Universidad Autónoma de Querétaro

Facultad de Ingeniería

Maestría en Instrumentación y

Control Automático

Control neuronal para un robot rígido de dos grados de libertad

TESIS

Que como parte de los requisitos para obtener el grado de

Maestro en Ciencias

Presenta:
Ing. Francisco Javier Ramírez Rangel

Dirigido por:
M.C Alfonso Noriega Ponce

SINODALES

M.C Alfonso Noriega Ponce
Presidente — Firma

Dr. Víctor Manuel Hernández Guzmán
Secretario — Firma

Dr. Roberto Valentín Carrillo Serrano
Vocal — Firma

Dr. Roberto Augusto Gómez Loenzo
Suplente — Firma

M.C. Guillermo Ronquillo Lomeli
Suplente — Firma

Dr. Aurelio Domínguez González
Director de la Facultad

Dr. Irineo Torres Pacheco
Director de Investigación y Posgrado

Centro Universitario
Querétaro, Qro.
Marzo de 2012
México

RESUMEN

En este trabajo de tesis se propone un controlador neuronal para el manejo de un robot manipulador de dos grados de libertad con incertidumbre parámetrica, concretamente, con desconocimiento de los vectores de gravedad, inercias, Coriolis y fricción. El controlador consta de un control Proporcional-Derivativo (PD) con compensación por redes neuronales artificiales. Mediante un análisis de Lyapunov se presentan las condiciones de estabilidad y se demuestra que todas las señales en lazo cerrado son finalmente uniformemente acotadas (UUB). Después, se muestran los resultados de simulación y experimentales de las leyes de control a comparar; PD con compensación, Proporcional-Integral-Derivativo (PID) y el control neuronal.

Palabras claves: (Control neuronal, Función de Lyapunov, incertidumbres parámetricas).

SUMMARY

This thesis proposes a neural network controller for operating of a robot manipulator with two degrees of freedom with parametric uncertainty, specifically, the lack of knowledge of the friction, Coriolis, inertias and gravity vectors. The proposed controller is a Proportional Derivative with neural networks compensation. A Lyapunov analysis determines the condition for stability, it show that all signals in the closed-loop system are uniformly ultimately bounded. Therefore, we present the experimental and simulation results for the follows control laws we compare: PD compensation, PID and neural controller.

Keywords: (Neural controller, Lyapunov Function, parametric uncertainties).

AGRADECIMIENTOS

Son numerosas las personas que de un modo u otro han hecho posible la realización de este trabajo, tanto en los aspectos técnicos, intercambiando ideas y dando soporte a los medios utilizados para su desarrollo, como en el aspecto humano, brindándome el apoyo necesario cuando me hizo falta. A todos ellos mi más sincero agradecimiento.

No puedo, sin embargo, dejar de mencionar de manera particular: A mi director de tesis, M.C. Alfonso Noriega Ponce, por su guía durante este largo viaje, a veces lleno de obstáculos, a veces satisfactorio.

Al Dr. R. Valentín Carrillo Serrano, que me inició en la curiosidad científica y en el estudio de la teoría y aplicaciones de la ingeniería de control en aquellos lejanos tiempos en los cuales realizaba mi proyecto de fin de semestre.

Al profesor Dr. Víctor Manuel Hernández Guzmán que tanto me ha enseñado sobre la teoría de control clásico y el placer de la buena docencia.

A los compañeros de la Universidad Autónoma de Querétaro Departamento de la Facultad de Ingeniería con los cuales he compartido tanto largas jornadas de trabajo como momentos de recreo y tertulia.

Finalmente y de manera muy especial, a mis padres Serafín y María, por el aliento y la comprensión que han tenido conmigo durante estos largos años. Ellos saben que mi agradecimiento y mis sentimientos son difícilmente expresables con palabras.

Deseo, también, agradecer la ayuda recibida por el CONACYT a través de su *Beca para la realización de mis estudios.*

Í N D I C E

Página

RESUMEN i
SUMMARY ii
AGRADECIMIENTOS iii
Í N D I C E iv
ÍNDICE DE TABLAS vi
ÍNDICE DE FIGURAS vii
I. INTRODUCCIÓN 1
1.1. Antecedentes 1
1.2. Descripción del problema 2
1.3. Justificación 5
1.4. Objetivos 7
1.5. Hipótesis 7
1.6. Alcances 7
1.7. Organización de la tesis 8
II. REVISIÓN DE LITERATURA 9
2.1. Redes neuronales artificiales 9
2.2. Modo de una neurona artificial 10
2.3. Modelo matemático 11
2.4. Tipos de funciones de activación. 13
2.5. Arquitecturas neuronales 14
2.5.1. Redes neuronales unicapa 14
2.5.2. Redes neuronales multicapa 15
2.5.3. Redes neuronales recurrentes 15
2.6. Algoritmo de retropropagación 16
2.7. Robots manipuladores 17
2.7.1. Sistema mecánico 19
2.7.2. Actuadores 20
2.7.3. Sensores y sistemas de control 20

2.7.4. Dinámica de los robots manipuladores 21
2.7.5. Método de Euler-Lagrange 21
2.8. Estado del arte 23
III. METODOLOGÍA 27
3.1. Modelado dinámico del brazo articulado 27
3.2. Arquitectura de la red neuronal propuesta 33
3.3. Algoritmo de aprendizaje 35
3.4. Inicialización de parámetros 40
3.5. Control de un robot manipulador 41
3.5.1. Control tipo PD de un robot manipulador 41
3.5.2. Control neuronal 43
IV. RESULTADOS Y DISCUSIÓN 49
4.1. Simulaciones 49
4.1.1. Modelo dinámico 49
4.1.2. Controlador PD con compensación 52
4.1.3. Controlador PID 57
4.1.4. Controlador neuronal 61
4.2. Resultados experimentales 66
4.2.1. Controlador PD con compensación 66
4.2.2. Controlador PID 71
4.2.3. Controlador neuronal 74
V. CONCLUSIONES 80
BIBLIOGRAFÍA 82
APÉNDICES 85
A. Prototipo experimental 86
B. Diagramas electrónicos del proyecto. 90
B.1 Adquisición de datos 90
B.2. Señales de control. 91
B.3 Etapa de potencia. 92
C. Código de programas 92

ÍNDICE DE TABLAS

Tabla **Página**

4.1. Parámetros de simulación del robot rígido. 50
4.2. Índice de desempeño del controla PD con compensación para la prueba 1. 53
4.3. Índice de desempeño del controla PD con compensación para la prueba 2. 55
4.4. Índice de desempeño del controla PID para la prueba 1. 58
4.5. Índice de desempeño del controla PID con compensación para la prueba 1. 60
4.6. Índice de desempeño del controla PD sin compensación para la prueba 1. 63
4.7. Índice de desempeño del controla neuronal para la prueba 1. 63
4.8. Índice de desempeño del control neuronal para la prueba 2. 65
4.9. Índice de desempeño del control PD con compensación para la prueba experimental 1. 67
4.10. Índice de desempeño del controla PD con compensación para la prueba experimental 2. 70
4.11. Índice de desempeño del controla PID para la prueba experimental1. 72
4.12. Índice de desempeño del controla neuronal con 8 neuronas ocultas (prueba experimental 1) 78
4.13. Índices de desempeño cuando existe un aumento de neuronas (prueba experimental 1). 78
4.14. Comparación de índices de desempeño entre el Algoritmo BP y el recursivo con 24 neuronas ocultas (prueba experimental 1) 79
5.1. Comparación de índices de desempeño (prueba experimental 1). 81

ÍNDICE DE FIGURAS

Figura **Página**

1.1. Diagrama a bloques: Control par calculado (Kelly y Santibáñez, 2003). 5
2.1. Representación de una red neuronal artificial. 10
2.2. Modelo de una neurona artificial 10
2.3. Otro modelo de una neurona artificial. 12
2.4. Funciones de activación: (a) Función escalón. (b) Función sigmoidal. 14
2.5. Estructuras neuronales: (a) Red neuronal unicapa. (b) Red neuronal multicapa. (c) Red neuronal recurrente con lazos de retroalimentación. (d) Red neuronal recurrente con neuronas ocultas 16
2.6. Robot manipulador y robot móvil 17
2.7. Grados de libertad 17
2.8.a) Motores de corriente continua. b) Motor paso a paso 20
2.9. Esquema de un robot típico 22
3.1. Robot rígido de dos grados de libertad 27
3.2. Análisis geométrico del manipulador de 2 grados de libertad 29
3.3. Red neuronal recurrente de tres capas 34
3.4. Controlador neuronal. 35
4.1. Resultados de la simulación 1 para valores de $\tau 1 = 0$ Nm y $\tau 2 = 0.1$Nm. 51
4.2. Resultados de la simulación 2 para valores de $\tau 1 = 2$ Nm y $\tau 2 = 0.1$Nm. 51
4.3. Resultados de la simulación 2 para valores de $\tau 1 = 3$ Nm y $\tau 2 = 0.8$Nm. 52
4.4. Evolución de las trayectorias del controlador PD con compensación prueba 1. 54
4.5. Señales de torque del control PD con compensación prueba 1. 55
4.6. Evolución de las trayectorias control PD con compensación prueba 2. 56
4.7. Señales de pares del control PD con compensación-prueba 2 57
4.8. Errores del control PD con compensación prueba 2 57
4.9. Evolución de las trayectorias de la prueba 1. 59
4.10. Señales de control de la prueba 1. 59
4.11. Evolución de las trayectorias de la prueba 2. 60

4.12. Señales de control de la prueba 2. 61
4.13. Errores de la prueba 2. 61
4.14. Control PD sin compensación. 63
4.15. Evolución de las trayectorias de la prueba 1 del controlador neuronal. 64
4.16. Señales de control para cada eslabón del controlador neuronal de la prueba 1. 64
4.17. Evolucion de las posiciones articulares del control neuronal de la prueba 2. 65
4.18. Pares de corrección del control neuronal de la prueba 2. 66
4.19. Trayectorias para la articulación 1 y 2 para el control PD con compensación de gravedad. 68
4.20. Señales de control tau1[N-m] para el control PD con compensación (prueba experimental 1). 68
4.21. Señales de control tau2[N-m] para el control PD con compensación (prueba experimental 1). 69
4.22. Trayectorias para la articulación 1 y 2 para el control PD con compensación con incertidumbre (prueba experimental 2). 70
4.23. Señales de control tau1[N-m] para el control PD con compensación (prueba experimental 2). 71
4.24. Señales de control tau2[N-m] para el control PD con compensación (prueba experimental 2). 71
4.25. Evolucion de las trayectorias para el control PID en la prueba experimental 1. 73
4.26. Señales de control tau1[N-m] para el control PID (prueba experimental 1). 73
4.27. Señales de control tau2[N-m] para el control PID (prueba experimental 1). 74
4.28. Trayectorias para la articulación 1 y 2 para el control neuronal con 8 neuronas ocultas (prueba experimental 1). 75
4.29: Señales tau1 [N-m] para el control neuronal con 8 neuronas ocultas (prueba experimental 1). 76
4.30. Señale tau2[N-m] para el control neuronal con 8 neuronas ocultas (prueba experimental 1). 76
4.31. Error de posición 1 para el control neuronal con 8 neuronas ocultas (prueba experimental 1). 77

4.32. Error de posición 2 para el control neuronal con 8 neuronas ocultas (prueba experimental 1)..... 77
4.33. Compensación neuronal con 8 neuronas ocultas (prueba experimental 1). 79
5.1. Comparación de los diferentes esquemas de control (prueba experimental 1). 81

I. INTRODUCCIÓN

En este capítulo se muestra una descripción general del proyecto realizado; para una mejor comprensión de la investigación se presenta una pequeña introducción, se muestra la descripción del problema a resolver, los objetivos que se pretenden alcanzar, las razones por las cuales se decidió trabajar en este tema, así como también los alcances del proyecto, finalmente se muestra la organización del trabajo.

1.1. Antecedentes

Dentro de la robótica en general, una de las áreas más estudiadas es la del control de brazos robóticos; dichos mecanismos se caracterizan por ser sistemas dinámicos de múltiples entradas y múltiples salidas. Se puede mencionar que el problema de controlar un robot manipulador es uno de los aspectos más complejos de esta área, que en numerosas ocasiones se ha evadido.

Desde hace varios años, algunas estrategias han sido utilizadas para resolver el control de posición para robots manipuladores, por ejemplo, Spong y Vidyasagar, (1989) y Ollero, (2001) proponen un esquema de control basado en el conocimiento del modelo dinámico; en Ortega y Spong, (1989) se propone un esquema de control adaptativo.

De los robots manipuladores que existen en la industria, un gran número utiliza un control proporcional-derivativo (control PD) cuyo proceso de sintonización, aun cuando tiene una estructura sencilla y de fácil realización, suele ser complicado: en varios casos se requiere de la experiencia o experimentación. Sin embargo, el control PD es incapaz de hacer que el robot siga una trayectoria continua y variante en el tiempo (una trayectoria cuyos valores cambien constantemente de manera continua) o pueda alcanzar una posición deseada sin que se presente un error en estado estable. Esta desventaja en el control PD

hace que el manipulador no pueda realizar tareas como: maquinado de piezas, soldadura por arco y perforación de placas.

En trabajos de investigación se muestra que es necesario añadir al control PD términos adicionales de compensación para garantizar que el robot siga una trayectoria variante y continua en forma exacta (Spong, 1992; Murray *et al.*, 1994; Kelly y Santibáñez, 2003; Craig, 2006); dichos términos de compensación deben considerar los efectos de gravedad, matriz de inercias, matriz de fuerza centrífuga y de Coriolis, fricción no lineal y dinámicas no modeladas del manipulador.

Otra desventaja del control PD es la medición de velocidad de cada eslabón del robot, lo cual introduce la necesidad de utilizar sensores de velocidad; de estos, los más utilizados son los tacómetros, pero presentan varias desventajas, a saber: entregar la medición contaminada de ruido (Canudas-de-Wit y Fixot, 1992); ser muy costosos; error en la medición, etc. Por otra parte, los sensores de posición conocidos como "encoders" son ampliamente utilizados y cualquier sistema que presente movimiento requiere de ellos.

1.2. Descripción del problema

De forma habitual el modelo dinámico de un manipulador de *n* articulaciones se puede representar por medio de la ecuación (1.1) que tiene la forma:

$$\tau = \mathrm{M}(\mathrm{q})\ddot{\mathrm{q}} + \mathrm{C}(\mathrm{q},\dot{\mathrm{q}})\dot{\mathrm{q}} + \mathrm{G}(\mathrm{q}) + \mathrm{F}\dot{\mathrm{q}} \qquad (1.1)$$

donde $M(q)\,\epsilon\, R^{nxn}$ es la matriz de inercias, $C(q,\dot{q})\dot{q}\,\epsilon\, R^{n}$ es el vector de fuerzas centrífugas y de Coriolis, $G(q)\epsilon\, R^{n}$ es el vector de fuerzas gravitacionales $F\dot{q}\,\epsilon\, R^{n}$ es el vector de fricción y $\tau\,\epsilon\, R^{n}$ es un vector de fuerzas y pares aplicados en la uniones mediante los actuadores. Los vectores $q,\dot{q},\ddot{q}\,\epsilon\, R^{n}$ indican la posición, velocidad y aceleración articular, respectivamente.

El problema de control de seguimiento de robots manipuladores se puede formular de la siguiente manera: si se considera la ecuación general que representa el

comportamiento de un robot (1.1), dado un conjunto de funciones vectoriales acotadas $q_d, \dot{q}_d, \ddot{q}_d$ descritas como posiciones, velocidades y aceleraciones deseadas, se trata de encontrar una función vectorial τ, de tal forma que las posiciones q asociadas a las coordenadas articulares del robot sigan con precisión a q_d.

El cálculo del vector τ llamado "**ley de control**", involucra generalmente a una función vectorial no lineal de $q, \dot{q}$ y $\ddot{q}$ debido a que cada matriz y vector $(M(q), C(q, \dot{q}), G(q))$ están formados por funciones seno y coseno, además, la dinámica del manipulador contiene parámetros tales como la masa e inercia, que pueden variar en el tiempo; es decir, si se requiere que el mecanismo tome un objeto en un tiempo establecido, la masa de los eslabones cambia, por lo tanto, hace más complejo el control de dicho sistema.

Existen varios esquemas de control convencional, como el control PD con compensación de gravedad, control par-calculado, los cuales poseen parámetros de diseño cuyos valores numéricos se determinan en función del modelo dinámico del manipulador al cual controlan. La dificultad principal de estas técnicas de control es el conocimiento exacto de los parámetros; de éstos, las masas, inercias y longitudes de cada eslabón son difíciles de medir o conocer en algunos casos. Además, existen términos que no se conocen como la fricción no lineal y la dinámica no modelada del robot. En el caso de que las fuerzas gravitacionales y de fricción no puedan ser modeladas, el controlador Proporcional-Integral-Derivativo (PID) puede ser el mejor controlador ya que sus ganancias pueden ser ajustadas en forma independiente (Spong y Vidyasagar, 1989). Para atenuar el error en estado estable, suele aumentarse la ganancia de la parte integral, sin embargo, esto lleva a un error grande en el transitorio que puede producir inestabilidad.

Por ejemplo, la ecuación (1.2) explica esto con más detalle, porque describe la estructura del controlador par-calculado, el cual trata de compensar los efectos de gravedad, centrífugos y rozamientos generando un par apropiado:

$$\tau = \hat{M}(q)\left[\ddot{q}_d + K_v\tilde{\dot{q}} + K_p\tilde{q}\right] + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q) + \hat{F}\dot{q} \quad (1.2)$$

Siendo $\hat{M}(q), \hat{C}(q,\dot{q})$ y $\hat{G}(q)$ y $\hat{F}\dot{q}$ estimaciones de las matrices y vectores correspondientes. Donde $\tilde{q}, \tilde{\dot{q}} \in R^n$ indican el error de posición y velocidad respectivamente:

$$\tilde{q} = q_d - q; \tilde{\dot{q}} = \dot{q}_d - \dot{q} \quad (1.3)$$

y $K_p, K_v \in R^{nxn}$ representan las ganancias proporcional y derivativa.

Sustituyendo (1.2) en (1.1) se obtiene:

$$\tau = \hat{M}(q)\left[\ddot{q}_d + K_v\tilde{\dot{q}} + K_p\tilde{q}\right] + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q) + \hat{F}\dot{q} = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + Fq \quad (1.4)$$

De donde, restando en ambos lados $\widehat{M}(q)\ddot{q}$, se llega a:

$$\ddot{\tilde{q}} + K_v\dot{\tilde{q}} + K_p\tilde{q} = \hat{M}(q)^{-1}\left[\left(M(q) - \hat{M}(q)\right)\ddot{q} + \left(C(q,\dot{q}) - \hat{C}(q,\dot{q})\right)\dot{q} + \left(G(q) - \hat{G}(q)\right) + \left(F(q,\dot{q}) - \hat{F}(q,\dot{q})\right)\right] \quad (1.5)$$

Si el modelo del mecanismo es exacto, es decir, $M = \widehat{M}, C = \hat{C}, G = \hat{G}$ y $F = \hat{F}$, el lado derecho de (1.5) es cero.

$$\ddot{\tilde{q}} + K_v\dot{\tilde{q}} + K_p\tilde{q} = 0 \quad (1.6)$$

En este caso, se tiene n ecuaciones lineales de segundo orden desacopladas. Los coeficientes de (1.6) se determinan de acuerdo a la respuesta de un sistema de control típico de segundo orden. De este modo, el diseño del controlador requiere el conocimiento exacto del modelo del robot

En la figura (1.1) se muestra un diagrama formado por un controlador común en lazo cerrado con un robot.

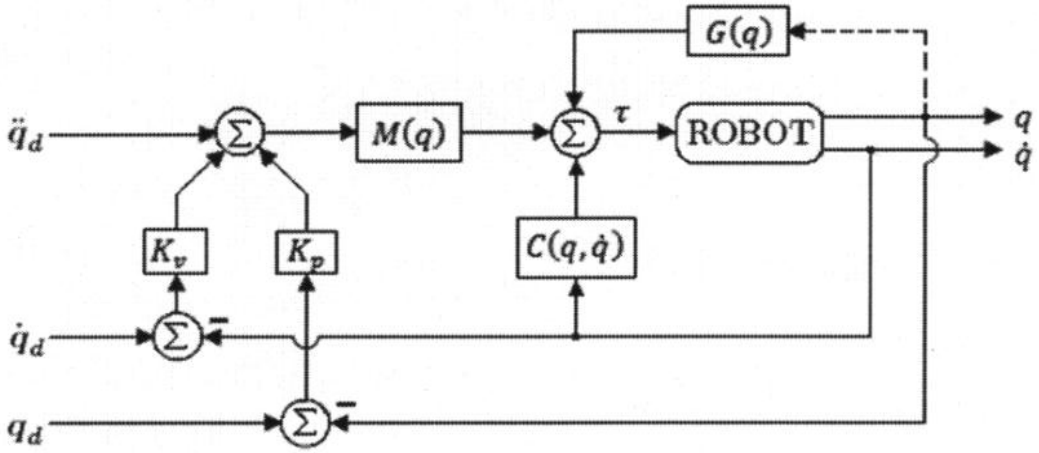


Figura 1.1. Diagrama a bloques: Control par calculado (Kelly y Santibáñez, 2003).

Es importante mencionar que varios mecanismos disponen de sensores de posición y velocidad para cada articulación, por lo que los vectores q y $\dot{q}$ son medibles y pueden ser empleados en los controladores. Algunos otros utilizan sólo sensores de posición, por lo tanto, es necesario estimar la velocidad a partir de la medición de posición, ya sea mediante filtrado, ya sean observadores o por diferenciación numérica.

1.3. Justificación

En la actualidad existen métodos a la hora de llevar a cabo el control de robots manipuladores, por ejemplo: control convencional, control inteligente entre otros.

El controlador más común es un PID con parámetros fijos. La acción derivativa es básica para suavizar las fuertes oscilaciones que aparecen. Sin embargo, se sabe que los robots son sistemas fuertemente no lineales, ningún controlador con parámetros fijos puede proporcionar un buen control en todo el espacio de trabajo.

Los robots controlados por PID serán excesivamente lentos en algunas regiones y pueden oscilar en otras. Conseguir movimientos suaves y precisos sobre una trayectoria

especificada requiere un conocimiento exacto de los parámetros cinemáticos y dinámicos, y en estas condiciones los algoritmos de control alcanzan un alto grado de complejidad.

En un intento de emular el comportamiento humano se han desarrollado sistemas de control adaptivo, en los cuales los parámetros del controlador se ajustan automáticamente dependiendo de la región del espacio de estados (posiciones y velocidades). Estos sistemas se comportan mejor que los sistemas con parámetros fijos, pero con el costo adicional de una gran complejidad. Es más, el diseño de un buen sistema de control adaptivo requiere incluso modelos más precisos de los robots.

Por lo mencionado anteriormente, es evidente la necesidad de conocer la dinámica del sistema para poder controlar el mecanismo. Como alternativa a estas estrategias de control se han desarrollado estrategias de control inteligente y, más concretamente, en redes neuronales artificiales. La principal ventaja que ofrecen estas estructuras es que el modelo dinámico del mecanismo se puede estimar mientras que los controladores existentes, controladores convencionales, a saber; PD compensado, par calculado, requieren del conocimiento exacto de los términos dinámicos. Por otro lado, una de las desventajas del control adaptable es que se necesita conocer la estructura del brazo manipulador.

Por esta razón, es necesario realizar este trabajo debido, especialmente, a que se considerará la estimación de las no linealidades por medio de redes neuronales, que tienen la capacidad de aproximar funciones no lineales, y que no se requiera conocer el modelo dinámico del sistema o planta. Estos controladores neuronales están adoptando nuevos enfoques de diseño como nuevas arquitecturas, algoritmos de aprendizaje e incluso ayuda de otros algoritmos de control para alcanzar un funcionamiento ideal.

Una característica que poseen los controladores neuronales es que no necesitan el modelo matemático del sistema a controlar, únicamente requieren de mediciones de la entrada y la salida del sistema para asignar una señal de control.

1.4. Objetivos

General:

Evaluar un controlador que, basado en la técnica de redes neuronales artificiales, permita el manejo de un robot rígido con uniones rotativas de dos grados de libertad.

Específicos:

-Calcular el modelo dinámico de un robot de dos grados de libertad (g.d.l), para que sirva como base para el desarrollo del controlador.

-Determinar el algoritmo de control para una red neuronal dinámica.

-Simular el sistema completo para observar su comportamiento cuando se aplican diferentes referencias deseadas.

-Desarrollar el análisis de estabilidad del controlador neuronal que apruebe su uso en un robot manipulador.

-Probar el control neuronal en el prototipo experimental para verificar la teoría.

1.5. Hipótesis

El controlador neuronal propuesto estima los términos no lineales que el control PD necesita para resolver el problema de seguimiento. Por lo tanto, se logra reducir el error de seguimiento a un valor cercano a cero.

1.6. Alcances

Para cumplir los objetivos propuestos el alcance del proyecto se ha extendido a los siguientes puntos:

- Realizar el control de seguimiento de un robot rígido con uniones rotativas de dos grados de libertad.
- Realizar pruebas experimentales que demuestren la factibilidad de operación de un control neuronal.
- Efectuar un análisis comparativo entre los controladores: PD compensado, PID y el control neuronal

1.7. Organización de la tesis

La organización de la presente tesis es la siguiente: en el capítulo 2 se da una introducción a la teoría de las redes neuronales, así como las bases teóricas necesarias para el desarrollo del trabajo de investigación.

En el capítulo 3, se presenta la metodología para el desarrollo del control neuronal. Se detalla, como primer punto, el desarrollo para obtener el modelo dinámico de un robot rígido de dos grados de libertad; como segundo punto, se desarrolla el algoritmo de aprendizaje de la red neuronal; finalmente, se realiza un análisis de estabilidad del sistema.

En el capítulo 4, se presentan los resultados obtenidos: primero la validación del modelo dinámico y, después, los resultados experimentales. Finalmente, en el capítulo 5, se dan a conocer las conclusiones obtenidas del trabajo realizado.

II. REVISIÓN DE LITERATURA

En este capítulo se presenta una breve revisión de los conceptos que fundamentan la teoría de las redes neuronales y de los robots manipuladores. Se inicia con las redes neuronales biológicas y los procesos que éstas involucran, así como su relación con las artificiales. Se introduce la teoría de grafos orientados aplicados a las redes neuronales. Además, se analizan brevemente las arquitecturas neuronales y su mecanismo de aprendizaje. Finalmente, se da a conocer los aspectos más relevantes de un robot manipulador.

2.1. Redes neuronales artificiales

Estas son modelos simplificados de las redes neuronales biológicas (Haykin, 1999). Tratan de extraer las excelentes capacidades del cerebro para resolver ciertos problemas complejos, por ejemplo: visión, reconocimiento de patrones, identificación o control de sistemas. En la figura (2.1) se puede observar la representación esquemática de una red neuronal artificial. Una red neuronal artificial, también llamada neuro-computadora, red conexionista, procesador paralelo distribuido, etc., es un procesador paralelo distribuido y masivamente interconectado que almacena conocimiento experimental (Haykin, 1999). Las redes neuronales artificiales presentan las siguientes características:

- El conocimiento es adquirido experimentalmente.
- Los pesos (ganancias) de interconexión (sinapsis) varían constantemente.

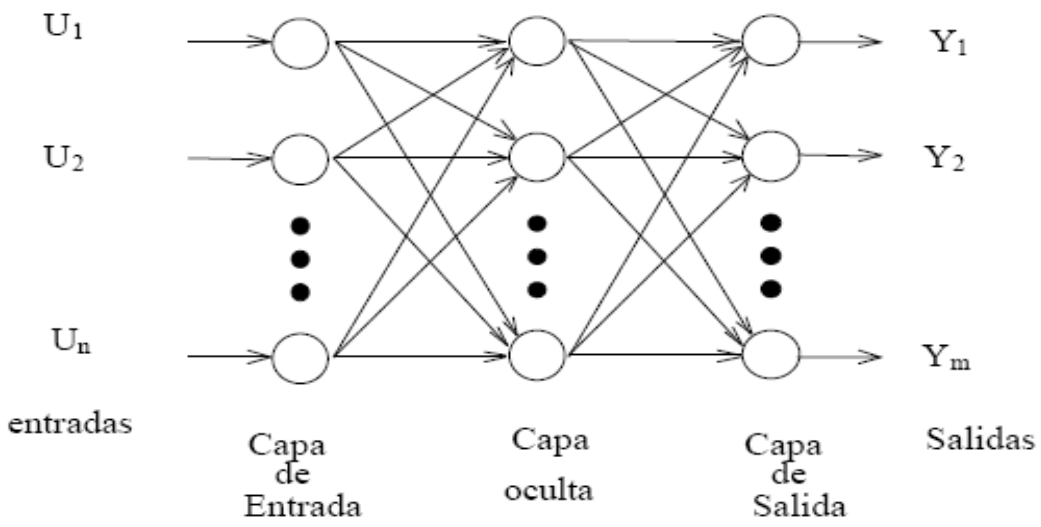


Figura 2.1. Representación de una red neuronal artificial.

2.2. Modo de una neurona artificial

La neurona es la unidad de proceso de información fundamental en una red neuronal (Haykin, 1999). En la figura (2.2) se muestra el modelo de una neurona; éste es el elemento básico de una red neuronal artificial.

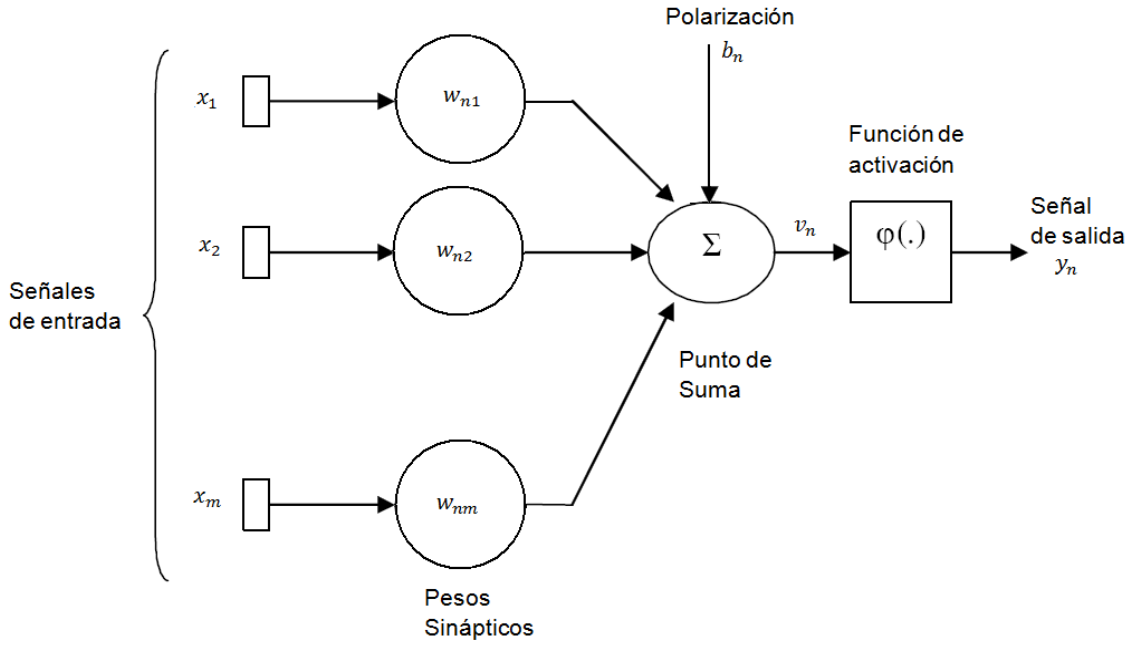


Figura 2.2. Modelo de una neurona artificial

En el modelo de una neurona presentado en la (figura 2.2) se pueden identificar cuatro elementos.

- *Enlaces de conexión.* Parametrizados por los pesos sinápticos w_{nj}. Es importante notar que el primer subíndice corresponde a la neurona receptora, mientras que el segundo corresponde a la neurona emisora. Si $w_{nj} > 0$, entonces la conexión es excitadora; así mismo, si $w_{nj} < 0$, la conexión es inhibidora.
- *Sumador* ($\sum$). Suma los componentes de las señales de entrada multiplicadas por w_{nj}.
- *Función de activación (φ).* Transformación no lineal.
- *Umbral.* Desplaza la entrada.

2.3. Modelo matemático

En términos matemáticos, es posible describir la neurona n de la (figura 2.2) por el siguiente par de ecuaciones:

$$u_n = \sum_{j=1}^{m} w_{nj} x_j \tag{2.1}$$

y

$$y_n = \varphi(u_n + b_n) \tag{2.2}$$

donde $x_1, x_2, \ldots, x_m$ son las señales de entrada; $w_{n1}, w_{n2} \ldots, w_{nm}$ son los pesos sinápticos de la neurona n; u_n es la combinación lineal de las entradas ponderadas por los pesos sinápticos; b_n es la polarización o umbral; $\varphi(\cdot)$ es la función de activación; y finalmente, y_n es la señal de salida de la neurona. La polarización es un parámetro externo de la neurona n, pero es posible considerarla como parte de las señales de entrada, de tal forma que si se combina la ecuación (2.1) y la ecuación (2.2) se tiene:

$$v_n = \sum_{j=0}^{m} w_{nj} x_j \tag{2.3}$$

y

$$y_n = \varphi(v_n) \tag{2.4}$$

A v_n se le denomina potencial de activación. En la ecuación (2.3) se le ha agregado una nueva sinapsis. Su entrada es:

$$x_0 = +1 \tag{2.5}$$

y el peso correspondiente es:

$$w_{n0} = b_n \tag{2.6}$$

De tal forma que el modelo neuronal presentado en la (figura 2.2) se puede representar como en la (figura 2.3). Estos modelos tienen una apariencia diferente pero matemáticamente son iguales. Hasta este punto sólo se ha nombrado a la función de activación; sin embargo, no se ha definido formalmente. A continuación se analizarán las funciones de activación más utilizadas en redes neuronales.

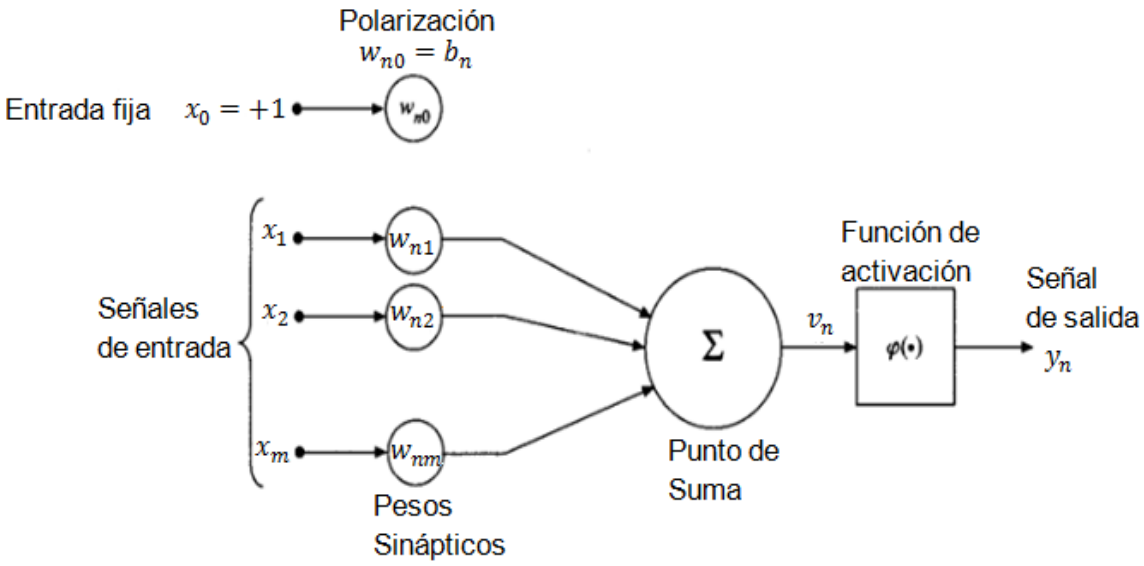


Figura 2.3. Otro modelo de una neurona artificial.

La ecuación (2.3) se pude escribir en forma matricial como:

$$v = Wx \tag{2.7}$$

donde W es un vector fila. La salida del perceptrón es:

$$y = \varphi(W^T x) \tag{2.8}$$

2.4. Tipos de funciones de activación.

Las funciones de activación, denotadas por la $\varphi(v)$, define la salida de la neurona en función de potencial de activación *v*. Se incluyen tres de los tipos básicos de la función de activación:

1. *Función escalón o umbral.* Para este tipo de función de activación, presentada en la (figura 2.4a) se tiene :

$$\varphi(v) = \begin{cases} 1 \text{ si } v \geq 0 \\ 0 \text{ si } v < 0 \end{cases} \tag{2.9}$$

Correspondientemente, la salida de la neurona *n* empleando esta función de activación queda expresada como:

$$y_n = \begin{cases} 1 \text{ si } v_n \geq 0 \\ 0 \text{ si } v_n < 0 \end{cases} \tag{2.10}$$

con v_n dada por la ecuación (2.3).

2. *Función sigmoidal.* Esta es la función más comúnmente utilizada en redes neuronales artificiales. Es estrictamente creciente con un comportamiento asintótico. Un ejemplo de la función sigmoidal es la función logística:

$$\varphi(v) = \frac{1}{1+e^{-ax}} \tag{2.11}$$

donde *a* es el parámetro que determina la pendiente de la función sigmoidal (figura 2.4b).

Las funciones de activación hasta aquí descritas toman valores en el intervalo cerrado [0,1]. Sin embargo, también pueden permitir que éstas tomen valores en el intervalo cerrado [-1,1]; en ese caso la función escalón queda definida como:

$$(\mathrm{v}) = \begin{cases} 1 \text{ si } \mathrm{v} > 0 \\ 0 \text{ si } \mathrm{v} = 0 \\ -1 \text{ si } \mathrm{v} < 0 \end{cases} \tag{2.12}$$

la cual es comúnmente llamada *función signo*. Para la forma correspondiente de la función sigmoidal se puede usar la función tangente hiperbólica, definida como:

$$\boldsymbol{\varphi}(\mathrm{v}) = \tanh(\mathrm{v}) \tag{2.13}$$

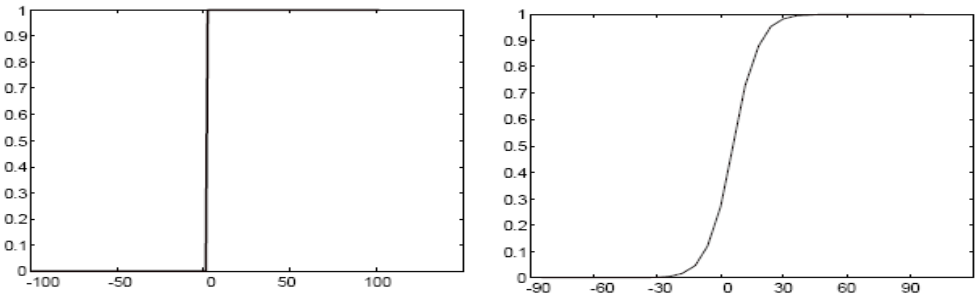


Figura 2.4.Funciones de activación: (a) Función escalón. (b) Función sigmoidal.

2.5. Arquitecturas neuronales

En general, es posible identificar tres clases de arquitecturas neuronales. Éstas se describen a continuación:

2.5.1. Redes neuronales unicapa

Generalmente, en las redes neuronales, las neuronas están organizadas por capas. En el caso más simple, una red neuronal unicapa, es donde la capa de entrada se conecta directamente a la capa de neuronas de la salida por medio de la sinapsis. Esto se ilustra en la (figura 2.15a) para el caso de tres nodos, tanto en la capa de entrada como en la capa de salida. Se le da la designación de unicapa, porque sólo tiene una capa con nodos

computacionales; para esta designación no se toma en cuenta la capa que contiene los nodos de entrada.

2.5.2. Redes neuronales multicapa

Las redes neuronales multicapa se distinguen por tener una o más capas de neuronas ocultas, cuyos nodos computacionales se denominan neuronas ocultas o unidades ocultas. La red neuronal mostrada en la (figura 2.15b) se dice totalmente conectada, en el sentido de que todos los nodos en cada capa de la red están conectados a todos los otros nodos en la siguiente capa. En el caso de que falte alguna de las sinapsis de la red, entonces se dice que la red está parcialmente conectada.

2.5.3. Redes neuronales recurrentes

Las redes neuronales recurrentes se distinguen de las anteriores en que por lo menos tienen un lazo de retroalimentación. Por ejemplo, una red recurrente puede consistir en una sola capa oculta con cada una de sus neuronas retroalimentando las señales de salida a la estrada de otras neuronas, como se ilustra en la (figura 2.15c), donde se introduce el operador de retardo unitario q^{-1} en el lazo de retroalimentación. Se observa que la red neuronal representada en la (figura 2.15c) no tiene neuronas ocultas. En la (figura 2.15d) se ilustra otra clase de red neuronal recurrente con neuronas ocultas. La presencia de una estructura recurrente tiene un profundo impacto en la capacidad de aprendizaje y de representación de la red neuronal.

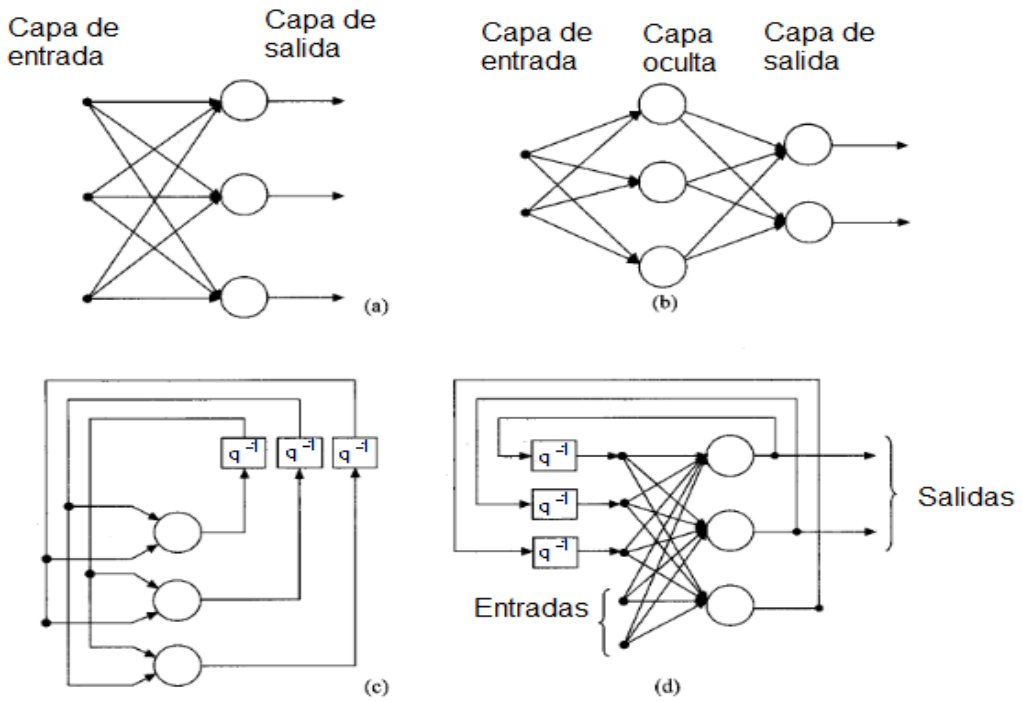


Figura 2.5.Estructuras neuronales: (a) Red neuronal unicapa. (b) Red neuronal multicapa. (c) Red neuronal recurrente con lazos de retroalimentación. (d) Red neuronal recurrente con neuronas ocultas

2.6. Algoritmo de retropropagación

La idea del algoritmo de retropropagación de error en las redes en neuronales fue utilizada por primera vez por Werbos en 1974 en su tesis doctoral (Werbos, 1974).

El proceso de retropopagación consiste en dos recorridos a través de las diferentes capas de la red: el recorrido hacia delante y hacia atrás. En el recorrido hacia delante, el vector de entrada es presentado a los nodos de la capa de entrada y el efecto se propaga a todas las demás neuronas de la red, capa por capa. Este hecho llega hasta las neuronas de salida, donde se constituye la respuesta de la red al patrón de entrada. En el segundo recorrido, los pesos sinápticos son ajustados conforme la regla de corrección de error. El recorrido hacia atrás comienza cuando se calcula el error, es decir, la respuesta de la red menos la respuesta deseada. Una vez calculado el error, éste es propagado hacia atrás, en contra de la dirección de las conexiones sinápticas. De aquí el término “retropropagación”.

Entonces, los pesos sinápticos de la red son ajustados, capa por capa, con la finalidad de que el error sea cada vez menor por cada patrón de entrada que es presentado a la red, obsérvese la (figura 2.18). El algoritmo de retropopagación se puede describir en (Haykin, 1999).

2.7. Robots manipuladores

Mecánicamente, se puede decir que un robot manipulador es un brazo mecánico articulado formado por eslabones conectados a través de uniones o articulaciones (Kelly y Santibáñez, 2003) en donde cada eslabón es un cuerpo rígido. Los robots se dividen principalmente en dos categorías: robots móviles y robots manipuladores, como se muestra en la (figura 2.6).

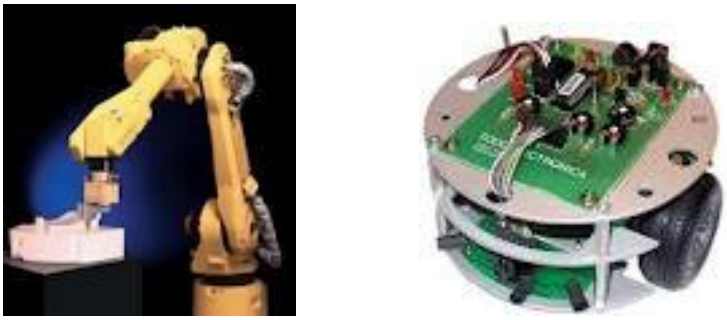


Figura 2.6.Robot manipulador y robot móvil

Un manipulador puede realizar diferentes movimientos en cada articulación como son: traslación, rotación o una combinación de los anteriores. El grado de libertad (g.d.l) de un brazo articulado está determinado por su número de articulaciones.

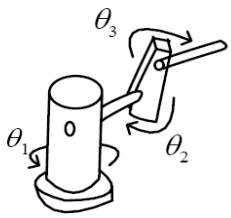


Figura 2.7.Grados de libertad

En la figura (2.7) se muestra las variables $\theta_1, \theta_2, \theta_3$, donde cada una se refiere a las posiciones articulares del robot y representan, además, los grados de libertad del mismo.

Las posiciones articulares se miden por medio de sensores colocados en los accionadores localizados habitualmente justo en las uniones articulares, éstos se agrupan para propósitos de análisis en el "vector de posiciones articulares q". Por lo tanto, para un brazo mecánico con n articulaciones, es decir, n g.d.l, el vector de posiciones articulares q ecuación (2.14) tendrá n elementos:

$$\mathbf{q} = \begin{bmatrix} \mathbf{q_1} \\ \mathbf{q_2} \\ \vdots \\ \mathbf{q_n} \end{bmatrix} \tag{2.14}$$

También resulta práctico escribir el vector de velocidades de las articulaciones en forma vectorial. Estas velocidades se miden a través de sensores que se colocan en el robot.

El vector se puede escribir como en la ecuación (2.15).

$$\dot{\mathrm{q}} = \begin{bmatrix} \dot{\mathrm{q}}_1 \\ \dot{\mathrm{q}}_2 \\ \vdots \\ \dot{\mathrm{q}}_\mathrm{n} \end{bmatrix} \tag{2.15}$$

Cada una de las articulaciones del manipulador cuenta con un actuador, que puede ser de tipo neumático, hidráulico o electromecánico. Estos accionadores generan la fuerza que produce el movimiento de los eslabones y, por consiguiente, el movimiento del manipulador.

Ese conjunto de fuerzas generado por los accionadores del robot también se puede agrupar en un vector llamado τ. El vector se muestra en la ecuación (2.16).

$$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \vdots \\ \tau_n \end{bmatrix} \tag{2.16}$$

Por otro lado, también resulta conveniente determinar la posición y orientación del efector final del robot, puesto que este dispositivo realiza la tarea encomendada al robot, tal como abrir o cerrar una pinza, aplicar soldadura, etc. Dicha posición y orientación se expresa en términos del marco de referencia coordenado cartesiano (x, y, z) colocado en la base del robot, así como eventualmente términos llamados ángulos de Euler. Por lo tanto, estas coordenadas se pueden expresar en forma vectorial x como se muestra en la ecuación (2.17).

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{2.17}$$

2.7.1. Sistema mecánico

El sistema mecánico está compuesto por diversas articulaciones. Normalmente se distingue entre el brazo y el efector final.

El aumento del número de articulaciones aporta mayor maniobrabilidad pero dificulta el problema de control, obteniéndose normalmente menores precisiones por acumulación de errores. Varios manipuladores industriales tienen menos de los seis grados de libertad de rotación o traslación que se requieren en general para realizar una tarea específica.

En este punto conviene indicar que las ecuaciones que describen el movimiento del brazo articulado son ecuaciones diferenciales no lineales y acopladas, para las que, en un caso general, resulta difícil obtener soluciones analíticas. Físicamente, los términos de acoplamiento representan: pares gravitacionales que dependen de las posiciones articulares, fuerzas centrifugas y de Coriolis, pares de reacción debido a las aceleraciones de otras articulaciones. Las magnitudes de éstas dependen de las características del robot y su carga.

2.7.2. Actuadores

Los actuadores generan las fuerzas o pares necesarios para levantar la estructura mecánica. Se utilizan dispositivos hidráulicos para desarrollar potencias importantes y neumáticas, pero en la actualidad se ha extendido el uso de motores eléctricos y, en particular, motores de corriente continua, empleándose en algunos casos motores paso a paso y otros actuadores electromecánicos sin escobillas. Existen también robots industriales de accionamiento directo que permiten eliminar los problemas mecánicos inherentes al empleo de engranajes y otras transmisiones, como se muestra en la (figura 2.8).

a) b)

Figura 2.8.a) Motores de corriente continua. b) Motor paso a paso

2.7.3. Sensores y sistemas de control

Los sistemas de control de robots pueden considerarse funcionalmente en jerarquías. En el nivel inferior se realizan tareas de servocontrol y supervisión de las articulaciones. La mayor parte de los robots industriales actuales emplean servomecanismos convencionales con retroalimentación de posición y velocidad para generar señales de control sobre los actuadores de las articulaciones. Típicamente, los parámetros del controlador son fijos aunque varíen significativamente las condiciones de trabajo con la carga o con el propio movimiento.

Se puede decir que las cargas inerciales, acoplamiento entre las articulaciones, y efectos de gravedad son todos dependientes de la posición. El problema se amplía al aumentar la velocidad. Como resultado, en la gran cantidad de robots industriales actuales, la velocidad de operación deber ser pequeña.

En el segundo nivel de control se ocupa la generación de trayectorias, entendiendo por tal la evolución del efector final cuando se desplaza de una posición a otra. El generador de trayectorias debe suministrar a los actuadores las referencias apropiadas para conseguir el recorrido deseado del efector final a partir de la especificación del movimiento deseado en el espacio de la tarea.

Los niveles superiores se ocupan de la comunicación con el usuario, de la interpretación de los programas, de la percepción sensorial y de la planificación.

2.7.4. Dinámica de los robots manipuladores

Se debe entender por dinámica a las ecuaciones de movimiento del manipulador, es decir, la manera en que el manipulador se mueve por consecuencia de fuerzas externas aplicadas a cada eslabón. La dinámica está constituida por un conjunto de ecuaciones diferenciales ordinarias de segundo orden no lineales, que dependen de las propiedades cinemáticas e inerciales del robot. La dinámica del manipulador presenta dos enfoques: el primero, considera un punto dentro de una trayectoria dada determinado por $q, \dot{q}, \ddot{q}$ y se desea encontrar el vector de fuerza necesario para satisfacer la ecuación dinámica; el segundo, calcula los movimientos del manipulador $q, \dot{q}, \ddot{q}$ como resultado de un vector de fuerza dado. Este enfoque es útil para simular al robot.

Los métodos de obtención del modelo dinámico de un brazo articulado emplean formulaciones tales como la de Euler-Lagrange o la de Newton-Euler. En esta tesis sólo se tratará con cierto detalle la primera formulación.

2.7.5. Método de Euler-Lagrange.

Las ecuaciones dinámicas de un robot manipulador pueden obtenerse a partir de las ecuaciones de movimiento de Lagrange, debido a que fue el primero que las dió a conocer en 1788.

Si se considera el robot manipulador que se compone de n eslabones mostrado en la figura (2.9), la energía total E de un mecanismo de n g.d.l es la suma de sus energías cinética K y potencial U.

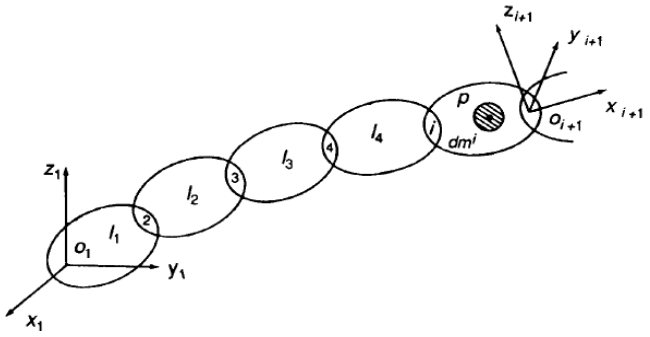


Figura 2.9.Esquema de un robot típico

$$E\big(q(t),\dot{q}(t)\big) = K\big(q(t),\dot{q}(t)\big) + U(q(t)) \tag{2.18}$$

donde $q(t) = [q_1(t), \dots q_n(t)]^T$

El *lagrangiano* $L(q,\dot{q})$ de un brazo manipulador de n g.d.l es la diferencia entre su energía cinética K y su energía potencial U:

$$\mathrm{L(q(t),\dot{q}(t)) = K(q(t),\dot{q}(t)) - U(q(t))} \tag{2.19}$$

Se considera que la energía potencial U se debe a fuerzas conservativas como la fuerza de gravedad y las fuerzas de resortes.

Las ecuaciones de movimiento de Lagrange para un manipulador de n grados de libertad, viene dada por:

$$\frac{\mathrm{d}}{\mathrm{dt}}\left[\frac{\partial \mathrm{L(q,\dot{q})}}{\partial \dot{\mathrm{q}}_\mathrm{i}}\right] - \frac{\partial \mathrm{L(q,\dot{q})}}{\partial \mathrm{q_i}} = \tau_\mathrm{i}, \quad \mathrm{i} = 1, \dots \mathrm{n} \tag{2.20}$$

donde τ_i son las fuerzas o pares ejercidos externamente por actuadores en cada articulación así como fuerzas no conservativas. Como fuerzas no conservativas se incluyen las de

fricción, las de resistencia al movimiento de un objeto dentro de un fluido y en general las que dependen del tiempo o de la velocidad.

Se puede notar que se tendrán tantas ecuaciones escalares dinámicas como g.d.l tenga el robot.

El uso de las ecuaciones de Lagrange para el modelado dinámico de manipuladores se puede formular en cuatro pasos:

1. Cálculo de la energía cinética: $K\big(q(t), \dot{q}(t)\big)$.
2. Cálculo de la energía potencial: $U\big(q(t)\big)$.
3. Cálculo del lagrangiano: $L(q(t), \dot{q}(t))$.
4. Desarrollo de las ecuaciones de Lagrange.

2.8. Estado del arte

Hasta el día de hoy se observa un constante incremento en el uso de controladores para identificar modelos suficientemente fiables de la dinámica del robot y métodos de control de articulaciones que permiten compensar las no linealidades, acoplamientos y optimizar el comportamiento dinámico

La aplicación de métodos conexionistas al área de identificación y control de sistemas dinámicos puede considerarse como un paso natural en la teoría de control puesto que supone un intento de aportar nuevas soluciones a las tres grandes necesidades históricas de la Automática (Antsaklis, 1990):

1. Trabajar con sistemas cada vez más complejos.
2. Satisfacer requerimientos de diseño cada vez más exigentes.
3. Cumplir el punto anterior de forma adecuada con un conocimiento anticipado de la planta y de su entorno cada vez menos preciso.

Esto se fundamenta en dos de las capacidades más importantes de las técnicas neuronales: la capacidad de aprender típicamente basada en la minimización de una función de energía adecuada (Narendra y Parthasarathy, 1990, 1991) y el buen comportamiento en la aproximación de funciones lineales y no lineales (Cybenko, 1989; Funahashi, 1989; Chen y Chen, 1993). Estas características se ven apoyadas en otras, como por ejemplo: el paralelismo masivo y la tolerancia a fallos, que podrán aprovecharse de forma completa cuando existan realizaciones analógicas de hardware de uso comercial, como el trabajo de (DeWeerth *et al*., 1991).

Si se considera que el área de aplicación de técnicas neuronales al control de sistemas está subdividida en la identificación y control; pueden citarse, entre otros, los trabajos que aparecen a continuación.

Identificación: La mayor parte de los trabajos en identificación de sistemas mediante redes neuronales artificiales están basados en modelos del tipo red neuronal retroalimentación hacia adelante (*feedforward)* con aprendizaje por retropropagación del error *(backpropagation),* ó variaciones más eficientes de este algoritmo. Estos métodos han sido aplicados a procesos reales y han mostrado un comportamiento correcto en las tareas de identificación (Funahashi, 1989; Bhat *et al*., 1990; Weerasooriya y El-Sharkawi, 1991; Chen y Chen, 1993; Ruiz y Torras, 1995). Es importante hacer notar que la mayor parte de ellos usan modelos estáticos de tiempo discreto que capturan la dinámica del proceso real mediante el uso de líneas de retardos en las entradas y salidas del modelo (Narendra y Parthasarathy, 1990). No obstante, en la identificación de sistemas dinámicos complejos pueden aparecer ciertos inconvenientes asociados a este tipo de modelos como, por ejemplo, dificultades para seleccionar el número necesario de retardos y, en algunos casos, predicciones pobres en operación en línea después de haberse efectuado el aprendizaje fuera de línea (Pearlmutter, 1995). Para evitar estas limitaciones se han empleado redes neuronales recurrentes con dinámica interna (Qin *et al*., 1992), pero una característica común de estos trabajos es que trabajan en tiempo discreto dando lugar a modelos de tiempo discreto del sistema real de tiempo continuo. Este hecho da lugar a una gran dependencia del período de muestreo usado en el procedimiento y no aporta información

sobre las trayectorias del modelo en los instantes de tiempo entre dos muestras consecutivas. Además, el soporte teórico para el posterior desarrollo de controladores para estos modelos es actualmente pobre.

Control: Esta es el área que ha despertado mayor interés y que registra el mayor número de publicaciones. Los primeros trabajos planteaban arquitecturas de control con redes multicapa *feedforward* con aprendizaje supervisado usando *backpropagation,* sirvan de ejemplo los trabajos (Psaltis *et al*., 1988; Nguyen y Widrow, 1990). No obstante, estos primeros trabajos eran propuestas teóricas o aplicaciones de carácter académico. En los trabajos anteriores y en otros, se requiere un entrenamiento previo de la red, que resulta un impedimento práctico para la realización de las soluciones propuestas.

Por otro lado en (Cui y Shin, 1993) se propone un esquema de control directo basado en una red que modela la dinámica inversa del sistema. Se demuestra que para la mayoría de los casos prácticos, la red se puede entrenar directamente con el error de control en lugar del error de salida de la red como se realiza habitualmente.

Conforme ha ido madurando el tema, se ha perfilado una relación de ideas entre el control adaptable y control neuronal, que ha llevado a plantear arquitecturas neuronales similares a las estándar. Algunos resultados de esta relación son los trabajos: (Narendra y Annaswamy, 1987; Psaltis *et al*., 1988 y Lewis *et al*., 1995), las principales desventajas de estos métodos residen en el requerimiento de linealidad en los parámetros desconocidos del sistema y la necesidad de calcular la matriz de regresión con funciones no lineales conocidas del manipulador, y es necesario destacar, también, los trabajos que hacen uso de arquitecturas conexionistas recurrentes (Guez *et al*., 1988).

El uso del perceptrón multicapa (MLP) con el algoritmo de retropopagación (BP) desató un disparo en el desarrollo de una multitud de aplicaciones en el control de sistemas. Este tipo de redes se caracterizan porque todos sus nodos están involucrados en el proceso de cualquier patrón de entradas. Algunos investigadores como (Kim y Lewis, 1999)

resaltaron los conocidos problemas relacionados con el MLP entrenado con BP tales como: la lentitud de convergencia del aprendizaje y las fallas en el mismo o la no convergencia.

Una red neuronal con (BP) fue el primer enfoque para control adaptable, el cual incorpora una capacidad de aprendizaje de acuerdo con (Lewis *et al*., 1996). La red neuronal fue usada en un lazo de retroalimentación con un controlador convencional PD para el manejo de un robot manipulador. La dinámica inversa del mecanismo fue considerada para ser obtenida por el aprendizaje de la red neuronal.

III. METODOLOGÍA

El desarrollo del controlador neuronal comienza con un entendimiento de la naturaleza física del sistema a controlar; en este caso se trata de un robot de eslabones rígidos con uniones rotativas de dos grados de libertad. Por lo tanto, se debe empezar con la obtención y simulación del modelo dinámico, enseguida la elección de la arquitectura neuronal con su algoritmo de aprendizaje para culminar con la evaluación del sistema completo (robot-controlador).

3.1. Modelado dinámico del brazo articulado

En esta sección se encuentra el modelo matemático de un robot rígido con uniones rotativas de dos grados de libertad, mostrado en la (figura 3.1), para ello se emplea el método de Euler-Lagrange. El brazo manipulador está formado por dos eslabones rígidos de longitudes l_1 y l_2, y masas m_1 y m_2, respectivamente. Las uniones 1 y 2 son rotacionales. Los desplazamientos del robot se llevan a cabo en el plano vertical x-y. La distancia entre los ejes de giro y los centros de masa se describen por l_{c1} y l_{c2}. Finalmente, I_1 e I_2 representan los momentos de inercia de los eslabones con respecto al eje que pasa a través de sus centros de masa y que es perpendicular al plano X-Y.

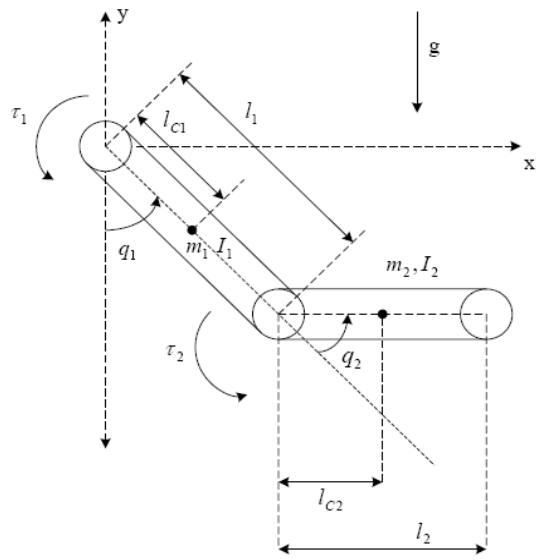


Figura 3.1.Robot rígido de dos grados de libertad

De acuerdo al método de Euler-Lagrange las ecuaciones de movimiento que representan a un robot rígido de *n* grados de libertad están determinadas por la ecuación (3.1):

$$\frac{\mathrm{d}}{\mathrm{dt}}\left[\frac{\partial \mathrm{L}(\mathrm{q},\dot{\mathrm{q}})}{\partial \dot{\mathrm{q}}_\mathrm{i}}\right] - \frac{\partial \mathrm{L}(\mathrm{q},\dot{\mathrm{q}})}{\partial \mathrm{q}_\mathrm{i}} = \tau_\mathrm{i}, \quad \mathrm{i} = 1, \ldots \mathrm{n} \tag{3.1}$$

Siendo $q(t) = [q_1(t), \ldots q_n(t)]^T$, $\dot{q}(t) = [\dot{q}_1(t), \ldots \dot{q}_n(t)]^T$, la posición y velocidad angular de los eslabones respectivamente, $\tau = [\tau_1, \ldots \tau_n(t)]^T$ es el vector de pares o torques aplicados al robot.

En el caso de dos grados de libertad, la energía cinética y potencial de los eslabones están determinadas por las ecuaciones (3.2) y (3.3) respectivamente;

$$\mathrm{K}(\mathrm{q},\dot{\mathrm{q}}) = \mathrm{K}_1(\mathrm{q},\dot{\mathrm{q}}) + \mathrm{K}_2(\mathrm{q},\dot{\mathrm{q}}) \tag{3.2}$$

y

$$\mathrm{U}(\mathrm{q}) = \mathrm{U}_1(\mathrm{q}) + \mathrm{U}_2(\mathrm{q}) \tag{3.3}$$

Por lo tanto, para obtener el modelo matemático de un robot de 2 grados de libertad se parte de la ecuación (3.1). Para encontrar la energía cinética asociada a los eslabones 1 y 2 se consideran los centros de masas de cada eslabón. De acuerdo a la ecuación (3.4), la energía cinética asociada al sistema está determinada por:

$$\mathrm{K}(\mathrm{q},\dot{\mathrm{q}}) = \mathrm{K}_1(\mathrm{q},\dot{\mathrm{q}}) + \mathrm{K}_2(\mathrm{q},\dot{\mathrm{q}}) = \left[\frac{1}{2}\mathrm{m}_1\mathrm{v}_1^2 + \frac{1}{2}\mathrm{I}_1\dot{\mathrm{q}}_1^2\right] + \left[\frac{1}{2}\mathrm{m}_2\mathrm{v}_2^2 + \frac{1}{2}\mathrm{I}_2[\dot{\mathrm{q}}_1 + \dot{\mathrm{q}}_2]^2\right] \tag{3.4}$$

donde v_1 y v_2 representan la velocidad del centro de masa de los eslabones, cuya forma en términos de las coordenadas cartesianas es $v_1 = [\dot{x}_1\ \dot{y}_1]^T$ y $v_2 = [\dot{x}_2\ \dot{y}_2]^T$. Además, las coordenadas del centro de masa del eslabón 1 expresadas en el plano X-Y (figura 3.2) son:

$$\mathrm{x}_1 = \mathrm{l}_{c1}\mathrm{sen}(\mathrm{q}_1) \tag{3.5}$$

$$\mathrm{y}_1 = -\mathrm{l}_{c1}\cos(\mathrm{q}_1) \tag{3.6}$$

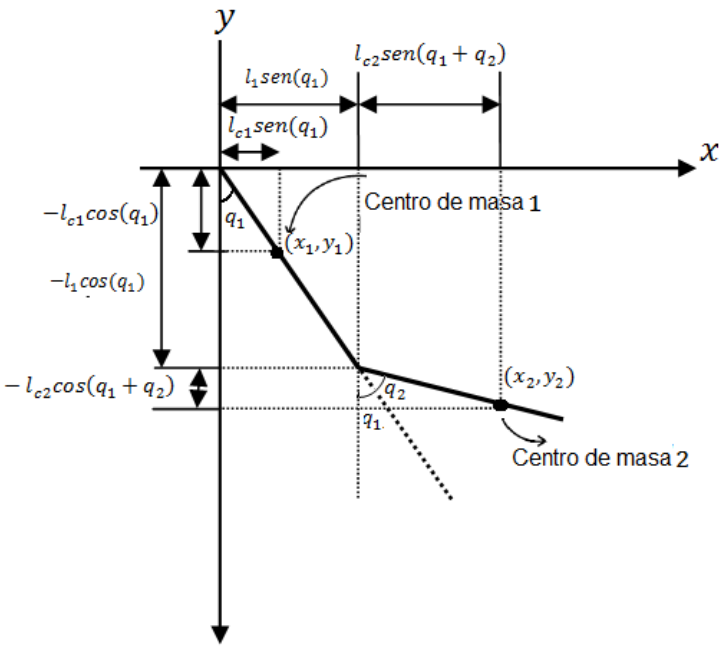


Figura 3.2.Análisis geométrico del manipulador de 2 grados de libertad

Por otro lado, las coordenadas para el centro de masa del eslabón 2 en el plano X-Y se pueden escribir como las siguientes ecuaciones (3.7) y (3.8):

$$x_2 = l_1 sen(q_1) + l_{c2} sen(q_1 + q_2) \tag{3.7}$$

$$y_2 = -l_1 cos(q_1) - l_{c2} cos(q_1 + q_2) \tag{3.8}$$

Por lo tanto, los vectores de velocidad de dichos eslabones son (3.9) y (3.10):

$$v_1 = \begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \end{bmatrix} = \begin{bmatrix} l_{c1} cos(q_1)\dot{q}_1 \\ l_{c1} sen(q_1)\dot{q}_1 \end{bmatrix} \tag{3.9}$$

$$v_2 = \begin{bmatrix} \dot{x}_2 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} l_1 cos(q_1)\dot{q}_1 + l_{c2} cos(q_1 + q_2)(\dot{q}_1 + \dot{q}_2) \\ l_1 sen(q_1)\dot{q}_1 + l_{c2} sen(q_1 + q_2)(\dot{q}_1 + \dot{q}_2) \end{bmatrix} \tag{3.10}$$

Por consiguiente, la velocidad al cuadrado de eslabón 1 resulta ser la ecuación (3.11):

$$v_1^2 = {v_1}^T v_1 = l_{c1}^2 \dot{q}_1^2 \tag{3.11}$$

Empleando las identidades trigonométricas $cos^2(\theta) + sen^2(\theta) = 1$ y $sen(q_1)sen(q_1 + q_2) + cos(q_1)cos(q_1 + q_2) = \cos(q_2)$ tiene la rapidez al cuadrado ${v_2}^T v_2$ del centro de masa del eslabón 2, ecuación (3.12):

$$v_2^2 = {v_2}^T v_2 = l_1^2 \dot{q}_1^2 + l_{c2}^2 [\dot{q}_1^2 + 2\dot{q}_1 \dot{q}_2 + \dot{q}_2^2] + 2 l_1 l_{c2} [\dot{q}_1^2 + \dot{q}_1 \dot{q}_2] \cos(q_2) \tag{3.12}$$

De esta manera las energías cinéticas correspondientes se muestran en las ecuaciones (3.13) y (3.14), (Kelly y Santibáñez, 2003):

$$K_1(q, \dot{q}) = \left[\frac{1}{2} m_1 v_1^2 + \frac{1}{2} I_1 \dot{q}_1^2\right] = \left[\frac{1}{2} m_1 l_{c1}^2 \dot{q}_1^2 + \frac{1}{2} I_1 \dot{q}_1^2\right] \tag{3.13}$$

$$\begin{aligned} K_2(q, \dot{q}) &= \left[\frac{1}{2} m_2 v_2^2 + \frac{1}{2} I_2 [\dot{q}_1 + \dot{q}_2]^2\right] \\ &= \left[\frac{1}{2} m_2 (l_1^2 \dot{q}_1^2 + l_{c2}^2 [\dot{q}_1^2 + 2\dot{q}_1 \dot{q}_2 + \dot{q}_2^2] + 2 l_1 l_{c2} [\dot{q}_1^2 + \dot{q}_1 \dot{q}_2] \cos(q_2)) \right. \\ &\quad \left. + \frac{1}{2} I_2 [\dot{q}_1 + \dot{q}_2]^2\right] \\ &= \frac{1}{2} m_2 l_1^2 \dot{q}_1^2 + \frac{1}{2} m_2 l_{c2}^2 [\dot{q}_1^2 + 2\dot{q}_1 \dot{q}_2 + \dot{q}_2^2] + m_2 l_1 l_{c2} [\dot{q}_1^2 + \dot{q}_1 \dot{q}_2] \cos(q_2) + \\ &\quad \frac{1}{2} I_2 [\dot{q}_1 + \dot{q}_2]^2 \end{aligned} \tag{3.14}$$

Por otro lado, la energía potencial asociada a los eslabones de la figura (3.1), de acuerdo a (3.3), están determinadas por la ecuación (3.15) y (3.16):

$$U_1(q) = -m_1 l_{c1} g \cos(q_1) \tag{3.15}$$

y

$$U_2(q) = -m_2 l_1 g \cos(q_1) - m_2 l_{c2} g \cos(q_1 + q_2) \tag{3.16}$$

A partir de las ecuaciones (3.13-3.16) puede obtenerse el lagrangiano, ecuación (3.17):

$$L(q,\dot{q}) = K(q,\dot{q}) - U(q) = K_1(q,\dot{q}) + K_2(q,\dot{q}) - U_1(q) - U_2(q) \tag{3.17}$$

$$L(q,\dot{q}) = \frac{1}{2}m_1 l_{c1}^2 \dot{q}_1^2 + \frac{1}{2}I_1 \dot{q}_1^2 + \frac{1}{2}m_2 l_1^2 \dot{q}_1^2 + \frac{1}{2}m_2 l_{c2}^2 [\dot{q}_1^2 + 2\dot{q}_1\dot{q}_2 + \dot{q}_2^2] + m_2 l_1 l_{c2} [\dot{q}_1^2 + \dot{q}_1\dot{q}_2] \cos(q_2) + \frac{1}{2}I_2 [\dot{q}_1 + \dot{q}_2]^2 + m_1 l_{c1} g \cos(q_1) + m_2 l_1 g \cos(q_1) + m_2 l_{c2} g \cos(q_1 + q_2) \tag{3.18}$$

La ecuación (3.18) se puede organizar como la ecuación (3.19):

$$L(q,\dot{q}) = \frac{1}{2}[m_1 l_{c1}^2 + m_2 l_1^2]\dot{q}_1^2 + \frac{1}{2}m_2 l_{c2}^2 [\dot{q}_1^2 + 2\dot{q}_1\dot{q}_2 + \dot{q}_2^2] + m_2 l_1 l_{c2} [\dot{q}_1^2 + \dot{q}_1\dot{q}_2] \cos(q_2) + m_1 l_{c1} g \cos(q_1) + m_2 l_1 g \cos(q_1) + m_2 l_{c2} g \cos(q_1 + q_2) + \frac{1}{2}I_2 [\dot{q}_1 + \dot{q}_2]^2 + \frac{1}{2}I_1 \dot{q}_1^2 \tag{3.19}$$

De esta última ecuación es posible obtener las siguientes ecuaciones, que son iguales a las obtenidas en (Kelly y Santibáñez, 2003):

$$\frac{\partial L(q,\dot{q})}{\partial \dot{q}_1} = [m_1 l_{c1}^2 + m_2 l_1^2]\dot{q}_1 + m_2 l_{c2}^2 \dot{q}_1 + m_2 l_{c2}^2 \dot{q}_2 + 2m_2 l_1 l_{c2} \cos(q_2)\dot{q}_1 + m_2 l_1 l_{c2} \cos(q_2)\dot{q}_2 + I_2[\dot{q}_1 + \dot{q}_2] + I_1 \dot{q}_1 \tag{3.20}$$

$$\frac{d}{dt}\left[\frac{\partial L(q,\dot{q})}{\partial \dot{q}_1}\right] = [m_1 l_{c1}^2 + m_2 l_1^2 + m_2 l_{c2}^2 + 2m_2 l_1 l_{c2} \cos(q_2)]\ddot{q}_1 + [m_2 l_{c2}^2 + m_2 l_1 l_{c2} \cos(q_2)]\ddot{q}_2 - 2m_2 l_1 l_{c2} \operatorname{sen}(q_2)\dot{q}_1\dot{q}_2 - m_2 l_1 l_{c2} \operatorname{sen}(q_2)\dot{q}_2^2 + I_2[\ddot{q}_1 + \ddot{q}_2] + I_1 \ddot{q}_1 \tag{3.21}$$

$$\frac{\partial L(q,\dot{q})}{\partial q_1} = -[m_1 l_{c1} + m_2 l_1] g \operatorname{sen}(q_1) - m_2 l_{c2} g \operatorname{sen}(q_1 + q_2) \tag{3.22}$$

Mientras que para el eslabón 2 se obtienen las ecuaciones (3.23), (3.24) y (3.25):

$$\frac{\partial L(q,\dot{q})}{\partial \dot{q}_2} = m_2 l_{c2}^2 \dot{q}_1 + m_2 l_{c2}^2 \dot{q}_2 + m_2 l_1 l_{c2} \cos(q_2)\dot{q}_1 + I_2[\dot{q}_1 + \dot{q}_2] \tag{3.23}$$

$$\frac{d}{dt}\left[\frac{\partial L(q,\dot{q})}{\partial \dot{q}_2}\right] = m_2 l_{c2}^2 \ddot{q}_1 + m_2 l_{c2}^2 \ddot{q}_2 + m_2 l_1 l_{c2} \cos(q_2)\ddot{q}_1 - m_2 l_1 l_{c2} \text{sen}(q_2)\dot{q}_1\dot{q}_2 + I_2[\ddot{q}_1 + \ddot{q}_2] \tag{3.24}$$

$$\frac{\partial L(q,\dot{q})}{\partial q_2} = -m_2 l_1 l_{c2} \text{sen}(q_2)[\dot{q}_1^2 + \dot{q}_1\dot{q}_2] - m_2 l_{c2} g \text{sen}(q_1 + q_2) \tag{3.25}$$

Finalmente, las ecuaciones de movimiento para el brazo se obtienen aplicando la ecuación (3.1), por lo tanto se obtienen las ecuaciones (3.26) y (3.27):

$$\tau_1 = [m_1 l_{c1}^2 + m_2 l_1^2 + m_2 l_{c2}^2 + 2m_2 l_1 l_{c2} \cos(q_2) + I_1 + I_2]\ddot{q}_1 + [m_2 l_{c2}^2 + m_2 l_1 l_{c2} \cos(q_2) + I_2]\ddot{q}_2 - 2m_2 l_1 l_{c2} \text{sen}(q_2)\dot{q}_1\dot{q}_2 - m_2 l_1 l_{c2} \text{sen}(q_2)\dot{q}_2^2 + [m_1 l_{c1} + m_2 l_1] g \text{sen}(q_1) + m_2 l_{c2} g \text{sen}(q_1 + q_2) \tag{3.26}$$

$$\tau_2 = [m_2 l_{c2}^2 + m_2 l_1 l_{c2} \cos(q_2) + I_2]\ddot{q}_1 + [m_2 l_{c2}^2 + I_2]\ddot{q}_2 + m_2 l_1 l_{c2} \text{sen}(q_2)\dot{q}_1^2 + m_2 l_{c2} g \text{sen}(q_1 + q_2) \tag{3.27}$$

Siendo τ_1 y τ_2, los pares que actúan en las uniones 1 y 2.

Las ecuaciones (3.26) y (3.27) se pueden escribir en forma general como la ecuación (3.28), (Kelly y Santibáñez, 2003):

$$\boldsymbol{\tau} = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) \tag{3.28}$$

$$\begin{bmatrix}\tau_1 \\ \tau_2\end{bmatrix} = \begin{bmatrix}M_{11}(q) & M_{12}(q) \\ M_{21}(q) & M_{22}(q)\end{bmatrix}\begin{bmatrix}\ddot{q}_1 \\ \ddot{q}_2\end{bmatrix} + \begin{bmatrix}C_{11}(q,\dot{q}) & C_{12}(q,\dot{q}) \\ C_{21}(q,\dot{q}) & C_{22}(q,\dot{q})\end{bmatrix}\begin{bmatrix}\dot{q}_1 \\ \dot{q}_2\end{bmatrix} + \begin{bmatrix}g_1(q) \\ g_2(q)\end{bmatrix} \tag{3.29}$$

donde:

$$M_{11}(q) = m_1 l_{c1}^2 + m_2\left[l_1^2 + l_{c2}^2 + 2l_1 l_{c2}\cos(q_2)\right] + I_1 + I_2$$

$$M_{12}(q) = m_2\left[l_{c2}^2 + l_1 l_{c2}\cos(q_2)\right] + I_2$$

$$M_{21}(q) = m_2\left[l_{c2}^2 + l_1 l_{c2}\cos\left(q_2\right)\right] + I_2$$
$$M_{22}(q) = m_2 l_{c2}^2 + I_2$$
$$C_{11}(q,\dot{q}) = -m_2 l_1 l_{c2}\text{sen}\left(q_2\right)\dot{q}_2$$
$$C_{12}(q,\dot{q}) = -m_2 l_1 l_{c2}\text{sen}(q_2)(\dot{q}_1 + \dot{q}_2) \quad (3.30)$$
$$C_{21}(q,\dot{q}) = m_2 l_1 l_{c2}\text{sen}\left(q_2\right)\dot{q}_1$$
$$C_{22}(q,\dot{q}) = 0$$
$$g_1(q) = \left[m_1 l_{c1} + m_2 l_1\right]gsen(q_1) + m_2 l_{c2} gsen(q_1 + q_2)$$
$$g_2(q) = m_2 l_{c2} gsen(q_1 + q_2)$$

3.2. Arquitectura de la red neuronal propuesta

En esta sección se presenta la arquitectura de la red neuronal a utilizar en este trabajo. Las neuronas se disponen en forma de capas unas detrás de otras, con lo cual se adoptará la forma de una red hacia adelante, con la característica que las neuronas de la capa oculta tienen conexiones de realimentación consigo misma, como se muestra en la (figura 3.3). Este tipo de redes son llamadas redes recurrentes y son especialmente útiles en: predicción no lineal, modelado y control. Cada neurona viene descrita por las siguientes ecuaciones:

$$\mathrm{a_j(k)} = \boldsymbol{\sigma}(\mathrm{z_j(k)}) \quad (3.31)$$
$$\mathrm{z_j(k)} = \sum_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{W_{ij}^I(k)} * \mathrm{i_i(k)} + \mathrm{a_j(k-1)} * \mathrm{W_j^D(k)} \quad (3.32)$$
$$\boldsymbol{\sigma}\left(\mathrm{z_j(k)}\right) = \frac{1}{1+\exp(-\boldsymbol{\alpha} * \mathrm{z_j})} \qquad \boldsymbol{\alpha} > 0 \quad (3.33)$$
$$\mathrm{o_k(k)} = \sum_{\mathrm{j=1}}^{\mathrm{l}} \mathrm{W_{jk}^o(k)} * \mathrm{a_j(k)} \quad (3.34)$$

donde:
Los subíndices i, j, k identifican las diferentes capas de la red.

$a_j(k)$= Salida de la j-ésima neurona de la capa oculta.
$z_j(k)$ = La excitación para la j-ésima neurona de la capa oculta.

$\sigma(\cdot) =$ Es la función de activación de la j-ésima neurona.

$W_{ij}^{I} =$ Representa el peso que conecta la i-ésima neurona a la j-ésima neurona.

$W_{J}^{D} =$ Representa el peso de la j-ésima neurona de la capa oculta.

$W_{jk}^{o} =$ Representa el peso de la k-ésima neurona de la capa de salida.

$o_k =$ Salida de la k-ésima neurona de la capa de salida.

La diferencia fundamental entre el modelo propuesto y las redes neuronales estáticas está en la aparición de una realimentación en las neuronas de la capa oculta.

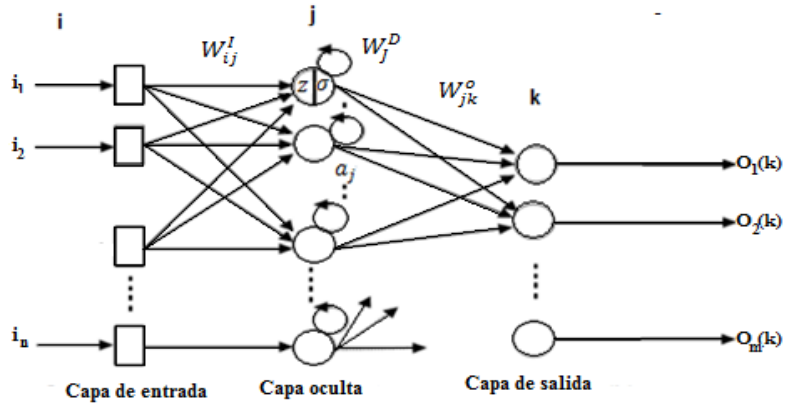


Figura 3.3.Red neuronal recurrente de tres capas

La fórmula matemática que describe la salida de la red neuronal indicada en la ecuación (3.35), se puede representar como:

$$o_k(k) = \sum_{j=1}^{l}\left[W_{jk}^{o}\boldsymbol{\sigma}\left(\sum_{i=1}^{n} W_{ij}^{I}(k)i_i(k) + a_j(k-1) * W_J^D(k)\right)\right] \quad k = 1..m \tag{3.35}$$

Las ecuaciones de la red neuronal favorablemente pueden ser expresadas en forma matricial. Por definición $i^T = [i_1, \ i_2 \ \cdots \ i_n]$ y $o^T = [o_1, \ o_2 \ \cdots \ o_m]$, respectivamente y los pesos $W_{ij}^{T} = W^T = [w_{ij}]$, $W_{jk}^{o\,T} = V^T = [v_{jk}{}^{o}]$, $W_J^{D\,T} = P = [w_j{}^{D}]$ y $a^T = diag\{a_1, \ a_2 \ \cdots \ a_l\}$, entonces se obtiene la ecuación (3.36):

$$O = V^T\sigma(W^T i + aP) \tag{3.36}$$

Por comodidad y conveniencia, las matrices de pesos sinápticos de la red neuronal están definidas como la ecuación (3.37):

$$V = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1l} \\ \vdots & \vdots & \cdots & \vdots \\ v_{m1} & v_{m2} & \cdots & v_{ml} \end{bmatrix}, W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1l} \\ \vdots & \vdots & \cdots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nl} \end{bmatrix}, P = \begin{bmatrix} w_1{}^D \\ \vdots \\ w_j{}^D \end{bmatrix} \quad (3.37)$$

Entonces, el valor de salida de cada neurona en la capa oculta está definido en la ecuación (3.38):

$$\boldsymbol{\sigma(z) = \frac{1}{1+e^{-\alpha z}} \quad \alpha > 0} \quad (3.38)$$

donde $z = [z_1 \quad z_2 \quad \cdots \quad z_l]^T$ es un vector, por lo tanto, se puede definir $\sigma(z) = [\sigma(z_1) \quad \sigma(z_2) \quad \cdots \quad \sigma(z_l)]^T$ como vector de salida de la capa oculta.

3.3. Algoritmo de aprendizaje

Dada la arquitectura establecida anteriormente, se establece un método que permita determinar los pesos sinápticos de la red que minimicen la función del error. Tal método se basará en el ajuste conveniente de los pesos sinápticos de la red. Concretamente el método empleado está basado en el gradiente descendiente, los ajustes sucesivos se aplican al vector de los pesos W_{jk}^o, W_{ij}^I y W_J^D en la dirección opuesta del gradiente descendiente. El algoritmo se basa en las ecuaciones obtenidas en (Cui y Shin, 1993), pero con la característica que se utiliza para dos neuronas de salida y con retroalimentación de las neuronas de la capa oculta.

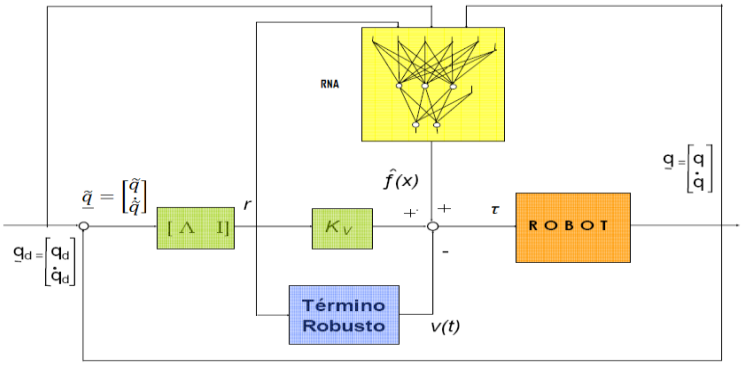


Figura 3.4.Controlador neuronal.

En la (figura 3.4) se muestra el esquema propuesto para el controlador neuronal. A continuación se muestra el algoritmo para la modificación de los pesos de la red neuronal artificial (RNA).

El primer paso consiste en definir una función de error que debe minimizarse. Dicha función de error viene definida en la ecuación (3.39).

$$\mathrm{E(k)} = \frac{1}{2}\sum_{\mathrm{p=1}}^{\mathrm{j}}(\mathrm{q_{dk}(k)} - \mathrm{q_k(k)})^2 = \frac{1}{2}\sum_{\mathrm{p=1}}^{\mathrm{j}} \mathrm{e_k^2(k)} \tag{3.39}$$

donde:

j = número de neuronas en la última capa.
$q_d =$ Posición deseada.
$q =$ Posición actual del robot.

El procedimiento de minimización consiste en moverse en dirección contraria al gradiente de la función (3.39) con respecto a los pesos W_{jk}^{o}, W_{ij}^{I} y W_{j}^{D}. El gradiente contiene las siguientes derivadas parciales.

$$\nabla \mathrm{E} = \begin{bmatrix} \frac{\partial \mathrm{E}}{\partial \mathrm{W_{jk}^{o}}} \\ \frac{\partial \mathrm{E}}{\partial \mathrm{W_{j}^{D}}} \\ \frac{\partial \mathrm{E}}{\partial \mathrm{W_{ij}^{i}}} \end{bmatrix} \tag{3.40}$$

Para actualizar los pesos, sólo es necesario calcular las derivadas parciales del error con respecto a los pesos W_{jk}^{o}, W_{j}^{D} y W_{ij}^{I}. Se obtienen, en primer lugar, las derivadas parciales del error con respecto a los pesos sinápticos de la neurona de salida W_{jk}^{o}.

$$\frac{\partial \mathrm{E(k)}}{\partial \mathrm{W_{kj}^{o}}} = \frac{\partial \mathrm{E(k)}}{\partial \mathrm{e_k(k)}} \frac{\partial \mathrm{e_k(k)}}{\partial \mathrm{e_{uk}(k)}} \frac{\partial \mathrm{e_{uk}(k)}}{\partial \mathrm{O(k)}} \frac{\partial \mathrm{O(k)}}{\partial \mathrm{W_{kj}^{o}(k)}} = \mathrm{e_k(k)} \frac{\partial \mathrm{e_k(k)}}{\partial \mathrm{e_{uk}(k)}} (-1) * \mathrm{a_j(k)} = -\mathrm{e_k(k)a_j(k)} \frac{\partial \mathrm{e_k(k)}}{\partial \mathrm{e_{uk}(k)}} \tag{3.41}$$

Siendo:

$$\frac{\partial \mathrm{E(k)}}{\partial \mathrm{e_k(k)}} = \frac{\partial}{\partial \mathrm{e_k(k)}}\left[\frac{1}{2}\sum_{\mathrm{p=1}}^{\mathrm{k}} \mathrm{e_k^2(k)}\right] = \sum_{\mathrm{p=1}}^{\mathrm{k}} \mathrm{e_k(k)} \qquad \mathrm{k} = 1 \ldots, \mathrm{m} \tag{3.42}$$

Si se considera el error de salida de la red, como se muestra en la ecuación (3.43):

$$\mathrm{e_{uk}(k)} = (\mathrm{r_k} - \mathrm{o_k}) \tag{3.43}$$

donde:

r_k = salidas deseada de la red.

o_k = salidas de la red.

Se llega a:

$$\frac{\partial \mathrm{e_{uk}(k)}}{\partial \mathrm{O(k)}} = \frac{\partial}{\partial \mathrm{O(k)}}(\mathrm{r_k} - \mathrm{o_k}) = -1 \tag{3.44}$$

Mientras que la derivada parcial de la salida de la red con respecto a los pesos de la capa de salida es la ecuación (3.45):

$$\frac{\partial \mathrm{o(k)}}{\partial \mathrm{W_{kj}^o(k)}} = \frac{\partial}{\partial \mathrm{W_{kj}^o(k)}}\left[\sum_{\mathrm{j=1}}^{\mathrm{M}} \mathrm{W_{kj}^o(k)} * \mathrm{a_j(k)}\right] = \mathrm{a_j(k)} \tag{3.45}$$

La ecuación de actualización para los pesos W_{kj}^o se puede expresar como la ecuación (3.46):

$$\mathrm{W_{kj}^o(k+1)} = \mathrm{W_{kj}^o(k)} - \boldsymbol{\eta}\frac{\partial \mathrm{E(k)}}{\partial \mathrm{W_{kj}^o}} = \mathrm{W_{kj}^o(k)} + \boldsymbol{\eta}\mathrm{e_k(k)a_j(k)}\frac{\partial \mathrm{e_k(k)}}{\partial \mathrm{e_{uk}(k)}} \tag{3.46}$$

donde $\eta > 0$ es una constante positiva llamada parámetro de velocidad de aprendizaje. Donde el único parámetro desconocido en la ecuación (3.46) es $\frac{\partial e_k(k)}{\partial e_{uk}(k)}$. En la investigación de (Cui y Shin, 1993) se muestra que la condición necesaria y suficiente para

asegurar la convergencia de los pesos sinápticos es conocer el signo, porque la magnitud puede ser incorporada en el coeficiente de aprendizaje de dicho término:

$$\text{sign}\left[\frac{\partial e_k(k)}{\partial e_{uk}(k)}\right] \tag{3.47}$$

Para sistemas no lineales, no es fácil determinar el signo de $\frac{\partial e_k(k)}{\partial e_{uk}(k)}$ en cada instante. Por lo tanto, se considera que el signo de la salida de respuesta es conocida y

$$\left|\frac{\partial e_k(k)}{\partial e_{uk}(k)}\right| < \infty \tag{3.48}$$

Al tomar esta consideración, la ecuación de actualización de los pesos W_{jk}^{o} puede ser definida como la ecuación (3.49):

$$W_{jk}^{o}(k+1) = W_{kj}^{o}(k) - \boldsymbol{\eta}\frac{\partial E(k)}{\partial W_{kj}^{o}} = W_{jk}^{o}(k) + \boldsymbol{\eta} e_k(k) a_j(k) \text{sign}\left(\frac{\partial e_k(k)}{\partial e_{uk}(k)}\right) \tag{3.49}$$

La derivada parcial de la función E con respecto a los coeficientes W_j^D se obtiene al aplicar de nuevo la regla de la cadena como se muestra en la ecuación (3.50):

$$\frac{\partial E(k)}{\partial W_j^D} = \frac{\partial E(k)}{\partial e_k(k)}\frac{\partial e_k(k)}{\partial e_{uk}(k)}\frac{\partial e_{uk}(k)}{\partial O(k)}\frac{\partial O(k)}{\partial a_j(k)}\frac{\partial a_j(k)}{\partial z_j(k)}\frac{\partial z_j(k)}{\partial W_j^D} \tag{3.50}$$

donde la ecuación (3.51) representa la derivada parcial de la salida de la red con respecto a la salida de las neuronas de la capa oculta:

$$\frac{\partial O(k)}{\partial a_j(k)} = \frac{\partial}{\partial a_j(k)}\left[\sum_{j=1}^{l} W_{kj}^{o}(k) * a_j(k)\right] = W_{kj}^{o}(k) \tag{3.51}$$

Mientras que para a_j se tiene la ecuación (3.52):

$$\frac{\partial a_j(k)}{\partial z_j(k)} = \frac{\partial}{\partial z_j(k)}\left[\sigma(z_j(k)\right] = \sigma'(z_j(k)) \tag{3.52}$$

donde $\sigma\ es\ la\ función\ de\ activación$ $(sigmoide)$ que viene dada por la fórmula (3.53), algunas propiedades se puede ver en (Haykin,1999).

$$\sigma\left(z_j(k)\right) = \frac{1}{1+\exp(-\alpha * z_j(k))} \tag{3.53}$$

Ahora se considera la derivada parcial de z_j con respecto a los pesos de la capa oculta. Siendo resultante la ecuación (3.54).

$$\frac{\partial z_j(k)}{\partial W_j^D} = \frac{\partial}{\partial W_j^D}\left[\sum_{i=1}^{n} W_{ij}^I(k) * i_i + a_j(k-1) * W_J^D\right] = a_j(k-1) + W_J^D \frac{\partial a_j(k-1)}{\partial W_j^D} \tag{3.54}$$

La ecuación de actualización para los pesos W_j^D se puede expresar como la ecuación (3.55):

$$\frac{\partial E(k)}{\partial W_j^D} = -e_k(k) \frac{\partial e_k(k)}{\partial e_{uk}(k)} W_{jk}^o(k) \sigma'(z_j(k)\left(a_j(k-1) + W_J^D \frac{\partial a_j(k-1)}{\partial W_j^D}\right) \tag{3.55}$$

Si se hace uso de la ecuación (3.56), entonces se llega a la ecuación (3.57):

$$\delta_j(k) = \sigma'\left(z_j(k)\right)\left(a_j(k-1) + W_J^D \delta(k-1)\right) \text{ con } \boldsymbol{\delta}_j(0) = 0 \tag{3.56}$$

$$W_j^D(k+1) = W_j^D(k) - \eta \frac{\partial E(k)}{\partial W_j^D} = W_j^D(k) + \eta e_k(k) W_{jk}^o(k) \delta_j(k) \text{sign}\left(\frac{\partial e_k(k)}{\partial e_{uk}(k)}\right) \tag{3.57}$$

Por último, la ecuación de actualización de los pesos W_{ij}^I si se aplica la regla de la cadena se tiene la ecuación (3.58):

$$\frac{\partial E(k)}{\partial W_{ij}^I} = \frac{\partial E(k)}{\partial e_k(k)} \frac{\partial e_k(k)}{\partial e_{uk}(k)} \frac{\partial e_{uk}(k)}{\partial O(k)} \frac{\partial O(k)}{\partial a_j(k)} \frac{\partial a_j(k)}{\partial z_j(k)} \frac{\partial z_j(k)}{\partial W_{ij}^I} \tag{3.58}$$

donde:

$$\frac{\partial z_j(k)}{\partial W_{ij}^I} = \frac{\partial}{\partial W_{ij}^I}\left[\sum_{i=1}^{n} W_{ij}^I(k) * I_i + a_j(k-1) * W_J^D\right]$$

$$= \frac{\partial}{\partial W_{ij}^I}\left[\sum_{i=1}^{n} W_{ij}^I(k) * I_i\right] + \frac{\partial}{\partial W_{ij}^I}\left[a_j(k-1) * W_J^D\right] \tag{3.59}$$

A continuación se muestra la derivada parcial de z_j con respecto a los pesos de la capa de entrada.

$$\frac{\partial z_j(k)}{\partial W_{ij}^I} = I_i + W_J^D \frac{\partial a_j(k-1)}{\partial W_{ij}^I} \tag{3.60}$$

Entonces, se obtiene la ecuación (3.61):

$$\frac{\partial E(k)}{\partial W_{ij}^I} = e_k(k) \frac{\partial e_k(k)}{\partial e_{uk}(k)} (-1) W_{jk}^o(k) \sigma'(z_j(k) \left[I_i + W_J^D \frac{\partial a_j(k-1)}{\partial W_{ij}^I}\right] \tag{3.61}$$

Pero se tiene que la derivada de a_j con respecto a los pesos de la capa de entrada es la ecuación (3.62):

$$\frac{\partial a_j(k-1)}{\partial W_{ij}^I} = \frac{\partial a_j(k-1)}{\partial z_j(k)} \frac{\partial z_j(k))}{\partial W_{ij}^I} \tag{3.62}$$

Por lo tanto, si se define la ecuación (3.63) la actualización de los pesos W_{ij}^I se puede expresar como la ecuación (3.64):

$$\rho_{ij}(k) = \sigma'\left(z_j(k)\right)\left[I_i + W_J^D \rho_{ij}(k-1)\right] \text{ con } \rho_{ij}(0) = 0. \tag{3.63}$$

La ecuación de actualización para los pesos se puede expresar como:

$$W_{ij}^I(k+1) = W_{ij}^I(k) - \eta \frac{\partial E(k)}{\partial W_{ij}^I} = W_{ij}^I(k) + \eta e_k(k) W_{jk}^o(k) \rho_{ij}(k) \text{sign}\left(\frac{\partial e_k(k)}{\partial e_{uk}(k)}\right) \tag{3.64}$$

3.4. Inicialización de parámetros

En la sección anterior se dedujo el algoritmo de aprendizaje para la red neuronal, pero en ningún caso se ha hablado de qué forma se inicializan los pesos sinápticos. En general, éstos se inicializan al azar dentro de un rango de magnitud pequeño. La razón de escogerlos inicialmente pequeños se encuentra en evitar que exista una inestabilidad en la red neuronal, (Lewis *et al*., 1996).

3.5. Control de un robot manipulador

En esta sección se presentan los resultados teóricos ya conocidos del control PD convencional (Kelly y Santibáñez, 2003) y el PD con compensación neuronal (PD+NN), (Lewis *et al*., 1996).

3.5.1. Control tipo PD de un robot manipulador

La estructura de un control PD convencional es la ecuación (3.65):

$$\tau = K_p(q_d - q) + K_d(\dot{q}_d - \dot{q}) \tag{3.65}$$

donde $K_p, K_d \epsilon R^{nxn}$, son matrices diagonales definidas positivas, simétricas y constantes, las cuales corresponden a las ganancias proporcional y derivativa, $q_d \epsilon R^n$ es la posición articular deseada, $\dot{q}_d \epsilon R^n$es la velocidad articular deseada. En esta investigación se discute el problema de seguimiento de trayectoria, por lo que $\dot{q}_d \neq 0$. Entonces se puede definir el error de seguimiento y la derivada del error de seguimiento como:

$$\tilde{q} = q_d - q, \ \dot{\tilde{q}} = \dot{q}_d - \dot{q} \tag{3.66}$$

En (Lewis, 1996) se define r como una combinación lineal del error de seguimiento y su respectiva derivada, se obtiene la ecuación (3.67):

$$r = \dot{\tilde{q}} + \Lambda\tilde{q} \tag{3.67}$$

donde $\Lambda = \Lambda^{T} > 0$. El control PD ecuación (3.68) se puede expresar como:

$$\tau = K_v r \tag{3.68}$$

donde $K_v \epsilon R^{nxn}$, es una matriz simétrica definida positiva.

El comportamiento en malla cerrada de un robot de n g. d. l. bajo el control PD se obtiene combinando el modelo (1.1) con la ley de control (3.69).

$$M(q)\ddot{q}_d + C(q,\dot{q})\dot{q} + G(q) + F\dot{q} = K_v r \tag{3.69}$$

Si el modelo (1.1) se expresa en términos de r y manipulando la ecuación (3.69) el sistema en lazo cerrado se muestra en la ecuación (3.70):

$$M(q)\dot{r} = -K_v r - C(q,\dot{q})r + z + G(q) + F\dot{q} \tag{3.70}$$

donde $z = M(q)\left(\ddot{q}_d + \Lambda\dot{\tilde{q}}\right) + C(q,\dot{q})(\dot{q}_d + \Lambda\tilde{q})$.

El análisis de estabilidad de la ecuación en malla cerrada se realiza considerando la función candidata de Lyapunov:

$$V(r) = \frac{1}{2} r^{T} M r \tag{3.71}$$

Se percibe que la función candidata de Lyapunov es definida positiva ya que la matriz M es definida positiva.

La derivada de la función $V(r)$ es la ecuación (3.72):

$$\dot{V}(r) = r^{T} M\dot{r} + \frac{1}{2} r^{T}\dot{M} r \tag{3.72}$$

Si se utiliza la ecuación (3.72) y sustituyendo en la ecuación anterior se llega a la ecuación (3.73):

$$\dot{\mathrm{V}}(\mathrm{r}) = -\mathrm{r}^{\mathrm{T}}\mathrm{K}_{\mathrm{v}}\mathrm{r} + \mathrm{r}^{\mathrm{T}}(\mathrm{z} + \mathrm{G}(\mathrm{q}) + \mathrm{F}\dot{\mathrm{q}}) \tag{3.73}$$

donde se eliminó el término $\frac{1}{2}\mathrm{r}^{\mathrm{T}}\left[\dot{\mathrm{M}}\mathrm{r} - \mathrm{C}(\mathrm{q}, \dot{\mathrm{q}})\right]\mathrm{r}$. Las propiedades del modelo dinámico puede ser vistas en (Kelly y Santibáñez, 2003).

Con el fin de asegurar que $\dot{V}(r) \leq 0$, se necesita que la ecuación (3.73) cumpla:

$$\|\mathrm{r}\| = \frac{\mathrm{d}}{\lambda_{\min}\mathrm{k}_{\mathrm{v}}} \tag{3.74}$$

donde $\|z + G(q) + F\dot{q}\| \leq d$, siendo d una constante positiva, por lo tanto, significa que el error de seguimiento r va a converger a una bola con radio determinado por $\lambda_{min}k_v$ de manera inversamente proporcional. Se puede concluir estabilidad del sistema el lazo cerrado con el controlador tipo PD.

3.5.2. Control neuronal

La principal propiedad de las redes neuronales es la propiedad de aproximación de funciones (Cibenko, 1989). Esta propiedad se enuncia a continuación.

Propiedad. Dada cualquier función $f \in C(s)$ (el espacio de funciones continuas) con un S un subconjunto compacto de R^n y cualquier $\varepsilon > 0$, existe una suma $G(x)$ de la forma:

$$G(x) = \sum_{k=0}^{L} \mu_k \sigma(m_k^T x + n_k)$$

Para algún $L, m_k \in R^n, n_k \in R$ tal que

$$G(x) - f(x) < \varepsilon$$

Para toda $x \in S$.

Generalmente, si ε más pequeño, entonces el numero L de neuronas de la capa oculta es más grande, es decir, es requerida una red neuronal artificial más grande para una mayor precisión.

La función $\sigma(\cdot)$ puede ser cualquier función sigmoidal continua, donde una función sigmoidal está definida como:

$$\sigma(x) \rightarrow \begin{cases} 1 \ldots para\ x \rightarrow +\infty \\ 0 \ldots para\ x \rightarrow -\infty \end{cases}$$

En este resultado muestra que cualquier función continua puede ser aproximada usando una combinación lineal de funciones sigmoidales. Esta es conocida como la propiedad de aproximación universal de las redes neuronales.

Si los términos $G(q)\ y\ F\dot{q}$ no se conocen, una RNA tipo perceptrón multicapa con retroalimentación local, puede usarse para aproximar dichos términos.

$$\widehat{\mathrm{V}}^{\mathrm{T}}\sigma(\widehat{\mathrm{W}}^{\mathrm{T}}\mathrm{i} + \mathrm{a}\widehat{\mathrm{P}}) \approx \mathrm{G(q)} + \mathrm{F\dot{q}} + \mathrm{z} \qquad (3.75)$$

done $z = M(q)\left(\ddot{q_d} + \Lambda\dot{\tilde{q}}\right) + C(q, \dot{q})(\dot{q}_d + \Lambda\tilde{q})$. Dado que existe una red neuronal ideal, entonces se puede escribir la ecuación (3.76).

$$\mathrm{G(q)} + \mathrm{F\dot{q}} + \mathrm{z} = \mathrm{V}^{\mathrm{T}}\boldsymbol{\sigma}(\mathrm{W}^{\mathrm{T}}\mathrm{i} + \mathrm{aP}) + \boldsymbol{\varepsilon}(\mathrm{i}) \qquad (3.76)$$

Siendo W,V y P matrices de pesos constantes ideales, $\varepsilon(i)$ es el error de estructura de la RNA que satisface $\|\varepsilon\| < \varepsilon_N$, con $\varepsilon_N \in R$, $\sigma(\cdot)$ es una función de activación e $\mathrm{i} = \left[\tilde{q}^T\ \dot{\tilde{q}}^T\ {q_d}^T\ {\dot{q}_d}^T\ {\ddot{q}_d}^T\ r^T\ q^T\ \dot{q}^T\right]^T$ representa las entradas de la red.

Se asume que los pesos sinápticos ideales W,V y P esta acotados por valores conocidos, tal que

$$\|W\|_F < W_M, \|V\|_F < V_M \text{ y } \|P\|_F < P_M \quad (3.77)$$

con W_M, V_M y P_M constantes conocidas. La norma frobenius está definida en (Lewis, 1996).

El control PD+red neuronal se puede representar por la ecuación (3.78)

$$\boldsymbol{\tau} = K_v r + \hat{V}^T \sigma\left(\hat{W}^T x + a\hat{P}\right) - v \quad (3.78)$$

con $\hat{V}, \hat{W}$ y $\hat{P}$ los valores estimados de los pesos ideales de la red proporcionados por el algoritmo de actualización. El término v es una función que provee de robustez en el enfrentamiento de los errores de la red (Lewis *et al.*, 1996). El sistema en lazo cerrado de la figura (3.4) se puede obtener sustituyendo la acción de control (3.78) en la ecuación (1.1) como se muestra en la ecuación (3.79).

$$M(q)\ddot{q}_d + C(q,\dot{q})\dot{q} + G(q) + F\dot{q} = K_v r + \hat{V}^T \sigma\left(\hat{W}^T x + a\hat{P}\right) - v \quad (3.79)$$

Agregando y restando $V^T \sigma\left(\hat{W}^T i + a\hat{P}\right)$ de la ecuación (3.79) y haciendo uso de las ecuaciones (3.78) y (3.79), se puede escribir la ecuación (3.80):

$$M(q)\dot{r} = -(K_v + C)r + e_V{}^T \boldsymbol{\sigma}\left(\hat{W}^T i + \hat{P}^T a\right) + \zeta + v \quad (3.80)$$

donde se puede describir los siguientes términos.

$e_V = V - \hat{V}$, es el error de estimación de los pesos de la red (3.81)

$\zeta = V^T\left[\sigma(W^T i + Pa) - \sigma\left(\hat{W}^T i + \hat{P} a\right)\right] + \varepsilon$, es un término de perturbación acotado $\|\zeta\| \leq \bar{\zeta}$ para alguna constante positiva $\bar{\zeta}$.

A continuación se da el algoritmo de ajuste de pesos que garantiza la estabilidad del sistema. Se requiere demostrar que el error de seguimiento $\tilde{q}$ esta acotado.

El algoritmo de actualización de los pesos de la red se escriben como:

$$\dot{\hat{V}} = S\boldsymbol{\sigma}\tilde{q}^{T} \tag{3.82}$$

$$\dot{\hat{P}} = T\hat{\nabla}\tilde{q} \tag{3.83}$$

$$\dot{\hat{W}} = \rho\hat{\nabla}\tilde{q} \tag{3.84}$$

donde:

$$S = \boldsymbol{\eta}_1 S > S^T > 0 \tag{3.85}$$

$$T = \boldsymbol{\eta}_2 \boldsymbol{\delta},\ T = T^{T} > 0 \tag{3.86}$$

$\boldsymbol{\eta}_1, \boldsymbol{\eta}_2 > 0$ son el factor de aprendizaje.

La introducción de las matrices S y T en los términos de corrección permiten simplificar el estudio de estabilidad.

Las reglas de aprendizaje ecuaciones (3.82), (3.83) y (3.84) en términos de los errores de los pesos $e_V = V - \hat{V}$, $e_W = W - \hat{W}$ y $e_P = P - \hat{P}$ pueden ser escritas como las ecuaciones (3.87), (3.88) y (3.89):

$$\dot{e}_V = -S\sigma\tilde{q}^{T} \tag{3.87}$$

$$\dot{e}_P = -T\hat{\nabla}\tilde{q} \tag{3.88}$$

$$\dot{e}_W = -\rho\hat{\nabla}\tilde{q} \tag{3.89}$$

Para estudiar la estabilidad se empleará el método directo de Lyapunov. Si se considera la siguiente función candidata de Lyapunov.

$$\vartheta(d_c) = \frac{1}{2}\tilde{q}^{T}\tilde{q} + \frac{1}{2}r^{T}Mr + \frac{1}{2}tr(e_V{}^{T}S^{-1}e_V) + \frac{1}{2}tr(e_P{}^{T}T^{-1}e_P) + \frac{1}{2}tr(e_W{}^{T}e_W) \tag{3.90}$$

con $d_c = \{\tilde{q}, r, e_V, e_P \text{ y } e_W\}$. Se observa que ésta es definida positiva y radialmente desacotada porque tanto $M(q), S, T$ y son matrices definidas positivas.

La derivada con respecto al tiempo de $\vartheta(\mathrm{d_c})$ es:

$$\dot{\vartheta} = \tilde{\mathrm{q}}^{\mathrm{T}}\dot{\tilde{\mathrm{q}}} + \mathrm{r}^{\mathrm{T}}\mathrm{M}\dot{\mathrm{r}} + \frac{1}{2}\mathrm{r}^{\mathrm{T}}\dot{\mathrm{M}}\mathrm{r} + \mathrm{tr}(\mathrm{e_V}^{\mathrm{T}}\mathrm{S}^{-1}\dot{\mathrm{e_V}}) + \mathrm{tr}(\mathrm{e_p}^{\mathrm{T}}\mathrm{T}^{-1}\dot{\mathrm{e_p}}) + \mathrm{tr}(\mathrm{e_W}^{\mathrm{T}}\dot{\mathrm{e_W}}) \tag{3.91}$$

Al sustituir $\mathrm{M}\dot{\mathrm{r}}$ de la ecuación de lazo cerrado (3.80) y los errores de los pesos de la red (3.87), (3.88), (3.89) se llega a la ecuación (3.92):

$$\dot{\vartheta} = \tilde{\mathrm{q}}^{\mathrm{T}}\mathrm{r} - \tilde{\mathrm{q}}^{\mathrm{T}}\boldsymbol{\Lambda}\tilde{\mathrm{q}} + \mathrm{r}^{\mathrm{T}}\left[-(\mathrm{K_v} + \mathrm{C})\mathrm{r} + \mathrm{e_V}^{\mathrm{T}}\boldsymbol{\sigma}\left(\widehat{\mathrm{W}}^{\mathrm{T}}\mathrm{i} + \mathrm{a}\widehat{\mathrm{P}}\right) + \boldsymbol{\zeta} + \mathrm{v}\right] + \frac{1}{2}\mathrm{r}^{\mathrm{T}}\dot{\mathrm{M}}\mathrm{r} + \mathrm{tr}(-\mathrm{e_V}^{\mathrm{T}}\boldsymbol{\sigma}\tilde{\mathrm{q}}^{\mathrm{T}}) + \mathrm{tr}(-\mathrm{e_P}^{\mathrm{T}}(\widehat{\nabla}\tilde{\mathrm{q}})) + \mathrm{tr}(-\mathrm{e_w}^{\mathrm{T}}(\boldsymbol{\rho}\widehat{\nabla}\tilde{\mathrm{q}})) \tag{3.92}$$

Si se manipula la ecuación (3.92) se tiene la ecuación (3.93):

$$\dot{\vartheta} = \tilde{\mathrm{q}}^{\mathrm{T}}\mathrm{r} - \tilde{\mathrm{q}}^{\mathrm{T}}\boldsymbol{\Lambda}\tilde{\mathrm{q}} - \mathrm{r}^{\mathrm{T}}\mathrm{K_v}\mathrm{r} + \mathrm{r}^{\mathrm{T}}\left[\mathrm{e_V}^{\mathrm{T}}\sigma\left(\widehat{\mathrm{W}}^{\mathrm{T}}\mathrm{i} + \widehat{\mathrm{P}}\mathrm{a}\right)\right] + \mathrm{r}^{\mathrm{T}}\zeta + \mathrm{r}^{\mathrm{T}}\mathrm{v} + \mathrm{tr}(-\mathrm{e_V}^{\mathrm{T}}\sigma\tilde{\mathrm{q}}^{\mathrm{T}}) + \mathrm{tr}\left(-\mathrm{e_P}^{\mathrm{T}}(\widehat{\nabla}\tilde{\mathrm{q}})\right) + \mathrm{tr}(-\mathrm{e_w}^{\mathrm{T}}(\rho\widehat{\nabla}\tilde{\mathrm{q}})) \tag{3.93}$$

donde se ha eliminado el término $r^T\left[\frac{1}{2}\dot{M} - C\right]r$ en virtud de la propiedad de la matriz C, ver apéndice B. Se buscará ahora acotar superiormente a $\dot{\vartheta}$ por una función definida negativa en términos de r y $\tilde{q}$. por lo tanto $\dot{\vartheta}$ queda:

$$\begin{aligned}\dot{\vartheta} \leq -\|\mathrm{r}\|\Bigg[&\left(\boldsymbol{\lambda}_{\min}\{\mathrm{K_v}\}\|\mathrm{r}\| - \boldsymbol{\sigma}_{\mathrm{m}}\|\mathrm{e_v^T}\| - \bar{\boldsymbol{\zeta}} + \mathrm{K_z}\left\|\hat{\mathrm{Z}}\right\|_{\mathrm{F}}\left(\left\|\hat{\mathrm{Z}}\right\|_{\mathrm{F}} + \mathrm{Z_M}\right)\|\mathrm{r}\|\right)\\ &+\left(\|\tilde{\mathrm{q}}\|\left[\frac{\boldsymbol{\lambda}_{\min}\{\boldsymbol{\Lambda}\}\|\tilde{\mathrm{q}}\|}{\|\mathrm{r}\|} - 1 - \frac{\boldsymbol{\sigma}_{\mathrm{m}}\|\mathrm{e_v^T}\|}{\|\mathrm{r}\|} - \frac{\left\|\mathrm{e_p}\right\|(\mathrm{V_m} + \|\mathrm{e_V}\|)}{\|\mathrm{r}\|}\right.\right.\\ &\left.\left.- \frac{\|\mathrm{e_W}\|[\boldsymbol{\sigma}_{\mathrm{m}}(\mathrm{C_1Q_d} + \mathrm{C_2}\|\mathrm{r}\| + \mathrm{P_M}\|\boldsymbol{\rho}\|)\mathrm{k_1}]}{\|\mathrm{r}\|}\right]\right)\Bigg]\end{aligned} \tag{3.94}$$

Se ha definido $\mathrm{k_1} = (\mathrm{V_m} + \|\mathrm{e_V}\|)$, por definición, $\mathrm{C_1}, \mathrm{C_2}, \mathrm{Q_d}, \sigma_{\mathrm{m}} > 0$ son constantes positivas, $\mathrm{k_2} = \sigma_{\mathrm{m}}(\mathrm{C_1Q_d} + \mathrm{C_2}\|\mathrm{r}\| + \mathrm{P_M}\|\rho\|)$.

Entonces, $\dot{\vartheta}$ es semidefinida negativa si los términos entre paréntesis de (3.94) son positivos. Por lo tanto se pueden encontrar las siguientes condiciones.

$$\|r\| > \frac{\bar{\zeta}}{\lambda_{min}\{K_v\} - \frac{K_z Z_M^2}{4}} \text{ si } \lambda_{\min}\{K_v\} > \frac{K_z Z_M^2}{4} \tag{3.95}$$

Por otro lado el error de seguimiento debe cumplir la ecuación (3.96).

$$\|\tilde{q}\| > \frac{\|r\|}{\lambda_{min}\{\Lambda\}} \tag{3.96}$$

Finalmente, también se puede decir que los errores de los pesos están acotados

$$\left\|\tilde{Z}\right\|_F > \frac{Z_M}{2} + \sqrt{\frac{\frac{Z_M^2 K_z}{4} + \bar{\zeta}}{K_z}} \tag{3.97}$$

De acuerdo al teorema de Lyapunov si se cumple (3.95) y (3.96) la variable r definida como la combinación del error de velocidad y error de seguimiento es últimamente acotado. Con el fin de asegurar que $\dot{\vartheta} \leq 0$, se necesita que $\|r\| > \frac{\bar{\zeta}}{\lambda_{min}\{K_v\} - \frac{K_z Z_M^2}{4}}$ lo cual significa que el error de seguimiento r va a converger a un bola con radio determinado por K_v de manera inversamente proporcional.

IV. RESULTADOS Y DISCUSIÓN

Una vez que se ha presentado el algoritmo de control, en esta sección se muestra algunos de los resultados de la aplicación de los siguientes controladores: Control neuronal, PD compensado y el control PID, además, se muestra las simulaciones del modelo obtenido.

Los experimentos realizados se dividieron en dos pruebas: En el primero se muestra el desempeño de los controladores cuando no existe incertidumbre en el modelo dinámico, concretamente cuando la masa de los eslabones no cambia y el segundo cuando existe un cambio en la masa del eslabón dos y el termino de fricción se agrega en el modelo dinámico.

4.1. Simulaciones

Para efectos de simulación se usa como modelo de referencia, un robot manipulador de dos grados de libertad para evaluar los algoritmos de control utilizados y el algoritmo propuesto. Se sabe que el modelo dinámico de un robot es la ecuación (1.1).

4.1.1. Modelo dinámico

En esta sección se realizan las simulaciones del modelo dinámico obtenido, para ello se emplea el software MATLAB-SIMULINK 7.2. Si se pretende simular el comportamiento dinámico del manipulador ante determinados pares, es necesario resolver el modelo (1.1) para obtener las trayectorias articulares.

$$\begin{bmatrix}\ddot{q}_1\\ \ddot{q}_2\end{bmatrix} = M^{-1}\left[\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix} - \begin{bmatrix}C_{11}(q,\dot{q}) & C_{12}(q,\dot{q})\\ C_{21}(q,\dot{q}) & C_{22}(q,\dot{q})\end{bmatrix}\begin{bmatrix}\dot{q}_1\\ \dot{q}_2\end{bmatrix} - \begin{bmatrix}G_1(q)\\ G_2(q)\end{bmatrix}\right] \tag{4.1}$$

donde, $M, C\ y\ G$ están definidas en el apéndice B.

En las simulaciones llevadas a cabo, se consideraron los parámetros mostrados en la tabla 4.1. Cabe mencionar que estos valores fueron obtenidos de un robot prototipo que nos facilitó el Dr. Víctor Manuel Hernández Guzmán.

Tabla 4.1.Parámetros de simulación del robot rígido.

Variable	Significado	Valor
m_1	Masa del eslabón 1	0.763 Kg
m_2	Masa del eslabón 2	0.343 Kg
l_1	Longitud del eslabón 1	0.193 m
l_2	Longitud del eslabón 2	0.145 m
l_{c1}	Distancia al centro de masa del eslabón 1	0.136 m
l_{c2}	Distancia al centro de masa del eslabón 2	0.048 m
I_1	Momento de inercia del eslabón 1	0.0186 $Kg\ m^2$
I_2	Momento de inercia del eslabón 2	0.0026987 $Kg\ m^2$
G	Gravedad	9.81 m/s^2

Se trata de obtener las trayectorias que seguiría el robot de la (figura 3.1) al aplicarle los diferentes pares.

Simulación 1: Los pares aplicados corresponden a $\tau_1 = 0$ Nm y $\tau_2 = 0.18$ Nm. Cuando se asignan dichos valores se genera una especie de círculo en el plano X-Y, cuyo centro no es fijo debido a la inercia que genera el eslabón 2 al moverse en el plano X-Y, obsérvese la (figura 4.1). Como resultado, el extremo del segmento de l_2 seguirá una trayectoria dada por:

$$x = l_1 sen(q_1) + l_2 sen(q_1 + q_2)$$
$$\mathrm{y} = -\mathrm{l}_1\cos(\mathrm{q}_1) - \mathrm{l}_2\cos(\mathrm{q}_1 + \mathrm{q}_2) \qquad (4.2)$$

Es importante mencionar que se consideran en todos los resultados numéricos la condición inicial del robot es ($q = 0, \dot{q} = 0$).

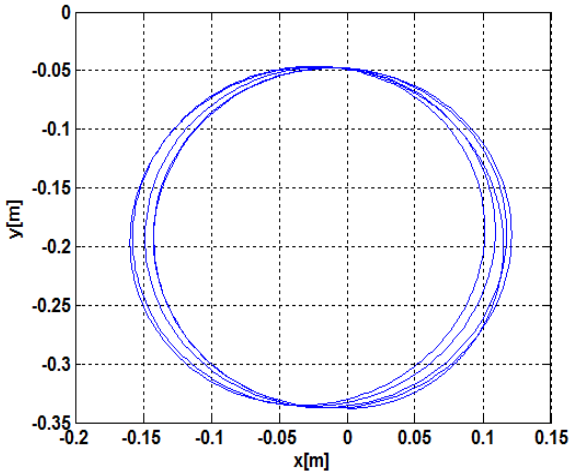


Figura 4.1.Resultados de la simulación 1 para valores de $\boldsymbol{\tau_1 = 0\ \mathbf{Nm}}$ y $\boldsymbol{\tau_2 = 0.1\mathbf{Nm}}$.

Simulación 2. Para este caso, los valores de los pares son: $\tau_1 = 2$ Nm y $\tau_2 = 0$ Nm. Para esta simulación se genera un círculo de radio mayor que en la simulación 1, debido a que ahora el desplazamiento es realizado por el eslabón 1 en el plano X-Y, obsérvese la (figura 4.2).

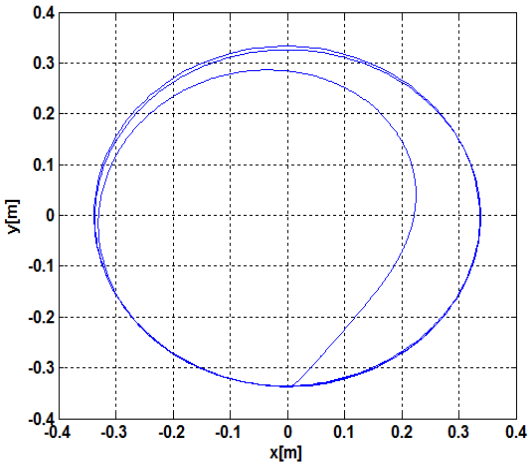


Figura 4.2.Resultados de la simulación 2 para valores de $\boldsymbol{\tau_1 = 2\ \mathbf{Nm}}$ y $\boldsymbol{\tau_2 = 0.1\mathbf{Nm}}$.

Simulación 3. En este caso, se asignan valores diferentes de cero a ambos pares, es decir, $\tau_1 = 3$ Nm y $\tau_1 = 0.8$ Nm. Según se observa en la (figura 4.3), ambos eslabones se mueven en el plano X-Y generando un movimiento con forma de espiral.

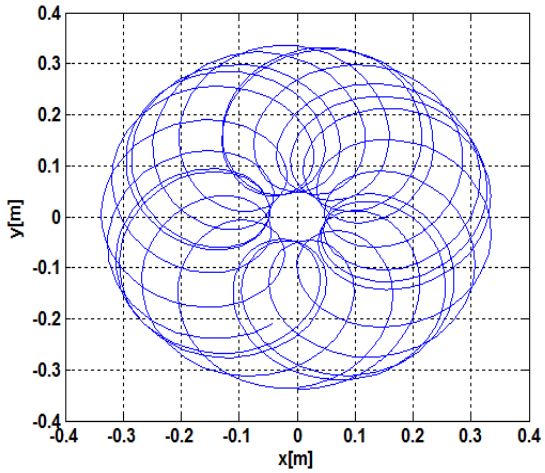


Figura 4.3.Resultados de la simulación 2 para valores de $\boldsymbol{\tau_1 = 3\ Nm}$ y $\boldsymbol{\tau_2 = 0.8 Nm}$.

4.1.2. Controlador PD con compensación

Un controlador PD con compensación para el seguimiento de trayectoria de un mecanismo de dos eslabones de acuerdo a (Kelly y Santibáñez, 2003) es:

$$\tau = K_p\tilde{q} + K_v\dot{\tilde{q}} + M(q)\left[\ddot{q}_d + \Lambda\dot{\tilde{q}}\right] + C(q,\dot{q})[\dot{q}_d + \Lambda\tilde{q}] + g(q) \qquad (4.3)$$

donde $K_p, K_v \in R^{nxn}$ son matrices simétricas definidas positivas de diseño y Λ se define de la siguiente manera:

$$\Lambda = {K_v}^{-1}K_p \qquad (4.4)$$

Este control se basa en el modelo dinámico del robot, es decir, añade términos no lineales, $M(q)$, $C(q,\dot{q})$ y $g(q)$. Dado que en esta investigación se trata de estimar dichos términos, es deseable realizar comparaciones con los resultados obtenidos del controlador neuronal, por esta razón, se ha decidido realizar simulaciones del control PD con compensación. También podría cuestionarse el motivo de comparar los resultados con un controlador menos elaborado, por tal razón, también se realizan simulaciones del controlador tipo PID.

Prueba 1 del controlador PD con compensación.

Como se detalló anteriormente, para esta simulación se sigue una trayectoria seno, ecuaciones (4.5) y (4.6), considerando que se tiene un conocimiento exacto en los valores que presentan los términos no lineales del robot.

$$q_{d1} = -6.2832 \sin(0.05t) \tag{4.5}$$

$$q_{d2} = 3.1416 \sin(0.1t) \tag{4.6}$$

donde las ganancias proporcional y derivativa se escogieron como:

$$K_p = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.7}$$

$$K_v = \begin{bmatrix} 0.4 & 0 \\ 0 & 0.1 \end{bmatrix} \tag{4.8}$$

y por tanto $\Lambda = {K_v}^{-1} K_p = diag\{10,10\}$.

Las condiciones iniciales correspondientes a las posiciones y velocidades son:

$$q_1(0) = 0, \quad q_2(0) = 0,$$
$$\dot{q}_1(0) = 0, \quad \dot{q}_2(0) = 0,$$

Para calcular el error cometido se utilizo el índice de desempeño error cuadrático promedio (ISE), (Sharkawy *et al*., 2011), el cual se puede escribir como:

$$E_{prom} = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(q_d - q)^2} \tag{4.9}$$

donde N representa el número de muestras, q_d la posición deseada y q la posición actual.

La tabla 4.2 muestra los valores del índice de desempeño presentado por el controlador PD con compensación.

Tabla 4.2.Índice de desempeño del controla PD con compensación para la prueba 1.

Controlador PD con compensación	ISE
Eslabón1	0.0008066
Eslabón2	0.0003772

En la (figura 4.4) se observa el comportamiento de las articulaciones 1 y 2 respectivamente. Se puede percatar que las posiciones q_1 y q_2 siguen a las trayectorias deseadas con un error mínimo como se puede apreciar en la tabla 4.2.

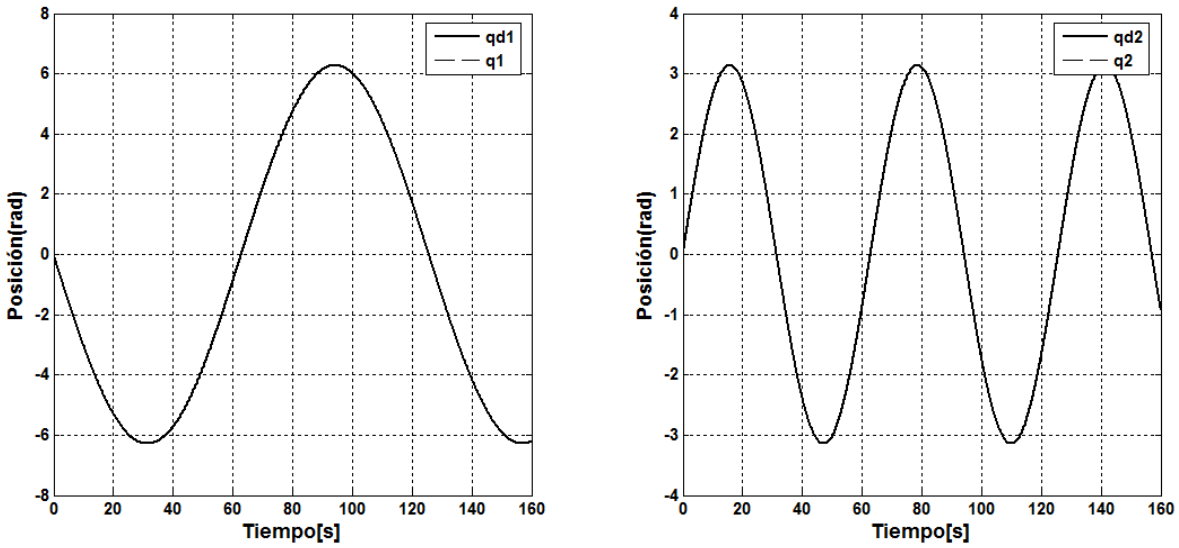


Figura 4.4. Evolución de las trayectorias del controlador PD con compensación prueba 1.

A partir de los datos mostrados en la tabla 4.2, se observa que la articulación presenta un esfuerzo de control más elevado presentado en la articulación 2. El par 1 oscila entre $\pm 1.7\ N-m$, mientras que el par 2 entre $\pm 0.16\ N-m$, (figura 4.5).

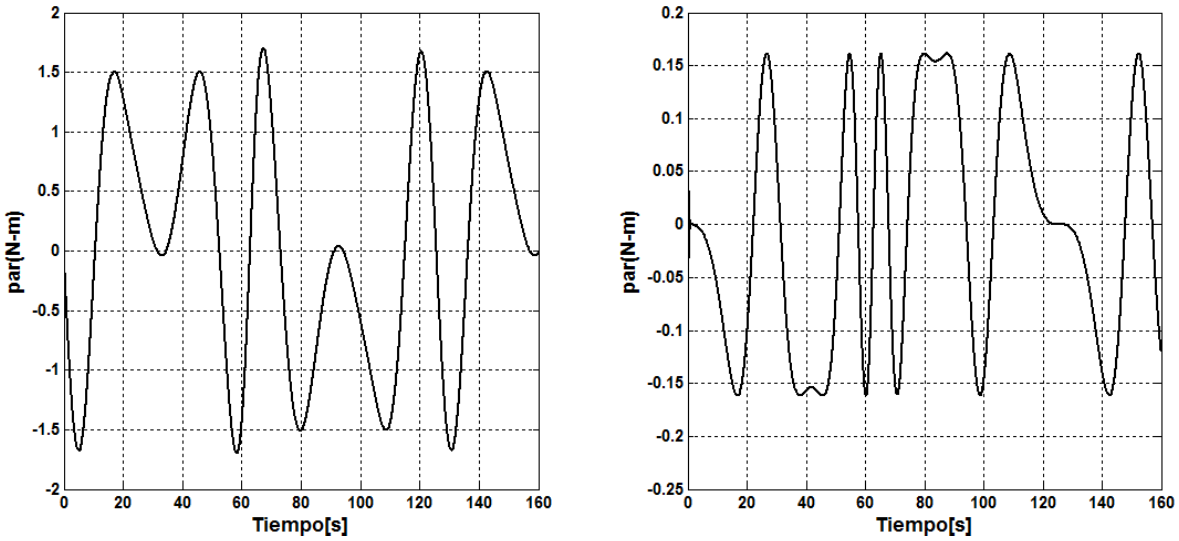


Figura 4.5. Señales de torque del control PD con compensación prueba 1.

Prueba 2 del controlador PD con compensación.

Esta simulación es similar a la prueba uno al buscar seguir una trayectoria, pero para esta simulación se considera una incertidumbre del 30% en el valor de la masa del eslabón 2 y se agrega un término de fricción $F = diag[0.2\ 0.2]\dot{q}$ durante el intervalo de 80 a 130 s. La figura 4.6 muestra la evolución de las posiciones articulares uno y dos, mientras que la tabla 4.3 muestra el índice producido por el controlador PD con compensación.

Tabla 4.3.Índice de desempeño del controla PD con compensación para la prueba 2.

Controlador PD con compensación	ISE
Eslabón1	1.4922
Eslabón2	0.5870

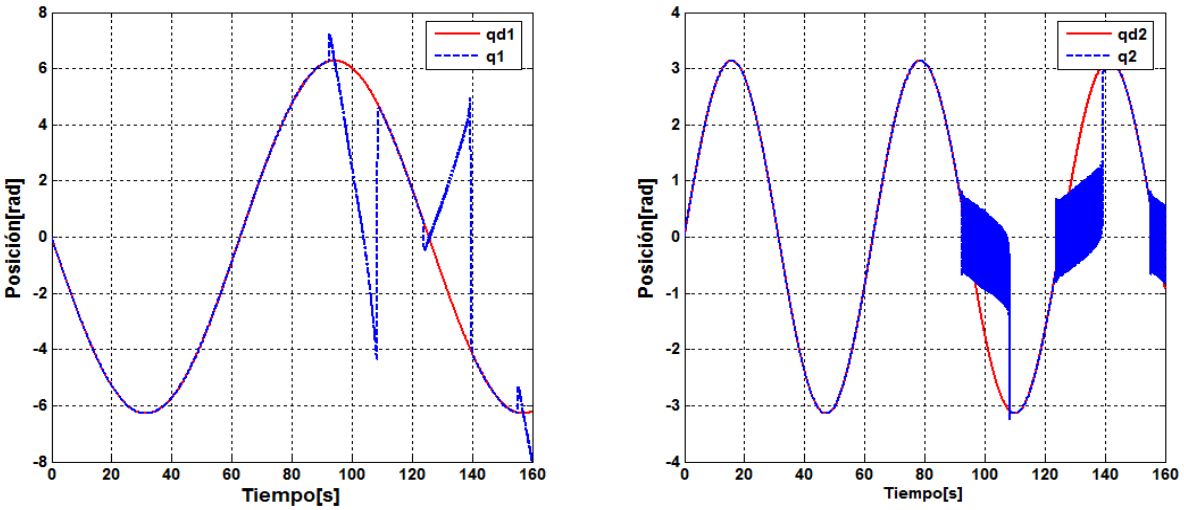


Figura 4.6. Evolución de las trayectorias control PD con compensación prueba 2.

A partir de la figura 4.6 se puede observar, al variar el valor de la masa del eslabón 2, ya no se cumple el objetivo de seguimiento. La posición del eslabón uno sufre varias variaciones, al igual que la posición dos durante el intervalo de tiempo 80 a 130 s.

En la (figura 4.7) se muestra que el par de control de seguimiento de trayectoria uno presenta un impulso de 70 N-m más elevado que la señal de control dos que presenta un par de 17 N-m, sin embargo, en ambos seguimientos de trayectoria tienen pares de control pequeños cuando no existe cambio de masa.

En la (figura 4.8) se confirma lo que se mostro en la tabla 4.3, que el error de seguimiento uno presenta el error más grande cuando existe un cambio de masa y cuando el termino de fricción se añade.

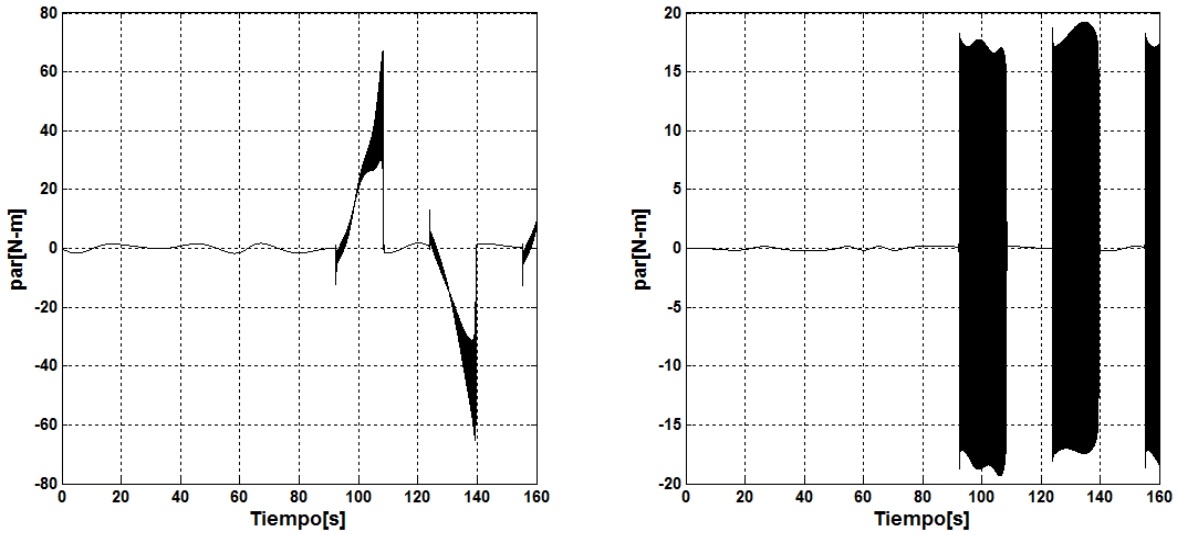


Figura 4.7. Señales de pares del control PD con compensación-prueba 2.

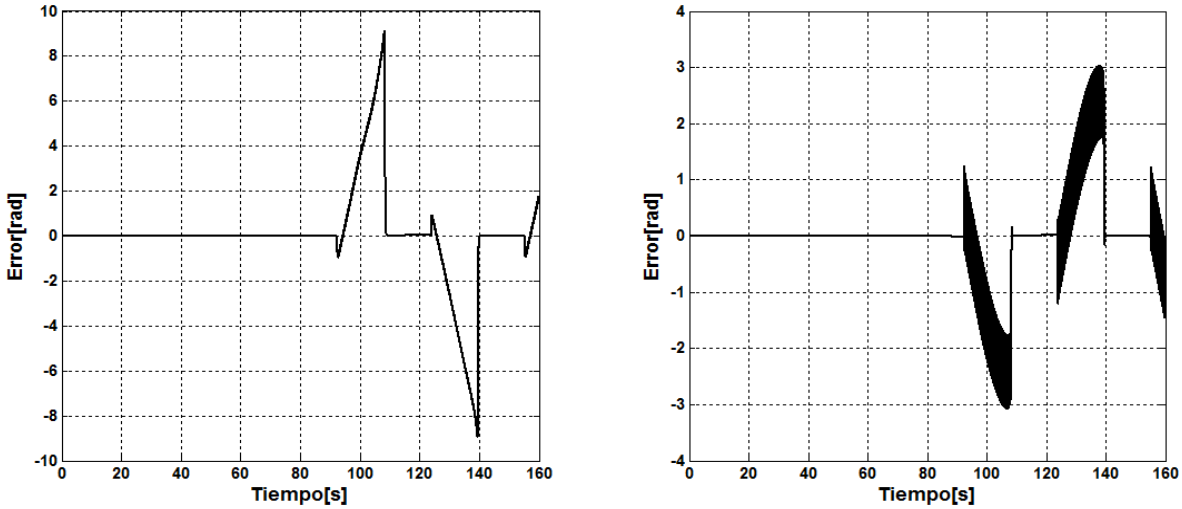


Figura 4.8. Errores del control PD con compensación prueba 2.

4.1.3. Controlador PID

La ley de control PID puede expresarse como se muestra en la ecuación (4.10), de acuerdo a (Kelly y Santibáñez, 2003):

$$\tau = K_p\tilde{q} + K_v\dot{\tilde{q}} + K_i\int_0^t q(\sigma)d\sigma \qquad (4.10)$$

donde las matrices de diseño $K_p, K_v, K_i \in R^{nxn}$, llamadas respectivamente las ganancias proporcional, derivativa e integral, son matrices simétricas y definidas positivas.

El objetivo de estas simulaciones es observar el comportamiento del seguimiento de trayectoria al aplicar las mismas pruebas que se realizaron con el control PD con compensación. Siguiendo el procedimiento de sintonía de acuerdo a las condiciones que se estudiaron en la clase de tópicos de control no lineal, finalmente se obtuvieron las siguientes matrices:

$$K_p = \begin{bmatrix} 30 & 0 \\ 0 & 25 \end{bmatrix}, K_v = \begin{bmatrix} 0.8 & 0 \\ 0 & 0.45 \end{bmatrix}, K_i = \begin{bmatrix} 9 & 0 \\ 0 & 2 \end{bmatrix}$$

Prueba 1 del controlador PID

Como se detalló anteriormente, para esta prueba se siguen las mismas trayectorias (4.5) y (4.6). La tabla 4.4 muestra los valores del índice de desempeño presentado por el controlador PID. Se observa que existe un error mayor en el eslabón 1. Esto se debe a que la articulación mueve a dos eslabones, mientras que la articulación dos mueve a una sola masa.

Tabla 4.4.Índice de desempeño del controla PID para la prueba 1.

Controlador PID	ISE
Eslabón1	0.0201
Eslabón2	3.968×10^{-4}

En la (figura 4.9) se observa el comportamiento del controlador PID ante seguimiento de trayectoria, se nota también un comportamiento favorable en los índices del error como se muestra en la tabla 4.4. La (figura 4.10) se percibe que las señales de control son muy similares a las obtenidas en el control PD con compensación, es de destacar que el controlador PID no necesita términos adicionales como los demás controladores.

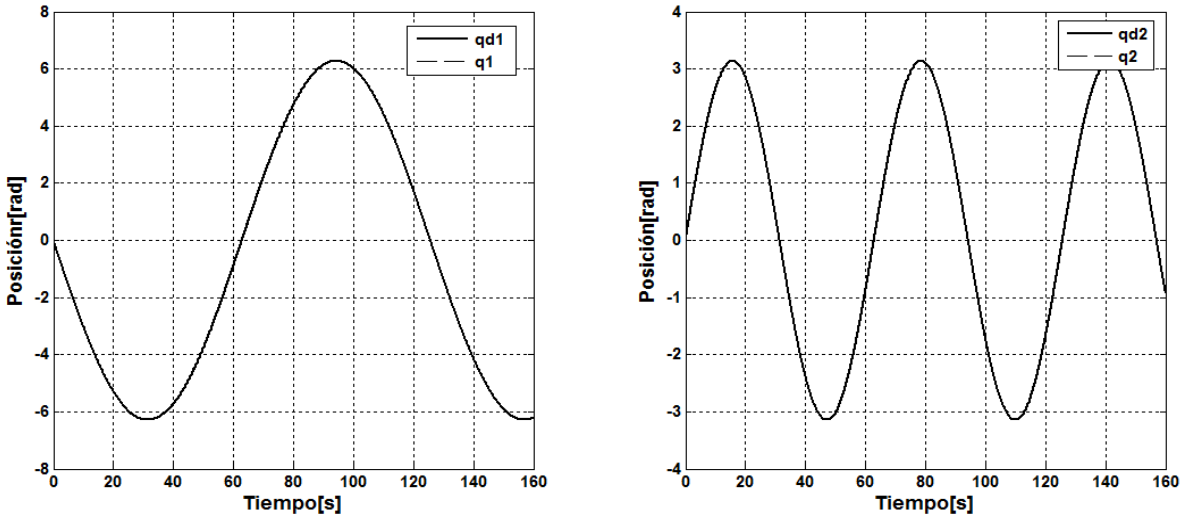


Figura 4.9.Evolución de las trayectorias de la prueba 1.

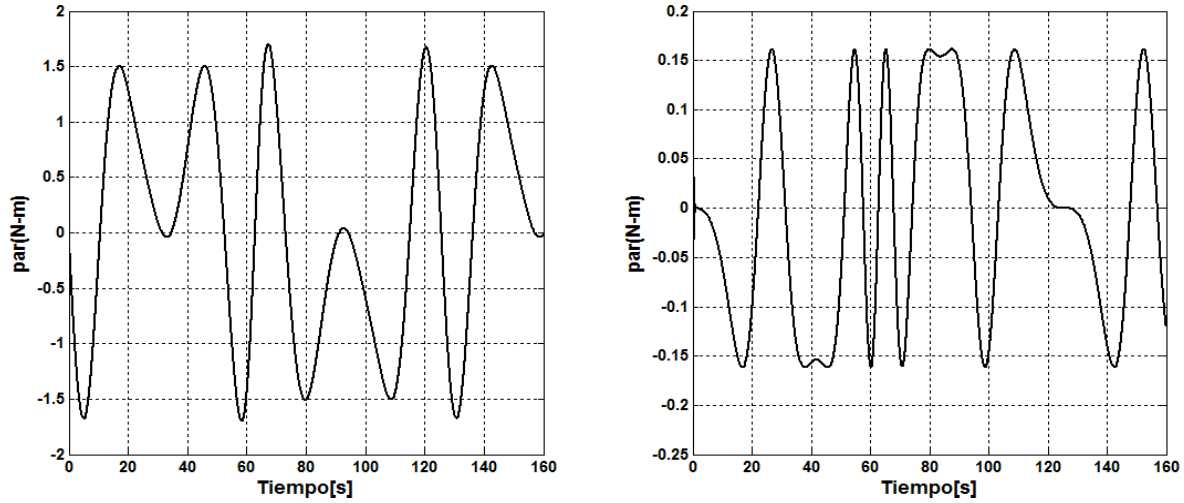


Figura 4.10.Señales de control de la prueba 1.

Prueba 2 del controlador PID

Esta simulación es similar a la prueba uno al buscar seguir una trayectoria, pero para esta simulación se considera una incertidumbre del 30% en el valor de la masa del eslabón 2 y se agrega un término de fricción $F = diag[0.2\ 0.2]\dot{q}$ durante el intervalo de 80 a 130 s. La (figura 4.11) muestra la evolución de las posiciones articulares uno y dos, donde se observa que el control PID es capaz de sobreponerse al incremento de la masa del

eslabón 2 y con un termino de fricción, mientras que la tabla 4.5 muestra el índice producido por el controlador PID.

Tabla 4.5. Índice de desempeño del controla PID con compensación para la prueba 1.

Controlador PID	ISE
Eslabón1	0.0212
Eslabón2	0.0046

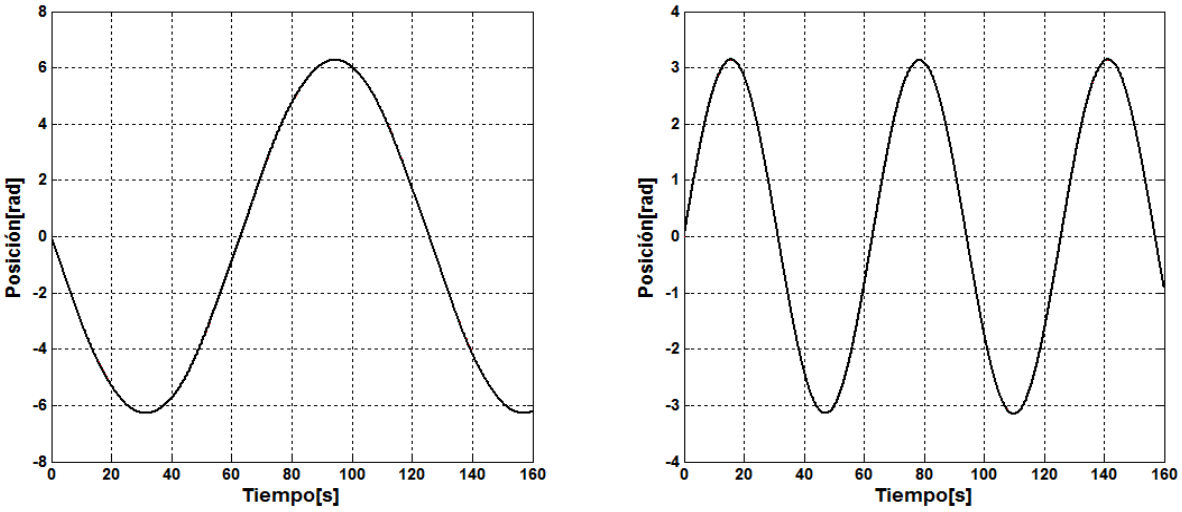


Figura 4.11.Evolución de las trayectorias de la prueba 2.

En la siguiente (figura 4.12), se muestran los pares de control que son similares a los del control PD con compensación. Al variar la masa del eslabón 2 y con el término de fricción durante 80 a 130 s, se observa que el controlador puede manejar el incremento en el valor de los parámetros del sistema.

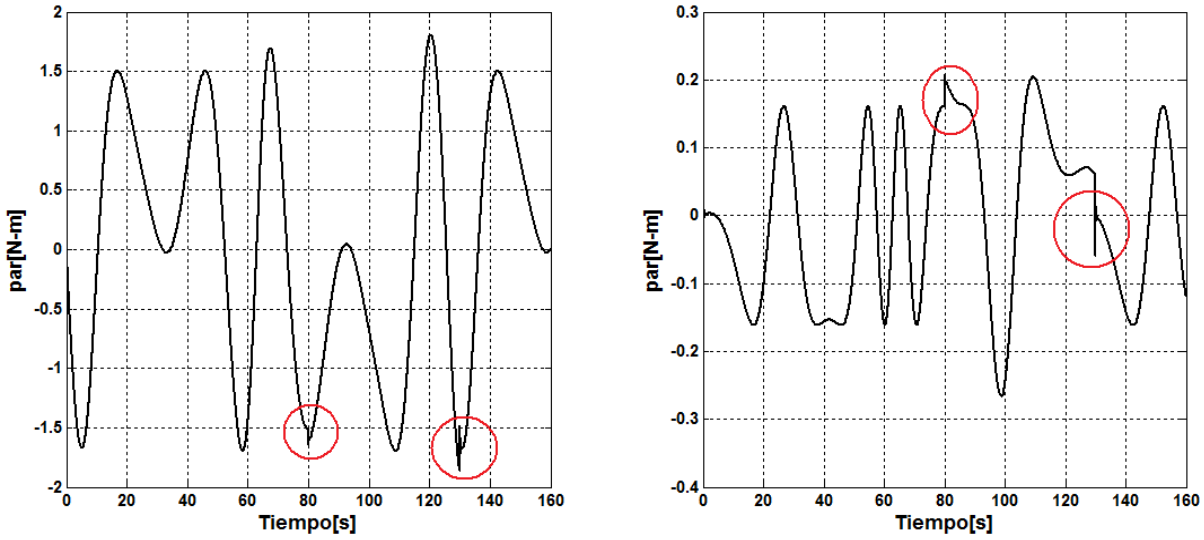


Figura 4.12.Señales de control de la prueba 2.

En la (figura 4.13) se confirma los resultados de los índices de error que presenta el controlador al cambio de masa.

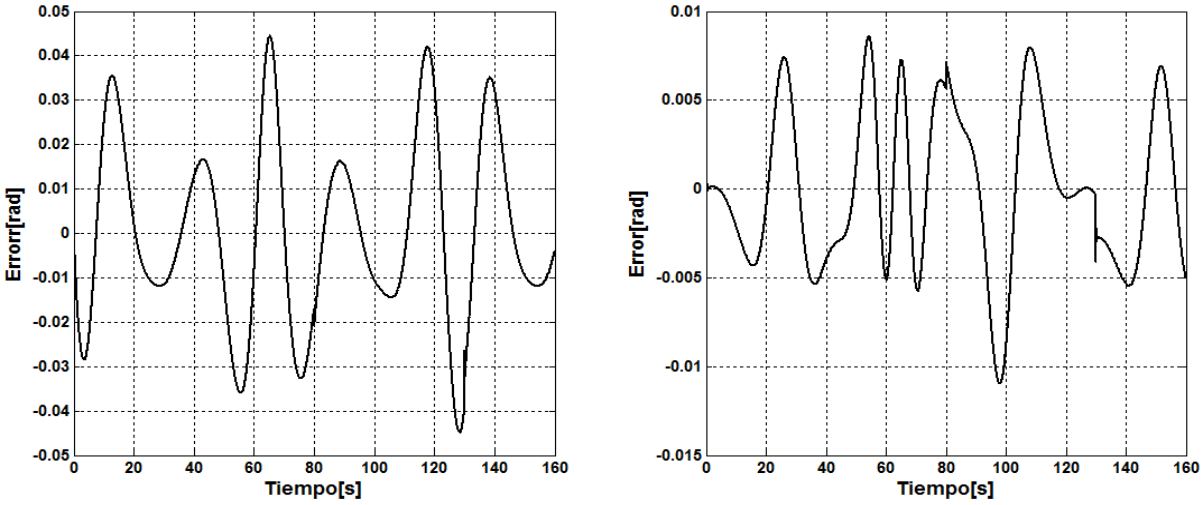


Figura 4.13.Errores de la prueba 2.

4.1.4. Controlador neuronal

A continuación se presenta las simulaciones realizadas al controlador neuronal. Esta simulación tiene el objetivo de observar la respuesta cuando no existe cambio en la

masa del manipulador y el término de fricción no se encuentra en el modelo dinámico del mecanismo.

La ley de control es

$$\tau = K_v r + \hat{\mathrm{V}}^{\mathrm{T}}\sigma\left(\hat{\mathrm{W}}^{\mathrm{T}}\mathrm{i} + \mathrm{a}\hat{\mathrm{P}}\right) - v \qquad (4.13)$$

Básicamente, la red en el controlador puede ser caracterizada por:

- Numero de neuronas en la capa oculta: 8
- Función de activación de las neuronas ocultas: $\sigma(z) = \frac{1}{1+e^{-\alpha z}}$
- Función de activación de las neuronas de salida: $\sigma(z) = z$
- Velocidad de aprendizaje de la ley de actualización: LChide=0.1, LCout=0.1
- Constante de momento= momentum=0.4
- Vector de entrada: $i = \left[\tilde{q}^T\ \dot{\tilde{q}}^T\ {q_d}^T\ {\dot{q}_d}^T\ {\ddot{q}_d}^T\ r^T\ q^T\ \dot{q}^T\right]^T$.
- Tiempo de simulación: 160 s.

Los valores de las ganancias del controlador, $K_v = \begin{bmatrix} 0.8 & 0 \\ 0 & 0.1 \end{bmatrix}$, $\Lambda = \begin{bmatrix} 12 & 0 \\ 0 & 11 \end{bmatrix}$ fueron seleccionadas. Con condiciones iniciales para los pesos sinápticos de 0.001, $K_z = -0.09,\ Z_M = 2.$

La ley de actualización de pesos son las ecuaciones (3.105), (3.106) y (3.107).

Prueba 1 del controlador neuronal

Para empezar esta simulación es necesario revelar el desempeño de un control PD sin compensación. La (figura 4.14) muestra dicho control, mientras que la tabla 4.6 se observa el índice de desempeño del error.

Tabla 4.6.Índice de desempeño del controla PD sin compensación para la prueba 1.

Controlador PD sin compensación	ISE
Eslabón1	0.1052
Eslabón2	0.1053

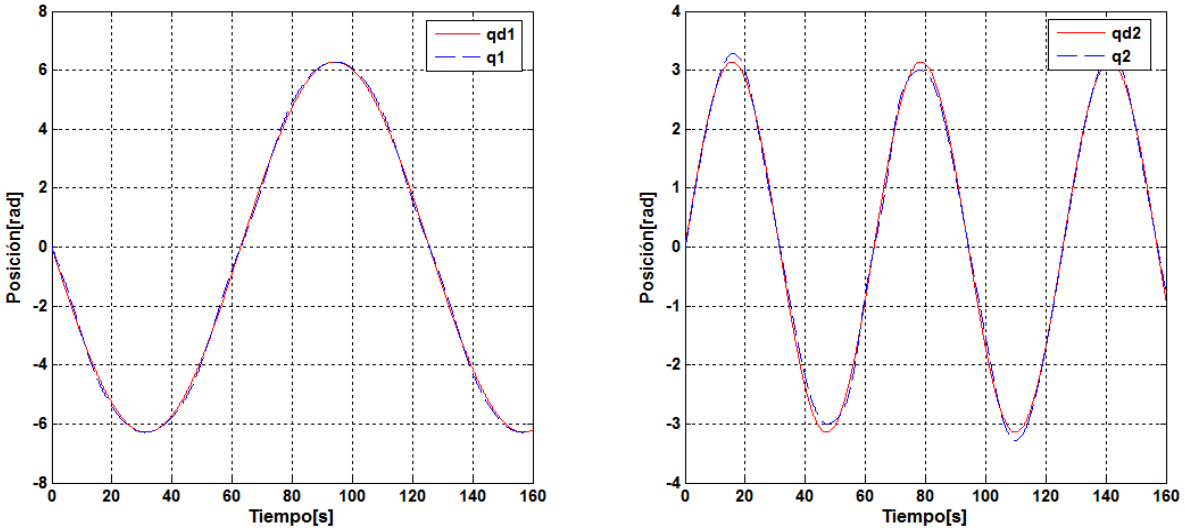


Figura 4.14.Control PD sin compensación.

A continuación se muestra lo que sucede cuando existe compensación neuronal. La tabla 4.7 muestra los valores del índice de desempeño presentado por el controlador neuronal. Se observa que existe un error mayor en el eslabón 1. Esto se debe a que la articulación mueve a dos eslabones, mientras que la articulación dos mueve a una sola masa.

Tabla 4.7.Índice de desempeño del controla neuronal para la prueba 1.

Controlador neuronal	ISE
Eslabón1	0.0134
Eslabón2	0.0043

En la (figura 4.15) se observa el comportamiento del controlador neuronal ante seguimiento de trayectoria, se nota que las posiciones articulares siguen a las trayectorias deseadas.

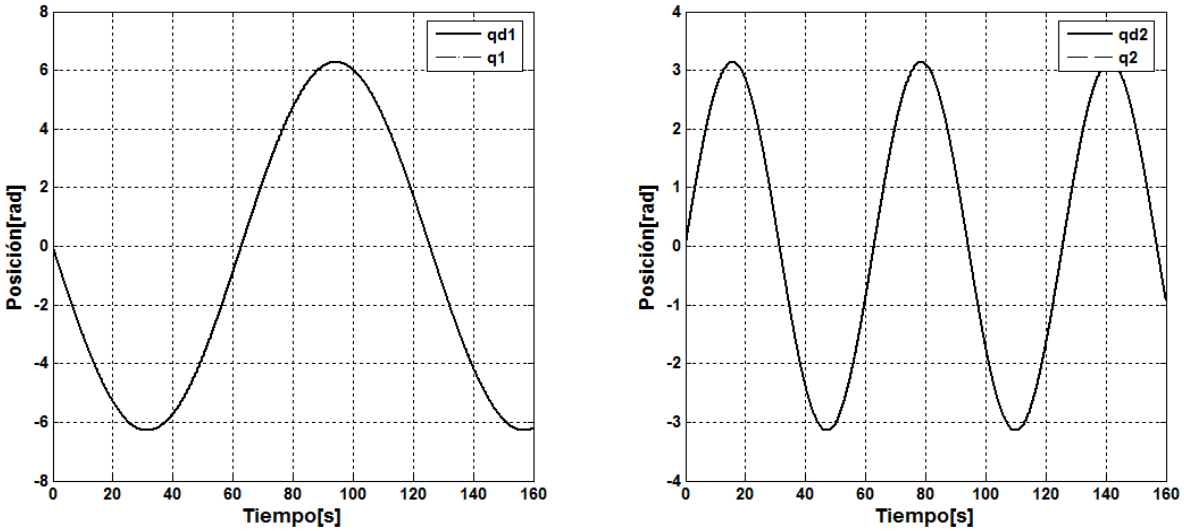


Figura 4.15. Evolución de las trayectorias de la prueba 1 del controlador neuronal.

En la siguiente figura 4.16, se muestran los pares de control que son similares a los del control PID, excepto que al inicio de la señales de control presentan un solo impulso inicial.

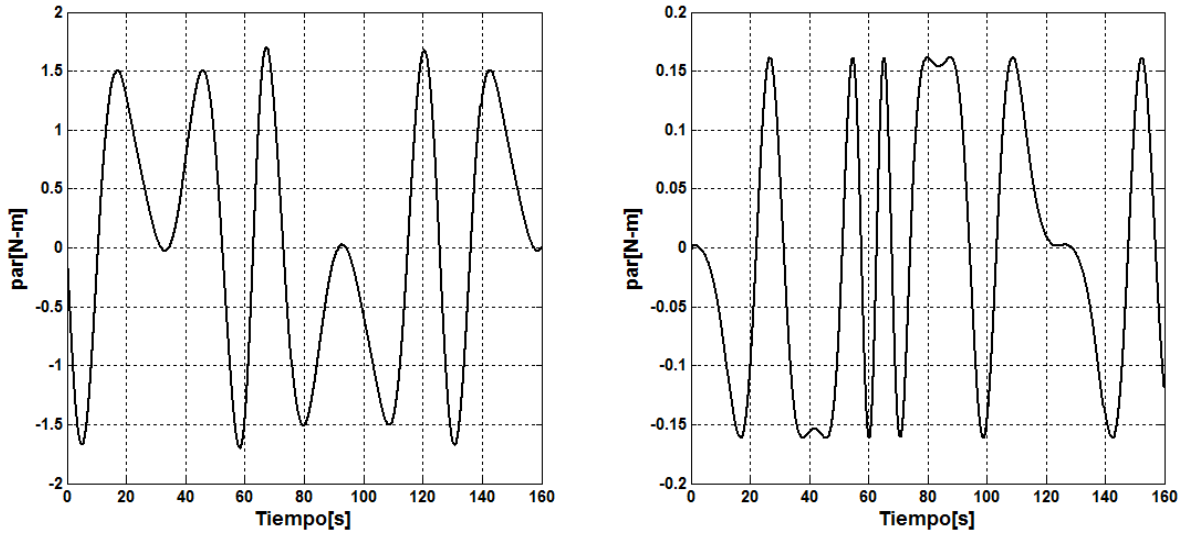


Figura 4.16.Señales de control para cada eslabón del controlador neuronal de la prueba 1.

Prueba 2 del controlador neuronal.

Como ya es conocido, esta simulación es similar a las anteriores. La (figura 4.17) muestra la evolución de las posiciones articulares uno y dos, donde se observa que el control neuronal es capaz de sobreponerse al incremento de la masa del eslabón 2, mientras que la tabla 4.8 muestra el índice producido por el sistema para diferentes incrementos de neuronas en la capa oculta.

Tabla 4.8.Índice de desempeño del control neuronal para la prueba 2.

Controlador neuronal	ISE (8 neuronas)	ISE (16 neuronas)	ISE (24 neuronas)	ISE (32 neuronas)
Eslabón1	0.0165	0.0041	0.00266	0.00203
Eslabón2	0.0068	0.00074	0.00050	0.00053

Observación: Para disminuir el error en estado estable se sugiere aumentar el número de neuronas en la capa oculta. En la tabla 4.8 se muestran varios casos donde se puede apreciar la disminución del error de seguimiento al aumentar las neuronas de la capa oculta.

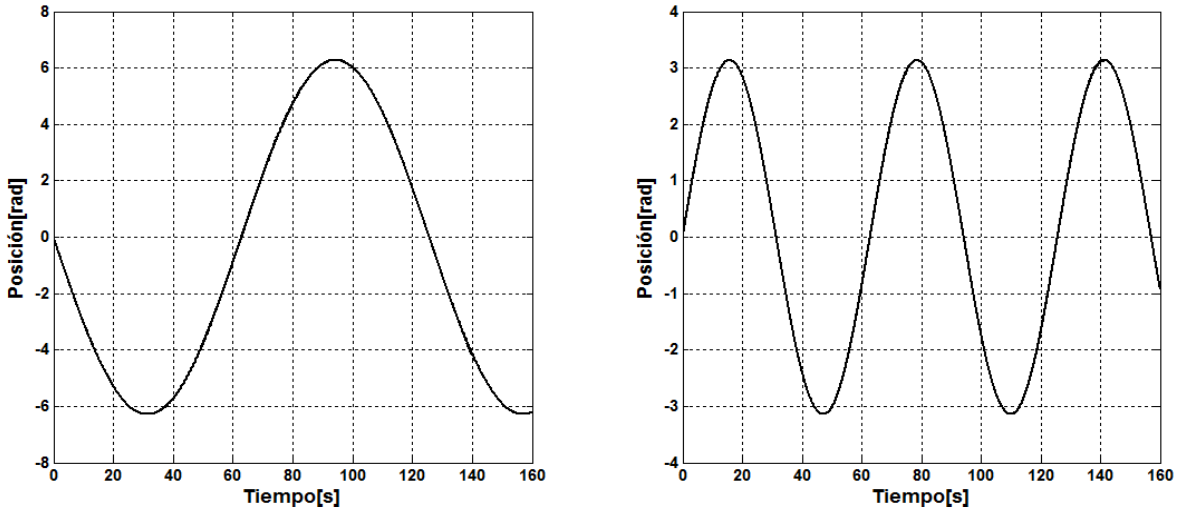


Figura 4.17.Evolucion de las posiciones articulares del control neuronal de la prueba 2.

En la (figura 4.18) se muestra que el par de control uno oscila entre $\pm 1.7\ N-m$, mientras que el par dos oscila entre $\pm 0.2\ N-m$.

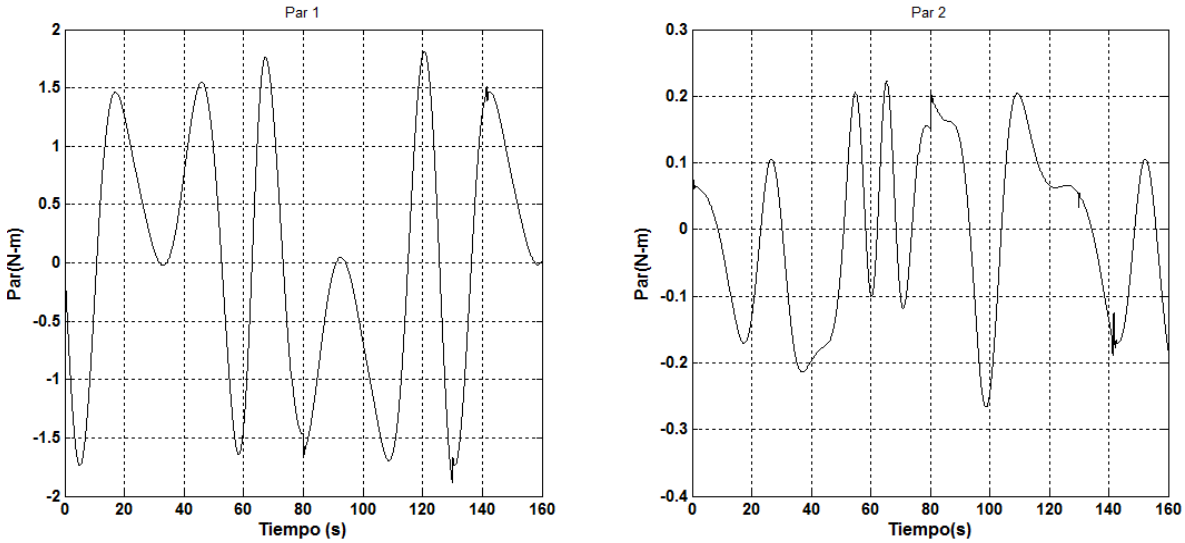


Figura 4.18. Pares de corrección del control neuronal de la prueba 2.

4.2. Resultados experimentales

En esta sección se muestran las pruebas realizadas a los esquemas de control; control PD con compensación, control PID y el controlador neuronal.

Las pruebas fueron realizadas en un robot rígido de dos grados de libertad. Los ensayos que se realizaron son; seguimiento de trayectoria y evaluación del comportamiento de los esquemas de control a través del error RMS.

La realización de pruebas se llevó a cabo mediante el uso de una herramienta de lenguaje C++ ofrecido por Builder C++ versión 6.0.

Las funciones que realiza este sistema son las siguientes:

- Se generan las variables principales necesarias para el sistema de control, como son: la velocidad angular del robot, la posición de cada eslabón, matrices de almacenamiento para los pesos de la red entre otras.
- Se genera el algoritmo de control.

4.2.1. Controlador PD con compensación

Como se mencionó anteriormente, una de las desventajas más relevantes que se genera con el esquema de control PD compensado, es que la ley de control emplea explícitamente los términos del modelo dinámico (1.1): $M(q), C(q,\dot{q})\dot{q}$ y $G(q)$. Si dichos términos no se conocen con exactitud existirá un error en estado estable. A continuación se presentan las trayectorias de los eslabones que revelan dicha desventaja.

Prueba experimental 1

Las matrices simétricas definidas positivas K_p y K_v se escogieron como:

$$K_p = \begin{bmatrix} 3 & 0 \\ 0 & 0.6 \end{bmatrix}, K_v = \begin{bmatrix} 0.29 & 0 \\ 0 & 0.1 \end{bmatrix} \; y \; \Lambda = {K_v}^{-1} K_p = \begin{bmatrix} 10.3 & 0 \\ 0 & 6 \end{bmatrix}$$

Las condiciones iniciales correspondientes a las posiciones y velocidades son:

$$q_1(0) = 0, \;\; q_2(0) = 0,$$
$$\dot{q}_1(0) = 0, \;\; \dot{q}_2(0) = 0,$$

Las trayectorias deseadas fueron las ecuaciones (4.5) y (4.6). La figura 4.19 presenta las trayectorias de los eslabones. Como era de esperarse, existe un error en estado estable en la articulación uno y dos, debido que la compensación no es exacta. Esto se puede ver mejor en la tabla 4.9.

Tabla 4.9.Índice de desempeño del control PD con compensación para la prueba experimental 1.

Controlador PD con compensación	ISE
Eslabón1	0.348
Eslabón2	0.083

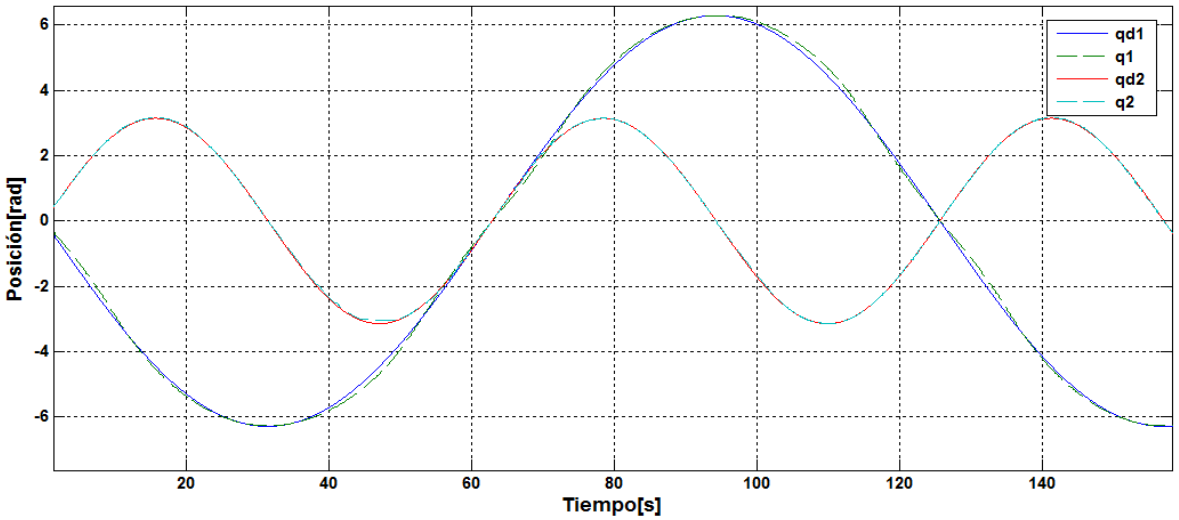


Figura 4.19.Trayectorias para la articulación 1 y 2 para el control PD con compensación de gravedad.

Las (figuras 4.20) y (4.21) se muestran las señales de control proporcionadas por el PD con compensación. Como se aprecia en la (figura 4.20) la señal de control oscila entre 3 y -3 N-m, mientras que el par aplicado a la articulación 2 oscila entre 0.2 y -0.25 N-m como se observa en la (figura 4.21).

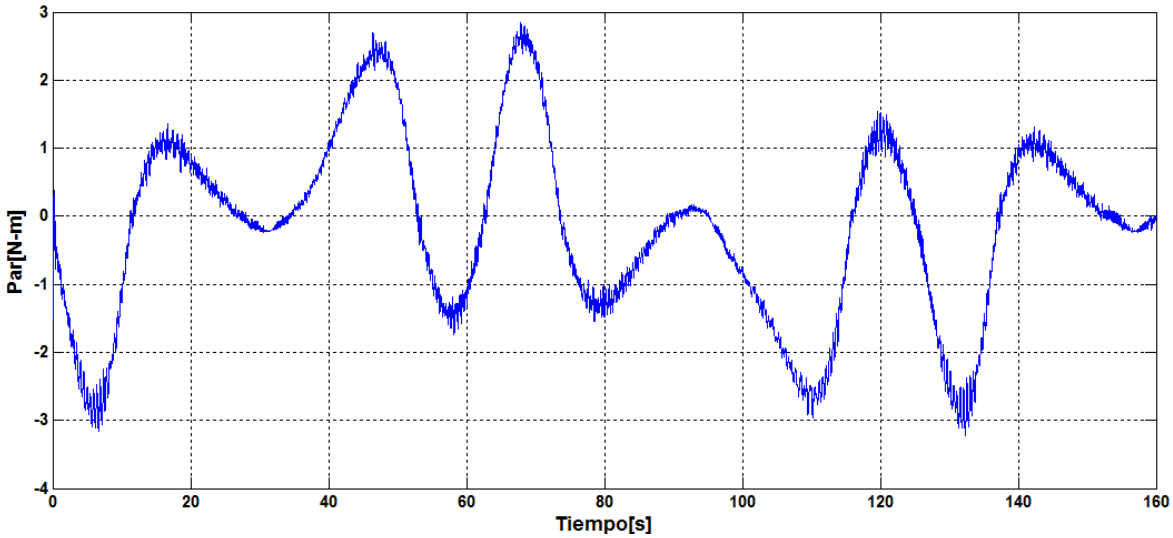


Figura 4.20.Señales de control tau1[N-m] para el control PD con compensación (prueba experimental 1).

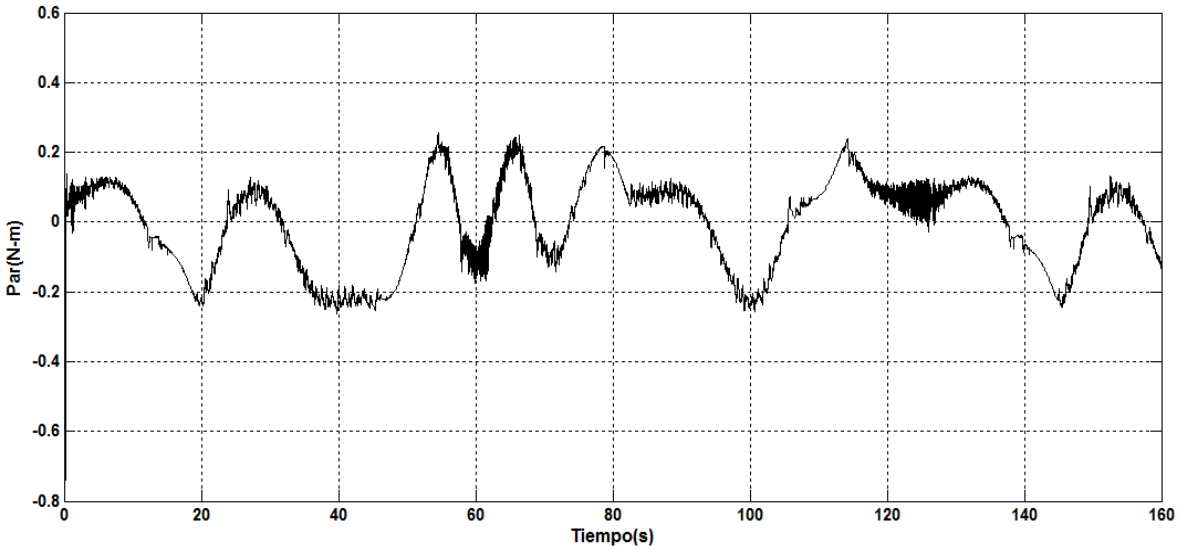


Figura 4.21.Señales de control tau2[N-m] para el control PD con compensación (prueba experimental 1).

Prueba experimental 2.

Esta prueba se considera que existe una mala estimación en la masa del eslabón 2, que su valor medido es 0.343 Kg. Se escoge un 30% de incertidumbre en la masa dos durante el intervalo de 80 a 160s, es decir una masa de 0.4459 Kg. Este efecto se hace por software ya que por precaución de no dañar los engranes del mecanismo no se coloca una masa al robot.

Las matrices simétricas definidas positivas K_p y K_v se escogieron como:

$$K_p = \begin{bmatrix} 3 & 0 \\ 0 & 0.6 \end{bmatrix}, K_v = \begin{bmatrix} 0.29 & 0 \\ 0 & 0.1 \end{bmatrix} \; y \; \Lambda = {K_v}^{-1} K_p = \begin{bmatrix} 10.3 & 0 \\ 0 & 6 \end{bmatrix}$$

Las condiciones iniciales correspondientes a las posiciones y velocidades son:

$$q_1(0) = 0, \;\; q_2(0) = 0,$$
$$\dot{q}_1(0) = 0, \;\; \dot{q}_2(0) = 0,$$

La (figura 4.22) muestra la evolución de las posiciones articulares uno y dos, mientras que la tabla 4.3 muestra el índice producido por el controlador PD con compensación.

Tabla 4.10.Índice de desempeño del controla PD con compensación para la prueba experimental 2.

Controlador PD con compensación	ISE
Eslabón1	0.3570
Eslabón2	0.102

En la figura 4.22 se observa que la compensación permite un error mayor. El efecto cuando la masa cambia se observa mejor en el eslabón 2 en el tiempo inicial de 80 s. Podemos decir que el efecto de la incertidumbre recae en la respuesta de la trayectoria de seguimiento, lo cual no sucede en el caso cuando no existe incertidumbre.

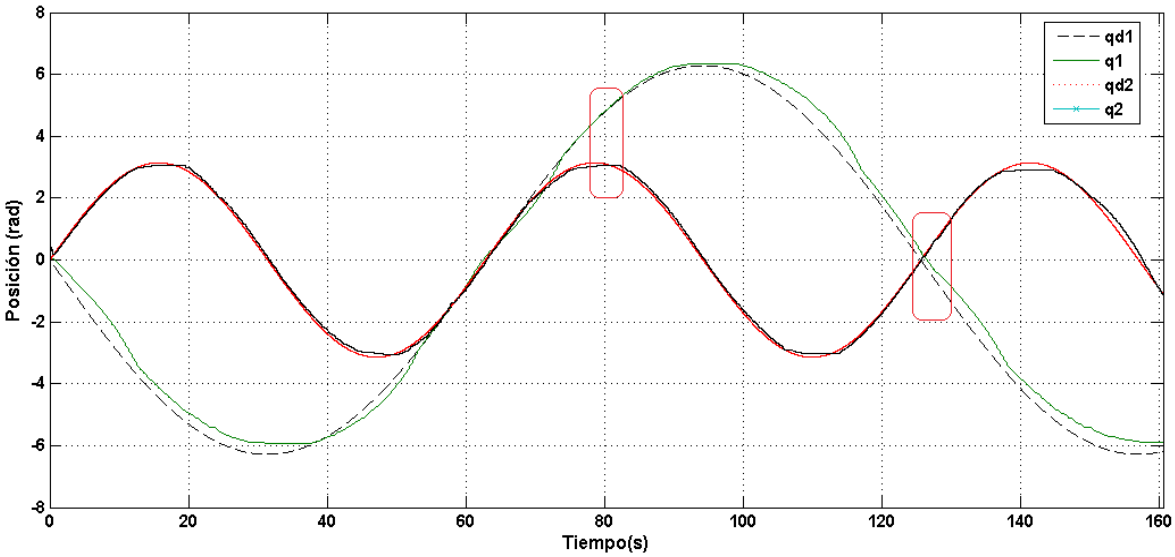


Figura 4.22.Trayectorias para la articulación 1 y 2 para el control PD con compensación con incertidumbre (prueba experimental 2).

En la (figuras 4.23) se muestra el par aplicado a la articulación uno, mientras que la figura 4.24 se muestra el par aplicado a la articulación dos.

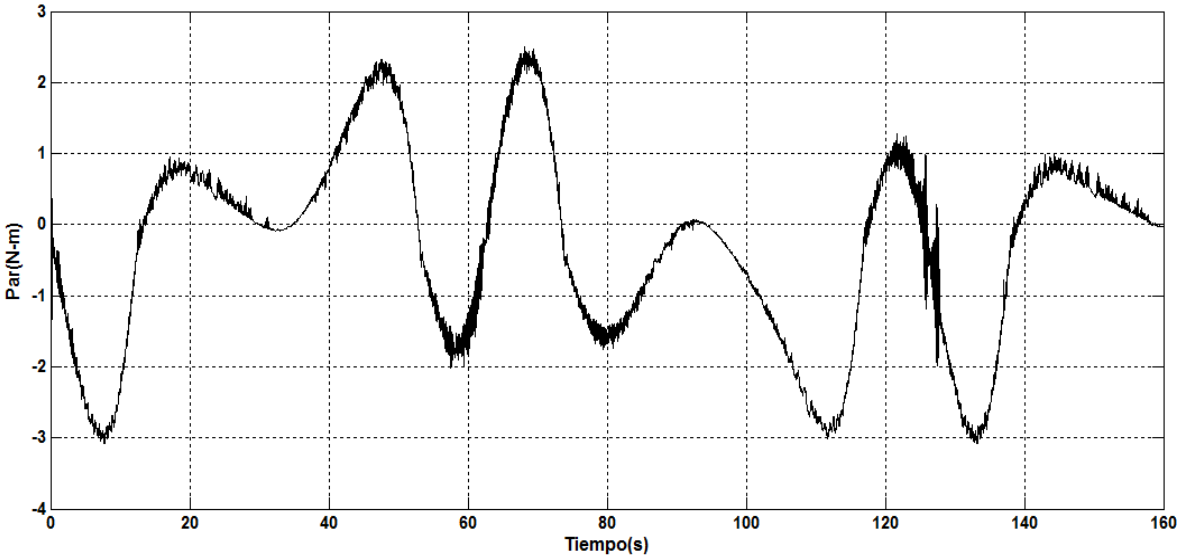


Figura 4.23.Señales de control tau1[N-m] para el control PD con compensación (prueba experimental 2).

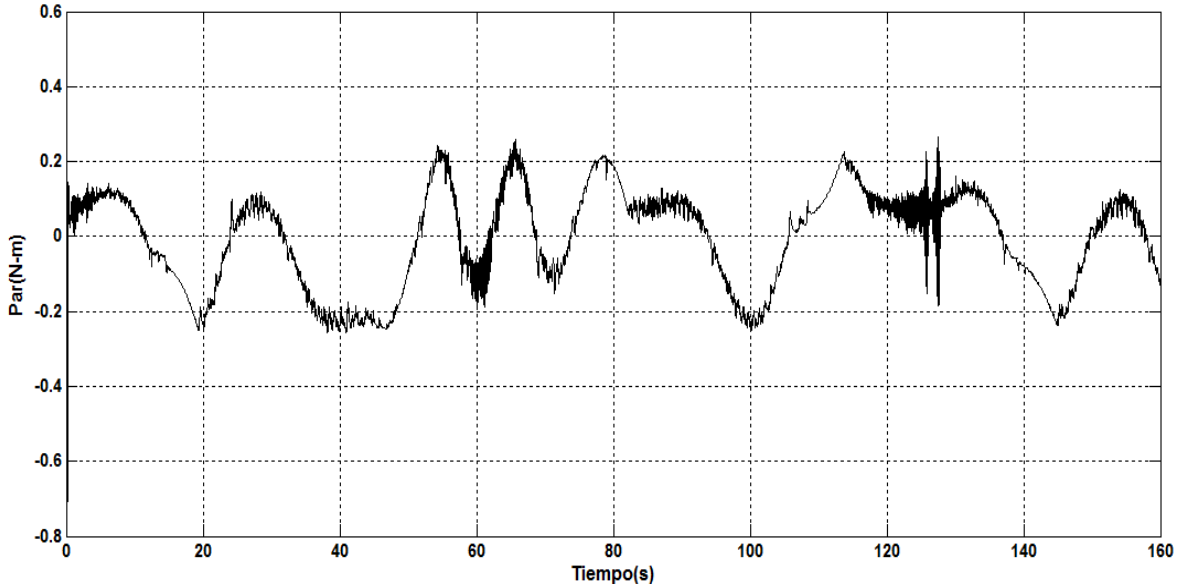


Figura 4.24.Señales de control tau2[N-m] para el control PD con compensación (prueba experimental 2).

4.2.2. Controlador PID

Como se mencionó anteriormente, una de las desventajas más relevantes que se generan con el esquema de control PID, es la respuesta transitoria. De modo que no permite realizar ajustes finos sin que estos ajustes afecten el desempeño del sistema. Esta limitante tiene inferencia sobre el seguimiento de la trayectoria de referencia.

Las matrices simétricas definidas positivas K_p, K_i y K_v se escogieron como:

$$K_p = \begin{bmatrix} 11 & 0 \\ 0 & 2 \end{bmatrix}, K_i = \begin{bmatrix} 8 & 0 \\ 0 & 1.2 \end{bmatrix} \; y \; K_v = \begin{bmatrix} 0.2 & 0 \\ 0 & 0.1 \end{bmatrix}$$

Las condiciones iniciales correspondientes a las posiciones y velocidades son:

$$q_1(0) = 0, \;\; q_2(0) = 0,$$
$$\dot{q}_1(0) = 0, \;\; \dot{q}_2(0) = 0,$$

Las trayectorias deseadas son:

$$\mathrm{q_{d1}} = -6.2832\sin(0.05\mathrm{t})$$
$$\mathrm{q_{d2}} = -3.1416\sin(0.1\mathrm{t})$$

Observación: Las ganancias finales se seleccionaron de acuerdo al comportamiento de la respuesta del robot manipulador.

Ahora bien, en la (figura 4.25) se presenta la evolución de las variables articulares obtenidas del controlador PID. En estas se observa como la trayectoria sigue a la trayectoria de referencia con un pequeño error, esto se puede ver mejor en la tabla 4.11.

Tabla 4.11.Índice de desempeño del controla PID para la prueba experimental1.

Controlador PID	ISE
Eslabón1	0.038
Eslabón2	0.031

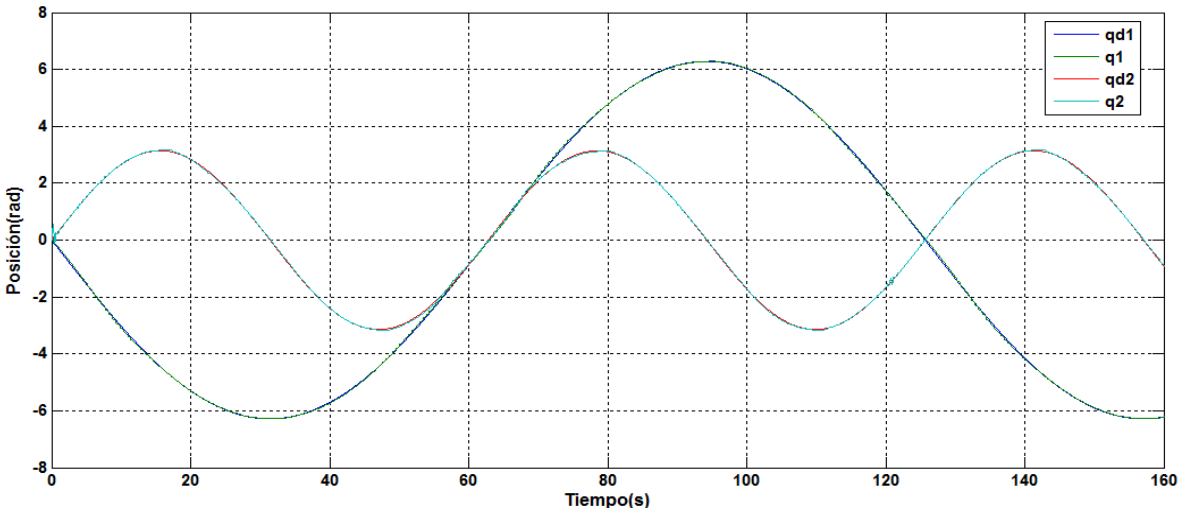


Figura 4.25.Evolucion de las trayectorias para el control PID en la prueba experimental 1.

En la (figura 4.26) y la (figura 4.27) aparecen los pares de control $tau1$ y $tau2$ aplicados a las articulaciones. Se observa que el par aplicado en la articulacion1 es mayor que el de la articulación 2, lo cual es lógico, ya que en la articulación 1 tiene que mover dos eslabones, mientras que la articulación 2 sólo tiene que mover uno.

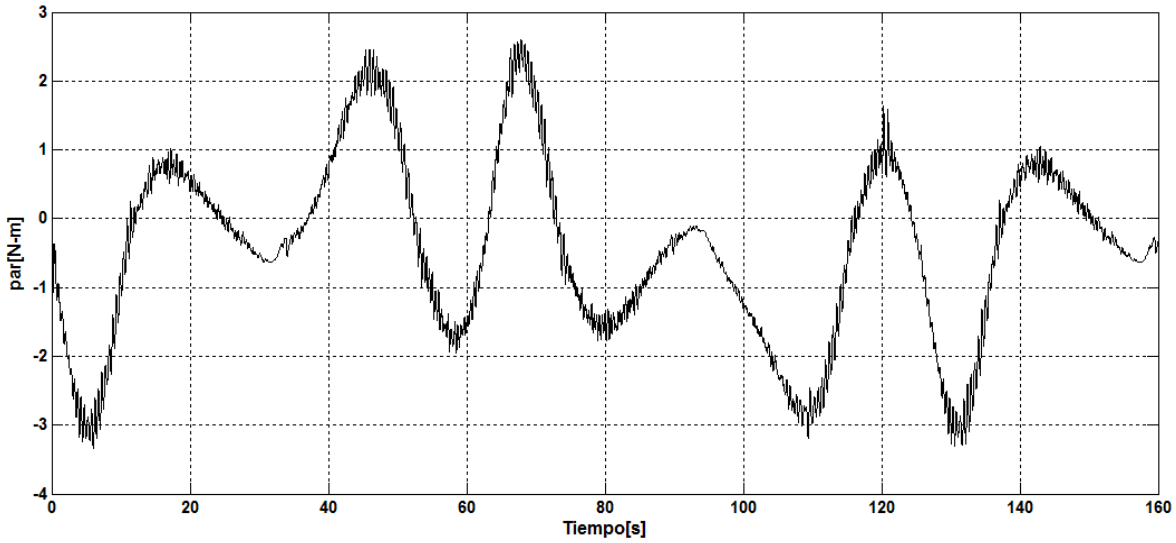


Figura 4.26.Señales de control tau1[N-m] para el control PID (prueba experimental 1).

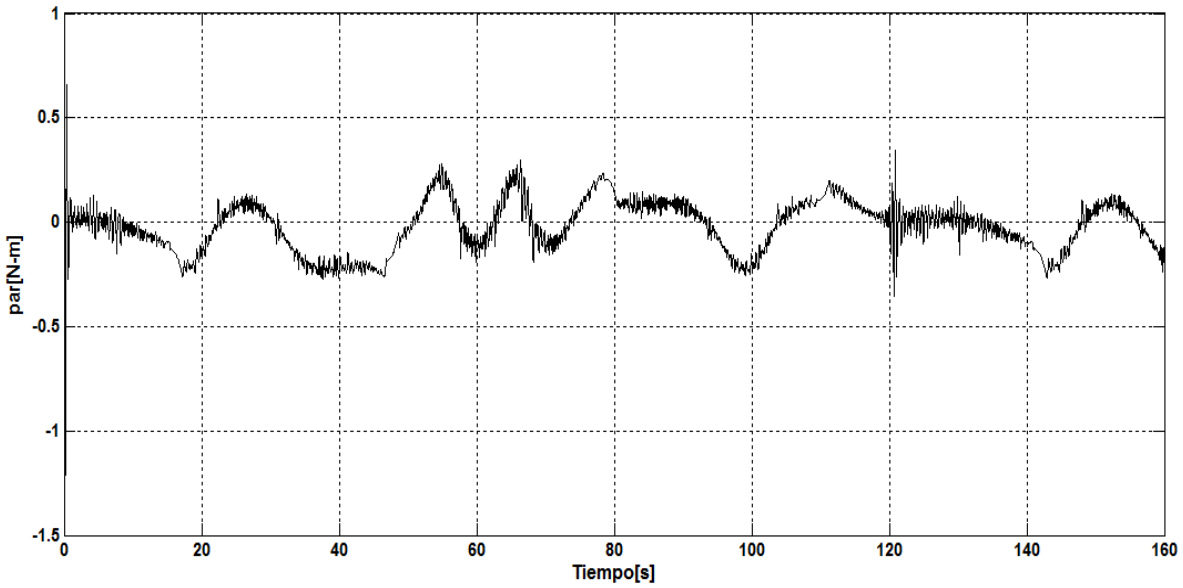


Figura 4.27.Señales de control tau2[N-m] para el control PID (prueba experimental 1).

4.2.3. Controlador neuronal

A continuación se presentan los resultados del control neuronal que se implemento para el control de seguimiento de trayectorias del mecanismo de dos grados de libertad. Aunque se realizaron varios experimentos, no todos tuvieron resultados satisfactorios, es decir, el sistema no se lograba controlar, esto se debía a que la posición de los eslabones no era exactamente la misma en todos los experimentos.

Las características de la red neuronal son:

- Numero de neuronas en la capa oculta: 8
- Función de activación de las neuronas ocultas: $\sigma(z) = \frac{1}{1+e^{-\alpha z}}$
- Función de activación de las neuronas de salida: : $\sigma(z) = \frac{1}{1+e^{-\alpha o}}$
- Velocidad de aprendizaje de la ley de actualización: LChide=0.05, LCout=0.05
- Constante de momento= momentum=0.14
- Vector de entrada: $i = \left[\tilde{q}^T\ \dot{\tilde{q}}^T\ {q_d}^T {\dot{q}_d}^T\ {\ddot{q}_d}^T\ r^T\ q^T\ \dot{q}^T\right]^T$.
- Tiempo de experimento: 160 s.

Los valores de las ganancias del controlador, $K_v = \begin{bmatrix} 0.19 & 0 \\ 0 & 0.1 \end{bmatrix}$, $\Lambda = \begin{bmatrix} 66 & 0 \\ 0 & 28 \end{bmatrix}$ fueron seleccionadas. Con condiciones iniciales para los pesos sinápticos (whide1, whide2 wout) de 0.001, $K_z = -0.012$, $Z_M = 0.044$.

A continuación se muestra el desempeño del controlador neuronal para la prueba de seguimiento de trayectoria. En la (figura 4.28) se observa un seguimiento aceptable con un ligero error como lo revela la tabla 4.12.

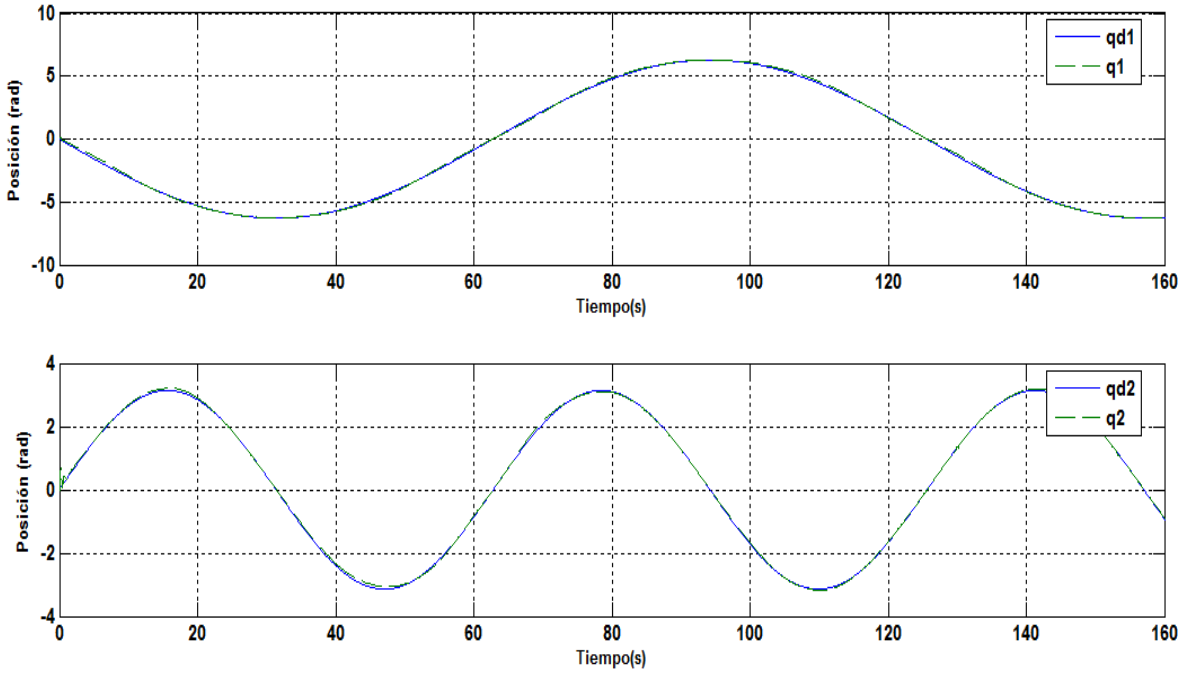


Figura 4.28.Trayectorias para la articulación 1 y 2 para el control neuronal con 8 neuronas ocultas (prueba experimental 1).

En las figuras 4.29 y 4.30 se muestras los pares aplicados a las articulaciones. Se puede ver, como era de esperarse, cuando el robot inicia, el par aplicado en la articulación 1 es mayor que el par de la articulación 2. Este valor varía entre 2.5 y -3 N-m, mientras que el par aplicado a la articulación 2 oscila entre 0.2 y -0.2N-m.

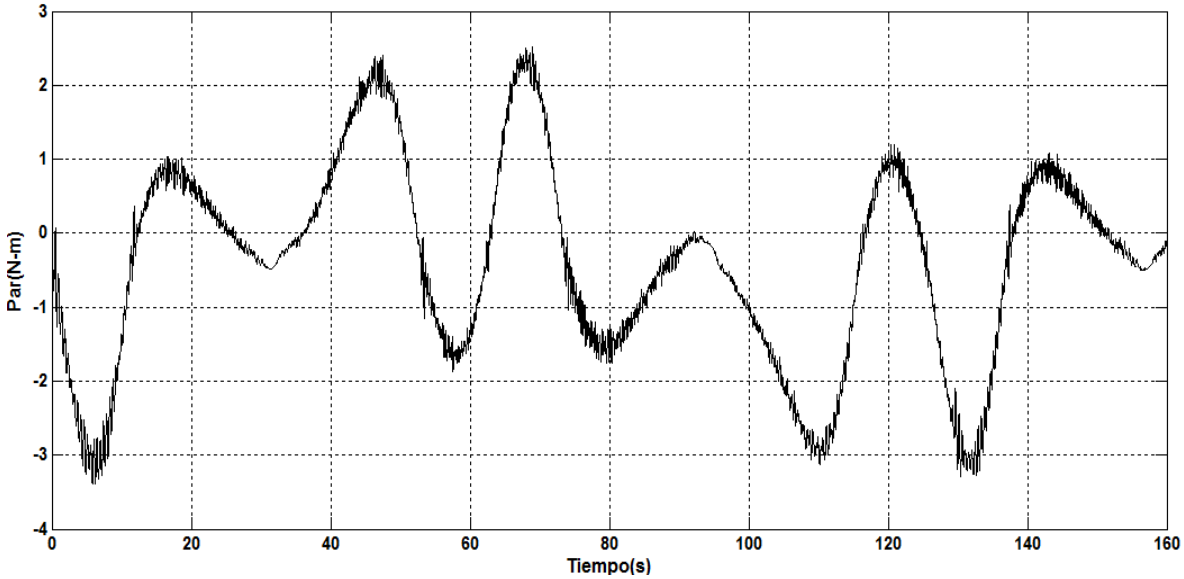


Figura 4.29: Señales tau1 [N-m] para el control neuronal con 8 neuronas ocultas (prueba experimental 1).

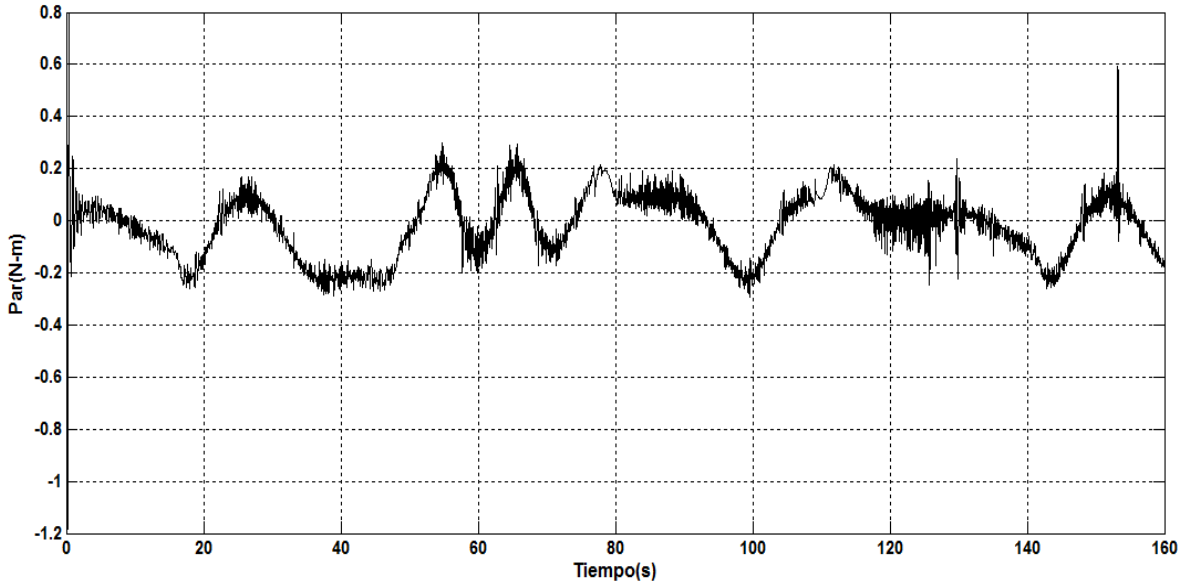


Figura 4.30.Señale tau2[N-m] para el control neuronal con 8 neuronas ocultas (prueba experimental 1).

La evolución de los errores en las articulaciones son las que se muestran en las figuras 4.31 y 4.32. Se observa que el error en estado estable no es cero como se muestra en la tabla 4.12. Los resultados ponen de manifiesto la necesidad de ajustar las ganancias del controlador y los parámetros de la red: a saber, el coeficiente de aprendizaje, la estructura de la red, el algoritmo de entrenamiento.

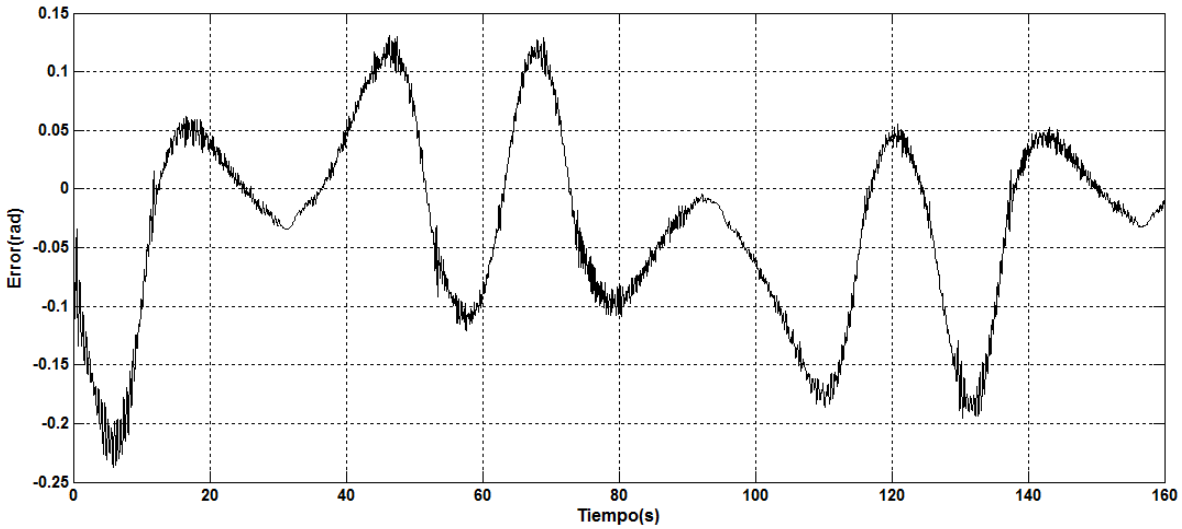


Figura 4.31. Error de posición 1 para el control neuronal con 8 neuronas ocultas (prueba experimental 1).

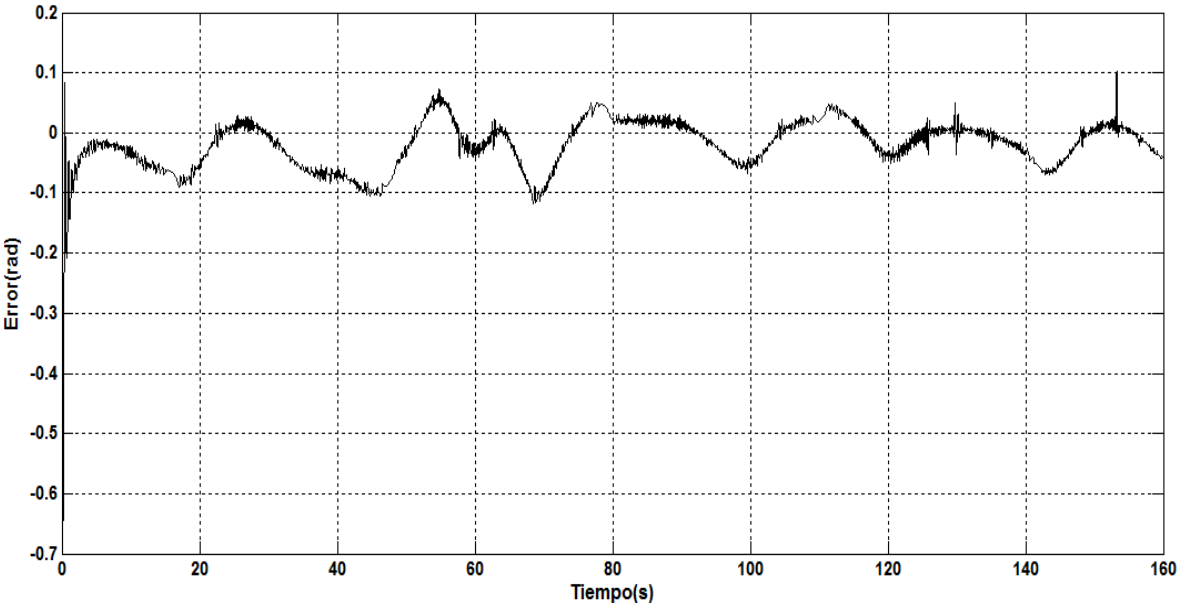


Figura 4.32. Error de posición 2 para el control neuronal con 8 neuronas ocultas (prueba experimental 1)

Tabla 4.12.Índice de desempeño del controla neuronal con 8 neuronas ocultas (prueba experimental 1).

Controlador neuronal	ISE
Eslabón1	0.084
Eslabón2	0.046

En la tabla 4.13 se aprecian los resultados para el caso de aumento de neuronas en la capa oculta.

Tabla 4.13. Índices de desempeño cuando existe un aumento de neuronas (prueba experimental 1).

Controlador neuronal	ISE (8 neuronas)	ISE (16 neuronas)	ISE (24 neuronas)
Eslabón1	0.084	0.033	0.015
Eslabón2	0.046	0.021	0.012

Basados en la tabla 4.13 se puede concluir que el error disminuye cuando el número de neuronas aumenta. De acuerdo a las repeticiones de las pruebas experimentales, se aprecia claramente como el sistema de control se ve afectado por tales aumentos de neuronas en la capa oculta. También, se observa que después de 16 neuronas el transitorio aumenta.

A continuación se muestra la compensación del controlador neuronal con 8 neuronas en la capa oculta. En la figura 4.33 se aprecia la compensación para el eslabón uno y dos en la prueba experimental 1.

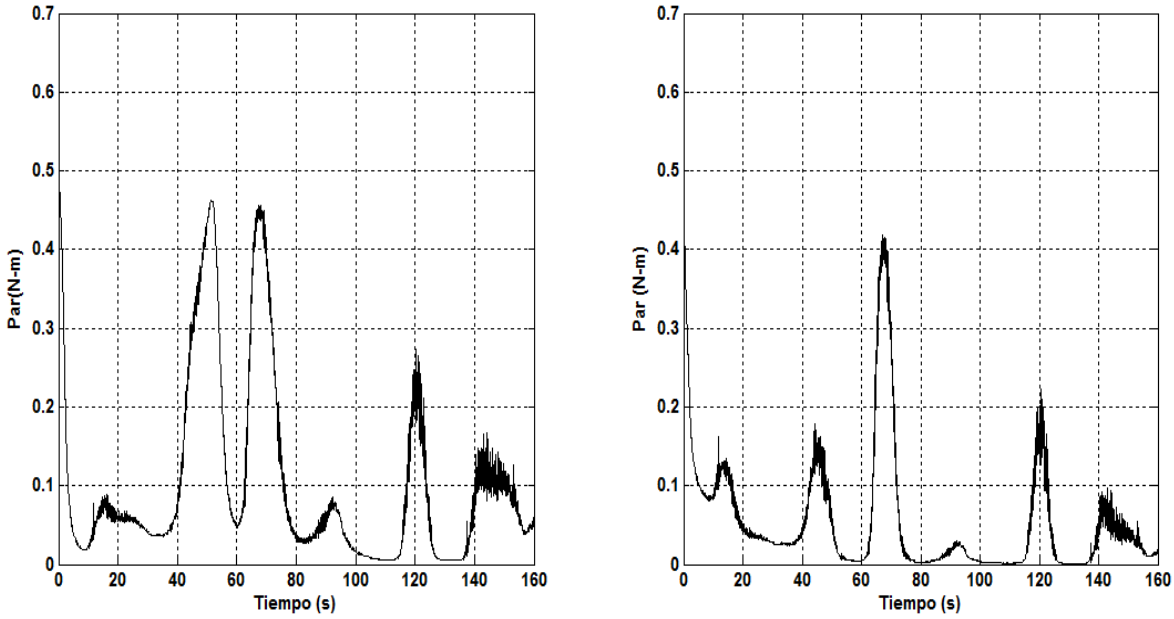


Figura 4.33. Compensación neuronal con 8 neuronas ocultas (prueba experimental 1).

A continuación se presenta la comparación entre el algoritmo normal de retropropagación y el aprendizaje recursivo. Usando la misma red neuronal, 16 entradas, 8 neuronas en la capa oculta, 2 neuronas en la capa de salida, y una velocidad de aprendizaje $\mathrm{LChide} = 0.05, \mathrm{LCout} = 0.05$ con un término de momentum=0.14. La comparación del desempeño se puede realizar por el error cuadrático ecuación (4.9). Los resultados comparados se muestran en la tabla 4.14. Se puede ver que el algoritmo recursivo es mejor con respecto al error de identificación, esto se debe a que las redes estáticas solo dependen de las entradas actuales, en ningún caso de las anteriores entradas o salidas, mientras que las redes recursivas dependen de salidas anteriores. Aunque el algoritmo de retropropagación tiene el error de seguimiento mayor es simple y fácil de implementar.

Tabla 4.14. Comparación de índices de desempeño entre el Algoritmo BP y el recursivo con 24 neuronas ocultas (prueba experimental 1).

	ISE Algoritmo BP	ISE Recursivo
Eslabón1	0.0416	0.015
Eslabón2	0.0209	0.012

V. CONCLUSIONES

En la presente tesis, se evaluó el controlador neuronal para un robot manipulador rígido de dos grados de libertad, además, se mostraron algunos resultados de los controladores PD con compensación y el control PID. En la simulación de los controladores anteriores, se apreció claramente cómo al aplicar la variación del valor de la masa del eslabón dos, ver figuras 4.6, 4.11 y 4.17, el controlador PD con compensación fue seriamente afectado. Con esto se demostró que los sistemas de control que requieren la dinámica del robot son sensibles a las variaciones de sus parámetros, en este caso, al valor de la masa. Los resultados experimentales por la ley de control PD compensado, el cual mostró un comportamiento estable si los parámetros de la dinámica no lineal del robot son cercanos a su valor real y además las ganancias del controlador PD no son muy grandes. En el caso del controlador PID no fue necesario conocer la dinámica no lineal del mecanismo solo el ajuste de las ganancias que lograron atenuar el error de seguimiento, pero con la característica que un ajuste puede reducir el error o puede empeorar la respuesta transitoria.

Los resultados experimentales del controlador neuronal cumplieron el objetivo de seguimiento de trayectoria, ver figura 4.28. En resumen, podemos concluir que la arquitectura neuronal estimó en forma correcta la dinámica no lineal del mecanismo el cual originó un error mínimo, ver tabla 4.13. Al observar que el error disminuye conforme Λ crece, se optó por aumentar este valor, además, se modificaron los parámetros K_z, factores de aprendizaje de cada capa de la red, $\mathrm{LChide}, \mathrm{LCout}$ a un valor pequeño en el cual el sistema presentó un buen desempeño, lo cual no ocurrió cuando los factores de aprendizaje son altos, con esto se logró que el sistema llegara a la inestabilidad, se entiende este fenómeno, como consecuencia de una respuesta rápida que presenta sobrepasos altos.

Podemos concluir que al usar el algoritmo recurrente se llaga a mejores aproximaciones que con el algoritmo de retropropagación comúnmente utilizado; es posible observar que la red neuronal recursiva con el algoritmo recurrente presentó un menor error cuadrático medio que la red neuronal multicapa, ver tabla 4.1, por lo tanto, el algoritmo de control neuronal aquí desarrollado constituye una alternativa para controlar sistemas con

dinámica difícil, en particular con altas no linealidades y cambios en los parámetros, lo cual podemos ver en varios procesos industriales. Finalmente, se presenta una tabla comparativa de los resultados experimentales de la prueba 1 de los tres algoritmos de control; Neuronal, PD compensado y el PID, de acuerdo al error rms o promedio como se muestra en la siguiente ecuación.

$$E_{prom} = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(q_d - q)^2}$$

Tabla 5.1. Comparación de índices de desempeño (prueba experimental 1).

Algoritmo de control	$\tilde{q}_1$	$\tilde{q}_2$
Neuronal	0.015	0.012
PD compensado	0.348	0.083
PID	0.038	0.031

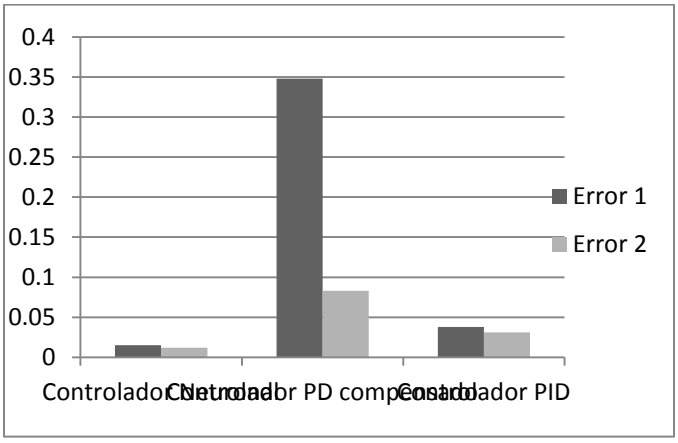


Figura 5.1. Comparación de los diferentes esquemas de control (prueba experimental 1).

Del grafico anterior, se observa que el controlador neuronal presenta el menor error promedio que los demás controladores, sin embargo, no quiere decir que los demás no puedan ser utilizados en el control de robots manipuladores.

BIBLIOGRAFÍA

Antsaklis, P. J. 1990. Neural networks in control systems. IEEE Control Systems Magazine. 3-5.

Bhat, N. V., P. A. Minderman., T. J. McAvoy., y W. N. Sun. 1990. Modeling Chemical process systems via neural computation. IEEE Control Systems Magazine. 24-29.

Craig, J.J. 2006. ROBÓTICA. (3 rd Ed). Pearson Educación, México.

Canudas-de-Wit, C., y N. Fixot. 1992. Adaptive Control of Robot Manipulators via velocity estimated feedback. IEEE Transactions on Automatic Control. 8:1234-1237.

Chen, T., and H. Chen. 1993. Approximation of continuous functionals by neural networks with applications to dynamic systems. IEEE Transactions on Neural Networks. 6:910-918.

Cybenko, G. 1989. Approximations by superpositions of a sigmoidal functions. Math of contr.sig. and Sys. 4:303-214.

Cui, X., and K.G. Shin. 1993. Direct control and coordination using neural networks. IEEE Transactions on Systems, Man and Cybernetics. 3:686-697.

Guez, A., V. Protopopsecu., y J. Barhen. 1988. On the stability, storage capacity, and design of nonlinear nontinuous neural networks. IEEE Transactions on. Systems, Man, and Cybernetics. 1:80-87.

DeWeerth, S. P., A. C. M. Nielsen., y K. J. Aström. 1991. A Simple Neuron Servo. IEEE Transactions on Neural Networks. 2:248-251.

Funahashi, K. 1989. On the approximate realization of continuous mappings by de neural networks. IEEE Transactions on Neural Networks. 183- 192.

Haykin, S. 1999. Neural networks: a comprehensive foundation (2nd Ed). Prentice Hall, New Jersey, USA.

Kim, Y.H., y F.L. Lewis. 1999. Neural network output feedback control of robot manipulators. IEEE Transactions on Robot Automatic. 2:301-309.

Kelly, R., V. Santibáñez. 2003. Control de Movimientos de Robots Manipuladores. Prentice Hall. Madrid.

Lewis, F.L., L. Kai., y A. Yesildirek. 1995. Neural net robot controller with guaranteed tracking performance. IEEE Transactions on Neural Networks. 6:703-715.

Lewis, F.L., L. Kai., y A. Yesildirek. 1996. Multilayer neural-net robot controller with guaranteed tracking performance. IEEE. Transactions on Neural Networks. 2:388-399.

Murray, R.M., L. Zexiang., y S. Shankar-Sastry. 1994. A mathematical introduction to robotic manipulation (1ª Ed). CRC Press.

Narendra, K.S., y A.M. Annaswamy. 1987. A new adaptive law for robust adaptation without persistent excitation. IEEE Transactions on Automatic Control. 2:134-145.

Narendra, K. S., y K. Parthasarathy. 1990. Identification and control of dynamical systems using neural networks. IEEE. Transacyions on Neural Networks. 1:4-26.

Narendra, K. S., y K. Parthasarathy. 1991. Gradient methods for the optimization of dynamical systems containing neural networks. IEEE Transactions on Neural Networks. 2:252-262.

Nguyen, D. H., y B. Widrow. 1990. Neural networks for self- learning control systems. IEEE Control Systems Magazine.18-23.

Ollero, B. A. 2001. ROBÓTICA manipuladores y robots móviles (1st Ed). Marcombo. Barcelona.

Ortega, R., y M.W. Spong. 1989. Adaptive motion control of rigid robot: A Tutorial. Automática. 6:887-888.

Pearlmutter, B. A. 1995. Gradient calculations for dynamic recurrent neural networks: A survey. IEEE Transactions on. Neural Networks. 5:1212-1227.

Psaltis, D. S., y A. A. Yamamura. 1988. A multilayered neural network controller. IEEE Control Systems Magazine. 17-21.

Qin, Si-Zhao., Hong-Te Su., y T. J. McAvoy. 1992. Comparison of four neural net learning methods for dynamic system identification. IEEE Trans. Neural Networks. 1:122-130.

Ruiz, V., y C. Torras. 1995. On-line learning with minimal degradation in feedforward networks. IEEE Transactions on Neural Networks. 3:657-668.

Spong, M.W., y M. Vidyasgar. 1989. Robot dynamics and control. Wiley. New York.

Spong, M.W. 1992. On the robust control of robot manipulators. IEEE Transactions on Automatic Control. 11:1782-1787.

Stevens, C.F. 1979. The neuron. Scientific American, 3:54-65.

Simpson, P.K. 1990. Artificial neural system: Foundation, paradigms, applications and implementations. Pergamon Press, New York, USA.

Sanchez, E. N., A.Y. Alanis., y J. Rico. 2004. Electric load demand prediction using neural networks trained by kalman filtering. Proceedings of the International Joint Conference on Neural Networks. Budapest, Hungria.

Weerasooriya, Siri., y M. A. El-Sharkawi. 1991. Identification and control of a DC motor using back-propagation neural networks. IEEE Transactions on Energy Conversion. 4:663-669.

Werbos P.J. 1974. Beyond regression:new tools for predicition and analysis in the behavioral sciences, Ph. D. Thesis. Harvard University, Cambridge, MA.

APÉNDICES

A. Prototipo experimental

En la (figura A.1) se muestra la estructura mecánica del brazo utilizado en esta tesis donde se aprecian los dos grados de libertad.

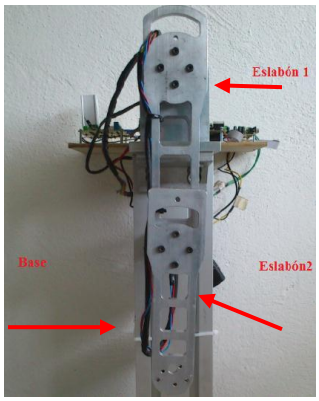


Figura A.1.Prototipo experimental del robot manipulador de dos grados de libertad.

En la tabla A.1 se presentan los parámetros del brazo articulado.

Tabla C.1 Parámetros del mecanismo.

Variable	Significado	Valor
$\boldsymbol{m_1}$	Masa del eslabón 1	0.763 Kg
$\boldsymbol{m_2}$	Masa del eslabón 2	0.343 Kg
$\boldsymbol{l_1}$	Longitud del eslabón 1	0.193 m
$\boldsymbol{l_2}$	Longitud del eslabón 2	0.145 m
$\boldsymbol{l_{c1}}$	Distancia al centro de masa del eslabón 1	0.136 m
$\boldsymbol{l_{c2}}$	Distancia al centro de masa del eslabón 2	0.048 m
$\boldsymbol{I_1}$	Momento de inercia del eslabón 1	0.0186 $Kg\ m^2$
$\boldsymbol{I_2}$	Momento de inercia del eslabón 2	0.0026987 $Kg\ m^2$
G	Gravedad	9.81 m/s^2

Cada articulación del robot prototipo se controla por un motor de CD. Este tipo de motor es utilizado en el control de manipuladores debido a sus características dinámicas. Los motores que controlan cada articulación son de la marca Pittman y están acoplados a una caja de engranes y un codificador de posición (encoder). Para el eslabón 1 se utilizó el

GM14904S014 y para el eslabón 2 fue el GM8724S015. En la figura A.2 se muestran los motores comerciales que se utilizaron en cada articulación del mecanismo.

a) b)

Figura A.2. Motores Pittman a) GM14904S014. b) GM8724S015.

El brazo articulado cuenta con una tarjeta de: adquisición de datos, señales de control y una de potencia.

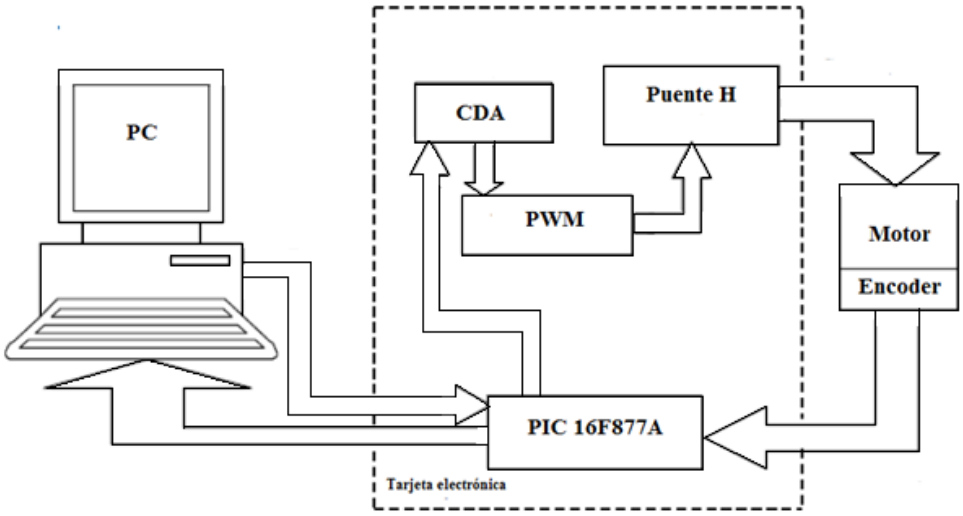


Figura A.3. Esquema general para el control del robot manipulador.

Etapa de adquisición de datos

La localización y el sentido de giro de cada eje son medidos por un encoder óptico incremental acoplado a la flecha del motor. A través de la generación de pulsos de los encoders se determinan tanto la posición como el sentido de giro de cada eslabón. El

microcontrolador PIC16F877A de Microchip detecta los cambios de posición de dichos encoders para después transmitir los datos a la computadora por medio de la comunicación serial. El circuito MAX232 utilizado para enviar los datos a la computadora en forma serial a 115200 baudios por segundo.

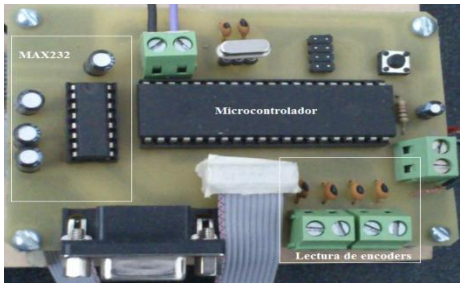


Figura A.4. Tarjeta de adquisición de datos.

Señales de control

En esta etapa la computadora manda la señal de control al microcontrolador en forma de bits, después, el PIC la obtiene y la convierte en una palabra de 8 bits que se manda a un circuito integrado DAC0800, uno por cada motor, dicho dispositivo es un convertidor digital analógico. Al ser convertida la señal en forma analógica se debe conectar un primer amplificador. La señal de salida de dicho amplificador está en un rango de -1 a 1 volts para la articulación 1 y -10 a 10 volts para la articulación dos. Enseguida se compara la señal obtenida con la señal derivada mediante una resistencia sensora colocada en serie con el motor. Para mejores resultados se coloco un filtro RC después del amplificador que realiza la diferenciación de potencial de la resistencia sensora. Éste a su vez se toma como referencia para que un generador analógico de PWM entregue la señal al manejador de potencia (el puente H), ver figura A.5.

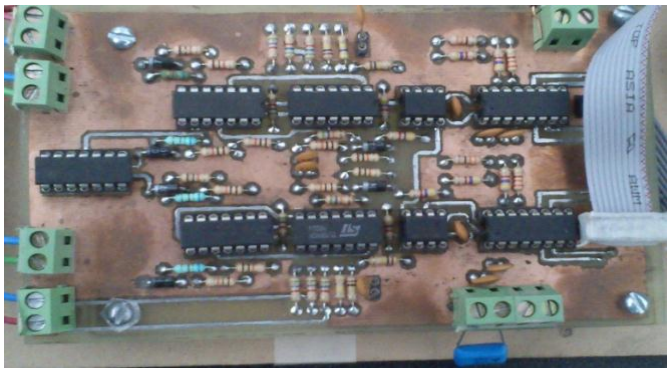


Figura A.5. Tarjeta para las señales de control.

Etapa de potencia

Como entradas a esta etapa se requieren señales de sentido de giro (expresadas en forma de PWM) y otra para habilitar el dispositivo (Puente H L298). Las señales C y D se encargan de manejar el sentido de giro del motor, mientras que la señal E se utiliza para habilitar el dispositivo. El L298 es driver de potencia básicamente sirve para amplificar la corriente para el motor.

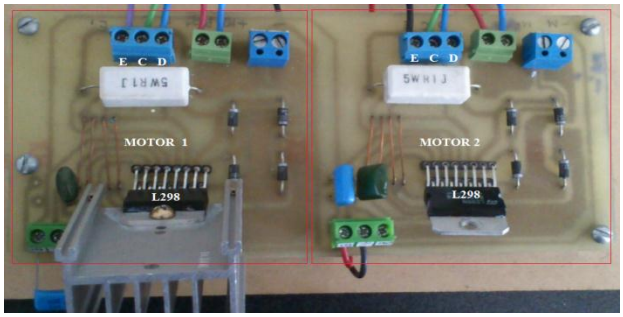
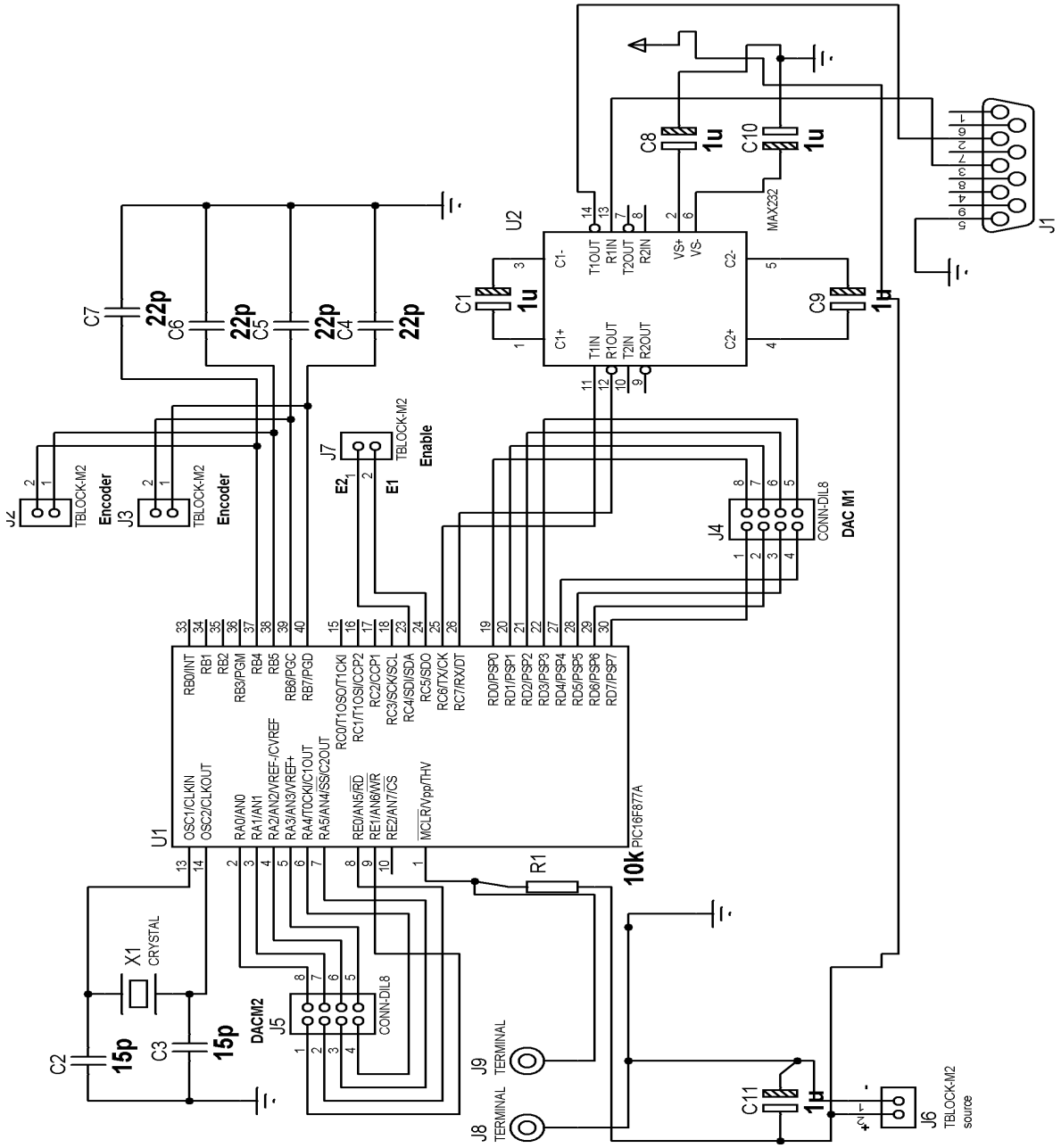


Figura A.6. Tarjeta etapa de potencia.

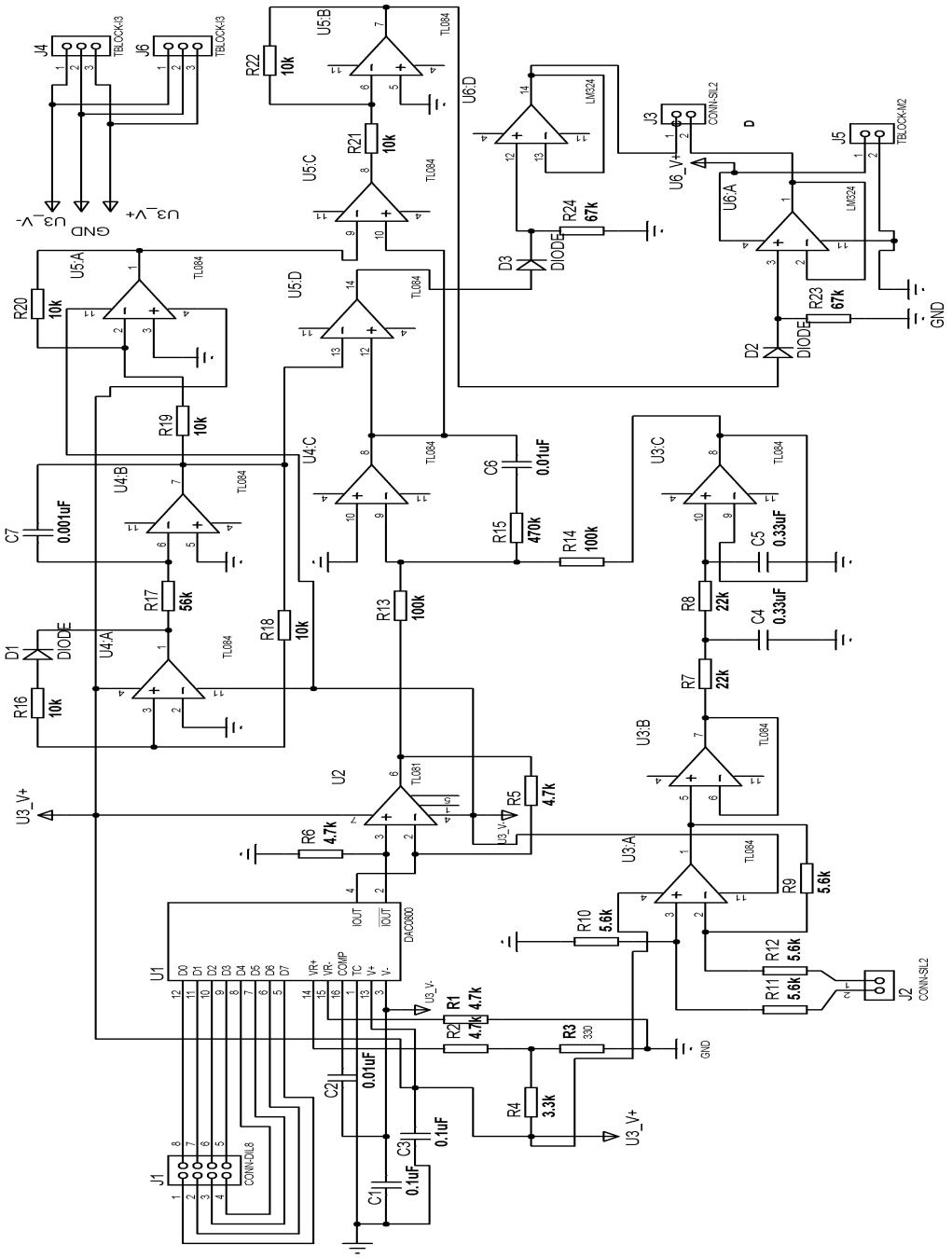
B. Diagramas electrónicos del proyecto.

B.1 Adquisición de datos

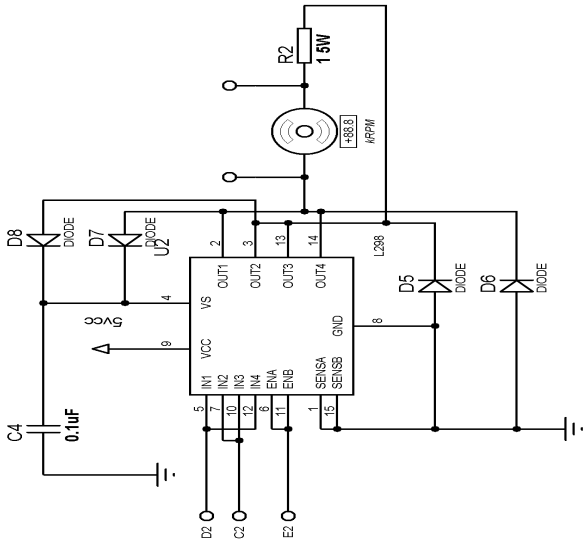


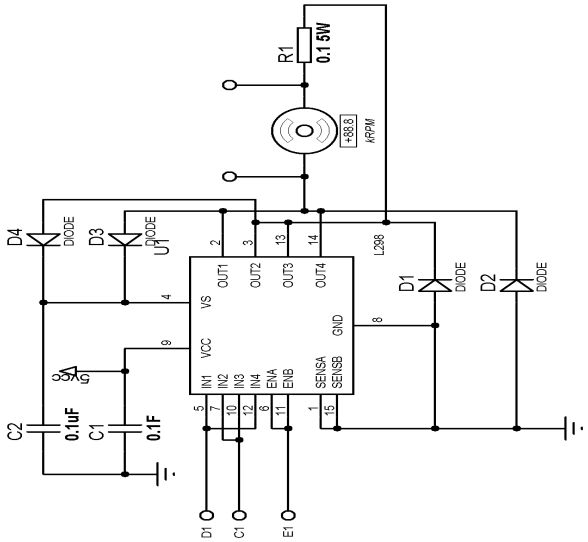
B.2. Señales de control.

En la siguiente figura se muestra el diagrama electrónico para la generar las señales de control. Para cada eslabón se realizó una tarjeta de control y potencia, con la única diferencia que para la tarjeta de control del eslabón dos, R4=1k y R3=4.7k.



B.3 Etapa de potencia.





C. Código de programas

Para la simulación de los diferentes esquemas de control se utilizo el programa Dev C++ 4.9.9.2.

Código para el PD compensado

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include <conio.h>
#define time_step  0.01
#define T          160
**** Definición las constantes utilizadas durante el programa ****
#define kp1     4
#define kp2     1
#define kd1     0.4
#define kd2     0.1
#define lam1    10
#define lam2    10
//**** ***************************************
//**** P R O G R A M A P R I N C I P A L ****
//**** ***************************************
double t;
```

```
double q1=0;              /*posicion 1 del eslabon 1*/
double q2=0;              /*posicion 2 del eslabon 2*/
double dq1=0;             /*vel 1 del eslabon 1*/
double dq2=0;             /*vel 2 del eslabon 2*/
double tau1=0;
double tau2=0;
double qd1=0.0,qd2=0.0,qdp1=0.0,qdp2=0.0,qdpp1=0.0,qdpp2=0.0;
double qtilde1=0,qtilde2=0,qptilde1=0.0,qptilde2=0.0;
double m11=0,m12=0,m21=0,m22=0,c11=0,c21=0,c12=0,g1=0,g2=0,det,v1=0,v2=0,f2=0,f1=0,f3=0,f4=0;
double aux1=0,aux2=0,aux3=0,aux4=0;
double amp1=-6.2832,amp2=3.1416;
double fact2=0.1;
double fact1=0.05;
double L1=  0.193;              //longitud de eslabón 1
double L2=  0.145;              //longitud de eslabón 2
double lc1= 0.136;              //distancia al centro de masa 1
double lc2= 0.048;              //distancia al centro de masa 2
double m1=  0.763;              //masa 1
double m2=  0.343;              //masa 2
double I1=  0.0186;             //inercia eslabón 1
double I2=  0.0026987;          //inercia eslabón 2
double g=   9.81;               //aceleración de gravedad
double A=m1*(lc1*lc1)+m2*((L1*L1)+(lc2*lc2))+I1+I2;
double B=m2*L1*lc2;
double C=m2*(lc2*lc2)+I2;
double H=m1*g*lc1+m2*g*L1;
double F=m2*g*lc2;
double G=m2*L1*lc2;
//*************************************************************************************************/
 main()
{/*::::::::::::::::::::::::::::::Principio de main::::::::::::::::::::::::::::::::::::::::::::::*/
     FILE *frobot2xx;        /* Archivo de datos  */
     frobot2xx=fopen("Par_calculado.txt","w");
 /*****************************************************************************************
do
{
     qd1=amp1*sin(fact1*t);
     qd2=amp2*sin(fact2*t);
     qdp1=amp1*fact1*cos(fact1*t);
     qdp2=amp2*fact2*cos(fact2*t);
     qdpp1=-amp1*fact1*fact1*sin(fact1*t);
     qdpp2=-amp2*fact2*fact2*sin(fact2*t);
     qtilde1=qd1-q1;
     qtilde2=qd2-q2;
     qptilde1=qdp1-dq1;
     qptilde2=qdp2-dq2;
     aux1=qdpp1+lam1*qptilde1;
     aux2=qdpp2+lam2*qptilde2;
     aux3=qdp1+lam1*qtilde1;
     aux4=qdp2+lam2*qtilde2;
     tau1=kp1*qtilde1+kd1*qptilde1+(m11*aux1+m12*aux2)+c11*aux3+c12*aux4+g1;
     tau2=kp2*qtilde2+kd2*qptilde2+(m21*aux1+m22*aux2)+c21*aux3+g2;
     /**********************modelo dinamico del robot de 2 grados de libertad******/
     if(t>80&&t<160){
     m2=(0.343*.3)+0.343;
     A=m1*(lc1*lc1)+m2*((L1*L1)+(lc2*lc2))+I1+I2;
     B=m2*L1*lc2;
     C=m2*(lc2*lc2)+I2;
     H=m1*g*lc1+m2*g*L1;
     F=m2*g*lc2;
     G=m2*L1*lc2;
     m11=A+2*B*cos(q2);
```

```
    m12=C+B*cos(q2);
    m21=m12;
    m22=C;
    c11=-B*sin(q2)*dq2;
    c12=-B*sin(q2)*(dq1+dq2);
    c21=B*sin(q2)*dq1;
    g1=H*sin(q1)+F*sin(q1+q2);
    g2=F*sin(q1+q2);
    det=m11*m22-m12*m21;
    v1= tau1-c11*dq1-c12*dq2-g1-0.2*dq1;
    v2= tau2-c21*dq1-g2-0.2*dq2;
    f1=dq1;
    f2=dq2;
    f3=((m22*v1)-(m12*v2))/det;
    f4=((-m12*v1)+(m11*v2))/det;
    q1=f1*time_step+q1;
    q2=f2*time_step+q2;
    dq1=f3*time_step+dq1;
    dq2=f4*time_step+dq2;
              }
         else{
    m2=0.343;
    A=m1*(lc1*lc1)+m2*((L1*L1)+(lc2*lc2))+I1+I2;
    B=m2*L1*lc2;
    C=m2*(lc2*lc2)+I2;
    H=m1*g*lc1+m2*g*L1;
    F=m2*g*lc2;
    G=m2*L1*lc2;
    m11=A+2*B*cos(q2);
    m12=C+B*cos(q2);
    m21=m12;
    m22=C;
    c11=-B*sin(q2)*dq2;
    c12=-B*sin(q2)*(dq1+dq2);
    c21=B*sin(q2)*dq1;
    g1=H*sin(q1)+F*sin(q1+q2);
    g2=F*sin(q1+q2);
    det=m11*m22-m12*m21;
    v1= tau1-c11*dq1-c12*dq2-g1;
    v2= tau2-c21*dq1-g2;
          }
    f1=dq1;
    f2=dq2;
    f3=((m22*v1)-(m12*v2))/det;
    f4=((-m12*v1)+(m11*v2))/det;
    q1=f1*time_step+q1;
    q2=f2*time_step+q2;
    dq1=f3*time_step+dq1;
    dq2=f4*time_step+dq2;
  /*****************************************************************************/
    fprintf(frobot2xx,"%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\n",t,qd1,q1,qd2,q2,qtilde1,qtilde2,tau1,tau2);
    t=t+time_step;
   }while(t<=T);
   fclose(frobot2xx);
}/*::::::::::::::::::::::::::::::::::Fin de main::::::::::::::::::::::::::::::::::::::::::::::::*/
```

Código para el PID

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
```

```
#include <math.h>
#include <time.h>
#include <conio.h>
#define time_step  0.01
#define T          160
/*************************************************************************************************
#define kp1     30
#define kp2     25
#define kd1     .8
#define kd2     .45
#define ki1       9
#define ki2       2
//**** ***************************************
//**** P R O G R A M A P R I N C I P A L ****
//**** ***************************************
double t;
double q1=0;              /*posicion 1 del eslabon 1*/
double q2=0;              /*posicion 2 del eslabon 2*/
double dq1=0;              /*vel 1 del eslabon 1*/
double dq2=0;              /*vel 2 del eslabon 2*/
double tau1=0;
double tau2=0;
double qd1=0.0,qd2=0.0,qdp1=0.0,qdp2=0.0,qdpp1=0.0,qdpp2=0.0;
double qtilde1=0,qtilde2=0,qptilde1=0.0,qptilde2=0.0;
double m11=0,m12=0,m21=0,m22=0,c11=0,c21=0,c12=0,g1=0,g2=0,det,v1=0,v2=0,f2=0,f1=0,f3=0,f4=0;
double aux1=0,aux2=0,inte1=0,inte2=0,area1=0,area2=0;
double amp1=-6.2832,amp2=3.1416;
double fact2=0.1;
double fact1=0.05;
double L1=  0.193;                //longitud de eslabón 1
double L2=  0.145;                //longitud de eslabón 2
double lc1= 0.136;                //distancia al centro de masa 1
double lc2= 0.048;                //distancia al centro de masa 2
double m1=  0.763;                //masa 1
double m2=  0.343;                //masa 2
double I1=  0.0186;               //inercia eslabón 1
double I2=  0.0026987;            //inercia eslabón 2
double g=   9.81;                 //aceleración de gravedad
double A=m1*(lc1*lc1)+m2*((L1*L1)+(lc2*lc2))+I1+I2;
double B=m2*L1*lc2;
double C=m2*(lc2*lc2)+I2;
double H=m1*g*lc1+m2*g*L1;
double F=m2*g*lc2;
double G=m2*L1*lc2;
//*************************************************************************************************
 main()
{/*::::::::::::::::::::::::::::::Principio de main::::::::::::::::::::::::::::::::::::::::::::::*/
     FILE *frobot2xx;        /* Archivo de datos  */
     frobot2xx=fopen("ControlPID.txt","w");
/* inicio
/*************************************************************************************************
do
{
     qd1=amp1*sin(fact1*t);
     qd2=amp2*sin(fact2*t);
     qdp1=amp1*fact1*cos(fact1*t);
     qdp2=amp2*fact2*cos(fact2*t);
     qdpp1=-amp1*fact1*fact1*sin(fact1*t);
     qdpp2=-amp2*fact2*fact2*sin(fact2*t);
     qtilde1=qd1-q1;
     qtilde2=qd2-q2;
     qptilde1=qdp1-dq1;
```

```
qptilde2=qdp2-dq2;
area1=qtilde1*time_step;
inte1=inte1+area1;
area2=qtilde2*time_step;
inte2=inte2+area2;

tau1=kp1*qtilde1+kd1*qptilde1+ki1*inte1;
tau2=kp2*qtilde2+kd2*qptilde2+ki2*inte2;
 /**********************modelo dinamico del robot de 2 grados de libertad******/
if(t>80&&t<130){
m2=(0.343*.3)+0.343;
A=m1*(lc1*lc1)+m2*((L1*L1)+(lc2*lc2))+I1+I2;
B=m2*L1*lc2;
C=m2*(lc2*lc2)+I2;
H=m1*g*lc1+m2*g*L1;
F=m2*g*lc2;
G=m2*L1*lc2;
m11=A+2*B*cos(q2);
m12=C+B*cos(q2);
m21=m12;
m22=C;
c11=-B*sin(q2)*dq2;
c12=-B*sin(q2)*(dq1+dq2);
c21=B*sin(q2)*dq1;
g1=H*sin(q1)+F*sin(q1+q2);
g2=F*sin(q1+q2);
det=m11*m22-m12*m21;
v1= tau1-c11*dq1-c12*dq2-g1-0.2*dq1;
v2= tau2-c21*dq1-g2-0.2*dq2;
}
else{
m2=0.343;
A=m1*(lc1*lc1)+m2*((L1*L1)+(lc2*lc2))+I1+I2;
B=m2*L1*lc2;
C=m2*(lc2*lc2)+I2;
H=m1*g*lc1+m2*g*L1;
F=m2*g*lc2;
G=m2*L1*lc2;
m11=A+2*B*cos(q2);
m12=C+B*cos(q2);
m21=m12;
m22=C;
c11=-B*sin(q2)*dq2;
c12=-B*sin(q2)*(dq1+dq2);
c21=B*sin(q2)*dq1;
g1=H*sin(q1)+F*sin(q1+q2);
g2=F*sin(q1+q2);
det=m11*m22-m12*m21;
v1= tau1-c11*dq1-c12*dq2-g1;
v2= tau2-c21*dq1-g2;
}
f1=dq1;
f2=dq2;
f3=((m22*v1)-(m12*v2))/det;
f4=((-m12*v1)+(m11*v2))/det;
q1=f1*time_step+q1;
q2=f2*time_step+q2;
dq1=f3*time_step+dq1;
dq2=f4*time_step+dq2;
```

```
 /*************************************************************************/
     fprintf(frobot2xx,"%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\n",t,qd1,q1,qd2,q2,qtilde1,qtilde2,tau1,tau2);
      t=t+time_step;
      }while(t<=T);
    fclose(frobot2xx);
}/*::::::::::::::::::::::::::::::::::::Fin de main::::::::::::::::::::::::::::::::::::::::::::::::::*/
```

Código para el control neuronal

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include <conio.h>
#define time_step  0.01
#define T          160
**** Declaración de las funciones utilizadas durante el programa (prototipos) ****
double Sigm (double);      /* función de activacion */
double Create_desired();   /* función de trayectorias deseadas*/
/*************************************************************************************************
**** Definición las constantes utilizadas durante el programa ****
#define n_in    16        /* num. de unidades de entrada */
#define n_hide1 8         /* num. de unidades ocultas */
#define n_out   2         /* num. de unidades de salida */
#define Kv1     0.8
#define Kv2     0.1
#define Kz      -0.09
#define lam1    12
#define lam2    11
//**** ***************************************
//**** P R O G R A M A P R I N C I P A L ****
//**** ***************************************
double t;
double q1=0;              /*posición 1 del eslabón 1*/
double q2=0;              /*posición 2 del eslabón 2*/
double dq1=0;             /*vel 1 del eslabón 1*/
double dq2=0;             /*vel 2 del eslabón 2*/
double tau1=0;
double tau2=0;
double qd1=0.0,qd2=0.0,qdp1=0.0,qdp2=0.0,qdpp1=0.0,qdpp2=0.0;
double qtilde1=0,qtilde2=0,qptilde1=0.0,qptilde2=0.0,r1=0.0,r2=0.0;
double norm_weight=0;
double m11,m12,m21,m22,c11,c21,c12,g1,g2,det,v1,v2,f2,f1,f3,f4,Sc,rc;
double L1=  0.193;                //longitud de eslabón 1
double L2=  0.145;                //longitud de eslabón 2
double lc1= 0.136;                //distancia al centro de masa 1
double lc2= 0.048;                //distancia al centro de masa 2
double m1=  0.763;                //masa 1
double m2=  0.343;                //masa 2
double I1=  0.0186;               //inercia eslabón 1
double I2=  0.0026987;            //inercia eslabón 2
double g=   9.81;                 //aceleración de gravedad
double A=m1*(lc1*lc1)+m2*((L1*L1)+(lc2*lc2))+I1+I2;
double B=m2*L1*lc2;
double C=m2*(lc2*lc2)+I2;
double H=m1*g*lc1+m2*g*L1;
double F=m2*g*lc2;
double G=m2*L1*lc2;
double norm_r=0.0,aux,v_t[n_out];
double W_max=2.0;
double amp1=-6.2832,amp2=3.1416;
```

```
double fact2=0.1;
   double fact1=0.05;
   int i,j,k;
     double error[n_out];              /* errores en las capas de la red*/
      double oldincwhide1[n_in][n_hide1]; /*Anteriores incrementos del error de la capa oculta*/
      double oldincwhide2[n_hide1];      /*Anteriores incrementos del error de la retro capa oculta*/
      double oldincwout[n_hide1][n_out];  /*Anteriores incrementos del error de la capa de salida*/
      double incw;                       /*incremento del peso*/
      double delta[n_out];               /*Señal de error (derivada del error respecto al peso: Capa de salida)*/
      double error_hide;                 /*Señal de error (derivada del error respecto al peso: Capa oculta)*/
      double in[n_in],hide1[n_hide1],out[n_out],hide1_a[n_hide1],deltas[n_hide1],hide_1[n_hide1]; /*entradas a las
unidades de de proceso*/
      double whide1[n_in][n_hide1],whide2[n_hide1],wout[n_hide1][n_out],ro[n_in][n_hide1]; /* pesos de las conexiones
*/
      //t i,j,k,K;                       /* contadores*/
      double LCout=0.1;                  /* coeficiente de aprendizaje de la capa de salida */
      double LChide=0.1;                 /* coeficiente de aprendizaje de la capa oculta */
      double momentum=0.4;               /* coeficiente de la ultima modificación del peso */
     double a;
/* los pesos son incializados*/
main()
{/*::::::::::::::::::::::::::::::Principio de main::::::::::::::::::::::::::::::::::::::::::::::::::*/
      FILE *frobot2xx;        /* Archivo de datos  */
     frobot2xx=fopen("entrenamiento.txt","w");
      for(i=0;i<n_in;i++)
         {
          for(j=0;j<n_hide1;j++)
                 {
                  a=rand();
                  whide1[i][j]=0.001; /*Pesos de la capa oculta*/
                  oldincwhide1[i][j]=0;
                  ro[i][j]=0.0;
                 }
         }
         for(j=0;j<n_hide1;j++)
                {
                  a=rand();
                  a=a/33879.0;
                  whide2[j]=0.001; /*Pesos de retroalimentación de la capa oculta*/
                  oldincwhide2[j]=0;
                  hide1[j]=0;
                  deltas[j]=0;
                }
        for(i=0;i<n_hide1;i++)
          {
           for(j=0;j<n_out+1;j++)
                 {
                  a=rand();
                  a=a/33879.0;
                  wout[i][j]=0.001; /*Pesos de la capa de salida*/
                  oldincwout[i][j]=0.0;
                 }
          }

     // s=fopen("pesos.txt","w");
     // printf("\n\n INICIO DE LA SIMULACION\n\n\n\n");
/* EMPIEZA EL ENTRENAMIENTO */
do
{
      qd1=amp1*sin(fact1*t);
      qd2=amp2*sin(fact2*t);
      qdp1=amp1*fact1*cos(fact1*t);
```

```
      qdp2=amp2*fact2*cos(fact2*t);
      qdpp1=-amp1*fact1*fact1*sin(fact1*t);
      qdpp2=-amp2*fact2*fact2*sin(fact2*t);
      qtilde1=qd1-q1;
      qtilde2=qd2-q2;
      qptilde1=qdp1-dq1;
      qptilde2=qdp2-dq2;
      r1=qptilde1+lam1*qtilde1;
      r2=qptilde2+lam2*qtilde2;
      in[0]=qtilde2;
      in[1]=qd1;
      in[2]=qd2;
      in[3]=qptilde1;
      in[4]=qptilde2;
      in[5]=qdpp1;
      in[6]=qdpp2;
      in[7]=qdp1;
      in[8]=qdp2;
      in[9]=qtilde1;
      in[10]=r1;
      in[11]=r2;
      in[12]=q1;
      in[13]=q2;
      in[14]=dq1;
      in[15]=dq2;
/*************************************************************************************************************/
/* Propagacion del error*/
/************************/
/* Primera Capa oculta */
/************************/
        for(j=0;j<n_hide1;j++)
          {
           hide_1[j]=0;
              for(i=0;i<n_in;i++)
                   {
                    hide_1[j]+=in[i]*whide1[i][j]; /*Suma de las entradas multip.por los pesos*/
                   }
                    hide_1[j]+=whide2[j]*hide1[j]; /*Suma ponderada*/
                    hide1_a[j]=hide1[j];
                    hide1[j]=1/(1+exp(-hide_1[j])); /*Salida de cada una de las neuronas ocultas*/
           }
//******************/
/* Capa de salida */
/******************/
        for(k=0;k<n_out;k++)
          {
           out[k]=0;
              for(j=0;j<n_hide1;j++)
                   {
                   out[k]+=hide1[j]*wout[j][k]; /*Suma de las entradas multip.por los pesos*/
                   }
                  // out[k]=Sigm(out[k]); /*Salida de cada una de las neuronas de salida*/
          }
/*************************************************************************************************************/
/* Calculo del error cometido, de la señal de error, */
/* y actualización de los pesos por cada unidad de salida */
/*************************************************************************************************************/
/******************/
/* Capa de salida */
/******************/
              for(k=0;k<n_out;k++) /* Capa de salida */
                   {
```

```
                error[0]=qd1-q1;    /*Calculo del error en cada unidad de salida*/
                error[1]=qd2-q2;
                delta[k] = error[k];//*out[k]*(1-out[k]); /*Calculo de la señal de error*/
                                        /* en cada unidad de salida */
              for (j=0;j<n_hide1;j++)
                   {
                   incw= LCout*delta[k]*hide1[j]+ momentum*oldincwout[j][k]; /*Incremento*/
                   wout[j][k]+=incw; /*Actualización del peso en las salidas*/
                   oldincwout[j][k]=incw; /*Se guarda el incremento anterior*/
                   }
               }
/************************/
/* Primera Capa oculta */
/************************/
            for(j=0;j<n_hide1;j++)/*Capa oculta*/
               {
               error_hide=0;
                   for(k=0;k<n_out;k++)
                   {
                   error_hide=error_hide+delta[k]*wout[j][k]; /*Calculo del error en cada unidad oculta*/
                   }
               deltas[j]=hide1[j]*(1-hide1[j])*(hide1_a[j]+whide2[j]*deltas[j]);
               incw=LChide*error_hide*deltas[j]+momentum*oldincwhide2[j]; /*Incremento*/
               whide2[j]+=incw;/*Actualización del peso retro en la capa culta*/
               oldincwhide2[j]=incw;/*Se guarda el incremento anterior*/
               }
/************************/
/* Capa de entrada */
/************************/
            for(j=0;j<n_hide1;j++)
                {
                error_hide=0;
                  for(k=0;k<n_out;k++)
                      {
                       error_hide+=delta[k]*wout[j][k];
                      }

                  for (i=0;i<n_in;i++)
                     {
                      ro[i][j]=hide1[j]*(1-hide1[j])*(in[i]+(whide2[j]*ro[i][j]));
                      incw=LChide*error_hide*ro[i][j]+momentum*oldincwhide1[i][j];/*Incremeto*/
                      whide1[i][j]+=incw;/*Actualización del peso de la capa entrada*/
                      oldincwhide1[i][j]=incw;/*Se guarda el incremento anterior*/
                     }
                }
///////////////termino robusto-------------------------
    for (int k=0;k<n_out;++k)
    {
    for(int j=0;j<n_hide1;++j)
       {
       norm_weight=norm_weight+wout[j][k]*wout[j][k];
       }
    }
    for(int j=0;j<n_hide1;++j)
       {
       norm_weight=norm_weight+whide2[j]*whide2[j];
       }
    for (int i=0;i<n_in;++i)
    {
    for(int j=0;j<n_hide1;++j)
       {
       norm_weight=norm_weight+whide1[i][j]*whide1[i][j];
```

```
        }
    }
    norm_weight=sqrt(norm_weight);
    v_t[0]=Kz*(norm_weight+W_max)*r1;
    v_t[1]=Kz*(norm_weight+W_max)*r2;

    tau1=r1*Kv1+out[0]-v_t[0];
    tau2=r2*Kv2+out[1]-v_t[1];
/**********************modelo dinamico del robot de 2 grados de libertad******/
     if(t>80&&t<130){
     m2=(0.343*.3)+0.343;
     A=m1*(lc1*lc1)+m2*((L1*L1)+(lc2*lc2))+I1+I2;
     B=m2*L1*lc2;
     C=m2*(lc2*lc2)+I2;
     H=m1*g*lc1+m2*g*L1;
     F=m2*g*lc2;
     G=m2*L1*lc2;
     m11=A+2*B*cos(q2);
     m12=C+B*cos(q2);
     m21=m12;
     m22=C;
     c11=-B*sin(q2)*dq2;
     c12=-B*sin(q2)*(dq1+dq2);
     c21=B*sin(q2)*dq1;
     g1=H*sin(q1)+F*sin(q1+q2);
     g2=F*sin(q1+q2);
     det=m11*m22-m12*m21;
     v1= tau1-c11*dq1-c12*dq2-g1-0.2*dq1;
     v2= tau2-c21*dq1-g2-0.2*dq2;
               }
            else{
     m2=0.343;
     A=m1*(lc1*lc1)+m2*((L1*L1)+(lc2*lc2))+I1+I2;
     B=m2*L1*lc2;
     C=m2*(lc2*lc2)+I2;
     H=m1*g*lc1+m2*g*L1;
     F=m2*g*lc2;
     G=m2*L1*lc2;
     m11=A+2*B*cos(q2);
     m12=C+B*cos(q2);
     m21=m12;
     m22=C;
     c11=-B*sin(q2)*dq2;
     c12=-B*sin(q2)*(dq1+dq2);
     c21=B*sin(q2)*dq1;
     g1=H*sin(q1)+F*sin(q1+q2);
     g2=F*sin(q1+q2);
     det=m11*m22-m12*m21;
     v1= tau1-c11*dq1-c12*dq2-g1-0.2*dq1;
     v2= tau2-c21*dq1-g2-0.2*dq2;
     }
     f1=dq1;
     f2=dq2;
     f3=((m22*v1)-(m12*v2))/det;
     f4=((-m12*v1)+(m11*v2))/det;
     q1=f1*time_step+q1;
     q2=f2*time_step+q2;
     dq1=f3*time_step+dq1;
     dq2=f4*time_step+dq2;


 /*****************************************************************************/
```

```
        fprintf(frobot2xx,"%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\n",t,qd1,q1,qd2,q2,qtilde1,qtilde2,tau1,tau2);

        t=t+time_step;
        norm_weight=0;

      }while(t<=T);
      fclose(frobot2xx);

   }/*::::::::::::::::::::::::::::::::::::::Fin de main::::::::::::::::::::::::::::::::::::::::::::::*/
/*************************************************************************************************************
**** FUNCIONES UTILIZADAS DURANTE EL PROGRAMA ******
*************************************************************************************************************/
double Sigm (double x)
{
return 1/(1+exp(-x));
}
```

Enseguida se muestra el código en Builder C++utilizado en la prueba experimental del controlador neuronal.

Código en Builder C++ para el control neuronal

```
//---------------------------------------------------------------------------
//Programa para controlar un manipualador de 2 GDL
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Main.h"
#include <math.h>
#include <stdio.h>
#include <share.h>
#include <conio.h>
#include <dos.h>
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "CSPIN"
#pragma resource "*.dfm"
//---------------------------------------------------------------------------
//----------DEFINICIÓN DE CONSTANTES-----------------------------------------
//---------------------------------------------------------------------------
#define Ts        0.01     //con el while del PIC, cada cuenta vale (4/FXtal)*256 seg
#define iTs       1.0/Ts    //inverso del ts
#define ppr1      500       // 500 para robot
#define ppr2      500
#define pi_       3.1416    // valor de pi
#define idref1    1.0       //
#define idref2    1.0       // corriente máxima proporcionada por la fuente
#define n1        19.7      // Relación caja de engranes 1 robot
#define invn1     (1.0/n1)  // Inverso de la Relación caja de engranes 1 robot
#define n2        19.5      // Relación caja de engranes 2 robot
#define invn2     (1.0/n2)  // Inverso de la Relación caja de engranes 2 robot

#define km1       0.0306    //constantes del motor 1 y 2
#define km2       0.0218
/*************************************************************************************************************
**** Definición las constantes utilizadas durante el programa ****
*************************************************************************************************************/
#define n_in    16        /* num. de unidades de entrada */
#define n_hide1 8         /* num. de unidades ocultas */
#define n_out   2         /* num. de unidades de salida */
//***********ganancias del controlador*************************/
#define Kv1     0.6     //si aumenta kv disminuye lam
```

```
#define Kv2     0.02
#define Kz      -0.027
#define lam1    34//21
#define lam2    66//61
//---------------------------------------------------------------------------
FILE *ptrMonit;
TMainForm *MainForm;
//---------------------------------------------------------------------------
//-------------VARIABLES GLOBALES-------------------------------------------
//---------------------------------------------------------------------------
float norm_r=0.0,aux,v_t[n_out];
float W_max=0.12;
float amp1=-6.2832,amp2=3.1416;
float fact2=0.1;
float fact1=0.05;
float qd1=0;     // set point 1
float qd2=0;     // set point 2
float qdp1=0.0,qdp2=0.0,qdpp1=0.0,qdpp2=0.0;
float qptilde1=0.0,qptilde2=0.0,r1=0.0,r2=0.0;
float norm_weight=0;

//-----------------
unsigned char flagcom=0,flagfile=0,signo_sal,pwm;  //de 8 bits
unsigned int pos1,pos1a,pos1b,pos2,pos2a,pos2b;    //de 32 bits
unsigned int cuenta1=0, cuenta2=0;                 //cuenta para el dac 1 y 2
float teta1=0,teta2=0;                  /*POSICION EN RADIANES DEL MOTOR 1 Y 2*/
float teta1_1=0,teta2_1=0,teta1P,teta2P;
//-----*TETA ANTERIOR Y VELOCIDAD RESPECTO A TETA*----------------------------------/
float t=0;                               /*VARIABLE PARA LLEVAR REGISTRO DEL TIEMPO*/
float tau1=0,tau2=0;
float esc1=pi_/(2*ppr1), esc2=pi_/(2*ppr2), escs1=255/(2.0*idref1);
float escs2=255/(2.0*idref2),inte1=0,inte2=0,area1=0,area2=0;
/*CONVERSION A RADIANES PARA EL MOTOR Y ESCALAMIENTO DE SALIDA PARA EL DAC*/
float q1=0, q2=0;/*posición física del grado de libertad (eslabón) DEL ROBOT*/
float q1_1=0,q2_1=0,qp1,qp2;/*Q ANTERIOR Y VELOCIDAD RESPECTO A Q*/
float qtil1=0, qtil2=0, qtil1_1=0, qtil2_1=0,qtilp1,qtilp2; /*SEÑALES DE ERROR Q'=Qd-Q*/
float tau1max=10*n1*km1*idref1,tau2max=10*n2*km2*idref2;
float id1, id2;                          //CORRIENTE DESEADA
int i,j,k;                      /* contadores*/
float esctau1=1.0/(km1*n1), esctau2=1.0/(km2*n2);
int flag=0;
      float error[n_out]={0};              /* errores en las capas de la red*/
      //float oldincwhide1[n_in][n_hide1]; /*Anteriores incrementos del error de la capa oculta*/
      float oldincwhide2[n_hide1]={0};     /*Anteriores incrementos del error de la retro capa oculta*/
     // float oldincwout[n_hide1][n_out];  /*Anteriores incrementos del error de la capa de salida*/
      float incw=0;                    /*incremento del peso*/
      float delta[n_out]={0};              /*Señal de error (derivada del error respecto al peso: Capa de salida)*/
      float error_hide=0;               /*Señal de error (derivada del error respecto al peso: Capa oculta)*/
      float in[n_in],hide1[n_hide1]={0},out[n_out]={0};
      float hide1_a[n_hide1]={0};
      float deltas[n_hide1]={0},hide_1[n_hide1]={0}; /*entradas a las unidades de de proceso*/
      //double whide1[n_in][n_hide1],whide2[n_hide1],wout[n_hide1][n_out];
     // float ro[n_in][n_hide1]; /* pesos de las conexiones */

      float LCout=0.05;                /* coeficiente de aprendizaje de la capa de salida */
      float LChide=0.05;                /* coedifiente de aprendizaje de la capa oculta */
      float momentum=0.1;               /* coeficiente de la ultima modificacion del peso */
/***********************************************/
float whide1[16][8]={
  {0.001,0.001,0.001,0.001,0.001,0.001,0.001,0.001},
  {0.001,0.001,0.001,0.001,0.001,0.001,0.001,0.001},
  {0.001,0.001,0.001,0.001,0.001,0.001,0.001,0.001},
```

```
   {0.001,0.001,0.001,0.001,0.001,0.001,0.001,0.001},
   {0.001,0.001,0.001,0.001,0.001,0.001,0.001,0.001},
   {0.001,0.001,0.001,0.001,0.001,0.001,0.001,0.001},
   {0.001,0.001,0.001,0.001,0.001,0.001,0.001,0.001},
   {0.001,0.001,0.001,0.001,0.001,0.001,0.001,0.001},
   {0.001,0.001,0.001,0.001,0.001,0.001,0.001,0.001},
   {0.001,0.001,0.001,0.001,0.001,0.001,0.001,0.001},
   {0.001,0.001,0.001,0.001,0.001,0.001,0.001,0.001},
   {0.001,0.001,0.001,0.001,0.001,0.001,0.001,0.001},
   {0.001,0.001,0.001,0.001,0.001,0.001,0.001,0.001},
   {0.001,0.001,0.001,0.001,0.001,0.001,0.001,0.001},
   {0.001,0.001,0.001,0.001,0.001,0.001,0.001,0.001},
   {0.001,0.001,0.001,0.001,0.001,0.001,0.001,0.001},
                  };
float oldincwhide1[16][8]={
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},

                       };
float ro[16][8]={
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
   {0,0,0,0,0,0,0,0},
                      };
float whide2[8]= {0.001};
float wout[8][2]={
                {0.001,0.001},
                {0.001,0.001},
                {0.001,0.001},
                {0.001,0.001},
                {0.001,0.001},
                {0.001,0.001},
                {0.001,0.001},
                {0.001,0.001},
```

```
            };
float oldincwout[8][2]={
            {0,0},
            {0,0},
            {0,0},
            {0,0},
            {0,0},
            {0,0},
            {0,0},
            {0,0},
                };
//----------------PROGRAMA PRINCIPAL-----------------------------------------
//---------------------------------------------------------------------------

void ProcessByte(BYTE byte)
  {
if(flagcom!=0)                    //SI NO SE HA TRANSFERIDO NINGUN BYTE
      flagcom++;                  //DEL MICRO A LA PC...
if((byte==0xAA)&&(flagcom==0))        //INICIALIZA LAS VARIABLES NECESARIAS
  {
      pos1=0;  pos1a=0;  pos1b=0;
      pos2=0;  pos2a=0;  pos2b=0;

      flagcom=1;
  }
if(flagcom==2)                    //TRANSFERENCIA DE BYTES PARA POS1
  {
      pos1=byte;
      pos1=pos1<<24;
  }
if(flagcom==3)
  {
      pos1a=byte;
      pos1a=pos1a<<16;
  }
if(flagcom==4)
  {
      pos1b=byte;
      pos1b=pos1b<<8;
  }
if(flagcom==5)
  {
     pos1=pos1+pos1a+pos1b+byte;    //CONCATENA LOS BYTES RECIBIDOS EN UNA
  }                             //SOLA VARIABLE POS1

if(flagcom==6)                    //TRANSFERENCIA DE BYTES PARA POS2
  {
      pos2=byte;
      pos2=pos2<<24;
  }
if(flagcom==7)
  {
      pos2a=byte;
      pos2a=pos2a<<16;
  }
if(flagcom==8)
  {
      pos2b=byte;
      pos2b=pos2b<<8;
  }
if(flagcom==9)
  {
```

```
      pos2=pos2+pos2a+pos2b+byte;    //CONCATENA LOS BYTES RECIBIDOS EN UNA
                                     //SOLA VARIABLE POS2

//---------------CALCULO DE POSICION DE MOTORES---------------------------------
      teta2_1=teta2;
      teta2=(signed int)pos2;
      teta2=esc2*teta2;              //CALCULA LA POSICION FÍSICA DEL MOTOR 1
      teta2P=(teta2-teta2_1)*iTs;    //CALCULA LA VELOCIDAD DEL CAMBIO DE POS1

      teta1_1=teta1;
      teta1=(signed int)pos1;
      teta1=esc1*teta1;              //CALCULA LA POSICION FÍSICA DEL MOTOR 2
      teta1P=(teta1-teta1_1)*iTs;    //CALCULA LA VELOCIDAD DEL CAMBIO DE POS2

//----------CALCULO DE SET POINTS-------------------------------------------------
     qd1=amp1*sin(fact1*t);
     qd2=amp2*sin(fact2*t);
     qdp1=amp1*fact1*cos(fact1*t);
     qdp2=amp2*fact2*cos(fact2*t);
     qdpp1=-amp1*fact1*fact1*sin(fact1*t);
     qdpp2=-amp2*fact2*fact2*sin(fact2*t);
//----------CALCULO DE POSICION DE ESLABONES---------------------------------
      q1_1=q1;
      q1=teta1*invn1;     //CALCULA LA POSICION FÍSICA DEL GRADO DE LIBERTAD 1
      qtil1_1=qtil1;
      qtil1=qd1-q1;       //SEÑAL DE ERROR PARA EL GRAD. DE LIB 1
      qp1=(q1-q1_1)*iTs;  //VELOCIDAD CAMBIO DE POS DEL GRADO DE LIB 1
      qptilde1=qdp1-qp1;
      r1=(qtil1-qtil1_1)*iTs+(lam1*qtil1);

      q2_1=q2;
      q2=teta2*invn2;     //CALCULA LA POSICION FÍSICA DEL GRADO DE LIBERTAD 2
      qtil2_1=qtil2;
      qtil2=qd2-q2;       //SEÑAL DE ERROR PARA EL GRAD. DE LIB 2
      qp2=(q2-q2_1)*iTs;  //VELOCIDAD CAMBIO DE POS DEL GRADO DE LIB 2
      qptilde2=qdp2-qp2;
      r2=(qtil2-qtil2_1)*iTs+(lam2*qtil2);
//-----------------------------------------------------------
     in[0]=qtil2;
     in[1]=qd1;
     in[2]=qd2;
     in[3]=(qtil1-qtil1_1)*iTs;
     in[4]=(qtil2-qtil2_1)*iTs;
     in[5]=qdpp1;
     in[6]=qdpp2;
     in[7]=qdp1;
     in[8]=qdp2;
     in[9]=qtil1;
     in[10]=r1;
     in[11]=r2;
     in[12]=q1;
     in[13]=q2;
     in[14]=qp1;
     in[15]=qp2;


/*****************************************************************************************************/
/* Propagacion del error*/
/*************************/
/* Primera Capa oculta */
/*************************/
       for(j=0;j<n_hide1;j++)
```

```
          {
           hide_1[j]=0;
              for(i=0;i<n_in;i++)
                   {
                    hide_1[j]+=in[i]*whide1[i][j]; /*Suma de las entradas multip.por los pesos*/
                   }
                    hide_1[j]+=whide2[j]*hide1[j]; /*Suma ponderada*/
                    hide1_a[j]=hide1[j];
                    hide1[j]=1/(1+exp(-hide_1[j])); /*Salida de cada una de las neuronas ocultas*/
          }
//******************/
/* Capa de salida */
/******************/
        for(k=0;k<n_out;k++)
          {
           out[k]=0;
              for(j=0;j<n_hide1;j++)
                   {
                   out[k]+=hide1[j]*wout[j][k]; /*Suma de las entradas multip.por los pesos*/
                   }
                  // out[k]=Sigm(out[k]); /*Salida de cada una de las neuronas de salida*/
          }
/*****************************************************************************************************/
/* Calculo del error cometido, de la señal de error, */
/* y actualización de los pesos por cada unidad de salida */
/*****************************************************************************************************/
/******************/
/* Capa de salida */
/******************/
              for(k=0;k<n_out;k++) /* Capa de salida */
                   {
                    error[0]=qd1-q1;    /*Calculo del error en cada unidad de salida*/
                    error[1]=qd2-q2;
                    delta[k] = error[k];//*out[k]*(1-out[k]); /*Calculo de la señal de error*/
                                        /* en cada unidad de salida */
                 for (j=0;j<n_hide1;j++)
                       {
                       incw= LCout*delta[k]*hide1[j]+ momentum*oldincwout[j][k]; /*Incremeto*/
                       wout[j][k]+=incw; /*Actualización del peso en las salidas*/
                       oldincwout[j][k]=incw; /*Se guarda el incremento anterior*/
                       }
                  }
/************************/
/* Primera Capa oculta */
/************************/
               for(j=0;j<n_hide1;j++)/*Capa oculta*/
                   {
                   error_hide=0;
                       for(k=0;k<n_out;k++)
                       {
                       error_hide=error_hide+delta[k]*wout[j][k]; /*Calculo del error en cada unidad oculta*/
                       }
                   deltas[j]=hide1[j]*(1-hide1[j])*(hide1_a[j]+whide2[j]*deltas[j]);
                   incw=LChide*error_hide*deltas[j]+momentum*oldincwhide2[j]; /*Incremeto*/
                   whide2[j]+=incw;/*Actualización del peso retro en la capa culta*/
                   oldincwhide2[j]=incw;/*Se guarda el incremento anterior*/
                   }
/************************/
/* Capa de entrada */
/************************/
               for(j=0;j<n_hide1;j++)
                    {
```

```
                    error_hide=0;
                      for(k=0;k<n_out;k++)
                          {
                           error_hide+=delta[k]*wout[j][k];
                          }

                      for (i=0;i<n_in;i++)
                         {
                          ro[i][j]=hide1[j]*(1-hide1[j])*(in[i]+(whide2[j]*ro[i][j]));
                          incw=LChide*error_hide*ro[i][j]+momentum*oldincwhide1[i][j];/*Incremeto*/
                          whide1[i][j]+=incw;/*Actualización del peso de la capa entrada*/
                          oldincwhide1[i][j]=incw;/*Se guarda el incremento anterior*/
                         }
                    }
///////////////termino robusto--------------------------
     for (int k=0;k<n_out;++k)
     {
     for(int j=0;j<n_hide1;++j)
        {
        norm_weight=norm_weight+wout[j][k]*wout[j][k];
        }
     }
     for(int j=0;j<n_hide1;++j)
        {
        norm_weight=norm_weight+whide2[j]*whide2[j];
        }
     for (int i=0;i<n_in;++i)
     {
     for(int j=0;j<n_hide1;++j)
        {
        norm_weight=norm_weight+whide1[i][j]*whide1[i][j];
        }
     }
     norm_weight=sqrt(norm_weight);
     v_t[0]=Kz*(norm_weight+W_max)*r1;
     v_t[1]=Kz*(norm_weight+W_max)*r2;


      tau1=r1*Kv1+out[0]-v_t[0];
      id1=esctau1*tau1;
      id1=-id1;
      id1=0.1*id1;
      if (id1>idref1)         //ASEGURA MAXIMO VREF1
            id1=idref1;
      if (id1<-idref1)        //ASEGURA MINIMO -VREF1
            id1=-idref1;
      tau2=r2*Kv2+out[1]-v_t[1];
      id2=esctau2*tau2;
      id2=-id2;
      id2=id2;
      if (id2>idref2)         //ASEGURA MAXIMO VREF1
            id2=idref2;
      if (id2<-idref2)        //ASEGURA MINIMO -VREF1
            id2=-idref2;
//-----------SALIDA DE LOS DAC---------------------------------------
      cuenta1=escs1*(id1+idref1);
      cuenta2=escs2*(id2+idref2);

//-----------MUESTRA EN PANTALLA-------------------------------------
      MainForm->Edit1->Text = FloatToStr (q1);
      MainForm->Edit2->Text = FloatToStr (id1);
      MainForm->Edit3->Text = FloatToStr (tau1);
```

```
        MainForm->Edit4->Text = IntToStr (pos1);
        MainForm->Edit5->Text = FloatToStr (t);
        MainForm->Edit6->Text = FloatToStr (q2);
        MainForm->Edit7->Text = FloatToStr (id2);
        MainForm->Edit8->Text = IntToStr (pos2);
        MainForm->Edit9->Text = IntToStr (cuenta1);
        MainForm->Edit10->Text = IntToStr (cuenta2);
        MainForm->Edit11->Text = FloatToStr (qtil1);
        MainForm->Edit12->Text = FloatToStr (qtil2);
        MainForm->Edit13->Text = FloatToStr (tau2);
        MainForm->Edit14->Text = FloatToStr (qd1);
        MainForm->Edit15->Text = FloatToStr (teta1);
        MainForm->Edit16->Text = FloatToStr (inte1);
        MainForm->Edit17->Text = FloatToStr (esctau1);
        MainForm->Edit18->Text = FloatToStr (teta2);
        MainForm->Edit19->Text = FloatToStr (inte2);
        MainForm->Edit20->Text = FloatToStr (esctau2);
        MainForm->Edit21->Text = FloatToStr (qd2);

        //mandando primer byte
        MainForm->Acknowledge();
        MainForm->send_byte(cuenta1);

        //mandando segundo byte
        MainForm->Acknowledge();
        MainForm->send_byte(cuenta2);

        flagcom=0;

              /*abrir/crear un archivo*/

       if(flagfile==0)
          if((ptrMonit=fopen("neural_network_8Hideneuron.TXT", "w"))==NULL){}
        flagfile=1;
       //escribir algunos datos en el archivo

fprintf(ptrMonit,"%3.3f\t%3.3f\t%3.3f\t%3.3f\t%3.3f\t%3.3f\t%3.3f\t%3.3f\t%3.3f\t%3.3f\t%3.3f\t%3.3f\t%3.3f\t%3.3f\t%3.3f\t
%3.3f\t%3.3f\n",t,qd1,q1,qd2,q2,qtil1,qtil2,tau1,tau2,out[0],out[1],whide1[1][1],whide1[2][2],whide2[1],wout[1][1],wout
[1][2]);

        t=t+Ts;
        norm_weight=0;

    }
}

//---------------------------------------------------------------------------
__fastcall TMainForm::TMainForm(TComponent* Owner)
        : TForm(Owner), SerialPort(1, ProcessByte), fAcknowledge(true)
{
        if (SerialPort.IsReady() != TRUE)
                MessageBox(NULL, "Problemas con el puerto", "Error", MB_OK);
}
//---------------------------------------------------------------------------
void TMainForm::send_byte(unsigned char byte_sal)
{
        if (fAcknowledge == false)
                return;
        SerialPort.WriteByte(byte_sal);
        fAcknowledge = false;
}
//---------------------------------------------------------------------------
```

```
void TMainForm::Acknowledge()
{
	fAcknowledge = true;
}
void __fastcall TMainForm::Button1Click(TObject *Sender)
{
/* close the file */
fclose(ptrMonit);
Close();
}
```

Programa general para graficar resultados

Este programa fue realizado en MATLAB

```
clc;
clear all;
close all;
%leyendo las muestras tomadas al sistema
fid=fopen('entrenamiento.TXT','r');
datos=fscanf(fid, '%f', [9 inf]);    % datos tiene 4 renglones
fclose(fid);
t=datos(1,:);
qd1=datos(2,:);
q1=datos(3,:);
qd2=datos(4,:);
q2=datos(5,:);
qtil1=datos(6,:);
qtil2=datos(7,:);
tau1=datos(8,:);
tau2=datos(9,:);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
aux=0;
aux2=0;
N=size(qd1);
%for i=1:N
%    e=qd1-q1;
%end
N=size(qtil1);
for i=1:N(1,2)
    aux=qtil1(i)^2+aux;
    aux2=qtil2(i)^2+aux2;
    if t(i)<=2
        qtil1p(i)=0;
        qtil2p(i)=0;
    else
        qtil1p(i)=abs(qtil1(i));
        qtil2p(i)=abs(qtil2(i));
    end

end
    error1=sqrt(aux/N(1,2));
    error2=sqrt(aux2/N(1,2));
    emax1=max(qtil1p);
    emax2=max(qtil2p);
subplot(1,2,1), plot(t,qd1,t,q1);
grid on;
%legend('q2','qd2');
xlabel('Tiempo (s)');
ylabel('Par[N-m]');
subplot(1,2,2),plot(t,qd2,t,q2);
xlabel('Tiempo (s)');
ylabel('Par[N-m]');
```

```
grid on;
```