*VZ231;
PRV231=-0.45*VZ131;
PRV132=0.45*C3;
PRV232=-0.45*S3;
VZ141=-(C2*S4) + C4*VZ131;
VZ241=-(C2*C4) - S4*VZ131;
JPI141=0.45*S4;
JPI241=0.45*C4;
PRV141=JPI241*VZ231;
PRV241=-(JPI141*VZ231);
PRV341=-(JPI241*VZ141) + JPI141*VZ241;
VZ142=C4*S3;
VZ242=-(S3*S4);
JPI142=0.45*S4;
JPI242=0.45*C4;
PRV142=C3*JPI242;
PRV242=-(C3*JPI142);
PRV342=-(JPI242*VZ142) + JPI142*VZ242;
VZ151=C5*VZ141 + S5*VZ231;
VZ251=-(S5*VZ141) + C5*VZ231;
JPI151=C5*JPI141;
JPI251=-(JPI141*S5);
```

```
JPI351=-0.48 - JPI241;
PRV151=-(JPI251*VZ241) - JPI351*VZ251;
PRV251=JPI351*VZ151 + JPI151*VZ241;
PRV351=-(JPI251*VZ151) + JPI151*VZ251;
VZ152=C3*S5 + C5*VZ142;
VZ252=C3*C5 - S5*VZ142;
JPI152=C5*JPI142;
JPI252=-(JPI142*S5);
JPI352=-0.48 - JPI242;
PRV152=-(JPI252*VZ242) - JPI352*VZ252;
PRV252=JPI352*VZ152 + JPI152*VZ242;
PRV352=-(JPI252*VZ152) + JPI152*VZ252;
VZ153=-(C5*S4);
VZ253=S4*S5;
PRV153=0.48*VZ253;
PRV253=-0.48*VZ153;
PRV154=0.48*C5;
PRV254=-0.48*S5;
VZ161=C6*VZ151 + S6*VZ241;
VZ261=-(S6*VZ151) + C6*VZ241;
JPI161=C6*JPI151 - JPI351*S6;
JPI261=-(C6*JPI351) - JPI151*S6;
PRV161=JPI261*VZ251 - JPI251*VZ261;
PRV261=JPI251*VZ161 - JPI161*VZ251;
PRV361=-(JPI261*VZ161) + JPI161*VZ261;
VZ162=C6*VZ152 + S6*VZ242;
VZ262=-(S6*VZ152) + C6*VZ242;
JPI162=C6*JPI152 - JPI352*S6;
JPI262=-(C6*JPI352) - JPI152*S6;
PRV162=JPI262*VZ252 - JPI252*VZ262;
PRV262=JPI252*VZ162 - JPI162*VZ252;
PRV362=-(JPI262*VZ162) + JPI162*VZ262;
VZ163=-(C4*S6) + C6*VZ153;
VZ263=-(C4*C6) - S6*VZ153;
JPI163=0.48*S6;
JPI263=0.48*C6;
PRV163=JPI263*VZ253;
PRV263=-(JPI163*VZ253);
PRV363=-(JPI263*VZ163) + JPI163*VZ263;
VZ164=C6*S5;
VZ264=-(S5*S6);
JPI164=0.48*S6;
JPI264=0.48*C6;
PRV164=C5*JPI264;
PRV264=-(C5*JPI164);
PRV364=-(JPI264*VZ164) + JPI164*VZ264;
VZ171=C7*VZ161 + S7*VZ251;
VZ271=-(S7*VZ161) + C7*VZ251;
JPI171=C7*JPI161 + JPI251*S7;
JPI271=C7*JPI251 - JPI161*S7;
JPI371=-0.07 - JPI261;
PRV171=-(JPI271*VZ261) - JPI371*VZ271;
PRV271=JPI371*VZ171 + JPI171*VZ261;
PRV371=-(JPI271*VZ171) + JPI171*VZ271;
VZ172=C7*VZ162 + S7*VZ252;
VZ272=-(S7*VZ162) + C7*VZ252;
```

```
JPI172=C7*JPI162 + JPI252*S7;
JPI272=C7*JPI252 - JPI162*S7;
JPI372=-0.07 - JPI262;
PRV172=-(JPI272*VZ262) - JPI372*VZ272;
PRV272=JPI372*VZ172 + JPI172*VZ262;
PRV372=-(JPI272*VZ172) + JPI172*VZ272;
VZ173=C7*VZ163 + S7*VZ253;
VZ273=-(S7*VZ163) + C7*VZ253;
JPI173=C7*JPI163;
JPI273=-(JPI163*S7);
JPI373=-0.07 - JPI263;
PRV173=-(JPI273*VZ263) - JPI373*VZ273;
PRV273=JPI373*VZ173 + JPI173*VZ263;
PRV373=-(JPI273*VZ173) + JPI173*VZ273;
VZ174=C5*S7 + C7*VZ164;
VZ274=C5*C7 - S7*VZ164;
JPI174=C7*JPI164;
JPI274=-(JPI164*S7);
JPI374=-0.07 - JPI264;
PRV174=-(JPI274*VZ264) - JPI374*VZ274;
PRV274=JPI374*VZ174 + JPI174*VZ264;
PRV374=-(JPI274*VZ174) + JPI174*VZ274;
VZ175=-(C7*S6);
VZ275=S6*S7;
PRV175=0.07*VZ275;
PRV275=-0.07*VZ175;
PRV176=0.07*C7;
PRV276=-0.07*S7;
EA21=-(AJA313*S2) - C2*YZP2;
EA32=AJ334*C3 + AJA314*S3;
FA32=-(MYP3*PRV132) + MXP3*PRV232;
GA32=EA32 + FA32;
EA31=AJA314*VZ131 + AJ334*VZ231;
FA31=-(MYP3*PRV131) + MXP3*PRV231 + C2*ZZP3;
GA31=EA31 + FA31;
EA43=-(AJA315*S4) - C4*YZP4;
EA42=AJA315*VZ142 + VZ242*YZP4;
FA42=-(MYP4*PRV142) + MXP4*PRV242 + C3*ZZP4;
GA42=EA42 + FA42;
EA41=AJA315*VZ141 + VZ241*YZP4;
FA41=-(MYP4*PRV141) + MXP4*PRV241 + VZ231*ZZP4;
GA41=EA41 + FA41;
EA54=AJ336*C5 + AJA316*S5;
FA54=-(MYP5*PRV154) + MXP5*PRV254;
GA54=EA54 + FA54;
EA53=AJA316*VZ153 + AJ336*VZ253;
FA53=-(MYP5*PRV153) + MXP5*PRV253 + C4*ZZP5;
GA53=EA53 + FA53;
EA52=AJA316*VZ152 + AJ336*VZ252;
FA52=-(MYP5*PRV152) + MXP5*PRV252 - VZ242*ZZP5;
GA52=EA52 + FA52;
EA51=AJA316*VZ151 + AJ336*VZ251;
FA51=-(MYP5*PRV151) + MXP5*PRV251 - VZ241*ZZP5;
GA51=EA51 + FA51;
EA65=C6*PAS327 - AJA317*S6;
EA64=AJA317*VZ164 - PAS327*VZ264;
```

```
FA64=-0.08*PRV164 + MXP6*PRV264 + C5*ZZP6;
GA64=EA64 + FA64;
EA63=AJA317*VZ163 - PAS327*VZ263;
FA63=-0.08*PRV163 + MXP6*PRV263 + VZ253*ZZP6;
GA63=EA63 + FA63;
EA62=AJA317*VZ162 - PAS327*VZ262;
FA62=-0.08*PRV162 + MXP6*PRV262 + VZ252*ZZP6;
GA62=EA62 + FA62;
EA61=AJA317*VZ161 - PAS327*VZ261;
FA61=-0.08*PRV161 + MXP6*PRV261 + VZ251*ZZP6;
GA61=EA61 + FA61;
FA76=-0.001*PRV176 + 0.001*PRV276;
FA75=0.01*C6 - 0.001*PRV175 + 0.001*PRV275;
FA74=-0.001*PRV174 + 0.001*PRV274 - 0.01*VZ264;
FA73=-0.001*PRV173 + 0.001*PRV273 - 0.01*VZ263;
FA72=-0.001*PRV172 + 0.001*PRV272 - 0.01*VZ262;
FA71=-0.001*PRV171 + 0.001*PRV271 - 0.01*VZ261;
A(1,1)=ZZP1;
A(2,1)=EA21;
A(3,1)=GA31;
A(4,1)=GA41;
A(5,1)=GA51;
A(6,1)=GA61;
A(7,1)=FA71;
A(2,2)=0.04 + ZZP2;
A(3,2)=GA32;
A(4,2)=GA42;
A(5,2)=GA52;
A(6,2)=GA62;
A(7,2)=FA72;
A(3,3)=0.04 + ZZP3;
A(4,3)=EA43;
A(5,3)=GA53;
A(6,3)=GA63;
A(7,3)=FA73;
A(4,4)=0.02 + ZZP4;
A(5,4)=GA54;
A(6,4)=GA64;
A(7,4)=FA74;
A(5,5)=0.02 + ZZP5;
A(6,5)=EA65;
A(7,5)=FA75;
A(6,6)=0.02 + ZZP6;
A(7,6)=FA76;
A(7,7)=0.02;
PMA0112=C1*C2;
PMA0122=-(C1*S2);
PMA0212=C2*S1;
PMA0222=-(S1*S2);
PMA0113=C3*PMA0112 - S1*S3;
PMA0123=-(C3*S1) - PMA0112*S3;
PMA0213=C3*PMA0212 + C1*S3;
PMA0223=C1*C3 - PMA0212*S3;
PMA0313=-(C3*S2);
PMA0323=S2*S3;
PMA0114=C4*PMA0113 + PMA0122*S4;
```

```
PMA0124=C4*PMA0122 - PMA0113*S4;
PMA0214=C4*PMA0213 + PMA0222*S4;
PMA0224=C4*PMA0222 - PMA0213*S4;
PMA0314=C4*PMA0313 - C2*S4;
PMA0324=-(C2*C4) - PMA0313*S4;
PMA0115=C5*PMA0114 + PMA0123*S5;
PMA0125=C5*PMA0123 - PMA0114*S5;
PMA0215=C5*PMA0214 + PMA0223*S5;
PMA0225=C5*PMA0223 - PMA0214*S5;
PMA0315=C5*PMA0314 + PMA0323*S5;
PMA0325=C5*PMA0323 - PMA0314*S5;
PMA0116=C6*PMA0115 + PMA0124*S6;
PMA0126=C6*PMA0124 - PMA0115*S6;
PMA0216=C6*PMA0215 + PMA0224*S6;
PMA0226=C6*PMA0224 - PMA0215*S6;
PMA0316=C6*PMA0315 + PMA0324*S6;
PMA0326=C6*PMA0324 - PMA0315*S6;
PMA0117=C7*PMA0116 + PMA0125*S7;
PMA0127=C7*PMA0125 - PMA0116*S7;
PMA0217=C7*PMA0216 + PMA0225*S7;
PMA0227=C7*PMA0225 - PMA0216*S7;
PMA0317=C7*PMA0316 + PMA0325*S7;
PMA0327=C7*PMA0325 - PMA0316*S7;
VQ11=-(AS33*C1) - AS12*S1;
VQ21=AS12*C1 - AS33*S1;
VQ12=-(MYP2*PMA0112) + MXP2*PMA0122;
VQ22=-(MYP2*PMA0212) + MXP2*PMA0222;
VQ32=-(C2*MXP2) + MYP2*S2;
VQ13=-(MYP3*PMA0113) + MXP3*PMA0123;
VQ23=-(MYP3*PMA0213) + MXP3*PMA0223;
VQ33=-(MYP3*PMA0313) + MXP3*PMA0323;
VQ14=-(MYP4*PMA0114) + MXP4*PMA0124;
VQ24=-(MYP4*PMA0214) + MXP4*PMA0224;
VQ34=-(MYP4*PMA0314) + MXP4*PMA0324;
VQ15=-(MYP5*PMA0115) + MXP5*PMA0125;
VQ25=-(MYP5*PMA0215) + MXP5*PMA0225;
VQ35=-(MYP5*PMA0315) + MXP5*PMA0325;
VQ16=-0.08*PMA0116 + MXP6*PMA0126;
VQ26=-0.08*PMA0216 + MXP6*PMA0226;
VQ36=-0.08*PMA0316 + MXP6*PMA0326;
VQ17=-0.001*PMA0117 + 0.001*PMA0127;
VQ27=-0.001*PMA0217 + 0.001*PMA0227;
VQ37=-0.001*PMA0317 + 0.001*PMA0327;
Q(1)=0;
Q(2)=-9.81*VQ32;
Q(3)=-9.81*VQ33;
Q(4)=-9.81*VQ34;
Q(5)=-9.81*VQ35;
Q(6)=-9.81*VQ36;
Q(7)=-9.81*VQ37;
RP2=(-XXP2 + ZZP2 + ZZP3)/2.;
NP2=(XXP2 + ZZP2 - ZZP3)/2.;
PH21=-(C2*NP2) + S2*XYP2;
PS21=RP2*S2 + C2*XYP2;
TE21=-(C2*MYP2) - MXP2*S2;
RP3=(-XXP3 + ZZP3 + ZZP4)/2.;
```

```
NP3=(XXP3 + ZZP3 - ZZP4)/2.;
PH32=C3*NP3 - AJA214*S3;
PS32=-(AJA214*C3) - RP3*S3;
TE32=C3*MYP3 + MXP3*S3;
LAI31=-(C2*MXP3);
MUI31=-(C2*MYP3);
PH31=-(AJ334*C2) - AJA214*VZ131 + NP3*VZ231;
PS31=AJA314*C2 - RP3*VZ131 - AJA214*VZ231;
TE31=MXP3*VZ131 + MYP3*VZ231;
RP4=(-XXP4 + ZZP4 + ZZP5)/2.;
NP4=(XXP4 + ZZP4 - ZZP5)/2.;
PH43=-(C4*NP4) + S4*XYP4;
PS43=RP4*S4 + C4*XYP4;
TE43=-(C4*MYP4) - MXP4*S4;
LAI42=-(C3*MXP4);
MUI42=-(C3*MYP4);
PH42=NP4*VZ242 - VZ142*XYP4 - C3*YZP4;
PS42=AJA315*C3 - RP4*VZ142 - VZ242*XYP4;
TE42=MXP4*VZ142 + MYP4*VZ242;
LAI41=-(MXP4*VZ231);
MUI41=-(MYP4*VZ231);
PH41=NP4*VZ241 - VZ141*XYP4 - VZ231*YZP4;
PS41=-(RP4*VZ141) + AJA315*VZ231 - VZ241*XYP4;
TE41=MXP4*VZ141 + MYP4*VZ241;
RP5=(-XXP5 + ZZP5 + ZZP6)/2.;
NP5=(XXP5 + ZZP5 - ZZP6)/2.;
PH54=C5*NP5 - AJA216*S5;
PS54=-(AJA216*C5) - RP5*S5;
TE54=C5*MYP5 + MXP5*S5;
LAI53=-(C4*MXP5);
MUI53=-(C4*MYP5);
PH53=-(AJ336*C4) - AJA216*VZ153 + NP5*VZ253;
PS53=AJA316*C4 - RP5*VZ153 - AJA216*VZ253;
TE53=MXP5*VZ153 + MYP5*VZ253;
LAI52=MXP5*VZ242;
MUI52=MYP5*VZ242;
PH52=-(AJA216*VZ152) + AJ336*VZ242 + NP5*VZ252;
PS52=-(RP5*VZ152) - AJA316*VZ242 - AJA216*VZ252;
TE52=MXP5*VZ152 + MYP5*VZ252;
LAI51=MXP5*VZ241;
MUI51=MYP5*VZ241;
PH51=-(AJA216*VZ151) + AJ336*VZ241 + NP5*VZ251;
PS51=-(RP5*VZ151) - AJA316*VZ241 - AJA216*VZ251;
TE51=MXP5*VZ151 + MYP5*VZ251;
RP6=(0.01 - XXP6 + ZZP6)/2.;
NP6=(-0.01 + XXP6 + ZZP6)/2.;
PH65=-(C6*NP6) - PAS127*S6;
PS65=-(C6*PAS127) + RP6*S6;
TE65=-0.08*C6 - MXP6*S6;
LAI64=-(C5*MXP6);
MUI64=-0.08*C5;
PH64=C5*PAS327 + PAS127*VZ164 + NP6*VZ264;
PS64=AJA317*C5 - RP6*VZ164 + PAS127*VZ264;
TE64=MXP6*VZ164 + 0.08*VZ264;
LAI63=-(MXP6*VZ253);
MUI63=-0.08*VZ253;
```

```
PH63=PAS127*VZ163 + PAS327*VZ253 + NP6*VZ263;
PS63=-(RP6*VZ163) + AJA317*VZ253 + PAS127*VZ263;
TE63=MXP6*VZ163 + 0.08*VZ263;
LAI62=-(MXP6*VZ252);
MUI62=-0.08*VZ252;
PH62=PAS127*VZ162 + PAS327*VZ252 + NP6*VZ262;
PS62=-(RP6*VZ162) + AJA317*VZ252 + PAS127*VZ262;
TE62=MXP6*VZ162 + 0.08*VZ262;
LAI61=-(MXP6*VZ251);
MUI61=-0.08*VZ251;
PH61=PAS127*VZ161 + PAS327*VZ251 + NP6*VZ261;
PS61=-(RP6*VZ161) + AJA317*VZ251 + PAS127*VZ261;
TE61=MXP6*VZ161 + 0.08*VZ261;
PH76=0.01*C7;
TE76=0.001*C7 + 0.001*S7;
LAI75=-0.001*C6;
MUI75=-0.001*C6;
PH75=0.01*VZ275;
TE75=0.001*VZ175 + 0.001*VZ275;
LAI74=0.001*VZ264;
MUI74=0.001*VZ264;
PH74=0.01*VZ274;
TE74=0.001*VZ174 + 0.001*VZ274;
LAI73=0.001*VZ263;
MUI73=0.001*VZ263;
PH73=0.01*VZ273;
TE73=0.001*VZ173 + 0.001*VZ273;
LAI72=0.001*VZ262;
MUI72=0.001*VZ262;
PH72=0.01*VZ272;
TE72=0.001*VZ172 + 0.001*VZ272;
LAI71=0.001*VZ261;
MUI71=0.001*VZ261;
PH71=0.01*VZ271;
TE71=0.001*VZ171 + 0.001*VZ271;
CI212=-(AJA313*C2) + S2*YZP2;
CI211=-(C2*PS21) - PH21*S2;
CI313=-(AJ334*VZ131) + AJA314*VZ231;
CJ313=-(MXP3*PRV131) - MYP3*PRV231;
CL313=CI313 + CJ313;
CI312=PH32*VZ131 + PS32*VZ231;
CI311=PH31*VZ131 + PS31*VZ231;
CJ311=LAI31*PRV131 + MUI31*PRV231;
CL311=CI311 + CJ311;
CI323=AJA314*C3 - AJ334*S3;
CJ323=-(MXP3*PRV132) - MYP3*PRV232;
CL323=CI323 + CJ323;
CI322=C3*PS32 + PH32*S3;
CI321=C3*PS31 + PH31*S3;
CJ321=LAI31*PRV132 + MUI31*PRV232;
CL321=CI321 + CJ321;
CI414=AJA315*VZ241 - VZ141*YZP4;
CJ414=-(MXP4*PRV141) - MYP4*PRV241;
CL414=CI414 + CJ414;
CI413=PH43*VZ141 + PS43*VZ241;
CK413=PRV341*TE43;
```

```
CL413=CI413 + CK413;
CI412=PH42*VZ141 + PS42*VZ241;
CJ412=LAI42*PRV141 + MUI42*PRV241;
CK412=PRV341*TE42;
CL412=CI412 + CJ412 + CK412;
CI411=PH41*VZ141 + PS41*VZ241;
CJ411=LAI41*PRV141 + MUI41*PRV241;
CK411=PRV341*TE41;
CL411=CI411 + CJ411 + CK411;
CI424=AJA315*VZ242 - VZ142*YZP4;
CJ424=-(MXP4*PRV142) - MYP4*PRV242;
CL424=CI424 + CJ424;
CI423=PH43*VZ142 + PS43*VZ242;
CK423=PRV342*TE43;
CL423=CI423 + CK423;
CI422=PH42*VZ142 + PS42*VZ242;
CJ422=LAI42*PRV142 + MUI42*PRV242;
CK422=PRV342*TE42;
CL422=CI422 + CJ422 + CK422;
CI421=PH41*VZ142 + PS41*VZ242;
CJ421=LAI41*PRV142 + MUI41*PRV242;
CK421=PRV342*TE41;
CL421=CI421 + CJ421 + CK421;
CI434=-(AJA315*C4) + S4*YZP4;
CI433=-(C4*PS43) - PH43*S4;
CI432=-(C4*PS42) - PH42*S4;
CI431=-(C4*PS41) - PH41*S4;
CI515=-(AJ336*VZ151) + AJA316*VZ251;
CJ515=-(MXP5*PRV151) - MYP5*PRV251;
CL515=CI515 + CJ515;
CI514=PH54*VZ151 + PS54*VZ251;
CK514=PRV351*TE54;
CL514=CI514 + CK514;
CI513=PH53*VZ151 + PS53*VZ251;
CJ513=LAI53*PRV151 + MUI53*PRV251;
CK513=PRV351*TE53;
CL513=CI513 + CJ513 + CK513;
CI512=PH52*VZ151 + PS52*VZ251;
CJ512=LAI52*PRV151 + MUI52*PRV251;
CK512=PRV351*TE52;
CL512=CI512 + CJ512 + CK512;
CI511=PH51*VZ151 + PS51*VZ251;
CJ511=LAI51*PRV151 + MUI51*PRV251;
CK511=PRV351*TE51;
CL511=CI511 + CJ511 + CK511;
CI525=-(AJ336*VZ152) + AJA316*VZ252;
CJ525=-(MXP5*PRV152) - MYP5*PRV252;
CL525=CI525 + CJ525;
CI524=PH54*VZ152 + PS54*VZ252;
CK524=PRV352*TE54;
CL524=CI524 + CK524;
CI523=PH53*VZ152 + PS53*VZ252;
CJ523=LAI53*PRV152 + MUI53*PRV252;
CK523=PRV352*TE53;
CL523=CI523 + CJ523 + CK523;
CI522=PH52*VZ152 + PS52*VZ252;
```

```
CJ522=LAI52*PRV152 + MUI52*PRV252;
CK522=PRV352*TE52;
CL522=CI522 + CJ522 + CK522;
CI521=PH51*VZ152 + PS51*VZ252;
CJ521=LAI51*PRV152 + MUI51*PRV252;
CK521=PRV352*TE51;
CL521=CI521 + CJ521 + CK521;
CI535=-(AJ336*VZ153) + AJA316*VZ253;
CJ535=-(MXP5*PRV153) - MYP5*PRV253;
CL535=CI535 + CJ535;
CI534=PH54*VZ153 + PS54*VZ253;
CI533=PH53*VZ153 + PS53*VZ253;
CJ533=LAI53*PRV153 + MUI53*PRV253;
CL533=CI533 + CJ533;
CI532=PH52*VZ153 + PS52*VZ253;
CJ532=LAI52*PRV153 + MUI52*PRV253;
CL532=CI532 + CJ532;
CI531=PH51*VZ153 + PS51*VZ253;
CJ531=LAI51*PRV153 + MUI51*PRV253;
CL531=CI531 + CJ531;
CI545=AJA316*C5 - AJ336*S5;
CJ545=-(MXP5*PRV154) - MYP5*PRV254;
CL545=CI545 + CJ545;
CI544=C5*PS54 + PH54*S5;
CI543=C5*PS53 + PH53*S5;
CJ543=LAI53*PRV154 + MUI53*PRV254;
CL543=CI543 + CJ543;
CI542=C5*PS52 + PH52*S5;
CJ542=LAI52*PRV154 + MUI52*PRV254;
CL542=CI542 + CJ542;
CI541=C5*PS51 + PH51*S5;
CJ541=LAI51*PRV154 + MUI51*PRV254;
CL541=CI541 + CJ541;
CI616=PAS327*VZ161 + AJA317*VZ261;
CJ616=-(MXP6*PRV161) - 0.08*PRV261;
CL616=CI616 + CJ616;
CI615=PH65*VZ161 + PS65*VZ261;
CK615=PRV361*TE65;
CL615=CI615 + CK615;
CI614=PH64*VZ161 + PS64*VZ261;
CJ614=LAI64*PRV161 + MUI64*PRV261;
CK614=PRV361*TE64;
CL614=CI614 + CJ614 + CK614;
CI613=PH63*VZ161 + PS63*VZ261;
CJ613=LAI63*PRV161 + MUI63*PRV261;
CK613=PRV361*TE63;
CL613=CI613 + CJ613 + CK613;
CI612=PH62*VZ161 + PS62*VZ261;
CJ612=LAI62*PRV161 + MUI62*PRV261;
CK612=PRV361*TE62;
CL612=CI612 + CJ612 + CK612;
CI611=PH61*VZ161 + PS61*VZ261;
CJ611=LAI61*PRV161 + MUI61*PRV261;
CK611=PRV361*TE61;
CL611=CI611 + CJ611 + CK611;
CI626=PAS327*VZ162 + AJA317*VZ262;
```

```
CJ626=-(MXP6*PRV162) - 0.08*PRV262;
CL626=CI626 + CJ626;
CI625=PH65*VZ162 + PS65*VZ262;
CK625=PRV362*TE65;
CL625=CI625 + CK625;
CI624=PH64*VZ162 + PS64*VZ262;
CJ624=LAI64*PRV162 + MUI64*PRV262;
CK624=PRV362*TE64;
CL624=CI624 + CJ624 + CK624;
CI623=PH63*VZ162 + PS63*VZ262;
CJ623=LAI63*PRV162 + MUI63*PRV262;
CK623=PRV362*TE63;
CL623=CI623 + CJ623 + CK623;
CI622=PH62*VZ162 + PS62*VZ262;
CJ622=LAI62*PRV162 + MUI62*PRV262;
CK622=PRV362*TE62;
CL622=CI622 + CJ622 + CK622;
CI621=PH61*VZ162 + PS61*VZ262;
CJ621=LAI61*PRV162 + MUI61*PRV262;
CK621=PRV362*TE61;
CL621=CI621 + CJ621 + CK621;
CI636=PAS327*VZ163 + AJA317*VZ263;
CJ636=-(MXP6*PRV163) - 0.08*PRV263;
CL636=CI636 + CJ636;
CI635=PH65*VZ163 + PS65*VZ263;
CK635=PRV363*TE65;
CL635=CI635 + CK635;
CI634=PH64*VZ163 + PS64*VZ263;
CJ634=LAI64*PRV163 + MUI64*PRV263;
CK634=PRV363*TE64;
CL634=CI634 + CJ634 + CK634;
CI633=PH63*VZ163 + PS63*VZ263;
CJ633=LAI63*PRV163 + MUI63*PRV263;
CK633=PRV363*TE63;
CL633=CI633 + CJ633 + CK633;
CI632=PH62*VZ163 + PS62*VZ263;
CJ632=LAI62*PRV163 + MUI62*PRV263;
CK632=PRV363*TE62;
CL632=CI632 + CJ632 + CK632;
CI631=PH61*VZ163 + PS61*VZ263;
CJ631=LAI61*PRV163 + MUI61*PRV263;
CK631=PRV363*TE61;
CL631=CI631 + CJ631 + CK631;
CI646=PAS327*VZ164 + AJA317*VZ264;
CJ646=-(MXP6*PRV164) - 0.08*PRV264;
CL646=CI646 + CJ646;
CI645=PH65*VZ164 + PS65*VZ264;
CK645=PRV364*TE65;
CL645=CI645 + CK645;
CI644=PH64*VZ164 + PS64*VZ264;
CJ644=LAI64*PRV164 + MUI64*PRV264;
CK644=PRV364*TE64;
CL644=CI644 + CJ644 + CK644;
CI643=PH63*VZ164 + PS63*VZ264;
CJ643=LAI63*PRV164 + MUI63*PRV264;
CK643=PRV364*TE63;
```

```
CL643=CI643 + CJ643 + CK643;
CI642=PH62*VZ164 + PS62*VZ264;
CJ642=LAI62*PRV164 + MUI62*PRV264;
CK642=PRV364*TE62;
CL642=CI642 + CJ642 + CK642;
CI641=PH61*VZ164 + PS61*VZ264;
CJ641=LAI61*PRV164 + MUI61*PRV264;
CK641=PRV364*TE61;
CL641=CI641 + CJ641 + CK641;
CI656=-(AJA317*C6) - PAS327*S6;
CI655=-(C6*PS65) - PH65*S6;
CI654=-(C6*PS64) - PH64*S6;
CI653=-(C6*PS63) - PH63*S6;
CI652=-(C6*PS62) - PH62*S6;
CI651=-(C6*PS61) - PH61*S6;
CJ717=-0.001*PRV171 - 0.001*PRV271;
CI716=PH76*VZ171;
CK716=PRV371*TE76;
CL716=CI716 + CK716;
CI715=PH75*VZ171;
CJ715=LAI75*PRV171 + MUI75*PRV271;
CK715=PRV371*TE75;
CL715=CI715 + CJ715 + CK715;
CI714=PH74*VZ171;
CJ714=LAI74*PRV171 + MUI74*PRV271;
CK714=PRV371*TE74;
CL714=CI714 + CJ714 + CK714;
CI713=PH73*VZ171;
CJ713=LAI73*PRV171 + MUI73*PRV271;
CK713=PRV371*TE73;
CL713=CI713 + CJ713 + CK713;
CI712=PH72*VZ171;
CJ712=LAI72*PRV171 + MUI72*PRV271;
CK712=PRV371*TE72;
CL712=CI712 + CJ712 + CK712;
CI711=PH71*VZ171;
CJ711=LAI71*PRV171 + MUI71*PRV271;
CK711=PRV371*TE71;
CL711=CI711 + CJ711 + CK711;
CJ727=-0.001*PRV172 - 0.001*PRV272;
CI726=PH76*VZ172;
CK726=PRV372*TE76;
CL726=CI726 + CK726;
CI725=PH75*VZ172;
CJ725=LAI75*PRV172 + MUI75*PRV272;
CK725=PRV372*TE75;
CL725=CI725 + CJ725 + CK725;
CI724=PH74*VZ172;
CJ724=LAI74*PRV172 + MUI74*PRV272;
CK724=PRV372*TE74;
CL724=CI724 + CJ724 + CK724;
CI723=PH73*VZ172;
CJ723=LAI73*PRV172 + MUI73*PRV272;
CK723=PRV372*TE73;
CL723=CI723 + CJ723 + CK723;
CI722=PH72*VZ172;
```

```
CJ722=LAI72*PRV172 + MUI72*PRV272;
CK722=PRV372*TE72;
CL722=CI722 + CJ722 + CK722;
CI721=PH71*VZ172;
CJ721=LAI71*PRV172 + MUI71*PRV272;
CK721=PRV372*TE71;
CL721=CI721 + CJ721 + CK721;
CJ737=-0.001*PRV173 - 0.001*PRV273;
CI736=PH76*VZ173;
CK736=PRV373*TE76;
CL736=CI736 + CK736;
CI735=PH75*VZ173;
CJ735=LAI75*PRV173 + MUI75*PRV273;
CK735=PRV373*TE75;
CL735=CI735 + CJ735 + CK735;
CI734=PH74*VZ173;
CJ734=LAI74*PRV173 + MUI74*PRV273;
CK734=PRV373*TE74;
CL734=CI734 + CJ734 + CK734;
CI733=PH73*VZ173;
CJ733=LAI73*PRV173 + MUI73*PRV273;
CK733=PRV373*TE73;
CL733=CI733 + CJ733 + CK733;
CI732=PH72*VZ173;
CJ732=LAI72*PRV173 + MUI72*PRV273;
CK732=PRV373*TE72;
CL732=CI732 + CJ732 + CK732;
CI731=PH71*VZ173;
CJ731=LAI71*PRV173 + MUI71*PRV273;
CK731=PRV373*TE71;
CL731=CI731 + CJ731 + CK731;
CJ747=-0.001*PRV174 - 0.001*PRV274;
CI746=PH76*VZ174;
CK746=PRV374*TE76;
CL746=CI746 + CK746;
CI745=PH75*VZ174;
CJ745=LAI75*PRV174 + MUI75*PRV274;
CK745=PRV374*TE75;
CL745=CI745 + CJ745 + CK745;
CI744=PH74*VZ174;
CJ744=LAI74*PRV174 + MUI74*PRV274;
CK744=PRV374*TE74;
CL744=CI744 + CJ744 + CK744;
CI743=PH73*VZ174;
CJ743=LAI73*PRV174 + MUI73*PRV274;
CK743=PRV374*TE73;
CL743=CI743 + CJ743 + CK743;
CI742=PH72*VZ174;
CJ742=LAI72*PRV174 + MUI72*PRV274;
CK742=PRV374*TE72;
CL742=CI742 + CJ742 + CK742;
CI741=PH71*VZ174;
CJ741=LAI71*PRV174 + MUI71*PRV274;
CK741=PRV374*TE71;
CL741=CI741 + CJ741 + CK741;
CJ757=-0.001*PRV175 - 0.001*PRV275;
```

```
CI756=PH76*VZ175;
CI755=PH75*VZ175;
CJ755=LAI75*PRV175 + MUI75*PRV275;
CL755=CI755 + CJ755;
CI754=PH74*VZ175;
CJ754=LAI74*PRV175 + MUI74*PRV275;
CL754=CI754 + CJ754;
CI753=PH73*VZ175;
CJ753=LAI73*PRV175 + MUI73*PRV275;
CL753=CI753 + CJ753;
CI752=PH72*VZ175;
CJ752=LAI72*PRV175 + MUI72*PRV275;
CL752=CI752 + CJ752;
CI751=PH71*VZ175;
CJ751=LAI71*PRV175 + MUI71*PRV275;
CL751=CI751 + CJ751;
CJ767=-0.001*PRV176 - 0.001*PRV276;
CI766=PH76*S7;
CI765=PH75*S7;
CJ765=LAI75*PRV176 + MUI75*PRV276;
CL765=CI765 + CJ765;
CI764=PH74*S7;
CJ764=LAI74*PRV176 + MUI74*PRV276;
CL764=CI764 + CJ764;
CI763=PH73*S7;
CJ763=LAI73*PRV176 + MUI73*PRV276;
CL763=CI763 + CJ763;
CI762=PH72*S7;
CJ762=LAI72*PRV176 + MUI72*PRV276;
CL762=CI762 + CJ762;
CI761=PH71*S7;
CJ761=LAI71*PRV176 + MUI71*PRV276;
CL761=CI761 + CJ761;
B(1,1,2)=2*CI211;
B(1,1,3)=2*CL311;
B(1,1,4)=2*CL411;
B(1,1,5)=2*CL511;
B(1,1,6)=2*CL611;
B(1,1,7)=2*CL711;
B(1,2,3)=2*CI312;
B(1,2,4)=2*CL412;
B(1,2,5)=2*CL512;
B(1,2,6)=2*CL612;
B(1,2,7)=2*CL712;
B(1,3,4)=2*CL413;
B(1,3,5)=2*CL513;
B(1,3,6)=2*CL613;
B(1,3,7)=2*CL713;
B(1,4,5)=2*CL514;
B(1,4,6)=2*CL614;
B(1,4,7)=2*CL714;
B(1,5,6)=2*CL615;
B(1,5,7)=2*CL715;
B(1,6,7)=2*CL716;
B(2,1,3)=2*CL321;
B(2,1,4)=2*CL421;
```

```
B(2,1,5)=2*CL521;
B(2,1,6)=2*CL621;
B(2,1,7)=2*CL721;
B(2,2,3)=2*CI322;
B(2,2,4)=2*CL422;
B(2,2,5)=2*CL522;
B(2,2,6)=2*CL622;
B(2,2,7)=2*CL722;
B(2,3,4)=2*CL423;
B(2,3,5)=2*CL523;
B(2,3,6)=2*CL623;
B(2,3,7)=2*CL723;
B(2,4,5)=2*CL524;
B(2,4,6)=2*CL624;
B(2,4,7)=2*CL724;
B(2,5,6)=2*CL625;
B(2,5,7)=2*CL725;
B(2,6,7)=2*CL726;
B(3,1,2)=-2*CL321;
B(3,1,4)=2*CI431;
B(3,1,5)=2*CL531;
B(3,1,6)=2*CL631;
B(3,1,7)=2*CL731;
B(3,2,4)=2*CI432;
B(3,2,5)=2*CL532;
B(3,2,6)=2*CL632;
B(3,2,7)=2*CL732;
B(3,3,4)=2*CI433;
B(3,3,5)=2*CL533;
B(3,3,6)=2*CL633;
B(3,3,7)=2*CL733;
B(3,4,5)=2*CI534;
B(3,4,6)=2*CL634;
B(3,4,7)=2*CL734;
B(3,5,6)=2*CL635;
B(3,5,7)=2*CL735;
B(3,6,7)=2*CL736;
B(4,1,2)=-2*CL421;
B(4,1,3)=-2*CI431;
B(4,1,5)=2*CL541;
B(4,1,6)=2*CL641;
B(4,1,7)=2*CL741;
B(4,2,3)=-2*CI432;
B(4,2,5)=2*CL542;
B(4,2,6)=2*CL642;
B(4,2,7)=2*CL742;
B(4,3,5)=2*CL543;
B(4,3,6)=2*CL643;
B(4,3,7)=2*CL743;
B(4,4,5)=2*CI544;
B(4,4,6)=2*CL644;
B(4,4,7)=2*CL744;
B(4,5,6)=2*CL645;
B(4,5,7)=2*CL745;
B(4,6,7)=2*CL746;
B(5,1,2)=-2*CL521;
```

```
B(5,1,3)=-2*CL531;
B(5,1,4)=-2*CL541;
B(5,1,6)=2*CI651;
B(5,1,7)=2*CL751;
B(5,2,3)=-2*CL532;
B(5,2,4)=-2*CL542;
B(5,2,6)=2*CI652;
B(5,2,7)=2*CL752;
B(5,3,4)=-2*CL543;
B(5,3,6)=2*CI653;
B(5,3,7)=2*CL753;
B(5,4,6)=2*CI654;
B(5,4,7)=2*CL754;
B(5,5,6)=2*CI655;
B(5,5,7)=2*CL755;
B(5,6,7)=2*CI756;
B(6,1,2)=-2*CL621;
B(6,1,3)=-2*CL631;
B(6,1,4)=-2*CL641;
B(6,1,5)=-2*CI651;
B(6,1,7)=2*CL761;
B(6,2,3)=-2*CL632;
B(6,2,4)=-2*CL642;
B(6,2,5)=-2*CI652;
B(6,2,7)=2*CL762;
B(6,3,4)=-2*CL643;
B(6,3,5)=-2*CI653;
B(6,3,7)=2*CL763;
B(6,4,5)=-2*CI654;
B(6,4,7)=2*CL764;
B(6,5,7)=2*CL765;
B(6,6,7)=2*CI766;
B(7,1,2)=-2*CL721;
B(7,1,3)=-2*CL731;
B(7,1,4)=-2*CL741;
B(7,1,5)=-2*CL751;
B(7,1,6)=-2*CL761;
B(7,2,3)=-2*CL732;
B(7,2,4)=-2*CL742;
B(7,2,5)=-2*CL752;
B(7,2,6)=-2*CL762;
B(7,3,4)=-2*CL743;
B(7,3,5)=-2*CL753;
B(7,3,6)=-2*CL763;
B(7,4,5)=-2*CL754;
B(7,4,6)=-2*CL764;
B(7,5,6)=-2*CL765;
C(1,2)=CI212;
C(1,3)=CL313;
C(1,4)=CL414;
C(1,5)=CL515;
C(1,6)=CL616;
C(1,7)=CJ717;
C(2,1)=-CI211;
C(2,3)=CL323;
C(2,4)=CL424;
```

```
C(2,5)=CL525;
C(2,6)=CL626;
C(2,7)=CJ727;
C(3,1)=-CL311;
C(3,2)=-CI322;
C(3,4)=CI434;
C(3,5)=CL535;
C(3,6)=CL636;
C(3,7)=CJ737;
C(4,1)=-CL411;
C(4,2)=-CL422;
C(4,3)=-CI433;
C(4,5)=CL545;
C(4,6)=CL646;
C(4,7)=CJ747;
C(5,1)=-CL511;
C(5,2)=-CL522;
C(5,3)=-CL533;
C(5,4)=-CI544;
C(5,6)=CI656;
C(5,7)=CJ757;
C(6,1)=-CL611;
C(6,2)=-CL622;
C(6,3)=-CL633;
C(6,4)=-CL644;
C(6,5)=-CI655;
C(6,7)=CJ767;
C(7,1)=-CL711;
C(7,2)=-CL722;
C(7,3)=-CL733;
C(7,4)=-CL744;
C(7,5)=-CL755;
C(7,6)=-CI766;
```

## D.2. Modelo dinámico generado por HEMERO

```
%---------- Modelo y parámetros del Mitsubishi PA10 7CE ---------- %
%-------------------------------------------------------------------%
% Autor:          Juan F. Guerra Cano
% Creado:         04/03/2019
% Modificado:     25/04/2019
%-------------------------------------------------------------------%
%---------------- Parámetros de Denavit-Hartenberg ---------------- %
%-------------------------------------------------------------------%
%  i  |  a_i [m]  |  alpha_i [rad]  |  d_i [m]  |  theta_i [rad]  |
%-------------------------------------------------------------------%
%  1  |     0     |        0        |     0     |        q1       |
%-------------------------------------------------------------------%
%  2  |     0     |      -pi/2      |     0     |        q2       |
%-------------------------------------------------------------------%
%  3  |     0     |       pi/2      |   0.450   |        q3       |
%-------------------------------------------------------------------%
%  4  |     0     |      -pi/2      |     0     |        q4       |
%-------------------------------------------------------------------%
%  5  |     0     |       pi/2      |   0.480   |        q5       |
%-------------------------------------------------------------------%
%  6  |     0     |      -pi/2      |     0     |        q6       |
%-------------------------------------------------------------------%
%  7  |     0     |       pi/2      |   0.070   |        q7       |
%-------------------------------------------------------------------%

q=[q1 q2 q3 q4 q5 q6 q7];
qp=[q1p q2p q3p q4p q5p q6p q7p];

%-- Parámetros dinámicos y cinemáticos del robot manipulador PA10 7CE --- %

%    a    alfa    theta    d    sigma    m     rx      ry      rz       Ixx          Iyy          Izz        Ixy  Iyx  Ixz
       Jm          G       B          Tc+      Tc-
dyn=[0   -pi/2    q1       0     0     9.28   0.00    0.00   -0.011   0.122706     0.122706     0.0530564   0    0    0
     0.000037     1    14.845    13.885    13.365;
     0    pi/2    q2       0     0     4.51   0.00   -0.188   0.0     0.055035     0.055035     0.018144    0    0    0
          0.000037     1    14.295     22.33     21.8;
     0   -pi/2    q3     0.45    0     5.64   0.00    0.00   -0.035   0.008124     0.008124     0.00572413  0    0    0
          0.0000588    1     5.57       5.66      5.21;
     0    pi/2    q4       0     0     2.04   0.00   -0.164   0.0     0.008124     0.008124     0.00572413  0    0    0
          0.0000909    1     6.825      5.685     6.395;
     0   -pi/2    q5     0.48    0     2.61   0.00    0.00   -0.084   0.002546007  0.002546007  0.002747347 0    0    0
          0.000227     1     0.885      2.005     2.11;
     0    pi/2    q6       0     0     2.07   0.00   -0.042   0.0     0.002977     0.002977     0.001141    0    0    0
          0.0005       1     1.722      3.807     3.865;
     0      0     q7     0.07    0     1.62   0.00    0.00    0.018   0.0005294    0.0005294    0.0004      0    0    0
          0.000556     1     0.7055     2.105     2.00];

M = inertia(dyn,q);
C = coriolis(dyn,q,qp);
g = gravity(dyn,q);
```

# Bibliografía

[Alanis et al., 2007] Alanis, A. Y., Sanchez, E. N., and Loukianov, A. G. (2007). Discrete-time adaptive backstepping nonlinear control via high-order neural networks. *IEEE Transactions on Neural Networks*, 18(4):1185–1195.

[Campa and Kelly, 2008] Campa, R. and Kelly, R. (2008). Winmechlab: A windows-based software tool for real-time control of mechatronic systems. *IFAC Proceedings Volumes*, 41(2):9755–9760.

[Castrejon-Lozano, 2008] Castrejon-Lozano, J. G. (2008). *Localización Tridimensional: Aplicación a Robots móviles*. Tesis de Doctorado, Instituto Tecnológico de La Laguna.

[Dimirovski et al., 2012] Dimirovski, G. M., Ying, J., and Xu, J. (2012). Discrete-time Unscented Kalman filter: Comprehensive study of stochastic stability. *Itzhack Y. Bar-Itzhack Memorial Symposium on Estimation, Navigation, and Spacecraft Control. Haifa, Israel*.

[Felix et al., 2005] Felix, R. A., Sanchez, E. N., and Loukianov, A. G. (2005). Avoiding controller singularities in adaptive recurrent neural control. *IFAC Proceedings Volumes*, 38(1):109–114.

[Flores-Ávila, 2014] Flores-Ávila, J. A. (2014). Control Descentralizado Difuso Adaptable: Aplicación al Robot Mitsubishi PA10-7CE. Tesis de Maestría, Instituto Tecnológico de La Laguna.

[García-Hernández et al., 2017] García-Hernández, R., Lopez-Franco, M., Sanchez, E. N., Alanis, A. Y., and Ruz-Hernandez, J. (2017). *Decentralized Neural Control: Application to Robotics*, volume 96 of *Studies in Systems, Decision and Control*. Springer, Switzerland.

[García-Hernández et al., 2009] García-Hernández, R., Sanchez, E. N., Santibáñez, V., Llama, M. A., and Bayro-Corrochano, E. (2009). Real-time decentralized neural block controller for a robot manipulator. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 906–911. IEEE.

[García-Hernández et al., 2011] García-Hernández, R., Sanchez, E. N., Santibáñez, V., and Ruz-Hernandez, J. (2011). Decentralized neural block control for an industrial PA10-7CE robot arm.

[Ge et al., 2004] Ge, S. S., Zhang, J., and Lee, T. H. (2004). Adaptive neural network control for a class of mimo nonlinear systems with disturbances in discrete-time. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(4):1630–1645.

[Gupta and Rao, 1994] Gupta, M. M. and Rao, D. H. (1994). *Neuro-control systems: theory and applications*. IEEE press.

[Haykin, 2004] Haykin, S. (2004). *Kalman filtering and neural networks*, volume 47. John Wiley and Sons.

[Hespanha, 2018] Hespanha, J. P. (2018). *Linear systems theory*. Princeton university press.

[Jin, 1998] Jin, Y. (1998). Decentralized adaptive fuzzy control of robot manipulators. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 28(1):47–57.

[Julier et al., 1996] Julier, S., Uhlmann, J., and Durrant-Whyte, H. (1996). A new approach for the nonlinear transformation of means and covariances in linear filters. *IEEE Transactions on Automatic Control*, 92.

[Julier and Uhlmann, 1997] Julier, S. J. and Uhlmann, J. K. (1997). A new extension of the Kalman Filter to nonlinear systems. *Signal processing, sensor fusion, and target recognition VI*, 3068(International Society for Optics and Photonics.):182–194.

[Julier et al., 1995] Julier, S. J., Uhlmann, J. K., and Durrant-Whyte, H. F. (1995). A new approach for filtering nonlinear systems. In *Proceedings of 1995 American Control Conference-ACC'95*, volume 3, pages 1628–1632. IEEE.

[Kalman, 1960] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45.

[Kalman and Bucy, 1961] Kalman, R. E. and Bucy, R. S. (1961). New results in linear filtering and prediction theory. *Journal of Basic Engineering*, 83(1):95–108.

[Karvonen et al., 2014] Karvonen, T. et al. (2014). Stability of linear and non-linear Kalman filters. *Helda, University of Helsinki*.

[Kelly and Santibáñez, 2003] Kelly, R. and Santibáñez, V. (2003). *Control de movimiento de robots manipuladores*. Pearson educación.

[Khalil, 2002] Khalil, H. K. (2002). *Nonlinear Systems*. Prentice Hall.

[Khalil et al., 1989] Khalil, W., Bennis, F., Chevallereau, C., and Kleinfinger, J. (1989). SYMORO: A software package for the symbolic modelling of robots. In *Proceedings of the 20th ISIR*.

[Li et al., 2012] Li, S., Chen, S., Liu, B., Li, Y., and Liang, Y. (2012). Decentralized kinematic control of a class of collaborative redundant manipulators via recurrent neural networks. *Neurocomputing*, 91:1–10.

[Lin and Byrnes, 1994] Lin, W. and Byrnes, C. I. (1994). Design of discrete-time nonlinear control systems via smooth feedback. *IEEE Transactions on Automatic Control*, 39(11):2340–2346.

[Liu, 1999] Liu, M. (1999). Decentralized control of robot manipulators: nonlinear and adaptive approaches. *IEEE Transactions on Automatic control*, 44(2):357–363.

[Márquez, 1995] Márquez, A. (1995). Diseño de un manipulador de 2 grados de libertad. *Informe Técnico. Comunicaciones Académicas, Series Electrónica y Telecominicaciones*, CICESE 33 pp. CTETT9503.

[Maza and Ollero, 2001] Maza, J. I. and Ollero, A. (2001). Hemero: a Matlab-Simulink toolbox for robotics. In *1st Workshop on Robotics Education and Training*, pages 43–50.

[McCulloch and Pitts, 1943] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

[Nguyen and Widrow, 1990] Nguyen, D. H. and Widrow, B. (1990). Neural networks for self-learning control systems. *IEEE Control systems magazine*, 10(3):18–23.

[Ogata, 2003] Ogata, K. (2003). *Ingeniería de control moderna*. Pearson Educación.

[Pascual, 2004] Pascual, A. (2004). EKF y UKF: dos extensiones del filtro de Kalman para sistemas no lineales aplicadas al control de un péndulo invertido. *Monografía para el curso: Tratamiento Estadístico de Señales*, page 35.

[Pholsiri, 2004] Pholsiri, C. (2004). *Task-based decision making and control of robotic manipulators*. PhD thesis, Doctoral Dissertation, The University of Texas at Austin.

[Ramírez-Woo, 2008] Ramírez-Woo, C. (2008). Modelado Dinámico y Control en modo Par del Robot Mitsubishi PA10-7CE. Tesis de Maestría, Instituto Tecnológico de La Laguna.

[Reyes and Kelly, 1996] Reyes, F. and Kelly, R. (1996). A direct-drive robot for control research. *CICESE, México, IASTED-ACTA Press, Anaheim, CA, USA. Hamza, MH*, pages 181–4.

[Reyes and Kelly, 1997] Reyes, F. and Kelly, R. (1997). Experimental evaluation of identification schemes on a direct drive robot. *Robotica*, 15(5):563–571.

[Reyes and Kelly, 2001] Reyes, F. and Kelly, R. (2001). Experimental evaluation of model-based controllers on a direct-drive robot arm. *Mechatronics*, 11(3):267–282.

[Rhudy and Gu, 2013] Rhudy, M. and Gu, Y. (2013). Understanding nonlinear Kalman filters, part II: An implementation guide. *Interactive Robotics Letters*. http://www2.statler.wvu.edu/ %7Eirl/page13.html.

[Rhudy et al., 2013] Rhudy, M., Gu, Y., Gross, J., Gururajan, S., and Napolitano, M. (2013). Sensitivity and robustness analysis of EKF and UKF design parameters for GPS/INS sensor fusion. *AIAA Journal of Aerospace Information Systems*, 10(3):131–143.

[Rhudy et al., 2011] Rhudy, M., Gu, Y., Gross, J., and Napolitano, M. R. (2011). Evaluation of matrix square root operations for UKF within a UAV GPS/INS sensor fusion application. *International Journal of Navigation and Observation*, 2011.

[Rugh and Rugh, 1996] Rugh, W. J. and Rugh, W. J. (1996). *Linear system theory*, volume 2. prentice hall Upper Saddle River, NJ.

[Safaric and Rodic, 2000] Safaric, R. and Rodic, J. (2000). Decentralized neural-network sliding-mode robot controller. In *Industrial Electronics Society, 2000. IECON 2000. 26th Annual Conference of the IEEE*, volume 2, pages 906–911. IEEE.

[Sanchez et al., 2008] Sanchez, E. N., Alanís, A. Y., and Loukianov, A. G. (2008). *Discrete-time high order neural control: trained with Kalman filtering*, volume 112. Springer Science, Business Media.

[Siljak, 2011] Siljak, D. D. (2011). *Decentralized control of complex systems*. Courier Corporation.

[Simon, 2006] Simon, D. (2006). *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons.

[Sánchez and Alanís, 2006] Sánchez, E. N. and Alanís, A. Y. (2006). *Redes neuronales: conceptos fundamentales y aplicaciones a control automático*. Pearson Educación.

[Sánchez-Mazuca, 2013] Sánchez-Mazuca, S. (2013). *Análisis y evaluación de métodos de estimación paramétrica aplicados al control de sistemas mecatrónicos*. Tesis de Doctorado, Instituto Tecnológico de La Laguna.

[Utkin et al., 2009] Utkin, V., Guldner, J., and Shi, J. (2009). *Sliding mode control in electromechanical systems*. CRC press.

[Van Der Merwe et al., 2004] Van Der Merwe, R. et al. (2004). *Sigma-point Kalman filters for probabilistic inference in dynamic state-space models*. PhD thesis, OGI School of Science & Engineering at OHSU.

[Walpole et al., 1999] Walpole, R. E., Myers, R. H., and Myers, S. L. (1999). *Probabilidad y estadística para ingenieros*. Pearson Educación.

[Wan and Van Der Merwe, 2000] Wan, E. A. and Van Der Merwe, R. (2000). The unscented Kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158. IEEE.

[Zhang and Ioannou, 1996] Zhang, Y. and Ioannou, P. A. (1996). A new class of nonlinear robust adaptive controllers. *International Journal of Control*, 65(5):745–769.