Universidad Autónoma de Querétaro
Facultad de Ingeniería
Maestría en Ciencias en Instrumentación y Control Automático

Serie de Fourier como controlador auto-ajustable para sistemas sometidos a perturbaciones periódicas

TESIS

Que como parte de los requisitos para obtener el grado de
Maestro en Ciencias en Instrumentación y Control Automático

Presenta:

Eduardo Espíndola López

Dirigido por:

Dr. Roberto Valentín Carrillo Serrano

SINODALES

Dr. Roberto Valentín Carrillo Serrano Presidente	Firma
Dr. Alfonso Gómez Espinosa Secretario	Firma
Dr. Víctor Manuel Hernández Guzmán Vocal	Firma
Dr. Edgar Alejandro Rivas Araiza Suplente	Firma
M.C. Alfonso Noriega Ponce Suplente	Firma

Dr. Aurelio Domínguez González
Director de la Facultad

Dra. Ma. Guadalupe Flavia Loarca Piña
Directora de Investigación y Posgrado

Centro Universitario
Querétaro, Qro.
Enero de 2017
México

RESUMEN

Una serie de Fourier como controlador de aprendizaje (FSLC) es propuesta e implementada para el control de velocidad de un motor síncrono de imán permanente (PMSM). Se presenta un análisis de convergencia del error para el FSLC y se especifíca la ley de actualización de coeficientes de la serie de Fourier. Se usa el control por campo orientado, como elemento básico para implementar tres controladores diferentes en un PMSM. Se compara el desempeño del FSLC con un controlador PI clásico y una red neuronal artificial, para un PMSM comercial operado en bajas velocidades. Se analiza la naturaleza periódica del rizado de par en PMSM's y se considera como una perturbación periódica; la cual, debe ser compensada por el controlador. Se obtiene una reducción substancial del rizado de par cuando se implementa el FSLC. Además, se alcanza una velocidad de aprendizaje más grade con el FSLC en comparación con la red neuronal artificial.

(**Palabras clave:** Series de Fourier, Rizado de par, PMSM, Controlador de aprendizaje, Control por campo orientado.)

SUMMARY

A new Fourier series learning controller (FSLC) is proposed and implemented for velocity control on a Permanent Magnet Synchronous Motor (PMSM). An analysis of error convergence for the FSLC is presented and the update law for the Fourier series coefficients is specified. The field oriented control method is used as a basic element to implement three different controllers for a PMSM. The performance of the FSLC is compared with a classical PI controller and an artificial neural network controller, for a commercial PMSM operated at low velocity. The periodic nature of torque ripple in PMSMs is analyzed and considered as a periodic disturbance which must be compensated by the controller. With the FSLC implementation is obtained a substantial reduction of the velocity ripple. Furthermore a higher speed of learning is achieved with the FSLC in comparison with the artificial neural network.

(**Key words:** Fourier series, Torque ripple, PMSM, Learning controller, Field oriented control.)

A mis padres por todo el apoyo y confianza que me brindaron.
A mis profesores y a mis amigos por su ayuda y aportaciones a este trabajo.
—por todo lo recibido, *Gracias*.

AGRADECIMIENTOS

- A la Universidad Autónoma de Querétaro (UAQ) por brindarme la oportunidad de formarme como maestro en ciencias.
- Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo económico para el desarrollo de esta investigación.
- A los profesores de la división de investigación y posgrado de la facultad de ingeniería por todas sus enseñanzas, en especial a mi asesor el Dr. Roberto Valentín Carrillo Serrano por su constante retro-alimentación, apoyo en este trabajo, confianza y amistad.
- Al Dr. Alfonso Gómez Espinosa por su ayuda, comentarios y recomendaciones a este trabajo.
- Al Dr. Víctor Manuel Hernández Guzmán, al M.C. Alfonso Noriega Ponce y al Dr. Edgar Alejandro Rivas Araiza por sus observaciones y sugerencias a este trabajo.
- A mi familia y amigos por las motivaciones y ánimos brindados.

ÍNDICE GENERAL

ÍNDICE DE CUADROS

ÍNDICE DE FIGURAS

Capítulo 1

INTRODUCCIÓN

1.1. Descripción del problema.

Las perturbaciones periódicas se presentan por diversas razones dependiendo del tipo de sistema; en sistemas mecánicos, una de las perturbaciones periódicas más conocidas son las vibraciones, normalmente causadas por partes rotativas dañadas, desgastadas o desbalanceadas. Estas vibraciones mecánicas pueden causar variaciones en la homogeneidad de los materiales, imperfecciones en los procesos de maquinados y en la manufactura de piezas, variaciones en las tolerancias de fabricación y en el ensamble de las máquinas, entre otros problemas (Lozano-Guzmán y Jáuregui-Correa, 2013). La frecuencia y amplitud de las vibraciones mecánicas son determinadas por diversos factores, incluyendo la velocidad de rotación y la parte mecánica que causa la vibración. El conjunto de todas las fuentes de vibración, podría producir una perturbación periódica que contiene las frecuencias y amplitudes de todas las fuentes, como se muestra en la Figura 1.1.

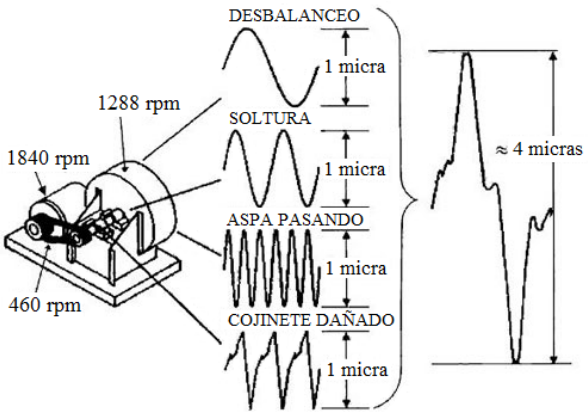

Figura 1.1: Ejemplos de vibraciones mecánicas (modificado de Sitenordeste (2015)).

Los rizados de par y de velocidad son otro tipo de perturbaciones periódicas que se presentan en algunos motores eléctricos; tal es el caso del motor sincrono de imán permanente (PMSM por sus siglas en inglés). Estas perturbaciones periódicas en los PMSM's son muy visibles cuando se utilizan a bajas velocidades. Los rizados de par y de velocidad pueden ser observados en ciertas situaciones; por ejemplo, en el control por campo orientado (FOC por sus siglas en inglés) de velocidad para PMSM's, el rizado de par, evita que la referencia sea alcanzada y en cambio, la velocidad medida oscila alrededor de la velocidad deseada. En las gráficas de la posición angular, el rizado de par se observa como una distorsión de las lineas rectas que deben formarse cuando el motor gira a una velocidad constante (ver Figura 1.2) (Gómez-Espinosa et al., 2013).

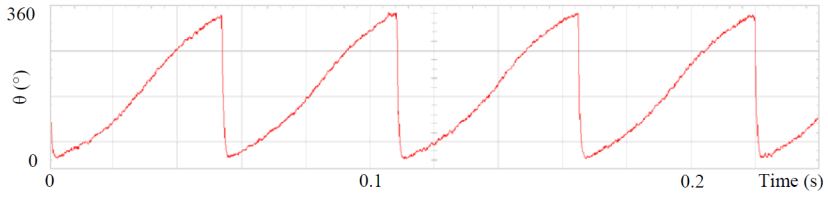

Figura 1.2: Señal de posición de un PMSM con rizado de par (Gómez-Espinosa et al., 2013).

Algunos de los problemas que causan los rizados de par y de velocidad en motores eléctricos como el PMSM están relacionados con las máquinas que usan estos actuadores en bajas velocidades; por ejemplo, en la manufactura se pueden generar maquinados de baja calidad, en los robots manipuladores, el seguimiento de trayectorias a velocidades bajas puede ser poco preciso y es peligroso cuando el robot es utilizado en ciertas áreas como la medicina. En general, cuando un motor eléctrico no tiene un buen desempeño, es posible que los sistemas electromecánicos no cumplan sus tareas apropiadamente.

En resumen, las perturbaciones periódicas pueden causar severos inconvenientes; sin embargo, en el área de control automático, se observa un problema que podría englobar las desventajas de los sistemas en lazo cerrado sometidos a perturbaciones periódicas. Este problema es el error oscilante de la salida de la planta respecto a la referencia cuando el controlador no es capaz de compensar estas perturbaciones. Es por ello que el siguiente trabajo de investigación se ocupa de diseñar un controlador que sea capaz de compensar las perturbaciones periódicas y a su vez pueda eliminar el error oscilante del sistema en lazo

cerrado.

En este trabajo de investigación se propone el uso de la serie de Fourier como controlador auto ajustable para reducir las perturbaciones periódicas. La justificación del uso de la serie de Fourier como controlador se muestra en la siguiente sección; sin embargo, es intuitivo el uso de la serie de Fourier por su naturaleza periódica.

Con la idea de usar la serie de Fourier como controlador auto ajustable, surge el problema de la investigación. Si la ley de control de un sistema lineal o no lineal esta dado por:

$$u(t) = \frac{a_0}{2} + \sum_{n=0}^{N} a_n \cos(\omega_n t) + b_n \sin(\omega_n t) \tag{1.1}$$

Entonces, ¿Cuál debe ser la regla de actualización de los coeficientes a_0, a_n y b_n, para que la ecuación (1.1) sirva como el controlador de un sistema sometido a perturbaciones periódicas?.

1.2. Justificación.

En numerosas ocasiones, a diferencia de las perturbaciones constantes, las perturbaciones periódicas requieren una inyección de energía variable que se anticipe a la siguiente perturbación. En el caso de las perturbaciones aleatorias no se puede predecir cuando ocurrirán; sin embargo, en las perturbaciones periódicas existe una o varias frecuencias con la cuales se podría saber el momento en el que ocurrirá la siguiente perturbación. Por lo tanto, las perturbaciones periódicas podrían ser compensadas antes de que perjudiquen al sistema si se conoce la frecuencia o frecuencias que las producen.

En otras palabras, para el caso de las perturbaciones aleatorias, el controlador normalmente actúa después de que la perturbación ocurre, lo que ocasionaría que el sistema sea alterado hasta que el controlador reaccione; pero en el caso de las perturbaciones periódicas el controlador si podría actuar antes de que la siguiente perturbación afecte al sistema.

Como ya se observó en la sección anterior, las perturbaciones periódicas pueden causar diversos problemas que los controladores de los sistemas en lazo cerrado deben evitar. Es decir, los controladores son los responsables de compensar este tipo de perturbaciones.

En la revisión de la literatura se observó que son pocos lo controladores que con-

templan alguna parte en el dominio de la frecuencia y específicamente las perturbaciones periódicas. Además, la mayor parte de los controladores actuales para sistemas no lineales, que podrían contemplar perturbaciones periódicas o que tienen alguna parte en el dominio de la frecuencia, están formados por un controlador principal en el dominio del tiempo y una parte auxiliar en el dominio de la frecuencia. Sin embargo, la estabilidad del sistema en lazo cerrado depende únicamente del controlador principal, por lo que las perturbaciones periódicas aún podrían alterar a los sistemas.

Por otro lado, se ha demostrado que el hecho de tener un controlador que haga modificaciones en los armónicos de la señal de control disminuye las incertidumbres del modelado determinista (Huang y Cai, 2000). Entonces, se puede decir que el requerimiento general para el controlador de un sistema lineal o no lineal sometido a perturbaciones periódicas es que pueda agregar un armónico o armónicos equivalentes a las frecuencias de la perturbación, con el objetivo de predecir la salida de control necesaria y eliminar la perturbación.

Debido a la naturaleza periódica de la serie de Fourier y a su capacidad de aproximar cualquier función por compleja que sea (ver Figura 1.3), en este trabajo de investigación se propone a la serie de Fourier como un controlador auto ajustable para sistemas sometidos a perturbaciones periódicas. Se demuestra que la serie de Fourier es capaz de controlar un sistema no lineal como el PMSM y además puede auto ajustarse agregando los armónicos necesarios, reduciendo las perturbaciones periódicas.

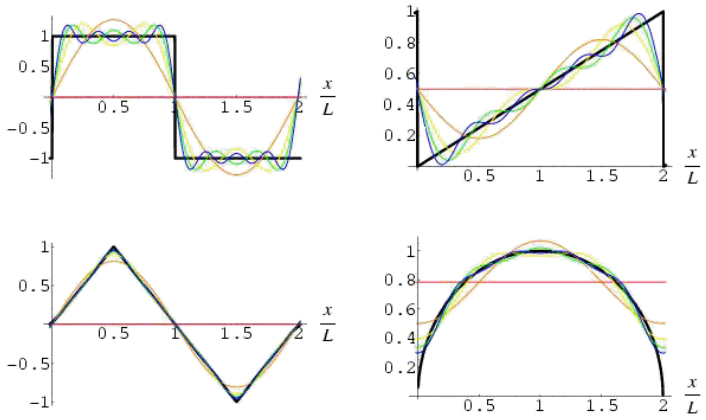

Figura 1.3: Ejemplos de aproximaciones con la serie de Fourier (modificado de Rodrigo507 (2010)).

1.3. Objetivo.

Diseñar una regla de actualización de coeficientes de una serie de Fourier aplicada como ley de control en sistemas de lazo cerrado sometidos a perturbaciones periódicas, mediante el análisis de convergencia del error.

1.4. Hipótesis.

Una serie de Fourier con coeficientes adaptables aplicada como ley de control en sistemas de lazo cerrado sometidos a perturbaciones periódicas asegura la convergencia del error.

1.5. Antecedentes y estado del arte.

Las series de Fourier son herramientas útiles en distintas disciplinas debido a su potencialidad como aproximadores universales a partir de las funciones propias del seno y el coseno, las cuales dejan a la vista las propiedades de las funciones en términos de frecuencia. En el área de control automático la serie de Fourier ha tenido algunas aportaciones como se muestra en Hung y Ding (1993). En este artículo se diseña una forma eficiente de excitar un motor sin escobillas de corriente directa (BLDC por sus siglas en inglés) para reducir las variaciones de par, reducir los efectos del par de retención y minimizar el promedio de potencia suministrada; además se asegura que se reducen las vibraciones mecánicas, el ruido en aplicaciones de control de velocidad y precisión en aplicaciones de control de posición. Se hace un análisis del rizado de par usando una serie de Fourier en su forma exponencial en el modelo del par. El análisis arroja información acerca de la presencia de componentes armónicos adicionales; a partir de los cuales se menciona que el diseño de una ponderación optima de los armónicos en las corrientes del estator es un tipo de problema de minimización constreñida. Por último se aplica un algoritmo de corrección para disminuir los armónicos indeseados y conseguir las corrientes optimas.

Un año después en 1994 las series de Fourier volvieron a tener aplicación en los controladores; esta vez en un controlador digital de trayectorias autoajustable para un motor síncrono de imán permanente. Según Shouse y Taylor (1994), este controlador puede ser aplicado para el seguimiento de trayectorias de velocidad o posición cuando se desconocen

los parámetros eléctricos y mecánicos del motor. Se utiliza un nuevo método de parametrización por tramos lineales de las funciones características de ángulo-par y par de detención del motor que permiten la identificación arbitrariamente precisa de armónicos de orden superior, sin requerir más operaciones que las descripciones sinusoidales de término único. Para el control de par-aceleración se utiliza, entre otros términos, una serie de Fourier truncada que estima ciertos parámetros del motor a partir de la rotación mecánica que es medida de manera discreta.

En el año de 1995 se menciona posiblemente por primera vez el término series de Fourier como Red Neuronal (FSNN por sus siglas en inglés) por parte de Zhu y Paul (1995) quienes utilizan una arquitectura de red neuronal donde las funciones de activación son exponenciales complejas (ver Figura 1.4), lo que genera una serie de Fourier en su forma compleja y sus coeficientes son denominados pesos; los cuales, son modificados mediante la regla Δ o regla del gradiente para lograr un sistema de identificación de múltiples entradas y múltiples salidas. Se menciona que la topología propuesta esta libre de mínimos locales y se demuestra estabilidad global del aprendizaje dinámico de la FSNN usando el la regla Δ. Además, se muestra la capacidad de la FSNN para estimar las amplitudes de los espectros para la identificación de modelos lineales y no lineales, funciones de transferencia de segundo orden y modelos no lineales de cinemática inversa para robots.

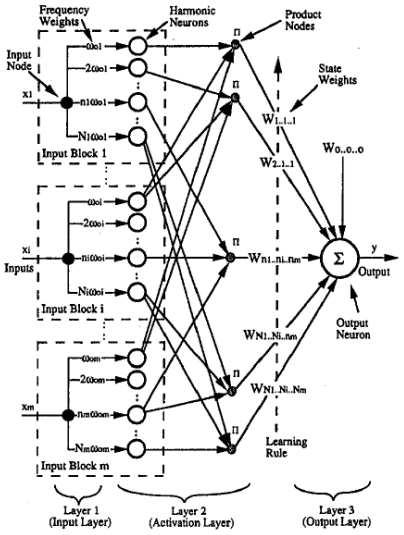

Figura 1.4: Topología de la FSNN propuesta por Zhu y Paul (1995).

Dos años después Hwang (1997) aplicó una FSNN para aproximar una función no lineal, compleja y en esos momentos desconocida causada por el fenómeno de fricción; se explicá que la mayor parte de las leyes de control basadas en un modelo con fricción no pueden alcanzar un excelente desarrollo de control para aplicaciones específicas, con posicionamiento de alta precisión y trayectorias de baja velocidad. El algoritmo de aprendizaje procesa una ganancia acotada; la cual, si se aumenta, genera una derivada negativa para un proceso de tiempo finito. La red neuronal es excitada con una señal acotada aleatoria y la señal de referencia; con estas condiciones, los pesos de la red neuronal son forzados a llegar a sus valores verdaderos. La estabilidad de este sistema de control se verifica con el criterio de estabilidad de Lyapunov y por ultimo el autor muestra los resultados obtenidos de simulación.

En el control de motores de inducción electromagnética (ACIM) Lee y Lee (1998) utilizaron una serie de Fourier para hacer una aproximación de la referencia de voltaje de excitación del motor; la cual, en su método corresponde a los indices de modulación cuando se aplica un inversor para alimentar al motor a través de un técnica de sobre-modulación de PWM (modulación por ancho de pulso) en un espacio vectorial. Se explicá que el rango de sobre-modulación se divide en dos modos dependiendo de los indices de modulación. En el primer modo los ángulos de referencia son derivados de la aproximación con la serie de Fourier y en el segundo modo la aproximación entrega los ángulos de participación. La estrategia planteada produce una relación lineal entre la salida de voltaje y los indices de modulación después del sexto paso de operación; por lo que se explica que se pueden guardar los datos de la relación en una tabla y aplicarse en linea. Demuestran el buen funcionamiento del algoritmo con resultados experimentales haciendo una transición del control V/f del ACIM al control propuesto después del sexto paso.

En el año 1999 se volvió a utilizar una serie de Fourier en un controlador. En este caso la serie de Fourier basada en un esquema de control inteligente fue propuesta por Cai y Huang (1999) quienes utilizan un controlador basado en dos partes (ver Figura 1.5) aplicado a una mesa de posicionamiento; la primer parte se basa en un controlador clásico proporcional-derivativo (PD) diseñado para estabilizar el sistema y mejorar la robustez ante perturbaciones aleatorias. La segunda parte del controlador es una serie de Fourier basada en un controlador inteligente que se usa para generar la señal de control ante incertidumbres del modelado

determinista. En este trabajo diseñan un algoritmo iterativo basado el la aproximación de las series de Fourier para generar una señal de control optima que force la trayectoria de estado convergente a una superficie deslizante estable. Por último se demuestra el funcionamiento del controlador propuesto con pruebas experimentales en una mesa de posicionamiento.

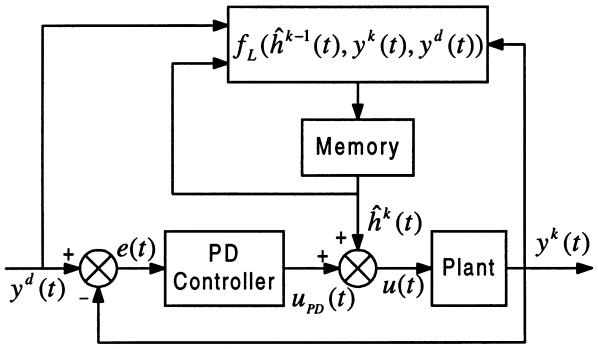

Figura 1.5: Diagrama del controlador propuesto por Cai y Huang (1999).

Con la idea de controlar eficientemente, en el estado transitorio y en el estado estable, a los sistemas con no linealidades y con incertidumbres, Huang y Cai (2000) proponen un controlador híbrido que identifica y compensa las incertidumbres deterministas simultáneamente. El algoritmo de control tiene dos partes, una retro-alimentación en el dominio del tiempo (controlador PD) y controlador iterativo de aprendizaje en el dominio de la frecuencia; el cual, esta basado en una serie de Fourier como red neuronal donde sus coeficientes o pesos son modificados mediante una regla diseñada por los autores. Se menciona que el controlador propuesto disminuye la variabilidad del sistema y reduce el efecto de las perturbaciones aleatorias. El controlador propuesto no requiere conocimiento del modelo matemático de la planta y las incertidumbres pueden ser estructuradas o no estructuradas. Se muestran resultados experimentales cuando se controla un servo sistema con una caja de cambios.

Basados en la idea de que el movimiento de un robot manipulador que realiza una trayectoria de tiempo finito se puede separar en subsistemas que a su vez se pueden modelar mediante la aproximación por series de Fourier; Tang et al. (2000) proponen un controlador de aprendizaje para el control de seguimiento descentralizado de un robot manipulador. Utilizando los elementos de la base de las series de Fourier que son funciones ortonormales, se observa que el problema del control de trayectorias en el dominio del tiempo es descentra-

lizado en un número de problemas independientes de regulación, es decir, los coeficientes de la serie de Fourier. Además se diseña un controlador de aprendizaje para controlar individualmente cada componente armónico de la señal de salida para cada subsistema del robot. El algoritmo de aprendizaje se diseña de tal forma que cada componente armónico de la señal de control converja a la trayectoria deseada dentro del ancho de banda del sistema. El controlador únicamente requiere la información de la entrada de referencia local y la salida del sistema. Por último se muestra el comportamiento del controlador propuesto con resultados experimentales sobre un robot de 3 grados de libertad.

Una aplicación de la serie de Fourier en el área de control inteligente se dio por parte de Qian et al. (2004). El problema que desean resolver es la variación de par que ocurre en PMSM's; se menciona que en estos motores ocurren pulsaciones de par parásitas debidas a la distribución de la densidad de flujo no sinusoidal en el entre-hierro, los errores en las mediciones de las corrientes y la reluctancia magnética variable del entre-hierro debida a las ranuras del estator. Se explica que las pulsaciones de par varían periódicamente con la posición angular del rotor y son reflejadas como variaciones de velocidad; las cuales degradan el comportamiento del control del PMSM especialmente a bajas velocidades. Con lo cual se propone un control de aprendizaje iterativo en el domino del tiempo para reducir las variaciones de par. Posteriormente para incrementar la robustez del algoritmo de control se modifica el esquema del controlador de aprendizaje para implementarlo en el dominio de la frecuencia por medio del desarrollo en series de Fourier; componente que es agregada a la señal de control proveniente del controlador diseñado en el dominio del tiempo (ver Figura 1.6). Se muestra resultados experimentales obtenidos en una plataforma basada en un procesador digital de señales (DSP) para el control de PMSM's.

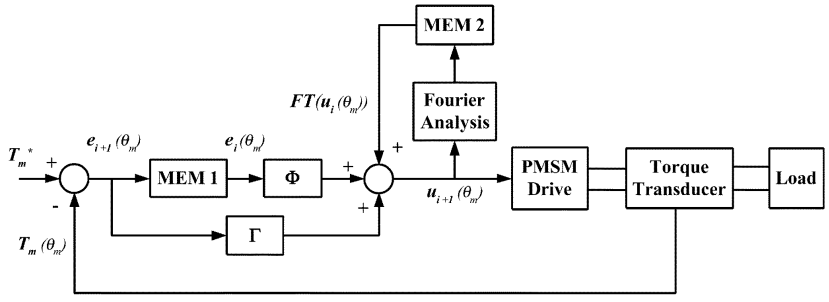

Figura 1.6: Diagrama del controlador propuesto por Qian et al. (2004).

Interesados por bajar el consumo de energía en unidades de discos duros micro, Kang et al. (2004) proponen un servo controlador para minimizar la corriente suministrada a estos dispositivos. Explican que la demanda de las unidades de discos duros de alta capacidad de almacenamiento extraíble para dispositivos electrónicos portátiles incrementa rápidamente, por lo cual, se requiere poner atención al consumo de energía eléctrica para alargar la duración de las baterías. Se menciona que el mayor consumo de corriente en unidades de discos duros se debe a transistores de potencia y resistencias en las bobinas de motores; por lo que proponen un servo controlador basado en un análisis a partir de la descomposición por series de Fourier y programación restringida para determinar el perfil optimo de consumo de energía. Al final se muestran resultados experimentales de una unidad de disco duro comercial.

En Enero de 2006 apareció una aplicación de un controlador para unidades de disco; esta vez por parte de Chang y Liu (2006) quienes proponen un controlador inteligente de aprendizaje en linea para cancelar errores repetitivos en unidades de disco. Explican que los errores periódicos en este tipo de unidades pueden ser aproximados por funciones como la serie de Fourier o la transformada wavelet; sin embargo en comparación con la transformada de Fourier la transformada wavelet puede proveer de información local más detallada de las señales. Por tanto, se diseña un controlador basado en una red wavelet para eliminar los errores causados por perturbaciones repetitivas. Se muestran resultados experimentales del controlador propuesto implementado en una unidad de disco real y se hace la comparación con un controlador basado en una serie de Fourier como red neuronal.

Para simplificar el control de las máquinas o motores de reluctancia conmutada (SRM), Khalil y Husain (2007) proponen un modelo matemático invertible y generalizado de la relación flujo/corriente de un SRM basado en la expansión por series de Fourier. Se explica que el modelo se deriva de la geometría de la máquina y de las propiedades de los materiales; es suficientemente preciso para representaciones actuales de máquinas y para poder simplificar los controladores en tiempo real. Muestran resultados experimentales y de simulación donde se observan las ventajas de la nueva representación de la SRM comparadas con otras representaciones más complejas.

Las aplicaciones de la serie de Fourier en el área de control automático siguieron en

el año 2008, por parte de Aghili (2008) quien diseña un control de par preciso de un motor BLDC. El autor menciona que el control preciso de estos motores requiere las características del par del motor, el cual sigue una función periódica de la posición angular del rotor. Se presenta un controlador adaptativo para el control de par de un motor BLDC, el cual estima los coeficientes de una serie de Fourier que aproxima una función periódica basada en las mediciones de los voltajes en las fases del motor y su posición angular. Se muestra analíticamente que el controlador adaptativo propuesto logra el seguimiento de par independientemente de las trayectorias de las señales de entrada. El controlador propuesto no se basa en el modelo matemático del motor por lo que la aplicación del controlador es simple y modular. Se muestran los resultados que obtuvo el autor de forma experimental donde se muestra que el troque del motor converge al par deseado.

En Wu et al. (2008) se diseña un conector digital en el dominio de la frecuencia basado en un esquema de control repetitivo para minimizar los armónicos de orden impar en la linea de corriente de las tres fases de señales PWM de un rectificador tipo boost en condiciones donde la fuente de voltaje está distorsionada y desbalanceada. Se menciona que el controlador se divide en dos análisis debido al modelo matemático del rectificador trifásico PWM tipo boost; la primer parte es el control de armónicos de la fuente de voltaje de corriente directa (DC) y la segunda parte es el control de armónicos de la linea de corriente que entrega el rectificador. Se diseña el controlador repetitivo junto con un controlador proporcional-integral (PI) clásico para lograr baja distorsión armónica en las corrientes del rectificador trifásico. Se implementa el controlador con un algoritmo de aprendizaje en el dominio de la frecuencia basado en aproximaciones por series de Fourier; con lo cual se obtiene flexibilidad para obtener diferentes ganancias de aprendizaje y retrasos individuales en el ángulo de fase para cada componente armónico. Obtienen resultados experimentales de un rectificador de $1.6[kVA]$ donde muestran la reducción de la distorsión total armónica del $21.09\,\%$ al $4.12\,\%$.

En el siguiente año Chen y Liang (2010a) proponen un sistema de control adaptativo por modos deslizantes para controlar un sistema piezoeléctricamente actuado. Se menciona que este tipo de sistemas tienen un comportamiento no lineal e invariante en el tiempo, por lo cual se hace difícil estabilizarlo. La idea original es usar una función aproximada con una serie de Fourier para establecer la función desconocida que requiere el control por modos

deslizantes basado en modelo. Además, se utiliza un esquema fuzzy con aprendizaje en linea para compensar la aproximación de la serie de Fourier y mejorar el desarrollo del control. Se tienen mejoras importantes en cuanto al diseño del controlador por modos deslizantes sin necesitar el modelo dinámico del sistema y con un error disminuido por la aproximación. Las reglas de actualización de los coeficientes de la serie de Fourier y de los parámetros de sintonización del fuzzy son derivados de una función de Lyapunov que garantiza la estabilidad del sistema de control. Este controlador es probado en una mesa de coordenadas $x - y$ actuada piezoelectricamente.

Posteriormente, se diseña un controlador de aprendizaje iterativo basado en una serie de Fourier como red neuronal, el cual es presentado para el control de seguimiento de trayectorias de una clase de sistemas no lineales con incertidumbres deterministas (Zuo y Cai, 2010). El controlador propuesto consiste en dos lazos; el lazo interno reduce la variabilidad del sistema y reduce la influencia de perturbaciones aleatorias. El lazo externo es una FSNN que suprime los errores causados por no linealidades e incertidumbres deterministas. La FSNN emplea como funciones de activación exponenciales complejas, por lo que es un método que esta esencialmente en el dominio de la frecuencia (ver Figura 1.7). Además se cuenta con una técnica de compensación de fase que hace posible usar componentes de alta frecuencia y mejorar el comportamiento de la trayectoria. Se muestran los resultados experimentales obtenidos de pruebas con una caja de cambios y una mesa de posicionamiento accionada por correas.

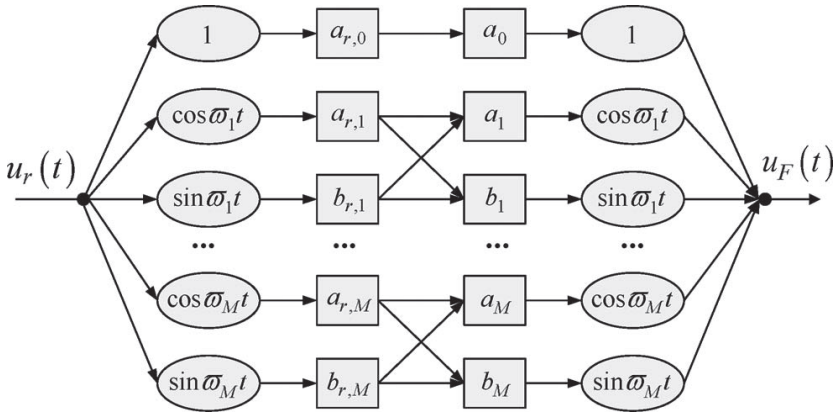

Figura 1.7: Estructura de la FSNN propuesta por Zuo y Cai (2010).

En ese mismo año se propuso otro controlador para sistemas con actuadores piezoeléctricos con no linealidades y con comportamiento variante en el tiempo. Se menciona que debido a que es difícil establecer con precisión un modelo dinámico para el diseño de un controlador basado en el modelo, se justifica el uso de un controlador de modos deslizantes adaptativo de modelo libre que pueda mejorar viajes pequeños y defectos de histéresis de este tipos de sistemas (Chen y Liang, 2010b). Al igual que en el trabajo anterior se utiliza una serie de Fourier para aproximar una función desconocida eliminando el requerimiento del modelo utilizado por el método de control por modos deslizantes. Se utiliza una regla de actualización de coeficientes de la serie de Fourier derivada de una función de Lyapunov que asegura la estabilidad del sistema de control. Por último se muestran resultados cuando se implementa el controlador en una mesa de posicionamiento $x-y$ con actuadores piezoeléctricos. El comportamiento del controlador propuesto se compara con un controlador por modos deslizantes basado en modelo tradicional donde se observan las mejoras en el seguimiento de trayectorias.

En el siguiente año, Dogruel y Hüseyin Celik (2011) presentan un método para controlar sistemas con referencias periódicas y perturbaciones mediante el empleo de controladores en una estructura matricial para cada componente armónico disperso. El método se basa en establecer automática y apropiadamente los niveles complejos de los componentes armónicos de la señal de control; para lo cual se genera una serie de Fourier con los componentes necesarios para la señal de control, la cual posteriormente es agregada a un controlador clásico PI para finalmente obtener una señal de control con los componentes armónicos deseados y con las propiedades de un control PI. Se presentan los resultados experimentales sobre el control de velocidad de un motor de CD con escobillas, donde se observan las ventajas y desventajas del método.

En el año 2012 se obtuvo una aplicación importante de la serie de Fourier en el área de control automático y específicamente en materia aeroespacial. En el trabajo publicado por Cho et al. (2012) se utiliza una serie de Fourier para representar las aceleraciones continuas y variables de bajo empuje, junto con las condiciones de contorno inicial y final para establecer las restricciones en las condiciones de empuje de la formación de satélites cuando vuelan en una órbita específica de perturbación llamada J_2. Se explica que la optima reconfiguración

de la formación de satélites volando sobre esta órbita es un problema de gran importancia para evitar la perdida de ubicación de estos equipos. Las funciones de empuje que son implementadas mediante los coeficientes optimos de la serie de Fourier reducen el costo durante la maniobra. Se muestran resultados de simulaciones numéricas donde se comparan los datos obtenidos por el método propuesto contra los resultados de otros artículos donde no se consideran las perturbaciones. Se menciona que la solución analítica desarrollada es más precisa que el comportamiento del control optimo por historiales y las trayectorias son calculadas más fácilmente aunque estén en presencia de perturbaciones.

Para minimizar las variaciones de par en máquinas de reluctancia conmutada Mikail et al. (2013) presentan un nuevo método para generar el perfil de las corrientes en las fases de la máquina. El método es la combinación del diseño de la máquina y un algoritmo de control. En primera instancia se diseña una máquina que tiene características de par simétrico con una porción plana extendida, en seguida se encuentra el perfil de corriente adecuado a través de una simulación y la identificación por una serie de Fourier, y finalmente se pone a punto el perfil. La simulación está hecha para verificar el método a través del acoplamiento del modelo de la máquina basada en elementos finitos y la simulación de un controlador dinámico. La simulación considera no linealidades, perdidas eléctricas, perdidas magnéticas y acoplamiento mutuo. Por último se verifica experimentalmente en un motor el funcionamiento del algoritmo con la reducción de las variaciones de par.

Con la idea de reducir las variaciones de par en un PMSM Gómez-Espinosa et al. (2013) proponen un nuevo algoritmo autoajustable para determinar una serie de Fourier, que al ser evaluada y agregada a la señal de control proveniente del control por campo orientado (FOC) para la regulación de velocidad de un PMSM puede reducir las variaciones de par que se ven representadas como variaciones de velocidad. Se explica que el algoritmo de control ajusta los parámetros del controlador basados en los componentes armónicos de distorsión para la compensación de la señal. Se muestran resultados experimentales satisfactorios de reducción de variación de par obtenidos con una plataforma de evaluación basada en un DSP en donde se programó el algoritmo propuesto.

En años recientes, diversos investigadores han estado interesados en el dominio de la frecuencia para controladores, o en la aplicación de la serie de Fourier como ayuda para

los sistemas de control. Las aplicaciones exitosas de este tipo de controladores van desde el control de tornillos de bola flexibles con incertidumbres y perturbaciones paramétricas variantes en el tiempo (Dong y Tang, 2014), algoritmos de control para alto rendimiento de maniobras periódicas de cuadricópteros (Hehn y D. Andrea, 2014), hasta el control de sumersión y seguimiento de trayectorias en vehículos roboticos sumergibles no tripulados (Adhami-Mirhosseini et al., 2014).

Como se ha visto en esta sección, el uso de los controladores en el dominio de la frecuencia y el uso de la series de Fourier en el área de control es cada vez más común, debido a las ventajas que tienen. Se ha logrado reducir las perturbaciones periódicas como las vibraciones y el rizado de par, con distintos métodos y en diferentes porcentajes. Sin embargo, las series de Fourier han sido únicamente herramientas complementarias en los sistemas de control; además, la convergencia de las series de Fourier depende de la estabilidad del controlador principal. Más aún, para obtener los coeficientes de las series, se ha utilizado la transformada continua de Fourier; la cual, requiere el periodo de la función que se representará y normalmente es desconocido.

En este trabajo de investigación, se demuestra que la serie de Fourier con un número finito de términos y con la correcta ley de actualización de coeficientes, puede representar una señal de control adecuada, para estabilizar un sistema no lineal sin ayuda de otros controladores. Es decir, la serie de Fourier puede ser utilizada como ley de control, inclusive si la referencia y/o el error no son periódicos. Además, se verifica que para las perturbaciones periódicas, como el rizado de par en PMSM's, la serie de Fourier puede reducir este tipo de perturbaciones suministrando los armónicos necesarios.

Capítulo 2

FUNDAMENTACIÓN TEÓRICA

2.1. Transformada discreta de Fourier (DFT).

Uno de los análisis más reveladores que se le puede hacer a un conjunto de datos, o a cualquier señal, es la descomposición de los datos en base a sus diferentes frecuencias. Dado un conjunto de datos definidos en el espacio del tiempo, la transformada discreta de Fourier (DFT por sus siglas en inglés) sirve como una herramienta que entrega un arreglo de frecuencias de esa función o conjunto de datos (Briggs y Emden, 1995).

La forma discreta de la transformada de Fourier se obtiene a partir de la transformada de Fourier en tiempo continuo de una señal $x(t)$:

$$X(\omega) = \int_{-\infty}^{\infty} x(t)e^{-i\omega t}dt, \qquad \omega \in (-\infty, \infty) \tag{2.1}$$

Reemplazando la integral infinita de (2.1) con una suma finita de números complejos, la DFT se define mediante la ecuación (Smith, 2007):

$$X(\omega_n) = \sum_{j=0}^{N-1} x(t_j)e^{-i\omega_n t_j}, \qquad n = 0, 1, ..., N-1 \tag{2.2}$$

Donde:

- $x(t_j)$ es la amplitud de la señal de entrada (real o compleja) en el tiempo $t_j[s]$.
- $t_j = jT_s$ es el j-ésimo instante de muestreo $[s]$, con $j \in \mathbb{Z}^+$.
- T_s es el periodo de muestreo $[s]$.
- $X(\omega_n) \in \mathbb{C}$ es el espectro de x, en la frecuencia ω_n.

- $\omega_n = n\Omega$ es la n-ésima muestra de frecuencia [$\frac{rad}{s}$], con $n \in \mathbb{Z}^+$.
- $\Omega = \frac{2\pi}{NT_s}$ es el intervalo de muestreo de las frecuencias [$\frac{rad}{s}$].
- $f_s = \frac{1}{T_s}$ es la frecuencia de muestreo [Hz].
- $N \in \mathbb{R}^+$ es el número de muestras en el tiempo = número de muestras en la frecuencia.

Asimismo, la transformada inversa discreta de Fourier (IDFT) esta dada por:

$$x(t_j) = \frac{1}{N}\sum_{n=0}^{N-1} X(\omega_n)e^{i\omega_n t_j}, \qquad j = 0, 1, ..., N-1 \tag{2.3}$$

En procesamiento digital de señales es más común la forma pura de la DFT y su inversa, para lo cual se utiliza la notación X_n y x_j respectivamente, quedando como:

$$X_n = \sum_{j=0}^{N-1} x_j e^{-i\frac{2\pi jn}{N}}, \qquad n = 0, 1, ..., N-1 \tag{2.4}$$

$$x_j = \frac{1}{N}\sum_{n=0}^{N-1} X_n e^{i\frac{2\pi jn}{N}}, \qquad j = 0, 1, ..., N-1 \tag{2.5}$$

En consecuencia, con la ecuación (2.4) es posible pasar de un conjunto de N datos discretos en el dominio del tiempo a un conjunto del mismo tamaño discreto en el dominio de la frecuencia. Asimismo, regresar del dominio de la frecuencia al tiempo con la ecuación (2.5). A estos procesos se les denominan *análisis* y *síntesis* respectivamente.

Por otra parte, la DFT y su inversa tienen ciertas propiedades que las hacen herramientas muy significativas y útiles en aplicaciones reales de procesamiento digital de señales. Algunas de estas propiedades se muestran a continuación (Cara, 2012-2013):

- Para $n = 0$ el resultado de la DFT es un número real y es la suma de las muestras x_j. A este componente de frecuencia se le denomina componente de *DC* haciendo referencia a la corriente continua de un circuito eléctrico y es la componente de frecuencia cero ó energía constante.

$$X_0 = \sum_{j=0}^{N-1} x_j e^0 = \sum_{j=0}^{N-1} x_j \tag{2.6}$$

- Si N es un número par, el resultado del la DFT para $n = N/2$ es un número real.

$$X_{\frac{N}{2}} = \sum_{j=0}^{N-1} x_j e^{-i\frac{2\pi j \frac{N}{2}}{N}} = \sum_{j=0}^{N-1} x_j e^{-i\pi j} = \sum_{j=0}^{N-1} x_j \cos(\pi j) \tag{2.7}$$

- La IDFT reproduce exactamente los valores de x_j, pero no reproduce completamente la señal continua $x(t)$. La representación de $x(t)$ será más precisa conforme el periodo de muestreo T_s tienda a cero.

- El espectro $(\omega_n, |X_n|)$ con $n = 0, 1, ..., N-1$ es simétrico respecto al eje y. Es decir, usando la notación de X_n^* para el conjugado de X_n y siendo N un número par, se tienen que:

$$X_n = X_{N-n}^*, \qquad n = 1, 2, ..., \frac{N}{2} - 1$$

Luego la mitad de los datos es redundante, ya que conociendo los términos desde $n = 1$ hasta $n = \frac{N}{2} - 1$ se pueden conocer los términos desde $n = \frac{N}{2} + 1$ hasta $n = N - 1$.

Y para N un número impar:

$$X_n = X_{N-n}^*, \qquad n = 1, 2, ..., \frac{N-1}{2}$$

Entonces, la mitad de la información vuelve a ser redundante, ya que conociendo los términos desde $n = 1$ hasta $n = \frac{N-1}{2}$ se pueden conocer los términos desde $n = \frac{N+1}{2}$ hasta $n = N - 1$.

- La frecuencia más alta que se puede evaluar con la DFT es la frecuencia de Nyquist $f_{nq} = \frac{f_s}{2}[Hz]$. Lo cual se deduce del siguiente análisis.

Para N un número par:

$$f_{nq} = \frac{f_s}{2}\,[Hz] = \frac{f_s}{2}\left[\frac{1}{s}\right]$$

$$f_{nq} = \frac{f_s}{2}\left[\frac{1}{s}\right]\left[\frac{2\pi}{1}\right] = \pi f_s \left[\frac{rad}{s}\right]$$

En donde, a partir de las definiciones $\omega_n = n\Omega$ y $\Omega = \frac{2\pi}{NT_s}$ se deduce que:

$$\omega_{\frac{N}{2}} = \frac{N}{2}\Omega = \frac{N}{2}\frac{2\pi}{NT_s} = \frac{\pi}{T_s} = \pi f_s = f_{nq}$$

Luego la frecuencia más grande que se puede evaluar con la DFT esta en $\omega_{\frac{N}{2}}$ y equivale a la frecuencia de Nyquist f_{nq}. Por lo tanto, cualquier frecuencia mayor a $\omega_{\frac{N}{2}}$ no puede ser evaluada por la DFT ya que se produce *aliasing*.

Para N un número impar:

$$\omega_{\frac{N-1}{2}} = \frac{N-1}{2}\Omega = \frac{N-1}{2}\frac{2\pi}{NT_s} = \frac{N-1}{N}\frac{\pi}{T_s} = \frac{N-1}{N}\pi f_s = \frac{N-1}{N}f_{nq}$$

La frecuencia $\omega_{\frac{N}{2}}$ no existe y la frecuencia $\omega_{\frac{N-1}{2}}$ es la frecuencia más grande que la DFT puede representar; es decir, cualquier frecuencia más grande a partir de $\omega_{\frac{N+1}{2}}$ no se puede evaluar ya que se produce *aliasing*.

Como se puede observar, la DFT respeta naturalmente el teorema de muestreo con la frecuencia de Nyquist; al igual que es responsabilidad del usuario respetar este teorema, cuando se obtienen las muestras x_j de $x(t)$ para evitar el fenómeno del *aliasing*.

Por otra parte, la implementación real en dispositivos para el procesamiento digital de señales, puede ser muy difícil debido al manejo de números complejos en la DFT y la IDFT. No obstante, en el caso del análisis, los datos más representativos del dominio de la frecuencia, son las magnitudes de los valores obtenidos por la DFT $|X_n|$; en cuyo caso, estas magnitudes se podrían obtener a partir de las partes reales e imaginarias de X_n, sin necesidad de realizar ninguna operación con números complejos. Sin embargo, en el caso de la síntesis, cuando se pretende regresar a los datos x_j con la IDFT; si se necesitan operaciones de números complejos, siempre y cuando, los valores x_j también sean complejos. En caso contrario, se puede usar la versión de la IDFT para números reales que se deduce de (2.5) como sigue (Cara, 2012-2013):

Considere los valores obtenidos con la DFT X_n de un conjunto de datos reales x_j, para $n, j = 0, 1, 2, ..., N-1$ con N un número par. Teniendo en cuenta las propiedades antes vistas, la IDFT de esos valores se puede expresar como:

$$
\begin{aligned}
x_j &= \frac{1}{N}\sum_{n=0}^{N-1} X_n e^{i\frac{2\pi jn}{N}} \\
&= \frac{1}{N}X_0 + \frac{1}{N}X_{\frac{N}{2}}\cos(\pi j) + \frac{1}{N}\sum_{n=1}^{\frac{N}{2}-1}\left(X_n e^{i\frac{2\pi jn}{N}} + X_{N-n}e^{i\frac{2\pi j(N-n)}{N}}\right) \\
&= \frac{1}{N}X_0 + \frac{1}{N}X_{\frac{N}{2}}\cos(\pi j) + \frac{1}{N}\sum_{n=1}^{\frac{N}{2}-1}\left(X_n e^{i\frac{2\pi jn}{N}} + X_n^* e^{-i\frac{2\pi jn}{N}}\right) \qquad (2.8)
\end{aligned}
$$

Donde X_n^* es el conjugado de X_n. Definiendo a $X_n = z_n + iy_n$ se tiene que:

$$
\begin{aligned}
X_n e^{i\frac{2\pi jn}{N}} + X_n^* e^{-i\frac{2\pi jn}{N}} &= (z_n + iy_n)\left(\cos\left(\frac{2\pi jn}{N}\right) + i\sin\left(\frac{2\pi jn}{N}\right)\right) \\
&\quad + (z_n - iy_n)\left(\cos\left(\frac{2\pi jn}{N}\right) - i\sin\left(\frac{2\pi jn}{N}\right)\right) \\
&= 2z_n\cos\left(\frac{2\pi jn}{N}\right) - 2y_n\sin\left(\frac{2\pi jn}{N}\right) \qquad (2.9)
\end{aligned}
$$

Sustituyendo (2.9) en (2.8) se obtiene:

$$
\begin{aligned}
x_j &= \frac{1}{N}X_0 + \frac{1}{N}X_{\frac{N}{2}}\cos(\pi j) + \frac{1}{N}\sum_{n=1}^{\frac{N}{2}-1}\left(2z_n\cos\left(\frac{2\pi jn}{N}\right) - 2y_n\sin\left(\frac{2\pi jn}{N}\right)\right) \\
&= \sum_{n=0}^{\frac{N}{2}}\left(a_n\cos\left(\frac{2\pi jn}{N}\right) + b_n\sin\left(\frac{2\pi jn}{N}\right)\right) \qquad (2.10)
\end{aligned}
$$

Donde:

- $a_0 = \frac{1}{N}X_0 = \frac{1}{N}z_0, \qquad b_0 = 0$
- $a_{\frac{N}{2}} = \frac{1}{N}X_{\frac{N}{2}} = \frac{1}{N}z_{\frac{N}{2}}, \quad b_{\frac{N}{2}} = 0$
- $a_n = \frac{2}{N}z_n, \qquad b_n = -\frac{2}{N}y_n, \qquad n = 1, 2, ..., \frac{N}{2} - 1$

De manera similar, para N un número impar, se puede llegar a la siguiente conclusión:

$$
\begin{aligned}
x_j &= \frac{1}{N}\sum_{n=0}^{N-1} X_n e^{i\frac{2\pi jn}{N}} \\
&= \frac{1}{N}X_0 + \frac{1}{N}\sum_{n=1}^{\frac{N-1}{2}} \left(X_n e^{i\frac{2\pi jn}{N}} + X_{N-n} e^{i\frac{2\pi j(N-n)}{N}} \right) \\
&= \frac{1}{N}X_0 + \frac{1}{N}\sum_{n=1}^{\frac{N-1}{2}} \left(X_n e^{i\frac{2\pi jn}{N}} + X_n^* e^{-i\frac{2\pi jn}{N}} \right) \\
&= \frac{1}{N}X_0 + \frac{1}{N}\sum_{n=1}^{\frac{N-1}{2}} \left(2z_n \cos\left(\frac{2\pi jn}{N}\right) - 2y_n \sin\left(\frac{2\pi jn}{N}\right) \right) \\
&= \sum_{n=0}^{\frac{N-1}{2}} \left(a_n \cos\left(\frac{2\pi jn}{N}\right) + b_n \sin\left(\frac{2\pi jn}{N}\right) \right) \qquad (2.11)
\end{aligned}
$$

Donde:

- $a_0 = \frac{1}{N}X_0 = \frac{1}{N}z_0, \qquad b_0 = 0$
- $a_n = \frac{2}{N}z_n, \qquad b_n = -\frac{2}{N}y_n, \qquad n = 1, 2, ..., \frac{N-1}{2}$

De esta manera, las ecuaciones (2.10) y (2.11) son las versiones de la IDFT para números reales, cuando N es un número par o impar respectivamente. Por lo tanto, cuando los datos x_j son números reales, el análisis y la síntesis no requieren operaciones con números complejos, cuando se utilizan la DFT y la IDFT.

Nótese que la DFT y su inversa tienen ventajas importantes respecto a la transformada de Fourier en tiempo continuo. Una de ellas es que la DFT requiere únicamente un conjunto de datos x_j; aunque estos datos pueden provenir de una señal continua $x(t)$, no es un requisito que los datos x_j tengan relación con la señal $x(t)$, más aún, los datos x_j pueden ser números aleatorios y sin embargo, la DFT puede ser calculada. Por otro lado, si los datos x_j resultan del muestreo de la señal $x(t)$; no es una condición que la señal $x(t)$ sea periódica y si lo es, no es un requisito conocer el periodo para calcular la DFT. En comparación con la transformada de Fourier en tiempo continuo; la cual es utilizada para señales periódicas, con periodo definido y continuas en el tiempo. Para más información acerca de este tema refiérase a Smith (2007).

2.2. Modelo matemático del PMSM y rizado de par.

El PMSM es uno de los motores más eficientes, confiables y de costo competitivo que existen actualmente. Además, los PMSM's disponen de una de las relaciones más altas de par a pérdidas. Asimismo tienen aplicación natural en los sistemas de posicionamiento de dinámica rápida y en máquinas herramienta (Mattavelli et al., 2005). Sin embargo, estos motores tienen una desventaja importante conocida como rizado de par; el cual, se debe principalmente a la magnetización y fabricación de sus devanados. El rizado de par deteriora la velocidad y posición de las trayectorias en bajas velocidades y en altas velocidades causa un incremento de ruido acústico y pérdidas adicionales de energía (Štumberger et al., 2006).

Algunas causas del rizado de par se pueden explicar observando el modelo matemático del PMSM. Asimismo, es posible esclarecer la naturaleza periódica del rizado de par a partir de la relación entre el par generado y la posición del rotor. En ese sentido, la presente sección es utilizada para hacer una revisión del modelo matemático del PMSM y algunas consideraciones adicionales estudiadas en Gómez-Espinosa et al. (2013).

Primeramente, la relación ente las tres corrientes de un PMSM $\boldsymbol{i}_{abcs}$, sus inductancias mutuas $\boldsymbol{L}_s$, la atribución del flujo magnético del rotor $\boldsymbol{\Psi}_m$ y el flujo magnético total de las tres fases $\boldsymbol{\Psi}_{abcs}$ esta dada por:

$$\boldsymbol{\Psi}_{abcs} = \boldsymbol{L}_s \boldsymbol{i}_{abcs} + \boldsymbol{\Psi}_m \quad \in \mathbb{R}^3 \tag{2.12}$$

En forma matricial, la ecuación (2.12) queda como:

$$\begin{bmatrix} \Psi_{as} \\ \Psi_{bs} \\ \Psi_{cs} \end{bmatrix} = \begin{bmatrix} L_{asas} & L_{asbs} & L_{ascs} \\ L_{bsas} & L_{bsbs} & L_{bscs} \\ L_{csas} & L_{csbs} & L_{cscs} \end{bmatrix} \begin{bmatrix} i_{as} \\ i_{bs} \\ i_{cs} \end{bmatrix} + \begin{bmatrix} \Psi_{asm} \\ \Psi_{bsm} \\ \Psi_{csm} \end{bmatrix} \tag{2.13}$$

A partir del flujo magnético total $\boldsymbol{\Psi}_{abcs}$ y las resistencias de las bobinas en el estator $\boldsymbol{r}_s$, se definen los voltajes del motor como:

$$\boldsymbol{u}_{abcs} = \boldsymbol{r}_s \boldsymbol{i}_{abcs} + \frac{d\,\boldsymbol{\Psi}_{abcs}}{dt} \quad \in \mathbb{R}^3 \tag{2.14}$$

Reescribiendo en forma matricial, la ecuación (2.14) es:

$$\begin{bmatrix} u_{as} \\ u_{bs} \\ u_{cs} \end{bmatrix} = \begin{bmatrix} r_s & 0 & 0 \\ 0 & r_s & 0 \\ 0 & 0 & r_s \end{bmatrix} \begin{bmatrix} i_{as} \\ i_{bs} \\ i_{cs} \end{bmatrix} + \begin{bmatrix} \frac{d\Psi_{as}}{dt} \\ \frac{d\Psi_{bs}}{dt} \\ \frac{d\Psi_{cs}}{dt} \end{bmatrix} \tag{2.15}$$

Asumiendo que las bobinas en el estátor están separadas 120^{o} y que los flujos magnéticos Ψ_{asm}, Ψ_{asm}, Ψ_{asm} establecidos por el imán permanente del rotor, son funciones periódicas del desplazamiento angular eléctrico θ_r con magnitud Ψ_m, se puede establecer que los flujos magnéticos son funciones sinusoidales, de la siguiente manera:

$$\boldsymbol{\Psi}_m = \begin{bmatrix} \Psi_{asm} \\ \Psi_{bsm} \\ \Psi_{csm} \end{bmatrix} = \begin{bmatrix} \Psi_m \sin(\theta_r) \\ \Psi_m \sin(\theta_r - \frac{2}{3}\pi) \\ \Psi_m \sin(\theta_r + \frac{2}{3}\pi) \end{bmatrix} \tag{2.16}$$

De las ecuaciones (2.12) y (2.14) y asumiendo que $\boldsymbol{L}_s$ es constante, se obtiene:

$$\boldsymbol{u}_{abcs} = \boldsymbol{r}_s \boldsymbol{i}_{abcs} + \boldsymbol{L}_s \frac{d\,\boldsymbol{i}_{abcs}}{dt} + \frac{d\,\boldsymbol{\Psi}_m}{dt} \tag{2.17}$$

Al definir la velocidad angular eléctrica como $\omega_r = \frac{d\theta_r}{dt}$, se tiene que:

$$\frac{d\,\boldsymbol{\Psi}_m}{dt} = \Psi_m \begin{bmatrix} \omega_r \cos(\theta_r) \\ \omega_r \cos(\theta_r - \frac{2}{3}\pi) \\ \omega_r \cos(\theta_r + \frac{2}{3}\pi) \end{bmatrix} \tag{2.18}$$

Además, en la forma de Cauchy, usando $\boldsymbol{L}_s^{-1}$ como la inversa de $\boldsymbol{L}_s$; la ecuación (2.17) se reescribe como:

$$\frac{d\,\boldsymbol{i}_{abcs}}{dt} = -\boldsymbol{L}_s^{-1}\boldsymbol{r}_s\boldsymbol{i}_{abcs} - \boldsymbol{L}_s^{-1}\frac{d\,\boldsymbol{\Psi}_m}{dt} + \boldsymbol{L}_s^{-1}\boldsymbol{u}_{abcs} \tag{2.19}$$

Por otro lado, de acuerdo a la segunda ley de Newton, el modelo matemático del sistema mecánico de un motor eléctrico es:

$$T_e - B_m\omega_{rm} - T_L = J\frac{d^2\theta_{rm}}{dt^2} \tag{2.20}$$

$$\frac{d\omega_{rm}}{dt} = \frac{1}{J}(T_e - B_m\omega_{rm} - T_L) \tag{2.21}$$

donde:

- T_e es el torque electromagnético generado.
- B_m es la constante de fricción viscosa del motor.
- ω_{rm} es la velocidad angular mecánica del motor.
- T_L es el torque de carga.
- J es el momento de inercia del rotor.
- θ_{rm} es la posición angular mecánica del motor.

Según Lyshevski (2000), el torque electromagnético generado T_e está definido a partir de la energía magnética permanente W_{PM} y la co-energía W_c. Entonces, si P es el número de polos del motor, se tiene que:

$$T_e = \frac{P}{2}\frac{\partial W_c}{\partial \theta_r} \tag{2.22}$$

Donde, la co-energía W_c se define como:

$$W_c = \frac{1}{2}\begin{bmatrix} i_{as} & i_{bs} & i_{cs} \end{bmatrix}\mathbf{L}_s\begin{bmatrix} i_{as} \\ i_{bs} \\ i_{cs} \end{bmatrix} + \begin{bmatrix} i_{as} & i_{bs} & i_{cs} \end{bmatrix}\begin{bmatrix} \Psi_m \sin(\theta_r) \\ \Psi_m \sin(\theta_r - \frac{2}{3}\pi) \\ \Psi_m \sin(\theta_r + \frac{2}{3}\pi) \end{bmatrix} + W_{PM} \tag{2.23}$$

Entonces, de las ecuaciones (2.22) y (2.23) el torque electromagnético generado queda definido como:

$$T_e = \frac{P\Psi_m}{2}(i_{as}\cos(\theta_r) + i_{bs}\cos(\theta_r - \frac{2}{3}\pi) + i_{cs}\cos(\theta_r + \frac{2}{3}\pi)) \tag{2.24}$$

Sustituyendo (2.24) en (2.21):

$$\frac{d\omega_{rm}}{dt} = \frac{P\Psi_m}{2J}(i_{as}\cos(\theta_r) + i_{bs}\cos(\theta_r - \frac{2}{3}\pi) + i_{cs}\cos(\theta_r + \frac{2}{3}\pi)) - \frac{B_m}{J}\omega_{rm} - \frac{1}{J}T_L \tag{2.25}$$

Asimismo, usando la velocidad angular eléctrica ω_r y el desplazamiento angular eléctrico θ_r en la ecuación (2.25) y tomando en cuenta que $\omega_{rm} = \frac{2}{P}\omega_r$ y $\theta_{rm} = \frac{2}{P}\theta_r$, se obtiene la siguiente ecuación:

$$\frac{d\omega_r}{dt} = \frac{P^2\Psi_m}{4J}(i_{as}\cos(\theta_r) + i_{bs}\cos(\theta_r - \frac{2}{3}\pi) + i_{cs}\cos(\theta_r + \frac{2}{3}\pi)) - \frac{B_m}{J}\omega_r - \frac{P}{2J}T_L \quad (2.26)$$

A partir de la ecuación (2.26) queda explícita la relación entre la posición angular, la velocidad angular eléctrica y la corriente en cada una de las fases; con lo cual, se observa que para realizar el control de velocidad del PMSM se deben regular los voltajes o las corrientes aplicadas en las bobinas del estátor.

Por otro lado, el torque electromagnético máximo se alcanza cuando el motor es alimentado con un conjunto de corrientes balanceadas; es decir:

$$\begin{aligned} \boldsymbol{i}_{abcsMax}(t) &= \begin{bmatrix} i_{asMax}(t) \\ i_{bsMax}(t) \\ i_{csMax}(t) \end{bmatrix} = \begin{bmatrix} \sqrt{2}i_M\cos(\omega_r t) \\ \sqrt{2}i_M\cos(\omega_r t - \frac{2}{3}\pi) \\ \sqrt{2}i_M\cos(\omega_r t + \frac{2}{3}\pi) \end{bmatrix} \\ &= \begin{bmatrix} \sqrt{2}i_M\cos(\omega_e t) \\ \sqrt{2}i_M\cos(\omega_e t - \frac{2}{3}\pi) \\ \sqrt{2}i_M\cos(\omega_e t + \frac{2}{3}\pi) \end{bmatrix} = \begin{bmatrix} \sqrt{2}i_M\cos(\theta_r) \\ \sqrt{2}i_M\cos(\theta_r - \frac{2}{3}\pi) \\ \sqrt{2}i_M\cos(\theta_r + \frac{2}{3}\pi) \end{bmatrix} \end{aligned} \quad (2.27)$$

Y el torque electromagnético máximo estaría dado por:

$$T_{eMax} = \frac{P\Psi_m}{2}\sqrt{2}i_M(\cos^2(\theta_r) + \cos^2(\theta_r - \frac{2}{3}\pi) + \cos^2(\theta_r + \frac{2}{3}\pi)) = \frac{3P\Psi_m}{2\sqrt{2}}i_M \quad (2.28)$$

En donde, para producir las corrientes especificadas, los voltajes balanceados de las tres fases son:

$$\boldsymbol{u}_{abcsMax}(t) = \begin{bmatrix} u_{asMax}(t) \\ u_{bsMax}(t) \\ u_{csMax}(t) \end{bmatrix} = \begin{bmatrix} \sqrt{2}u_M\cos(\theta_r) \\ \sqrt{2}u_M\cos(\theta_r - \frac{2}{3}\pi) \\ \sqrt{2}u_M\cos(\theta_r + \frac{2}{3}\pi) \end{bmatrix} \quad (2.29)$$

Es común simplificar el modelo matemático del PMSM mediante una transformación al espacio $qd0$ para el cuadro de referencia del rotor (Krishnan, 2009). Luego, el modelo matemático del PMSM en el cuadro de referencia del rotor esta dado por (Gómez-Espinosa et al., 2013):

$$\begin{bmatrix} \frac{di_{qs}^r}{dt} \\ \frac{di_{ds}^r}{dt} \\ \frac{di_{0s}^r}{dt} \\ \frac{d\omega_r}{dt} \\ \frac{d\theta_r}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{r_s}{L} & 0 & 0 & -\frac{\Phi_m}{L} & 0 \\ 0 & -\frac{r_s}{L} & 0 & 0 & 0 \\ 0 & 0 & -\frac{r_s}{L_{ls}} & 0 & 0 \\ \frac{3P^2\Psi_m}{8J} & 0 & 0 & -\frac{B_m}{J} & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} i_{qs}^r \\ i_{ds}^r \\ i_{0s}^r \\ \omega_r \\ \theta_r \end{bmatrix} + \begin{bmatrix} -i_{qs}^r\omega_r \\ i_{ds}^r\omega_r \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \frac{1}{L} & 0 & 0 \\ 0 & \frac{1}{L} & 0 \\ 0 & 0 & \frac{1}{L_{ls}} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_{qs}^r \\ u_{ds}^r \\ u_{0s}^r \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{P}{2J} \\ 0 \end{bmatrix} T_L \tag{2.30}$$

Asimismo, con el modelo reducido del PMSM representado por la ecuación (2.30); las corrientes para regular la velocidad angular y garantizar condiciones de funcionamiento balanceadas son:

$$i_{qs}^r(t) = \sqrt{2}i_M, \quad i_{ds}^r(t) = 0, \quad i_{0s}^r(t) = 0 \tag{2.31}$$

Por tanto, asumiendo que las inductancias son despreciables, los voltajes aplicados deben ser:

$$u_{qs}^r(t) = \sqrt{2}u_M, \quad u_{ds}^r(t) = 0, \quad u_{0s}^r(t) = 0 \tag{2.32}$$

De esta manera, el control del PMSM se puede desarrollar modificando los voltajes $u_{qs}^r(t)$ y $u_{ds}^r(t)$. En donde, los valores de las ecuaciones (2.32) y (2.31) son los necesarios para obtener el par electromagnético máximo y un rendimiento balanceado.

Por otra parte, observando el modelo matemático del PMSM obtenido hasta el momento, no se pueden distinguir las causas del rizado de par. Esto se debe a que se establecieron

ciertas consideraciones durante el modelado; las cuales, no se pueden asegurar. Ejemplo de ello son las imperfecciones en la construcción del PMSM; estas imperfecciones causarían que los flujos magnéticos no sean perfectamente funciones sinusoidales y en consecuencia las fuerzas electromotrices $\boldsymbol{e}_{abcs}$ no son funciones coseno. Sin embargo, estas fuerzas electromotrices son periódicas con un valor pico E_p, es decir:

$$\boldsymbol{e}_{abcs} = \frac{d\,\Psi_m}{dt} = E_p \omega_r \begin{bmatrix} f_{as}(\theta_r) \\ f_{bs}(\theta_r - \frac{2}{3}\pi) \\ f_{cs}(\theta_r + \frac{2}{3}\pi) \end{bmatrix} \tag{2.33}$$

Donde $f_{as}(\theta_r)$, $f_{bs}(\theta_r)$ y $f_{cs}(\theta_r)$ son funciones periódicas con la misma forma y con magnitud máxima ± 1.

Asimismo, el par electromagnético es definido como:

$$T_e = [e_{as} ias + e_{bs} ibs + e_{cs} ics]\frac{1}{\omega_r} \tag{2.34}$$

Sustituyendo (2.33) en (2.34):

$$T_e = E_p[f_{as}(\theta_r) i_{as} + f_{bs}(\theta_r - \frac{2}{3}\pi) i_{bs} + f_{cs}(\theta_r + \frac{2}{3}\pi) i_{cs}] \tag{2.35}$$

Luego, el par electromagnético es periódico con θ_r. Además, para producir un par electromagnético constante, las formas de onda de las corrientes deberían ser calculadas en base a $f_{as}(\theta_r)$, $f_{bs}(\theta_r)$ y $f_{cs}(\theta_r)$.

Más aún, las inductancias de las bobinas del PMSM pueden ser diferentes en algunos instantes de tiempo, lo que ocasiona que la fuerza electromagnética varíe (Hanselman, 2003). Cuando las inductancias de las bobinas no son constantes, las fuerzas electromagnéticas también son modificadas. Esto es, tomando la ecuación (2.12), se tiene que:

$$\begin{aligned} \boldsymbol{e}_{abcs} &= \frac{d\,\boldsymbol{\Psi}_{abcs}}{dt} = \frac{d}{dt}[\boldsymbol{L}_s \boldsymbol{i}_{abcs} + \boldsymbol{\Psi}_m] \\ &= \boldsymbol{L}_s \frac{d\,\boldsymbol{i}_{abcs}}{dt} + \boldsymbol{i}_{abcs} \frac{d\boldsymbol{L}_s}{dt} + \frac{d\,\boldsymbol{\Psi}_m}{dt} \end{aligned} \tag{2.36}$$

En la ecuación (2.36) se puede ver que el rizado de par también puede ser producido por las variaciones de la inductancia.

Por otra parte, desde un punto de vista macroscópico, el par electromagnético producido por un PMSM es (Hanselman, 2003):

$$T_e = \frac{1}{2} i^2 \frac{dL}{d\theta} - \frac{1}{2N^2} \Psi_m^2 \frac{dR}{d\theta} + i \frac{d\Psi_m}{d\theta} \tag{2.37}$$

Con la ecuación (2.37) se observa otra relación del par electromagnético con la posición angular θ_r; además de una dependencia con las variaciones de las inductancias y resistencias de las bobinas en el estator.

Todos estos factores entre otros, provocan el rizado de par en PMSM's. En estos ejemplos se puede distinguir que el par electromagnético tiene una dependencia natural con la posición angular θ_r. Además, la posición angular es perturbada periódicamente cada que el rotor, siendo un imán permanente, intenta alinearse con los dientes o polos del PMSM. Adicionalmente ocurre una variación extra debido al flujo magnético del rotor que viaja a través de las bobinas del estator, las cuales pueden tener reluctancia variable como se mencionó anteriormente. Luego, el par electromagnético es perturbado periódicamente con la posición angular y por tanto, el rizado de par es una perturbación periódica natural en los PMSM's.

2.3. Control por campo orientado (FOC).

A inicios de 1970, la invención del control por campo orientado (FOC por sus siglas en inglés) demostró que un motor trifásico puede ser controlado como un motor de corriente directa con una fuente separada de CD; lo cual, incrementó la eficiencia de los equipos de control para máquinas trifásicas de CA (Adhavan et al., 2011). Este método hace posible el control independiente del campo electromagnético y el par, mediante la manipulación de las cantidades correspondientes al campo orientado; así, este sistema de control se adapta para cualquier variación de referencia o perturbación tan rápido como podría operar un motor de CD (Sathikumar y Vithayathil, 1984).

Ya que las corrientes de un motor trifásico están directamente relacionadas con el par electromagnético generado, en el FOC se realiza una transformación de las tres corrientes del motor para obtener dos variables separadas i_{sq} e i_{sd}; las cuales, representan la componente que genera el par eléctrico y la componente que produce el flujo de magnetización respectivamente. Estas nuevas componentes están alineadas con el flujo magnético del rotor y el

objetivo del FOC es dirigirlas hacia sus valores apropiados, en el marco de referencia giratorio. Posteriormente, a través de una transformación inversa se obtienen las tres corrientes que alimentan al motor.

En otras palabras, el FOC es un método de control que se encarga de controlar las corrientes en el estator de un motor trifásico, mediante la manipulación de un vector con dos componentes I_q e I_d que representan las tres corrientes del motor. Este vector, se encuentra en el marco de referencia giratorio (d, q) y esta alineado con el flujo magnético del rotor (Marulanda y Herrera, 2010).

De esta manera, para entender claramente el cambio de coordenadas que se utiliza en el FOC, a continuación se hace una revisión al trabajo de Akin y Bhardwaj (2013).

Asumiendo que i_a, i_b, i_c son las corrientes instantáneas en las fases del motor; el vector complejo de la corriente en el estator $\boldsymbol{i}_s$ esta definido por:

$$\boldsymbol{i}_s = i_a + \alpha i_b + \alpha^2 i_c \tag{2.38}$$

Donde $\alpha = e^{i\frac{2}{3}\pi}$ y $\alpha^2 = e^{i\frac{4}{3}\pi}$, representan los operadores en el plano complejo. Así, el vector de corriente del estator, se gráfica en el plano complejo como se muestra en la Figura 2.1.

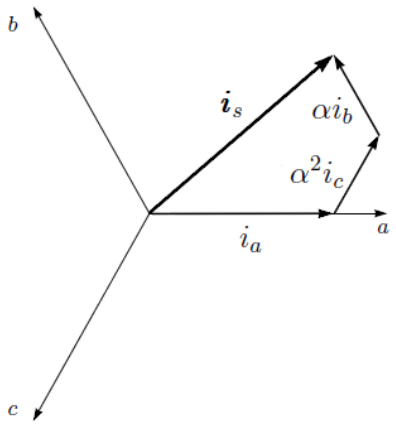

Figura 2.1: Vector de la corriente en el estator y sus componentes en (a, b, c) (Akin y Bhardwaj, 2013).

Donde (a, b, c) son los ejes de las tres fases del sistema. Así, el vector $\boldsymbol{i}_s$ espacial

de corriente, representa el sistema sinusoidal trifásico; el cual, requiere ser transformado a un sistema de dos coordenadas invariante en el tiempo. Para ello, se utilizan dos transformaciones. La primera, la transformación de Clarke, transforma el sistema (a, b, c) en un sistema de dos coordenadas (α, β) variante en el tiempo. La segunda, la transformación de Park, transforma el sistema (α, β) en un sistema de dos coordenadas (d, q) invariante en el tiempo.

Es decir, el espacio vectorial (a, b, c) puede ser proyectado a otro cuadro de referencia con solo dos ejes ortogonales llamados (α, β). Asimismo, si se asume que el eje a y el eje α tienen la misma dirección, se tiene el sistema de coordenadas (α, β) mostrado en la Figura 2.2.

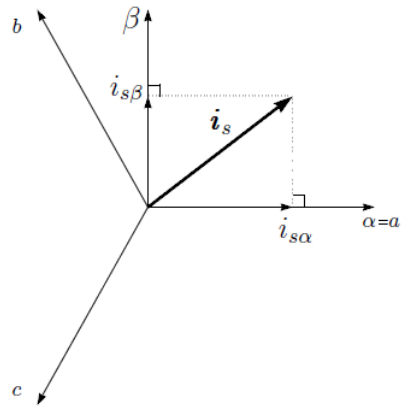

Figura 2.2: Vector de la corriente en el estator y sus componentes en (α, β) (Akin y Bhardwaj, 2013).

Luego, las componentes en el espacio (α, β), del sistema de tres fases, se calculan mediante la transformada de Clark como:

$$i_{s\alpha} = i_a \tag{2.39}$$

$$i_{s\beta} = \frac{1}{\sqrt{3}} i_a + \frac{2}{\sqrt{3}} i_b \tag{2.40}$$

De tal manera que, las dos nuevas corrientes $i_{s\alpha}$ e $i_{s\beta}$ siguen siendo dependientes del tiempo y la velocidad del motor.

Ahora, se modifica el sistema ortogonal de dos fases (α, β) en un cuadro de referencia rotacional (d, q). Este nuevo sistema de coordenadas gira con el flujo magnético del rotor;

es decir, se considera que el eje d esta alineado con el flujo magnético del rotor.

De esta manera, el vector de corrientes trifásico $\boldsymbol{i}_s$, se representa en el nuevo sistema de coordenadas (d, q), como se muestra en la Figura 2.3.

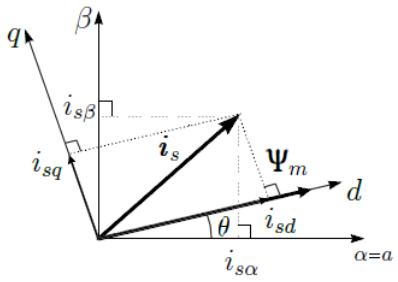

Figura 2.3: Vector de la corriente en el estator y sus componentes en (d, q) (Akin y Bhardwaj, 2013).

Donde, θ es la posición del flujo magnético del rotor. Así, las componentes del flujo magnético i_{sd} y el torque eléctrico generado i_{sq}, del vector de corriente del estator $\boldsymbol{i}_s$, se calculan mediante la transformada de Park como:

$$i_{sd} = i_{s\alpha}\cos(\theta) + i_{s\beta}\sin(\theta) \tag{2.41}$$

$$i_{sq} = -i_{s\alpha}\sin(\theta) + i_{s\beta}\cos(\theta) \tag{2.42}$$

Estas componentes dependen del vector de corriente en el espacio (α, β) y de la posición del flujo magnético del rotor. Sin embargo, si se conoce la posición del flujo magnético del rotor en cada instante, por su proyección, la componente (d, q) es constante. Luego, las dos corrientes i_{sd} e i_{sq} ahora se convierten en cantidades de CD invariantes en el tiempo. Asimismo, el control de par electromagnético en un motor trifásico, se logra mediante la manipulación de la variable i_{sq}; la cual, es independiente de la componente de flujo magnético i_{sd}.

En resumen, el método de control por campo orientado para un motor trifásico, se puede implementar como se muestra en el diagrama de bloques de la Figura 2.4. En donde, se desea alcanzar una velocidad deseada ω_{Ref}, a partir de un controlador tradicional PI y el método FOC.

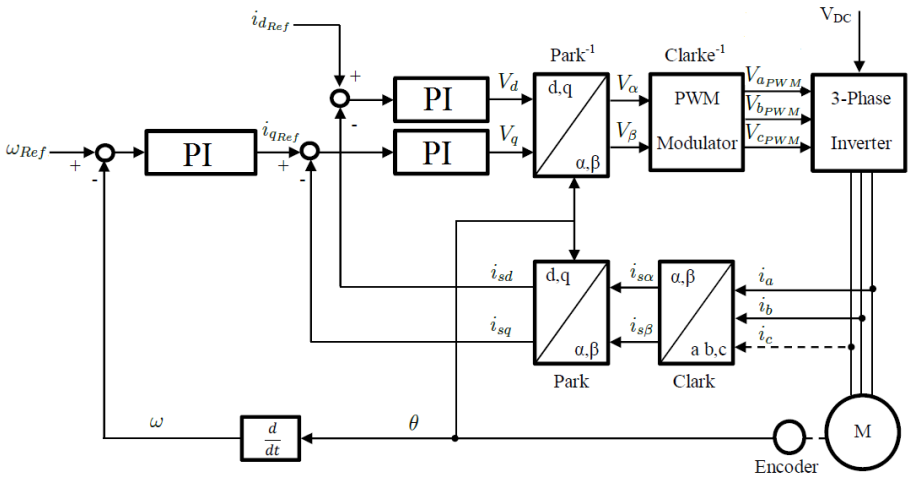

Figura 2.4: Control por campo orientado de velocidad para un motor trifásico (Akin y Bhardwaj, 2013).

El método únicamente requiere la medición de dos corrientes del motor y su posición angular. A partir de las corrientes adquiridas, se calculan la transformada de Clark y Park para obtener las corrientes i_{sq} e i_{sd}. El FOC tradicional consta de un lazo de control maestro y dos lazos esclavos; todos ellos utilizan controladores PI's. Los controladores esclavos son destinados a controlar las corrientes i_{sq} e i_{sd}; en donde, la referencia $i_{d_{Ref}}$, es una constante cuyo valor normalmente esta relacionado con el par máximo del motor y la referencia $i_{q_{Ref}}$ es proporcionada en linea por el controlador maestro. Es decir, el controlador PI de velocidad, calcula la corriente i_{sq} necesaria para alcanzar la velocidad deseada ω_{Ref}; este controlador, utiliza la velocidad del motor ω, normalmente obtenida con la derivada numérica de la posición angular del motor. La salida de los controladores esclavos, representan los voltajes en el espacio (d, q) necesarios para llegar a la velocidad deseada. Mediante las transformadas inversas de Clark y Park se obtienen los valores de voltaje en el espacio (a, b, c) que deben ser proporcionados al motor. Es común que estos voltajes, en una aplicación real, se suministren al motor a través de un inversor tirfásico; el cual, a partir de una fuente de voltaje de CD obtiene los voltajes de CA para las bobinas del motor. Luego, el lazo de control se cierra y el motor trifásico puede ser controlado como un motor de corriente directa con una fuente separada de CD y un controlador convencional PI. En PMSM's, la posición del rotor es igual a la posición del flujo magnético $\mathbf{\Psi}_m$; entonces, θ es medida con un sensor de posición angular.

Asimismo, el par máximo se alcanza con $i_{d_{Ref}} = 0$ (Adhavan et al., 2011).

2.4. Redes neuronales artificiales.

El principal propósito de todos los sistemas neuronales es el control centralizado de varias funciones biológicas; algunas de ellas, responsables del abastecimiento de energía. Con base en el estudio de las neuronas biológicas reales, se ha intentado imitar su comportamiento para su aplicación en diferentes áreas como: la biología, finanzas, manufactura, medicina, la milicia, entre otras. Los primeros experimentos, alrededor de 1960, estaban basados en redes neuronales artificiales (RNA) elementales, como el Perceptron, la red Adaline y Matrices de aprendizaje. A partir de las cuales, se comenzaron a desarrollar redes mas complejas (Hilera y Martínez, 1995).

Una razón por la cual las redes neuronales artificiales pueden ser utilizadas en áreas tan diferentes y con buenos resultados, es su capacidad de aprender y memorizar, tal cuál lo hacen las neuronas biológicas reales. Un ejemplo básico del uso de las RNA's, es el que se muestra en Freeman y Skapura (1993); en donde se plantea un problema de identificación de caracteres mediante un dispositivo electrónico, sobre el cual, se escribe el caracter a ser identificado y se obtienen las coordenadas de todos los puntos que fueron tocados. El objetivo principal es que el dispositivo interprete correctamente el caracter que fue escrito. La RNA en este caso, es entrenada para aprender y memorizar, mediante pruebas y errores, diversas formas en las que, los usuarios pueden escribir el mismo caracter con diferentes tamaños, orientaciones y formas. De esta manera, al finalizar el entrenamiento, la red neuronal artificial es capaz de identificar con precisión los caracteres, por muy distorsionados que se puedan escribir.

Ya sea para aplicaciones relativamente sencillas como la anterior, o para aplicaciones más complicadas como controladores de sistemas no lineales , el alcance de las redes neuronales artificiales es bastante grande y hasta le fecha se siguen haciendo diversos estudios en este tema.

Cualquier modelo de red neuronal consta de dispositivos elementales de proceso: las neuronas. A partir de ellas, se pueden generar representaciones específicas, de tal forma que un estado conjunto de ellas puede significar una letra, un número o cualquier otro objeto

(Gómez, 1999).

Según Gómez (1999), la neurona artificial pretende mimetizar las características más importantes de las neuronas biológicas. Cada i-ésima neurona está caracterizada en cualquier instante por un valor numérico denominado valor o estado de activación $a_i(t)$; asociado a cada unidad, existe una señal de salida f_i, que transforma el estado actual de activación en una señal de salida y_i. Dicha señal es enviada a través de los canales de comunicación unidireccionales a otras unidades de la red. En estos canales la señal se modifica de acuerdo a la sinapsis (el peso w_{ji}) asociada a cada uno de ellos, según una determinada regla. Las señales modulares que han llegado a la unidad j-ésima se combinan entre ellas, generando así la entrada total Net_j.

$$Net_j = \sum_i y_i w_{ji} \tag{2.43}$$

Una función de activación, F, determina el nuevo estado de activación $a_j(t+1)$ de la neurona, teniendo en cuenta la entrada total calculada y el anterior estado de activación $a_j(t)$ (ver Figura2.5).

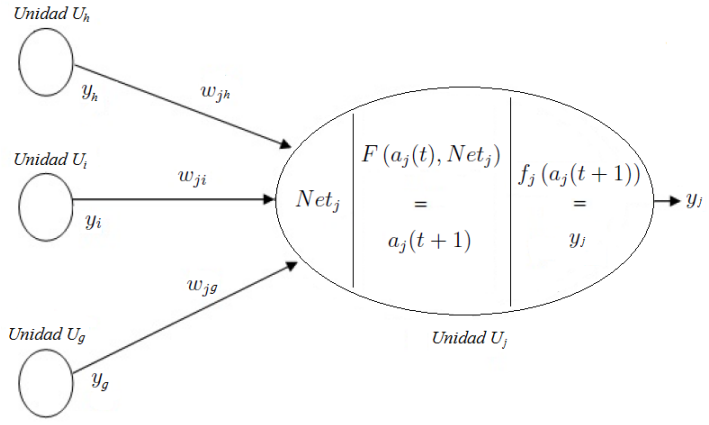

Figura 2.5: Esquema de una neurona artificial (modificado de Gómez (1999)).

Los pesos w_{ji} son los factores mas importantes en las redes neuronales; en éstos coeficientes se ve reflejado el aprendizaje y la memoria de una red neuronal, es decir, para que la red neuronal cumpla con su objetivo, debe tener los coeficientes de peso correctos. Es importante observar que entre más neuronas tenga la red, mayor es la cantidad de coeficientes

de peso y es más difícil encontrar los valores adecuados para cada coeficiente; por lo cual, los algoritmos para implementar las redes neuronales artificiales en un problema real, se basan en la modificación automática de los coeficientes de peso con relación a los errores cometidos por la red; de ésta forma, se auto-ajustarían sus coeficientes hasta obtener un margen de error aceptable.

2.5. Algoritmo de retropropagación.

Basada en una regla iterativa de cambio de pesos para RNA's sin unidades ocultas y con unidades de salida lineales, la regla del gradiente o regla delta se puede calcular como:

$$w(t+1)_i = w(t)_i + 2\eta\epsilon_k x_{ki} \tag{2.44}$$

donde:

- η es el coeficiente de aprendizaje (positivo).
- x_{ki} es la i-ésima componente del k-ésimo vector de entrenamiento.
- ϵ_k es la diferencia entre el valor obtenido y el valor deseado.
- $w(t)_i$ es el coeficiente de peso i-ésimo.

Esta regla se aplica en el método que Rumelhart, Hinton y Williams formalizaron para lograr que una red neuronal aprenda la asociación que existe entre los patrones de entrada a la misma y las clases correspondientes, utilizando más niveles de neuronas que los que utilizó Rosenblatt para desarrollar el Perceptron. El método es formalmente conocido como retropropagación (propagación del error hacia atrás) y es de los algoritmos más utilizados para el entrenamiento de redes neuronales, debido a que es un método iterativo, fácil de implementar y teóricamente sin limitaciones en cuanto a capas intermedias, entradas, salidas y número de neuronas (Gómez, 1999).

El método de retropropagación utiliza el error cuadrático medio, con el fin de reducirlo durante el proceso mediante un procedimiento iterativo. La hipótesis es, que con los valores adecuados en los pesos de la red, se alcanza un mínimo en la función del error cuadrático medio. Para ello, primeramente se calcula el valor del gradiente negativo de esta función

de error; en otras palabras, se obtiene la dirección en la que decrece el error cuadrado medio con respecto a los coeficientes o pesos de la red. De esta manera, se liga la función de error cuadrado con los pesos. Quedando, una serie de derivadas parciales del error respecto a los pesos de la red; las cuales, pueden ser calculadas numéricamente tomando muestras discretas de la función de error. El resultado se relaciona con la regla delta y se concluye con un método de 9 reglas para el entrenamiento de redes neuronales artificiales ante un único vector de entradas; las cuales, se listan a continuación (Gómez, 1999):

1. Se aplica el vector de entradas $X_p = (x_{p1}, x_{p2}, ..., x_{pN})^T$ a las unidades de entrada.

2. Se calculan los valores netos procedentes de las entradas para las unidades de la capa oculta:

$$neta_{pj}^{h} = \sum_{i=1}^{N} w_{ji}^{h} x_{pi} \tag{2.45}$$

3. Se calculan las salidas de la capa oculta:

$$i_{pj} = f_{j}^{h}(neta_{pj}^{h}) \tag{2.46}$$

4. Para la capa de salida. Se calculan los valores netos de las entradas para cada unidad:

$$neta_{pk}^{o} = \sum_{j=1}^{L} w_{kj}^{o} i_{pj} \tag{2.47}$$

5. Se calculan las salidas de la red:

$$o_{pk} = f_{k}^{o}(neta_{pk}^{o}) \tag{2.48}$$

6. Se calculan los términos de error para las unidades de salida:

$$\delta_{pk}^{o} = (y_{pk} - o_{pk}) f_{k}^{o'}(neta_{pk}^{o}) \tag{2.49}$$

7. Se calculan los términos de error para las unidades ocultas:

$$\delta_{pj}^{h} = f_{j}^{h'}(neta_{pj}^{h}) \sum_{k} \delta_{pj}^{o} w_{kj}^{o} \tag{2.50}$$

8. Se actualizan los pesos de la capa de salida:

$$w_{kj}^{o}(t+1) = w_{kj}^{o}(t) + \eta \delta_{pk}^{o} i_{pj} \tag{2.51}$$

9. Se actualizan los pesos de la capa oculta:

$$w_{ji}^{h}(t+1) = w_{ji}^{h}(t) + \eta \delta_{pj}^{h} x_{pi} \tag{2.52}$$

Por último, se recomienda calcular el error cuadrado medio mediante la ecuación (2.53) para asegurar que la red está aprendiendo correctamente. Cuando el error resulta suficientemente pequeño para todos los vectores de entrenamiento, el algoritmo se puede dar por concluido.

$$E_p = \frac{1}{2} \sum_{k=1}^{M} \delta_{pk}^{2} \tag{2.53}$$

2.6. Red neuronal artificial como controlador de sistemas SISO.

Dentro de las aplicaciones más importantes de las redes neuronales artificiales, están aquellas realizadas en el área de control automático. En esta área, se han diseñado diversas arquitecturas de RNA's para resolver problemas complejos de control. Un ejemplo de ello, se presenta en Ponce et al. (2004). En donde, se muestra una aplicación de las RNA's como regulador auto ajustable para sistemas de control de una entrada - una salida (SISO por sus siglas en inglés). En la presente sección se hace una revisión a su trabajo.

La Figura 2.6 muestra el esquema de control general, con una RNA como regulador. Siendo $y(t)$ la salida de la planta, $y_r(t)$ la referencia, $u(t)$ la señal de control o salida de la RNA y $e_y(t)$ el error del sistema.

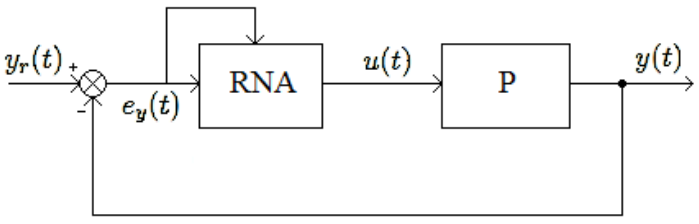

Figura 2.6: Esquema de una RNA como regulador auto ajustable para sistemas SISO (Ponce et al., 2004).

En este sistema de control existen dos errores; el error de salida de la red:

$$e_u(t) = u_d(t) - u(t) \tag{2.54}$$

y el error de salida del sistema:

$$e_y(t) = y_r(t) - y(t) \tag{2.55}$$

En este caso, se usa la ecuación (2.55) y un algoritmo de retropropagación modificado, para actualizar los pesos de la red.

Por otro lado, la arquitectura de la RNA, consta de 3 capas: una capa de entrada, una capa intermedia y una capa de salida. La capa de salida considera una neurona, la capa intermedia tiene tres neuronas y la capa de entrada consiste de tres elementos (ver Figura 2.7).

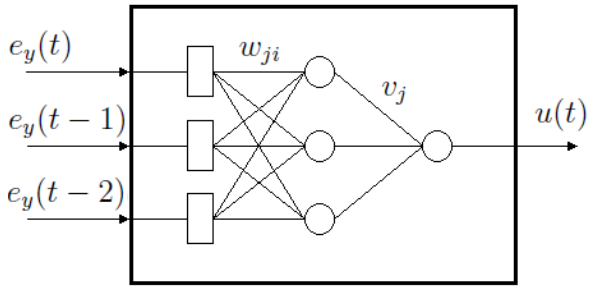

Figura 2.7: Arquitectura de la RNA propuesta (Ponce et al., 2004).

Los elementos del vector de entradas son el error de regulación y dos valores previos del error; es decir, el vector de entradas se obtiene como:

$$x(t) = [e_y(t) \quad e_y(t-1) \quad e_y(t-2)] \tag{2.56}$$

Asimismo, w_{ji} y v_j corresponden a los pesos para la capa intermedia y los pesos de las conexiones de entrada para la capa de salida, respectivamente. Estos pesos son modificados en linea, con el fin de asegurar la reducción de $e_y(t)$. El algoritmo de adaptación de estos pesos, se obtiene a través de un algoritmo de retropropagación modificado; el cual, se detalla a continuación.

La salida de la j-ésima neurona de la capa intermedia es calculada de la siguiente manera:

$$h_j = \frac{1}{1 + e^{-S_j}}, \qquad j = 1, 2, 3 \tag{2.57}$$

Donde:

$$S_j = \sum_{i=1}^{3} w_{ji} x_i, \qquad j = 1, 2, 3 \tag{2.58}$$

De igual forma, la salida en la neurona de la capa de salida, se calcula como sigue:

$$u(t) = \frac{1}{1 + e^{-r}} \tag{2.59}$$

Donde:

$$r = \sum_{j=1}^{3} v_j h_j \tag{2.60}$$

Además, la función para minimizar; es decir, el error cuadrado medio se obtiene como:

$$E(t) = \frac{1}{2} \sum_{k=1}^{t} e_y^2(k) \tag{2.61}$$

Donde el tiempo ha sido discretizado, usando un intervalo de tiempo T_s. Así, el procedimiento de minimización consiste en un movimiento en la dirección negativa del gradiente de $E(t)$ respecto a los pesos w_{ji} y v_j. Siendo el gradiente de $E(t)$, un vector multidimensional; el cual, se calcula con la siguiente ecuación:

$$\nabla E(t) = \begin{bmatrix} \frac{\partial E(t)}{\partial v_j} \\ \frac{\partial E(t)}{\partial w_{ji}} \end{bmatrix} \tag{2.62}$$

Aplicando la regla de la cadena para resolver las derivadas parciales de la ecuación (2.62), se obtiene:

$$\frac{\partial E(t)}{\partial v_j} = \frac{\partial E}{\partial e_y} \frac{\partial e_y}{\partial e_u} \frac{\partial e_u}{\partial u} \frac{\partial u}{\partial r} \frac{\partial r}{\partial v_j} \tag{2.63}$$

$$\frac{\partial E(t)}{\partial w_{ji}} = \frac{\partial E}{\partial e_y}\frac{\partial e_y}{\partial e_u}\frac{\partial e_u}{\partial u}\frac{\partial u}{\partial r}\frac{\partial r}{\partial h_j}\frac{\partial h_j}{\partial S_j}\frac{\partial S_j}{\partial w_{ji}} \tag{2.64}$$

En donde:

$$\begin{aligned}\frac{\partial u}{\partial r} &= \frac{\partial\left(\frac{1}{1+e^{-r}}\right)}{\partial r} = \frac{e^{-r}}{(1+e^{-r})^2} \\ &= \frac{e^{-r}}{1+e^{-r}}\left(\frac{1}{1+e^{-r}}\right) = u(t)(1-u(t))\end{aligned} \tag{2.65}$$

Sustituyendo (2.65) en (2.63) y reduciendo, se tiene que:

$$\frac{\partial E(t)}{\partial v_j} = e_y\frac{\partial e_y}{\partial e_u}(-1)u(t)(1-u(t))h_j \tag{2.66}$$

Además, se define δ_1 como:

$$\delta_1 = e_y u(t)(1-u(t)) \tag{2.67}$$

Sustituyendo (2.67) en (2.66):

$$\frac{\partial E(t)}{\partial v_j} = -\delta_1 h_j\frac{\partial e_y}{\partial e_u} \tag{2.68}$$

Asimismo, de la ecuación (2.64) se tiene que:

$$\begin{aligned}\frac{\partial E(t)}{\partial w_{ji}} &= e_y\frac{\partial e_y}{\partial e_u}(-1)u(t)(1-u(t))v_j h_j(1-h_j)x_i \\ &= -\delta_1 v_j h_j(1-h_j)x_i\frac{\partial e_y}{\partial e_u}\end{aligned} \tag{2.69}$$

Adicionalmente se define δ_{2j} como:

$$\delta_{2j} = \delta_1 v_j h_j(1-h_j) \tag{2.70}$$

Sustituyendo (2.70) en (2.69):

$$\frac{\partial E(t)}{\partial w_{ji}} = -\delta_{2j}x_i\frac{\partial e_y}{\partial e_u} \tag{2.71}$$

De esta manera, usando las ecuaciones (2.68) y (2.71) en la regla delta definida en la sección anterior, para obtener el ajuste en los pesos de la RNA; se definen las siguientes reglas de actualización:

$$v_j(t+1) = v_j(t) + \left(\eta \frac{\partial e_y}{\partial e_u}\right) \delta_1 h_j \tag{2.72}$$

$$w_{ji}(t+1) = w_{ji}(t) + \left(\eta \frac{\partial e_y}{\partial e_u}\right) \delta_{2j} x_i \tag{2.73}$$

Nótese que el signo del gradiente negativo o la dirección en la que decrece el error cuadrado medio es considerado en las ecuaciones (2.72) y (2.73). En donde η es el coeficiente de aprendizaje y $\frac{\partial e_y}{\partial e_u}$ es la ganancia equivalente del sistema.

Por otra parte, el término $\frac{\partial e_y}{\partial e_u}$ es desconocido; por lo que, las ecuaciones (2.72) y (2.73) no pueden ser aplicadas en un problema real. Sin embargo, en Cui y Shin (1993) se demuestra que solo se requiere saber el signo del término $\frac{\partial e_y}{\partial e_u}$ para asegurar la convergencia de los pesos, ya que la magnitud de este término, puede ser incorporada en la magnitud de η; siempre y cuando, la condición $\frac{\partial e_y}{\partial e_u} < \infty$ se cumpla. Adicionalmente, el signo de la ganancia equivalente se puede estimar con un simple experimento: induciendo una entrada escalón de signo conocido, se observa la respuesta de la planta; sí la planta responde con valores del mismo signo que la entrada, el signo de $\frac{\partial e_y}{\partial e_u}$ es positivo y viceversa.

De esta manera, las nuevas reglas de actualización de los pesos son:

$$v_j(t+1) = v_j(t) + \eta \text{sign}\left(\frac{\partial e_y}{\partial e_u}\right) \delta_1 h_j \tag{2.74}$$

$$w_{ji}(t+1) = w_{ji}(t) + \eta \text{sign}\left(\frac{\partial e_y}{\partial e_u}\right) \delta_{2j} x_i \tag{2.75}$$

El coeficiente de aprendizaje η se determina mediante experimentación, observando el comportamiento del sistema en lazo cerrado. Luego, actualizando los coeficientes de una RNA mediante las ecuaciones (2.74) y (2.75); la cual, tiene una arquitectura como se muestra en la Figura 2.7, se puede lograr la reducción del error de un sistema de control para sistemas de una entrada - una salida.

Capítulo 3
DISEÑO DEL CONTROLADOR

3.1. Serie de Fourier como controlador auto-ajustable.

En este trabajo de investigación, se pretende utilizar a la serie de Fourier como un controlador auto ajustable para sistemas lineales y no lineales. La razón principal de ello, es manipular su naturaleza periódica para reducir las perturbaciones del mismo tipo. Al igual que en una red neuronal artificial, en la serie de Fourier como controlador, se reajustan sus coeficientes durante el proceso, con el fin de eliminar el error del sistema en lazo cerrado. En dichos coeficientes, se guarda el aprendizaje de la serie de Fourier y por esta razón, el controlador también puede ser llamado serie de Fourier como controlador de aprendizaje (FSLC por sus siglas en inglés). En la presente sección se muestra el procedimiento matemático para obtener la regla de actualización de coeficientes que aseguran la estabilidad del FSLC así como su definición formal.

Considere la siguiente ecuación, como la representación de la dinámica del error, de un sistema en lazo cerrado:

$$s(t) = e(t) + \dot{e}(t) \tag{3.1}$$

Donde $e(t)$ es el error del sistema y $\dot{e}(t)$ es su derivada en el tiempo. Es decir, $e(t) = \dot{\theta}_d - \dot{\theta}$ es la diferencia entre el valor deseado y el valor medido en el sistema de control. De esta manera, la función $s(t)$ tiene la información de cualquier desviación entre la referencia y el valor real del sistema; incluyendo las perturbaciones y la velocidad de las mismas. Por lo tanto, el objetivo principal del FSLC es conseguir la convergencia de $s(t)$ a cero.

En primer lugar, se analiza el contenido armónico de la función $s(t)$. Para ello, se toman N muestras de $s(t)$, cada periodo de muestreo T_s. Así, se forma una ventana de datos $w_j^k = [w_0^k, w_1^k, ..., w_{N-1}^k]$ en cada instante k. En donde, la muestra s^k del k-ésimo instante, se asigna al término w_{N-1}^k y los otros valores en la ventana, son recorridos a la izquierda; es decir, $w_{N-1}^k = s^k, \quad w_{N-2}^k = s^{k-1}, \quad w_{N-3}^k = s^{k-2}, \quad ... \quad , \quad w_0^k = s^{k-(N-1)}$. De esta manera, cada ventana w_j^k tiene una secuencia discreta de N muestras tomadas de $s(t)$; la cual, puede ser transformada al dominio de la frecuencia usando la DFT, como sigue:

$$X_n^k = \sum_{j=0}^{N-1} w_j^k e^{-i\frac{2\pi nj}{N}}, \qquad n = 0, 1, ..., N-1 \tag{3.2}$$

Nótese que la ventana w_j^k con N elementos reales es suficiente para obtener una representación en frecuencia de la función $s(t)$ en cada instante k, como ya se mencionó en la sección 2.1; siempre y cuando, el periodo de muestreo T_s sea adecuado. Asimismo, la función $s(t)$ no necesita ser periódica.

Por otro lado, usando la formula de Euler en la ecuación (3.2), se obtienen las partes real e imaginaria de X_n^k, como sigue:

$$z_n^k = Re\{X_n^k\} = \sum_{j=0}^{N-1} w_j^k \cos\left(\frac{2\pi nj}{N}\right), \qquad n = 0, 1, ..., N-1 \tag{3.3}$$

$$y_n^k = Im\{X_n^k\} = -\sum_{j=0}^{N-1} w_j^k \sin\left(\frac{2\pi nj}{N}\right), \qquad n = 0, 1, ..., N-1 \tag{3.4}$$

Y los coeficientes para regresar del dominio de la frecuencia al dominio del tiempo en el k-ésimo instante, son:

$$p_0^k = \frac{z_0^k}{N}, \qquad q_0^k = 0 \tag{3.5}$$

$$p_{\frac{N}{2}}^k = \frac{z_{\frac{N}{2}}^k}{N}, \qquad q_{\frac{N}{2}}^k = 0 \tag{3.6}$$

$$p_n^k = \frac{2z_n^k}{N}, \qquad n = 1, ..., \frac{N}{2} - 1 \tag{3.7}$$

$$q_n^k = -\frac{2y_n^k}{N}, \qquad n = 1, ..., \frac{N}{2} - 1 \tag{3.8}$$

En otras palabras, aplicando la IDFT a la ecuación (3.2), considerando los coeficientes de las ecuaciones (3.5), (3.6), (3.7) y (3.8) y siendo N un número par; se obtiene la representación en series de Fourier de las muestras en la k-ésima ventana w_j^k como:

$$w_j^k = \sum_{n=0}^{N/2} \left(p_n^k \cos\left(\frac{2\pi nj}{N}\right) + q_n^k \sin\left(\frac{2\pi nj}{N}\right)\right), \qquad j = 0, 1, ..., N-1 \tag{3.9}$$

Se observa en la ecuación (3.9), que minimizando los coeficientes p_n^k y q_n^k, es posible disminuir los valores que componen a la ventana w_j^k y consecuentemente a la misma w_j^k; la cual, es la representación en frecuencia de la dinámica del error en el k-ésimo instante. Por tanto, los coeficientes del FSLC son ajustados para contribuir a la reducción de p_n^k y q_n^k en cada periodo de muestreo. Adicionalmente, el FSLC utiliza estos coeficientes p_n^k y q_n^k ponderados por una ganancia y sumados a una función especial; la cual, puede ser vista intuitivamente como la parte integral. Finalmente, el FSLC se define como:

$$u(k) = \sum_{n=0}^{N/2} \left(a_n^k \cos\left(\frac{2\pi n(N-1)}{N}\right) + b_n^k \sin\left(\frac{2\pi n(N-1)}{N}\right)\right) \tag{3.10}$$

Donde:

$$a_n^k = \alpha_n p_n^k + \hat{a}_n^k, \qquad n = 0, 1, ..., \frac{N}{2} \tag{3.11}$$

$$b_n^k = \alpha_n q_n^k + \hat{b}_n^k, \qquad n = 0, 1, ..., \frac{N}{2} \tag{3.12}$$

La ganancia α_n se utiliza para obtener una proporción de los armónicos de $s(t)$. Los coeficientes $\hat{a}_n^k$ y $\hat{b}_n^k$ son proporcionados por una función de aprendizaje; la cual, será vista más adelante y es diseñada para asegurar la disminución de p_n^k y q_n^k. Nótese, que la disminución de estos coeficientes afecta a los coeficientes a_n^k y b_n^k del controlador principal. Es por ello, que los coeficientes $\hat{a}_n^k$ y $\hat{b}_n^k$ juegan un papel muy importante en el controlador;

ellos evitan, que cuando los coeficientes p_n^k y q_n^k sean 0 la señal de control se pierda. Luego, los coeficientes $\hat{a}_n^k$ y $\hat{b}_n^k$ desarrollan la parte integral equivalente en un controlador PID.

Para este caso, en el dominio de la frecuencia, la parte integral debe ajustar cada coeficiente del controlador principal hasta que converjan a los valores correctos. En ese sentido, se usa la ley de actualización de coeficientes propuesta por Tang et al. (2000) para la definición de $\hat{a}_n^k$ y $\hat{b}_n^k$, como sigue:

$$\hat{a}_n^k = \gamma_n \sum_{j=0}^{k-1} p_n^j, \qquad n = 0, 1, ..., \frac{N}{2} \tag{3.13}$$

$$\hat{b}_n^k = \gamma_n \sum_{j=0}^{k-1} q_n^j, \qquad n = 0, 1, ..., \frac{N}{2} \tag{3.14}$$

Donde γ_n es la ganancia de aprendizaje para cada armónico y debe ser seleccionada, junto con α_n, de acuerdo a las condiciones de estabilidad, que se muestran en la sección siguiente. Por otra parte, se puede observar en las ecuaciones (3.13) y (3.14), que con la convergencia de p_n^k y q_n^k a cero, se asegura la convergencia de $\hat{a}_n^k$ y $\hat{b}_n^k$ a algún valor.

Nótese, que el FSLC de la ecuación (3.10), es obtenido a través de la IDFT, con los armónicos modificados de $s(t)$. Sin embargo, la IDFT entrega un conjunto de valores discretos; de los cuales, el controlador debería ser alguno de ellos, en cada ventana de datos que es transformada. En otras palabras, el controlador debe tomar un valor único en cada instante k. Es por ello, que debido al acomodo de los datos en la ventana w_j^k, se toma el valor en $j = N - 1$ como el valor actual del controlador; ya que, es en esa posición, en donde se encuentra el valor más reciente de la muestra s^k. Luego, el valor actual del controlador $u(k)$ esta en la posición $j = N - 1$.

Entonces, el algoritmo a seguir para aplicar el FSLC en un sistema de control, es como sigue. Primeramente, se inicia la primer ventana w_j^0 y se dan valores a las ganancias α_n y γ_n, cuidando que cumplan las condiciones de estabilidad que se muestran en la siguiente sección. En seguida, se inicia el sistema y se toma una muestra de $s(t)$. La muestra se agrega a la ventana w_j^k y se recorren los valores de la misma. Posteriormente, se calculan los coeficientes p_n^k y q_n^k con las ecuaciones (3.3) y (3.4). Después, se calculan los coeficientes en las ecuaciones (3.5), (3.6), (3.7) y (3.8) seguido de los coeficientes del controlador a_n^k y b_n^k

con las ecuaciones (3.11) y (3.12). Finalmente, se calcula la ley de control de la ecuación (3.10) y el algoritmo se repite, tomando otra muestra de $s(t)$ y formando una nueva ventana. Es necesario, que en cada ciclo del algoritmo, antes de calcular la DFT de la nueva ventana, se reinicien las sumas de las ecuaciones (3.3) y (3.4). No obstante, las sumas para los coeficientes $\hat{a}_n^k$ y $\hat{b}_n^k$ no deben ser reiniciadas; ya que, ellas contienen la información del aprendizaje obtenido durante el experimento. Luego, el controlador contiene información de la dinámica del error y mantiene (en sus coeficientes o armónicos) la información aprendida durante el experimento; con la cual, puede compensar una perturbación periódica, agregando el armónico equivalente a la perturbación.

3.2. Convergencia del error del sistema en lazo cerrado.

A continuación se desarrolla la prueba de estabilidad para el FSLC. Este análisis, es una adaptación de la metodología seguida por Tang et al. (2000); para este caso, cuando la serie de Fourier es el único controlador del sistema y adicionalmente, cuando no se puede asegurar, que la dinámica del error es una función periódica.

Primeramente, se analiza el cambio en los coeficientes del controlador Δa_n^k y Δb_n^k; esto con la finalidad de encontrar una relación entre las ganancias del controlador y el cambio en los armónicos de $s(t)$. Este análisis es importante debido a que la salida de un sistema no lineal contiene en general, armónicos más grandes adicionales al armónico fundamental (Tang et al., 2000). En vista de, que los armónicos del sistema se presentan en la señal del error con $s(t)$ (aunque sean agregados, nuevos armónicos), el principal objetivo del controlador es reducir los armónicos de la función $s(t)$.

Dicho lo anterior, el cambio en el n-ésimo armónico del controlador es representado como:

$$\Delta a_n^k = a_n^{k+1} - a_n^k \tag{3.15}$$

Donde a_n^{k+1} es el siguiente valor del n-ésimo armónico a partir del k-ésimo instante. Es importante notar, que todo el procedimiento matemático que se muestra en esta sección, es válido también para b_n^k, q_n^k y $\hat{b}_n^k$. Por ello, se hará unicamente la demostración con los coeficientes a_n^k, p_n^k y $\hat{a}_n^k$ y se sugiere al lector seguir el procedimiento para los otros coeficientes.

Adicionalmente, el cambio en los armónicos de $s(t)$ se calcula como sigue:

$$\Delta p_n^k = p_n^{k+1} - p_n^k \tag{3.16}$$

Donde p_n^{k+1} es la magnitud del n-ésimo armónico de $s(t)$, a partir del k-ésimo instante. Así, para analizar la estabilidad y determinar las condiciones para la convergencia del error del sistema en lazo cerrado, se enuncia la siguiente proposición:

Proposición 3.1. *Cuando el controlador* (3.10) *es aplicado a un sistema lineal o no lineal del cual se puede obtener la función* (3.1) *y los coeficientes del controlador son actualizados de acuerdo las ecuaciones* (3.11)*,* (3.12)*,* (3.13) *y* (3.14)*, las condiciones necesarias y suficientes para la convergencia del error son:* $\alpha_n > \left|\frac{\Delta a_n^k}{\Delta p_n^k}\right|$ *y* $\gamma_n < \left|\frac{\Delta a_n^k}{\Delta p_n^k} - \alpha_n\right| \; \forall \; n, k \in \mathbb{Z}^+$.

Prueba. Usando las ecuaciones (3.11) y (3.12), el cambio en los armónicos del controlador se calcula como:

$$\Delta a_n^k = \alpha_n \Delta p_n^k + \Delta \hat{a}_n^k \tag{3.17}$$

Donde, aplicando la definición del cambio de las ecuaciones (3.15) y (3.16) en la ecuación (3.13), el término $\Delta \hat{a}_n^k$ es:

$$\Delta \hat{a}_n^k = \gamma_n p_n^k \tag{3.18}$$

Sustituyendo la ecuación (3.18) en (3.17), se obtienen las siguientes expresiones:

$$\Delta a_n^k = \alpha_n \Delta p_n^k + \gamma_n p_n^k \tag{3.19}$$

Dividiendo ambos miembros de (3.19) por Δp_n^k:

$$\frac{\Delta a_n^k}{\Delta p_n^k} = \alpha_n + \gamma_n \frac{p_n^k}{\Delta p_n^k} \tag{3.20}$$

Sustituyendo (3.16) en el segundo miembro de (3.20):

$$\frac{\Delta a_n^k}{\Delta p_n^k} = \alpha_n + \gamma_n \frac{p_n^k}{p_n^{k+1} - p_n^k} \tag{3.21}$$

$$\frac{\Delta a_n^k}{\Delta p_n^k} = \alpha_n + \gamma_n \frac{1}{\frac{p_n^{k+1}}{p_n^k} - 1} \tag{3.22}$$

$$\frac{p_n^{k+1}}{p_n^k} - 1 = \frac{\gamma_n}{\frac{\Delta a_n^k}{\Delta p_n^k} - \alpha_n} \tag{3.23}$$

$$\frac{p_n^{k+1}}{p_n^k} = \frac{\gamma_n}{\frac{\Delta a_n^k}{\Delta p_n^k} - \alpha_n} + 1 \tag{3.24}$$

Tomando la parte derecha de la ecuación (3.24), la siguiente desigualdad es posible:

$$-1 < \frac{\gamma_n}{\frac{\Delta a_n^k}{\Delta p_n^k} - \alpha_n} + 1 < 1 \tag{3.25}$$

$$-2 < \frac{\gamma_n}{\frac{\Delta a_n^k}{\Delta p_n^k} - \alpha_n} < 0 \tag{3.26}$$

Siempre y cuando, se cumplan dos condiciones. La primera, sí $\gamma_n > 0$, entonces:

$$\alpha_n > \left| \frac{\Delta a_n^k}{\Delta p_n^k} \right| \tag{3.27}$$

Y la segunda condición que asegura la desigualdad (3.26) es:

$$\gamma_n < \left| \frac{\Delta a_n^k}{\Delta p_n^k} - \alpha_n \right| \tag{3.28}$$

De esta manera, con las condiciones (3.27) y (3.28) la desigualdad (3.25) es cierta; lo cual, implica que:

$$-1 < \frac{p_n^{k+1}}{p_n^k} < 1 \tag{3.29}$$

En otras palabras:

$$\left| \frac{p_n^{k+1}}{p_n^k} \right| < \beta < 1 \tag{3.30}$$

Es decir, mientras k incrementa, en cada muestra tomada durante el experimento, los coeficientes p_n^k y q_n^k decrecen con la siguiente secuencia:

$$\left|p_n^k\right| < \beta \left|p_n^{k-1}\right| < \beta^2 \left|p_n^{k-2}\right| < \cdots < \beta^k \left|p_n^0\right| \tag{3.31}$$

Sí, los coeficientes de la representación en series de Fourier de $s(t)$, son disminuidos asintóticamente, en el dominio del tiempo la función $s(t)$ es minimizada; entonces:

$$s(t) = e(t) + \dot{e}(t) = 0 \tag{3.32}$$

Luego, la expresion (3.32) representa un sistema estable ya que la única forma en que (3.32) ocurra, es cuando $e(t)$ tiende a cero conforme el tiempo crece. □

Nótese, que las únicas condiciones para la estabilidad del sistema son (3.27) y (3.28). Sin embargo, sí α_n se elige muy grande, los coeficientes a_n^k y b_n^k también serán grandes, pero esto no implica que los coeficientes p_n^k y q_n^k aumenten; ya que sus magnitudes dependen únicamente del sistema. Finalmente, se podría dejar de cumplir la condición (3.27) ocasionando la inestabilidad del sistema.

Por otra parte, la ganancia γ_n está acotada por arriba debido a la condición (3.28) y es acotada por abajo por 0. No obstante, hacer muy pequeña esta ganancia ocasionaría un aprendizaje lento y el sistema de control quedaría dependiendo de la ganancia α_n; por lo que, encontrar un valor apropiado de α_n sería complicado.

Entonces, es conveniente escoger las ganancias α_n y γ_n con valores no muy extremos; pero, sus magnitudes adecuadas dependen únicamente del sistema a controlar. Adicionalmente, las ganancias α_n y γ_n se pueden elegir diferentes para cada armónico; siempre y cuando, no dejen de cumplir las condiciones de estabilidad. Así que, puede ser útil modificar los componentes de DC α_0 y γ_0 independientes de los otros armónicos; ya que, estos coeficientes podrían regular una inyección de energía constante a diferencia de los otros armónicos.

El número de términos N del controlador, depende de la respuesta deseada; es decir, entre más armónicos se agreguen, se puede desarrollar una señal de control más precisa. Sin embargo, el número de términos está limitado por la frecuencia de resonancia del propio sistema. Nótese, que los sistemas físicos reales tienen un ancho de banda finito y generalmente los componentes de baja frecuencia, dominan la dinámica del error (Tang et al., 2000). Ade-

más, un número grande de términos requiere una ventana grande de datos w_j^k; lo cual exige, más capacidad de memoria en los dispositivos de procesamiento y un tiempo más largo para calcular la salida de control.

Por otro lado, el periodo de muestreo T_s es un factor importante para considerar. Sí, el periodo de muestreo es muy grande, la función $s(t)$ no sería correctamente representada y el controlador no podría asegurar la estabilidad del sistema en lazo cerrado. Como ya se estudió en la sección 2.1, la DFT puede transformar cualquier arreglo de números reales o imaginarios, inclusive si estos números son aleatorios; por lo cual, la representación en series de Fourier, de alguna ventana de datos w_j^k, siempre existirá. Así, el periodo de muestreo sólo afecta a la correcta representación en frecuencia de $s(t)$, siendo la frecuencia de Nyquist, la frecuencia más grande que se pueda representar con la DFT.

Por último, el controlador unicamente considera la medición de la salida de la planta; la cual, se compara con la salida deseada y se obtiene el error. Después, se calcula la función $s(t)$ y posteriormente se obtiene la salida de control $u(k)$. Por tanto, este controlador no requiere el modelo del sistema; luego, el sistema de control es descentralizado (Tang et al., 2000).

Capítulo 4
METODOLOGÍA

4.1. Implementación de los controladores para la regulación de velocidad del PMSM.

Se desea demostrar la hipótesis de la sección 1.4 con el controlador (3.10) (FSLC). Donde, el sistema de prueba es el control de velocidad de un PMSM en bajas velocidades y las perturbaciones periódicas están dadas por el rizado de par en el motor.

En la sección 2.2, se demuestra que el rizado de par es una perturbación periódica y natural en los PMSM's; la cual, es totalmente indeseable en bajas velocidades. Adicionalmente, en el capítulo 3 se demuestra que el FLSC puede minimizar el error de un sistema en lazo cerrado; aunque sea sometido a perturbaciones periódicas. Ahora, se utiliza este controlador para regular la velocidad de un PMSM y se compara su comportamiento con otros dos controladores. Para ello; en esta sección se muestra como serán implementados en un esquema general, para asegurar las mismas condiciones para los tres controladores.

Dado que, el método FOC permite controlar un motor trifásico como si fuera un motor de CD, mediante un controlador maestro de una entrada - una salida; es el control por campo orientado, el esquema general usado para la implementación de los tres controladores. Siendo estos: el control PI tradicional del FOC, una red neuronal artificial y el FSLC. Así, el diagrama de bloques de la Figura 4.1 muestra la forma de implementar los controladores en el FOC para regular la velocidad de un PMSM. Donde, $\dot{\theta}_d[\frac{rad}{s}]$ es la velocidad angular de referencia, I_{dd} es la corriente de referencia para la fase d en el plano (d, q), $u(t) = I_{qd}$ es la señal de control o referencia para la fase q, $e_y(t)$, $e_y(t-1)$, y $e_y(t-2)$ son los errores para la RNA estudiados en la sección 2.6 y las otras variables son correspondientes a las señales

del FOC vistas en la sección 2.3, con excepción de la variable ϑ.

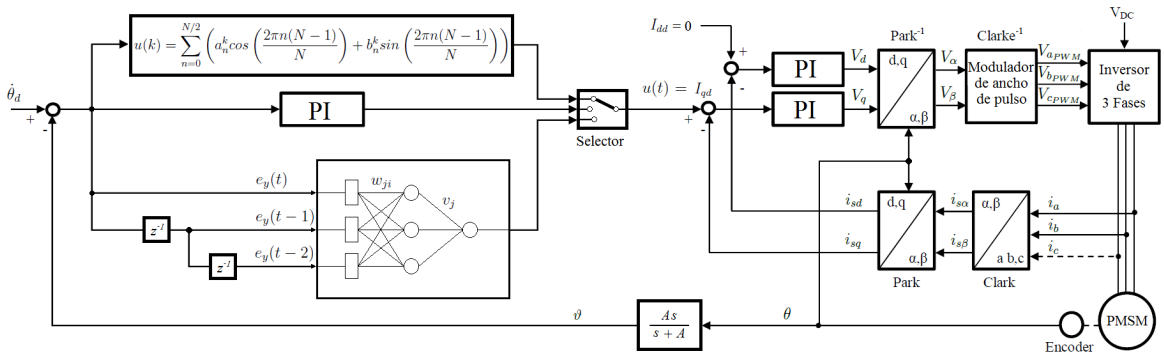

Figura 4.1: Diagrama de bloques de la implementación de los controladores en el FOC.

La variable ϑ es la salida de un filtro de posición propuesto por Kelly et al. (1994); el cual, esta compuesto por un polo en $-A$ y un cero en el origen. Asimismo, A es una constante positiva para asegurar la estabilidad del filtro. De esta manera, ϑ es una estimación de la velocidad del PMSM y se evita el cálculo de una derivada numérica; la cual, ocasionaría bastante ruido en la señal de velocidad y posiblemente un desempeño inadecuado del sistema de control.

Por otro lado, aunque este filtro en la Figura 4.1 se puede ver como:

$$\dot{\vartheta} + A\vartheta = A\dot{\theta} \tag{4.1}$$

La ecuación (4.1) sigue dependiendo de la velocidad $\dot{\theta}$; la cual, es desconocida ya que no es viable un sensor extra de velocidad y el objetivo del filtro es descartar la derivación numérica. Por tanto, la ecuación del filtro se extiende a dos ecuaciones, de la siguiente manera:

$$\dot{x} = -Ax - A^2\theta \tag{4.2}$$

$$\vartheta = x + A\theta \tag{4.3}$$

Es decir, sustituyendo la ecuación (4.2) en la derivada de (4.3), se obtiene la ecuación (4.1). Entonces, las ecuaciones (4.2) y (4.3) son la representación del filtro (4.1), sin embargo,

estas ecuaciones no dependen de $\dot{\theta}$ y pueden ser implementadas numéricamente en código de programación; luego, se puede obtener la estimación de velocidad a partir del filtro sin necesidad de la derivada numérica de la posición angular.

Por otra parte, en la Figura 4.1 se muestran los tres controladores, implementados de forma independiente en el método FOC; los cuales, a partir del error calculan la referencia I_{qd}. No obstante, la señal de control que se suministra a la siguiente etapa del FOC, es elegida mediante un selector según el diagrama de la Figura 4.1. En la aplicación real, este selector representa cada uno de los programas en donde se encuentran codificados los algoritmos de control. De esta manera, se compilan y se corren lineas de programación independientes para hacer la experimentación con cada controlador.

Nótese, que la RNA utilizada en esta investigación es la que se estudia en la sección 2.6; la cual, requiere del error y dos instantes anteriores del mismo. Por lo tanto, se agregan dos bloques en la Figura 4.1, con la variable z^{-1}, representando los retrasos del error respecto al error actual $e(k)$ tomado cada periodo de muestreo T_s. Adicionalmente, los bloques son conectados de tal forma que se obtengan los errores $e(k-1)$ y $e(k-2)$.

Además, es importante destacar, que a la referencia I_{dd} se le asigna el valor de cero; con el fin de obtener la sensibilidad al torque máximo a partir de la corriente en el estator (Adhavan et al., 2011). En otras palabras, como ya se estudio en la sección 2.3 la referencia para el control de la fase i_d es elegida según el tipo de motor con el fin de obtener el par máximo y en este caso, para un PMSM la referencia I_{dd} es cero.

Por último, con el método FOC conectado como se muestra en el esquema de la Figura 4.1, se pueden aplicar independientemente los tres controladores (PI, RNA ó FSLC), para regular la velocidad de un PMSM; en donde, los controladores PI's de las fases (d, q) son los mismos y con la misma sintonización para los tres controladores. Así, únicamente se intercambia el controlador maestro y los tres controladores están bajo las mismas condiciones; luego, el comportamiento de cada uno de ellos, depende del diseño de los mismos.

4.2. Conexión y distribución física del sistema de control.

Para la etapa de experimentación, se diseña un banco de pruebas, con el fin de asegurar que todas las consideraciones y operaciones de la teoría sean aplicadas correctamente

en la práctica. Asimismo, se pretende tener el dominio de todas las señales y tiempos de procesamiento en el sistema.

Adicionalmente, se desea tener la posibilidad de hacer fácilmente cambios en las operaciones matemáticas y guardar los datos obtenidos durante el experimento, sin realizar movimientos en el hardware. En este sentido, el diagrama de conexiones y distribución de componentes para el banco de pruebas, se muestra en la Figura 4.2.

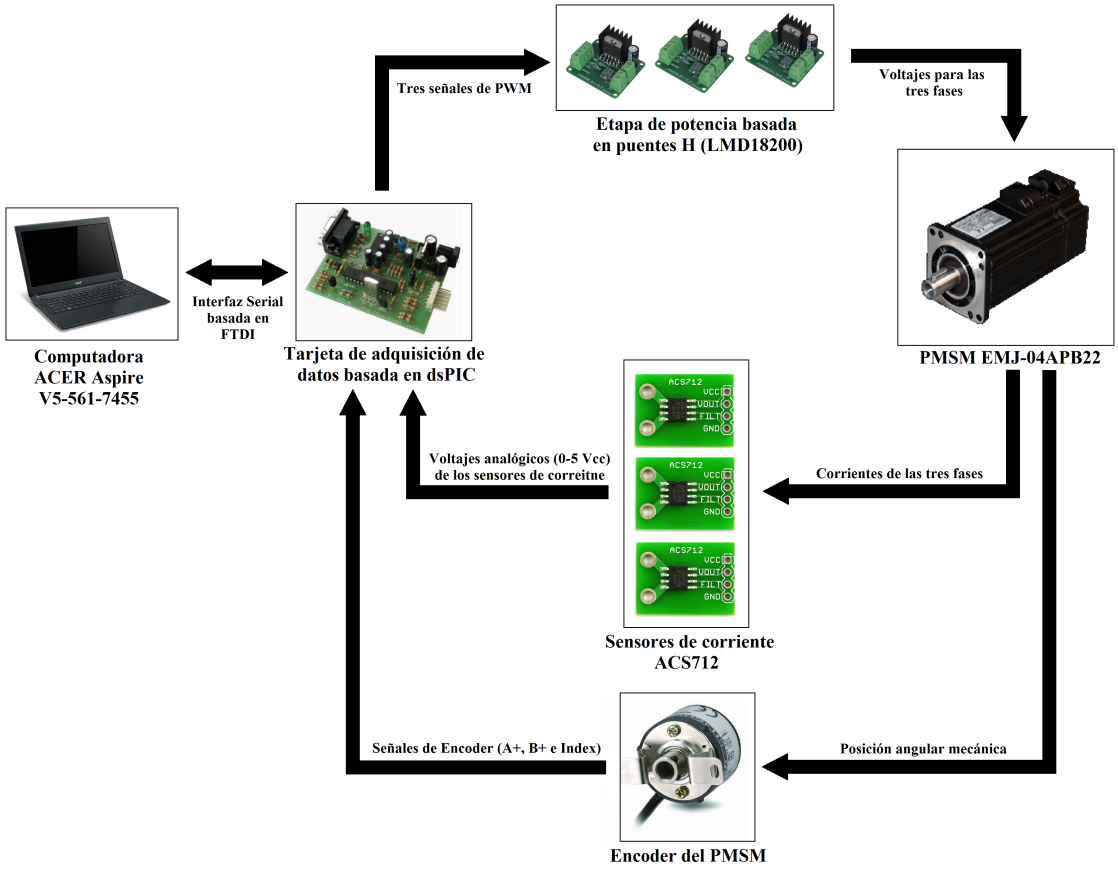

Figura 4.2: Diagrama de conexiones y distribución física de componentes para el banco de pruebas.

El componente principal del banco de pruebas es la computadora personal; en ella son procesados los algoritmos de control y son guardados los datos obtenidos durante el experimento. Aunque, el principal objetivo de este componente, es calcular la salida de control para enviarla a la siguiente etapa; la computadora realiza varias tareas importantes. Inicialmente recibe de una tarjeta de adquisición de datos, los valores binarios representativos de

las mediciones de los sensores de corriente y el codificador incremental óptico (encoder) del motor. Posteriormente, se calculan los valores reales de las mediciones en el sistema internacional y se realizan las transformaciones iniciales del método FOC. Después, se calcula la estimación de velocidad con las ecuaciones del filtro de posición y se obtiene el error del sistema. En seguida, se calcula la salida del controlador maestro, precedida de las salidas de los controladores esclavos. Con estos valores, se aplican las transformaciones inversas del método FOC y así se obtienen los voltajes necesarios para cada una de las tres fases. Por último, se escalan estos voltajes y se convierten en valores binarios; los cuales, son enviados de regreso a la tarjeta de adquisición. Adicionalmente, todos los datos que se necesiten guardar durante el experimento, son escritos en un archivo de texto; de esta manera, al final del experimento los datos pueden ser graficados y analizados.

La tarjeta de adquisición de datos está basada en un dsPIC y cumple con diversos objetivos; entre los cuales, uno de los más importantes es asegurar el tiempo T_s en el que se tomarán muestras de las mediciones hechas por los sensores. Una vez tomadas las muestras, son enviadas a la computadora mediante una interfaz serial. Seguido de esto, la tarjeta de adquisición espera en un bucle interno, hasta que la computadora reenvíe los valores en formato binario de los voltajes para las tres fases. Estos valores binarios son aplicados en funciones especiales del dsPIC, para la generación de tres señales de modulación por ancho de pulso (PWM's). Estas señales de PWM son usadas posteriormente por la etapa de potencia para producir los voltajes de las tres fases del motor.

La etapa de potencia está compuesta principalmente por tres puentes H, los cuales se encargan de convertir las señales de PWM provenientes de la tarjeta de adquisición de datos, en voltajes sinusoidales para las tres fases del motor. Cabe destacar que, los voltajes de las tres fases; siendo valores positivos y negativos, son posibles gracias a que la tarjeta de adquisición de datos, junto con las señales de PWM, envía ciertos bits que son interpretados por los puentes H como el cambio de signo que deben hacer para generar voltajes en ambos sentidos o polaridades.

Los voltajes generados en la etapa de potencia son enviados directamente al PMSM. En este, se generan las variables que son medidas por los sensores (sensores de corriente y encoder). Posteriormente, el lazo de control se cierra con la tarjeta de adquisición de datos

y el proceso de control vuelve a comenzar con el envío de datos a la computadora. Así, las consideraciones y operaciones matemáticas de la teoría, se pueden asegurar en la práctica, con un banco de pruebas diseñado en base al diagrama que se muestra en la Figura 4.2.

Por último, el usuario tiene la libertad de cambiar los algoritmos de control y analizar fácilmente los datos obtenidos durante los experimentos; únicamente, cambiando lineas de código desde una computadora personal, sin hacer cambios en el hardware.

4.3. Materiales y métodos.

A continuación se describen los componentes que se utilizan para la construcción del banco de pruebas; así como, los instrumentos y métodos utilizados para realizar las mediciones.

4.3.1 Metrología de atributos dinámicos.

- Posición angular mecánica del PMSM, $\theta[rad]$:

  Para medir la posición angular del motor, se utiliza el encoder del mismo; el cual, tiene una resolución de 2500 pulsos por revolución. Los pulsos del encoder son contados por el dsPIC en la tarjeta de adquisición; el cual, se detalla más adelante. Estos pulsos del encoder se multiplican por 4 debido a que, el microcontrolador cuenta los filos de subida y de bajada de las señales de cuadratura; después, las cuentas son directamente convertidas a radianes con la relación $2\pi[rad] = 10000[cuentas]$. De esta manera, se obtiene la posición angular del motor en radianes y cada que ocurre una revolución, mediante la señal de index (C+), se reicnician las cuentas en 0. La computadora recibe las cuentas que ha realizado el dsPIC cada $0.005[s]$. En la Figura 4.3 se muestra el conector del encoder con los pines enumerados; los cuales, corresponden al color de cables y señales de cuadratura que se muestra en la Tabla 4.1.

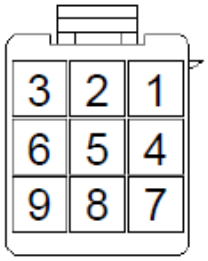

Figura 4.3: Conector del encoder (Manual, 2016).

Tabla 4.1: Pines correspondientes a las señales del encoder.

Pin No.	Señal	Color
1	A+	Azul
2	B+	Verde
3	C+	Amarillo
4	A-	Azul/Negro
5	B-	Verde/Negro
6	C-	Amarillo/Negro
7	PG 5V	Rojo
8	PG 0V	Negro
9	FG	Shield

- Corrientes de las fases del PMSM, $\boldsymbol{i}_{abc}[A]$:

  Las tres corrientes en las fases del motor son medidas a través de sensores ACS712 de la marca Allegro (ver Figura 4.4). El principio de funcionamiento del sensor, es el efecto Hall y debe ser alimentado con $5[V]$. Este sensor, entrega una señal analógica entre $1.6[V]$ a $3.4[V]$; de los cuales, el rango de $1.6[V]$ a $2.5[V]$ corresponden a la corriente medida entre $-5.0[A]$ a $0.0[A]$ y el rango de $2.5[V]$ a $3.4[V]$ corresponden a la corriente medida entre $0.0[A]$ a $5.0[A]$. Los voltajes analógicos de los tres sensores son convertidos a números binarios, mediante el convertidor analógico a digital (ADC) de 12 bits del dsPIC y posteriormente son todos enviados cada $0.005[s]$ a la computadora.

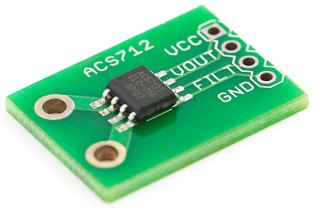

Figura 4.4: Sensor de corriente ACS712 (Openhacks, 2016).

- Estimación de velocidad del PMSM, $\vartheta[\frac{rad}{s}]$:

  La estimación de velocidad del motor se obtiene gracias al filtro de posición que se menciona en la sección 4.1, a través de las ecuaciones (4.2) y (4.3). Sin embargo, para programar el filtro, estas ecuaciones deben ser discretizadas. Para ello, se considera la derivada respecto al tiempo de $\dot{x}$ en su forma discreta como:

$$\dot{x} = \frac{x(k) - x(k-1)}{T_s} \tag{4.4}$$

  Donde $T_s = 0.005[s]$ es el periodo de muestreo, $x(k)$ es el valor actual de $x(t)$ y $x(k-1)$ es el valor pasado de $x(t)$. Así, la ecuación (4.2) en forma discreta es:

$$\frac{x(k) - x(k-1)}{T_s} = -Ax(k) - A^2\theta(k) \tag{4.5}$$

  Despejando $x(k)$ de la ecuación (4.5), se tiene que:

$$x(k) = \frac{x(k-1) - A^2\theta(k)T_s}{1 + AT_s} \tag{4.6}$$

  Asimismo, la forma discreta de la ecuación (4.3) es:

$$\vartheta(k) = x(k) + A\theta(k) \tag{4.7}$$

  De esta manera; primeramente se inicia el valor de $x(k-1) = 0$ y se procede de la siguiente manera: se obtiene una muestra de la posición angular $\theta(k)$, después se

obtiene $x(k)$ con la ecuación (4.6) y por último se obtiene la estimación de velocidad, con la ecuación (4.7). Adicionalmente, es importante recordar que antes de volver a calcular $x(k)$, se debe actualizar el valor de $x(k-1)$.

4.3.2 Componentes de la etapa de potencia.

- Puente H:

  Los puentes H que se utilizan para generar los voltajes en la etapa de potencia son fabricados por la empresa Texas Instruments, modelo LMD18200 (ver Figura 4.5).

  Este puente H tiene la capacidad de entregar $3[A]$ y opera con voltajes de hasta $55[V]$; además, el puente H cuenta con un pin específico para saber si la temperatura del dispositivo es peligrosa. Adicionalmente, cuenta con tres terminales más, para las señales de control; incluyendo, un pin de freno, uno pin para el cambio de dirección y un pin para la señal de PWM. Cabe destacar que, con el pin de freno en nivel bajo de voltaje (GND) y con la señal de PWM también en nivel bajo, el puente H junta las terminales de salida, ocasionando un corto en estas.

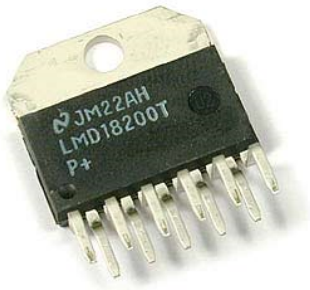

Figura 4.5: Puente H LMD18200 (Norte, 2016).

- Motor síncrono de imán permanente:

  El motor que se utiliza en esta investigación es el PMSM EMJ-04APB22 de la empresa Anaheim Automation (ver Figura 4.6). Es un motor de 4 polos, $200[V]$, $2.7[A]$, $400[W]$ de potencia máxima, $3000[rpm]$ de velocidad nominal, $4.7[\Omega]$ de resistencia en las bobinas del estator y $0.014[H]$ de inductancia en las bobinas del estator.

Figura 4.6: PMSM EMJ-04APB22 (Anaheim, 2016).

4.3.3 Componentes de la etapa de comunicación y adquisición de datos.

- dsPIC:

  El componente principal de la tarjeta de adquisición de datos es el dsPIC. En este trabajo de investigación se usa el microcontrolador dsPIC33FJ12MC202 de la empresa Microchip. Este dsPIC fue seleccionado debido a que cuenta con los recursos suficientes para esta aplicación; los cuales son: un ADC de hasta 12 bits con 6 canales diferentes y posibilidad de establecer 2 de ellos como voltajes de referencia, para mayor resolución, un modulo decodificador de cuadratura (QEI, Quadrature Encoder Interface) para la lectura de un encoder; el cual, es independiente al procesador interno del dispositivo, de esta manera, puede contar los pulsos de las señales de cuadratura, sin ocupar tiempo del programa principal del dsPIC. Cuenta con un puerto de comunicación UART; el cual, es utilizado a su máxima velocidad de $460800[baudios]$. Además, tiene la posibilidad de producir hasta 8 señales de PWM y cuenta con tres temporizadores (timers), entre otras ventajas.

- Interfaz serial entre el banco de pruebas y la computadora personal:

  La comunicación serial entre del banco de pruebas y la computadora, se lleva a cabo mediante el dispositivo FT232RL de la empresa FTDI Chip (ver Figura 4.7). Este dispositivo convierte la interfaz serial UART al protocolo USB para su comunicación con la computadora y puede trabajar con voltajes desde $3.3[V]$ hasta $5[V]$. En esta aplicación, el FT232RL es conectado directamente al dsPIC mediante el puerto de comunicación UART y con la computadora mediante un puerto USB; de esta forma, en la

computadora se crea un puerto virtual de comunicación serial (RS-232); el cual, puede ser abierto mediante alguna aplicación en la PC.

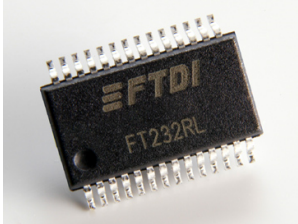

Figura 4.7: FTDI FT232RL (DIY, 2016).

4.3.4 Programación y procesamiento del sistema de control.

Los algoritmos de control y todos los cálculos necesarios para el sistema de control son programados y procesados en una computadora portátil Aspire V5-561-7455 de la empresa ACER. A continuación se mencionan sus características principales y los programas utilizados para este trabajo:

- Software:
  - Sistema operativo Windows 10 Home Single Language, 64 bits.
  - MATLAB R2012b versión 8.0.0.783, 64-Bit (para las gráficas).
  - Dev-C++ versión 5.8.2.
  - PCW C compiler versión 4.114.
- Hardware:
  - Procesador Intel Core i7-4500U, 1.8[GHz].
  - Memoria RAM de 8[GB].
  - Disco duro de 1[TB].
  - Dos puertos USB2 y uno puerto USB3.
  - PICkit 3 versión 3.10.

Capítulo 5

CONSTRUCCIÓN DEL BANCO DE PRUEBAS

La construcción de un banco de pruebas útil para la aplicación de diferentes controladores en motores trifásicos de baja potencia, se muestra en este capítulo. Se pretende, tener pleno control y conocimiento de todas las variables que interaccionan; así como, la seguridad de que el periodo de muestreo y los voltajes calculados por los controladores, sean correctamente aplicados. Para ello, el banco de pruebas esta diseñado en tres subsistemas: el de alimentación, el de potencia y el de instrumentación y comunicación con la PC. Cada uno de estos subsistemas es diseñado con el fin de asegurar que la práctica coincida en buena medida con la teoría y con la idea de que todas las modificaciones que se tengan que hacer, sean hechas en software mediante programación, en lugar de hacer cambios en hardware.

Adicionalmente, se desea que el banco de pruebas sea portable, didáctico y fácil de usar por cualquier usuario con los conocimientos suficientes de electrónica y programación. Para lo cual, se muestran los circuitos electrónicos de cada subsistema y sus respectivos componentes. Además, en el apendice A se presentan los programas utilizados para el microcontrolador, el programa en Dev-C++ para la codificación de los controladores y el programa de MATLAB para las gráficas de los experimentos.

5.1. Subsistema de alimentación para el banco de pruebas.

La alimentación del banco de pruebas consta de un circuito electrónico de 5 fuentes de voltaje; las cuales, son realizadas a partir de una conexión común a la corriente alterna de $127[V]$ a $60[Hz]$ en el caso de México. Se utilizan dos transformadores, uno de $12[V]$ a $3[A]$ para las fuentes de baja potencia y uno de $34[V]$ a $6[A]$ para la fuente que alimenta las bobinas del motor.

Los voltajes de salida de los transformadores, pasan a puentes de diodos para transformar la corriente alterna en corriente directa y posteriormente se fijan los voltajes mediante capacitores. En el caso de la fuente para las bobinas del motor, se fija un voltaje de $46.3[V]$ tomando el devanado primario del transformador. Sin embargo, para las fuentes de voltaje de baja potencia, se utiliza el devanado secundario del transformador de $12[V]$ y mediante reguladores de voltaje a $5[V]$ y $3.3[V]$ se obtienen tres fuentes de $5[V]$ y una de $3.3[V]$.

Estas fuentes, son aplicadas de la siguiente manera: una fuente de $5[V]$ comparte tierra común con la fuente de potencia para el motor; esta fuente, es utilizada para la parte digital de la etapa de potencia, las dos fuentes restantes de $5[V]$ son usadas para la etapa de instrumentación y comunicación, una de ellas alimenta los sensores de corriente y la otra alimenta el encoder del motor junto con una etapa de acoplamiento de impedancias. Por último, el voltaje de $3.3[V]$ se emplea para alimentar al dsPIC. El circuito electrónico del subsistema de alimentación se muestra en la Figura 5.1.

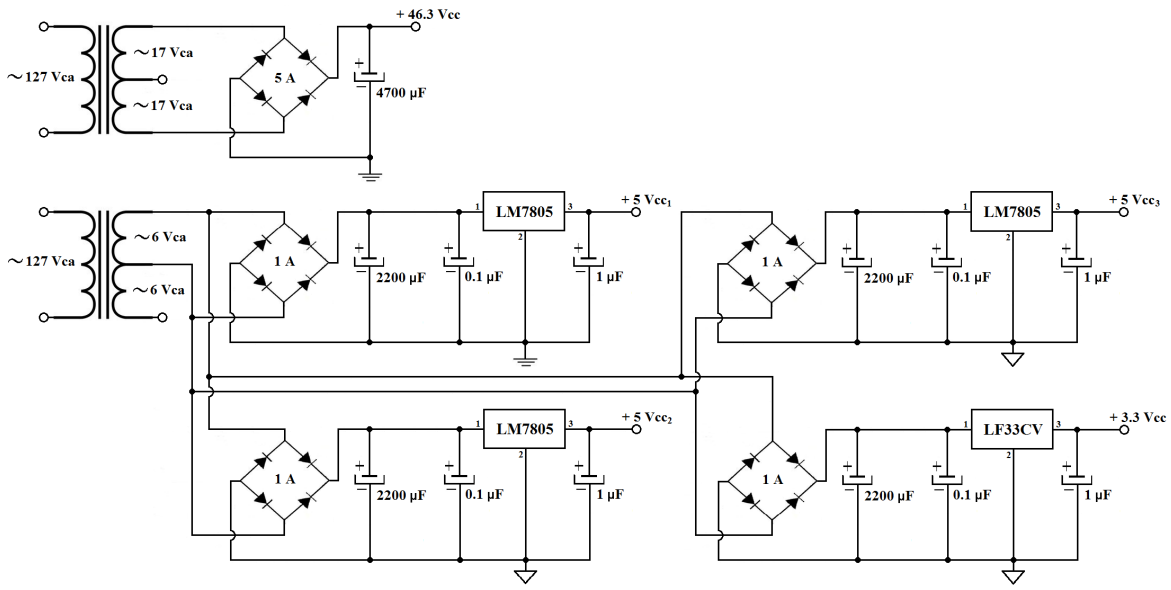

Figura 5.1: Diagrama de conexiones de las fuentes de voltaje.

5.2. Subsistema de la etapa de potencia.

La etapa de potencia es construida mediante el circuito electrónico de la Figura 5.2. Esta etapa es separada de la etapa de instrumentación, mediante opto-acopladores 4N28. Una

vez separadas las referencias de las tres señales de PWM y los tres bits de dirección, estos son aplicados a los puentes H; los cuales, son conectados con el circuito básico según su hoja de datos. Las señales de PWM y los bits de dirección son establecidos por la etapa de instrumentación; de esta manera, los puentes H pueden generar los voltajes sinusoidales de amplitud controlada y desfasados $120°$ para las bobinas del motor. Adicionalmente, se abre una terminal de cada bobina del motor, para medir su corriente en la siguiente etapa.

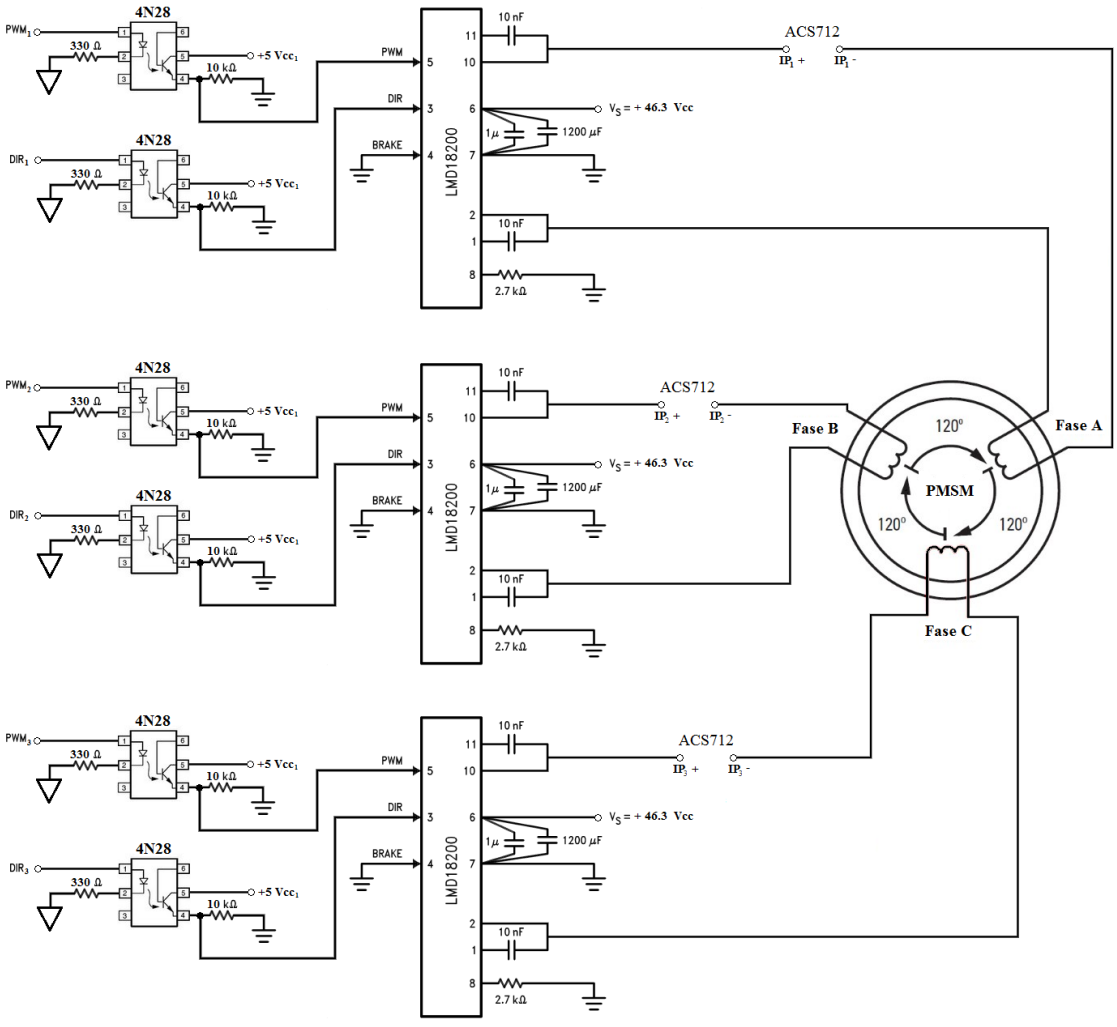

Figura 5.2: Diagrama de conexiones de la etapa de potencia.

5.3. Subsistema de instrumentación y comunicación con la PC.

Este subsistema es el nodo común entre el banco de pruebas y los controladores programados en la computadora personal. Su construcción se realiza mediante el circuito electrónico de la Figura 5.3. Esta basado en un dsPIC; el cual, se encarga de contar el tiempo en el que serán adquiridos los datos de los sensores de corriente y las cuentas del del encoder. Las señales de salida de los sensores de corriente, son voltajes analógicos que son codificados con el ADC de 12 bits del dsPIC; estos voltajes son comparados con los voltajes de referencia alto y bajo en los pines 2 y 3 del dsPIC, para obtener la resolución máxima; los cuales, se fijan mediante resistencias variables en $3.1[V]$ y $1.9[V]$. Es importante mencionar que los voltajes de salida de los sensores, primeramente pasan a través de filtros pasa bajos de primer orden; los cuales, son establecidos a una frecuencia de corte de $28.420[Hz]$, con la finalidad de evitar el fenómeno de *aliasing* respecto a la frecuencia de muestreo de $200[Hz]$.

Por otra parte, en esta etapa se realiza también la comunicación con la PC mediante el dispositivo FT232RL, el cual convierte la comunicación UART entre el dsPIC y el FTDI en una interfaz USB con la computadora; de esta manera, la computadora se puede conectar directamente con el banco de pruebas mediante USB. Posteriormente, en la computadora se genera un puerto serial virtual; el cual, se configura y se abre mediante el programa en Dev-C++.

Adicionalmente, el dsPIC es capaz de codificar los pulsos generados por el encoder del motor, mediante un componente interno del mismo; el cual, se dedica únicamente a realizar esta tarea. Posteriormente, mediante una instrucción programada, se pide al componente del dsPIC, el número de cuentas que lleva el encoder y así, se puede enviar este dato a la computadora. El encoder del motor es conectado, en primer lugar, a un acoplamiento de impedancias, constituido por seguidores de voltaje, mediante amplificadores operacionales de una fuente LM385N; esto, con la finalidad de evitar la caída de voltaje en las señales generadas por el encoder y eventualmente la pérdida de cuentas.

Además, se considera en este subsistema, la conexión del programador (PICkit 3) para el dsPIC, en caso de que se requiera hacer algún cambio en el programa del mismo. Así, el dsPIC tiene la posibilidad de ser re-programado desde su circuito sin necesidad de

desmontarse.

Por otra parte, el dsPIC, como ya se mencionó anteriormente, genera las señales de PWM y los bits de dirección para los puentes H; sin embargo, una de las señales del programador coincide con una señal de dirección para el puente H. Debido a esto, se utiliza un switch; el cual, cambia el uso de este pin en el dsPIC para el fin específico que se requiera.

De esta forma, el dsPIC se encarga de enviar, cada periodo de muestreo, las muestras obtenidas de los sensores; posteriormente recibe los valores codificados de voltajes para las bobinas del motor y por último, produce las señales necesarias para que, en los puentes H de la etapa de potencia, se representen correctamente los voltajes calculados por los controladores. Nótese, que únicamente sobra un pin del dsPIC, mientras que todos los demás, son utilizados; es decir, el dsPIC seleccionado tiene los recursos necesarios y suficientes para esta aplicación.

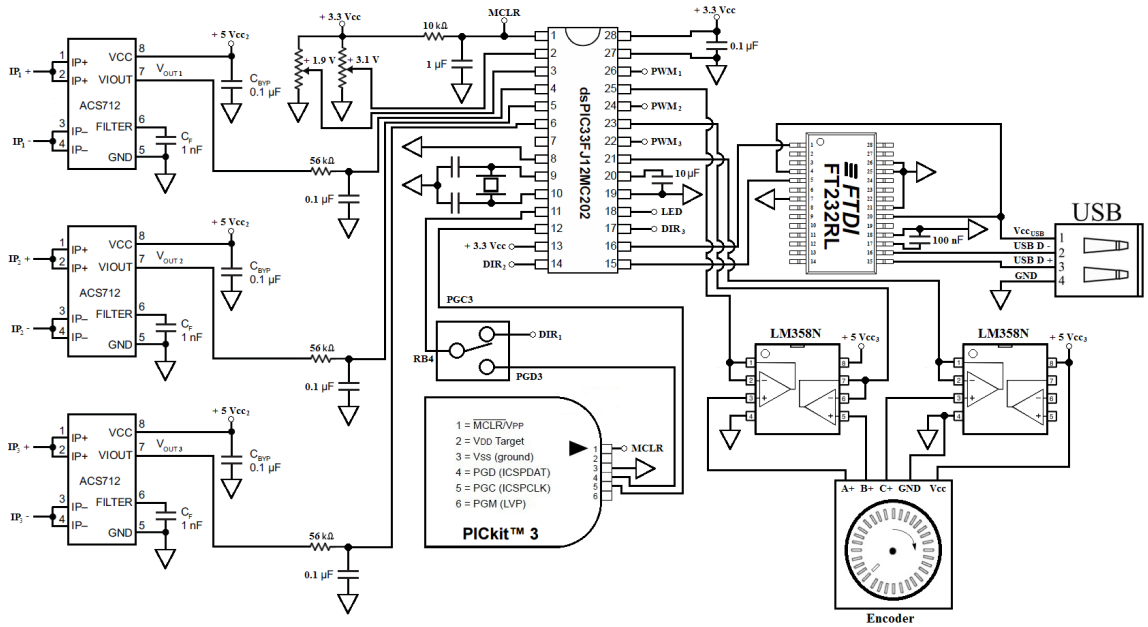

Figura 5.3: Diagrama de conexiones de la etapa comunicación e instrumentación.

5.4. Banco de pruebas completo.

La construcción del banco de pruebas resulta en el dispositivo que se muestra en la Figura 5.4. Los circuitos electrónicos fueron separados en los tres subsistemas vistos en este capítulo y son acomodados dentro de un gabinete de instrumentación para su fácil man-

tenimiento como se muestra en el lado derecho de la Figura 5.4. Por la parte de afuera del gabinete, se conectan las tres bobinas del motor junto con su encoder, se enchufa la clavija que alimentará todo el banco de pruebas y posteriormente se conecta el cable USB a la computadora personal. De esta manera, se puede comenzar la experimentación.

Como se puede observar, el sistema es flexible para diferentes motores trifásicos con encoder y se pueden realizar experimentos con distintos controladores, únicamente cambiando lineas de programación. Adicionalmente, se tiene la posibilidad de observar gráficamente el comportamiento del sistema de control, ya que todas las variables son manipuladas desde la computadora personal.

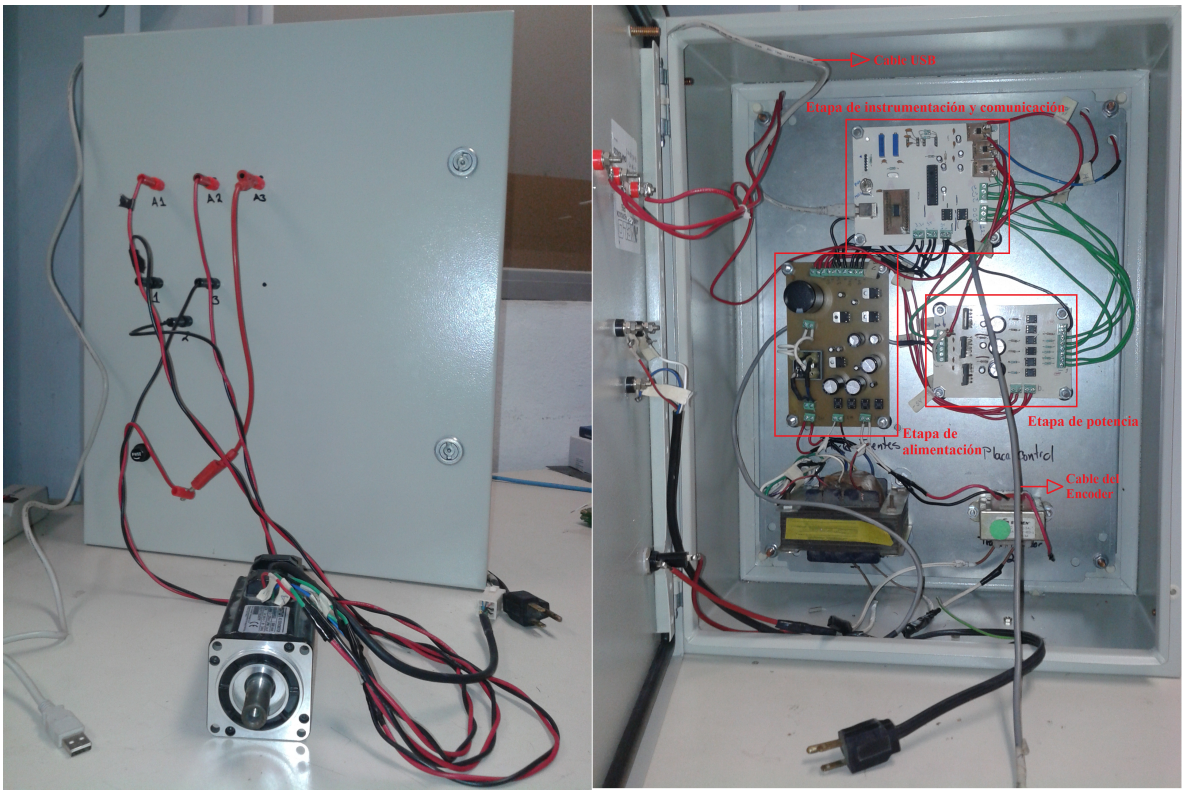

Figura 5.4: Banco de pruebas.

Capítulo 6

RESULTADOS Y DISCUSIÓN

En este capítulo se muestran los experimentos realizados, con el banco de pruebas visto en el capítulo 5 y basándose en la implementación estudiada en la sección 4.1. Se utiliza el método FOC, con los tres diferentes controladores PI, RNA y el FSLC como controladores de velocidad maestros para el PMSM mencionado en la sección de materiales y métodos.

Las ganancias del controlador PI maestro son tomadas del trabajo de Hernández-Guzmán et al. (2014); quienes utilizan un motor similar al de esta investigación. Estas ganancias son: $K_{pm} = 0.0477$ y $K_{im} = 2.38$. Asimismo, la velocidad de referencia para todos los controladores será, la velocidad de referencia más pequeña reportada en el mismo trabajo; esta es: $\dot{\theta}_d = 0.3142[\frac{rad}{s}]$.

Para los controladores esclavos de las corrientes I_q e I_d, se utilizan las ganancias $K_{ps} = 1.0$ y $K_{is} = 10.0$. En el caso de la RNA, se utilizan tres neuronas en la capa intermedia y un coeficiente de aprendizaje de $\eta = 4.9$. Por último, el FSLC se prueba con $N = 4$ términos y ganancias de $\alpha_n = 0.037$ y $\gamma_n = 0.03$. Adicionalmente, se usa un periodo de muestreo de $T_s = 0.005[s]$ y una ganancia para el filtro de posición de $A = 5.0$.

La experimentación se realiza de la siguiente manera: en primer lugar se selecciona el controlador que se pondrá a prueba, posteriormente se posiciona la flecha del motor en posición $0[rad]$, una vez posicionada la flecha, se corre el programa y después se enciende el banco de pruebas. En $t = 50[s]$ se deja caer una masa de $1.2[Kg]$ ocasionando un par de perturbación constante de aproximadamente $0.0824[Nm]$ en base al radio de la flecha del motor. En la Figura 6.1 se muestra la forma de aplicar el par de perturbación en la flecha del motor, para asegurar que la carga sea constante. Como se puede observar, se usa una abrazadera para sujetar un hilo cáñamo; sobre el cual, se cuelga la masa.

Al girar la flecha del motor, el hilo se enrolla y se recorre sobre la flecha; de esta manera, el radio de aplicación del peso no cambia y la perturbación se mantiene constante durante el resto del experimento.

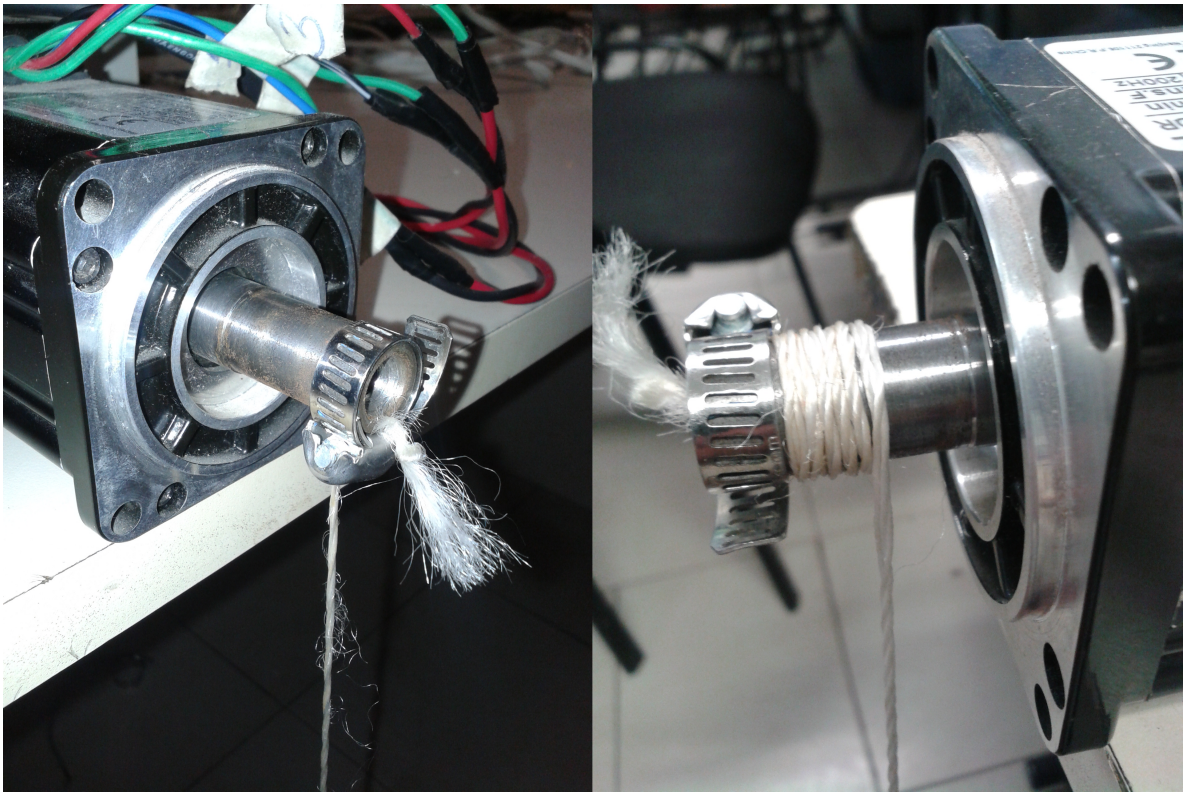

Figura 6.1: Flecha del motor con la perturbacion.

Al cabo de $t = 100[s]$ se detiene el experimento y se obtienen las gráficas resultantes. De la misma forma se prueban cada uno de los tres controladores; cabe mencionar, que desde el programa principal se puede forzar voltaje cero en las bobinas del motor; con lo cual, se puede correr el programa y encender el banco de pruebas, esto con el fin de mover manualmente la flecha del motor hasta posición cero.

Los resultados obtenidos se muestran a continuación. En la Figura 6.2 se presenta la gráfica de posición angular, obtenida cuando el controlador maestro de velocidad, es el PI. Se observa el efecto del rizado de par sobre la posición angular del motor y es evidente que el controlador PI de velocidad no puede compensar estas perturbaciones. Nótese, que la posición angular es un valor entre 0 y 2π en cada revolución; por lo que, para una velocidad constante se deben generar lineas rectas de 0 a 2π y para este caso no se pueden observar.

Cuando la masa se deja caer en $t = 50[s]$ aproximadamente, se observa que la posición angular se ve afectada ligeramente y posteriormente el controlador PI corrige la parturbación constante; sin embargo, el rizado de par continua hasta el final del experimento.

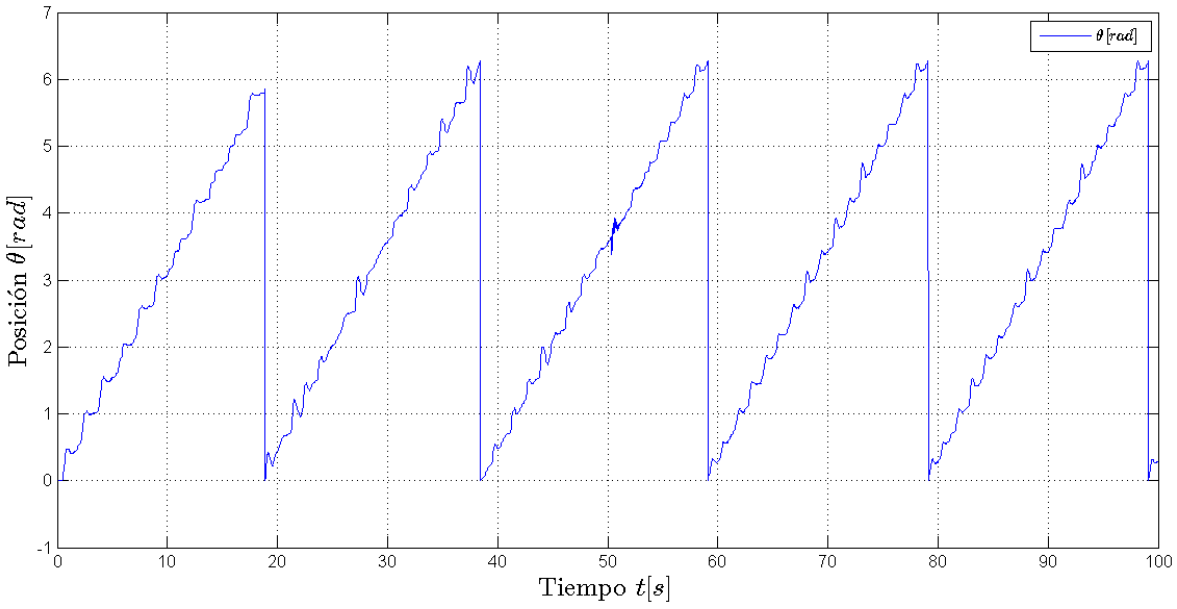

Figura 6.2: Posición angular cuando la velocidad es controlada con el PI, para una referencia de $0.3142[\frac{rad}{s}]$.

En la Figura 6.3 se muestra la gráfica de la posición angular del motor, obtenida cuando la RNA es el controlador maestro de velocidad. Se observa que durante los primeros $t = 18[s]$ mientras ocurre la primera revolución, la RNA esta aprendiendo del sistema y se presentan bastantes cambios de velocidad. Posteriormente, en la segunda revolución, la RNA ya tiene la información necesaria para mantener una velocidad más constante que en el caso del controlador PI. En la gráfica se puede apreciar que la RNA si puede compensar el rizado de par y son bastante visibles las lineas rectas de posición que se generan. Cuando se induce la pertubación de par, la posición varía ligeramente y enseguida, la red neuronal compensa esta perturbación y continua controlando la velocidad; sin embargo, se puede observar un ligero aumento en las variaciones de la posición a partir de $t = 50[s]$ aproximadamente, hasta el final del experimento.

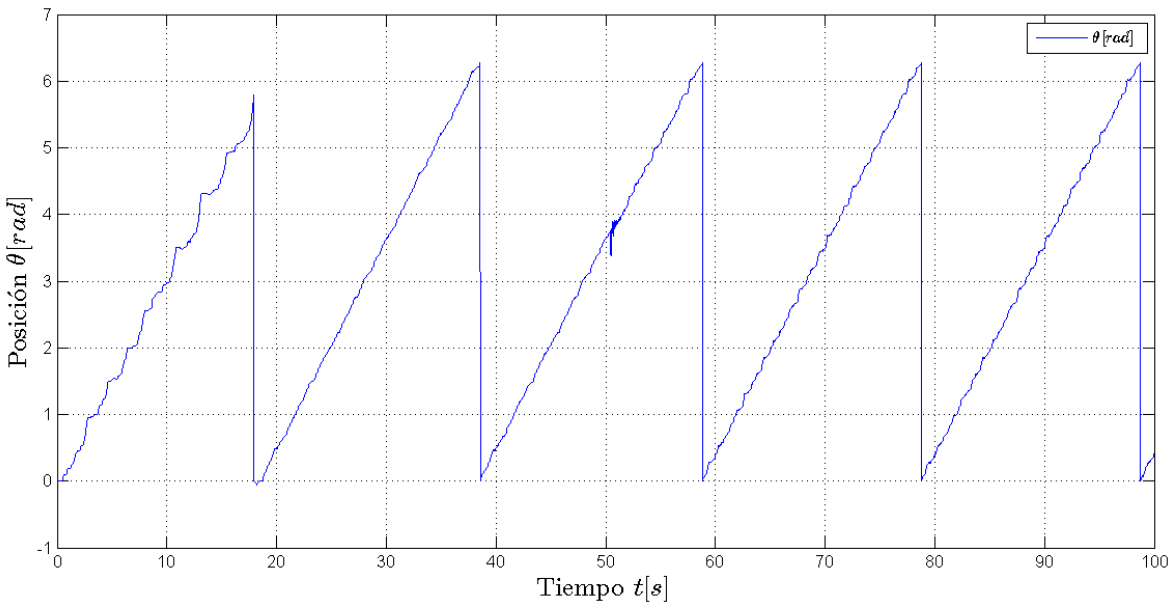

Figura 6.3: Posición angular cuando la velocidad es controlada con la RNA, para una referencia de $0.3142[\frac{rad}{s}]$.

En la Figura 6.4 se presenta la gráfica de la posición angular del motor, obtenida cuando el FSLC es el controlador maestro de velocidad. Al inicio del experimento, se observa que la posición angular disminuye en lugar de aumentar; sin embargo, poco tiempo después el controlador corrige y en seguida, mantiene una velocidad constante. A diferencia de los controladores anteriores, la serie de Fourier como controlador, puede compensar en solo $t = 2[s]$ las perturbaciones generadas por el rizado de par. A este controlador le toma menos tiempo ajustar sus coeficientes que a la RNA y también se observan las lineas rectas, representantes de una velocidad constante. Cuando la perturbación de par se induce en $t = 50[s]$, la posición tiene variaciónes más pequeñas que en el caso de los otros dos controladores; posteriormente, el FSLC compensa la perturbación y sigue controlando la velocidad, sin rizado de par, hasta el final del experimento.

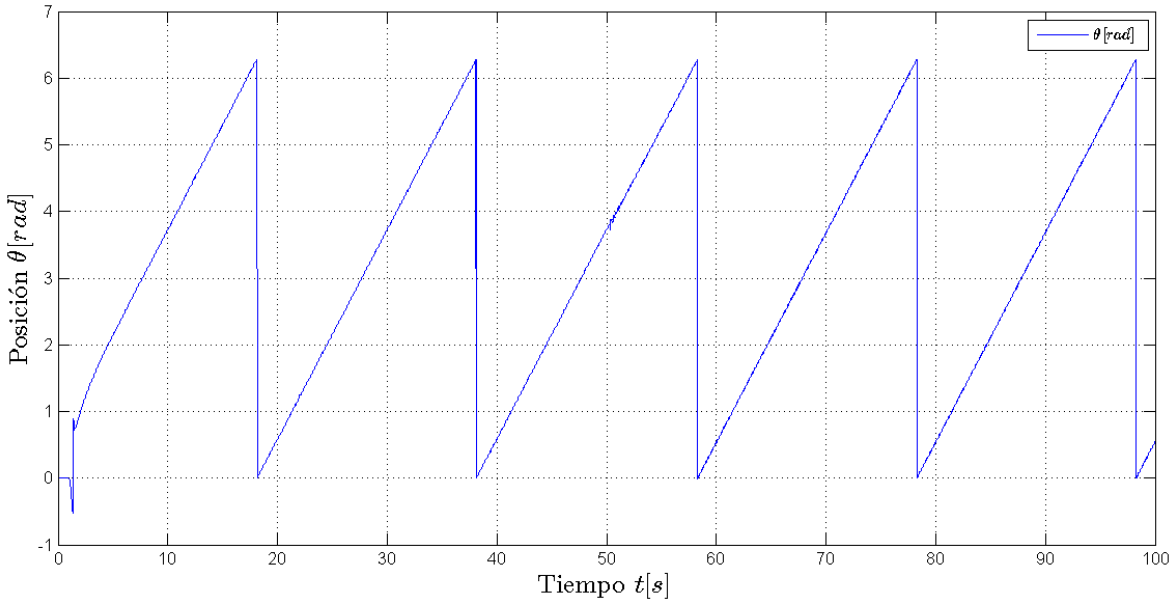

Figura 6.4: Posición angular cuando la velocidad es controlada con el FSLC, para una referencia de $0.3142[\frac{rad}{s}]$.

En las Figuras 6.5, 6.6 y 6.7 se muestran las salidas del filtro de posición ϑ o las estimaciones de velocidad angular, cuando el PI, la RNA y el FSLC son los controladores de velocidad, respectivamente. En la respuesta del controlador PI, se observa más claramente como el rizado de par afecta considerablemente a este controlador; el cual, no puede mantener una velocidad constante y en cambio la velocidad tiene variaciones de hasta $1.6[\frac{rad}{s}]$. En el resultado obtenido con la RNA se observa claramente, el periodo de tiempo que tarda en aprender el controlador; en donde, se presentan variaciones de velocidad de más de $1.3[\frac{rad}{s}]$ y posterior a la etapa de aprendizaje, la velocidad se mantiene más constante y cerca del valor deseado. Por último, en la respuesta del FSLC, se observa que en los primeros $t = 1.415[s]$ la velocidad tiene un valor pico que alcanza los $2.9[\frac{rad}{s}]$, mientras se ajustan los coeficientes del controlador. Sin embargo, después de este periodo, la velocidad se mantiene considerablemente constante y alrededor de la velocidad deseada. Cuando se deja caer la masa, en $t = 50[s]$, la velocidad se ve afectada para los tres controladores. En los casos del PI y el FSLC, después de compensar la perturbación, estos controladores tienen un comportamiento parecido a su comportamiento anterior a la perturbación; sin embargo, en el caso de la RNA la perturbación de par ocasiona una variación más grande que con los otros dos controlado-

res, de hasta $-1.3[\frac{rad}{s}]$. Adicionalmente, se puede observar que las variaciones de velocidad aumentan considerablemente y perduran hasta el final del experimento, cuando la RNA es el controlador maestro.

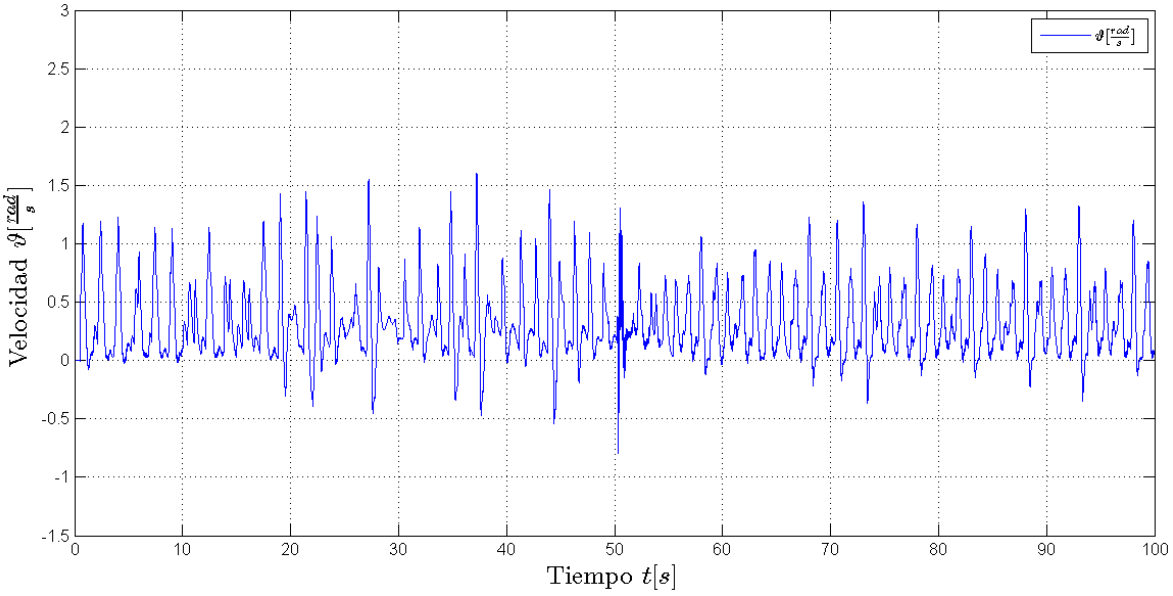

Figura 6.5: Velocidad angular cuando el controlador es el PI, para una referencia de $0.3142[\frac{rad}{s}]$.

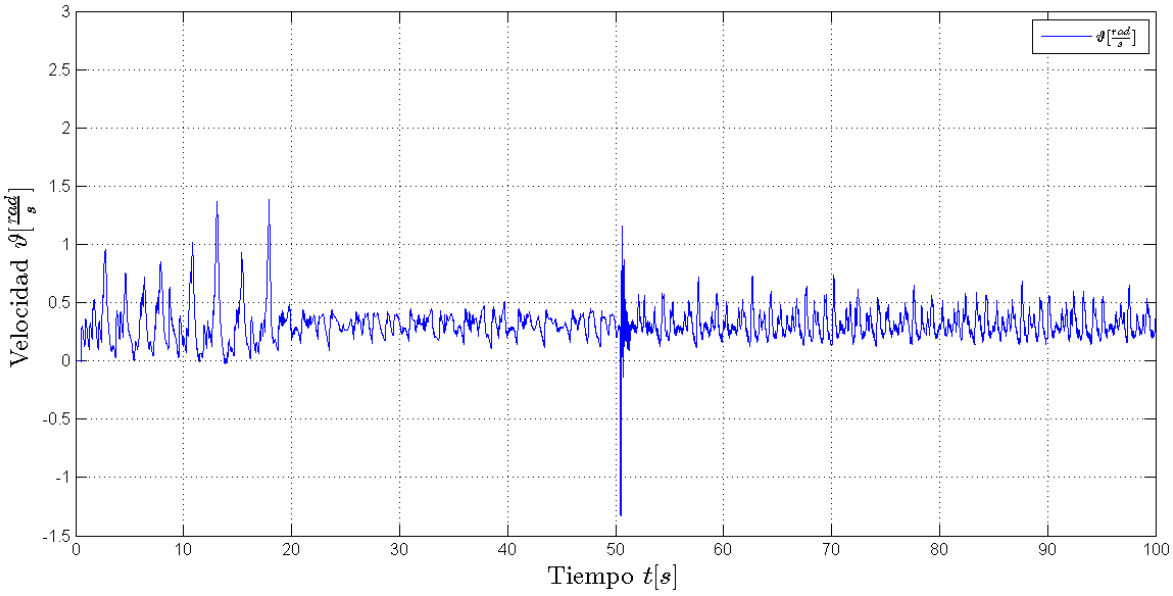

Figura 6.6: Velocidad angular cuando el controlador es la RNA, para una referencia de $0.3142[\frac{rad}{s}]$.

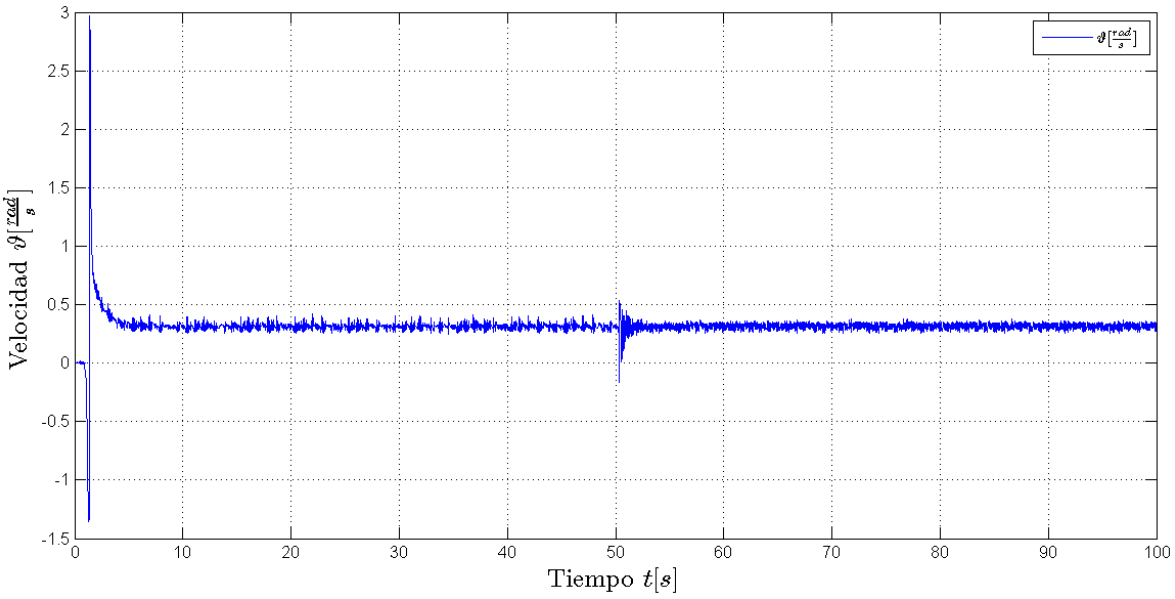

Figura 6.7: Velocidad angular cuando el controlador es el FSLC, para una referencia de $0.3142[\frac{rad}{s}]$.

En las Figuras 6.8, 6.9 y 6.10 se muestran las señales de control I_{qd} calculadas (deseadas) por de los controladores de velocidad PI, RNA y FSLC respectivamente; así como, las corrientes I_q medidas en cada experimento. En el caso del controlador PI, se observa que la corriente deseada y la corriente medida son bastante parecidas durante todo el experimento; además, el valor absoluto de la corriente máxima que se obtiene, es de $0.3[A]$ (ver Figura 6.8). Para el caso de la RNA como controlador, se observa que la corriente deseada y la corriente medida también son bastante parecidas; sin embargo, la corriente máxima que se presenta, es de $0.74[A]$, lo cual ocurre durante el tiempo de aprendizaje de la red (ver Figura 6.9). En el caso del FSLC como controlador, se observa que la corriente deseada y la corriente medida son ligeramente diferentes y durante el periodo de aprendizaje se satura la corriente deseada (mediante programación) en $6[A]$ y $-6[A]$; sin embargo, la corriente medida tiene un valor máximo de $3.4[A]$ y después de $t = 2[s]$ se mantienen valores de corrientes cercanos a cero y la corriente medida sigue de cerca a la corriente deseada (ver Figura 6.10). Cuando la perturbación de par es inducida, los tres controladores reaccionan aumentando la corriente deseada I_{qd} y con ella, aumenta la corriente medida I_q hasta mantenerse oscilando alrededor de $0.4[A]$ durante el resto del experimento.

No obstante, la corriente deseada más grande que se obtiene, es cuando el FSLC es el controlador mestro, en donde la corriente I_{qd} alcanza los $2.9[A]$; sin embargo, la corriente I_q no llega hasta ese valor, sino hasta $0.87[A]$ únicamente.

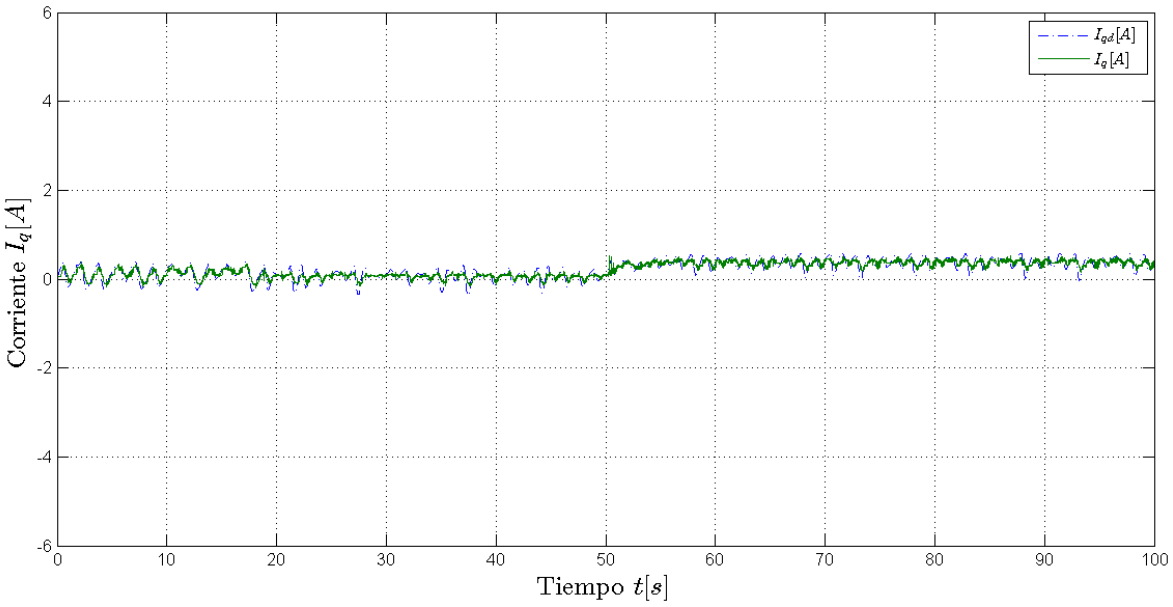

Figura 6.8: Señal de control I_{qd} calculada por el PI y corriente I_q medida.

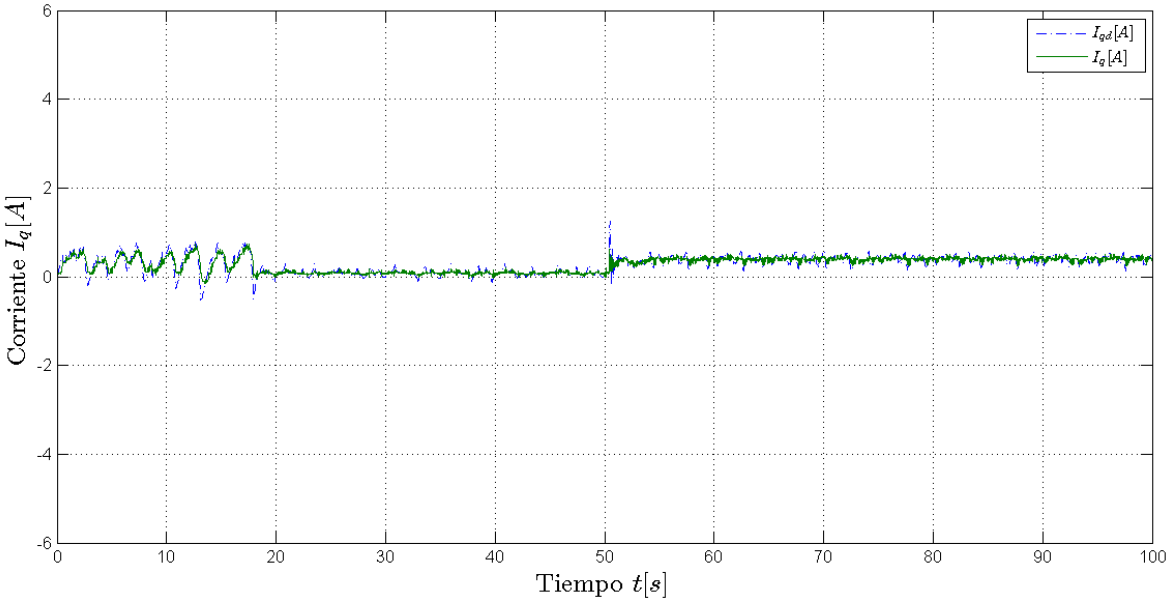

Figura 6.9: Señal de control I_{qd} calculada por la RNA y corriente I_q medida.

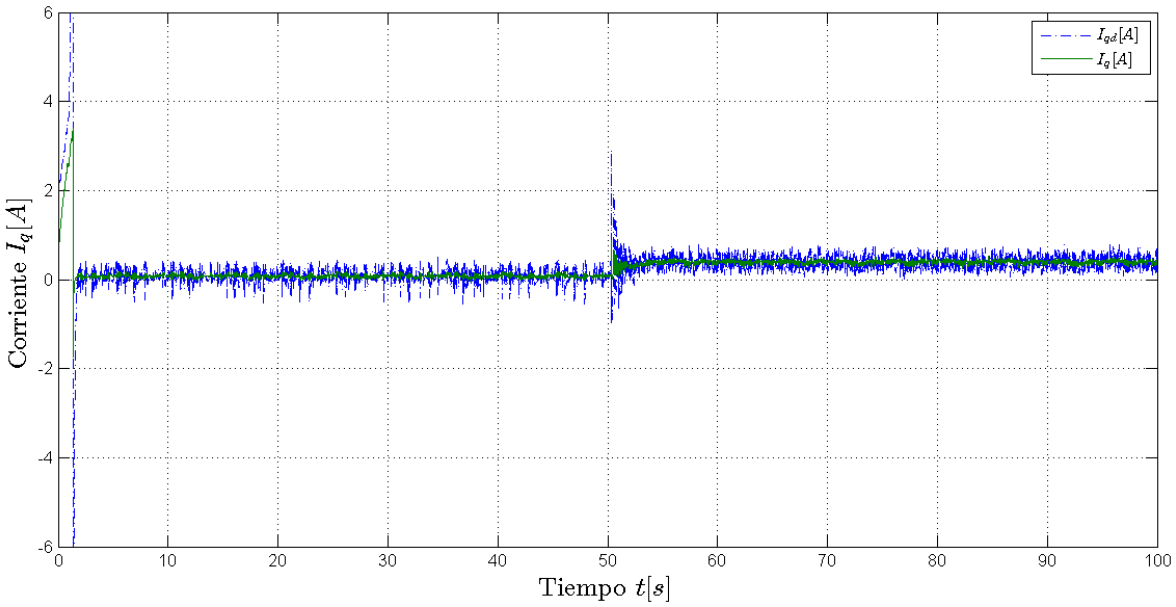

Figura 6.10: Señal de control I_{qd} calculada por el FSLC y corriente I_q medida.

En las Figuras 6.11, 6.12 y 6.13 se muestran las corrientes I_d obtenidas cuando se usan los controladores de velocidad PI, RNA y FSLC respectivamente. En estas gráficas se observa la corriente I_d no se ve muy afectada con el uso de estos controladores. Sin embargo, en el caso del FSLC (Figura 6.13) se observa un pico de corriente de $2.89[A]$ durante el aprendizaje en $t = 1.38[s]$ aproximadamente; a pesar de esto, la corriente I_d en los tres controladores se mantiene cerca de su referencia $0[A]$ en la mayor parte del experimento. Cuando la perturbación de par es inducida, la corriente I_d mantiene su comportamiento sin cambios relevantes en los tres casos.

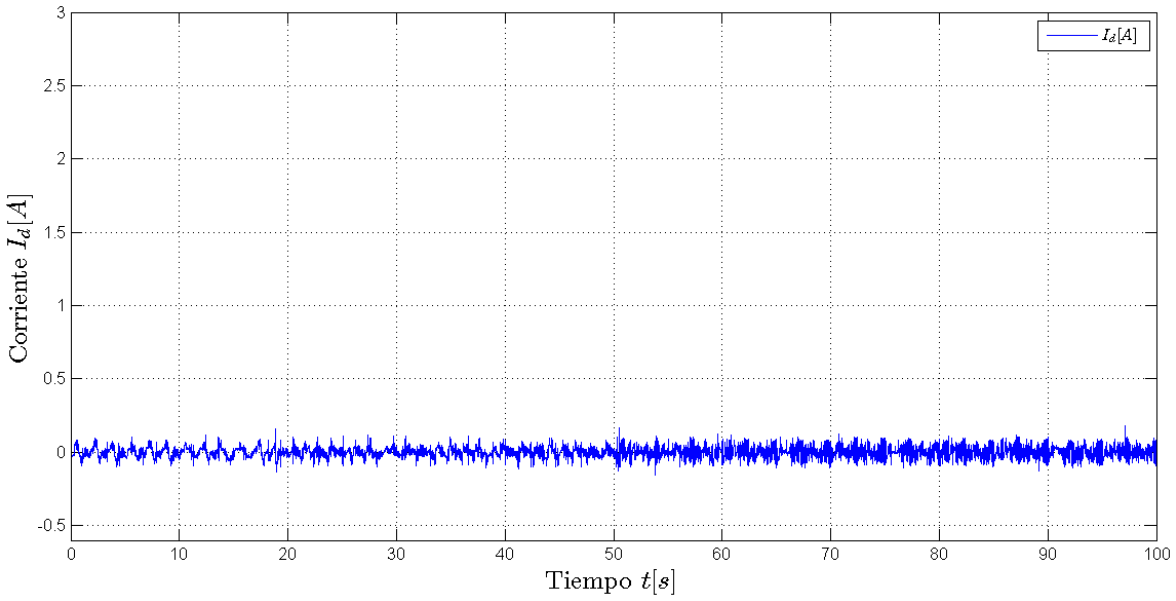

Figura 6.11: Corriente I_d cuando el controlador de velocidad es el PI.

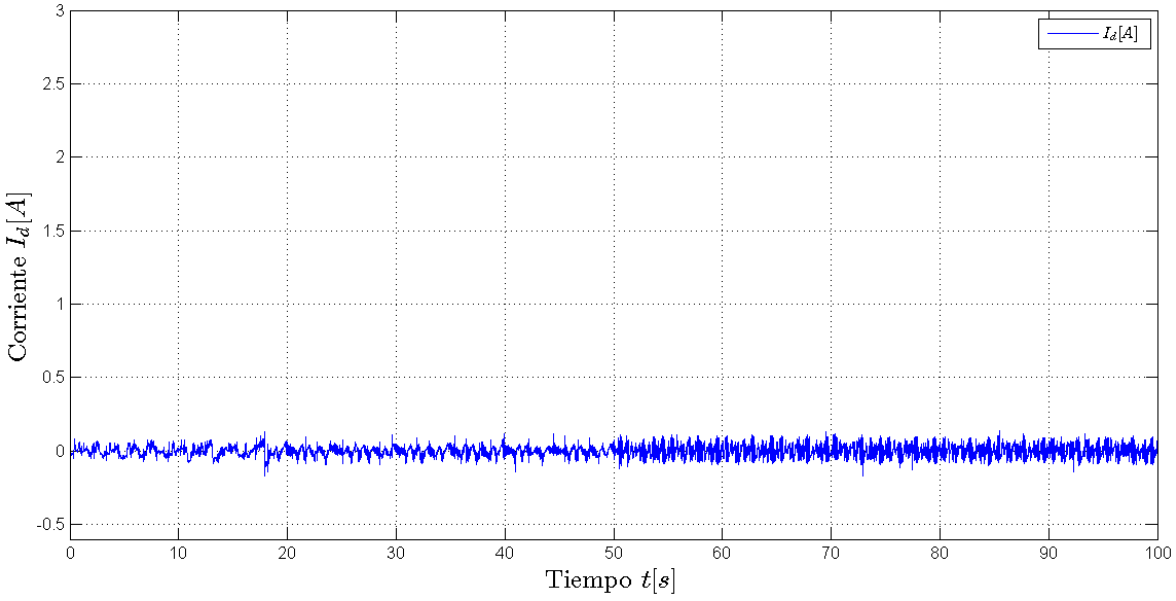

Figura 6.12: Corriente I_d cuando el controlador de velocidad es la RNA.

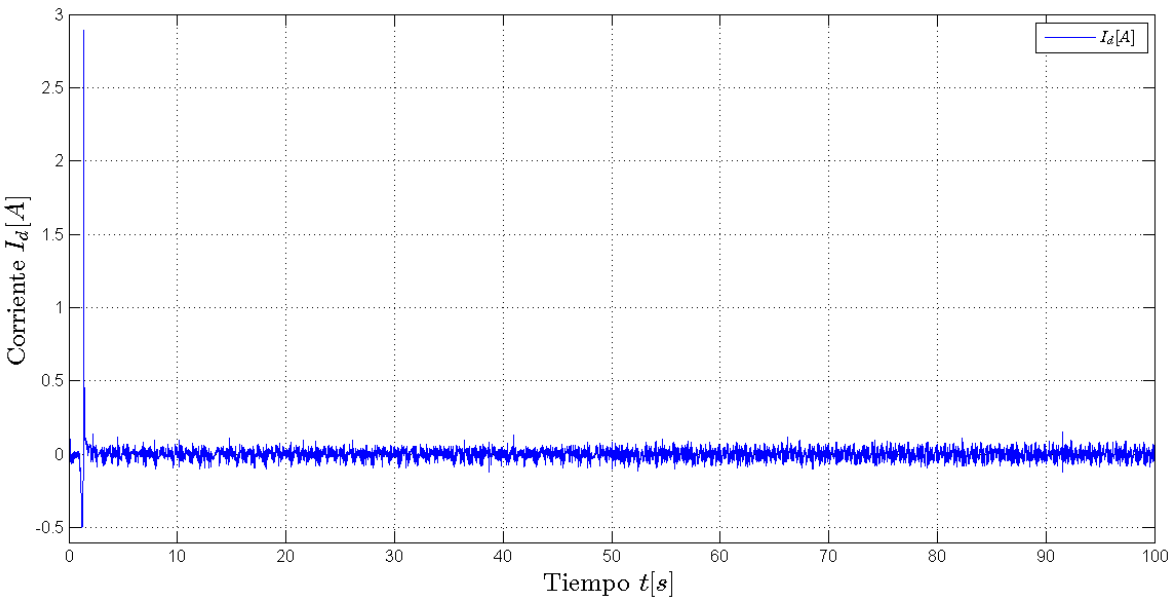

Figura 6.13: Corriente I_d cuando el controlador de velocidad es el FSLC.

En la Figura 6.14 se presenta el error obtenido cuando se controla la velocidad del PMSM con el controlador PI. En esta gráfica, se observa el comportamiento real de este controlador. Es muy visible, que el controlador PI no puede lograr un error considerablemente cercano a cero y en cambio se observan grandes variaciones del error; las cuales, pueden ser ocasionadas por el rizado de par, además, se percibe que el error alcanza valores de hasta $-1.2[\frac{rad}{s}]$. Nótese, que para tener una constante visual efectiva en el comportamiento de los controladores, la amplitud en el eje vertical de las gráficas del error, se deja en el rango de $-3.5[\frac{rad}{s}]$ a $1.5[\frac{rad}{s}]$. Cuando la perturbación de par es inducida en $t = 50[s]$ el sistema se ve afectado, alcanzando un error de $1.109[\frac{rad}{s}]$; sin embargo, el PI compensa esta perturbación y la señal de error se mantiene con el mismo comportamiento; es decir, sin compensar el rizado de par y con grandes variaciones hasta el final del experimento.

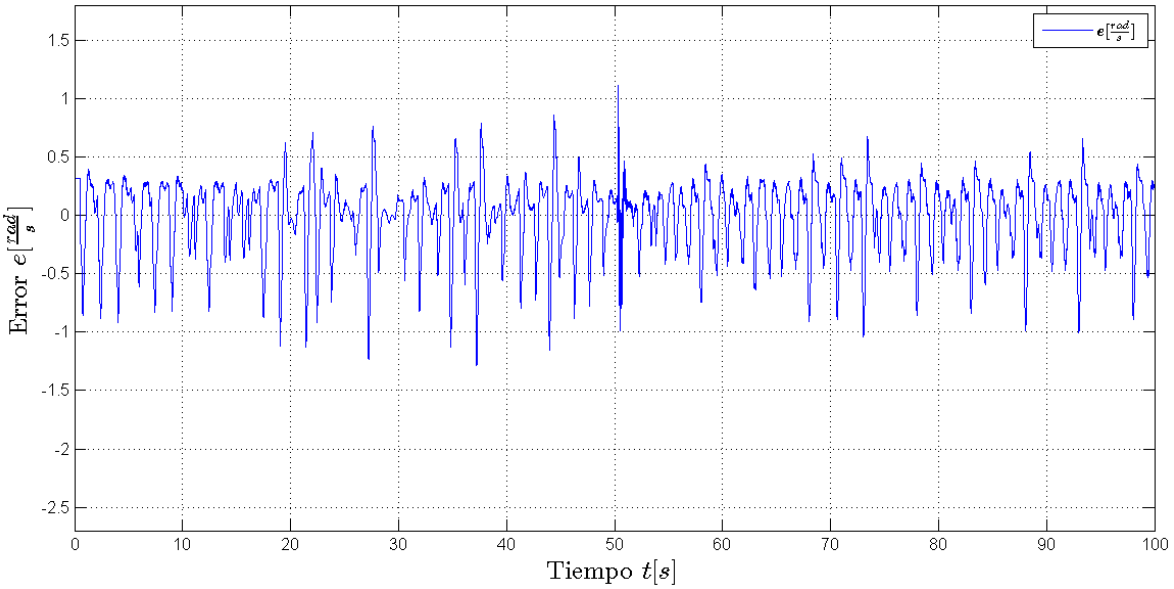

Figura 6.14: Error obtenido cuando el controlador maestro es el PI.

En la Figura 6.15 se muestra el error obtenido cuando se controla la velocidad del motor con la RNA. En esta gráfica se observa que al inicio del experimento, durante la etapa de aprendizaje, el error que se genera es de hasta $-0.4[\frac{rad}{s}]$; lo cual, es notoriamente menor, comparado con los errores alcanzados por el controlador PI. Después de la etapa de aprendizaje, el error que se obtiene con la RNA se mantiene acotado y permanece entre $-0.075[\frac{rad}{s}]$ y $0.1[\frac{rad}{s}]$. Cuando se induce la perturbación de par, el error del sistema alcanza los $0.4[\frac{rad}{s}]$ y posteriormente, la RNA compensa la perturbación acotando el error y manteniendolo alrededor de cero; sin embargo, el rango de variaciones aumenta y ahora el error se mantiene entre $-0.174[\frac{rad}{s}]$ y $0.084[\frac{rad}{s}]$, lo cual perdura hasta el final del experimento. Es decir, la RNA si se ve afectada por una perturbación de par constante a pesar de compensarla y mantener el error cerca de cero.

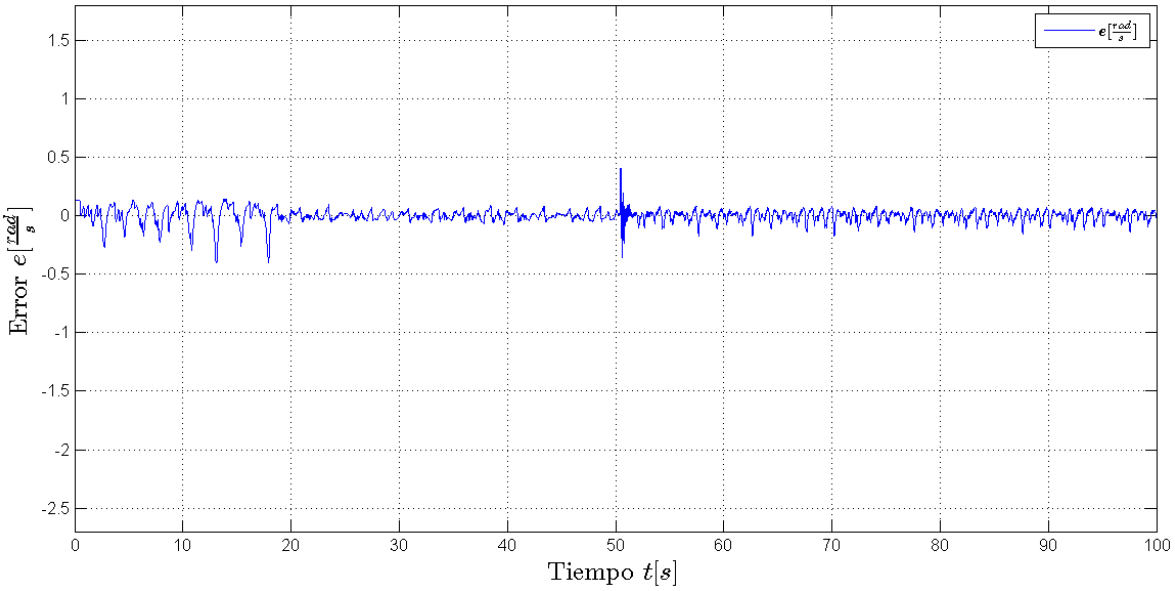

Figura 6.15: Error obtenido cuando el controlador maestro es la RNA.

En la Figura 6.16 se presenta el error obtenido con el FSLC como controlador maestro de velocidad. En esta gráfica se observa el resultado más relevante del controlador diseñado en esta investigación. Se puede ver que en los primeros $t = 1.0[s]$ el error obtenido con el FSLC es menor de $0.5[\frac{rad}{s}]$. Después, en el segundo 1.31 aproximadamente, ocurre un cambio brusco de velocidad, como parte del auto ajuste de la serie de Fourier y se alcanza un error de hasta $-2.65[\frac{rad}{s}]$; sin embargo, esto ocurre durante $t = 0.8[s]$ únicamente. Posterior a ello, el error converge y se mantiene acotado entre $-0.1[\frac{rad}{s}]$ y $0.06[\frac{rad}{s}]$, bastante cerca de cero, a diferencia del controlador PI. Posteriormente, en $t = 50[s]$ se induce la parturbación de par y el error alcanza los $0.48[\frac{rad}{s}]$, en seguida el FSLC somete a la perturbación y el error vuelve a converger y a mantenerse cerca de cero. Sin embargo, a diferencia de la RNA, el rango de variaciones del error se reducen ligeramente a $-0.06[\frac{rad}{s}]$ y $0.06[\frac{rad}{s}]$ en donde se mantiene acotado el error. Luego, se demuestra experimentalmente que el FSLC logra la convergencia del error a cero, de un sistema no lineal, aunque el sistema esté sometido a perturbaciones periódicas y a perturbaciones constantes aleatorias.

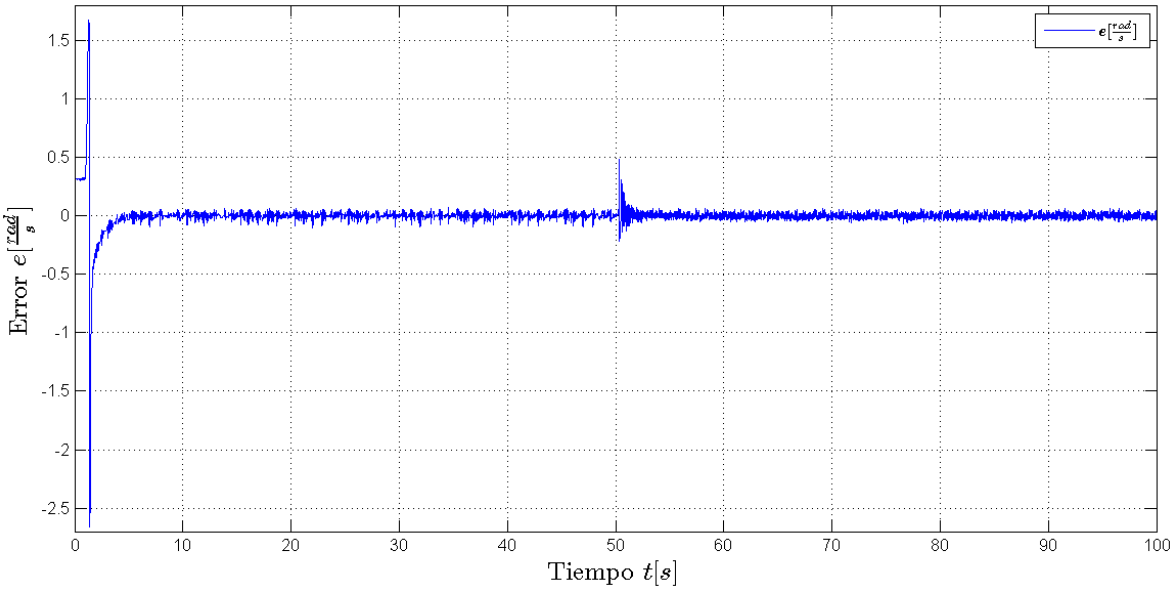

Figura 6.16: Error obtenido cuando el controlador maestro es el FSLC.

De esta manera, se puede observar que el FSLC es una opción viable para el control de velocidad de un PMSM. Ahora, se presentan más experimentos con este controlador; con el fin de mostrar su comportamiento cuando se agregan más términos a la serie de Fourier. No obstante, en primer lugar, se desea justificar el uso del filtro de posición para la estimación de velocidad; ya que, es importante darse cuenta que, la omisión de este filtro podría causar un desempeño no deseado en el sistema de control.

En la Figura 6.17 se muestra la comparación de la velocidad estimada con el filtro de posición (ϑ línea solida), contra la derivada numérica de la posición angular ($\dot{\theta}_N(k)$ línea punteada). Como se puede observar, es muy notable la diferencia entre la estimación de velocidad con el filtro y la derivada numérica; el filtro de posición es más viable para la obtención de la velocidad angular del motor ya que elimina bastante ruido que se genera al derivar numéricamente la posición angular. Nótese, que a pesar de que se usa una velocidad baja, el ruido es muy grande y es bastante complicado distinguir una velocidad útil. Por esta razón, se recomienda usar el filtro de posición, sobre todo, si el encoder que se utiliza, es de alta resolución.

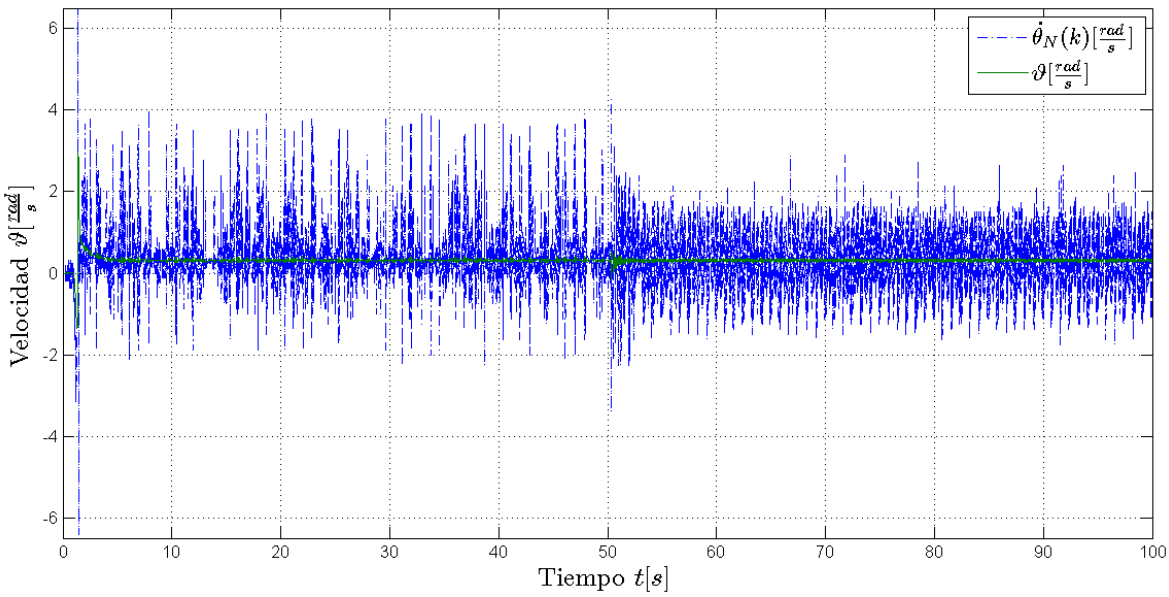

Figura 6.17: Velocidad calculada con la derivada numérica $\dot{\theta}_N(k) = \frac{\theta(k)-\theta(k-1)}{T_s}$ y estimación de velocidad con el filtro de posición ϑ, cuando se controla la velocidad con el FSLC a $0.3142[\frac{rad}{s}]$.

Cuando se aumenta el número de términos del FSLC como controlador de velocidad, a $N = 10$, $N = 20$ y $N = 100$, se obtienen los errores que se muestran en las Figuras 6.18, 6.19 y 6.20 respectivamente. Como se puede observar, la convergencia del error no se afecta con el aumento de términos en el controlador y la diferencia entre los resultados obtenidos, es muy sutil. Esto se debe a que, aumentando el número de términos N no se obtienen frecuencias más grandes, ya que la frecuencia más grande que se puede representar con la DFT, es la frecuencia de Nyquist; la cual, depende únicamente de la frecuencia de muestreo. Entonces, cuando se aumenta el valor de N, se obtienen más frecuencias proporcionales a la frecuencia de Nyquist; de esta manera, el FSLC únicamente se encarga de asegurar que los armónicos en cada experimento tiendan a los valores necesarios para que el error del sistema en lazo cerrado llegue a cero. Esto depenede de cada planta a controlar y de las carcterísticas de las perturbaciones que se puedan presentar en un sistema dado.

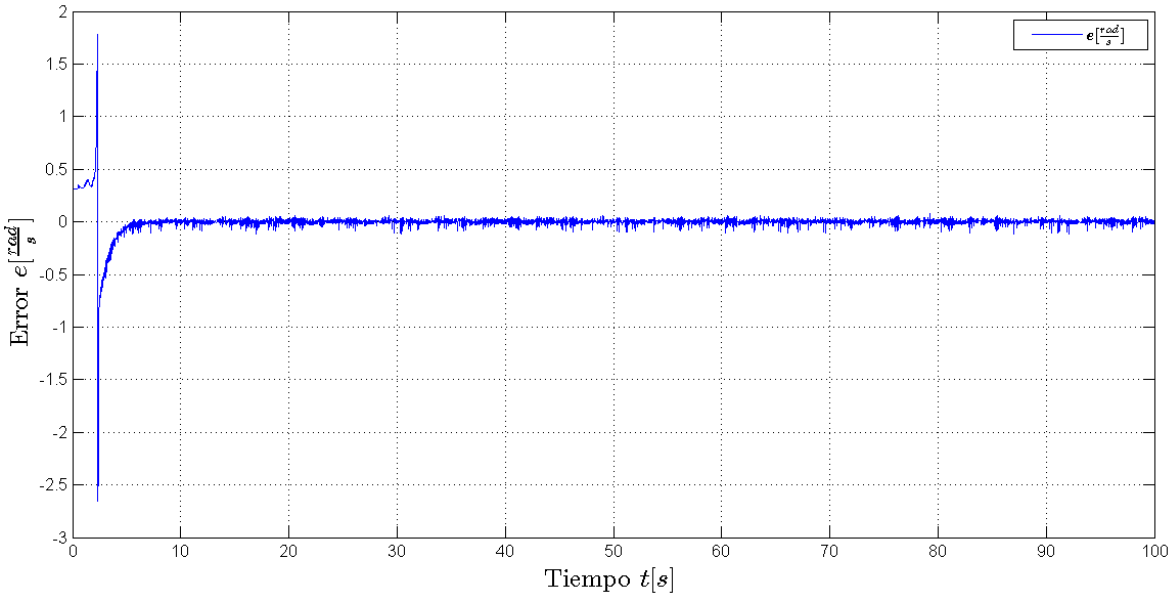

Figura 6.18: Error obtenido cuando el controlador maestro es el FSLC y el número de términos de la serie es $N = 10$.

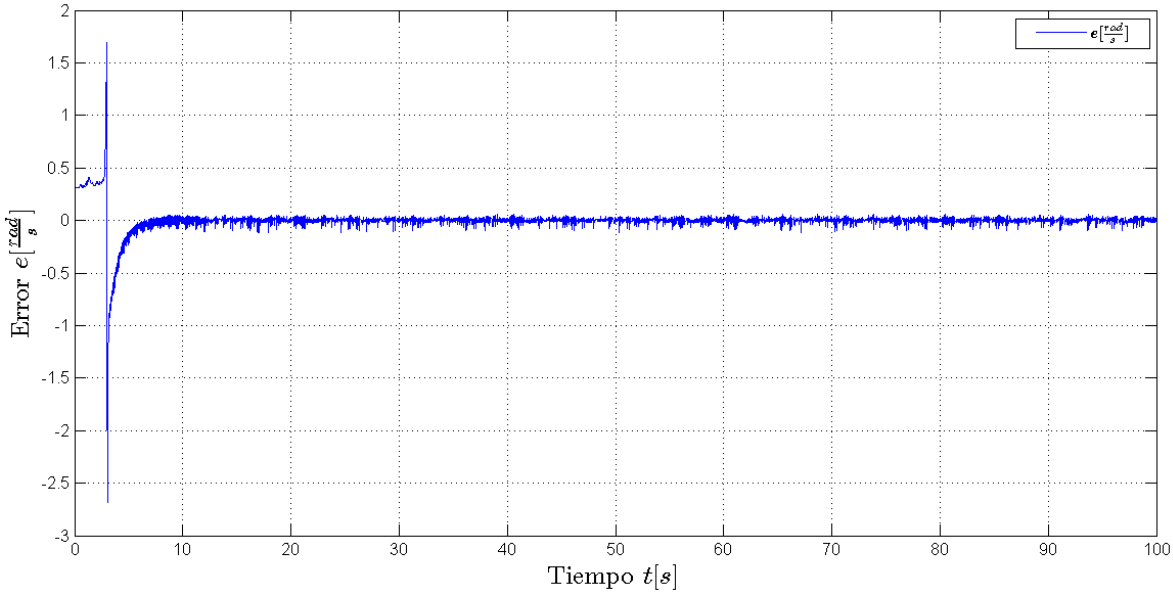

Figura 6.19: Error obtenido cuando el controlador maestro es el FSLC y el número de términos de la serie es $N = 20$.

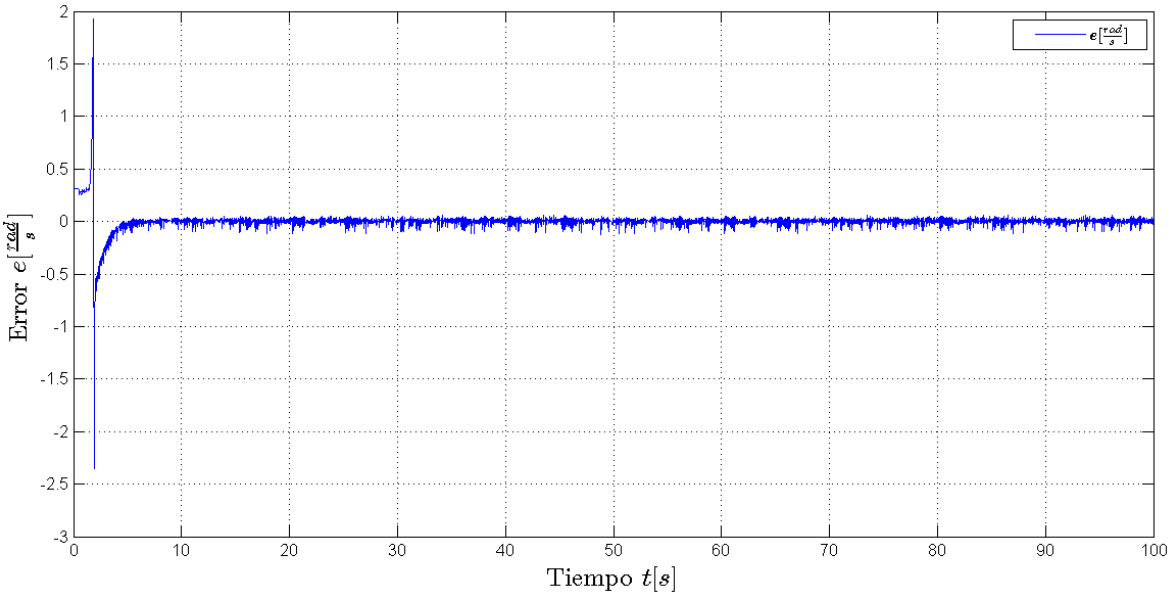

Figura 6.20: Error obtenido cuando el controlador maestro es el FSLC y el número de términos de la serie es $N = 100$.

Capítulo 7

CONCLUSIONES

En este trabajo de investigación se diseñó una serie de Fourier como controlador auto ajustable. Se diseñaron las reglas de actualización para sus coeficientes y se realizó un análisis de convergencia del error; en donde se demostró que, para las ganancias adecuadas de α_n y γ_n, el FSLC es estable y se logra la convergencia del error a cero.

Adicionalmente, se aplicó el FSLC como controlador de velocidad de un PMSM, en bajas velocidades y se demuestra experimentalmente, que el FSLC reduce las perturbaciones periódicas a las que está sometido el motor, eliminando el rizado de par. Además, se compara el comportamiento del controlador diseñado en esta investigación, con una red neuronal artificial y con un controlador tradicional PI; los tres, como controladores de velocidad del PMSM sobre la base del método FOC. Se observó que el controlador tradicional PI no puede compensar las perturbaciones generadas por el rizado de par, en cambio la RNA si logra compensar estas perturbaciones periódicas; sin embargo, a la RNA le toma $t = 18[s]$ auto ajustarse para minimizar el rizado de par. En cambio, al FSLC le toma únicamente $t = 2[s]$ auto ajustarse para reducir las perturbaciones generadas por el rizado de par y se logra la convergencia del error; el cuál, se mantiene alrededor de $0[\frac{rad}{s}]$.

Además, se indujo una perturbación de par constante, construida a partir de un hilo cáñamo, sujetado de un extremo a la flecha del motor, al cual se le asegura una masa conocida en el otro extremo; esta masa, se deja caer en $t = 50[s]$ para simular una perturbación constante y aleatoria para los tres controladores. Se analizó la reacción de los controladores ante esta perturbación y se observó que el comportamiento de la RNA se vió afectado con variaciones más grandes del error después de haber inducido la perturbación. Sin embargo, los controladores PI y FSLC lograron compensar esta perturbación sin problemas.

Se demuestra teóricamente y en la práctica que el FSLC logra la convergencia del error a cero de un sistemas no lineal, aunque el sistema esté sometido a perturbaciones periódicas y/o perturbaciones constantes aleatorias. Entonces, el controlador diseñado en esta investigación, es una buena alternativa para otros sistemas no lineales. Sin embargo, existen más ventajas importantes del FSLC, gracias a su naturaleza en el dominio de la frecuencia. Por ejemplo, con este controlador se podrían detectar ciertas fallas en los equipos con partes mecánicas rotativas; de la siguiente manera: sí, alguna parte rotativa del sistema no funciona adecuadamente, se podría generar una perturbación periódica en el actuador, en cuyo caso, el controlador reaccionaría para compensar esta perturbación, agregando el armónico equivalente; de esta manera, monitoreando los armónicos que el controlador agrega, se podría suponer la causa de ciertos armónicos, como alguna parte dañada en el sistema. Esta aplicación puede ser parte de un trabajo futuro; además, de la aplicación del FSLC en otros sistemas complejos.

Adicionalmente, se puede lograr la diminución en el tiempo de cómputo del FSLC con el uso de la Transformada Rápida de Fourier (FFT, por sus siglas en inglés) en lugar de la DFT; lo cual, es también considerado para un trabajo futuro. Cabe mencionar, que a pesar de que el FSLC requiere más capacidad de memoria y procesamiento que el controlador tradicional PID, el FSLC no es tan costoso computacionalmente como la RNA; esto se puede ver claramente, en la cantidad de ciclos y líneas de programación que se utilizaron para estos controladores (adjuntos en el apéndice A).

Por último, se desarrolló un banco de pruebas para el control de motores trifásicos de baja potencia y se muestran los circuitos utilizados; así como, los componentes más importantes y sus características. Además, el código utilizado para programar el microcontrolador; así como los programas de los controladores y el programa en MATLAB para la obtención de las gráficas, se muestran en el apéndice A. Este banco de pruebas, junto con la teoría del nuevo controlador, servirán para nuevas investigaciones.

BIBLIOGRAFÍA

Adhami-Mirhosseini, Aras, Mohammad J. Yazdanpanah and A. Pedro Aguiar. 2014. Automatic bottom-following for underwater robotic vehicles. Automatica 50(8):2155–2162.

Adhavan, B., A. Kuppuswamy, G. Jayabaskaran and V. Jagannathan. 2011. Field oriented control of Permanent Magnet Synchronous Motor (PMSM) using fuzzy logic controller. Recent Advances in Intelligent Computational Systems (RAICS), 2011 IEEE. IEEE :587–592.

Aghili, Farhad. 2008. Adaptive reshaping of excitation currents for accurate torque control of brushless motors. Control Systems Technology, IEEE Transactions on 16(2):356–364.

Akin, Bilal and Manish Bhardwaj. 2013. Sensored Field Oriented Control of 3-Phase Induction Motors. Texas Instruments Incorporated .

Anaheim, Automation. 2016. EMJ-AC Servo Motors with Encoder. Online; accessed 08 September 2016.
URL: *http://www.anaheimautomation.com/products/servo/servo-motor-item.php?sID=218&serID=2&pt=i&tID=350&cID=27*

Briggs, William L. and Van H. Emden. 1995. The DFT: an owner's manual for the discrete Fourier transform. Siam.

Cai, Lilong and Weiqing Huang. 1999. Fourier series based learning control and application to positioning table. Robotics and Autonomous systems 32(2):89–100.

Cara, F. Javier. 2012-2013. Análisis de Fourier. Curso ETSII-UPM.

Chang, CM and TS Liu. 2006. A wavelet network control method for disk drives. Control Systems Technology, IEEE Transactions on 14(1):63–68.

Chen, H-Y and J-W Liang. 2010a. Adaptive sliding control with self-tuning fuzzy compensation for a piezoelectrically actuated X–Y table. IET control theory & applications 4(11):2516–2526.

Chen, Hung-Yi and Jin-Wei Liang. 2010b. Model-Free Adaptive Sensing and Control for a Piezoelectrically Actuated System. Sensors 10(12):10545–10559.

Cho, Hancheol, Sang-Young Park, Han-Earl Park and Kyu-Hong Choi. 2012. Analytic Solution to Optimal Reconfigurations of Satellite Formation Flying in Circular Orbit under J_2 Perturbation. Aerospace and Electronic Systems, IEEE Transactions on 48(3):2180–2197.

Cui, Xianzhong and Kang G. Shin. 1993. Direct control and coordination using neural networks. Systems, Man and Cybernetics, IEEE Transactions on 23(3):686–697.

DIY, Electronics. 2016. FT232RL USB to Serial Adapter for PIC AVR ATMEGA ARDUINO MCUs. Online; accessed 08 September 2016.
URL: *http://electronics-diy.com/FT232RL_USB_to_Serial_Adapter_for_PIC_AVR_ATMEGA_ARDUINO_MCUs.php*

Dogruel, Murat and Hasan Hüseyin Celik. 2011. Harmonic Control Arrays Method With a Real Time Application to Periodic Position Control. IEEE transactions on control systems technology 19(3):521–530.

Dong, Liang and Wen Cheng Tang. 2014. Adaptive backstepping sliding mode control of flexible ball screw drives with time-varying parametric uncertainties and disturbances. ISA transactions 53(1):110–116.

Freeman, James A. and David M. Skapura. 1993. Redes Neuronales, Algoritmos, Aplicaciones y Técnicas de programación. Addison-Wesley Publishing Company, Inc. Massachusetts, E.U.A.

Gómez, Espinosa Alfonso. 1999. Criterios de diseño del controlador neuronal autoajustable y su aplicación a un sistema no lineal. Tesis de maestría Universidad Autónoma de Querétaro.

Gómez-Espinosa, Alfonso, Víctor M. Hernández-Guzmán, Manuel Bandala-Sánchez, Hugo Jiménez-Hernández, Edgar A. Rivas-Araiza, Juvenal Rodríguez-Reséndiz and Gilberto Herrera-Ruíz. 2013. A New Adaptive Self-Tuning Fourier Coefficients Algorithm for Periodic Torque Ripple Minimization in Permanent Magnet Synchronous Motors (PMSM). Sensors 13(3):3831–3847.

Hanselman, Duane C. 2003. Brushless permanent magnet motor design. The Writers' Collective.

Hehn, Markus and Raffaello D. Andrea. 2014. A frequency domain iterative learning algorithm for high-performance, periodic quadrocopter maneuvers. Mechatronics 24(8):954–965.

Hernández-Guzmán, VM, RV Carrillo-Serrano and A. Gómez-Espinosa. 2014. L2-stable velocity ripple minimization in PM synchronous motors. European Journal of Control 20(3):111–117.

Hilera, José R. and Víctor J. Martínez. 1995. Redes Neuronales Artificiales, Fundamentos, Modelos y Aplicaciones. RA-MA Editorial Madrid, España.

Huang, Weiqing and Lilong Cai. 2000. New hybrid controller for systems with deterministic uncertainties. Mechatronics, IEEE/ASME Transactions on 5(4):342–348.

Hung, JY and Z. Ding. 1993. Design of currents to reduce torque ripple in brushless permanent magnet motors. Electric Power Applications, IEE Proceedings B 140(4):260–266.

Hwang, C-L. 1997. Fourier series neural network-based adaptive variable structure control for servo systems with friction. Control Theory and Applications, IEE Proceedings. 144 IET :559–565.

Kang, Chang-Ik, Sang-Eun Baek and Jun-Seok Shim. 2004. A new seek servo controller for minimizing power consumption in micro hard disk drives. Magnetics, IEEE Transactions on 40(4):3127–3129.

Kelly, R., R. Ortega, A. Ailon and A. Loria. 1994. Global regulation of flexible joint robots using approximate differentiation. Automatic Control, IEEE Transactions on 39(6):1222–1224.

Khalil, Ahmed and Iqbal Husain. 2007. A fourier series generalized geometry-based analytical model of switched reluctance machines. Industry Applications, IEEE Transactions on 43(3):673–684.

Krishnan, Ramu. 2009. Permanent magnet synchronous and brushless DC motor drives. CRC press.

Lee, Dong-Choon and G-Myoung Lee. 1998. A novel overmodulation technique for space-vector PWM inverters. Power Electronics, IEEE Transactions on 13(6):1144–1151.

Lozano-Guzmán, Alejandro A. and Juan Carlos A. Jáuregui-Correa. 2013. Las vibraciones mecánicas en el mantenimiento predictivo. FUNDAP México.

Lyshevski, S. E. 2000. Electromechanical Systems, Electric Machines and Applied Mechatronics. CRC Press: Boca Raton FL, U.S.A.

Manual, Automation Anaheim. 2016. EMJ-04 Series Servo Motor. 910 East Orangefair Ln. Anaheim, CA 92801 Anaheim Automation. Online; accessed 08 September 2016: https://www.anaheimautomation.com/manuals/servo/L010976 %20- %20EMJ-04 %20Servo %20Motor.pdf.

Marulanda, Felipe A. and Julián Andrés Herrera. 2010. Control de velocidad de un motor de inducción por el método de control vectorial utilizando el software de simulación Simulink de Matlab. Tomado el día 13 de Abril de 2016, disponible en internet: http://repositorio.utp.edu.co/dspace/bitstream/handle/11059/1775/621313M389.pdf.

Mattavelli, Paolo, Luca Tubiana and Mauro Zigliotto. 2005. Torque-ripple reduction in PM synchronous motor drives using repetitive current control. IEEE transactions on power electronics 20(6):1423–1431.

Mikail, Rajib, Iqbal Husain, Yilmaz Sozer, Mohammad S. Islam and Tomy Sebastian. 2013. Torque-ripple minimization of switched reluctance machines through current profiling. Industry Applications, IEEE Transactions on 49(3):1258–1267.

Norte, Electrónica 60. 2016. LMD18200 puente H. Online; accessed 08 September 2016.
URL: *http://www.electronica60norte.com/detalle.php?sku=582*

Openhacks. 2016. Sensor de Corriente Hall 5A - ACS712. Online; accessed 08 September 2016.
URL: *http://www.openhacks.com/page/productos/id/255/title/Sensor-de-Corriente-Hall-5A—ACS712*

Ponce, A. Noriega, A. Aguado Behar, A. Ordaz Hernández and V. Rauch Sitar. 2004. Neural networks for self-tuning control systems. Acta Polytechnica 44(1).

Qian, Weizhe, Sanjib K. Panda and Jian-Xin Xu. 2004. Torque ripple minimization in PM synchronous motors using iterative learning control. Power Electronics, IEEE Transactions on 19(2):272–279.

Rodrigo507. 2010. series: SERIE DE FOURIER. Online; accessed 08 September 2016.
URL: *http://seriesdefoulier.blogspot.com/2010/04/serie-de-fourier.html*

Sathikumar, S. and Joseph Vithayathil. 1984. Digital simulation of field-oriented control of induction motor. IEEE Transactions on Industrial Electronics 31(2):141–148.

Shouse, Kenneth R. and David G. Taylor. 1994. A digital self-tuning tracking controller for permanent-magnet synchronous motors. Control Systems Technology, IEEE Transactions on 2(4):412–422.

Sitenordeste. 2015. Vibraciones mecánicas - Máquinas rotativas - con movimiento alternativo y lineal. Online; accessed 08 September 2016.
URL: *http://www.sitenordeste.com/mecanica/vibraciones_mecanicas.htm*

Smith, Julius O. 2007. Mathematics of the discrete Fourier transform (DFT): with audio applicaitons. Julius Smith.

Štumberger, Bojan, Gorazd Štumberger, Miralem Hadžiselimović and Ivan Zagradišnik. 2006. Torque ripple reduction in exterior-rotor permanent magnet synchronous motor. Journal of Magnetism and Magnetic Materials 304(2):e826–e828.

Tang, Xiaoqi, Lilong Cai and Weiqing Huang. 2000. A learning controller for robot manipulators using Fourier series. Robotics and Automation, IEEE Transactions on 16(1):36–45.

Wu, XH, SK Panda and JX Xu. 2008. DC Link Voltage and Supply-Side Current Harmonics-Minimization of Three Phase PWM BoostRectifiers Using Frequency Domain BasedRepetitive Current Controllers. Power Electronics, IEEE Transactions on 23(4):1987–1997.

Zhu, C. and FW Paul. 1995. A Fourier series neural network and its application to system identification. Journal of dynamic systems, measurement, and control 117(3):253–261.

Zuo, Wei and Lilong Cai. 2010. A new iterative learning controller using variable structure Fourier neural network. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 40(2):458–468.

A. PROGRAMAS UTILIZADOS

A.1. Programa para el dsPIC33FJ12MC202

```
#include <33FJ12MC202.h>
#device ADC=12
#include<stdlib.h>
#include<math.h>

#FUSES NOWDT                    //No Watch Dog Timer
#FUSES NOPUT                    //No Power Up Timer
#FUSES CKSFSM                   //Clock Switching is enabled
#FUSES NOJTAG                   //JTAG disabled
#FUSES PR_PLL
//#FUSES NOWRT,NOPROTECT,EC,FRC_PLL
#FUSES NOWRTB,NOPROTECT,HS,FRC_PLL

#use delay(clock=48000000)
#pin_select QEA1=PIN_B14
#pin_select QEB1=PIN_B12
#pin_select INDX1=PIN_B10
#Pin_select U1TX=PIN_B6
#Pin_select U1RX=PIN_B7
#use rs232(UART1,baud=460800,BITS=8,PARITY=N)//ConFigura P. Serie

#byte PORTB = 0x02CA
#byte TRISA = 0x02C0
#byte TRISB = 0x02C8
#byte AD1PCFGL = 0x032C
#byte AD1CSSL = 0x0330
#byte AD1CHS123 = 0x0326
#byte AD1CON1 = 0x0320
#byte AD1CON2 = 0x0322
#byte AD1CON3 = 0x0324
#byte T1CON = 0x0104
#byte TMR1 = 0x0100
#byte PR1 = 0x0102

#bit  PB4   = 0x02CA.4
```

```
#bit  PB8   = 0x02CA.8
#bit  PB9   = 0x02CA.9


void main()
{
  int16 PosCount=0;
  int16 Dt=0,Dt_2=0,Dt_3=0;
  int16 inter=0,pwm1=0,pwm2=0,pwm3=0;
  int8 PcH=0,PcL=0;
  int8 cvH_1=0,cvL_1=0,cvH_2=0,cvL_2=0,cvH_3=0,cvL_3=0;
  unsigned int8 flagcom=0,dat1=0,dat2=0,dat3=0;
  unsigned int8 dato=0,pwm=0,sign1=0,sign2=0,sign3=0;
  float pwmf=0;

  set_tris_a(0b111111);

  TRISA = 0x001F;
  TRISB = 0x008F;

  setup_adc_ports(ALL_ANALOG,VREF_VREF);
  setup_adc(ADC_CLOCK_INTERNAL);
  set_adc_channel(2);

setup_motor_pwm(1,MPWM_FREE_RUN,20000);

set_motor_unit(1,1,MPWM_ENABLE_L | MPWM_INDEPENDENT,0,0);
set_motor_unit(1,2,MPWM_ENABLE_L | MPWM_INDEPENDENT,0,0);
set_motor_unit(1,3,MPWM_ENABLE_L | MPWM_INDEPENDENT,0,0);

set_motor_pwm_duty(1,1,10);
set_motor_pwm_duty(1,2,10);
set_motor_pwm_duty(1,3,10);

  output_bit( PIN_B9, 1);

  setup_qei(QEI_MODE_X4 | QEI_RESET_WHEN_IDX_PULSE
            ,QEI_FILTER_DIV_1 | QEI_IDX_WHEN_A1_B1);

  setup_timer1(TMR_INTERNAL | TMR_DIV_BY_8 15002);//94  469---5ms
  set_timer1(0);
  qei_set_count(0);

  while(TRUE){
```

```
set_adc_channel(2);
delay_us(15);
Dt = read_adc();

set_adc_channel(3);
delay_us(15);
Dt_2=read_adc();

set_adc_channel(4);
delay_us(15);
Dt_3=read_adc();

inter=(Dt)&(0xFF00);
cvH_1=inter>>8;
cvL_1=(Dt)&(0x00FF);

inter=(Dt_2)&(0xFF00);
cvH_2=inter>>8;
cvL_2=(Dt_2)&(0x00FF);

inter=(Dt_3)&(0xFF00);
cvH_3=inter>>8;
cvL_3=(Dt_3)&(0x00FF);

PosCount = qei_get_count();

inter=(PosCount)&(0xFF00);
PcH=inter>>8;
PcL=(PosCount)&(0x00FF);

putc(0xAA);          //mandando dato de reconocimieno al puerto serial

putc(cvH_1);

putc(cvL_1);

putc(cvH_2);

putc(cvL_2);

putc(cvH_3);

putc(cvL_3);

putc(PcH);
```

```
 putc(PcL);

  pwm1=0;
  pwm2=0;
  pwm3=0;

do{
    if(kbhit())
    {
         dato = getc();//obteniendo el dato

         if(flagcom==0){
            dat1=dato;
            if(dat1>=128)
            {
               sign1=1;
               dat1 = dat1-128;
               output_low( PIN_B4);
            }
            else{
               sign1=0;
               output_high( PIN_B4);
            }
            flagcom = 1;
         }

         else if(flagcom==1){
            dat2=dato;
            if(dat2>=128)
            {
               sign2=1;
               dat2 = dat2-128;
               output_low(PIN_B5);
            }
            else{
               sign2=0;
               output_high(PIN_B5);
            }

            flagcom = 2;
         }
         else if(flagcom==2){
            dat3=dato;
             if(dat3>=128)
```

```
            {
              sign3 = 1;
              dat3 = dat3-128;
              output_low(PIN_B8);
            }
            else{
              sign3 = 0;
              output_high(PIN_B8);
            }
            flagcom = 0;

         }
         //flagcom++;
      }

  }while(get_timer1() < 15000);
  set_timer1(0);



  pwm=dat1<<1;
  pwmf=(float)pwm;
  pwm1=(int16)((pwmf*9.763779528)+10.0);
  pwm=0;

  pwm=dat2<<1;
  pwmf=(float)pwm;
  pwm2=(int16)((pwmf*9.763779528)+10.0);
  pwm=0;

  pwm=dat3<<1;
  pwmf=(float)pwm;
  pwm3=(int16)((pwmf*9.763779528)+10.0);//2480.0/254.0=9.763779528);
  pwm=0;


  //output_bit( PIN_B5, sign1);
  set_motor_pwm_duty(1,1,pwm1);
  //output_bit( PIN_B4, sign2);
  set_motor_pwm_duty(1,2,pwm2);
  //output_bit( PIN_B8, sign3);
  set_motor_pwm_duty(1,3,pwm3);


}
```

```
}
```

A.2. Programa para la implementación del controlador PI maestro e interfaz serial en Dev-C++

```
#include <iostream>
#include <string.h>
#include <dos.h>
#include <windows.h>
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <stdlib.h>

#define Ts      0.005
#define PI_     3.1415926535898
#define VM      46.0
#define VM2     9.0
#define IM      5.0
#define ppr     2500

#define kpq      1.0
#define kiq      10.0

#define kpd      1.0
#define kid      10.0

#define kpv      0.0477
#define kiv      2.38

#define aflt      5.0
#define bflt      5.0

int flagcom=0,flagfile=0,VecSector=0,j=0,i=0,BandF=0;
int cvADC_1=0,cvADC_2=0,cvADC_3=0;
unsigned short int PosCount=0; //de 16 bits
unsigned int inter=0,inter2=0,inter3=0;
float v_1=0.0,v_2=0.0,v_3=0.0,t=0.0,t_1=0.0,iTs=1/Ts,wd=0.3142;//wd[=]rad/s
unsigned char pwmH_1=0,pwmL_1=0,signo=0,signo2=0,signo3=0;
unsigned char pwmH_2=0,pwmL_2=0,pwmH_3=0,pwmL_3=0;
unsigned char pwm1=0,pwm2=0,pwm3=0;
float cvf_1,cvf_2,cvf_3,escs=127.0/VM,pwmf1,pwmf2,pwmf3;
float i1=0,i2=0,i3=0;
```

```
float theta=0.0,theta_1=0.0,thetaE=0.0,thetaF=0.0,thetaF_1=0.0;
float wr=0.0,ev=0.0,ev_1=0.0;
float z=0.0,z_1=0.0,vth=0.0,vth_1=0.0;

////////////////////////// Variables para FOC /////////////////
float iqr=0.0,idr=0.0,io=0.0;
float id=0.0,iq=0.0,ia=0.0,ib=0.0;
float inteq=0,propq=0;
float inted=0,propd=0;
float intev=0,propv=0;
float efq=0.0,efd=0.0;
float Vq=0.0,Vd=0.0;
float V1=0.0,V2=0.0,V3=0.0;
float Va=0.0,Vb=0.0,Vc=0.0,aux=0.0,Voff,taux=0.0;

int main()
{
    HANDLE h; /*handler, sera el descriptor del puerto*/
    DCB dcb; /*estructura de configuracion*/
    DWORD dwEventMask; /*mascara de eventos*/
    FILE *fp;

    if((fp=fopen("PruebaPI.txt","w+"))==NULL)
{
  printf("No se puede abrir el archivo.\n");
  exit(1);
}

////////////////////////////////////////////////////////////////////////////
    /*abrimos el puerto*/
    h=CreateFile("COM4",GENERIC_READ|GENERIC_WRITE,0,NULL,OPEN_EXISTING,0,NULL);

    if(h == INVALID_HANDLE_VALUE)
{
         /*ocurrio un error al intentar abrir el puerto*/
    }

/*obtenemos la configuracion actual*/
if(!GetCommState(h, &dcb))
{

         /*error: no se puede obtener la configuracion*/
    }

    /*Configuramos el puerto*/
```

```
dcb.BaudRate = 460800;
    dcb.ByteSize = 8;
    dcb.Parity = NOPARITY;//NOPARITY;
    dcb.StopBits = ONESTOPBIT;
    dcb.fBinary = TRUE;
    dcb.fParity = TRUE;

    /* Establecemos la nueva configuracion */
    if(!SetCommState(h, &dcb))
{
        /* Error al configurar el puerto */
    }

    DWORD n;
    unsigned char enviar;
    int recibido;

    /* Para que WaitCommEvent espere el evento RXCHAR */
    //SetCommMask(h, EV_RXCHAR);
    while(1)
{
     recibido=0;
        /* De la llamada a WaitCommEvent solo se retorna cuando ocurra 51.
 El evento seteado con SetCommMask */
        //WaitCommEvent(h, &dwEventMask, NULL);
        /* Recibimos algun dato!*/
        while(1)
{
//WaitCommEvent(h, &dwEventMask, NULL);
         ReadFile(h, &recibido, 1/* leemos un byte */, &n, NULL);
            if(!n)
             break;
            else
{
             if(flagcom!=0)
flagcom++;

if((recibido==0xAA)&&(flagcom==0))
    {
         cvADC_1=0;
         cvADC_2=0;
         cvADC_3=0;
         flagcom=1;
    }
```

```
    else if(flagcom==2)
    {
         cvADC_1=recibido;
         cvADC_1=cvADC_1<<8;
    }
    else if(flagcom==3)
    {
                  cvADC_1=cvADC_1+recibido;
    }

    else if(flagcom==4)
    {
         cvADC_2=recibido;
         cvADC_2=cvADC_2<<8;
    }
    else if(flagcom==5)
    {
                cvADC_2=cvADC_2+recibido;
    }

else if(flagcom==6)
    {
         cvADC_3=recibido;
         cvADC_3=cvADC_3<<8;
    }
    else if(flagcom==7)
    {
         cvADC_3=cvADC_3+recibido;
    }
    else if(flagcom==8)
    {
         PosCount=recibido;
         PosCount=PosCount<<8;
    }
else if(flagcom==9)
    {
         PosCount=PosCount+recibido;

theta=((signed short int)PosCount/(4.0*ppr))*2.0*PI_; //Posición angular mecánica

thetaE=theta*4.0; // Posición angular eléctrica (4 polos)

         if(fabs(theta-theta_1)<0.3){ // Para evitar el cambio brusco de velocidad
wr=(theta-theta_1)*iTs;  // Velocidad con la derivada numérica
```

```
////// Filtro de posición //////
z=(z_1-aflt*bflt*Ts*theta)/(1+aflt*Ts);
     vth=z+bflt*theta; // Vartheta: Estimación de velocidad
     ////////////////////////////////
}
else{
z=vth;
}

v_1=((float)cvADC_1)*(1.2/4096.0)+1.9-0.05273;
            v_2=((float)cvADC_2)*(1.2/4096.0)+1.9-0.05419;
            v_3=((float)cvADC_3)*(1.2/4096.0)+1.9-0.07060;

i1 =(5.0*v_1)-12.5;
            i2 =(5.0*v_2)-12.5;
            i3 =(5.0*v_3)-12.5;

          ///////////////Transformada de Park/////////////////
iq=0.816496*(i3*cos(thetaE)+i2*cos(thetaE-((2.0*PI_)/3.0))+i1*cos(thetaE+((2.0*PI_)/3.0)));
id=0.816496*(i3*sin(thetaE)+i2*sin(thetaE-((2.0*PI_)/3.0))+i1*sin(thetaE+((2.0*PI_)/3.0)));
////////////////////////////////////////////////

//ev=wd-wr; //error de velocidad
ev=wd-vth;

//////  PI maestro de velocidad  /////////
propv=kpv*ev;
         if((intev<IM)&&(intev>(-IM)))
                 intev=intev+kiv*Ts*ev;
         else
            {
                 if(intev>=IM)
                         intev=0.95*IM;
                 if(intev<=(-IM))
                         intev=-0.95*IM;
            }
         iqr=propv+intev;

//////// Saturación de Iqr ////////////////

if(iqr>=IM){
         iqr=0.95*IM;
}
                    else if(iqr<=(-IM)){
iqr=-0.95*IM;
```

```
}
////////////////////////////////////////////////////

////////// Errores para los lazos esclavos de corriente //////////
efq=iqr-iq;
     efd=idr-id;
     ////////////////////////////////////////////////////////////////

//PI de Iq:
         propq=kpq*efq;
         if((inteq<VM2)&&(inteq>(-VM2)))
                 inteq=inteq+kiq*Ts*efq;
         else
            {
                 if(inteq>=VM2)
                         inteq=0.95*VM2;
                 if(inteq<=(-VM2))
                         inteq=-0.95*VM2;
            }
         Vq=propq+inteq;

         //PI de Id:
         propd=kpd*efd;
         if((inted<VM2)&&(inted>(-VM2)))
                 inted=inted+kid*Ts*efd;
         else
            {
                 if(inted>=VM2)
                         inted=0.95*VM2;
                 if(inted<=(-VM2))
                         inted=-0.95*VM2;
            }
         Vd=propd+inted;

         ////////////////////////////Saturaciónes de los PI's////////////////////////
         /*if(Vq>=VM2){
         Vq=0.95*VM2;
}
                    else if(Vq<=(-VM2)){
Vq=-0.95*VM2;
}

if(Vd>=VM2){
         Vd=0.95*VM2;
}
```

```
                    else if(Vd<=(-VM2)){
Vd=-0.95*VM2;
}*/

     ////////////////Transformada Inversa de Park/////////////
     V1=0.816496*(Vq*cos(thetaE)+Vd*sin(thetaE));
     V2=0.816496*(Vq*cos(thetaE-((2.0*PI_)/3.0))+Vd*sin(thetaE-((2.0*PI_)/3.0)));
V3=0.816496*(Vq*cos(thetaE+((2.0*PI_)/3.0))+Vd*sin(thetaE+((2.0*PI_)/3.0)));
     ////////////////////////////////////////////////////////////////////////

cvf_1=V3;
         cvf_2=V2;
         cvf_3=V1;

                    if(cvf_1>=VM2){
         cvf_1=0.95*VM2;
}
                    else if(cvf_1<=(-VM2)){
cvf_1=-0.95*VM2;
}

if(cvf_2>=VM2){
         cvf_2=0.95*VM2;
}
                    else if(cvf_2<=(-VM2)){
cvf_2=-0.95*VM2;
}

if(cvf_3>=VM2){
         cvf_3=0.95*VM2;
}
                    else if(cvf_3<=(-VM2)){
cvf_3=-0.95*VM2;
}

//////// Voltajes forzados para el lazo abierto ///////
/*cvf_3=3*sin(2*PI_*3.0*t);
         cvf_2=3*sin(2*PI_*3.0*t-(2*PI_)/3);
         cvf_1=3*sin(2*PI_*3.0*t+(2*PI_)/3);
*/

/*
         cvf_1=0.0;
         cvf_2=0.0;
         cvf_3=0.0;
```

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

pwmf1=cvf_1;
        pwmf2=cvf_2;
        pwmf3=cvf_3;

        pwmf1=escs*pwmf1;
        pwmf2=escs*pwmf2;
        pwmf3=escs*pwmf3;

        pwm1=(unsigned char)fabs(pwmf1);
        pwm2=(unsigned char)fabs(pwmf2);
        pwm3=(unsigned char)fabs(pwmf3);

        if(pwmf1<0.0)
                pwm1=pwm1+128;
                 if(pwmf2<0.0)
                pwm2=pwm2+128;
             if(pwmf3<0.0)
                pwm3=pwm3+128;

                   enviar=pwm1;

                   if(!WriteFile(h, &enviar, 1, &n, NULL)){
                                                printf("Error al enviar...");
                   }

                  enviar=pwm2;
                   if(!WriteFile(h, &enviar, 1, &n, NULL)){
                                                printf("Error al enviar...");
                   }

                   enviar=pwm3;
                   if(!WriteFile(h, &enviar, 1, &n, NULL)){
                                                printf("Error al enviar...");
                   }

                   /*escribir algunos datos en el archivo de texto y en pantalla*/
     printf("\r%3.3f\t%3.3f\t%3.3f\t%3.3f\t%3.3f\t%3.3f\t%3.3f\t%3.3f",t,iqr,theta,wr,i1,i2,i3);
        fprintf(fp,"%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,iqr,iq,id,ev,vth,theta,wr);

t=t+Ts;
theta_1=theta;
```

```
thetaF_1=thetaF;
ev_1=ev;
t_1=t;
z_1=z;
vth_1=vth;
flagcom=0;
    }
    }//cierre del else
     }//CIERRE WHILE()
    }//CIERRE WHILE()
fclose(fp);
return 0;
}
```

A.3. Programa para la implementación de la RNA como controlador maestro e interfaz serial en Dev-C++

```
#include <iostream>
#include <string.h>
#include <dos.h>
#include <windows.h>
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <stdlib.h>

#define Ts      0.005
#define PI_     3.1415926535898
#define VM      46.0
#define VM2     9.0
#define IM      5.0
#define ppr     2500

#define N     3
#define Nin   3
#define eta   4.9

#define kpq      1.0
#define kiq      10.0

#define kpd      1.0
#define kid      10.0

#define aflt      5.0
```

```
#define bflt      5.0

int flagcom=0,flagfile=0,VecSector=0,j=0,i=0,BandF=0;
int cvADC_1=0,cvADC_2=0,cvADC_3=0;
unsigned short int PosCount=0; //de 16 bits
unsigned int inter=0,inter2=0,inter3=0;
float v_1=0.0,v_2=0.0,v_3=0.0,t=0.0,t_1=0.0,iTs=1/Ts,wd=0.3142;//wd[=]rad/s
unsigned char pwmH_1=0,pwmL_1=0,signo=0,signo2=0,signo3=0;
unsigned char pwmH_2=0,pwmL_2=0,pwmH_3=0,pwmL_3=0;
unsigned char pwm1=0,pwm2=0,pwm3=0;
float cvf_1,cvf_2,cvf_3,escs=127.0/VM,pwmf1,pwmf2,pwmf3;
float i1=0,i2=0,i3=0;
float theta=0.0,theta_1=0.0,thetaE=0.0,thetaF=0.0,thetaF_1=0.0;
float wr=0.0,ev=0.0,ev_1=0.0;
float z=0.0,z_1=0.0,vth=0.0,vth_1=0.0;

//////////////Variables para FOC//////////////
float iqr=0.0,idr=0.0,io=0.0;
float id=0.0,iq=0.0,ia=0.0,ib=0.0;
float inteq=0,propq=0;
float inted=0,propd=0;
float intev=0,propv=0;
float efq=0.0,efd=0.0;
float Vq=0.0,Vd=0.0;
float V1=0.0,V2=0.0,V3=0.0;
float Va=0.0,Vb=0.0,Vc=0.0,aux=0.0,Voff,taux=0.0;

////////////////////// Variables de la RNA ///////////////////
float SumaN[N],SumaS,x[Nin],w[N][Nin],ip[N],v[N];
float o=0.0,delta_1=0.0,delta_2[N];
float y_aux=0.0,y_aux2=0.0,err=0.0;

int main()
{
    HANDLE h; /*handler, sera el descriptor del puerto*/
    DCB dcb; /*estructura de configuracion*/
    DWORD dwEventMask; /*mascara de eventos*/
    FILE *fp;

    if((fp=fopen("PruebaRNA.txt","w+"))==NULL)
{
  printf("No se puede abrir el archivo.\n");
  exit(1);
}
```

```
///////////////// Inicialización RNA///////////////
for(j=0;j<Nin;j++){
x[j]=0.0;
}

for(j=0;j<N;j++){
for(i=0;i<Nin;i++){
w[j][i]=0.0;
}
delta_2[j]=0.0;
ip[j]=0.0;
v[j]=0.0;
}
////////////////////////////////////////////////////////////////////////

/*abrimos el puerto*/
    h=CreateFile("COM4",GENERIC_READ|GENERIC_WRITE,0,NULL,OPEN_EXISTING,0,NULL);

    if(h == INVALID_HANDLE_VALUE)
{
         /*ocurrio un error al intentar abrir el puerto*/
    }

/*obtenemos la configuracion actual*/
if(!GetCommState(h, &dcb))
{

         /*error: no se puede obtener la configuracion*/
    }

    /*Configuramos el puerto*/
dcb.BaudRate = 460800;
    dcb.ByteSize = 8;
    dcb.Parity = NOPARITY;//NOPARITY;
    dcb.StopBits = ONESTOPBIT;
    dcb.fBinary = TRUE;
    dcb.fParity = TRUE;

    /* Establecemos la nueva configuracion */
    if(!SetCommState(h, &dcb))
{
        /* Error al configurar el puerto */
    }
```

```
    DWORD n;
    unsigned char enviar;
    int recibido;

    /* Para que WaitCommEvent espere el evento RXCHAR */
    //SetCommMask(h, EV_RXCHAR);
    while(1)
{
     recibido=0;
        /* De la llamada a WaitCommEvent solo se retorna cuando ocurra51.
   El evento seteado con SetCommMask */
        //WaitCommEvent(h, &dwEventMask, NULL);
        /* Recibimos algun dato!*/
        while(1)
{
//WaitCommEvent(h, &dwEventMask, NULL);
         ReadFile(h, &recibido, 1/* leemos un byte */, &n, NULL);
           if(!n)
            break;
           else
{
            if(flagcom!=0)
flagcom++;

if((recibido==0xAA)&&(flagcom==0))
    {
         cvADC_1=0;
         cvADC_2=0;
         cvADC_3=0;
         flagcom=1;
    }

    else if(flagcom==2)
    {
         cvADC_1=recibido;
         cvADC_1=cvADC_1<<8;
    }
    else if(flagcom==3)
    {
                 cvADC_1=cvADC_1+recibido;
    }

    else if(flagcom==4)
    {
         cvADC_2=recibido;
```

```
          cvADC_2=cvADC_2<<8;
     }
     else if(flagcom==5)
     {
                    cvADC_2=cvADC_2+recibido;
     }

else if(flagcom==6)
     {
          cvADC_3=recibido;
          cvADC_3=cvADC_3<<8;
     }
     else if(flagcom==7)
     {
          cvADC_3=cvADC_3+recibido;
     }
     else if(flagcom==8)
     {
          PosCount=recibido;
          PosCount=PosCount<<8;
     }
else if(flagcom==9)
     {
          PosCount=PosCount+recibido;

theta=((signed short int)PosCount/(4.0*ppr))*2.0*PI_;//Posición angular mecánica
thetaE=theta*4.0; // Posición angular eléctrica

          if(fabs(theta-theta_1)<0.3){ // Para evitar un cambio brusco de velocidad
wr=(theta-theta_1)*iTs;  // Velocidad angular con la derivada numérica
////////////// Filtro de posición //////////////
z=(z_1-aflt*bflt*Ts*theta)/(1+aflt*Ts);
      vth=z+bflt*theta; // Vartheta: estimación de velocidad
      ////////////////////////////////////////////
}
else{
z=vth;
}

            v_1=((float)cvADC_1)*(1.2/4096.0)+1.9-0.05273;
            v_2=((float)cvADC_2)*(1.2/4096.0)+1.9-0.05419;
            v_3=((float)cvADC_3)*(1.2/4096.0)+1.9-0.07060;

            i1 =(5.0*v_1)-12.5;
            i2 =(5.0*v_2)-12.5;
```

```
            i3 =(5.0*v_3)-12.5;

            ///////////////Transformada de Park/////////////////
iq=0.816496*(i3*cos(thetaE)+i2*cos(thetaE-((2.0*PI_)/3.0))+i1*cos(thetaE+((2.0*PI_)/3.0)));
id=0.816496*(i3*sin(thetaE)+i2*sin(thetaE-((2.0*PI_)/3.0))+i1*sin(thetaE+((2.0*PI_)/3.0)));
////////////////////////////////////////////////

//ev=wd-wr;//error de velocidad
ev=wd-vth;

//////////////////// RNA ///////////////
for(j=0;j<N;j++){
        SumaN[j] = 0.0;
        }
    SumaS = 0.0;

if(ev>0.95){
ev=0.95;
}
else if(ev<-0.95){
ev=-0.95;
}

ev=0.4211*ev+0.5;

x[2]=x[1];
    x[1]=x[0];
    x[0]=ev;

    for(j=0;j<N;j++){
    for(i=0;i<Nin;i++){
    SumaN[j]=SumaN[j]+w[j][i]*x[i];
        }
           ip[j]=1.0/(1.0+exp(-1.0*SumaN[j]));
           SumaS=SumaS+ip[j]*v[j];
    }

    y_aux=wd*0.01333+0.5;
    y_aux2=wr*0.01333+0.5;

    err=y_aux-y_aux2;

    o=1.0/(1.0+exp(-1.0*SumaS));

    iqr=5.0*o-2.5;
```

```
    delta_1=err*o*(1.0-o);

for( j=0;j<N;j++){
           delta_2[j]=delta_1*v[j]*ip[j]*(1.0-ip[j]);
      }

      for (j=0;j<N;j++){
          v[j]=v[j]+eta*delta_1*ip[j];
      }

      for (j=0;j<N;j++){
      for( i=0;i<Nin;i++){
             w[j][i]=w[j][i]+eta*delta_2[j]*x[i];
    }
      }

/////////////// Saturación Iqr /////////////////
         /*if(iqr>=IM){
         iqr=0.95*IM;
}
                   else if(iqr<=(-IM)){
iqr=-0.95*IM;
}*/
//////////////////////////////////////////////////

////// Errores para los lazos esclavos de corriente ///////
efq=iqr-iq;
     efd=idr-id;

     //PI de la fase Q:
         propq=kpq*efq;
         if((inteq<VM2)&&(inteq>(-VM2)))
                 inteq=inteq+kiq*Ts*efq;
         else
            {
                 if(inteq>=VM2)
                         inteq=0.95*VM2;
                 if(inteq<=(-VM2))
                         inteq=-0.95*VM2;
            }
         Vq=propq+inteq;

         //PI de la fase D:
         propd=kpd*efd;
```

```
         if((inted<VM2)&&(inted>(-VM2)))
                 inted=inted+kid*Ts*efd;
         else
            {
                 if(inted>=VM2)
                         inted=0.95*VM2;
                 if(inted<=(-VM2))
                         inted=-0.95*VM2;
            }
         Vd=propd+inted;

         ////////////Saturaciónes de los PI's////////////
         /*if(Vq>=VM2){
         Vq=0.95*VM2;
}
          else if(Vq<=(-VM2)){
Vq=-0.95*VM2;
}

if(Vd>=VM2){
         Vd=0.95*VM2;
}
          else if(Vd<=(-VM2)){
Vd=-0.95*VM2;
}*/

     //////////////////Transformada Inversa de Park/////////////
     V1=0.816496*(Vq*cos(thetaE)+Vd*sin(thetaE));
     V2=0.816496*(Vq*cos(thetaE-((2.0*PI_)/3.0))+Vd*sin(thetaE-((2.0*PI_)/3.0)));
V3=0.816496*(Vq*cos(thetaE+((2.0*PI_)/3.0))+Vd*sin(thetaE+((2.0*PI_)/3.0)));
     ////////////////////////////////////////////////////////////////////////

cvf_1=V3;
         cvf_2=V2;
         cvf_3=V1;

                    if(cvf_1>=VM2){
         cvf_1=0.95*VM2;
}
                    else if(cvf_1<=(-VM2)){
cvf_1=-0.95*VM2;
}

if(cvf_2>=VM2){
         cvf_2=0.95*VM2;
```

```
}
                    else if(cvf_2<=(-VM2)){
cvf_2=-0.95*VM2;
}

if(cvf_3>=VM2){
         cvf_3=0.95*VM2;
}
                    else if(cvf_3<=(-VM2)){
cvf_3=-0.95*VM2;
}

//////// Voltajes forzados para lazo abierto ///////////

/*cvf_3=3*sin(2*PI_*3.0*t);
         cvf_2=3*sin(2*PI_*3.0*t-(2*PI_)/3);
         cvf_1=3*sin(2*PI_*3.0*t+(2*PI_)/3);
*/
/*
         cvf_1=0.0;
         cvf_2=0.0;
         cvf_3=0.0;
         */

//////////////////////////////////////////////////////
pwmf1=cvf_1;
         pwmf2=cvf_2;
         pwmf3=cvf_3;

         pwmf1=escs*pwmf1;
         pwmf2=escs*pwmf2;
         pwmf3=escs*pwmf3;

         pwm1=(unsigned char)fabs(pwmf1);
         pwm2=(unsigned char)fabs(pwmf2);
         pwm3=(unsigned char)fabs(pwmf3);

         if(pwmf1<0.0)
                 pwm1=pwm1+128;
                  if(pwmf2<0.0)
                 pwm2=pwm2+128;
              if(pwmf3<0.0)
                 pwm3=pwm3+128;
```

```
                    enviar=pwm1;

                    if(!WriteFile(h, &enviar, 1, &n, NULL)){
                     printf("Error al enviar...");
                    }

                  enviar=pwm2;
                    if(!WriteFile(h, &enviar, 1, &n, NULL)){
                     printf("Error al enviar...");
                    }

                   enviar=pwm3;
                    if(!WriteFile(h, &enviar, 1, &n, NULL)){
                     printf("Error al enviar...");
                    }

                    /*escribir algunos datos en el archivo de texto y en pantalla*/
printf("\r%3.3f\t%3.3f\t%3.3f\t%3.3f\t%3.3f\t%3.3f\t%3.3f",t,iqr,theta,wr,i1,i2,i3);
         fprintf(fp,"%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,iqr,iq,id,ev,vth,theta,wr);

t=t+Ts;
theta_1=theta;
thetaF_1=thetaF;
ev_1=ev;
t_1=t;
z_1=z;
vth_1=vth;
flagcom=0;
    }
    }//cierre del else
     }//CIERRE WHILE()
    }//CIERRE WHILE()
fclose(fp);
return 0;
}
```

A.4. Programa para la implementación del FSLC como controlador maestro e interfaz serial en Dev-C++

```
#include <iostream>
#include <string.h>
#include <dos.h>
#include <windows.h>
```

```
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <stdlib.h>

#define Ts      0.005
#define PI_     3.1415926535898
#define VM      46.0
#define VM2     9.0
#define IM      5.0
#define ppr     2500

#define kpq      1.0
#define kiq      10.0

#define kpd      1.0
#define kid      10.0

#define aflt      5.0
#define bflt      5.0

int flagcom=0,flagfile=0,VecSector=0,j=0,i=0,BandF=0;
int cvADC_1=0,cvADC_2=0,cvADC_3=0;
unsigned short int PosCount=0; //de 16 bits
unsigned int inter=0,inter2=0,inter3=0;
float v_1=0.0,v_2=0.0,v_3=0.0,t=0.0,t_1=0.0,iTs=1/Ts,wd=0.3142;//wd[=]rad/s
unsigned char pwmH_1=0,pwmL_1=0,signo=0,signo2=0,signo3=0;
unsigned char pwmH_2=0,pwmL_2=0,pwmH_3=0,pwmL_3=0;
unsigned char pwm1=0,pwm2=0,pwm3=0;
float cvf_1,cvf_2,cvf_3,escs=127.0/VM,pwmf1,pwmf2,pwmf3;
float i1=0,i2=0,i3=0;
float theta=0.0,theta_1=0.0,thetaE=0.0,thetaF=0.0,thetaF_1=0.0;
float wr=0.0,ev=0.0,ev_1=0.0;
float z=0.0,z_1=0.0,vth=0.0,vth_1=0.0;

////////////// Variables para FOC //////////////
float iqr=0.0,idr=0.0,io=0.0;
float id=0.0,iq=0.0,ia=0.0,ib=0.0;
float inteq=0,propq=0;
float inted=0,propd=0;
float intev=0,propv=0;
float efq=0.0,efd=0.0;
float Vq=0.0,Vd=0.0;
float V1=0.0,V2=0.0,V3=0.0;
float Va=0.0,Vb=0.0,Vc=0.0,aux=0.0,Voff,taux=0.0;
```

```
////////////// Variables para la FSNN /////////
#define Nt     4
float gamma[Nt],alpha[Nt];
float sum = 0.0,a[Nt],b[Nt],s[Nt],zf[Nt],yf[Nt],p[Nt],q[Nt],ag[Nt],bg[Nt];

int main()
{
    HANDLE h; /*handler, sera el descriptor del puerto*/
    DCB dcb; /*estructura de configuracion*/
    DWORD dwEventMask; /*mascara de eventos*/
    FILE *fp;

    if((fp=fopen("PruebaFSLC.txt","w+"))==NULL)
{
  printf("No se puede abrir el archivo.\n");
  exit(1);
}

///////////////// Inicialización del FSLC //////////////
        for(i=0;i<Nt;i++){
     a[i] = 0.0;
     b[i] = 0.0;

     s[i] = 0.0;
     zf[i] = 0.0;
     yf[i] = 0.0;

     p[i]=0.0;
        q[i]=0.0;

     ag[i]=0.0;
        bg[i]=0.0;

        gamma[i] = 0.03;//0.03 /// Ganancias de aprendizaje
        alpha[i] = 0.037;//0.037 /// Ganancias de proporción
}

//////////////////////////////////////////////////////////////////////////
    /*abrimos el puerto*/
    h=CreateFile("COM4",GENERIC_READ|GENERIC_WRITE,0,NULL,OPEN_EXISTING,0,NULL);

    if(h == INVALID_HANDLE_VALUE)
{
         /*ocurrio un error al intentar abrir el puerto*/
```

```
    }

/*obtenemos la configuracion actual*/
if(!GetCommState(h, &dcb))
{

         /*error: no se puede obtener la configuracion*/
    }

    /*Configuramos el puerto*/
dcb.BaudRate = 460800;
    dcb.ByteSize = 8;
    dcb.Parity = NOPARITY;//NOPARITY;
    dcb.StopBits = ONESTOPBIT;
    dcb.fBinary = TRUE;
    dcb.fParity = TRUE;

    /* Establecemos la nueva configuracion */
    if(!SetCommState(h, &dcb))
{
        /* Error al configurar el puerto */
    }

    DWORD n;
    unsigned char enviar;
    int recibido;

    /* Para que WaitCommEvent espere el evento RXCHAR */
    //SetCommMask(h, EV_RXCHAR);
    while(1)
{
     recibido=0;
        /* De la llamada a WaitCommEvent solo se retorna cuando ocurra 51.
   El evento seteado con SetCommMask */
        //WaitCommEvent(h, &dwEventMask, NULL);
        /* Recibimos algun dato!*/
        while(1)
{
//WaitCommEvent(h, &dwEventMask, NULL);
         ReadFile(h, &recibido, 1/* leemos un byte */, &n, NULL);
            if(!n)
             break;
            else
{
             if(flagcom!=0)
```

```
flagcom++;

if((recibido==0xAA)&&(flagcom==0))
    {
         cvADC_1=0;
         cvADC_2=0;
         cvADC_3=0;
         flagcom=1;
    }

    else if(flagcom==2)
    {
         cvADC_1=recibido;
         cvADC_1=cvADC_1<<8;
    }
    else if(flagcom==3)
    {
                  cvADC_1=cvADC_1+recibido;
    }

    else if(flagcom==4)
    {
         cvADC_2=recibido;
         cvADC_2=cvADC_2<<8;
    }
    else if(flagcom==5)
    {
                 cvADC_2=cvADC_2+recibido;
    }

else if(flagcom==6)
    {
         cvADC_3=recibido;
         cvADC_3=cvADC_3<<8;
    }
    else if(flagcom==7)
    {
         cvADC_3=cvADC_3+recibido;
    }
    else if(flagcom==8)
    {
         PosCount=recibido;
         PosCount=PosCount<<8;
    }
else if(flagcom==9)
```

```
    {
         PosCount=PosCount+recibido;

theta=((signed short int)PosCount/(4.0*ppr))*2.0*PI_;//Posición angular mecánica
thetaE=theta*4.0; //Posición angular eléctrica (4 polos)

         if(fabs(theta-theta_1)<0.3){ // Para evitar un cambio brusco de velocidad
wr=(theta-theta_1)*iTs;  // Velocidad con la derivada numérica

////////////// Filtro de Posición ///////////////
z=(z_1-aflt*bflt*Ts*theta)/(1+aflt*Ts);
     vth=z+bflt*theta; // Vartheta: estimación de velocidad
     ////////////////////////////////////////////////
}
else{
z=vth;
}

            v_1=((float)cvADC_1)*(1.2/4096.0)+1.9-0.05273;
            v_2=((float)cvADC_2)*(1.2/4096.0)+1.9-0.05419;
            v_3=((float)cvADC_3)*(1.2/4096.0)+1.9-0.07060;

            i1 =(5.0*v_1)-12.5;
            i2 =(5.0*v_2)-12.5;
            i3 =(5.0*v_3)-12.5;

            ///////////////Transformada de Park/////////////////
iq=0.816496*(i3*cos(thetaE)+i2*cos(thetaE-((2.0*PI_)/3.0))+i1*cos(thetaE+((2.0*PI_)/3.0)));
id=0.816496*(i3*sin(thetaE)+i2*sin(thetaE-((2.0*PI_)/3.0))+i1*sin(thetaE+((2.0*PI_)/3.0)));
////////////////////////////////////////////////

//ev=wd-wr;// error de velocidad
ev=wd-vth;

////////////////////// FSLC /////////////////
for(i=0;i<(Nt-1);i++){
s[i] = s[i+1];
}
s[Nt-1] = ev+(ev-ev_1)*iTs;

for(i=0;i<=(Nt/2);i++){
zf[i] = 0.0;
     yf[i] = 0.0;
}
```

```
for(i=0;i<=(Nt/2);i++){
ag[i] = ag[i] + gamma[i] * p[i];
bg[i] = bg[i] + gamma[i] * q[i];
for(j=0;j<Nt;j++){
zf[i] = zf[i] + s[j]*cos((2*PI_*i*j)/Nt);
yf[i] = yf[i] - s[j]*sin((2*PI_*i*j)/Nt);
}
p[i] = 2*zf[i]/Nt;
q[i] = -2*yf[i]/Nt;

p[0] = zf[0]/Nt;
q[0] = 0.0;

p[Nt/2] = zf[Nt/2]/Nt;
q[Nt/2] = 0.0;

a[i] = alpha[i]*p[i] + ag[i];
b[i] = alpha[i]*q[i] + bg[i];
}

sum=0.0;
for(i=0;i<=(Nt/2);i++){
sum = sum + (a[i]*cos((2*PI_*i*(Nt-1))/Nt) + b[i]*sin((2*PI_*i*(Nt-1))/Nt));
}

iqr = sum;
//////////////////////////////Saturación Isqr//////////////////
         if(iqr>=IM){
         iqr=0.95*IM;
}
                    else if(iqr<=(-IM)){
iqr=-0.95*IM;
}
////////////////////////////////////////////////////
//////// Errores para los lazos esclavos de corriente //////////
efq=iqr-iq;
     efd=idr-id;
     ////////////////////////////////////////////////////////////////

     //PI para la fase Q:
         propq=kpq*efq;
         if((inteq<VM2)&&(inteq>(-VM2)))
                 inteq=inteq+kiq*Ts*efq;
         else
            {
```

```
                if(inteq>=VM2)
                        inteq=0.95*VM2;
                if(inteq<=(-VM2))
                        inteq=-0.95*VM2;
           }
         Vq=propq+inteq;

         //PI para la fase D:
         propd=kpd*efd;
         if((inted<VM2)&&(inted>(-VM2)))
                inted=inted+kid*Ts*efd;
         else
           {
                if(inted>=VM2)
                        inted=0.95*VM2;
                if(inted<=(-VM2))
                        inted=-0.95*VM2;
           }
         Vd=propd+inted;

////////////// Saturaciónes de los PI's /////////////
         /*if(Vq>=VM2){
         Vq=0.95*VM2;
}
                    else if(Vq<=(-VM2)){
Vq=-0.95*VM2;
}

if(Vd>=VM2){
         Vd=0.95*VM2;
}
                    else if(Vd<=(-VM2)){
Vd=-0.95*VM2;
}*/

     /////////////////Transformada Inversa de Park/////////////
     V1=0.816496*(Vq*cos(thetaE)+Vd*sin(thetaE));
     V2=0.816496*(Vq*cos(thetaE-((2.0*PI_)/3.0))+Vd*sin(thetaE-((2.0*PI_)/3.0)));
V3=0.816496*(Vq*cos(thetaE+((2.0*PI_)/3.0))+Vd*sin(thetaE+((2.0*PI_)/3.0)));
     ////////////////////////////////////////////////////////////////////////

cvf_1=V3;
         cvf_2=V2;
         cvf_3=V1;
```

```
                    if(cvf_1>=VM2){
          cvf_1=0.95*VM2;
}
                    else if(cvf_1<=(-VM2)){
cvf_1=-0.95*VM2;
}

if(cvf_2>=VM2){
          cvf_2=0.95*VM2;
}
                    else if(cvf_2<=(-VM2)){
cvf_2=-0.95*VM2;
}

if(cvf_3>=VM2){
          cvf_3=0.95*VM2;
}
                    else if(cvf_3<=(-VM2)){
cvf_3=-0.95*VM2;
}

///////// Voltajes forzados para lazo abierto ///////////
/*cvf_3=3*sin(2*PI_*3.0*t);
          cvf_2=3*sin(2*PI_*3.0*t-(2*PI_)/3);
          cvf_1=3*sin(2*PI_*3.0*t+(2*PI_)/3);
*/
/*
          cvf_1=0.0;
          cvf_2=0.0;
          cvf_3=0.0;
          */

pwmf1=cvf_1;
          pwmf2=cvf_2;
          pwmf3=cvf_3;

          pwmf1=escs*pwmf1;
          pwmf2=escs*pwmf2;
          pwmf3=escs*pwmf3;

          pwm1=(unsigned char)fabs(pwmf1);
          pwm2=(unsigned char)fabs(pwmf2);
          pwm3=(unsigned char)fabs(pwmf3);

          if(pwmf1<0.0)
```

```
                pwm1=pwm1+128;
                 if(pwmf2<0.0)
                pwm2=pwm2+128;
             if(pwmf3<0.0)
                pwm3=pwm3+128;


                   enviar=pwm1;

                   if(!WriteFile(h, &enviar, 1, &n, NULL)){
                    printf("Error al enviar...");
                   }

                  enviar=pwm2;
                   if(!WriteFile(h, &enviar, 1, &n, NULL)){
           printf("Error al enviar...");
                   }

                  enviar=pwm3;
                   if(!WriteFile(h, &enviar, 1, &n, NULL)){
        printf("Error al enviar...");
                   }

                   /*escribir algunos datos en el archivo de texto y en pantalla*/
printf("\r%3.3f\t%3.3f\t%3.3f\t%3.3f\t%3.3f\t%3.3f\t%3.3f",t,iqr,theta,wr,i1,i2,i3);
fprintf(fp,"%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,iqr,iq,id,ev,vth,theta,wr);

t=t+Ts;
theta_1=theta;
thetaF_1=thetaF;
ev_1=ev;
t_1=t;
z_1=z;
vth_1=vth;
flagcom=0;
    }
    }//cierre del else
     }//CIERRE WHILE()
    }//CIERRE WHILE()
fclose(fp);
return 0;
}
```

A.5. Programa en MATLAB para la obtención de las gráficas

```
%%%%%%%%%%%%%%%%%%% Graficas del control de velocidad de un PMSM %%%%%%%%%%
clc;
clear all;
close all;

%%%%%%%%% Leyendo las muestras tomadas del experimento %%%%%%%%%
fidFSLC = fopen('PruebaFSLC.TXT','r');
fidRNA  = fopen('PruebaRNA.TXT','r');
fidPI   = fopen('PruebaPI.TXT','r');

datosFSLC = fscanf(fidFSLC, '%f', [8 inf]);
datosRNA  = fscanf(fidRNA, '%f', [8 inf]);
datosPI   = fscanf(fidPI, '%f', [8 inf]);

fclose(fidFSLC);
fclose(fidRNA);
fclose(fidPI);

%%%%%%%%%%%%%% Asignando las columnas de los datos obtenidos %%%%%%%%%%%%%
tFSLC     = datosFSLC(1,:);
iqrFSLC   = datosFSLC(2,:);
iqFSLC    = datosFSLC(3,:);
idFSLC    = datosFSLC(4,:);
evFSLC    = datosFSLC(5,:);
vthFSLC   = datosFSLC(6,:);
thetaFSLC = datosFSLC(7,:);
wrFSLC    = datosFSLC(8,:);

tRNA     = datosRNA(1,:);
iqrRNA   = datosRNA(2,:);
iqRNA    = datosRNA(3,:);
idRNA    = datosRNA(4,:);
evRNA    = datosRNA(5,:)-0.5;
vthRNA   = datosRNA(6,:);
thetaRNA = datosRNA(7,:);
wrRNA    = datosRNA(8,:);

tPI     = datosPI(1,:);
iqrPI   = datosPI(2,:);
iqPI    = datosPI(3,:);
idPI    = datosPI(4,:);
evPI    = datosPI(5,:);
vthPI   = datosPI(6,:);
```

```
thetaPI = datosPI(7,:);
wrPI    = datosPI(8,:);


%%%%%%%%%%%%%%%%%%% Grafica del filtro %%%%%%%%%%%%%%%%%%%
%{
plot(tFSLC,wrFSLC,'-.',tFSLC,vthFSLC)
set(legend('$\dot{\theta}_N(k)[\frac{rad}{s}]$','$\vartheta[\frac{rad}{s}]$')
,'FontSize',14, 'interpreter','latex');
axis([0,100,-6.5,6.5])
grid on;
ylabel('Velocidad $\vartheta$[$\frac{rad}{s}$]','FontSize',16,'interpreter','latex')
xlabel('Tiempo $t$[$s$]','FontSize',16,'interpreter', 'latex')
%}
%%%%%%%%%%%%%%%%%%% Graficas de las posiciones Forma 1 %%%%%%%%%%%%%%%%%%%
%{
a=subplot(3,1,1);
plot(tFSLC,thetaFSLC);
set(legend('$\theta[rad]$'), 'interpreter','latex');
axis([0,100,-0.6,6.5])
grid on;
title(a,'\fontsize{12} {a)}')

b=subplot(3,1,2);
plot(tPI,thetaPI);
set(legend('$\theta[rad]$'), 'interpreter','latex');
axis([0,100,-0.6,6.5])
grid on;
ylabel('Position [$rad$]','FontSize',16,'interpreter', 'latex')
title(b,'\fontsize{12} b)')

c=subplot(3,1,3);
plot(tRNA,thetaRNA);
set(legend('$\theta[rad]$'), 'interpreter','latex');
axis([0,100,-0.6,6.5])
grid on;
title(c,'\fontsize{12} c)')

xlabel('Time [$s$]','FontSize',16,'interpreter', 'latex')
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Posiciones Forma 2 %%%%%%%%%%%%%%%%%%%%%%%%%%
%{
plot(tFSLC,thetaFSLC);
set(legend('$\theta[rad]$'), 'interpreter','latex');
axis([0,100,-1,7])
```

```
grid on;
ylabel('Posici\''on $\theta$[$rad$]','FontSize',16,'interpreter', 'latex')
xlabel('Tiempo $t$[$s$]','FontSize',16,'interpreter', 'latex')
%}
%{
plot(tPI,thetaPI);
set(legend('$\theta[rad]$'), 'interpreter','latex');
axis([0,100,-1,7])
grid on;
ylabel('Posici\''on $\theta$[$rad$]','FontSize',16,'interpreter', 'latex')
xlabel('Tiempo $t$[$s$]','FontSize',16,'interpreter', 'latex')
%}
%{
plot(tRNA,thetaRNA);
set(legend('$\theta[rad]$'), 'interpreter','latex');
axis([0,100,-1,7])
grid on;
ylabel('Posici\''on $\theta$[$rad$]','FontSize',16,'interpreter', 'latex')
xlabel('Tiempo $t$[$s$]','FontSize',16,'interpreter', 'latex')
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Velocidades Forma 1 %%%%%%%%%%%%%%%%%%%%%%%%%
%{
a=subplot(3,1,1);
plot(tFSLC,vthFSLC);
set(legend('$\vartheta[rad/s]$'), 'interpreter','latex');
axis([0,100,-1.7,3.1])
grid on;
title(a,'\fontsize{12} {a)}')

b=subplot(3,1,2);
plot(tPI,vthPI);
set(legend('$\vartheta[rad/s]$'), 'interpreter','latex');
axis([0,100,-1.7,3.1])
grid on;
ylabel('Velocity [$rad/s$]','FontSize',16,'interpreter', 'latex')
title(b,'\fontsize{12} b)')

c=subplot(3,1,3);
plot(tRNA,vthRNA);
set(legend('$\vartheta[rad/s]$'), 'interpreter','latex');
axis([0,100,-1.7,3.1])
grid on;
title(c,'\fontsize{12} c)')

xlabel('Time [$s$]','FontSize',16,'interpreter', 'latex')
```

```
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Velocidades Forma 2 %%%%%%%%%%%%%%%%%%%%%%%%%
%{
plot(tFSLC,vthFSLC);
set(legend('$\vartheta[\frac{rad}{s}]$'), 'interpreter','latex');
axis([0,100,-1.5,3.0])
grid on;
ylabel('Velocidad $\vartheta$[$\frac{rad}{s}$]','FontSize',16,'interpreter', 'latex')
xlabel('Tiempo $t$[$s$]','FontSize',16,'interpreter', 'latex')
%}
%{
plot(tPI,vthPI);
set(legend('$\vartheta[\frac{rad}{s}]$'), 'interpreter','latex');
axis([0,100,-1.5,3.0])
grid on;
ylabel('Velocidad $\vartheta$[$\frac{rad}{s}$]','FontSize',16,'interpreter', 'latex')
xlabel('Tiempo $t$[$s$]','FontSize',16,'interpreter', 'latex')
%}
%{
plot(tRNA,vthRNA);
set(legend('$\vartheta[\frac{rad}{s}]$'), 'interpreter','latex');
axis([0,100,-1.5,3.0])
grid on;
ylabel('Velocidad $\vartheta$[$\frac{rad}{s}$]','FontSize',16,'interpreter', 'latex')
xlabel('Tiempo $t$[$s$]','FontSize',16,'interpreter', 'latex')
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%% Corrientes Iq Forma 1  %%%%%%%%%%%%%%%%%%%%%%%%%
%{
a=subplot(3,1,1);
plot(tFSLC,iqrFSLC,tFSLC,iqFSLC);
set(legend('$I_{qRef}[A]$','$I_{q}[A]$'), 'interpreter','latex');
axis([0,100,-6,6])
grid on;
title(a,'\fontsize{12} {a)}')

b=subplot(3,1,2);
plot(tPI,iqrPI,tPI,iqPI);
set(legend('$I_{qRef}[A]$','$I_{q}[A]$'), 'interpreter','latex');
axis([0,100,-6,6])
grid on;
ylabel('$I_q$ Current [$A$]','FontSize',16,'interpreter', 'latex')
title(b,'\fontsize{12} b)')

c=subplot(3,1,3);
plot(tRNA,iqrRNA,tRNA,iqRNA);
```

```
set(legend('$I_{qRef}[A]$','$I_{q}[A]$'), 'interpreter','latex');
axis([0,100,-6,6])
grid on;
title(c,'\fontsize{12} c)')

xlabel('Time [$s$]','FontSize',16,'interpreter', 'latex')
%}
%%%%%%%%%%%%%%%%%%%%%%%%%% Corrientes Iq Forma 2  %%%%%%%%%%%%%%%%%%%%%%%%
%{
plot(tFSLC,iqrFSLC,'-.',tFSLC,iqFSLC);
set(legend('$I_{qd}[A]$','$I_{q}[A]$'), 'interpreter','latex');
axis([0,100,-6,6])
grid on;
ylabel('Corriente $I_q$[$A$]','FontSize',16,'interpreter', 'latex')
xlabel('Tiempo $t$[$s$]','FontSize',16,'interpreter', 'latex')
%}
%{
plot(tPI,iqrPI,'-.',tPI,iqPI);
set(legend('$I_{qd}[A]$','$I_{q}[A]$'), 'interpreter','latex');
axis([0,100,-6,6])
grid on;
ylabel('Corriente $I_q$[$A$]','FontSize',16,'interpreter', 'latex')
xlabel('Tiempo $t$[$s$]','FontSize',16,'interpreter', 'latex')
%}
%{
plot(tRNA,iqrRNA,'-.',tRNA,iqRNA);
set(legend('$I_{qd}[A]$','$I_{q}[A]$'), 'interpreter','latex');
axis([0,100,-6,6])
grid on;
ylabel('Corriente $I_q$[$A$]','FontSize',16,'interpreter', 'latex')
xlabel('Tiempo $t$[$s$]','FontSize',16,'interpreter', 'latex')
%}
%%%%%%%%%%%%%%%%%%%%%%%%%% Corrientes Id Forma 1  %%%%%%%%%%%%%%%%%%%%%%%%
%{
a=subplot(3,1,1);
plot(tFSLC,idFSLC);
set(legend('$I_{qRef}[A]$','$I_{q}[A]$'), 'interpreter','latex');
axis([0,100,-5.5,5.5])
grid on;
title(a,'\fontsize{12} {a)}')

b=subplot(3,1,2);
plot(tPI,idPI);
set(legend('$I_{qRef}[A]$','$I_{q}[A]$'), 'interpreter','latex');
axis([0,100,-5.5,5.5])
```

```
grid on;
ylabel('Corriente $I_q$ [$A$]','FontSize',16,'interpreter', 'latex')
title(b,'\fontsize{12} b)')

c=subplot(3,1,3);
plot(tRNA,idRNA);
set(legend('$I_{qRef}[A]$','$I_{q}[A]$'), 'interpreter','latex');
axis([0,100,-5.5,5.5])
grid on;
title(c,'\fontsize{12} c)')

xlabel('Tiempo [$s$]','FontSize',16,'interpreter', 'latex')
%}
%%%%%%%%%%%%%%%%%%%%%%%%%% Corrientes Id Forma 2  %%%%%%%%%%%%%%%%%%%%%%%%
%{
plot(tFSLC,idFSLC);
set(legend('$I_{d}[A]$'), 'interpreter','latex');
axis([0,100,-0.6,3.0]);
grid on;
ylabel('Corriente $I_d$[$A$]','FontSize',16,'interpreter', 'latex')
xlabel('Tiempo $t$[$s$]','FontSize',16,'interpreter', 'latex')
%}
%{
plot(tPI,idPI);
set(legend('$I_{d}[A]$'), 'interpreter','latex');
axis([0,100,-0.6,3.0])
grid on;
ylabel('Corriente $I_d$[$A$]','FontSize',16,'interpreter', 'latex')
xlabel('Tiempo $t$[$s$]','FontSize',16,'interpreter', 'latex')
%}
%{
plot(tRNA,idRNA);
set(legend('$I_{d}[A]$'), 'interpreter','latex');
axis([0,100,-0.6,3.0])
grid on;
ylabel('Corriente $I_d$[$A$]','FontSize',16,'interpreter', 'latex')
xlabel('Tiempo $t$[$s$]','FontSize',16,'interpreter', 'latex')
%}
%%%%%%%%%%%%%%%%%%%%%%%%%% Errores Forma 1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%{
a=subplot(3,1,1);
plot(tFSLC,evFSLC);
set(legend('$e[rad/s]$'), 'interpreter','latex');
axis([0,100,-2.7,1.8])
grid on;
```

```
title(a,'\fontsize{12} {a)}')

b=subplot(3,1,2);
plot(tPI,evPI);
set(legend('$e[rad/s]$'), 'interpreter','latex');
axis([0,100,-2.7,1.8])
grid on;
ylabel('Error [$rad/s$]','FontSize',16,'interpreter', 'latex')
title(b,'\fontsize{12} b)')

c=subplot(3,1,3);
plot(tRNA,evRNA);
set(legend('$e[rad/s]$'), 'interpreter','latex');
axis([0,100,-2.7,1.8])
grid on;
title(c,'\fontsize{12} c)')

xlabel('Time [$s$]','FontSize',16,'interpreter', 'latex')
%}
%%%%%%%%%%%%%%%%%%%%%%%%% Errores Forma 2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%{
plot(tFSLC,evFSLC);
set(legend('$e[\frac{rad}{s}]$'), 'interpreter','latex');
axis([0,100,-2.7,1.8])
grid on;
ylabel('Error $e$[$\frac{rad}{s}$]','FontSize',16,'interpreter', 'latex')
xlabel('Tiempo $t$[$s$]','FontSize',16,'interpreter', 'latex')
%}
%{
plot(tPI,evPI);
set(legend('$e[\frac{rad}{s}]$'), 'interpreter','latex');
axis([0,100,-2.7,1.8])
grid on;
ylabel('Error $e$[$\frac{rad}{s}$]','FontSize',16,'interpreter', 'latex')
xlabel('Tiempo $t$[$s$]','FontSize',16,'interpreter', 'latex')
%}
%{
plot(tRNA,evRNA);
set(legend('$e[\frac{rad}{s}]$'), 'interpreter','latex');
axis([0,100,-2.7,1.8])
grid on;
ylabel('Error $e$[$\frac{rad}{s}$]','FontSize',16,'interpreter', 'latex')
xlabel('Tiempo $t$[$s$]','FontSize',16,'interpreter', 'latex')
%}
```