UNIVERSIDAD SAN FRANCISCO DE QUITO

COLEGIO DE CIENCIAS E INGENIERÍA

Estimación en Tiempo real de Torque y Velocidad en un Motor Trifásico

PROYECTO PREVIO A LA OBTENCION DEL TÍTULO DE INGENIERO ELECTRÓNICO

Danilo Xavier Llano Chiguano

DIRECTOR: Alberto Sánchez Terán, PhD.

31 de agosto de 2012

Universidad San Francisco de Quito
Colegio de Ciencias e Ingeniería Politécnico

HOJA DE APROBACIÓN DE TESIS

Estimación en Tiempo real de Torque y Velocidad en un Motor Trifásico.

Danilo Llano

Alberto Sánchez, PhD. ....
Director de la Tesis

René Játiva, PhD. .....
Miembro del Comité de Tesis

Eduardo Alba, Dr. .....
Miembro del Comité de Tesis

Omar Aguirre, M.Sc. .....
Director Ing. Electrónica

Santiago Gangotena, Ph.D. .....
Decano del Colegio de Ciencias e Ingeniería Politécnico

Quito, 31 de agosto de 2012

DERECHOS DE AUTOR

Yo Danilo Xavier Llano Chiguano, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido presentado previamente para ningún grado o calificación profesional; y, que he consultado las referencias bibliografícas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo a la Universidad San Francisco de Quito, según lo establecido por la Ley de Propiedad Intelectual, por su reglamento y por la normatividad institucional vigente.

Danilo Xavier Llano Chiguano

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Danilo Xavier Llano Chiguano, bajo mi supervisión.

Alberto Sánchez Terán, PhD

DIRECTOR DEL PROYECTO

AGRADECIMIENTOS

A mis padres Juan y María y mis padrinos Harry y Rosemarie; por su amor, comprensión y apoyo a lo largo de todo el camino recorrido.

A Alberto Sánchez, más que un profesor un amigo, por sus enseñanzas, recomendaciones, ideas y sugerencias durante este proyecto y en las clases que he tomado con él.

A Nelson Herrera por toda la ayuda brindada durante mi carrera en la universidad.

A mis profesores de las áreas de Ing. Electrónica, Física y Matemática: Omar, René, Darío, Carlos, David y Eduardo.

A mis amigos físicos y colegas ingenieros, especialmente a mi gran amiga Clarita, por su sincera amistad y todos los buenos momentos compartidos, pero por sobre todo por la confianza y fe que tenían en mí; incluso más de la que yo mismo tenía.

DEDICATORIA

A mis padres por su infinito amor, sacrificio y paciencia.

A Carolina, por haberme transmitido parte de su filosofía de vida y mostrarme que el lado oscuro siempre tiene un lado brillante por descubrirse.

Resumen

El presente proyecto comprende un análisis comparativo de dos técnicas de estimación de variables de estado en una máquina de inducción operando como motor, el filtro Kalman Extendido y el filtro H_∞ extendido. Los dos filtros han sido modificados para la estimación de sistemas no lineales como es el caso de la máquina de inducción. El análisis considera los resultados en simulación y en tiempo real. La implementación consideró no solo la estimación sino también la complejidad del algoritmo y la carga computacional. Para las pruebas experimentales se aplicaron perturbaciones de carga y cambios en frecuencia.

En términos generales, el filtro Kalman Extendido es una técnica de estimación clásica, en la que se asume ruido gaussiano y no correlación en el ruido de proceso y ruido de medición. Este filtro minimiza la varianza del error de forma óptima. Por años ha sido estudiado e implementado debido a su gran versatilidad y poca complejidad. Sin embargo, debido a sus limitaciones sugieron también métodos de estimación más robustos. En el caso de Kalman, el desempeño del filtro depende fuertemente de un conocimiento profundo de la planta, lo cual en la mayoría de aplicaciones reales no es posible. Así también, la sintonización de sus parámetros es engorrosa.

El filtro H_∞ extendido es una técnica de estimación robusta moderna, menos estudiada que el filtro Kalman. Debido a su formulación matemática, este filtro estima minimizando el error en el peor de los casos, es decir cuando el sistema se comporta de forma tal que el error en la estimación es máximo. No se requiere asumir condiciones sobre la naturaleza del ruido en el sistema, lo cual hace que el filtro sea más real en la medida que los parámetros de sintonización pueden determinarse experimentalmente y deben ajustarse a una forma en particular. No obstante, al ser un caso más general requiere de una mayor carga computacional que puede llegar a ser crítico en una implementación real. A nivel simulación este detalle no es fundamental.

Todo el análisis de estas dos técnicas de estimación parten de un modelo de la máquina de inducción que tiene a los flujos magnéticos $qd0$ de rotor y estator, la velocidad del rotor y el torque de carga como variables de estado. Las salidas del modelo son las corrientes $qd0$ del estator. Sobre este modelo se desarrollan los algoritmos de estimación. A través de los resultados de la simulación e implementación de los filtros se identificaron sus principales características, ventajas y desventajas.

Abstract

Internal induction motor variables estimators are highly required in field applications [3]. They are the entrance to modern control techniques for induction machines. Even though there is a huge range of variables or parameters which can be estimated, load torque and rotor speed are prior in sensorless control techniques because most industrial applications require control of these variables. To estimate rotor speed is neccesary to estimate rotor and stator currents or fluxes (depending on the model used) as well. As consequence with an estimator is possible to know and control almost all the performance in an induction machine in sensorless way.

One of the most known and used estimator is the Kalman filter and its extended version for parameter estimation, not only states in non-linear systems [10]. Theory regarding this estimator applied to induction machines is well known and authors have proposed different techniques and models [11, 12, 2] to approach. However, all the algorithms need stator voltages and currents measurements to work. Stator voltage measurements are used to calculate the states in the model and stator currents measurements are used to calculate the error between estimated and measured outputs in most of the models. All the estimation is based on the error mentioned above. To reduce computation requirements some authors have proposed alternative algorithms for Kalman estimation [13, 14], but they are usually restricted to particular problems or models.

H_∞ filter is less known and there is no much information about real-problem implementations, but this filter was specifically developed for robustness. H_∞ filters are robust filters that minimizes the worst-case error (bounded peak error gain). Perturbations are assumed to be energy-bounded signals without specific statistics. H_∞ filters do not make assumptions about error statistics and are robust to uncertain in the model parameters or errors in covariance matrices tunning.

This articule presents an estimation for both variables, speed and load torque by using a sixth order extended Kalman Filter and an extended H_∞ Filter. In the case of the H_∞ filter the cost function to be minimizated will be limited to rotor speed and load torque error estimation to reduce computation required. The space state model for the motor is writen as function of machine magnetic fluxes because it reduces significantly the number of non constant terms in the model and also reduces the number of computations perform by the DSP in real implementation.

Índice

1. Introducción **1**
1.1. Modelo dinámico del motor trifásico . . . 1
1.1.1. Operación básica de la máquina de inducción . . . 1
1.1.2. Transformación de coordenadas *abc* a *qd*0 . . . 2
1.1.3. Modelo continuo de la máquina de inducción trifásica . . . 3
1.1.4. Modelo discreto de la máquina de inducción trifásica . . . 11
1.1.5. Medición de los parámetros del motor. . . . 12
1.2. Estimadores de estado . . . 14
1.3. Reseña de objetivos . . . 15

2. Estimadores de estado **17**
2.1. Filtros H_2/ Algoritmo extendido de Kalman . . . 17
2.1.1. Filtro Kalman Discreto. . . . 17
2.1.2. Propiedades del Filtro Kalman . . . 19
2.1.3. Filtro Kalman extendido (Sistemas no lineales) . . . 20
2.2. Filtros H_∞ . . . 22
2.2.1. Filtro H_∞ Discreto. . . . 22
2.2.2. Propiedades del filtro H_∞ . . . 25
2.2.3. Filtro H_∞ extendido (Sistemas no lineales) . . . 26

3. Desarrollo de algoritmos de estimación **30**
3.1. Simulación filtro H_2/Kalman extendido. . . . 32
3.1.1. Aproximación lineal en T_s-Arranque . . . 33
3.1.2. Aproximación lineal en T_s-Estado estable sin perturbación . . . 35
3.1.3. Aproximación lineal en T_s-Perturbación de carga . . . 39
3.1.4. Aproximación lineal en T_s-Vista detallada de los efectos de la perturbación 41
3.1.5. Aproximación cuadrática en T_s-Arranque . . . 43
3.1.6. Aproximación cuadrática en T_s-Estado estable sin perturbación . . . 46
3.1.7. Aproximación cuadrática en T_s-Perturbación . . . 49
3.1.8. Aproximación cuadrática en T_s-Vista detallada de los efectos de la perturbación . . . 52
3.2. Simulación filtro H_∞ . . . 55
3.2.1. Filtro con $L = I_{6\times6}$- Arranque . . . 57
3.2.2. Filtro con $L = I_{6\times6}$- Estado estable sin perturbación . . . 60
3.2.3. Filtro con $L = I_{6\times6}$- Perturbación . . . 62
3.2.4. Filtro con $L = I_{6\times6}$- Vista detallada de los efectos de la perturbación . . 65
3.2.5. Filtro con $L = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ - Arranque . . . 68
3.2.6. Filtro con $L = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$- Estados estable sin perturbación . 71
3.2.7. Filtro con $L = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$- Perturbación . . . 74

3.2.8. Filtro con $L = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$-Vista detallada de los efectos de la perturbación 76

4. Implementación, pruebas de campo y sintonización de parámetros. 81
4.1. Generalidades sobre el software VisSim, DSP F2808 y modulo de potencia de Francecol 81
4.1.1. VisSim 81
4.1.2. DSP TMS320F2808 de Texas Instruments 83
4.1.3. Módulo de potencia V4 de Francecol 84
4.1.4. Hardware adicional 87
4.2. Implementación en hardware de filtros Kalman extendidos y H_∞ 87
4.2.1. Filtro Kalman extendido 91
4.2.2. Filtro H_∞ extendido 95
4.3. Resultados de pruebas en tiempo real de filtros Kalman extendido y H_∞ extendido 99
4.3.1. Filtro Kalman extendido-Alta frecuencia (40-60 Hz) 99
4.3.2. Filtro Kalman extendido- Frecuencia media (25-40 Hz) 120
4.3.3. Filtro Kalman extendido-Baja frecuencia (0-25 Hz) 141
4.3.4. Filtro H_∞ extendido-Alta frecuencia (40-60 Hz) 160
4.3.5. Filtro H_∞ extendido- Frecuencia media (25-40 Hz) 181
4.3.6. Filtro H_∞ extendido-Baja frecuencia (0-25 Hz) 201
4.4. Comparación del desempeño real de los estimadores. 220

5. Conclusiones y Recomendaciones 224
5.1. Conclusiones 224
5.2. Recomendaciones 227

Presentación

Hasta hace algunos años toda aplicación que requería de un control preciso de posición, torque o velocidad se realizaba con motores DC debido a su facilidad de control y al alto grado de precisión. No obstante hay una serie de inconvenientes al utilizar máquinas DC, como por ejemplo: un mayor número de componentes mecánicos, mantenimiento permanente, necesidad de rectificadores, entre otros. Por otra parte, si bien existe el fundamento teórico para tener un control preciso de máquinas AC desde los años 80, éstos no podían ser implementados debido a las limitaciones tecnológicas de la época, i.e. poca capacidad de procesamiento y velocidad en controladores y no disponibilidad de dispositivos de potencia adecuados.

Hoy en día, con el desarrollo de la tecnología en microprocesadores de alta velocidad como DSPs o FPGAs es posible implementar algoritmos de control sofisticados que requieren de filtros de estimación de variables internas del motor, las cuales son imposibles de medir, o muy costosas. Los estimadores son filtros capaces de estimar variables de interés a partir de mediciones de variables fáciles de obtener. Existen varios estimadores, entre ellos: Filtros Kalman Extendidos (H_2), H_∞ que permite calcular flujos, velocidades y torques a partir de mediciones de corrientes y voltajes en una máquina. La estimación se realiza mediante algoritmos matemáticos que minimizan efectos indeseables tales como ruido en las mediciones. Este tipo de filtros han permitido implementar técnicas de control como Control por Vector de Campo o Control de Torque Directo de forma sensorless, es decir sin necesidad de sensores como encoders, medidores de flujo magnético y dinamómetros; y minimizando el conocimiento exacto de los parámetros de la máquina.

Para entender mejor la teoría matemática y computacional detrás de los filtros mencionados, el presente trabajo muestra la base teórica de los algoritmos a implementarse, simulaciones del desempeño de los filtros en Simulink e implementación en tiempo real de los filtros Kalman extendido y H_∞ extendido. Para los dos filtros se mide las corrientes y los voltajes de cada fase. A partir de la variación o error entre los valores medidos para la salida y los valores calculados se obtiene una ganancia óptima estadísticamente que minimiza la función de costo del filtro y permite tener una estimación de las variables: velocidad del rotor y torque de carga. Adicionalmente se estiman los flujos magnéticos tanto en el estator como en el rotor.

En el primer capítulo se introduce el modelo dinámico de la máquina de inducción en variables de estado en tiempo continuo y discreto. El modelo deducido se basa en el modelo presentado por Krause [2]. Se utiliza el modelo en tiempo continuo para fines de simulación y como base para deducir el modelo discreto. El modelo discreto se utiliza en los algoritmos de estimación. Adicionalmente, se presenta el procedimiento de medición de las constantes eléctricas (resistencias y reactancias) y mecánicas del sistema (inercia de carga y constante de viscosidad) mediante las pruebas de rotor bloqueado y rotor libre. Posteriormente se introduce el concepto de estimadores de estados. Finalmente, en el capítulo 1 se resumen brevemente los objetivos del proyecto.

En el segundo capítulo se estudia en detalle los estimadores de estado: Filtro Kalman Extendido y Filtro H_∞ extendido. En este capítulo se detalla el fundamento matemático de los dos filtros, sus algoritmos computacionales y la adaptación del modelo dinámico del motor a los requerimientos de los respectivos algoritmos. Se pone particular énfasis en la estimación de las variables torque de carga y velocidad del rotor.

El capítulo 3 trata sobre la simulación en Simulink del modelo dinámico del motor y cada uno de los estimadores. Se muestran los resultados de simulación del sistema y los filtros en el caso de producirse perturbaciones de carga. A partir de los resultados obtenidos se procederá a seleccionar los modelos de filtro (uno para Kalman y otro para H_∞) más eficientes en términos de calidad de estimación y carga computacional.

Una vez finalizada la etapa de simulación se procede al desarrollo de algoritmos en VisSim. En el capítulo 4 se presenta una breve introducción al software VisSim y los componentes de hardware utilizados en la implementación. Se mencionan las ventajas y limitaciones del software VisSim así como su integración con el programa Code Composer Studio de Texas Instruments. En hardware se describe las características más importantes del DSP F2808 de Texas Instruments y del módulo de potencia de electrónica digital de Francecol. Posteriormente se describe la programación de los algoritmos en VisSim y su descarga al DSP F2808 para implementación en tiempo real. Finalmente se presentan los resultados de las pruebas en tiempo real para cada una de las perturbaciones realizadas, i.e. cambio en el torque de carga y cambio en la frecuencia de alimentación.

Finalmente, el Capítulo 5 es un resumen detallado de las conclusiones y recomendaciones que se han presentado a lo largo de este proyecto. Brevemente se resaltan los aspectos más sobresalientes de este trabajo: el modelo de la máquina de inducción a nivel teórico, medición de parámetros en la máquina, desarrollo del modelo del motor en tiempo discreto, estimadores a nivel simulación, programación de los algoritmos e implementación en tiempo real, resultados finales de la implementación y comparación del desempeño de los dos filtros.

1. Introducción

1.1. Modelo dinámico del motor trifásico

1.1.1. Operación básica de la máquina de inducción

Para describir el funcionamiento de una máquina de inducción se utilizan las leyes de Maxwell, en particular las leyes de Faraday y Lenz. Su gran versatilidad y desempeño han popularizado su uso a tal punto que resulta impresindible en aplicaciones desde los implementos del hogar como licuadoras o secadoras hasta grandes molinos, bandas transportadoras y sistemas de bombeo.

Un motor trifásico tiene una parte estatórica y otra rotórica. Las bobinas del estator están distribuídas a 120^{o} geométricos e idealmente se requiere que sean senosoidalmente distribuidas en el espacio para evitar pérdidas y potenciales daños al motor debido a bruscos cambios en el sentido del torque. Adicionalmente, por la bobinas señaladas circula corriente trifásica, con un desfase eléctrico de 120^{o}. Esto produce un campo magnético giratorio de amplitud constante (en el caso ideal) que rodea al rotor. Este campo giratorio rota a una velocidad definida por la frecuencia de la alimentación eléctrica y el número de polos de la máquina. La velocidad de giro es conocida como velocidad de sincronismo.

El flujo magnético que se genera debido a la corriente que circula por las bobinas del estator atraviesa los bobinados del rotor, induciendo corriente en dichos bobinados. Esto genera una fuerza que se dispone en la dirección del campo magnético rotatorio. Esta fuerza genera torque, haciendo que el bobinado del rotor (montado sobre un eje que le permite girar) empiece a rotar en el mismo sentido que el flujo pero a una velocidad ligeramente menor tratando de alinearse con el estator. El cociente de la diferencia de velocidades del rotor y sincronismo respecto a la velocidad de sincronismo se conoce como deslizamiento.

$$s = \frac{s_r - s_e}{s_e} \tag{1.1}$$

donde s_e es la velocidad de sincronismo y s_r se la velocidad a la que gira el rotor.

La velocidad de sincronismo s_e a la que gira el campo del estator debido a las corrientes trifásicas que circulan y la distribución espacial de las bobinas depende de la frecuencia eléctrica f_e y del número de polos de la máquina P y se define como:

$$s_e = 120\frac{fe}{P}[rpm] \tag{1.2}$$

En condiciones normales de operación el campo del estator y rotor giran a aproximadamente la misma velocidad mientras que el rotor, con su componentes mecánicos, gira a una velocidad ligeramente menor. El hecho de tener partes de una máquina en movimiento circular hace que los parámetros (reactancias) cambien en el tiempo debido a la orientación espacial del rotor. Las reactancias dependen fuertemente de su posición en el espacio y de la velocidad a la que gira. Esto aumenta obstensiblemente la complejidad de las ecuaciones diferenciales que modelan el sistema. Por tales motivos se desarrollaron algoritmos matemáticos que simplifiquen estos términos y permitan obtener expresiones más sencillas. Las transformaciones matemáticas referidas no son más que cambios en el marco de referencia del observador o transformación de coordenadas; de las llamadas abc a coordenadas $qd0$.

1.1.2. Transformación de coordenadas *abc* a *qd*0

El estudio de tranformación de coordenadas se remonta a 1920. Este año R. Park introduce un cambio de variables para la máquina sincrónica en el que se refieren las variables del estator a un eje que se mueve a la velocidad del rotor. Hasta 1965 existen una serie de estudios de parte de H. Stanley (1930), G. Kron y D. Brereton que trataban distintos cambios de coordenadas en diferentes sistemas de coordenadas. No obstante, en 1965 P. Krause sintetiza todos los estudios previos en una sola tranformación generalizada.

La transformación propuesta por Krause [2] se escribe matricialmente como:

$$f_{qd0s} = K_s f_{abcs} \tag{1.3}$$

$$f_{qd0s} = \begin{bmatrix} f_{qs} & f_{ds} & f_{0s} \end{bmatrix}^T \tag{1.4}$$

$$f_{abcs} = \begin{bmatrix} f_{as} & f_{bs} & f_{cs} \end{bmatrix}^T \tag{1.5}$$

$$\theta = \int \omega dt \tag{1.6}$$

$$K_s = \frac{2}{3} \begin{bmatrix} \cos\theta & \cos(\theta - 2\pi/3) & \cos(\theta + 2\pi/3) \\ \sin\theta & \sin(\theta - 2\pi/3) & \sin(\theta + 2\pi/3) \\ 1/2 & 1/2 & 1/2 \end{bmatrix} \tag{1.7}$$

La constante 2/3 permite conservar la potencia invariante en el cambio de coordenadas.

Además se define la respectiva transformación inversa mediante:

$$f_{abcs} = K_s^{-1} f_{qd0s} \tag{1.8}$$

$$K_s^{-1} = \begin{bmatrix} \cos\theta & \sin\theta & 1 \\ \cos(\theta - 2\pi/3) & \sin(\theta - 2\pi/3) & 1 \\ \cos(\theta + 2\pi/3) & \sin(\theta - 2\pi/3) & 1 \end{bmatrix} \tag{1.9}$$

Las ecuaciones (1.3) y (1.8) se pueden utilizar para cambiar de coordenadas a cualquier variable en estudio, a saber voltaje, corriente, flujos magnéticos, impendancia entre otros.

En el caso de un sistema trifásico balanceado se puede mostrar que el sistema se reduce a un sistema bifásico, dado que la componente 0 de la tranformación es cero, con lo que el sistema se modela perfectamente a partir de las coordenadas q y d únicamente. Esta es una de las características más importantes de la transformación de coordenadas y como se verá más adelante, la segunda ventaja es el desacoplamiento de las reactancias, de la posición y velocidad del rotor.

La transformación de coordenadas va más allá de lo mencionado, ya que permite relacionar sistemas de referencia completamente independientes sin pasar por las coordenadas *abc*. Si se requiere pasar de variables en un sistema referencial x a variables en un sistema referencial z la transformación se define como:

$$f^z_{qd0} =^x K^z f^x_{qd0} \tag{1.10}$$

$$f^x_{qd0} = K^x f_{abc} \tag{1.11}$$

Al reemplazar (1.11) en (1.10) y considerar:

$$f^z_{qd0} = K^z f_{abc} \tag{1.12}$$

por simple inspección se obtiene que

$${}^xK^z = K^z \left(K^x\right)^{-1} \tag{1.13}$$

con la matriz asociada:

$${}^xK^z = \frac{2}{3}\begin{bmatrix} \cos\left(\theta_z - \theta_x\right) & -\sin\left(\theta_z - \theta_x\right) & 0 \\ \sin\left(\theta_z - \theta_x\right) & \cos\left(\theta_z - \theta_x\right) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.14}$$

donde los ángulos θ_z y θ_x son los desplazamientos angulares del sistema.

En este caso se utilizará una transformación de coordenadas a ejes $qd0$ para simplificar la expresión matemática del modelo. Sin embargo, la teoría de ejes de referencia tiene un sinnúmero de aplicaciones como técnicas de control vectoriales en máquinas AC, análisis de estabilidad de sistemas eléctricos de potencia, estudio de máquinas síncronas, entre otros.

1.1.3. Modelo continuo de la máquina de inducción trifásica

En la primera sección de este capítulo se explicó el funcionamiento de la máquina de inducción. En esta sección se mostrará con detalle el modelo matemático que representa la descripción ya realizada. El modelo matemático para el motor se basa en el modelo propuesto por Krause [2]. Este modelo considera los flujos magnéticos como variables de estado.

Considére una máquina trifásica balanceada conectada en estrella (Y) y alimentada por una fuente de igual manera trifásica de voltajes v_a, v_b y v_c . Esto hace que circulen corrientes trifásica simétricas notadas como i_a, i_b e i_c. Al modelar el motor mediante un sistema ideal, cuyo entrehierro es uniforme, trabajar en la región lineal (se desprecia saturación magnética) y considerar la Figura 1.1 se pueden escribir las ecuaciones de voltaje para cada una de las fases tanto en el estator (subíndice s) como en el rotor (subíndice r).

$$v_{abcs} = r_s i_{abcs} + \frac{d\lambda_{abcs}}{dt} \tag{1.15}$$

$$v_{abcr} = r_r i_{abcr} + \frac{d\lambda_{abcr}}{dt} \tag{1.16}$$

Los índices abc indican que la ecuación es exactamente la misma para cada una de las fases.

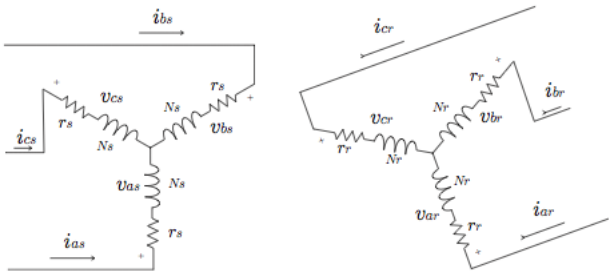


Figura 1.1: Máquina de Inducción- Fases. Tomado de [2]

Las ecuaciones (1.15) y (1.16) requieren conocer las concatenaciones de flujo λ_{abcs} y λ_{abcr}. Dichas concatenaciones se relacionan con las corrientes mediante:

$$\begin{bmatrix} \lambda_{abcs} \\ \lambda_{abcr} \end{bmatrix} = \begin{bmatrix} L_s & L_{sr} \\ L_{sr}^T & L_r \end{bmatrix} \begin{bmatrix} i_{abcs} \\ i_{abcr} \end{bmatrix} \tag{1.17}$$

con L_s y L_r como las matrices de inductancias de magnetización y dispersión del estator y rotor respectivamente; y L_{sr} como la matriz de inductancias mutuas entre el estator y rotor.

Las matrices mencionadas se definen mediante:

$$L_s = \begin{bmatrix} L_{ls} + L_{ms} & -0{,}5L_{ms} & -0{,}5L_{ms} \\ -0{,}5L_{ms} & L_{ls} + L_{ms} & -0{,}5L_{ms} \\ -0{,}5L_{ms} & -0{,}5L_{ms} & L_{ls} + L_{ms} \end{bmatrix} \tag{1.18}$$

$$L_r = \begin{bmatrix} L_{lr} + L_{mr} & -0{,}5L_{mr} & -0{,}5L_{mr} \\ -0{,}5L_{mr} & L_{lr} + L_{mr} & -0{,}5L_{mr} \\ -0{,}5L_{mr} & -0{,}5L_{mr} & L_{lr} + L_{mr} \end{bmatrix} \tag{1.19}$$

$$L_{sr} = L_{sr} \begin{bmatrix} \cos(\theta_r) & \cos\left(\theta_r + \frac{2\pi}{3}\right) & \cos\left(\theta_r - \frac{2\pi}{3}\right) \\ \cos\left(\theta_r - \frac{2\pi}{3}\right) & \cos(\theta_r) & \cos\left(\theta_r + \frac{2\pi}{3}\right) \\ \cos\left(\theta_r + \frac{2\pi}{3}\right) & \cos\left(\theta_r - \frac{2\pi}{3}\right) & \cos(\theta_r) \end{bmatrix} \tag{1.20}$$

Para expresar las variables del rotor referenciadas al estator se utiliza la relación de vueltas entre los devanados del estator y rotor

$$N = \frac{N_s}{N_r} \tag{1.21}$$

Por convención, las variables referidas se identifican mediante un apóstrofe como se muestra a continuación:

$$\begin{bmatrix} \lambda_{abcs} \\ \lambda'_{abcr} \end{bmatrix} = \begin{bmatrix} L_s & L'_{sr} \\ (L'_{sr})^T & L_r \end{bmatrix} \begin{bmatrix} i_{abcs} \\ i'_{abcr} \end{bmatrix} \tag{1.22}$$

Para poder simplificar el modelo se utiliza una transformación de coordenadas. En este caso se utilizará un eje de referencia estacionario, pero se pueden tener cualquiera de los casos mostrados en la Tabla 1.

Velocidad	Intepretación	Variables	Transformación
ω	Variables referidas a eje arbitrario	f_{qd0s}	K_s
0	Variables referidas a eje estacionario	f^s_{qd0s}	K^s_s
ω_r	Variables referidas a un eje a la velocidad del rotor	f^r_{qd0s}	K^r_s
ω_e	Variables referidas a un eje a la velocidad sincrónica	f^e_{qd0s}	K^e_s

Cuadro 1: Transformación de coordenadas

Considerando las transformaciones directa e inversa definidas en (1.3) y (1.8) respectivamente se obtiene:

$$v_{abcs} = K_s^{-1} v_{qd0s} \tag{1.23}$$

$$i_{abcs} = K_s^{-1} i_{qd0s} \tag{1.24}$$

$$\lambda_{abcs} = K_s^{-1} \lambda_{qd0s} \tag{1.25}$$

Se puede operar con (1.3) y (1.8) para demostrar que:

$$K_s p\left(K_s^{-1}\right) = \omega \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{1.26}$$

Al aplicar la transformación de coordenadas a (1.15) se tiene:

$$v_{dq0s} = r_s i_{qd0s} + K_s p\left(K_s^{-1}\right) \lambda_{qd0s} + K_s K_s^{-1} p \lambda_{qd0s} \tag{1.27}$$

A continuación se reemplaza $K_s p\left(K_s^{-1}\right)$ en la ecuación anterior y se tiene:

$$v_{dq0s} = r_s i_{qd0s} + \omega \lambda_{qd0s} + p \lambda_{qd0s} \tag{1.28}$$

con

$$\lambda^T_{qd0s} = \begin{bmatrix} \lambda_{ds} & -\lambda_{qs} & 0 \end{bmatrix} \tag{1.29}$$

En el rotor se obtiene:

$$v_{dq0r} = r'_r i_{qd0r} + (\omega - \omega_r) \lambda'_{qd0r} + p \lambda'_{qd0r} \tag{1.30}$$

y se define:

$$\lambda^T_{qd0r} = \begin{bmatrix} \lambda'_{dr} & -\lambda'_{qr} & 0 \end{bmatrix} \tag{1.31}$$

En la ecuación (1.30) ω se refiere a la velocidad del sistema de referencia escogido según la tabla (1.1.3) y ω_r a la velocidad del rotor.

De igual manera las concatenaciones de flujo descritas en (1.22) se pueden escribir de la siguiente manera en coordenadas $qd0$.

$$\begin{bmatrix} \lambda_{qd0s} \\ \lambda'_{qd0r} \end{bmatrix} = \begin{bmatrix} K_s L_s K_s^{-1} & K_s L'_{sr} K_r^{-1} \\ K_r \left(L'_{sr}\right)^T K_s^{-1} & K_r L_r K_r^{-1} \end{bmatrix} \begin{bmatrix} i_{qd0s} \\ i'_{qd0r} \end{bmatrix} \tag{1.32}$$

Operando sobre las entradas de la matriz de reactancias de la ecuación (1.32) se obtiene:

$$K_s L_s K_s^{-1} = \begin{bmatrix} L_{ls} + \frac{3}{2}L_{ms} & 0 & 0 \\ 0 & L_{ls} + \frac{3}{2}L_{ms} & 0 \\ 0 & 0 & L_{ls} \end{bmatrix} \tag{1.33}$$

$$K_r L'_r K_r^{-1} = \begin{bmatrix} L'_{lr} + \frac{3}{2}L_{ms} & 0 & 0 \\ 0 & L'_{lr} + \frac{3}{2}L_{ms} & 0 \\ 0 & 0 & L'_{lr} \end{bmatrix} \tag{1.34}$$

$$K_s L'_{sr} K_r^{-1} = K_r \left(L'_{sr}\right)^T K_s^{-1} = \begin{bmatrix} \frac{3}{2}L_{ms} & 0 & 0 \\ 0 & \frac{3}{2}L_{ms} & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{1.35}$$

Con las tranformaciones de las ecuaciones (1.33), (1.34) y (1.35) se tienen las siguientes ecuaciones tanto para los voltajes y concatenaciones de flujo en el estator como en el rotor.

$$v_{qs} = r_s i_{qs} + \omega \lambda_{ds} + p\lambda_{qs} \tag{1.36}$$

$$v_{ds} = r_s i_{ds} - \omega \lambda_{qs} + p\lambda_{ds} \tag{1.37}$$

$$v_{0s} = r_s i_{0s} + p\lambda_{0s} \tag{1.38}$$

$$v'_{qr} = r'_r i'_{qr} + (\omega - \omega_r)\, \lambda'_{dr} + p\lambda'_{qr} \tag{1.39}$$

$$v'_{dr} = r'_r i'_{dr} - (\omega - \omega_r)\, \lambda'_{qr} + p\lambda'_{dr} \tag{1.40}$$

$$v_{0r} = r'_r i'_{0r} + p\lambda'_{0r} \tag{1.41}$$

$$\begin{bmatrix} \lambda_{qs} \\ \lambda_{ds} \\ \lambda_{0s} \\ \lambda_{qr} \\ \lambda_{dr} \\ \lambda_{0r} \end{bmatrix} = \begin{bmatrix} L_{ls} + \frac{3}{2}L_{ms} & 0 & 0 & \frac{3}{2}L_{ms} & 0 & 0 \\ 0 & L_{ls} + \frac{3}{2}L_{ms} & 0 & 0 & \frac{3}{2}L_{ms} & 0 \\ 0 & 0 & L_{ls} & 0 & 0 & 0 \\ \frac{3}{2}L_{ms} & 0 & 0 & L_{lr} + \frac{3}{2}L_{ms} & 0 & 0 \\ 0 & \frac{3}{2}L_{ms} & 0 & 0 & L_{lr} + \frac{3}{2}L_{ms} & 0 \\ 0 & 0 & 0 & 0 & 0 & L_{lr} \end{bmatrix} \begin{bmatrix} i_{qs} \\ i_{ds} \\ i_{0s} \\ i_{qr} \\ i_{dr} \\ i_{0r} \end{bmatrix} \tag{1.42}$$

El torque de la máquina depende de una cantidad conocida como coenergía que se almacena en el campo magnético del entrehierro. De esta manera el torque dependerá tanto de las corrientes inducidas como de las corrientes propias del sistema. En una máquina trabajando en su zona lineal el torque electromagnético esta dado por:

$$T_{em} = \left(\frac{P}{2}\right) (i_{abcs})^T \frac{\partial}{\partial \theta_r} \left[L'_{sr}\right] i'_{abcr} \tag{1.43}$$

Si T_{em} es positivo la máquina trabaja como motor. Al aplicar las transformaciones $qd0$ a la ecuación de torque se obtienen tres ecuaciones equivalentes simplificadas en función de corrientes y concatenaciones de flujo:

$$T_{em} = \left(\frac{3}{2}\right)\left(\frac{P}{2}\right)\left(\frac{3}{2}L_{ms}\right)\left(i_{qs}i'_{dr} - i_{ds}i'_{qr}\right) \tag{1.44}$$

$$T_{em} = \left(\frac{3}{2}\right)\left(\frac{P}{2}\right)\left(i_{qs}\lambda_{ds} - i_{ds}\lambda_{qs}\right) \tag{1.45}$$

$$T_{em} = \left(\frac{3}{2}\right)\left(\frac{P}{2}\right)\left(i'_{qr}\lambda'_{dr} - i'_{qr}\lambda'_{dr}\right) \tag{1.46}$$

Es importante resaltar que el desplazamiento angular eléctrico y el desplazamiento mecánico están relacionados mediante el número de polos por:

$$\theta_r = \left(\frac{P}{2}\right)\theta_m \tag{1.47}$$

En el caso particular del motor que se utilizará en la implementación en este proyecto $P = 2$ por lo que no hay diferencia entre los desplazamientos angulares.

Si bien se tienen ya las ecuaciones necesarias para plantear el modelo dinámico de la máquina de inducción se procederá a reescribir las mismas en variables por unidad (notación común en la literatura referente al tema) mediante la introducción de dos definiciones, a saber, la concatenación de flujo por segundo ψ y las reactancias de la máquina X.

$$\psi = \omega_b \lambda \tag{1.48}$$

$$X = \omega_b L \tag{1.49}$$

donde ω_b se conoce como velocidad angular base. En general se utiliza $\omega_b = 120\pi$ por ser la velocidad asociada a la frecuencia de la red de alimentación $(60Hz)$

Al reescribir las ecuaciones desde (1.36) hasta (1.42) en variables por unidad de obtiene:

$$v_{qs} = r_s i_{qs} + \frac{\omega}{\omega_b}\psi_{ds} + \frac{p}{\omega_b}\psi_{qs} \tag{1.50}$$

$$v_{ds} = r_s i_{ds} - \frac{\omega}{\omega_b}\psi_{qs} + \frac{p}{\omega_b}\psi_{ds} \tag{1.51}$$

$$v_{0s} = r_s i_{0s} + \frac{p}{\omega_b}\psi_{0s} \tag{1.52}$$

$$v'_{qr} = r'_r i'_{qr} + \left(\frac{\omega - \omega_r}{\omega_b}\right)\psi'_{dr} + \frac{p}{\omega_b}\psi'_{qr} \tag{1.53}$$

$$v'_{dr} = r'_r i'_{dr} - \left(\frac{\omega - \omega_r}{\omega_b}\right)\psi'_{qr} + \frac{p}{\omega_b}\psi'_{dr} \tag{1.54}$$

$$v_{0r} = r'_r i'_{0r} + \frac{p}{\omega_b}\psi'_{0r} \tag{1.55}$$

$$\begin{bmatrix} \psi_{qs} \\ \psi_{ds} \\ \psi_{0s} \\ \psi_{qr} \\ \psi_{dr} \\ \psi_{0r} \end{bmatrix} = \begin{bmatrix} X_{ls} + \frac{3}{2}X_{ms} & 0 & 0 & \frac{3}{2}X_{ms} & 0 & 0 \\ 0 & X_{ls} + \frac{3}{2}X_{ms} & 0 & 0 & \frac{3}{2}X_{ms} & 0 \\ 0 & 0 & X_{ls} & 0 & 0 & 0 \\ \frac{3}{2}X_{ms} & 0 & 0 & X_{lr} + \frac{3}{2}X_{ms} & 0 & 0 \\ 0 & \frac{3}{2}X_{ms} & 0 & 0 & X_{lr} + \frac{3}{2}X_{ms} & 0 \\ 0 & 0 & 0 & 0 & 0 & X_{lr} \end{bmatrix} \begin{bmatrix} i_{qs} \\ i_{ds} \\ i_{0s} \\ i_{qr} \\ i_{dr} \\ i_{0r} \end{bmatrix} \tag{1.56}$$

Para facilitar la escritura del sistema en espacio de estado, de forma tal que las corrientes estatóricas queden como las salidas del sistema se reescriben las ecuaciones que relacionan a los flujos por unidad de tiempo con las corrientes i.e. la ecuación (1.56).

$$i_{qs} = \frac{1}{X_{ls}}\left(\psi_{qs} - \psi_{mq}\right) \tag{1.57}$$

$$i_{ds} = \frac{1}{X_{ls}}\left(\psi_{ds} - \psi_{md}\right) \tag{1.58}$$

$$i_{0s} = \frac{\psi_{0s}}{X_{ls}} \tag{1.59}$$

$$i'_{qr} = \frac{1}{X'_{lr}}\left(\psi'_{qs} - \psi_{mq}\right) \tag{1.60}$$

$$i'_{dr} = \frac{1}{X'_{lr}}\left(\psi'_{dr} - \psi_{md}\right) \tag{1.61}$$

$$i'_{0r} = \frac{\psi'_{0r}}{X'_{lr}} \tag{1.62}$$

Reemplazando las ecuaciones deducidas anteriormente, desde (1.57) hasta (1.62) en las ecuaciones del voltaje de la máquina, es decir las ecuaciones de (1.50) a (1.55) se obtienen las siguientes expresiones simplificadas y expresadas en la forma estándar para espacio de estados:

$$p\psi_{qs} = \omega_b\left[v_{qs} - \frac{\omega}{\omega_b}\psi_{ds} + \frac{r_s}{X_{ls}}\left(\psi_{mq} - \psi_{qs}\right)\right] \tag{1.63}$$

$$p\psi_{ds} = \omega_b\left[v_{ds} + \frac{\omega}{\omega_b}\psi_{qs} + \frac{r_s}{X_{ls}}\left(\psi_{md} - \psi_{ds}\right)\right] \tag{1.64}$$

$$p\psi_{0s} = \omega_b\left[v_{0s} - \frac{r_s}{X_{ls}}\psi_{0s}\right] \tag{1.65}$$

$$p\psi'_{qr} = \omega_b\left[v'_{qr} - \left(\frac{\omega - \omega_r}{\omega_b}\right)\psi'_{dr} + \frac{r'_r}{X'_{lr}}\left(\psi_{mq} - \psi'_{qr}\right)\right] \tag{1.66}$$

$$p\psi'_{dr} = \omega_b \left[v'_{dr} + \left(\frac{\omega - \omega_r}{\omega_b} \right) \psi_{qr} + \frac{r'_r}{X_{lr}} \left(\psi_{md} - \psi'_{dr} \right) \right] \tag{1.67}$$

$$p\psi'_{0r} = \omega_b \left[v_{0r} - \frac{r'_r}{X'_{lr}} \psi'_{0r} \right] \tag{1.68}$$

Donde los flujos de magnetización se definen como:

$$\psi_{mq} = X_{aq} \left(\frac{\psi_{qs}}{X_{ls}} + \frac{\psi'_{qr}}{X'_{lr}} \right) \tag{1.69}$$

$$\psi_{md} = X_{ad} \left(\frac{\psi_{ds}}{X_{ls}} + \frac{\psi'_{dr}}{X'_{lr}} \right) \tag{1.70}$$

definiendo:

$$X_{aq} = X_{ad} = \left(\frac{1}{X_{ls}} + \frac{1}{X_m} + \frac{1}{X'_{lr}} \right)^{-1} \tag{1.71}$$

Con las ecuaciones desarrolladas a lo largo del capítulo es posible escribir el modelo dinámico en espacio de estados para el motor de inducción. Por facilidad de computo de escoge el marco de referencia estacionario, $\omega = 0$ para escribir las ecuaciones ya que esto hace que algunas de las entradas en las matrices sean cero y el calculo numérico ya a nivel implementación (sobretodo) sea simplificado.

El modelo continuo en espacio de estados en un marco de referencia estacionario para el motor de inducción se representa de la siguiente manera:

$$\dot{X}(t) = A_c(\omega_r) X(t) + B_c U(t) \tag{1.72}$$

$$Y(t) = C X(t) \tag{1.73}$$

Se utiliza la nomenclatura $A_c(\omega_r)$ porque la matriz A_c varía en el tiempo conforme varía ω_r (velocidad angular del motor)

$$X(t) = \begin{bmatrix} \psi_{qs} & \psi_{ds} & \psi_{qr} & \psi_{dr} \end{bmatrix}^t \tag{1.74}$$

$$Y(t) = \begin{bmatrix} i_{qs} & i_{ds} \end{bmatrix}^t \tag{1.75}$$

$$U(t) = \begin{bmatrix} v_{qs} & v_{ds} \end{bmatrix}^t \tag{1.76}$$

$$A_c = \begin{bmatrix} a_{s1} & 0 & a_{s2} & 0 \\ 0 & a_{s1} & 0 & a_{s2} \\ a_{r2} & 0 & a_{r1} & \omega_r \\ 0 & a_{r2} & -\omega_r & a_{r1} \end{bmatrix} \tag{1.77}$$

$a_{s1} = \frac{\omega_b r_s}{X_{ls}} \left(\frac{X_M}{X_{ls}} - 1 \right)$ $\qquad$ $a_{s2} = \frac{\omega_b r_s X_M}{X_{ls} X_{lr}}$

$a_{r1} = \frac{\omega_b r_r}{X_{lr}} \left(\frac{X_M}{X_{lr}} - 1 \right)$ $\qquad$ $a_{r2} = \frac{\omega_b r_r X_M}{X_{ls} X_{lr}}$

$$B_c = \begin{bmatrix} b & 0 & 0 & 0 \\ 0 & b & 0 & 0 \end{bmatrix}^t \tag{1.78}$$

$$C = \begin{bmatrix} c_{s1} & 0 & c_{s2} & 0 \\ 0 & c_{s1} & 0 & c_{s2} \end{bmatrix} \tag{1.79}$$

$b = \omega_b$, $c_{s1} = \frac{1}{X_{ls}} \left(1 - \frac{X_M}{X_{ls}} \right)$ y $c_{s2} = -\frac{X_M}{X_{ls} X_{lr}}$

Los valores de las entradas del motor están escritas en función de los parámetros del motor, cuya naturaleza física se describe en la Tabla 2

Símbolo	Descripción
X_{lr}	Reactancia del Rotor
X_{ls}	Reactancia del Estator
X_m	Reactacia mútua
r_r	Resistencia rotor
r_s	Resistencia estator
ω_b	velocidad base
J	carga inercial
B	Factor de viscosidad
P	número de polos
X_M	Reactancia efectiva

Cuadro 2: Descripción de los parámetros del motor

Muchos autores usan modelos que tienen corrientes y flujos o solo corrientes como variables de estado [11, 12, 2]. En este trabajo ω_r será tratado como un estado a ser estimado por el filtro Kalman extendido y el filtro H_∞ extendido, por lo tanto la matriz A_c no es constante como ya se indicó. Se debe notar que ω_r está solo en dos entradas de la matriz A_c. En los modelos propuestos por [11, 12, 2] ω_r aparece en al menos cuatro entradas de A. Para fines teóricos y de simulación esta diferencia no es un problema, pero en implementaciones reales, el número de calculos es crítico porque el DSP o FPGA tiene que efectuar todas las operaciones en un intervalo de tiempo que es menor al tiempo de muestreo T_S. Al reducir el número de entradas variables en el sistema se reduce el tiempo de computo y se pueden alcanzar mayores frecuencias de trabajo. A mayor frecuencia de trabajo, mejor es la estimación.

Adicionalmente al modelo eléctrico del motor de inducción es fundamental considerar la ecuación mecánica que relaciona el torque y la velocidad. Esta ecuación se utilizará en el desarrollo del modelo matemático para el filtro Kalman extendido y el filtro H_∞ extendido.

$$\frac{d\omega_r}{dt} = \frac{P}{2J} \left(T_{em} - T_l \right) - \frac{B}{J} \omega_r \tag{1.80}$$

donde T_{em} está dado por (1.45) y escrito en función de variables por unidad:

$$T_{em} = \left(\frac{3}{4} \right) \left(\frac{P}{\omega_b} \right) \left(\psi_{ds} i_{qs} - \psi_{qs} i_{ds} \right) \tag{1.81}$$

1.1.4. Modelo discreto de la máquina de inducción trifásica

Si bien el modelo continuo para el motor de induccion deducido en la sección anterior describe de la mejor manera la dinámica del sistema y puede ser completamente válido para fines de simulación y descripción teórica no es el más adecuado para la implementación física en un DSP.

Para la implementación en un microcontrolador se necesita un modelo discreto del sistema en espacio de estados. Basado en la aproximación que $U(t)$ es aproximadamente constante en un periodo de muestreo constate (T_s) el modelo puede escribirse de la siguiente manera:

$$x[k+1] = A[\omega_r]X[k] + BU[k] \tag{1.82}$$

$$y[k] = CX[k] \tag{1.83}$$

En este caso, las matrices A y B pueden deducirse de las matrices del modelo continuo: A_c y B_c usando la aproximación de la exponencial por polinomios de diferentes grados como se muestra a continuación:

$$A_D = e^{AcTs} \approx I + T_s A_C + \frac{1}{2}T_s^2 A_C^2 \tag{1.84}$$

$$B_D = A_c^{-1}\left(e^{A_c T_s} - 1\right) B_c \approx T_s B_c + \frac{1}{2}A_c B_c T_s^2 \tag{1.85}$$

Varios autores [15, 16, 17, 18] consideran solo los términos lineales en T_s porque el tiempo de muestreo es suficientemente pequeño tal que los términos cuadráticos en T_s pueden ser despreciados. En este trabajo, los términos cuadráticos son considerados para fines de simulación. Luego de la simulación se considerará si la inclusión de los términos cuadráticos justifcan el calculo computacional adicional que se requiere.

La matriz C permanece igual a la matriz del modelo continuo, es decir:

$$C_D = C \tag{1.86}$$

Las matrices para el modelo en espacio de estados discreto, deducidas con (1.84) y (1.85) son las siguientes:

$$A_D = \begin{bmatrix} A_1 & 0 & A_3 & A_4 \\ 0 & A_6 & A_7 & A_8 \\ A_9 & A_{10} & A_{11} & A_{12} \\ A_{13} & A_{14} & A_{15} & A_{16} \end{bmatrix} \tag{1.87}$$

$A_1 = 1 + a_{s1}T_s + \frac{T_s}{2}^2\left(a_{s1}^2 + a_{s2}a_{r2}\right)$
$A_2 = A_5 = 0$
$A_3 = a_{s2}T_s + \frac{T_s}{2}^2\left(a_{s1}a_{s2} + a_{s2}a_{r1}\right)$
$A_4 = \frac{T_s^2}{2}\left(a_{s2}\omega\right)$
$A_6 = 1 + a_{s1}T_s + \frac{T_s^2}{2}\left(a_{s1}^2 + a_{s2}a_{r2}\right)$
$A_7 = -\frac{T_s^2}{2}a_{s2}\omega$

$$A_8 = a_{s2}T_s + \frac{T_s}{2}^2 (a_{s1}a_{s2} + a_{s2}a_{r1})$$
$$A_9 = a_{r2}T_s + \frac{T_s}{2}^2 (a_{r2}a_{s1} + a_{r1}a_{r2})$$
$$A_{10} = \frac{T_s^2}{2} (a_{r2}\omega)$$
$$A_{11} = 1 + a_{r1}T_s + \frac{T_s^2}{2} \left(a_{r2}a_{s2} + a_{r1}^2 - \omega^2\right)$$
$$A_{12} = \omega T_s + T_s^2 a_{r1}\omega$$
$$A_{13} = -\frac{T_s}{2}^2 a_{r2}\omega$$
$$A_{14} = T_s a_{r2} + \frac{T_s^2}{2} (a_{s1}a_{r2} + a_{r1}a_{r2})$$
$$A_{15} = -\omega T_s - T_s^2 a_{r1}\omega$$
$$A_{16} = 1 + T_s a_{r1} + \frac{T_s^2}{2} \left(a_{s2}a_{r2} - \omega^2 + a_{r1}^2\right)$$

$$B_D = \begin{bmatrix} B_1 = bT_s + a_{s1}b\frac{T_s^2}{2} & 0 \\ 0 & B_4 = bT_s + a_{s1}b\frac{T_s^2}{2} \\ B_5 = a_{r2}b\frac{T_s}{2}^2 & 0 \\ 0 & B_8 = a_{r2}b\frac{T_s}{2}^2 \end{bmatrix} \tag{1.88}$$

$$C_D = \begin{bmatrix} c_{s1} & 0 & c_{s2} & 0 \\ 0 & c_{s1} & 0 & c_{s2} \end{bmatrix} \tag{1.89}$$

Todos los términos multiplicados por T_s^2 pueden ser despreciados en el modelo lineal.

1.1.5. Medición de los parámetros del motor.

Para la medición de los parámetros del motor se realizaron las pruebas de rotor bloqueado y rotor sin carga descritas en [1].

La prueba de rotor sin carga se realiza a una frecuencia estándar (60Hz en general) con una fuente de voltaje trifásico balanceada y cuando el motor a operado el tiempo suficiente como para que sus partes mecánicas estén lubricadas adecuadamente. El procedimiento es el siguiente: Se conecta el motor sin carga alguna y se miden voltaje, corriente y potencia activa en cada una de las fases. En este caso se utilizó un analizador industrial, que adicionalmente permite medir el factor de potencia, la potencia reactiva y activa del motor. Con los datos anteriores y despreciando las pérdidas rotacionales se tienen las siguientes ecuaciones:

$$Q_{nl} = \sqrt{S_{nl}^2 - P_{nl}^2} \tag{1.90}$$

Donde Q_{nl} es la potencia reactiva, P_{nl} es la potencia activa medida y S_{nl} se define como:

$$S_{nl} = n_{ph}V_{1nl}I_{1nl} \tag{1.91}$$

siendo n_{ph} el número de fases (3 en este caso), V_{1nl} e I_{1nl} el voltaje y corriente en cada fase respectivamente.

A partir de los resultados anteriores se puede calcular el valor de la reactancia sin carga X_{nl} como:

$$X_{nl} = \frac{Q}{n_{ph}I_{1nl}^2} \tag{1.92}$$

Y

$$X_{nl} = X_{ls} + X_m \tag{1.93}$$

Se realiza también la prueba del rotor bloqueado en la cual el deslizamiento es uno y se aplican voltajes trifásicos balanceados al motor. Puesto que se trata de un motor pequeño (0.5 HP) se realizan las pruebas a frecuencia nominal de 60Hz. Se deben realizar las mismas mediciones que en el caso del rotor sin carga: V_{1lb}, I_{1lb} y P_{1lb}. siendo el voltaje, corriente y potencia activa de cada fase respectivamente. De los valores medidos se define:

$$Q_{bl} = \sqrt{S_{bl}^2 - P_{bl}^2} \tag{1.94}$$

$$S_{bl} = n_{ph} V_{1bl} I_{1bl} \tag{1.95}$$

Con los términos anteriores definidos de forma análoga a 1.90 y 1.91 para el caso del rotor bloqueado.

La reactancia de rotor bloqueado se define:

$$X_{bl} = \frac{Q_{bl}}{n_{ph} I_{1bl}^2} \tag{1.96}$$

La resistencia de rotor bloqueado se define:

$$R_{bl} = \frac{P_{bl}}{n_{ph} I_{1bl}^2} \tag{1.97}$$

Con los parámetros anteriores determinados y las aproximaciones sugeridas en [1] se obtienen las siguientes ecuaciones para los parámetros eléctricos de la máquina AC

$$R_{bl} = r_s + r_r \left(\frac{X_m}{X_{lr} + X_m} \right)^2 \tag{1.98}$$

Siendo R_{bl} la resistencia aparente de rotor bloqueado.

$$X_{bl} = X_{ls} + X_{lr} \left(\frac{X_m}{X_{lr} + X_m} \right) \tag{1.99}$$

Donde X_{bl} es la reactancia de rotor bloqueado aparente a 60Hz.

A partir de 1.98 y 1.99 se determinan las reactancias X_{ls} y X_{lr} i.e. reactancias del rotor y estator.

$$X_{lr} = (X_{bl} - X_{ls}) \left(\frac{X_m}{X_m + X_{ls} - X_{bl}} \right) \tag{1.100}$$

Además las resistencias r_r y r_s del rotor y estator

$$r_r = (R_{bl} - r_s) \left(\frac{X_{lr} + X_m}{X_m} \right)^2 \tag{1.101}$$

Al sustituir 1.93 en 1.100 se obtiene la expresión:

$$X_{lr} = (X_{bl} - X_{ls}) \left(\frac{X_{nl} - X_{ls}}{X_{nl} - X_{bl}} \right) \tag{1.102}$$

En este punto, para obtener el valor de los parámetros del motor se utiliza el estándar IEEE 112 [4] que sugiere empíricamente que $X_{ls} = X_{lr}$ en el caso de motores pequeños. En base a las ecuaciones 1.101, 1.102, 1.93 se obtienen expresiones para los parámetros eléctricos del motor en función de X_{bl}, R_{bl} y X_{nl}.

Para la medición de la carga inercial del motor se utiliza un cronómetro y un tacómetro. Se enciende el motor conectado a una fuente trifásica balanceada. Se apaga la alimentación y se toma el tiempo hasta que el motor se detiene. Se desprecia el transitorio eléctrico y se calcula la inercia efectiva como el cociente entre la variación de velocidad y variación de tiempo.

$$J^{-1} = \frac{\Delta\omega}{\Delta t} \tag{1.103}$$

Se realizaron 20 veces cada una de las pruebas de rotor bloqueado, rotor libre y tiempo de frenado. En la Tabla 3 se muestra la media de los parámetros de motor medidos y su respectiva desviación estándar.

Símbolo	Descripción	Valor
X_{lr}	Reactancia del Rotor	6.37±0.01 Ω
X_{ls}	Reactancia del Estator	6.37±0.01Ω
X_m	Reactacia mútua	207±0.5Ω
r_r	Resistencia rotor	15.20±0.02Ω
r_s	Resistencia estator	11.28±0.01Ω
ω_b	velocidad base	120π rad/s
J	carga inercial	0.015±0.0025kgm^2
B	Factor de viscosidad	0.0125±0.001 Nm/s
P	número de polos	2
X_M	Reactancia efectiva	3.137Ω

Cuadro 3: Parámetros medidos de la máquina de inducción

1.2. Estimadores de estado

Las aplicaciones de control de máquinas de inducción mediante técnicas vectoriales, ya sea por control de vector de campo o control de torque directo requieren de mediciones o estimaciones de calidad de las variables internas del motor, especialmente de flujos magnéticos y de la velocidad del rotor. Si bien existen sensores que permiten medir o estimar indirectamente estas variables; por ejemplo sensores de efecto Hall para medir flujo o sensores del tipo encoder para medir velocidad, estos son costosos y debido a su alto desgaste requieren de constante matenimiento. Además, en la integración de los sensores a la planta se debe considerar que los sensores mencionados tienen menor inmunidad al ruido y menor confiabilidad en la señal lo que requiere tratamiento y acondicionamiento adicional.

En vista de los problemas con los sensores disponibles, se empieza a estudiar y adaptar algoritmos que permitan conocer (estimar) la velocidad del rotor y flujos magnéticos sin necesidad de costosos sensores. En este sentido el filtro Kalman extendido (teoría de estimación clásica) representa una opción adecuada. Existe amplia documentación de su adaptación e implementación en motores reales. Por otra parte el filtro H_∞ (estimación robusta) es un filtro menos estudiado, del cual no existe información de la implementación previa en la estimación de variables de una máquina AC. Incluso la información referente al filtro en sí y su desarrollo teórico es menos extensa que la relacionada a Kalman.

En todo caso, los dos algoritmos de estimación pueden ser acoplados al modelo del motor y además para la estimación requieren mediciones de voltaje y corriente de fases, siendo éstas últimas mediciones sencillas y confiables. En este caso se utilizará un transformador para medir la corriente mediante el voltaje inducido y la medición de voltaje se hará con el acondicionamiento de una muestra tomada directamente de las borneras del motor.

El filtro Kalman ha sido estudiado desde los años 60. Kalman y Bucy empiezan a trabajar en la llamada teoría clásica de estimación [23]. Este filtro es un estimador de estados dinámicos lineales de forma tal que su estimación resulte estadísticamente óptima respecto a cualquier función cuadrática de error. En el caso de sistemas no lineales es posible deducir una versión extendida del filtro aunque se pierden ciertas propiedades de estimación óptima. No obstante, se debe considerar la gran versatilidad que se da al filtro al abarcar sistemas no lineales. Como se detallará en los capítulos siguientes este filtro requiere algunas hipótesis y conocimiento respecto a la naturaleza del ruido tanto del sistema como de las mediciones. Además, en general un modelo exacto del proceso que se estudia no está disponible especialmente a nivel industrial. Por esta razón surge la necesidad de desarrollar filtros más robustos y generalizados.

El filtro H_∞ empieza a ser desarrollado desde 1982 y es conocido por ser un filtro robusto, que considera el peor caso al momento de minimizar una función de error o llamada también función de costo. Uno de los primeros estudios se refiere a los trabajos de Zames y Francis [23]. Desde entonces diversos enfoques se han dado (polinomios, teoría de juegos, ecuaciones de Ricatti, etc). Sin embargo, su implementación y estudios son reducidos debido a la alta carga computacional que requiere y a lo complejo de su sustento matemático en sí. Por ejemplo, son pocos los trabajos y autores que han abordado el tema del filtro H_∞ considerando entradas determinísticas conocidas, como es el caso en el modelo de la máquina de inducción.

1.3. Reseña de objetivos

El presente proyecto tiene como objetivos principales:

1. Presentar el fundamento teórico de los estimadores Filtro Kalman Extendido y Filtro H_∞ Extendido aplicados en la estimación de variables de una máquina de inducción trifásica operando como motor.

2. Realizar una comparación mediante simulación de dos filtros de estimación: Filtro Kalman Extendido y Filtro H_∞ Extendido.

3. Desarrollar la implementación de los dos filtros en VisSim.

4. Realizar una comparación del desempeño real de los dos filtros estimadores.

El primer objetivo se estudia en el capítulo 2. Se hace un estudio extenso de los algoritmos de estimación de cada uno de los dos filtros aplicado en la máquina de inducción. Se presentan los filtros Kalman y H_∞extendidos para sistemas no lineales y sus respectivas características.

El segundo objetivo se estudia en el capítulo 3. Se realizan simulaciones en Simulink para comparar el desempeño de dos versiones diferentes de cada uno de los filtros. A partir de los resultados de simulación se escoge un filtro Kalman extendido con aproximación lineal en el modelo del motor y un filtro H_∞ extendido con la matriz $L(k) = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ en el algoritmo de estimación. La elección se realiza en base al desempeño de los filtros y la carga computacional.

El tercer y cuarto objetivos se describen en el capítulo 4. En la primera parte de este capítulo se abordan las características más importantes del software VisSim y los detalles de la programación de los dos filtros. Se describen los detalles de programación de las operaciones matriciales de los algoritmos mediante sumas y multiplicaciones en punto fijo. En la segunda parte del capítulo 4 se muestran los resultados de la estimación en tiempo real cuando existen perturbaciones de carga. A altas frecuencias de operación el desempeño de los filtros es comparable pero a frecuencias bajas e intermedias el filtro H_∞ presenta mejores resultados.

2. Estimadores de estado

2.1. Filtros H_2/ Algoritmo extendido de Kalman

2.1.1. Filtro Kalman Discreto.

El desarrollo de esta sección se basa en los trabajos de [9, 11, 14]
Supóngase un sistema discreto lineal descrito por las ecuaciones:

$$x_{k+1} = A_k x_k + B_k u_k + w_k \tag{2.1}$$

$$y_k = C_k x_k + v_k \tag{2.2}$$

Donde w_k y v_k son ruidos de proceso y medición respectivamente. Este ruido se modela como blanco, de media cero, no correlacionado y tiene matrices de covarianza Q_k y R_k que satisfacen:

$$E\left[w_k w_j^T\right] = Q_k \delta_{k-j} \tag{2.3}$$

$$E\left[v_k v_j^T\right] = R_k \delta_{k-j} \tag{2.4}$$

$$E\left[v_k w_j^T\right] = 0 \tag{2.5}$$

La función δ_{k-j} se conoce como delta de Kronecker y se define según:

$$\delta_{k-j} = \begin{cases} 1 & k = j \\ 0 & k \neq j \end{cases} \tag{2.6}$$

La meta del problema es calcular x_k en base a la dinámica del sistema y los procesos de ruidos inherentes. Para ello se definen dos estados de estimación, el estado conocido como a posteriori, es decir considerando las k mediciones disponibles

$$\hat{x}_{k|k} = E\left[x_k | y_1, y_2, ..., y_k\right] \tag{2.7}$$

donde el subíndice $k|k$ indica a posteriori y se utiliza $\hat{x}$ para diferenciar el estado estimado del estado real x_k.

En el caso del estado a priori se tienen todas las medidas pero no se incluye la $k - ésima$, es decir:

$$\hat{x}_{k|k-1} = E\left[x_k | y_1, y_2, ..., y_{k-1}\right] \tag{2.8}$$

donde el subíndice $k|k-1$ indica a priori.

Si bien las estimaciones a priori y posteriori son resultados válidos para el sistema, se debe considerar que la estimación a posteriori, por incluir las medidas en el tiempo k, es mejor en la medida que tiene más información.

Antes de proceder al desarrollo del algoritmo de Kalman es necesario definir P_k, matriz que denota la covarianza en el error de la estimación. Específicamente se define una matriz de covarianza de error para la estimación a priori y otra para la estimación a posteriori.

$$P_{k|k-1} = E\left[\left(x_k - \hat{x}_{k|k-1}\right)\left(x_k - \hat{x}_{k|k-1}\right)^T\right] \tag{2.9}$$

$$P_{k|k} = E\left[\left(x_k - \hat{x}_{k|k}\right)\left(x_k - \hat{x}_{k|k}\right)^T\right] \tag{2.10}$$

Por otra parte, se puede demostrar que la ecuación de actualización en tiempo para $\hat{x}$ esta dada por la expresión:

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + Bu_{k-1} \tag{2.11}$$

La covarianza de un estado lineal en tiempo discreto se propaga mediante:

$$P_{k|k-1} = AP_{k-1|k-1}A^T + Q_{k-1} \tag{2.12}$$

La ecuación anterior se conoce como actualización temporal de P.

Mediante la técnica conocida como mínimos cuadrados recursivos es posible calcular la actualización en el tiempo k de la matriz de covarianza y del estado estimado considerando la medición k. El sistema de ecuaciones matriciales que describen este proceso son:

$$K_k = P_{k|k-1}C_k^T\left(C_kP_{k-1}C_k^T + R_k\right)^{-1} \tag{2.13}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\left(y_k - C_k\hat{x}_{k|k-1}\right) \tag{2.14}$$

$$P_{k|k} = P_{k|k-1} - K_kC_kP_{k|k-1} \tag{2.15}$$

Se define $\hat{x}_{k|k-1}$ y $P_{k|k-1}$ como el estado estimado y su covarianza antes que la medición y_k sea procesada y $\hat{x}_{k|k}$ y $P_{k|k}$ como los resultados respectivos una vez que la medición es procesada.

El algoritmo de Kalman para tiempo discreto se resume de la siguiente manera:

- El sistema dinámico satisface las siguientes condiciones:

$$x_{k+1} = A_kx_k + B_ku_k + w_k \tag{2.16}$$

$$y_k = C_kx_k + v_k \tag{2.17}$$

$$E\left[w_kw_j^T\right] = Q_k\delta_{k-j} \tag{2.18}$$

$$E\left[v_kv_j^T\right] = R_k\delta_{k-j} \tag{2.19}$$

$$E\left[v_kw_j^T\right] = 0 \tag{2.20}$$

- Se fijan las siguientes condiciones iniciales:

$$\hat{x}_{0|0} = E\left(x_0\right) \tag{2.21}$$

$$P_{0|0} = E\left[\left(x_0 - \hat{x}_{0|0}\right)\left(x_0 - \hat{x}_{0|0}\right)^T\right] \tag{2.22}$$

- Actualización de variables cada instante k:

$$P_{k|k-1} = AP_{k-1|k-1}A^T + Q_{k-1} \tag{2.23}$$

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + Bu_{k-1} \tag{2.24}$$

$$K_k = P_{k|k-1}C_k^T\left(C_kP_{k-1}C_k^T + R_k\right)^{-1} \tag{2.25}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\left(y_k - C_k\hat{x}_{k|k-1}\right) \tag{2.26}$$

$$P_{k|k} = P_{k|k-1} - K_kC_kP_{k|k-1} \tag{2.27}$$

2.1.2. Propiedades del Filtro Kalman

El error en estimación de este filtro se define como

$$\widetilde{x}_k = x_k - \hat{x}_k \tag{2.28}$$

Debido a que el estado real x_k y estado estimado $\hat{x}_k$ estan afectados por procesos de ruido se dice que las dos son variables aleatorias, de tal manera que el error también resulta aleatorio.

Si el objetivo del filtro es estimar y a la vez minimizar en cada paso la norma ponderada del error de estimación se tendría:

$$\text{mín}\, E\left[\tilde{x}_k^T S_k \tilde{x}_k\right] \tag{2.29}$$

Donde S_k es la matriz de ponderación de la norma. Si S_k es diagonal, la norma a minimizar es

$$N = \sum_{n=1}^{L} S_k(n) E\left[\tilde{x}_k\left(n\right)\right] \tag{2.30}$$

Si los ruidos de proceso w_k y de medición v_k son blancos, de media cero, no correlacionados, entonces el filtro Kalman minimiza la norma descrita en (2.30). En el caso de un sistema lineal el filtro Kalman es la mejor solución al problema propuesto, sin embargo, en el caso de un sistema no lineal se puede mostrar que el filtro Kalman no es la solución más óptima.

2.1.3. Filtro Kalman extendido (Sistemas no lineales)

Como se destacó en el capítulo anterior, en particular en las ecuaciones (1.82) y (1.87) el modelo de la máquina de inducción es no lineal debido a que la matriz A_D contiene el término ω_r en algunas de sus entradas, haciendo que el sistema sea no lineal. En este caso se debe realizar un aproximación lineal del sistema en cada iteración. Al ser un sistema multivariable se debe considerar el carácter matricial de la expansión. Debido a la aproximación lineal que debe aplicarse, el filtro Kalman extendido deja de ser la mejor solución estadística debido a los errores de aproximación del modelo.

En el caso particular de la máquina de inducción, las únicas variables accesibles en este sistema son las corrientes y voltajes en el estator, y la estimación sin sensores debe ser basada en estas variables.

Un Filtro Kalman Extendido usa un algoritmo recursivo que depende de procesos estocásticos y ruido de medición en las salidas. La estimación cambia en el tiempo debido a las correcciones basadas en el error entre las variables estimadas y las variables medidas. El algoritmo para la estimación en cada tiempo de muestreo es el siguiente [10, 19]:

$$X_{k|k-1} = f(x(k), u(k), k) \tag{2.31}$$

$$P_{k|k-1} = \Phi_{k-1} P_{k-1|k-1} \Phi_{k-1}^t + Q_{k-1} \tag{2.32}$$

$$K_k = P_{k|k-1} \overline{H}_k^t \left(\overline{H}_k P_{k|k-1} \overline{H}_k^t + R_k \right)^{-1} \tag{2.33}$$

$$X_{k|k} = X_{k|k-1} + K_k \left(Y_k - \overline{H}_k X_{k|k-1} \right) \tag{2.34}$$

$$P_{k|k} = P_{k|k-1} - K_k \overline{H}_k P_{k|k-1} \tag{2.35}$$

donde Φ se define como sigue:

$$\Phi \approx \left. \frac{\partial f\left(x(k), u(k), k\right)}{\partial x} \right|_{X_{k|k}} \tag{2.36}$$

$$\overline{H}_k = \begin{bmatrix} C_k & D_k' \end{bmatrix} \tag{2.37}$$

$$D_k' = \frac{\partial}{\partial x^*} \left(C_k x_k \right) \tag{2.38}$$

Q_k y R_k son matrices diagonales relacionadas al ruido de proceso y medición respectivamente. Q_k y R_k son definidas diagonales asumiendo que el ruido del sistema no está correlacionado. x^* denota a los estados del filtro extendido, no los estados originales del modelo. En este caso:

$$x^* = \begin{bmatrix} \omega_r & T_l \end{bmatrix}^T \tag{2.39}$$

Nótese que el algoritmo anterior es muy similar al definido por las ecuaciones desde (2.23) a (2.27) donde el cambio más representativo es haber reemplazado la matriz A es por la aproximación lineal definida para Φ y la matriz $\overline{H}$ en lugar de C.

En este caso, un filtro Kalman extendido de sexto orden es usado para estimar el torque de carga y la velocidad del rotor en el motor. El tamaño del vector de estados aumenta debido a que se tratará la velocidad del rotor ω_r y el torque de carga como dos estados más del sistema. A continuación se explica como realizar dicha aproximación

Una versión discreta de la ecuación diferencial (1.80) se requiere para la estimación de velocidad [14], i.e.

$$\omega[k] = z_1\omega[k-1] + \frac{1-z_1}{B}\left(T_{em}\left[k-1\right] - T_l[k-1]\right) \tag{2.40}$$

donde $z_1 = e^{-BT_s/J}$

Por otra parte, el torque de carga es modelado como una perturbación, que se supone aproximadamente constante en el tiempo, dado lugar a la expresión:

$$\frac{d}{dt}Tl = 0 \Leftrightarrow Tl[k] \approx Tl[k-1] \tag{2.41}$$

Considerando las ecuaciones (1.87), (1.88), (1.89), (2.40) y (2.41) se propone el siguiente modelo extendido en espacio de estados para el filtro Kalman extendido:

$$x(k) = \begin{bmatrix} \psi_{qs} & \psi_{ds} & \psi_{qr} & \psi_{dr} & \omega_r & T_l \end{bmatrix}^t \tag{2.42}$$

$$y(k) = \begin{bmatrix} i_{qs} & i_{ds} \end{bmatrix}^t \tag{2.43}$$

$$u(k) = \begin{bmatrix} v_{qs} & v_{ds} \end{bmatrix}^t \tag{2.44}$$

$$f(x(k),u(k),k) = \begin{bmatrix} A_1x_1 + A_3x_3 + A_4x_4 + B_1u_1 \\ A_6x_2 + A_7x_3 + A_8x_4 + B_4u_2 \\ A_9x_1 + A_{10}x_2 + A_{11}x_3 + A_{12}x_4 + B_5u_1 \\ A_{13}x_1 + A_{14}x_2 + A_{15}x_3 + A_{16}x_4 + B_8u_2 \\ \Gamma c_{s2}x_3x_2 - \Gamma c_{s2}x_1x_4 + z_1x_5 - \frac{1-z_1}{B}x_6 \\ x_6 \end{bmatrix} \tag{2.45}$$

donde
$\Gamma = \frac{3P}{4\omega_b}\left(\frac{1-z_1}{J}\right)$

Además, se muestran las matrices Φ y $\overline{H}$ requeridas en el algoritmo del filtro:

$$\Phi = \begin{bmatrix} A_1 & 0 & A_3 & A_4 & f_1 & 0 \\ 0 & A_6 & A_7 & A_8 & f_2 & 0 \\ A_9 & A_{10} & A_{11} & A_{12} & f_3 & 0 \\ A_{13} & A_{14} & A_{15} & A_{16} & f_4 & 0 \\ -\Gamma c_{s2}x_4 & \Gamma c_{s2}x_3 & \Gamma c_{s2}x_2 & -\Gamma cs_2x_1 & z_1 & -\frac{1-z_1}{B} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.46}$$

$f_1 = \frac{a_{s2}}{2} T_s^2 x_4$
$f_2 = -\frac{a_{s2}}{2} T_s^2 x_3$
$f_3 = a_{r2} \frac{T_s^2}{2} x_2 - T_s^2 x_3 x_5 + \left(T_s + T_s^2 a_{r1}\right) x_4$
$f_4 = -a_{r2} \frac{T_s}{2}^2 x_1 - \left(T_s + T_s^2 a_{r1}\right) x_3 - T_s^2 x_4 x_5$

$$\overline{H} = \begin{bmatrix} c_{s1} & 0 & c_{s2} & 0 & 0 & 0 \\ 0 & c_{s1} & 0 & c_{s2} & 0 & 0 \end{bmatrix} \tag{2.47}$$

2.2. Filtros $\mathbf{H}_\infty$

2.2.1. Filtro H_∞ Discreto.

El siguiente desarrollo se basa en la deducción presentada en [7, 21].
Supóngase un sistema discreto lineal descrito por las ecuaciones:

$$x_{k+1} = A_k x_k + B_k u_k + w_k \tag{2.48}$$

$$y_k = C_k x_k + v_k \tag{2.49}$$

$$z_k = L_k x_k \tag{2.50}$$

Donde w_k y v_k son ruidos de proceso y medición respectivamente. Ademász_k es una combinación lineal de los estados x_k mediante la matriz L_k. Si $z_k = x_k$ entonces $L_k = I$. Sin embargo, como se verá en este trabajo, en ocasiones existen ciertas combinaciones lineales de x_k que son de mayor interés por lo que L_k adquiere un valor diferente de la identidad. Más adelante se detallará el papel que desempeña la matriz L_k sobretodo en lo que se refiere al algoritmo de computación y la función de costo a minimizar.

En caso de tener un modelo como el descrito por las ecuaciones (2.48), (2.49) y (2.50) la función de costo que se busca minimizar mediante la teoría de juegos basada en las $N-1$ mediciones es:

$$J_a = \frac{\sum_{k=0}^{N-1} \|z_k - \hat{z_k}\|_{S_k}^2}{\|x_0 - \hat{x_0}\|_{P_0^{-1}}^2 + \sum_{k=0}^{N-1} \left(\|w_k\|_{Q_k^{-1}}^2 + \|v_k\|_{R_k^{-1}}^2 \right)} \tag{2.51}$$

La idea de teoría de juegos nace del hecho que el diseñador intenta encontrar $\hat{z_k}$ que minimiza J_a. Por otra parte, la dinámica del sistema (visto como adversario) intenta encontrar las perturbaciones w_k, v_k y x_0 tal que maximizen J_a. Por naturaleza el sistema intenta maximizar $(z_k - \hat{z_k})$. Sin embargo, si bien $(z_k - \hat{z_k})$ aumenta con w_k, v_k y x_0 tendiendo a infinito, se debe tener en cuenta que J_a no lo hace necesariamente (es un cociente de las variables nombradas), por lo que la interacción del sistema se vuelve compleja al considerar la relación existente entre las variables. De esta manera se define un "juego" complejo de maximización y minimización entre el diseñador y el sistema. Por la razón anterior este filtro también es conocido como minimax.

P_0, Q_k , R_k y S_k mostradas en (2.51) son matrices simétricas definidas positivas y que pueden ser escogidas por el diseñador en función de los requerimientos del problema. Por

ejemplo si se requiere mayor precisión en la estimación del segundo elemento del vector z_k se debe elegir la entrada $S_k(2,2)$ mayor en relación a los demás términos de la matriz S_k. De igual manera, si el conocimiento previo del proceso indica que alguna entrada de w_k es pequeña, la entrada en la diagonal de Q_k correspondiente a la entrada de w_k debe ser pequeña también.

No es posible hallar un mínimo total para (2.51) pero se puede acotar la función de forma tal que:

$$J_a < \gamma^2 \tag{2.52}$$

siendo γ un número real.
Es decir:

$$J = -\gamma^2 \left\| x_0 - \hat{x_0} \right\|^2_{P_0^{-1}} + \sum_{k=0}^{N-1} \left[\left\| z_k - \hat{z_k} \right\|^2_{S_k} - \gamma^2 \left(\left\| w_k \right\|^2_{Q_k^{-1}} + \left\| v_k \right\|^2_{R_k^{-1}} \right) \right] < 1 \tag{2.53}$$

En este escenario el problema se reduce a hallar J^* tal que:

$$J^* = \min_{\hat{z_k}} \max_{w_k, v_k, x_0} J \tag{2.54}$$

Como $z_k = L_k x_k$ se elige:

$$\hat{z_k} = L_k \hat{x_k} \tag{2.55}$$

Finalmente el problema se reduce a hallar $\hat{x_k}$ que minimiza J

$$J^* = \min_{\hat{x_k}} \max_{w_k, v_k, x_0} J \tag{2.56}$$

De (2.49) se puede escribir:

$$v_k = y_k - C_k x_k \tag{2.57}$$

y por lo tanto:

$$\left\| v_k \right\|^2_{R_k^{-1}} = \left\| y_k - H_k x_k \right\|^2_{R_k^{-1}} \tag{2.58}$$

De igual manera de (2.50) y (2.55) se puede deducir:

$$\left\| z_k - \hat{z_k} \right\|^2_{Sk} = \left\| x_k - \hat{x_k} \right\|^2_{\tilde{S}_k} \tag{2.59}$$

con

$$\tilde{S_k} = L_k^T S_k L_k \tag{2.60}$$

En base a (2.58) y (2.60) es posible describir J (2.53) como sigue:

$$J = -\gamma^2 \left\| x_0 - \hat{x_0} \right\|^2_{P_0^{-1}} + \sum_{k=0}^{N-1} \left[\left\| x_k - \hat{x_k} \right\|^2_{\hat{S}_k} - \gamma^2 \left(\left\| w_k \right\|^2_{Q_k^{-1}} + \left\| y_k - H_k x_k \right\|^2_{R_k^{-1}} \right) \right] \tag{2.61}$$

De esta forma la función de costo depende únicamente de los estados x_k y de las salidas y_k. Se eliminó la dependencia con v_k y z_k.

Se puede reescribir la función de costo de la siguiente forma:

$$J = \psi\left(x_0\right) + \sum_{k=0}^{N-1} \mathcal{L}_k \tag{2.62}$$

Para poder deducir las ecuaciones necesarias del algoritmo se consideran dos casos o escenarios. El primero se refiere a la maximización de (2.62) con respecto a x_0 y w_k (maximización dada por el sistema) pero considerando que $x_{k+1} = A_k x_k + B_k u_k$. El Hamiltoniano para este problema es:

$$\mathcal{H}_k = \mathcal{L}_k + 2\gamma^2 \theta_{k+1}^T \left(A_k x_k + B_k u_k + w_k\right) \tag{2.63}$$

$2\gamma^2\theta_{k+1}$ es un multiplicador de Lagrange variante en el tiempo que se debe calcular en cada ciclo o iteración. Al resolver este problema de optimización se obtienen las siguientes ecuaciones necesarias para el algoritmo de estimación:

$$x_0 = \hat{x_0} + P_0\theta_0 \tag{2.64}$$

$$w_k = Q_k\theta_{k+1} \tag{2.65}$$

$$\theta_N = 0 \tag{2.66}$$

$$K_k = P_k\left[I - \frac{1}{\gamma^2}\tilde{S}_k P_k + C_k^T R_k^{-1} C_k P_k\right]^{-1} C_k^T R_k^{-1} \tag{2.67}$$

$$P_{k+1} = A_k P_k\left[I - \frac{1}{\gamma^2}\tilde{S}_k P_k + C_k R_k^{-1} C_k P_k\right]^{-1} A_k^T + Q_k \tag{2.68}$$

Ahora se considera la minimización de J (2.62) con respecto a $\hat{x_k}$ y y_k. Para resolver este problema se asume que ya resolvió el problema anterior, es decir se maximizó la función con respecto a x_0 y w_k. Bajo dicha hipótesis y utilizando nuevamente $x_{k+1} = A_k x_k + B_k u_k$ como condición de restricción. De este nuevo procedimiento se logra demostrar un resultado muy similar al del filtro Kalman:

$$\hat{x}_{k+1} = A_k\hat{x}_k + B_k u_k + A_k K_k\left(y_k - C_k\hat{x}_k\right) \tag{2.69}$$

Además se desprende una condición necesaria para poder resolver el problema. En cada instante de tiempo k se debe cumplir:

$$P_k^{-1} - \frac{1}{\gamma^2}\tilde{S}_k + C_k^T R_k^{-1} C_k > 0 \tag{2.70}$$

El algoritmo H_∞ para tiempo discreto se resume de la siguiente manera:

- El sistema dinámico satisface las siguientes condiciones:

$$x_{k+1} = A_k x_k + B_k u_k + w_k \tag{2.71}$$

$$y_k = C_k x_k + v_k \tag{2.72}$$

$$z_k = L_k x_k \tag{2.73}$$

- Se fijan las siguientes condiciones iniciales:

$$\hat{x}_{0|0} = E\left(x_0\right) \tag{2.74}$$

$$P_{0|0} = E\left[\left(x_0 - \hat{x}_{0|0}\right)\left(x_0 - \hat{x}_{0|0}\right)^T\right] \tag{2.75}$$

- Actualización de variables cada instante k:

$$\tilde{S_k} = L_k^T S_k L_k \tag{2.76}$$

$$K_k = P_k\left[I - \frac{1}{\gamma^2}\tilde{S}_k P_k + C_k^T R_k^{-1} C_k P_k\right]^{-1} C_k^T R_k^{-1} \tag{2.77}$$

$$\hat{x}_{k+1} = A_k \hat{x}_k + B_k u_k + A_k K_k\left(y_k - C_k \hat{x}_k\right) \tag{2.78}$$

$$P_{k+1} = A_k P_k\left[I - \frac{1}{\gamma^2}\tilde{S}_k P_k + C_k R_k^{-1} C_k P_k\right]^{-1} A_k^T + Q_k \tag{2.79}$$

Nótese que el algoritmo presentado para el filtro H_∞ es similar al algoritmo para el filtro Kalman, sin embargo existe la matriz $\tilde{S_k}$ que deja un parámetro de diseño adicional que puede explotarse a la hora de sintonizar el estimador. Adicionalmente, el algoritmo depende del parámetro γ^2, que no es más que una cota para definir el rendimiento del sistema. En términos energéticos se dice que γ^2 define la máxima energía que tiene la función de costo J.

2.2.2. Propiedades del filtro H_∞

En el filtro Kalman, la naturaleza del sistema de cierta forma se desprecia pues se asume ruido blanco y a partir de ello se construye la teoría y el algoritmo. Sin embargo, en el filtro H_∞ se asume que el sistema trata de degradar la estimación realizada en el mayor grado posible (si bien en la realidad esto puede no ser cierto, es el supuesto del cual se parte en la deducción del algoritmo).

En el filtro Kalman las matrices Q_k, R_k y P_0 son escogidas por el diseñador del filtro, éstas deben ser diagonales para satisfacer la condición de no correlación entre variables. Por otra parte, en el filtro H_∞ las matrices Q_k, R_k y P_0 son elegidas por el diseñador pero éstas se

basan en el conocimiento a priori que se tenga acerca de los ruidos w_k y v_k y del estado inicial de la estimación $(x_0 - \hat{x_0})$. No existe restricción en cuando forma de dichas matrices, lo cual hace el diseño del filtro mucho más flexible y adaptable.

Como se puede notar, de la matriz $\tilde{S_k}$ es evidente que la forma y el desempeño del filtro H_∞ depende de la combinación lineal de x_k que se utiliza dada por el valor de L_k dado en el modelo. En el caso del filtro Kalman el valor es el mismo sin importar el valor de L_k que se escoja. En el caso H_∞ se pueden lograr mejores resultados al considerar que en muchas ocasiones el objetivo final de la estimación no es un estado en sí sino una combinación lineal de los estados del modelo. De esta manera se puede adaptar el filtro a una necesidad específica y mejorar su desempeño.

La condición dada en (2.70) supone cierta limitante al momento del diseño. Si bien el rendimiento global del filtro esta acotado por γ^2 y a menor valor de este parámetro menor error, se debe considerar que al disminuir γ^2 su inverso aumenta y hace más complicado satisfacer la condición definida en (2.70). De esta forma $\tilde{S_k}$ debería ser también pequeña para mantener la consistencia y asegurar la existencia de una solución. En general tanto L_k como S_k son constantes por lo que $\tilde{S_k}$ también lo es. En este sentido resulta relativamente sencillo hallar los valores de γ y $\tilde{S_k}$ que mejor se adaptan al modelo, ya que como se mencionó anteriormente, mientras mayor son las entradas en la diagonal de la matriz $\tilde{S_k}$ más precisa es la estimación del estado relacionado (mismo índice) a dicha entrada.

Si se considera $\gamma = \infty$ el filtro H_∞ se reduce al filtro Kalman ya detallado anteriormente. Desde esa óptica, el filtro Kalman no es más que un filtro minimax cuando el límite de rendimiento se fija en infinito, siendo de esta forma un caso particular del filtro H_∞. Por esta razón existen autores como [6, 7] que mencionan al filtro H_∞ como una versión robusta del filtro Kalman clásico. La diferencia está en su modelamiento que le agrega tolerancia a la dinámica del proceso o ruido no modelado.

En la literatura es posible encontrar diversas versiones del algoritmo de estimación H_∞. Las más comunes son el algoritmo presentado por Lewis [9] que se basa en la ecuación de Ricatti, el algoritmo de Hassibi [8] que parte de un estudio de desigualdades matriciales indefinidas y el estudio de Simon [7] que considera la teoría de juegos y el uso de multiplicadores de Lagrange para hallar la solución óptima. En este proyecto se presenta el algoritmo de [7] por se el de más fácil comprensión ya que utiliza técnicas de cálculo vectorial para su deducción y porque computacionalmente es más sencillo respecto al de Lewis. En el caso de Simon solo hay una inversión de matrices en cada iteración. El algoritmo de Lewis requiere de la inversión de dos matrices, lo cual complica la implementación real del filtro. Por otra parte, el algoritmo de Hassibi se presenta más complicado de entender ya que requiere de una serie de lemas previos para poder demostrar que la solución del filtro depende de la solución de una desigualdad matricial. Computacionalmente hallar una solución a dicha desigualdad requeriría del uso de lazos for o while que aumentan significativamente el tiempo necesario para ejecutar el algoritmo por lo que el filtro puede verse afectado en su desempeño en tiempo real.

2.2.3. Filtro H_∞ extendido (Sistemas no lineales)

Como ya se discutió en la sección referente al filtro Kalman, el modelo de la máquina de inducción es no lineal. La matriz A_D contiene el término ω_r en algunas de sus entradas,

haciendo que el sistema sea no lineal al momento de considerar ω_r y T_l como estados también. En este caso se debe realizar un aproximación lineal del sistema en cada iteración. Al ser un sistema multivariable se debe considerar el carácter matricial de la expansión. Si bien en la expansión en Taylor se producen errores por la aproximación misma, el rendimiento del filtro no se ve afectado de forma sustancial, dado que dichos errores de aproximación pueden ser modelados como ruido (ya que no existe ninguna limitación respecto a su naturaleza) como es el caso en el filtro Kalman. Así también, la elección de la matriz L_k es sumamente importante ya que a más de definir el filtro en sí y la función de costo del sistema juega un papel fundamental en la determinación de la carga computacional requerida (en implementación este detalle es crítico).

Un filtro H_∞ extendido usa un algoritmo recursivo que depende de procesos estocásticos y del error (diferencia) entre las salidas estimadas del sistema y las salidas medidas en el mismo. La estimación cambia en el tiempo debido a las correcciones o cambios que se dan en la ganancia (matrizK_k) que garantizan los mejores resultados posibles. El algoritmo para la estimación en cada tiempo de muestreo es el siguiente [6]:

$$X_{k|k-1} = f(x(k), u(k), k) \tag{2.80}$$

$$P_{k|k-1} = \Phi_{k-1} P_{k-1|k-1} \Phi_{k-1}^T + G_{k-1} \tag{2.81}$$

$$K_k = P_{k|k-1} \overline{H}_k^T \left(\overline{H}_k P_{k|k-1} \overline{H}_k^T + I_{2\times 2} \right)^{-1} \tag{2.82}$$

$$X_{k|k} = X_{k|k-1} + K_k \left(Y_k - \overline{H}_k X_{k|k-1} \right) \tag{2.83}$$

$$P_{k|k} = P_{k|k-1} - P_{k|k-1} \begin{bmatrix} \overline{H}_k \\ L_k \end{bmatrix} Rw_k^{-1} \begin{bmatrix} \overline{H}_k \\ L_k \end{bmatrix}^T P_{k|k-1} \tag{2.84}$$

$$Rw_k = \begin{bmatrix} Rw_1 & Rw_2 \\ Rw_3 & Rw_4 \end{bmatrix} \tag{2.85}$$

$$Rw_1 = I_{2\times 2} + \overline{H}_k P_{k|k-1} \overline{H}_k^T$$

$$Rw_1 = \overline{H}_k P_{k|k-1} L_k^T$$

$$Rw_3 = L_k P_{k|k-1} \overline{H}_k^T$$

$$Rw_4 = -\gamma^2 I_{2\times 2} + L_k P_{k|k-1} L_k^T$$

y Φ se define como:

$$\Phi \approx \left. \frac{\partial f\left(x(k), u(k), k\right)}{\partial x} \right|_{X(k|k)} \tag{2.86}$$

$$\overline{H}_k = \begin{bmatrix} C_k & D'_k \end{bmatrix} \tag{2.87}$$

$$D'_k = \frac{\partial}{\partial x^*}\left(C_k x_k\right) \tag{2.88}$$

x^* denota a los estados del filtro extendido, no los estados originales del modelo. En este caso:

$$x^* = \begin{bmatrix} \omega_r & T_l \end{bmatrix}^T \tag{2.89}$$

El tamaño del vector de estados aumenta debido a que se tratará la velocidad del rotor ω_r y el torque de carga T_l como dos estados más del sistema. Esto además hace que el modelo en espacio de estados para motor sea no lineal.

Una versión discreta de la ecuación diferencial (1.80) se requiere para la estimación de velocidad [14], i.e.

$$\omega[k] = z_1\omega[k-1] + \frac{1-z_1}{B}\left(T_{em}\left[k-1\right] - T_l[k-1]\right) \tag{2.90}$$

donde $z_1 = e^{-BT_s/J}$

Por otra parte, el torque de carga es modelado como una perturbación, que se supone aproximadamente constante en el tiempo, dado lugar a la expresión:

$$\frac{d}{dt}Tl = 0 \Leftrightarrow Tl[k] \approx Tl[k-1] \tag{2.91}$$

Considerando las ecuaciones (1.87), (1.88), (1.89), (2.90) y (2.91) se propone el siguiente modelo extendido en espacio de estados para el filtro H_∞ extendido:

$$x(k) = \begin{bmatrix} \psi_{qs} & \psi_{ds} & \psi_{qr} & \psi_{dr} & \omega_r & T_l \end{bmatrix}^t \tag{2.92}$$

$$y(k) = \begin{bmatrix} i_{qs} & i_{ds} \end{bmatrix}^t \tag{2.93}$$

$$u(k) = \begin{bmatrix} v_{qs} & v_{ds} \end{bmatrix}^t \tag{2.94}$$

$$z(k) = L(k)x(k) \tag{2.95}$$

$$f(x(k), u(k), k) \tag{2.96}$$

$$= \begin{bmatrix} A_1x_1 + A_3x_3 + A_4x_4 + B_1u_1 \\ A_6x_2 + A_7x_3 + A_8x_4 + B_4u_2 \\ A_9x_1 + A_{10}x_2 + A_{11}x_3 + A_{12}x_4 + B_5u_1 \\ A_{13}x_1 + A_{14}x_2 + A_{15}x_3 + A_{16}x_4 + B_8u_2 \\ \Gamma c_{s2}x_3x_2 - \Gamma c_{s2}x_1x_4 + z_1x_5 - \frac{1-z_1}{B}x_6 \\ x_6 \end{bmatrix} \tag{2.97}$$

donde $\Gamma = \frac{3P}{4\omega_b}\left(\frac{1-z_1}{J}\right)$

En este filtro se considerarán dos casos $L(k) = I_{6\times6}$ y $L(k) = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$. Esto debido a que al elegir $L(k)$ como la matriz identidad se tiene una carga computacional extra sumamente alta dado que R_w definida en (2.85) sería de dimensiones 8x8, y el calculo de su inversa resulta complicada y pesada a nivel computacional, sobretodo al momento de la implementación física. Por otra parte con $L(k) = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ la matriz R_w es 4x4, y su inversa puede ser calculada con mayor facilidad y rapidez. Además, puesto que las dos variables cuya estimación son el centro de atención de este trabajo son la velocidad del rotor y el torque de carga, al elegir $L(k) = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ se hace énfasis en estas variables. Mediante simulación se considerará los dos casos descritos y en base a los resultados obtenidos se eligirá si es necesario considerar $L(k) = I_{6\times6}$ al momento de la implementación.

Finalmente se muestran las matrices Φ y $\overline{H}$ requeridas en el algoritmo del filtro:

$$\Phi = \begin{bmatrix} A_1 & 0 & A_3 & A_4 & f_1 & 0 \\ 0 & A_6 & A_7 & A_8 & f_2 & 0 \\ A_9 & A_{10} & A_{11} & A_{12} & f_3 & 0 \\ A_{13} & A_{14} & A_{15} & A_{16} & f_4 & 0 \\ -\Gamma c_{s2}x_4 & \Gamma c_{s2}x_3 & \Gamma c_{s2}x_2 & -\Gamma cs_2x_1 & z_1 & -\frac{1-z_1}{B} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.98}$$

$f_1 = \frac{a_{s2}}{2}T_s^2x_4$
$f_2 = -\frac{a_{s2}}{2}T_s^2x_3$
$f_3 = a_{r2}\frac{T_s^2}{2}x_2 - T_s^2x_3x_5 + \left(T_s + T_s^2a_{r1}\right)x_4$
$f_4 = -a_{r2}\frac{T_s}{2}^2x_1 - \left(T_s + T_s^2a_{r1}\right)x_3 - T_s^2x_4x_5$

$$H = \begin{bmatrix} c_{s1} & 0 & c_{s2} & 0 & 0 & 0 \\ 0 & c_{s1} & 0 & c_{s2} & 0 & 0 \end{bmatrix} \tag{2.99}$$

3. Desarrollo de algoritmos de estimación

En este capítulo se muestran los resultados de la simulación de los filtros Kalman y H_∞ bajo diferentes condiciones de trabajo. Las gráficas que se muestran corresponden a las variables estimadas, es decir flujos $qd0$ en el estator y rotor, corrientes $qd0$ en el estator, velocidad del rotor y torque de carga. En las gráficas presentadas se considera el comportamiento del filtro en el arranque, es decir en el estado transitorio. Luego, se considera el comportamiento en el estado estable sin ningún tipo de perturbación. A continuación se presentan los resultados en el caso de existir perturbaciones (variaciones en el torque de carga). Finalmente se presenta una vista detallada (ampliada) del comportamiento de las variables ya mencionadas durante una perturbación para tener una idea visual clara de la respuesta del filtro a este tipo de factores externos.

Para la simulación se utilizaron los parámetros del motor de 3 HP descritos por Krause [2]. Todas las reactancias presentadas en la Tabla 4 están referidas a 60Hz.

Símbolo	Descripción	Valor
X_{lr}	Reactancia del Rotor	0.754 Ω
X_{ls}	Reactancia del Estator	0.754Ω
X_m	Reactacia mútua	26.13Ω
r_r	Resistencia rotor	0.816Ω
r_s	Resistencia estator	0.435Ω
ω_b	velocidad base	120π rad/s
J	carga inercial	$0.089 kgm^2$
B	Factor de viscosidad	0.085 Nm/s
P	número de polos	4
X_M	Reactancia efectiva	0.372Ω

Cuadro 4: Párametros del motor utilizados para simulación

En la Figura 3.1 se observa el diagrama de bloques construido en Simulink para la simulación. Este diagrama consta de bloques cuya operación es definida por el usuario principalmente. La ventaja de utilizar este tipo de bloques es que se puede ingresar directamente archivos de código programado en C o como funciones m de Matlab en lugar de utilizar esquemas de bloques que pueden resultar complicados y complejos debido a la cantidad de operaciones y pasos que se deben realizar tanto en la simulación del modelo continuo del motor como en la simulación de los filtros en tiempo discreto.

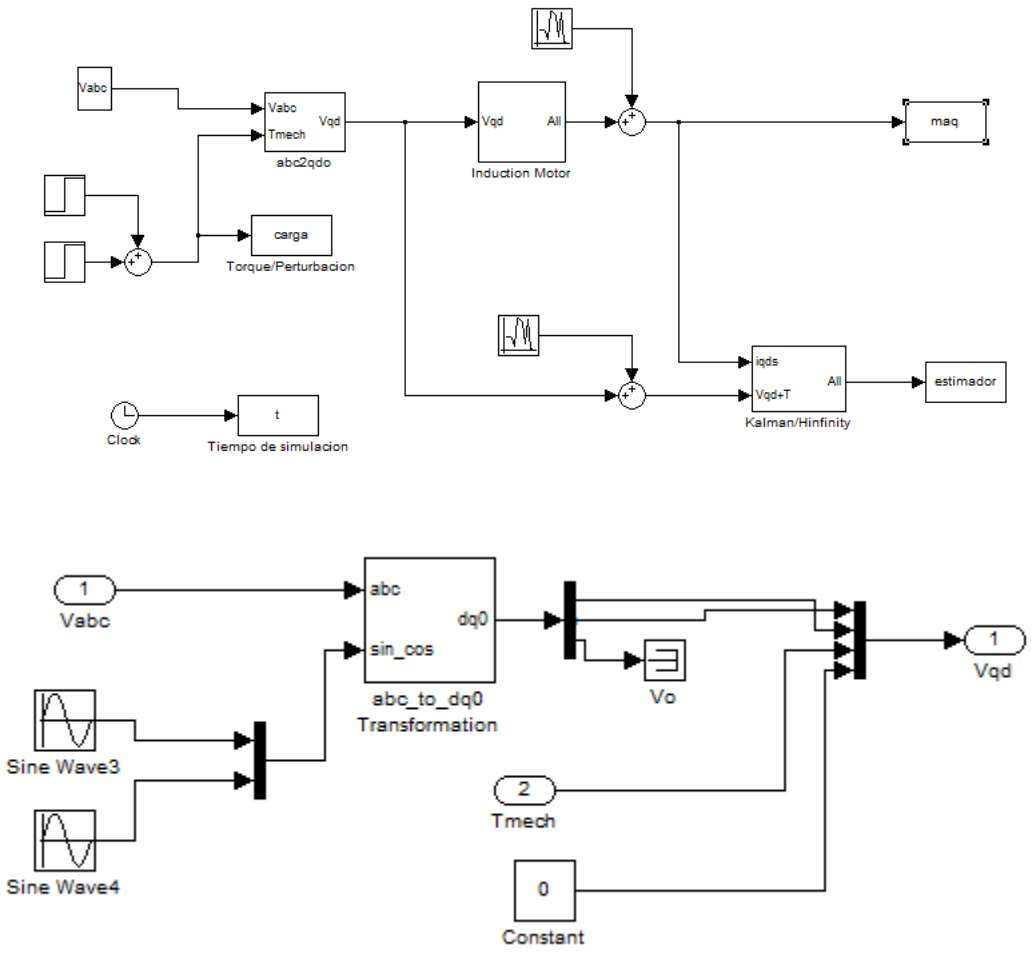


Figura 3.1: Esquema de bloques en Simulink para simulación de la máquina de inducción y filtros Kalman extendido y H_∞

El modelo del motor esta programado en un código C mientras que los filtros están programados en archivos .m de Matlab. Además se utilizan bloques internos de Simulink que permiten realizar operaciones genéricas como enrutamiento de las señales (multiplexores y demultiplexores), transformación de eje de referencia (al eje $qd0$ estacionario $\omega = 0$) y almacenamiento de datos en el espacio de trabajo de Matlab para su posterior procesamiento y presentación. Los archivos de Simulink, código del modelo de la máquina de inducción en C y la programación de los filtros están disponibles en la sección Anexos al final de este trabajo.

Como pertubación se utilizaron señales del tipo escalón unitario ya que la respuesta de un sistema (en el área de control) por lo general se estudia mendiante el uso de este tipo de señales de prueba. Sin embargo, se debe anotar que el torque de carga al ser una variable mecánica tiene una dinámica más lenta.

3.1. Simulación filtro H_2/Kalman extendido.

A continuación se muestran los resultados de la simulación en el caso del Filtro Kalman Extendido. Además de los escenarios descritos en la introducción de este capítulo se debe tener en cuenta que se realizaron dos tipos de simulaciones, una basada en un modelo lineal y otra en un modelo cuadrático. En la Sección 1, ecuación (1.87), cuando se discutió la deducción del modelo discreto de la máquina de inducción a partir del modelo continuo mediante una expasión en series de Taylor en términos de T_s ya se habló brevemente acerca de este detalle. Se mencionó brevemente la utilidad o no de conservar los términos cuadráticos en T_s. Allí se estimó cualitativamente que a medida que T_s es pequeño (típicamente del orden de 10^{-4}), incluir términos cuadráticos despreciables (del orden de 10^{-8}) no debería afectar el rendimiento del filtro e incluso el nivel de exactitud que se gana con la inclusión de estos términos no justifica la cantidad extra de computos que se requieren, especialmente a nivel implementación física. Por esta razón, en esta sección también se espera comparar los resultados de las simulaciones en base al modelo de aproximación lineal en T_s y al modelo de aproximación cuadrática en T_s para así determinar si existen diferencias notables en los resultados. En caso de no haber diferencias sustanciales se optará por el modelo lineal a la hora de la implementación (se reduce el número de cálculos, lo cual puede ser un factor crítico).

Para finalizar los comentarios previos a la simulación se muestran los valores de las matrices Q y R que se utilizaron en los filtros (valores hallados experimentalmente mediante multiples simulaciones de prueba). El tiempo de muestreo utilizado fue $T_s = 200\mu s$.

$$Q = \begin{bmatrix} 0{,}2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0{,}2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0{,}02 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0{,}02 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0{,}1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0{,}001 \end{bmatrix} \tag{3.1}$$

$$R = \begin{bmatrix} 0{,}45 & 0 \\ 0 & 0{,}45 \end{bmatrix} \tag{3.2}$$

3.1.1. Aproximación lineal en T_s-Arranque

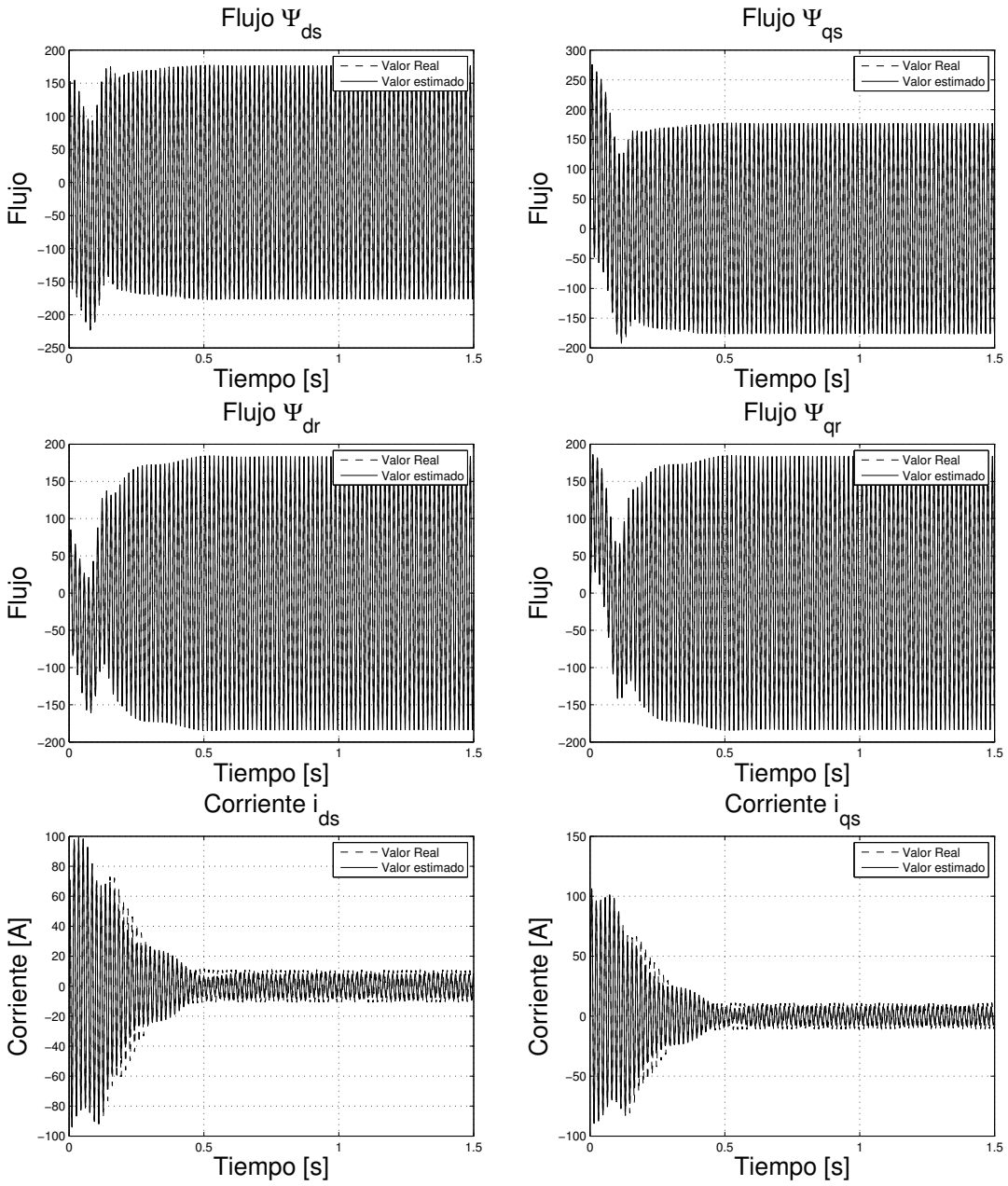


Figura 3.2: Filtro Kalman en aproximación lineal al arranque.

Se puede observar una correcta estimación de los flujos y corrientes (combinaciones lineales de los flujos) en el arranque del motor, es decir en el estado transitorio.

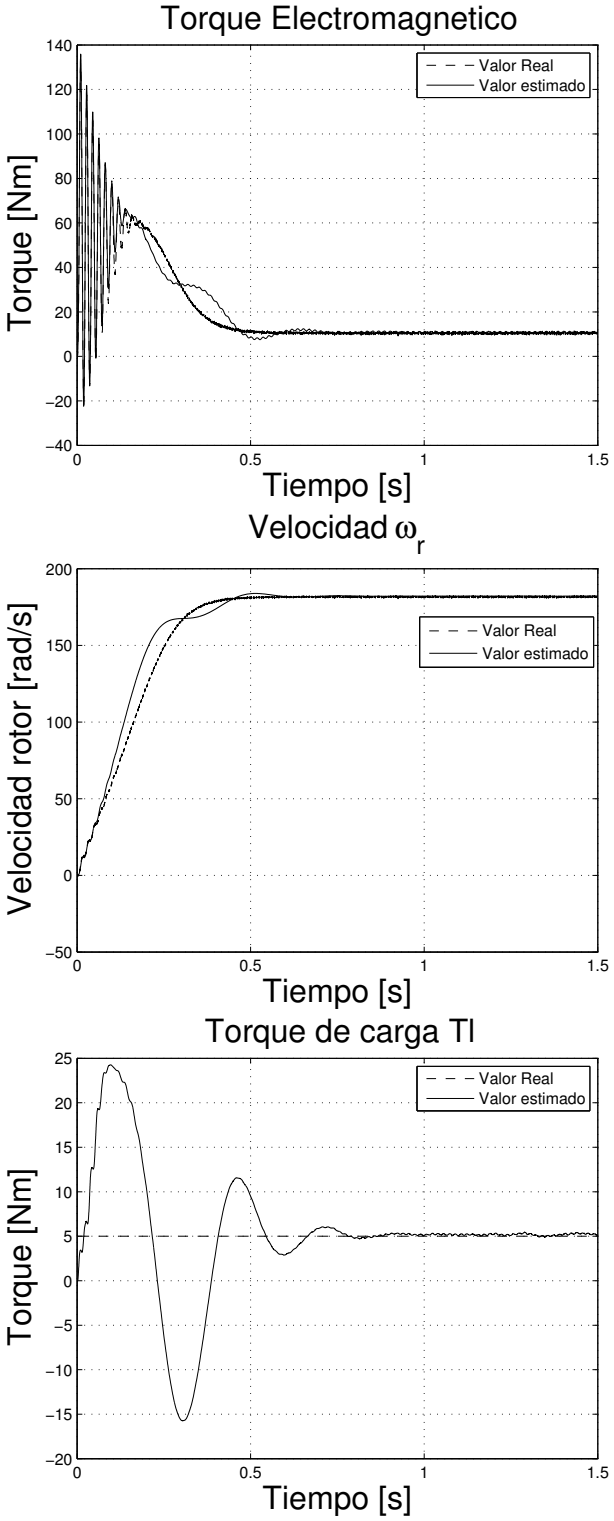


Figura 3.3: Filtro Kalman en aproximación lineal al arranque

Se observa que existen oscilaciones en la estimación durante el estado transitorio dando

lugar a diferencias entre los valores estimados y los valores reales. Sin embargo, se observa que al cesar el transitorio la estimación y el valor real tienden hacia un mismo valor. En el caso del torque de carga se observan grandes oscilaciones iniciales.

3.1.2. Aproximación lineal en T_s-Estado estable sin perturbación

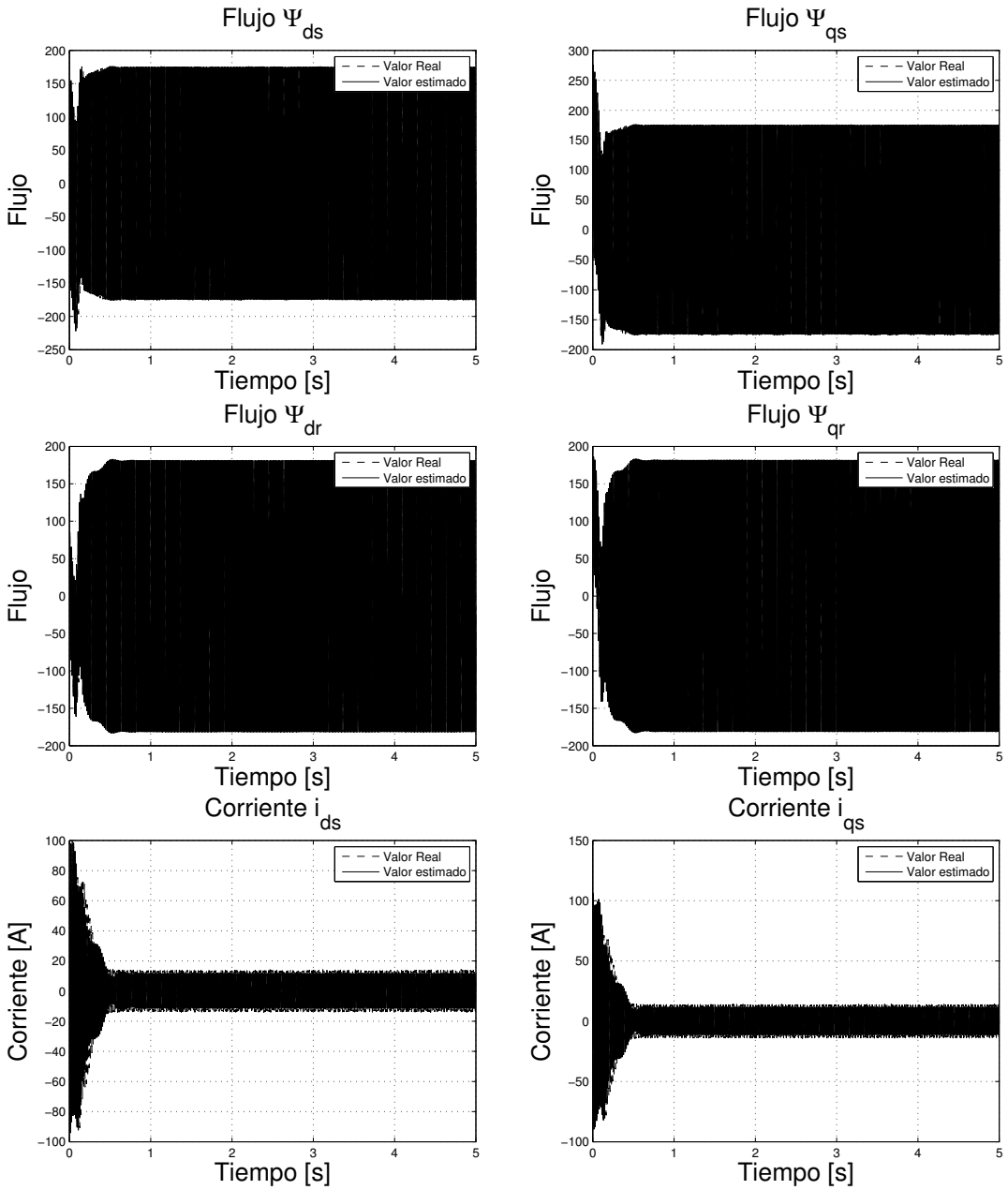


Figura 3.4: Filtro Kalman en aproximación lineal en estado estable sin perturbación

En la gráfica de flujos y corrientes si bien no se puede apreciar con claridad el comportamiento de las variables en estado estable sin perturbación, la no presencia de picos significativos o comportamientos irregulares hace preveer que la estimación esta acorde a lo que se espera del filtro.

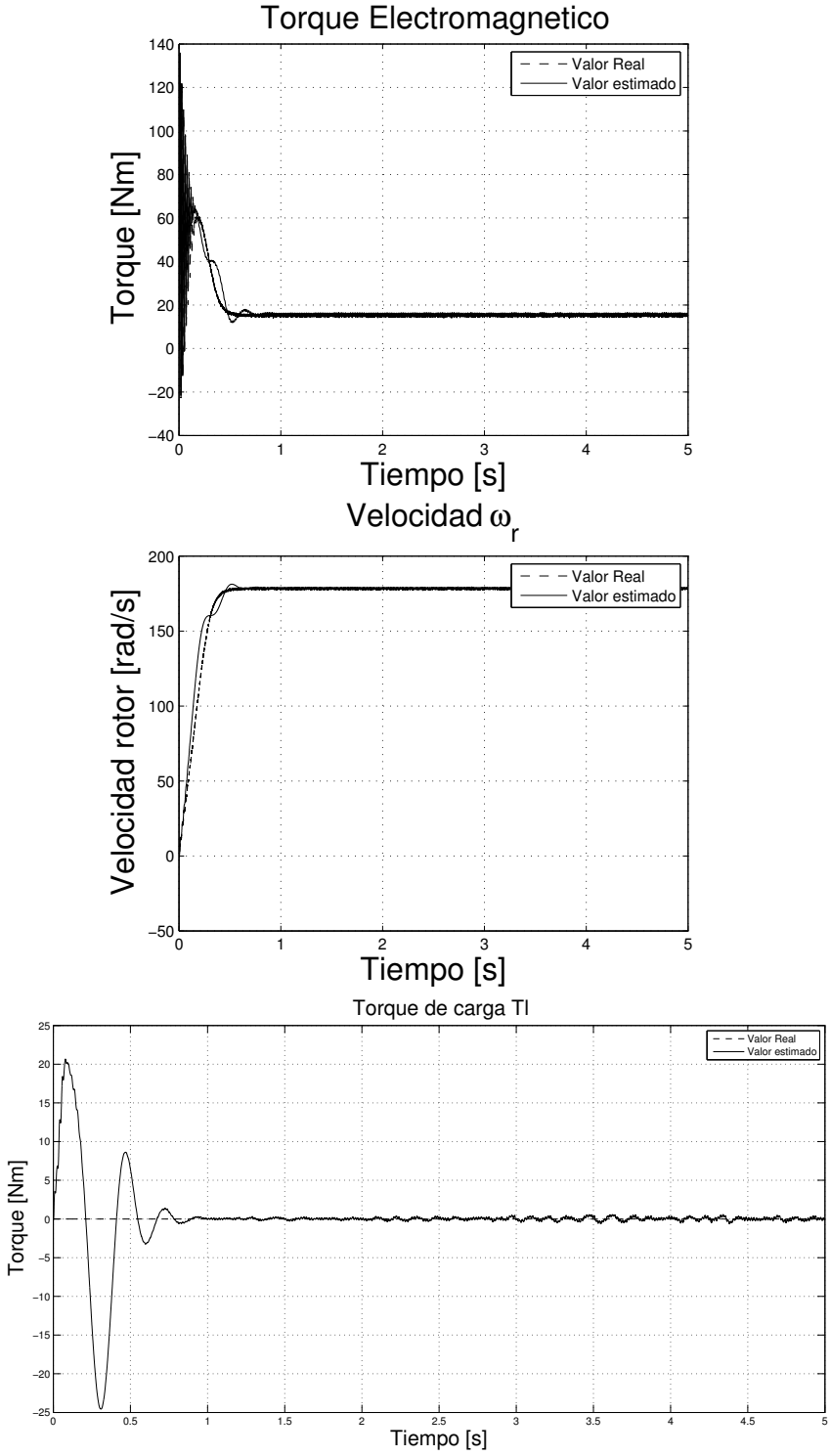


Figura 3.5: Filtro Kalman en aproximación lineal en estado estable sin perturbación

En el caso del torque electromagnético y de la velocidad del rotor se observan errores

en estimación notorios en la transición del arranque del motor (oscilaciones) hacia el estado estable. Sin embargo esta transición es rápida y el filtro efectivamente mejora su desempeño en los siguientes tramos. En estado estable no hay error significativo. En el caso del torque de carga se observa una fuerte oscilación inicial con grandes picos pero posteriormente el sistema se estabiliza y se obversa una estimación de gran calidad si se considera que no se tiene ningún información sobre la naturaleza del torque de carga. Se lo asume como una perturbación constante en el tiempo.

3.1.3. Aproximación lineal en T_s-Perturbación de carga

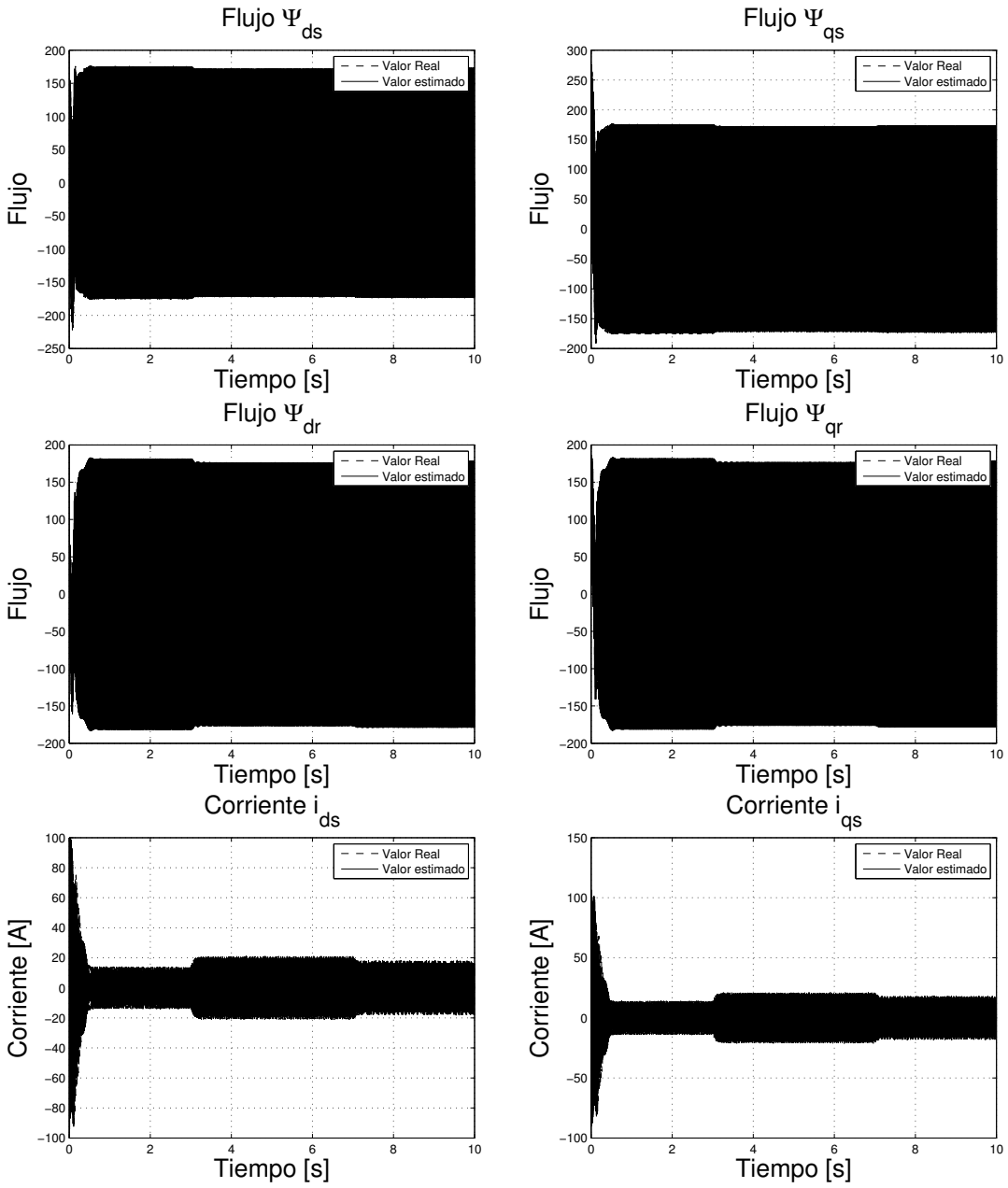


Figura 3.6: Filtro Kalman en aproximación lineal con perturbación (torque de carga)

En el caso de las perturbaciones se observa claramente como varían los estados estimados al momento de la perturbación. El efecto es particularmente notorio en las corrientes, donde ésta aumenta significativamente cuando aumenta el torque de carga (perturbación en $t = 3s$). Y nuevamente disminuye al reducirse el efecto ($t = 7s$).

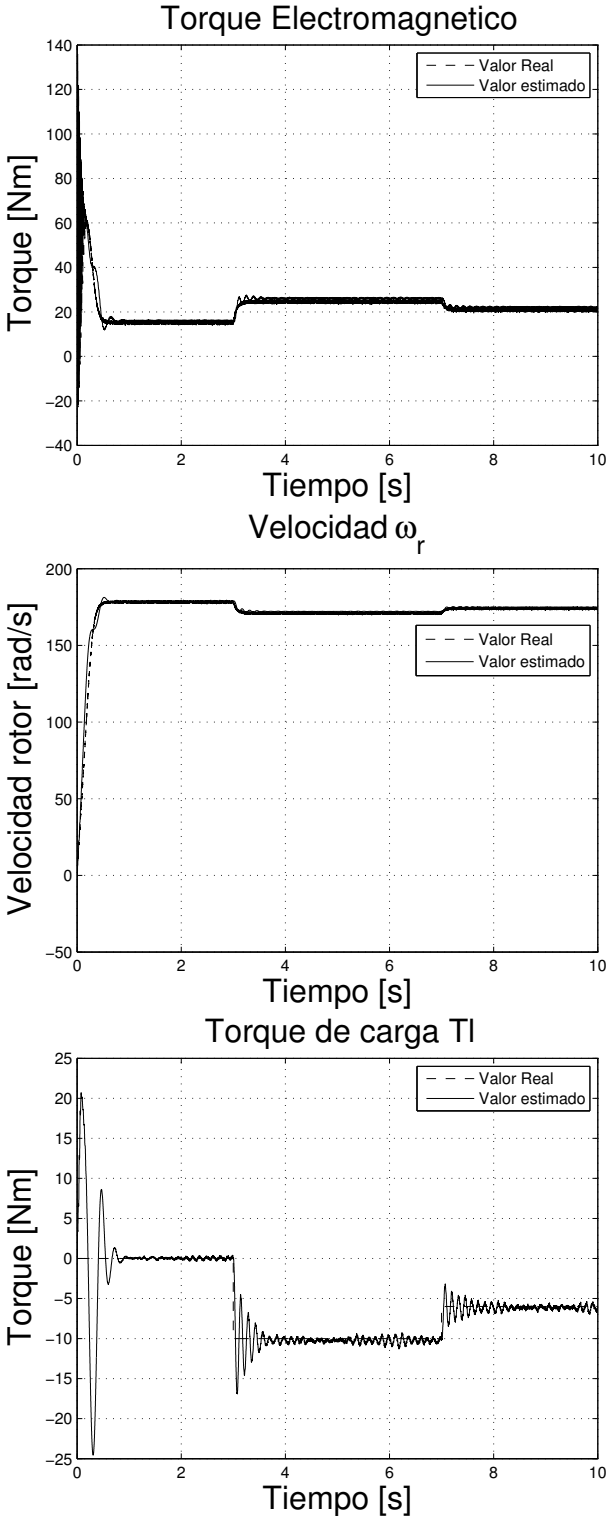


Figura 3.7: Filtro Kalman en aproximación lineal con perturbación (torque de carga)

Se observa que el filtro tiene muy buena respuesta a las perturbaciones en el caso del

torque electromagnético y la velocidad del rotor. La señal estimada y real prácticamente no se distinguen y la reacción es inmediata. En el caso del torque de carga se observan picos que se atenuan en las transiciones del torque. Esto se asemeja a la respuesta subamortiguada de un sistema de segundo orden. Una vez alcanzado el estado estable no hay mayor oscilación en el torque.

3.1.4. Aproximación lineal en T_s-Vista detallada de los efectos de la perturbación

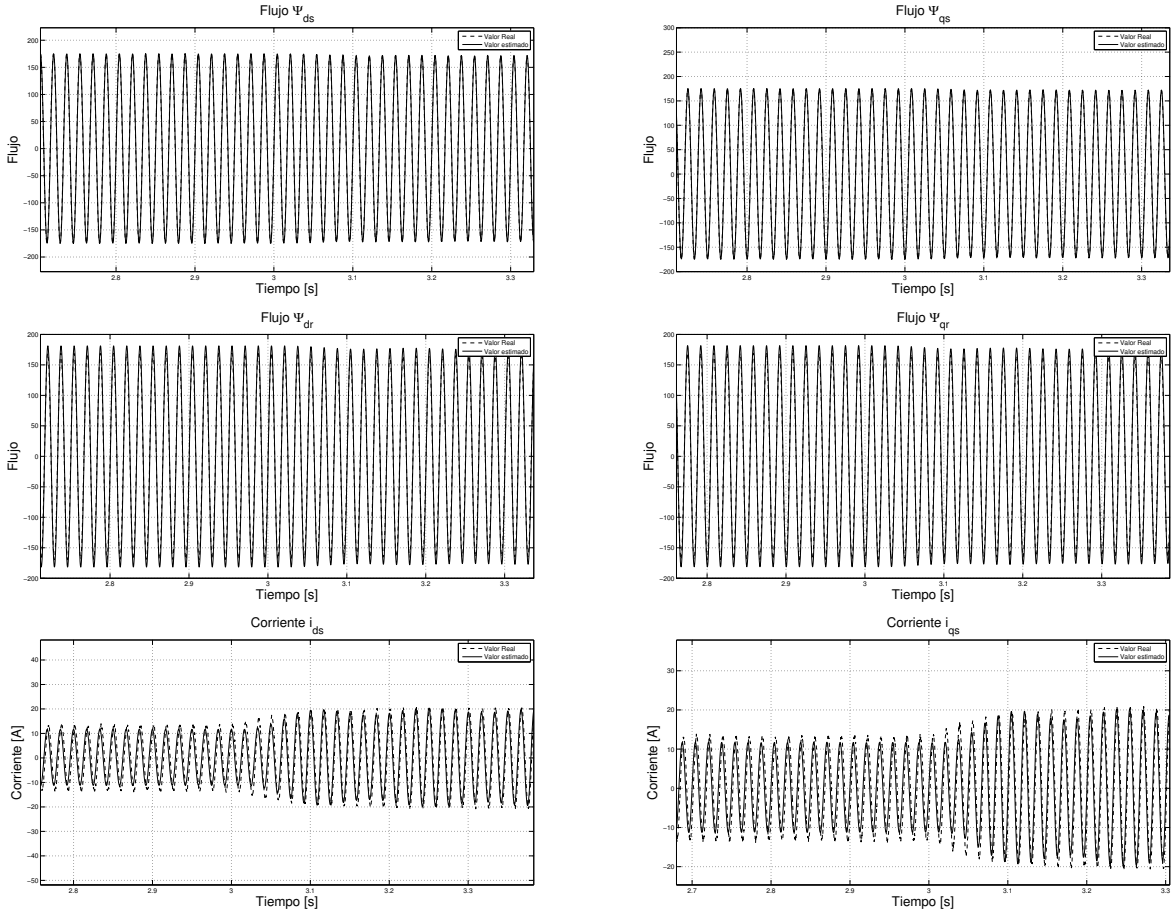


Figura 3.8: Filtro Kalman en aproximación lineal con perturbación-Vista detallada de los efectos de la perturbación

Se observa con claridad la calidad de la estimación. Prácticamente no se diferencian las variables reales de las estimadas. En el transitorio de corriente se observa una leve diferencia que rápidamente se compensa y se equipara en menos de 0.1 s. (en $t = 3s$)

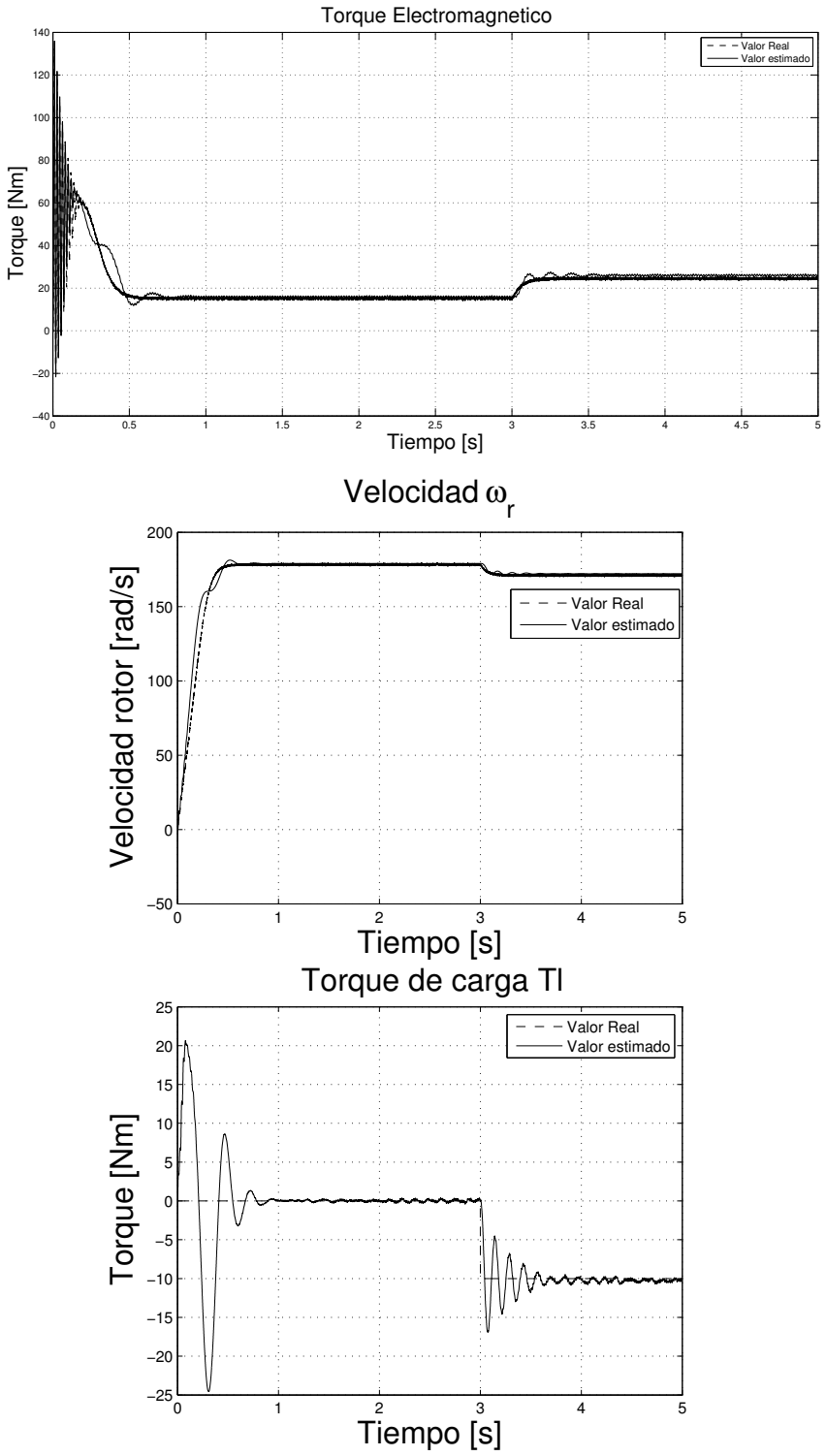


Figura 3.9: Filtro Kalman en aproximación lineal con perturbación-Vista detallada de los efectos de la perturbación

En el torque electromagnético se observan pequeñas diferencias en las transiciones, sin embargo el sistema rápidamente se estabiliza y se obtienen los mismos resultados para la variable estimada y la variable real. De igual manera en el caso de la velocidad rotórica. Finalmente, en el caso del torque de carga se observan oscilaciones considerables alrededor del punto referencial en los transitorios que se atenuan rápidamente y siguen la dinámica esperada.

3.1.5. Aproximación cuadrática en T_s-Arranque

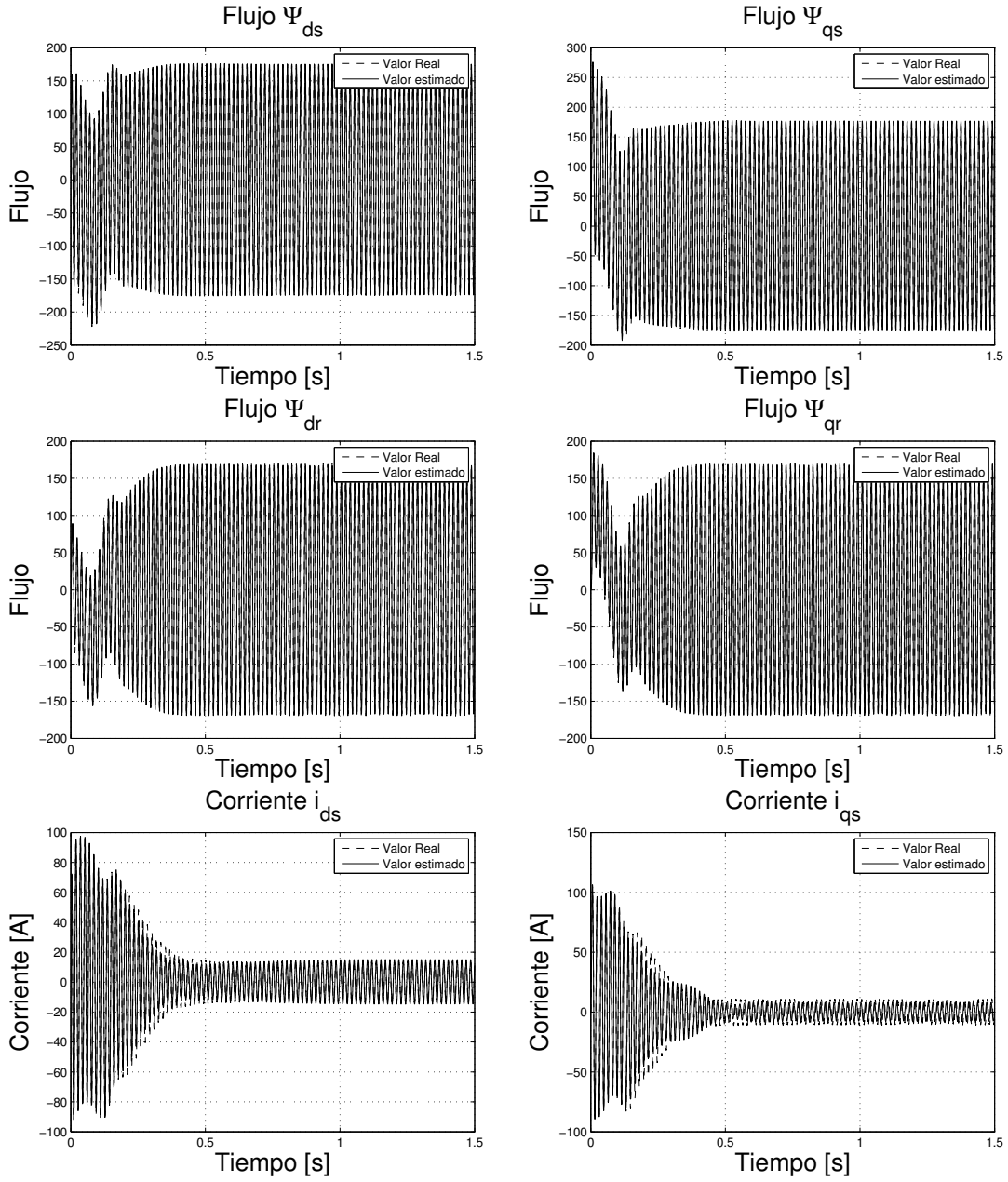


Figura 3.10: Filtro Kalman en aproximación cuadrática al arranque.

En el arranque se observa que la aproximación de los flujos y corrientes es bastante buena. Si bien no se aprecia claramente las dos señales la no presencia de picos o cambios bruscos indican resultados favorables.

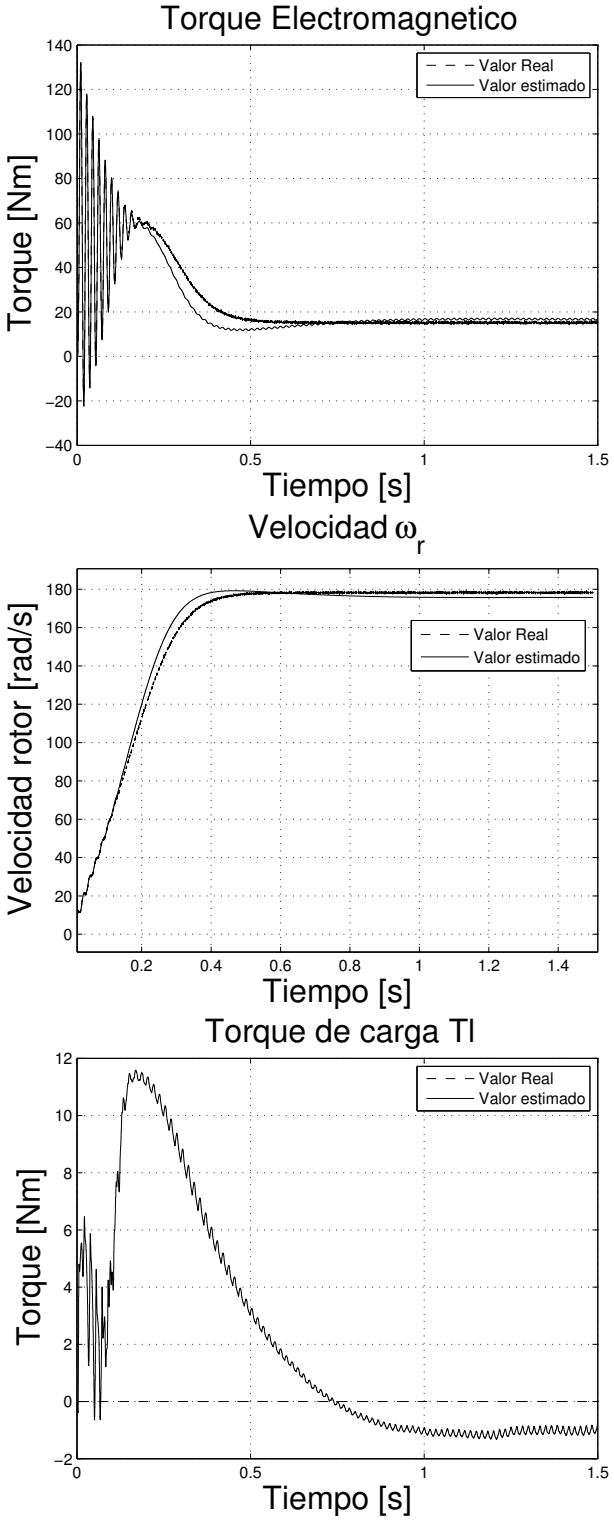


Figura 3.11: Filtro Kalman en aproximación cuadrática al arranque.

Se observan diferencias en el torque electromagnético y en la velocidad del rotor al mo-

mento de la transición del arranque hacia el estado estable. No obstante el filtro se estabiliza rápidamente. Por otra parte, el torque de carga muestra una respuesta más atenuada en el sentido que ya no existen oscilaciones sino un solo pico (menor que las oscilaciones del modelo lineal). Sin embargo, se observa también una respuesta más lenta ya que a $t = 1{,}5s$ la variable todavía no converge a la señal de referencia ($T_l = 0$)

3.1.6. Aproximación cuadrática en T_s-Estado estable sin perturbación

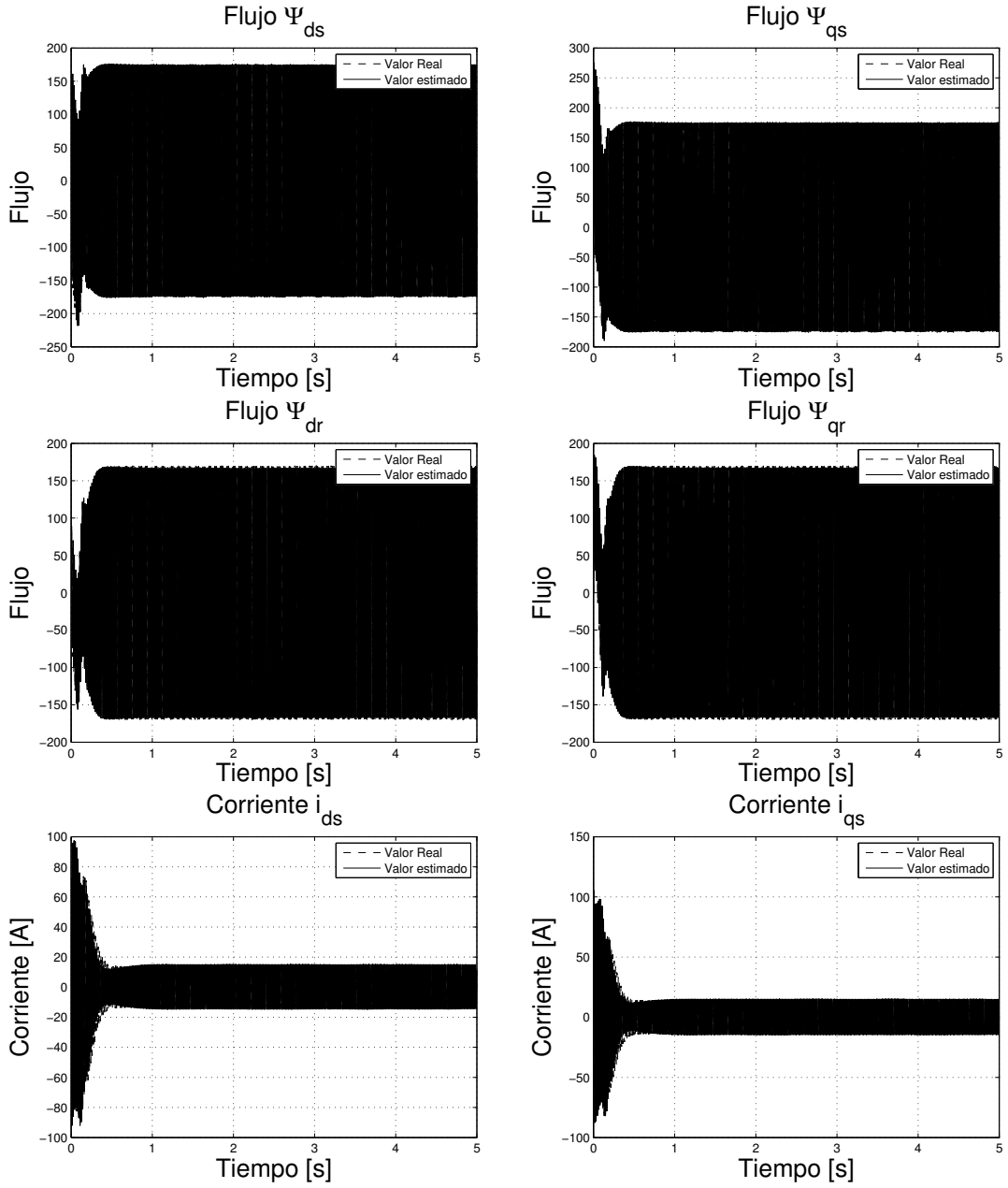


Figura 3.12: Filtro Kalman en aproximación cuadrática en estado estable sin perturbaciones.

No se puede apreciar de forma clara la calidad de la estimación pero se puede esperar que la tendencia este correcta dado que no existen saltos o discontinuidades que resalten en alguna de las gráficas.

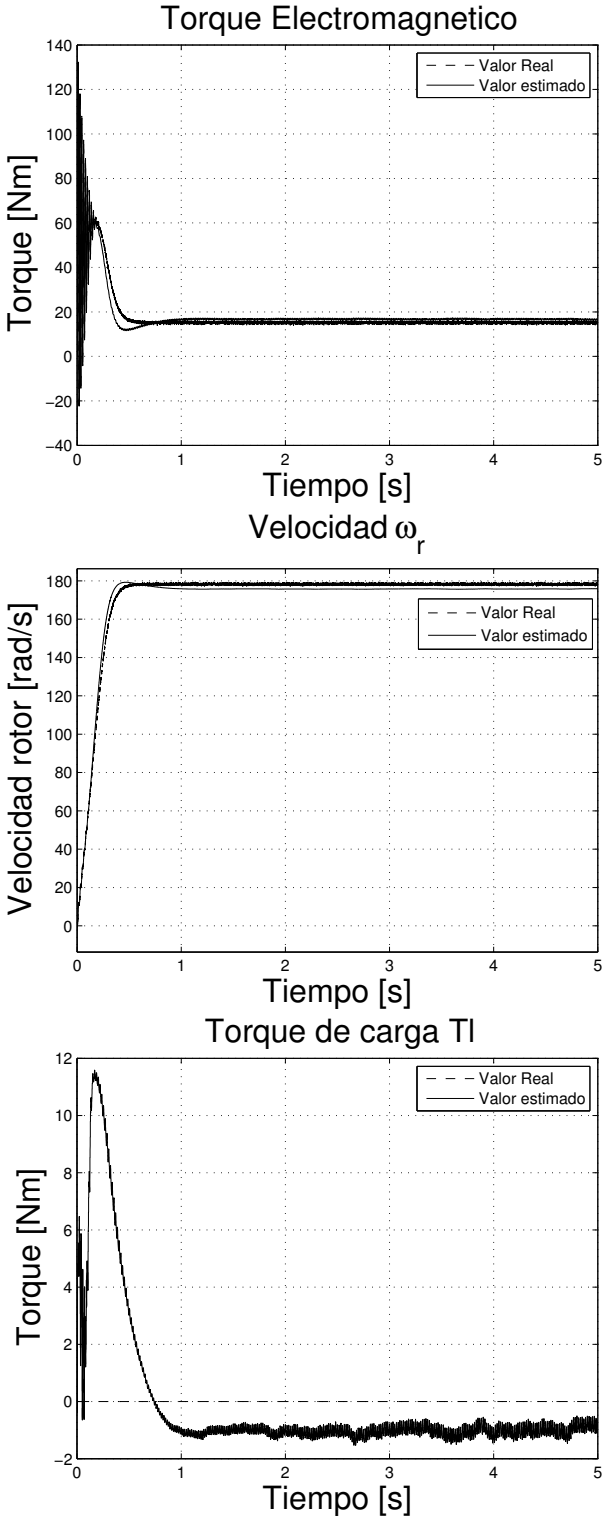


Figura 3.13: Filtro Kalman en aproximación cuadrática en estado estable sin perturbaciones.

El torque electromagnético se observa correctamente estimado no obstante en la gráfica

de la velocidad del rotor se observa una ligera desviación en estado estable. El valor estimado se muestra menor al valor real. En el caso del torque de carga se ve un pico al inicio de la estimación pero éste se atenúa rápidamente. Sin embargo, en el estado estable se observa una desviación considerable en el valor estimado. Esta desviación o error de estimación tiene tendencia a mantenerse.

3.1.7. Aproximación cuadrática en T_s-Perturbación

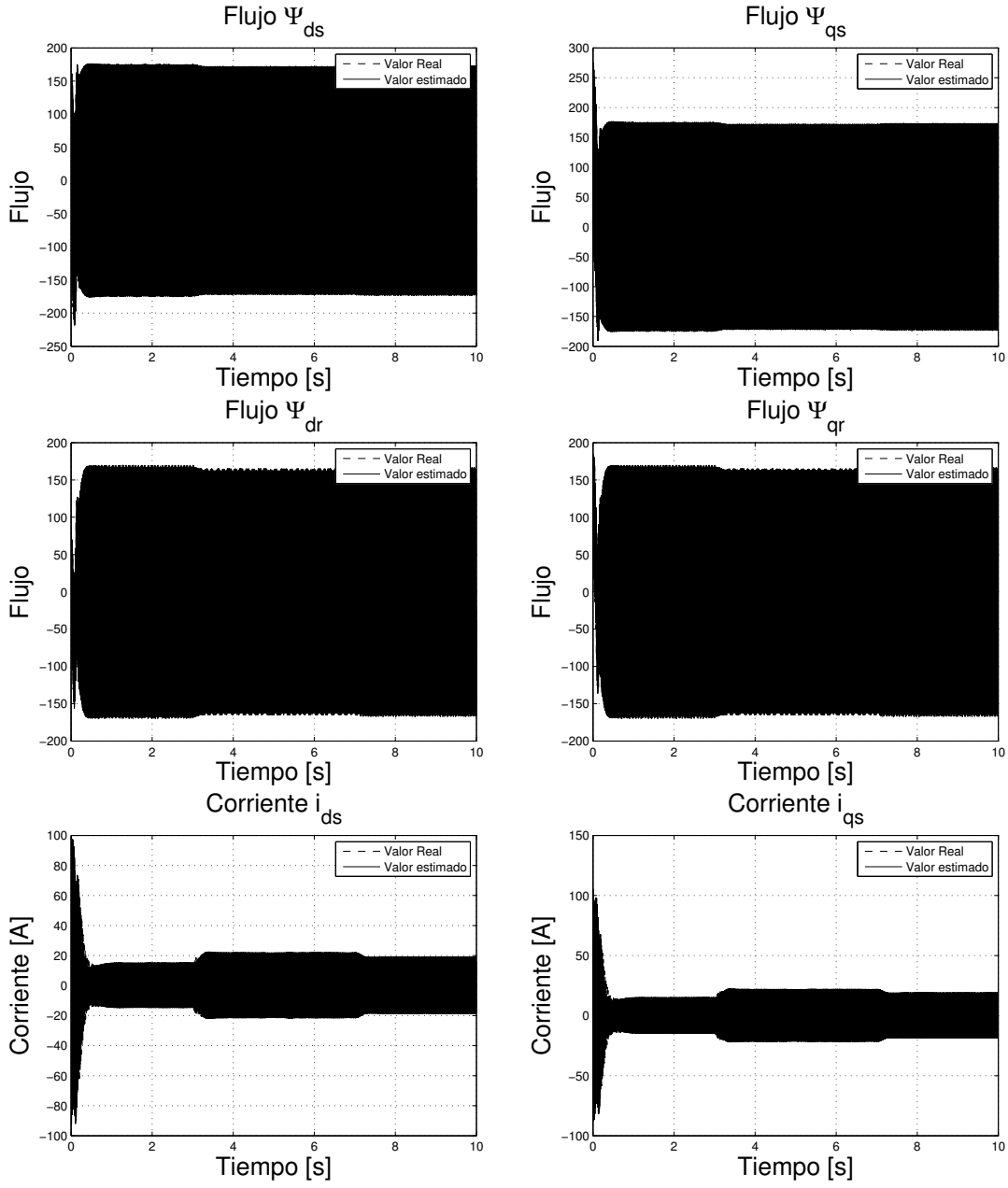


Figura 3.14: Filtro Kalman en aproximación cuadrática con perturbaciones.

Se observa que tanto la estimación de los flujos como las corrientes responden de manera adecuada a los cambios generados por la perturbación (cambio en el torque de carga). En $t = 3s$ existe un aumento de la corriente debido al aumento de la carga y esta corriente se reduce en $t = 7s$ debido a una reducción en el torque de carga. La dinámica del estimador es consistente con la dinámica del modelo.

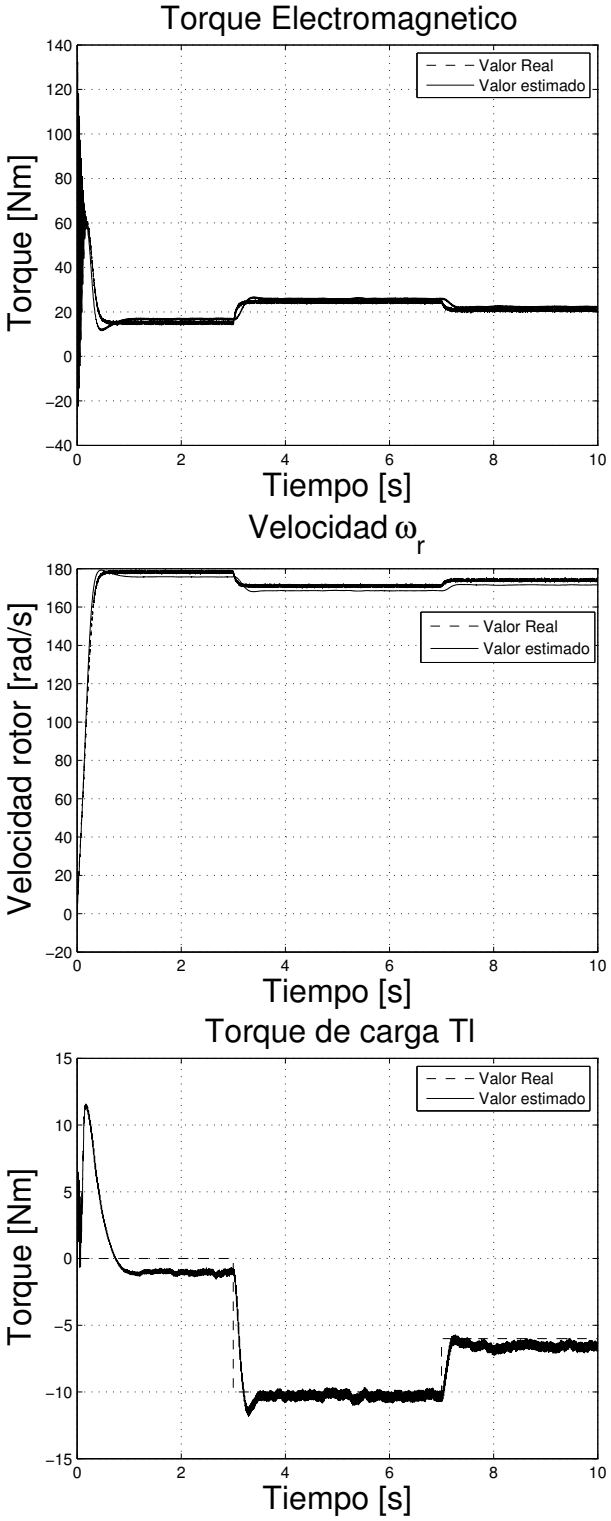


Figura 3.15: Filtro Kalman en aproximación cuadrática con perturbaciones.

Se observan diferencias entre las variables reales y las variables estimadas en los instantes

en los que ocurren las perturbaciones. Sin embargo, el filtro reacciona rápidamente y sigue la dinámica del sistema. En el caso de la velocidad del rotor se observa un valor de desviación en la estimación. En el caso del torque de carga, se observan picos al momento de las transiciones, pero no hay mayores oscilaciones amortiguadas. Así también, si bien alrededor de torque de carga $T_l = 0Nm$ se observa desviación, para otros valores en dicha variable la desviación desaparece y la señal estimada corresponde a la señal de referencia.

3.1.8. Aproximación cuadrática en T_s-Vista detallada de los efectos de la perturbación

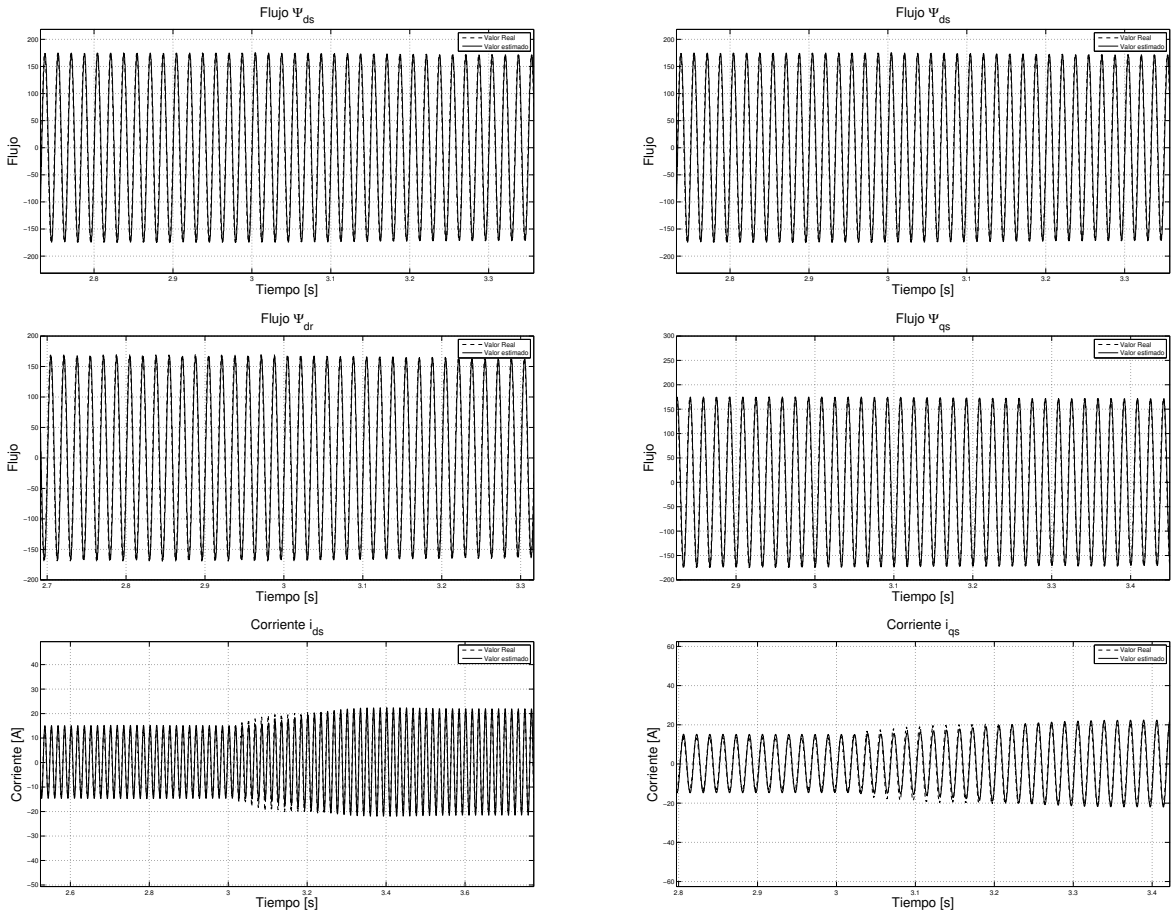


Figura 3.16: Filtro Kalman en aproximación cuadrática con perturbaciones- Vista detallada de los efectos de la perturbación

En el detalle de los flujos y corrientes durante una perturbación se observa como el filtro responde adecuadamente a la dinámica del sistema. Únicamente en la corriente se observa un breve desnivel que rápidamente es compensado y las señales estimada y real se sobreponen. De igual manera no se observa algún tipo de retraso en las respuesta estimada.

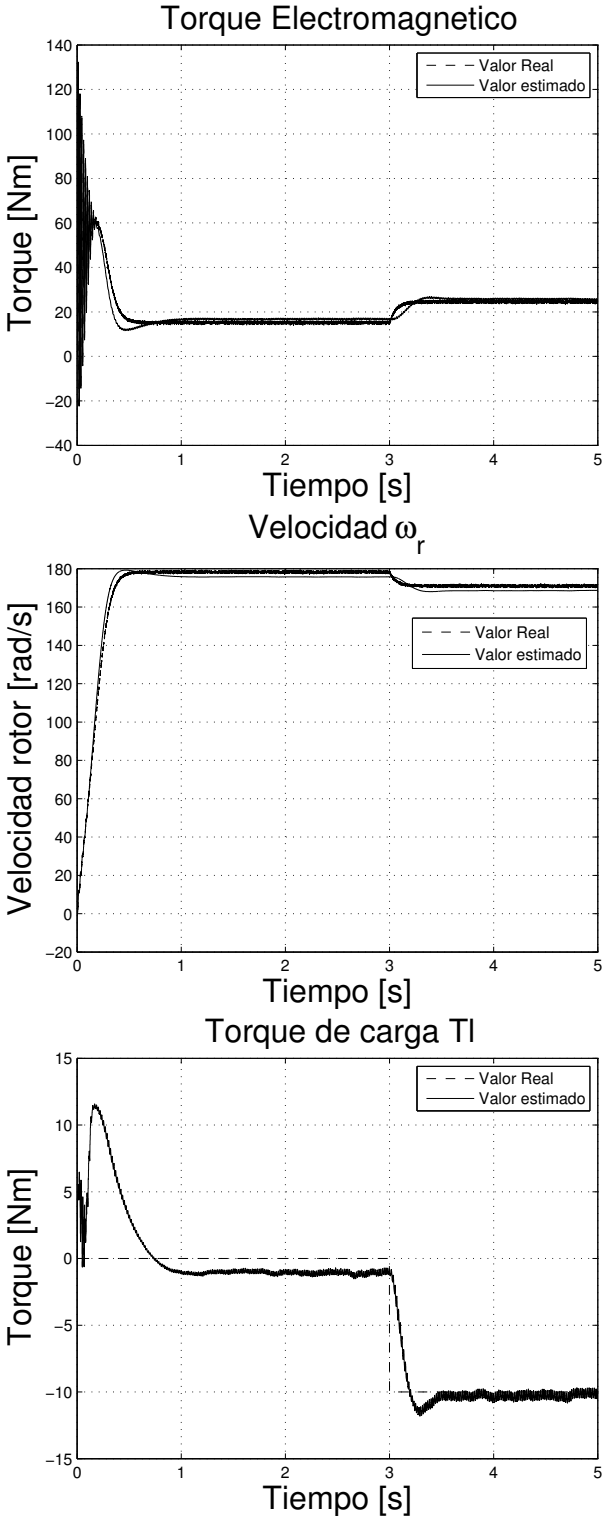


Figura 3.17: Filtro Kalman en aproximación cuadrática con perturbaciones- Vista detallada de los efectos de la perturbación

La curva de torque electromagnético y la curva de velocidad muestran diferencias en la transición que se da al momento de producirse la perturbación. En el caso de la velocidad del rotor también se observa un pequeño valor de desviación en la señal luego de producirse la perturbación. En la gráfica de torque de carga se ve que existe un pico inicial al producirse la perturbación, pero éste se atenúa rápidamente, sin oscilaciones considerables. A continuación sigue la dinámica de la señal referenciada. Al inicio se observa un valor de desviación, sin embargo, luego de la perturbación este valor desaparece e incluso el ruido de oscilación es bajo.

De los escenarios simulados anteriormente: arranque del motor, estado estable y perturbaciones se concluye que la respuesta del filtro es favorable en los dos casos. En lo referente a estimación de flujos y corrientes prácticamente no hay diferencia entre una y otra estimación. En lo referente a estimación de torque electromagnético y velocidad del rotor se observa que la aproximación lineal tiene un mejor comportamiento. En el caso de la aproximación cuadrática existen valores de desviación en la estimación de las variables mencionadas sobretodo luego de producirse una perturbación. Esta desviación puede ser crítica en una aplicación real donde se precisa conocer con exactitud la velocidad a la que gira la máquina. En este sentido la aproximación lineal parece ser la opción más viable. En el caso del torque de carga, se observa dos respuestas completamente diferentes. En la aproximación lineal se observan oscilaciones subamortiguadas en cada transición. Éstas se atenúan rápidamente a pesar que las primeras oscilaciones son de gran amplitud. Una vez atenudas se ve que el sistema sigue de manera adecuada la referencia. Por otra parte, en la aproximación cuadrática se observa no existen oscilaciones de consideración. Únicamente al producirse la perturbación se observa un pico significativo que se atenúa y la señal estimada sigue perfectamente a la señal de referencia. No obstante, al arranque sin torque de carga la estimación tiene un valor de desviación de aproximadamente $1{,}2Nm$ que es poco menos del 10 % del torque nominal de la máquina. En términos de aplicación es una desviación importante que puede limitar seriamente el desempeño del filtro en un sistema de control.

En base a lo analizado se considera que la aproximación lineal es suficiente para la implementación real del filtro por tres razones fundamentales:

- La carga computacional es menor ya que varios términos en las matrices son cero y esto reduce el tiempo y número de cálculos. Además, en la aproximación lineal algunos de los términos son constantes (en el caso cuadrático dichos términos varían en el tiempo) por lo que pueden ser calculados al inicio del algoritmo y no requieren de actualización en cada iteración del algoritmo. Esto también contribuye a reducir la carga computacional sobre el DSP.

- La simulación muestra que no hay diferencia fundamental en el desempeño del filtro en aproximación lineal y el filtro en aproximación cuadrática. En los dos casos existen pros y contras que no son determinantes ni críticos a la hora de tomar una decisión. Basado en las ventajas computacionales se opta por la aproximación lineal.

- Además se considera que al utilizar un tiempo de muestreo $T_s = 200\mu s$, se puede despreciar los términos con $T_s^2 = 40ns^2$. Incluso en este punto al tener en cuenta que el DSP es de punto fijo con 32bits muchos términos pueden ser numéricamente tan pequeños que el DSP los trate como 0 debido a la falta de resolución y finalmente no

aporten nada pero si utilicen espacio de memoria y tiempo de ejecución, que puede llegar a ser crítico en algún momento de la iteración.

3.2. Simulación filtro $\mathbf{H}_\infty$

A continuación se muestran los resultados de la simulación en el caso del Filtro H_∞ Extendido. Al igual que en la sección precedente se considera la estimación al arranque del motor, la estimación en estado estable y la estimación en el caso de perturbaciones (incluye un estudio detallado del comportamiento de las variables de interés). Se muestran los flujos magnéticos $qd0$ del estator y rotor así como las corrientes $qd0$ del estator. Adicionalmente se tienen gráficas del torque electromagnético, velocidad del rotor y torque de carga. A partir de los resultados y conclusiones obtenidas de la simulación del filtro Kalman extendido se considera suficiente utilizar el modelo lineal en T_s para la simulación de este filtro. Sin embargo, nuevamente con el objetivo de buscar la implementación más eficiente computacionalmente sin sacrificar el desempeño del filtro se probarán dos variaciones del filtro. En la primera se utiliza $L_k = I_{6\times6}$ como la matriz de combinación lineal definida en (2.50). En la segunda simulación se utliza $L_k = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

La diferencia fundamental radica en la carga computacional del algoritmo, ya que al incluirse la matriz $I_{6\times6}$ se requiere invertir una matriz R_w definida en (2.85) de dimensiones 8x8. Esta inversión es sumamente pesada desde el punto de vista del número de operaciones que debe realizarse y de los requerimientos de memoria y velocidad de procesamiento en el dispositivo. Por otra parte si se utiliza $L_k = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ la matriz R_w tiene dimensiones 4x4, y su inversa es significativamente menor en términos de operaciones y memoria necesaria. Se elige la matriz $L_k = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ dado que de esta forma al operar $z_k = L_k x_k$ se obtiene:

$$z_k = \begin{bmatrix} \omega_r \\ T_l \end{bmatrix} \tag{3.3}$$

Es decir el desempeño del filtro se centra en la estimación de la velocidad del rotor y torque de carga, que son el enfoque principal de este trabajo. Así también la función de costo (2.51) queda ligada directamente a estas variables.

A partir de los resultados en simulación se espera determinar si la elección de L_k es crítica en el desempeño del filtro. En caso de no serlo se optará por la matriz cuya carga computacional sea menor.

Para finalizar los comentarios previos a la simulación se muestran los valores de la matriz G_k y del parámetro γ utilizados en este trabajo.

Para el filtro con $L_k = I_{6\times6}$ se utilizó

$$G = 0{,}24 \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\gamma = 0{,}4$$

Para el filtro con $L_k = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ se utilizó

$$G = 0{,}1 \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\gamma = 0{,}2$$

3.2.1. Filtro con $L = I_{6\times6}$- Arranque

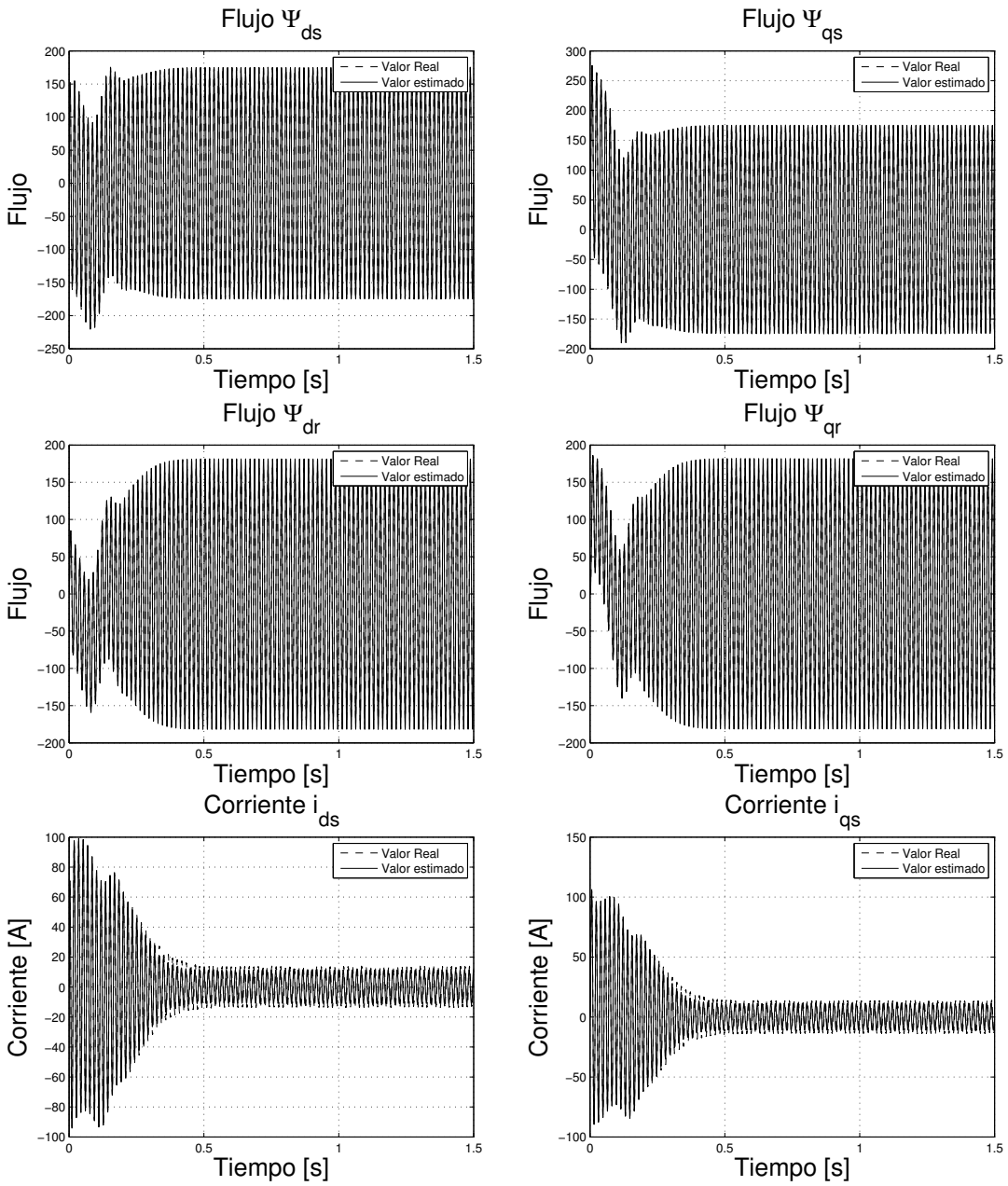


Figura 3.18: Filtro H_∞ con $L_k = I_{6\times6}$ al arranque.

La estimación es adecuada ya que no se observan picos o discontinuidades que indiquen problemas en el algoritmo. Así también se observa que las variables estimadas siguen la dinámica del sistema sin mayor problema sobretodo durante los 0.5 s iniciales de simulación que son donde se observa el arranque del motor.

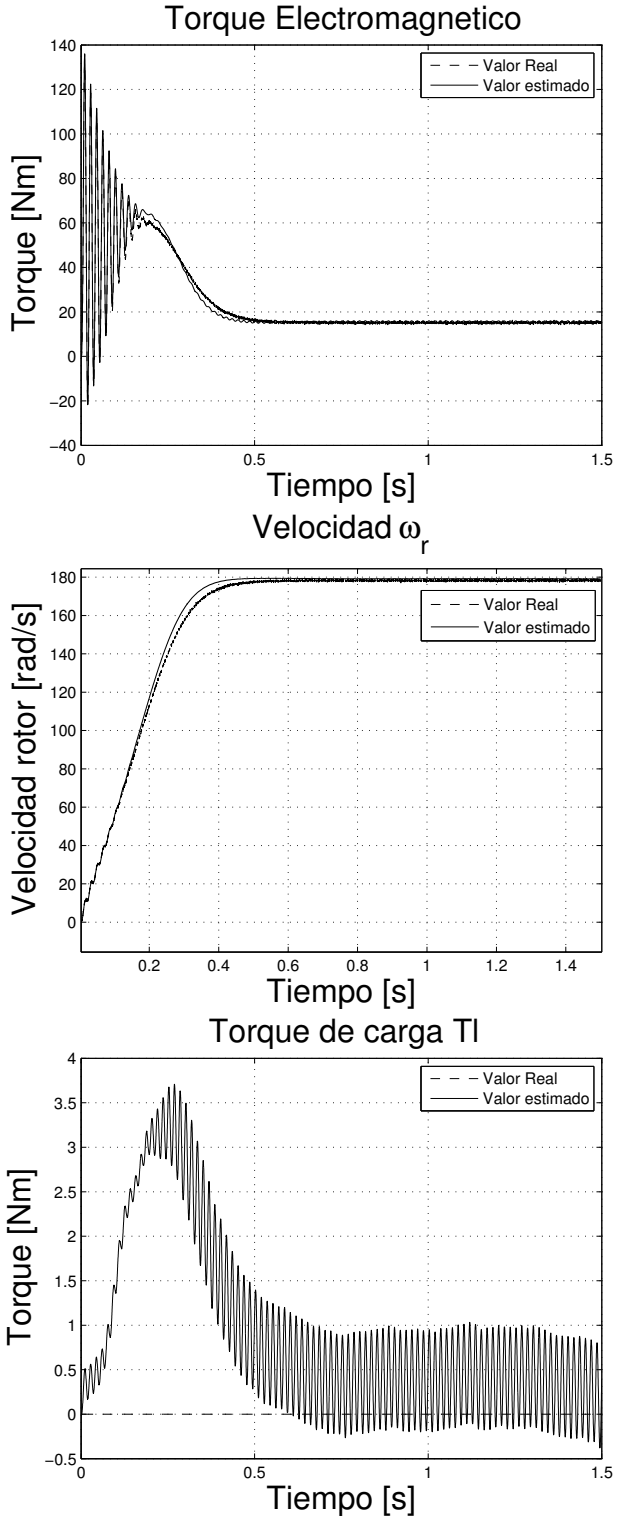


Figura 3.19: Filtro H_∞ con $L_k = I_{6\times 6}$ al arranque.

En la gráfica de torque electromagnético y velocidad del rotor se observa pequeñas varia-

ciones entre las señales reales y estimadas en la transición entre el arranque y el estado estable, sin embargo, esta situación se corrige rápidamente y el sistema responde de forma tal que cerca a $t = 0{,}6s$ no se aprecia diferencia alguna entre las señales. En el caso del torque de carga se observa un pico al inicio de la simulación. Sin embargo, este pico (3.6 Nm) es mucho menor (aproximadamente 33 %) al pico que se generaba en el filtro Kalman (10 Nm). No obstante, se observa que existe un fuerte oscilación del torque alrededor de la señal de referencia. La oscilación es de aproximadamente el 10 % del torque nominal, por lo que puede ser considerada significativa. La tendencia de la señal es la esperada.

3.2.2. Filtro con $L = I_{6\times6}$- Estado estable sin perturbación

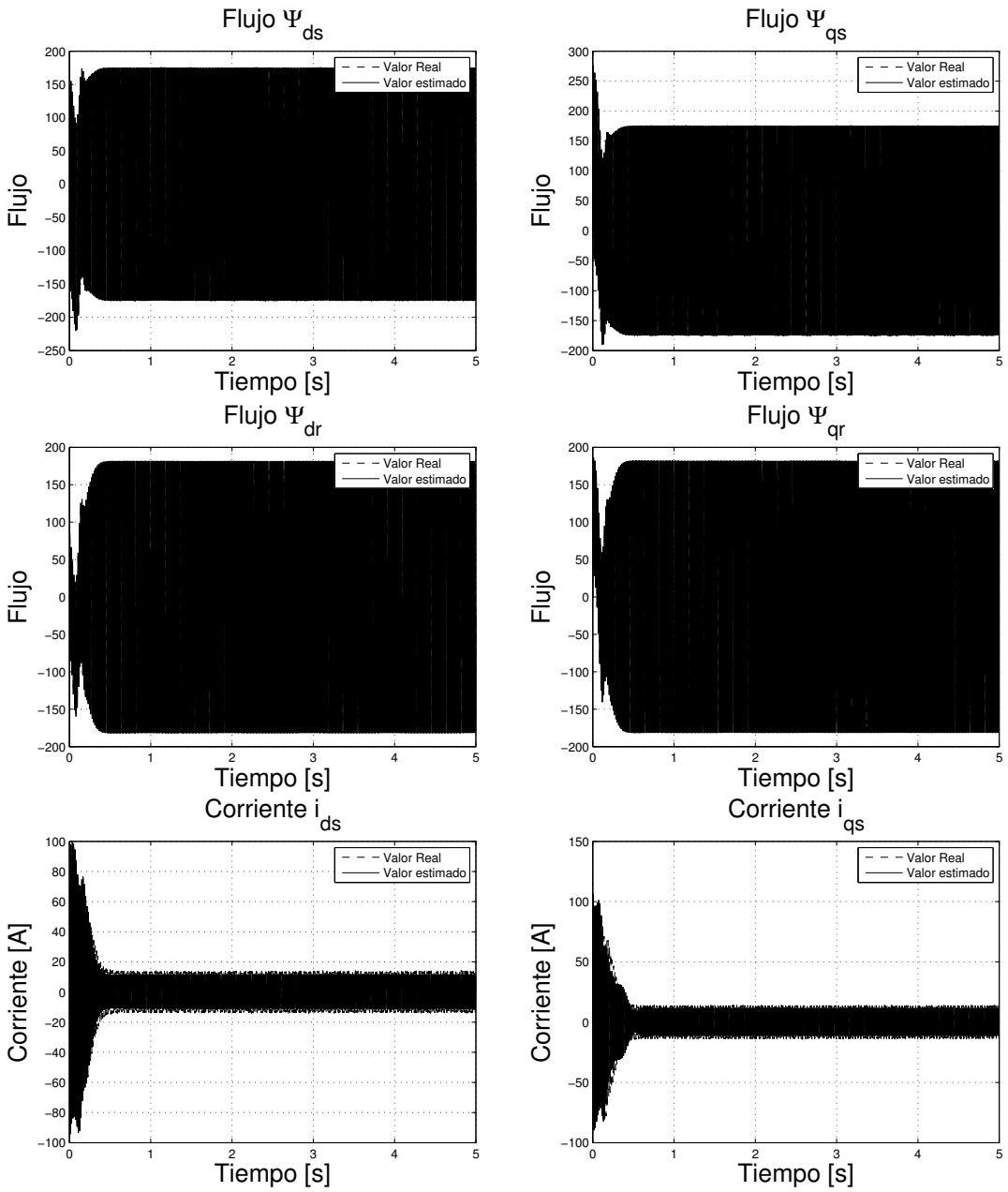


Figura 3.20: Filtro H_∞ con $L_k = I_{6\times6}$ en estado estable.

En el estado estable se puede observar que las señales reales y estimadas se sobreponen. Sin embargo, el pico de la señal estimada es menor al pico de la señal real. Este detalle debe ser considerado en el análisis.

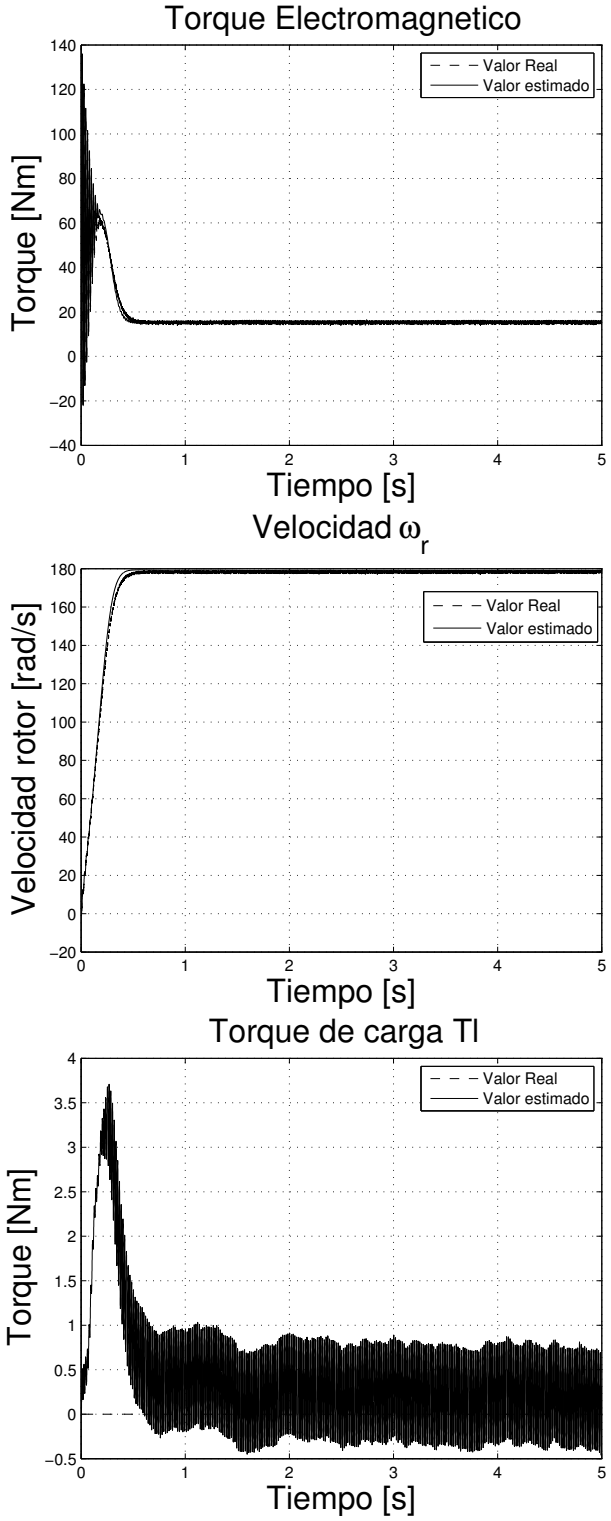


Figura 3.21: Filtro H_∞ con $L_k = I_{6\times 6}$ en estado estable.

En el caso del torque electromagnético y la velocidad del rotor se observa que existe una

diferencia mínima entre las señales reales y estimadas en $t = 0{,}5s$ aproximadamente. En los periodos anteriores y posteriores no se aprecia diferencia. En el caso del torque de carga se observa un pico pequeño rápidamente atenuado al inicio. Sin embargo, ya en el estado estable se observa ruido y oscilaciones de consideración en la señal estimada. La oscilación se da alrededor de la señal referencial.

3.2.3. Filtro con $L = I_{6\times6}$- Perturbación

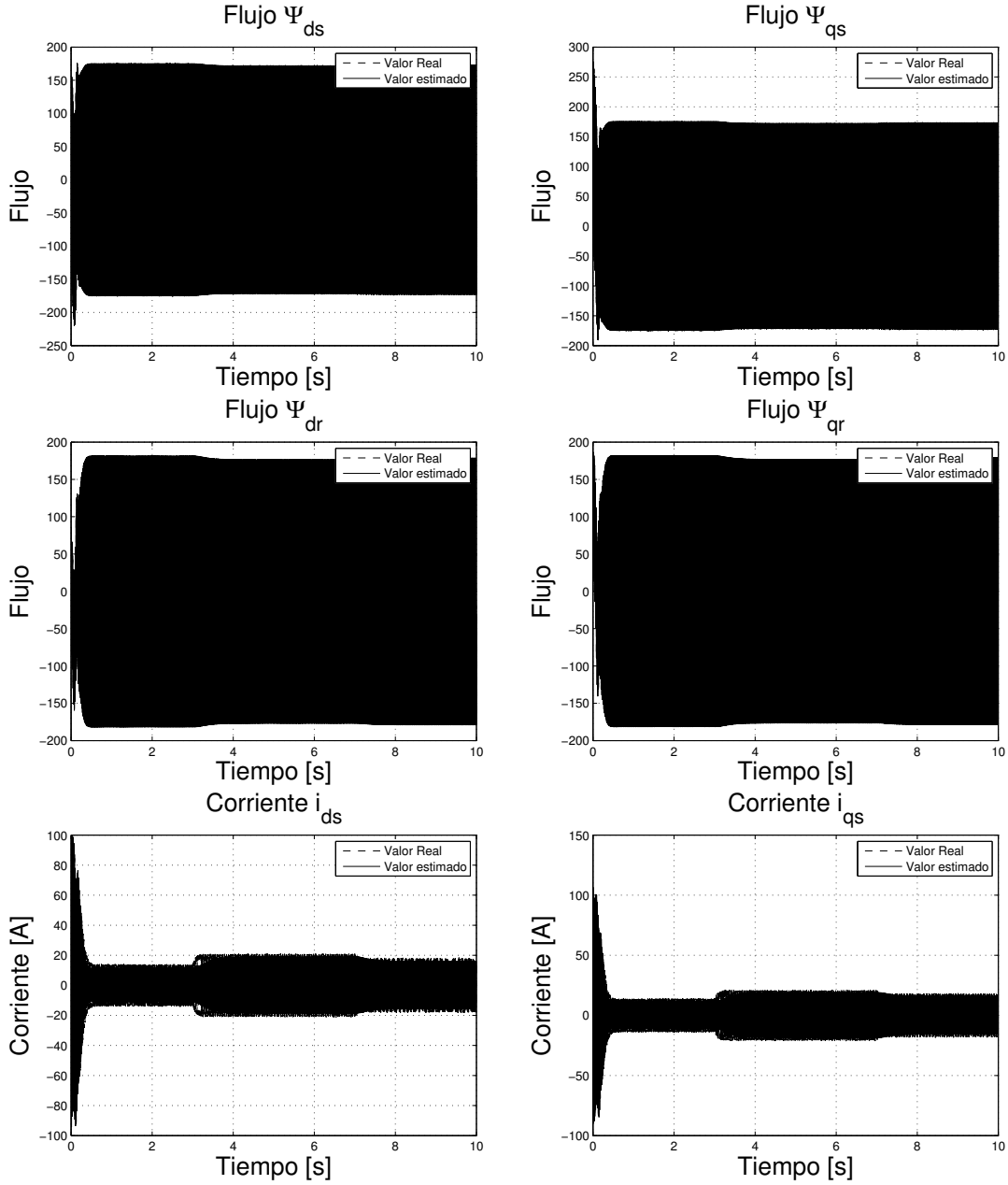


Figura 3.22: Filtro H_∞ con $L_k = I_{6\times6}$ con perturbaciones.

Al incluir perturbaciones en el sistema se puede observar que las variables responden según lo esperado. Al aumentar el torque de carga se producen aumentos en las corrientes, lo cual se refleja en las gráficas del modelo. De igual manera no se observan picos o discontinuidades que harian presumir de una falla en la estimación.

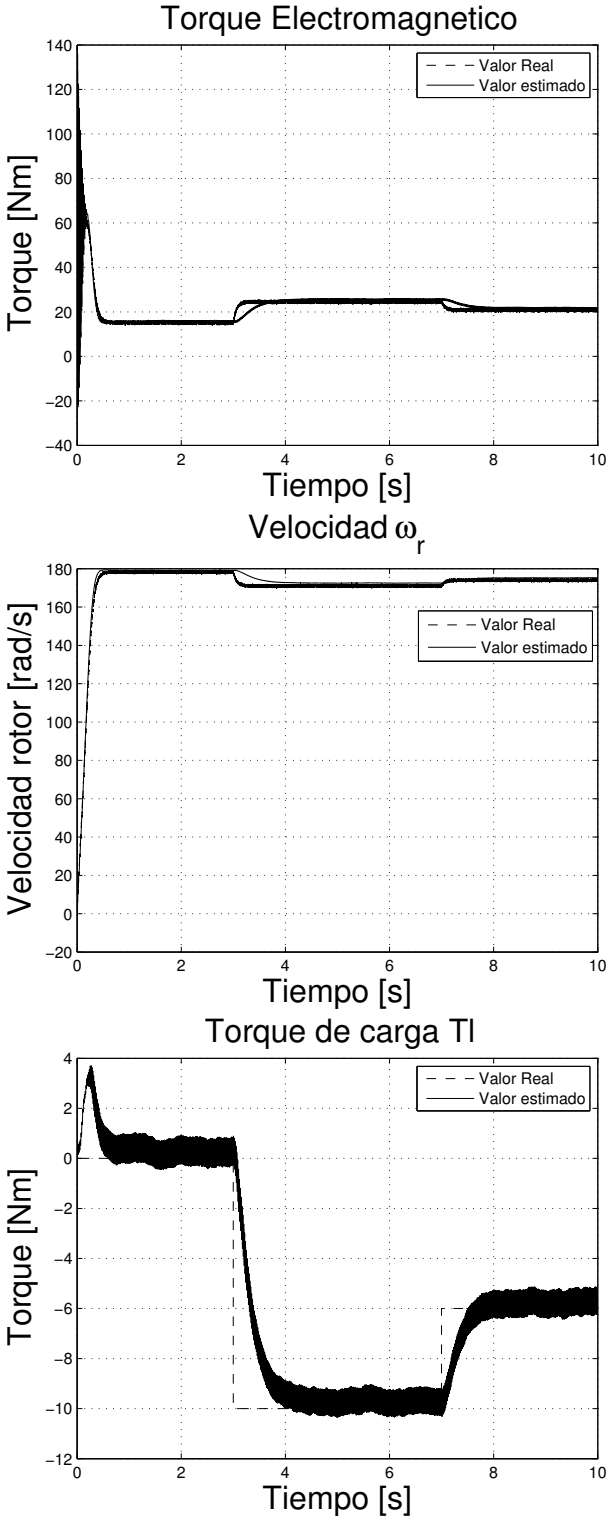


Figura 3.23: Filtro H_∞ con $L_k = I_{6\times 6}$ con perturbaciones.

En las gráficas de torque electromagnético y velocidad del rotor se puede apreciar que al

momento de las perturbaciones el filtro responde de forma algo lenta en la estimación haciendo que la señal estimada no coincida con la señal real. Sin embargo, al alcanzar el estado estable no existe diferencia entre las variables. En la gráfica de torque de carga se observa que el filtro responde sin picos u oscilaciones. Es decir el sistema parece más bien sobreamortiguado. Sin embargo, persiste la presencia de ruido y oscilaciones al alcanzar el estado estable.

3.2.4. Filtro con $L = I_{6\times6}$- Vista detallada de los efectos de la perturbación

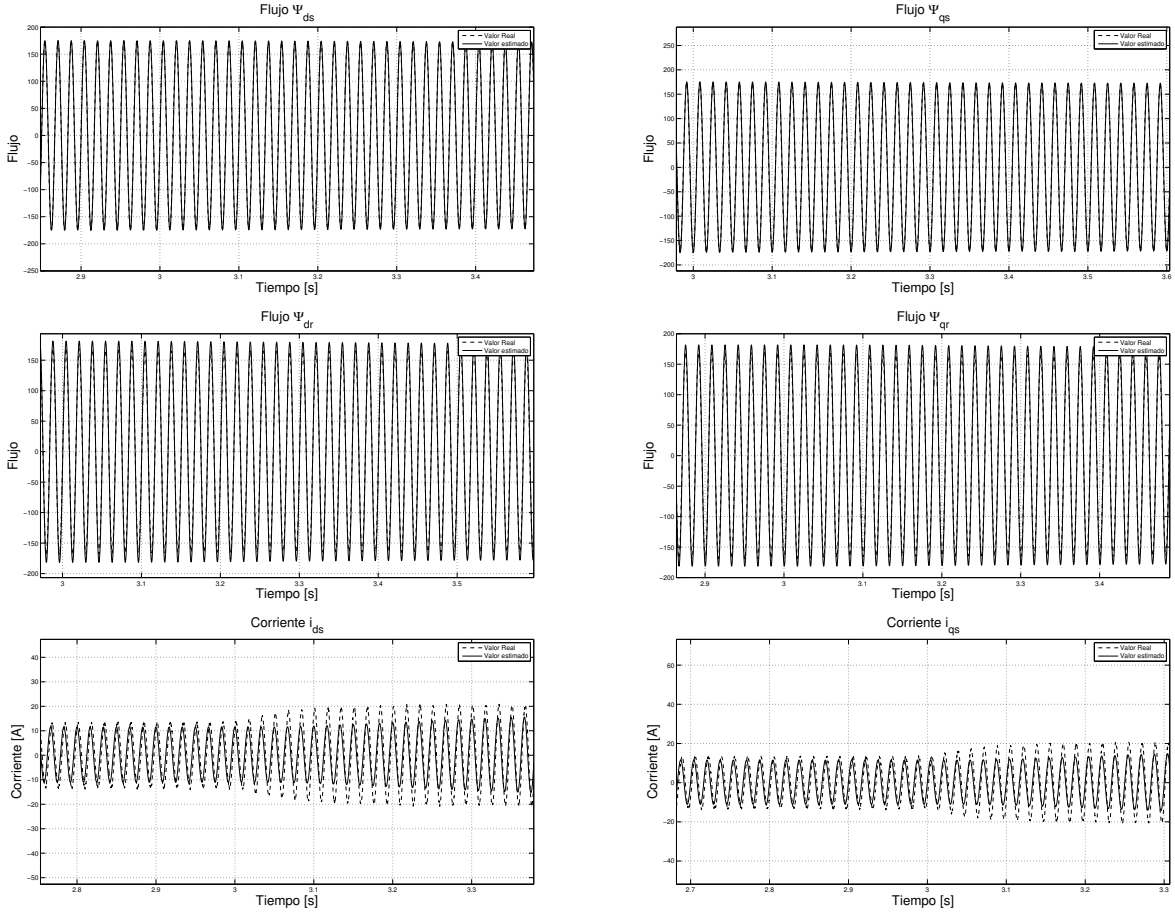


Figura 3.24: Filtro H_∞ con $L_k = I_{6\times6}$ con perturbaciones- Vista detallada de los efectos de la perturbación.

Al observar en detalle el comportamiento de la señal se aprecia que la estimación de flujos es de alta calidad ya que la señal real y estimada prácticamente se sobreponen una con otra incluso en presencia de perturbaciones ($t = 3s$). En el caso de las corrientes se observa que el filtro tiene una respuesta un tanto lenta ya que el cambio en la corriente mostrado en las curvas reales es mayor al cambio que sufren las corrientes en las curvas estimadas. Sin embargo, se aprecia que la tendencia de las corrientes estimadas es acercarse a los valores reales por su forma creciente en amplitud. Esta diferencia en corrientes puede ser la razón por la que los transitorios de torque electromagnético y velocidad sean algo lentos como ya se mencionó.

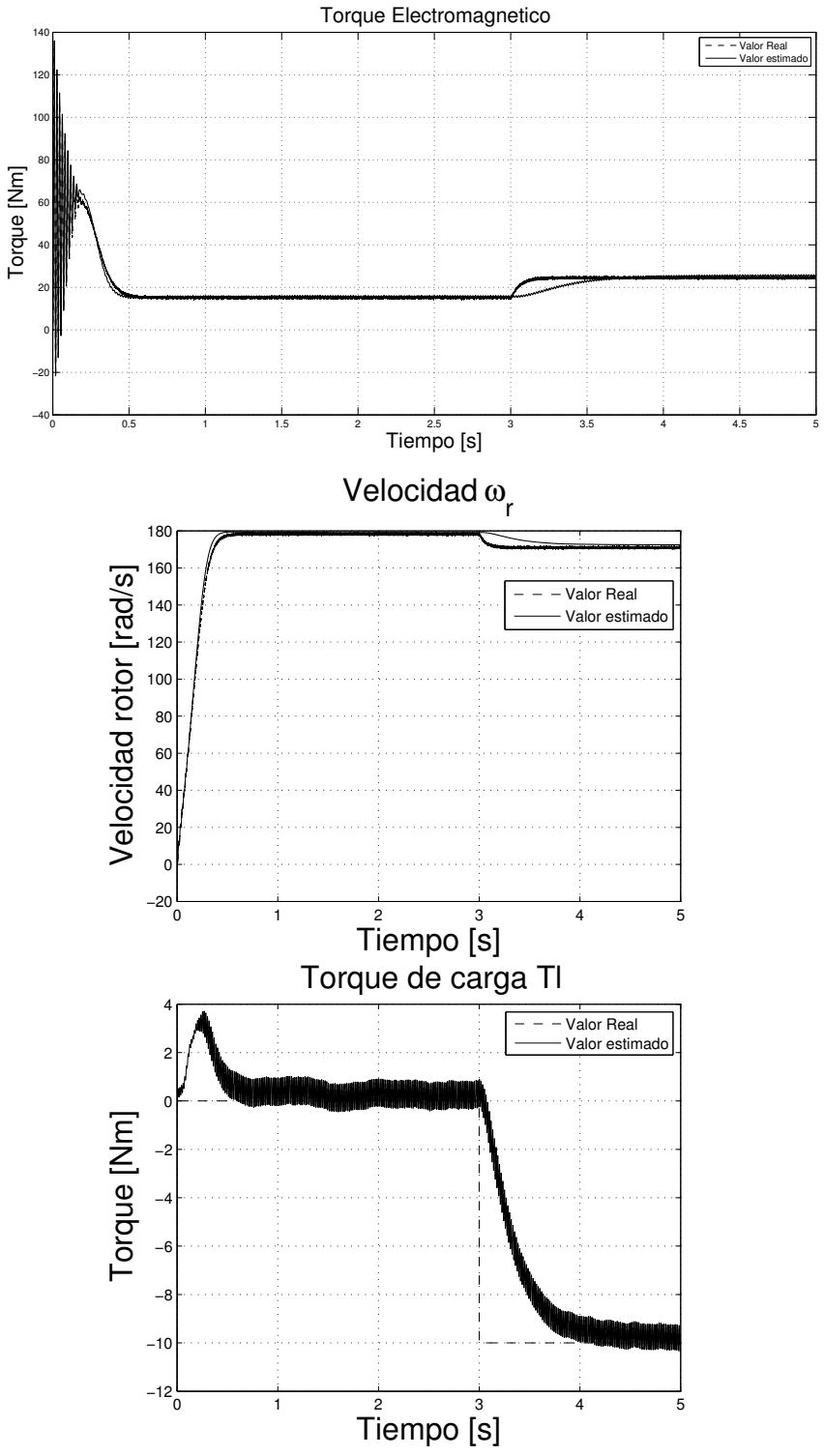


Figura 3.25: Filtro H_∞ con $L_k = I_{6\times 6}$ con perturbaciones- Vista detallada de los efectos de la perturbación

En estas gráficas se observa como el torque electromagnético y la velocidad tienen transitorios de respuesta lenta al momento de una perturbación. Como ya se mencionó esto puede ser principalmente debido a la diferencia entre la corriente real y estimada que se mostró en la gráfica anterior. Sin embargo, en el estado estable no se observa desviación o problema alguno. En el caso del torque de carga se mantiene el pico al inicio, la transición lenta en la perturbación y las oscilaciones sostenidas alrededor del punto referencial en el estado estable.

3.2.5. Filtro con $L = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ - Arranque

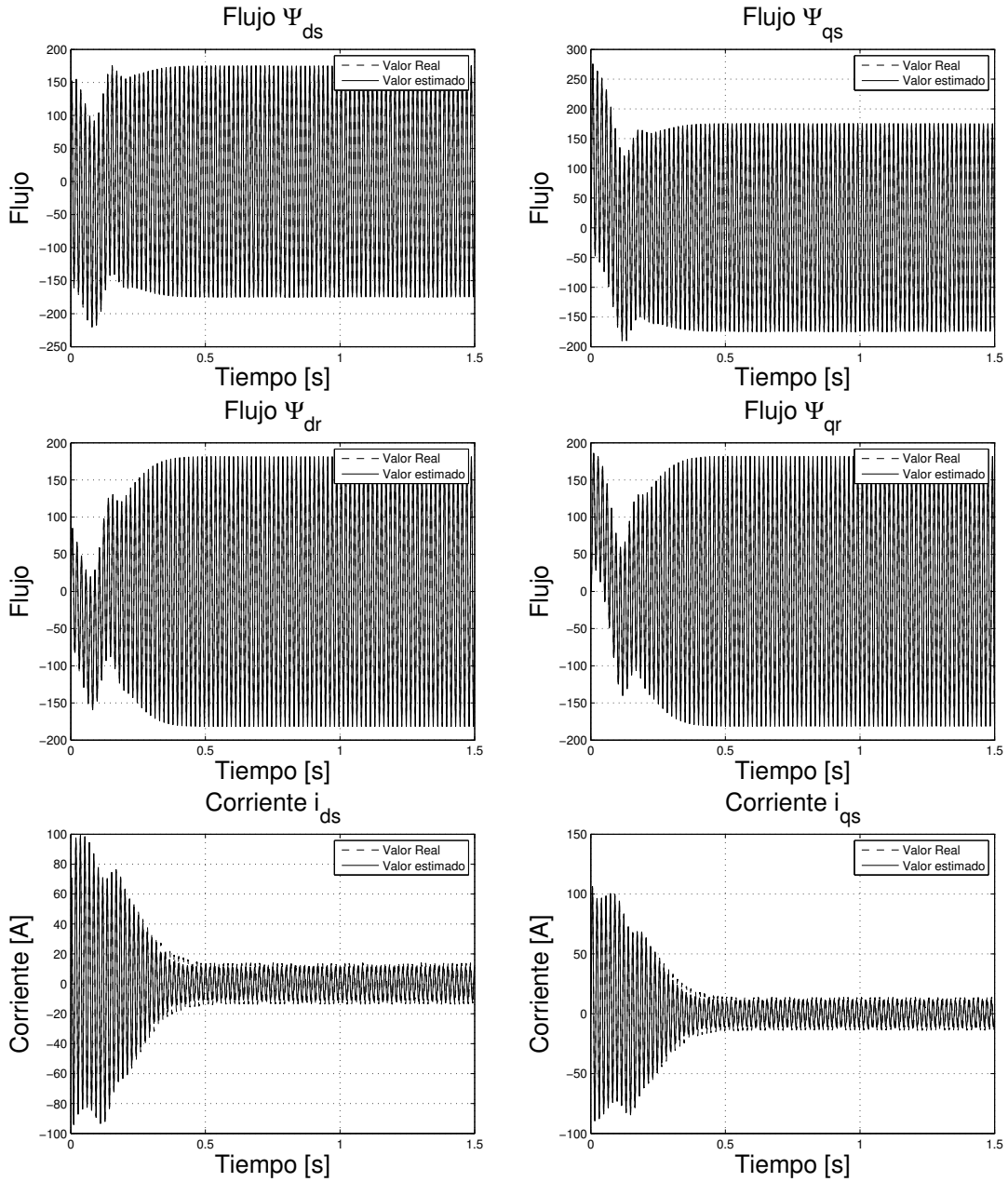


Figura 3.26: Filtro H_∞ con $L = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ al arranque.

En la estimación de flujos y corrientes al arranque no se observa ningún problema o indicio de error en los resultados. Las gráficas muestran que la señal estimada y real corresponde correctamente en cada caso.

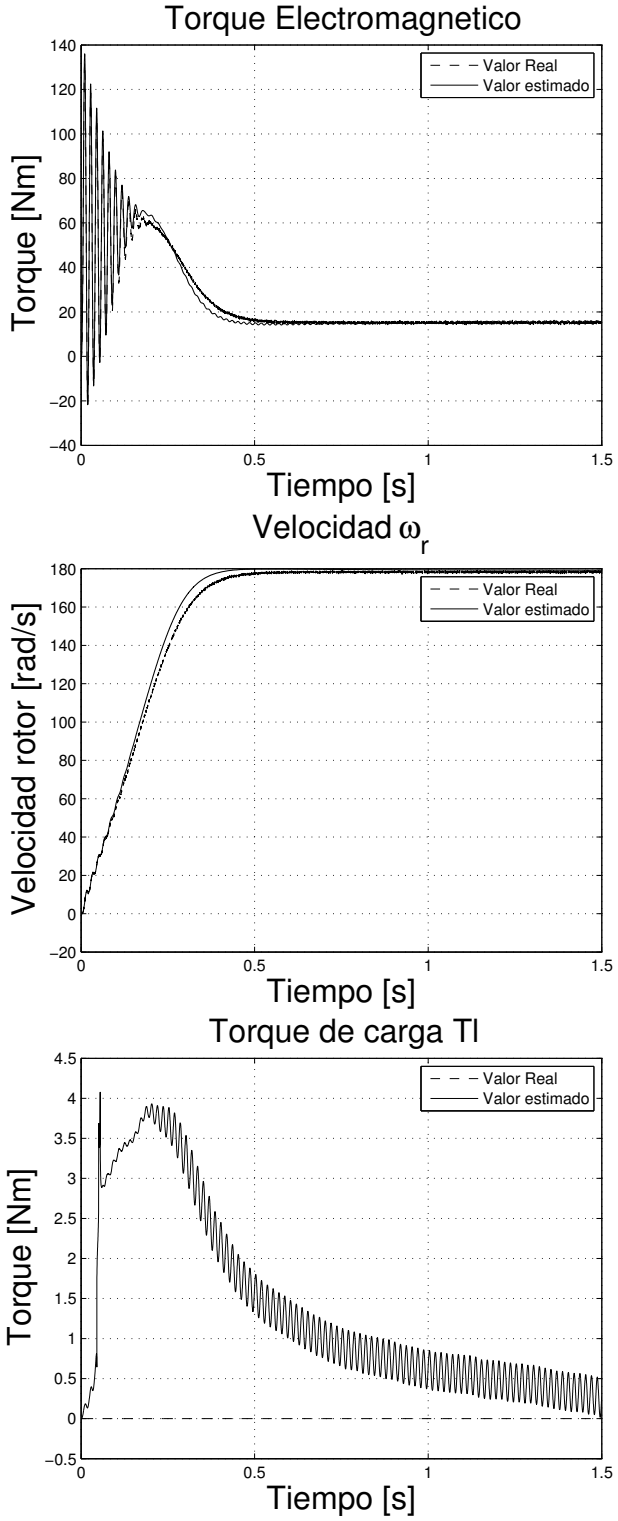


Figura 3.27: Variables reales y estimadas. Se muestra: 1. Torque electromagnético [Nm] 2. Velocidad del rotor [rad/s] 3. Torque de carga [Nm]

Al igual que en las gráficas para el filtro H_∞ con $L_k = I_{6\times6}$ se observa discrepancias entre las señales reales y estimadas de torque electromagnético y velocidad al momento de la transición entre el arranque y el esatdo estable. Sin embargo, esa diferencia se reduce y es imperceptible a partir de $t = 0{,}5s$. Para el torque de carga se observa un pico inicial que corresponde a aproximadamente el 35 % del torque nominal. Sin embargo, este se atenua y se observa que la señal estimada tiende a 0, es decir a la referencia real. Nuevamente existen oscilaciones durante el proceso.

3.2.6. Filtro con $L = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ - Estados estable sin perturbación

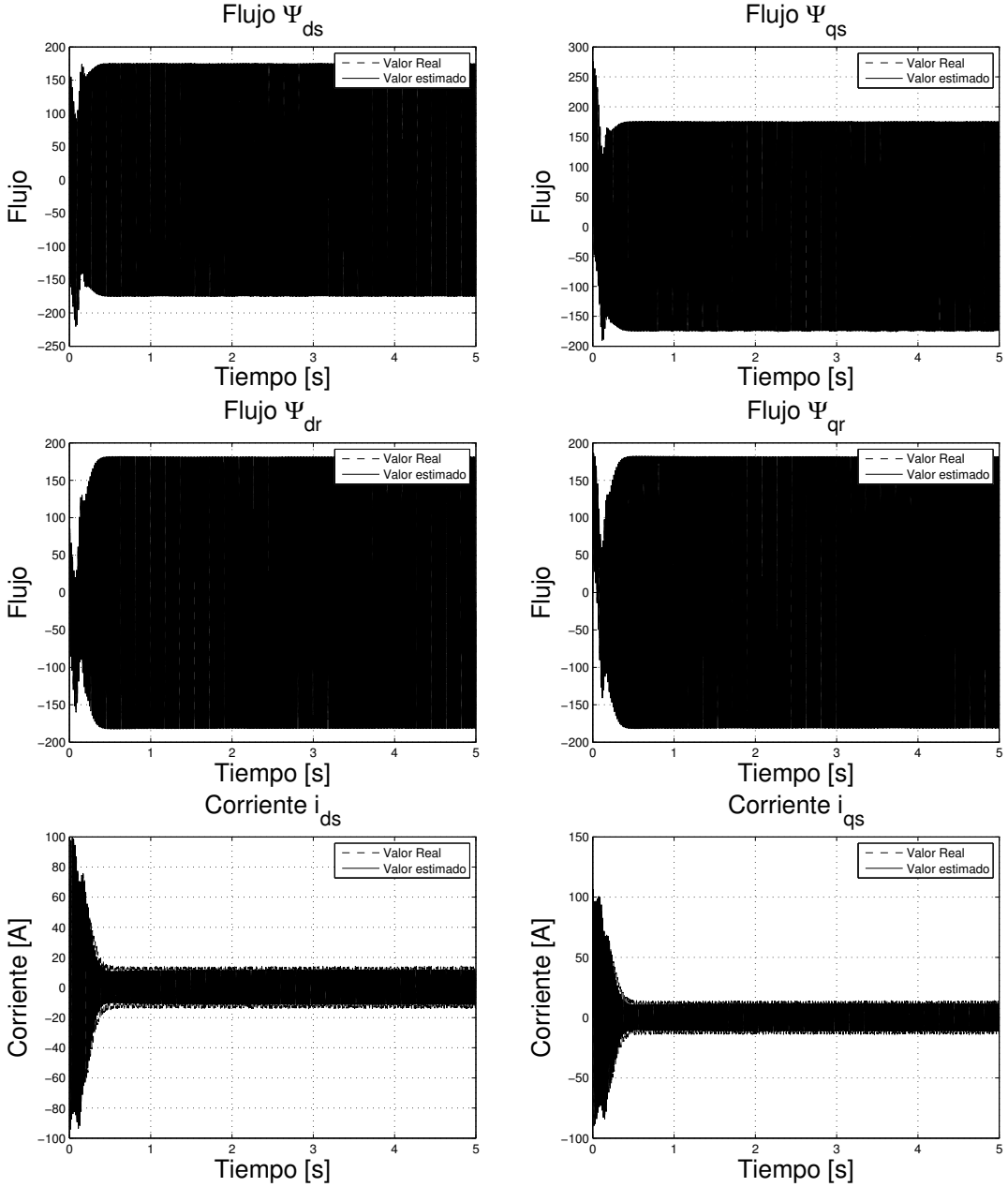


Figura 3.28: Filtro H_∞ con $L = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ en estado estable.

En las gráficas de flujo y corriente se observa que la estimación no tiene problemas en estado estable. Las lineas reales y estimadas no se distinguen. El filtro trabaja de la manera esperada incluso cuando su desempeño no esta centrado en la estimación de éstas variables.

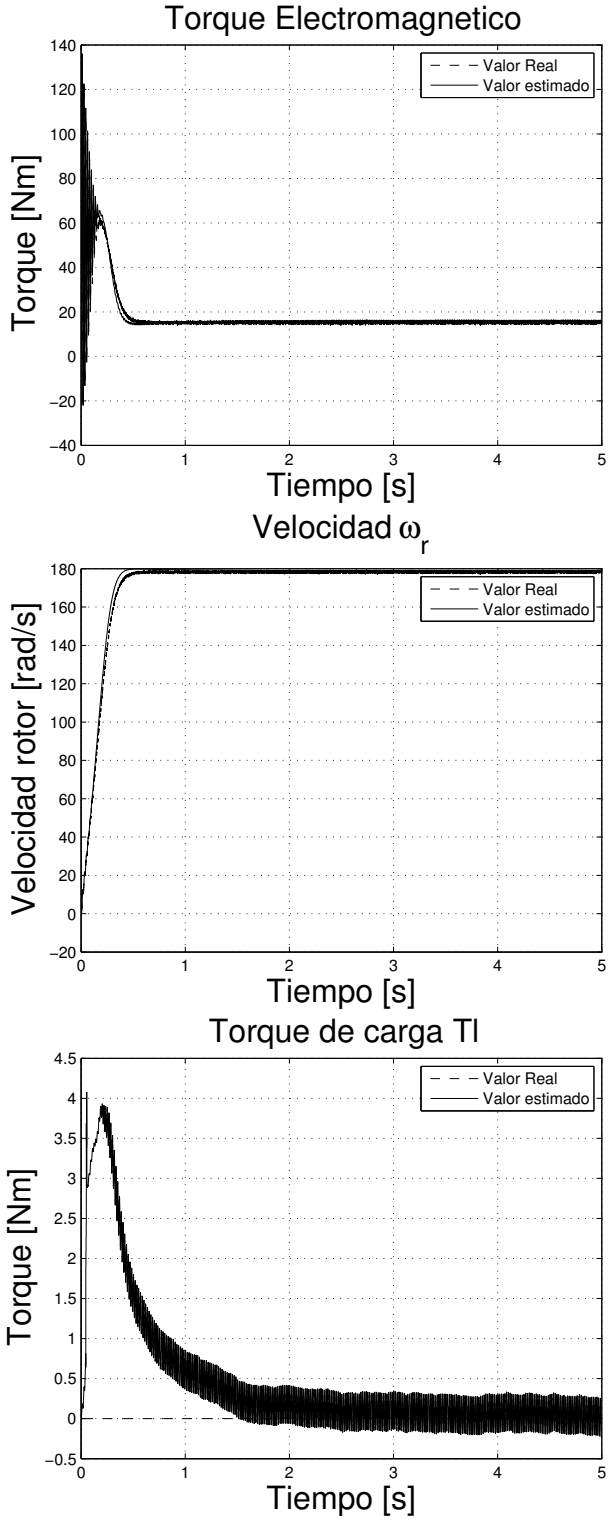


Figura 3.29: Filtro H_∞ con $L = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ en estado estable

En el torque electromagnético y velocidad se observa una pequeña diferencia durante el transitorio, sin embargo este error se corrigue rápidamente. La estimación es correcta ya en estado estable. En el caso del torque de carga se aprecia existe un pico inicial que se atenúa y decae hasta el valor real de torque. Sin embargo, se observan oscilaciones sostenidas durante el proceso. Estas oscilaciones se dan alrededor del punto referencial.

3.2.7. Filtro con $L = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$- Perturbación

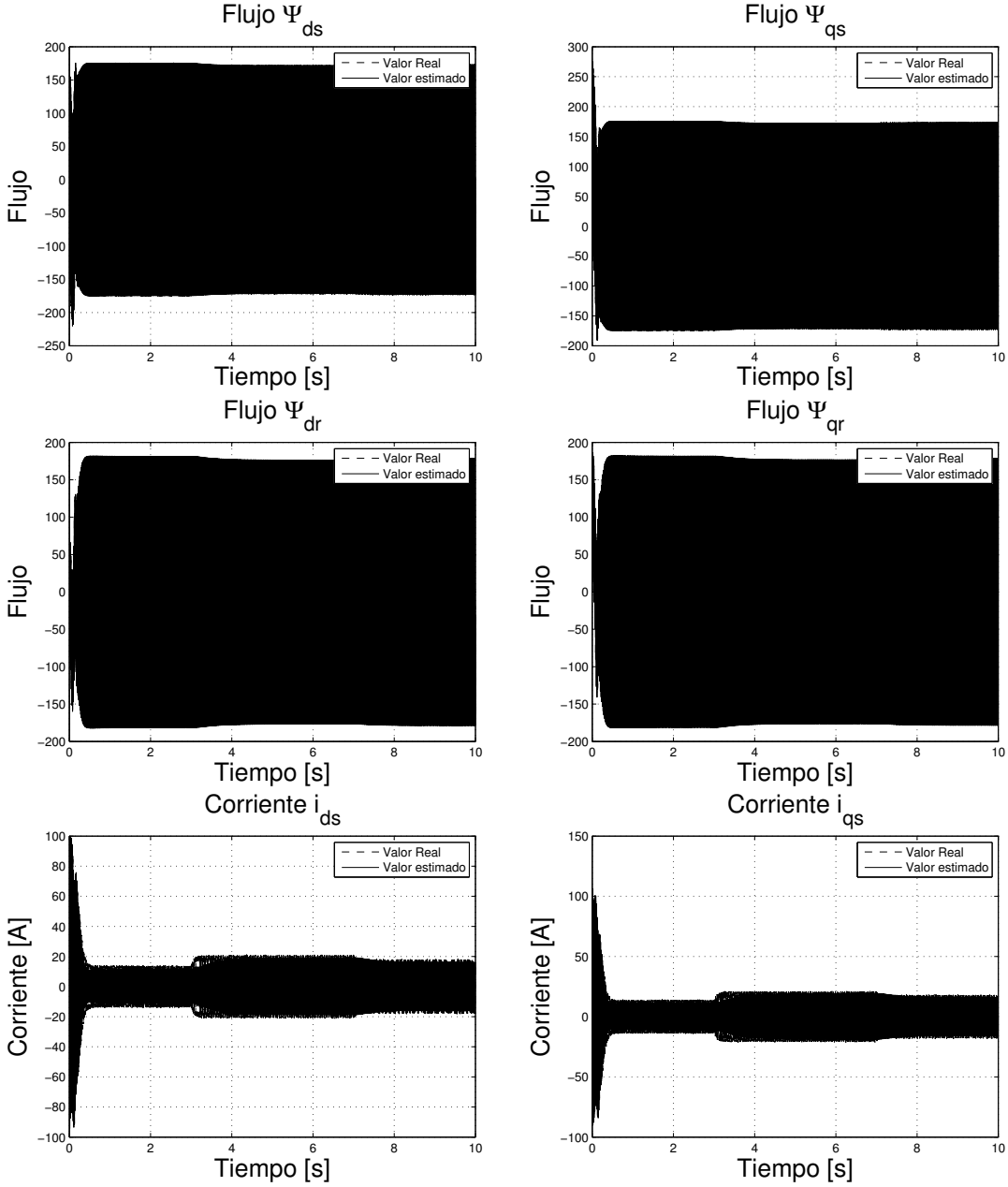


Figura 3.30: Filtro H_∞ con $L = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ con perturbación.

Se puede observar que el filtro responde a las perturbaciones de forma correcta, ya que al aumentar el torque de carga en $t = 3s$ se observa un aumento en las corrientes de la máquina. De igual manera al producirse una nueva perturbación $t = 7s$ donde el torque de carga se reduce se aprecia que las corrientes también se reducen.

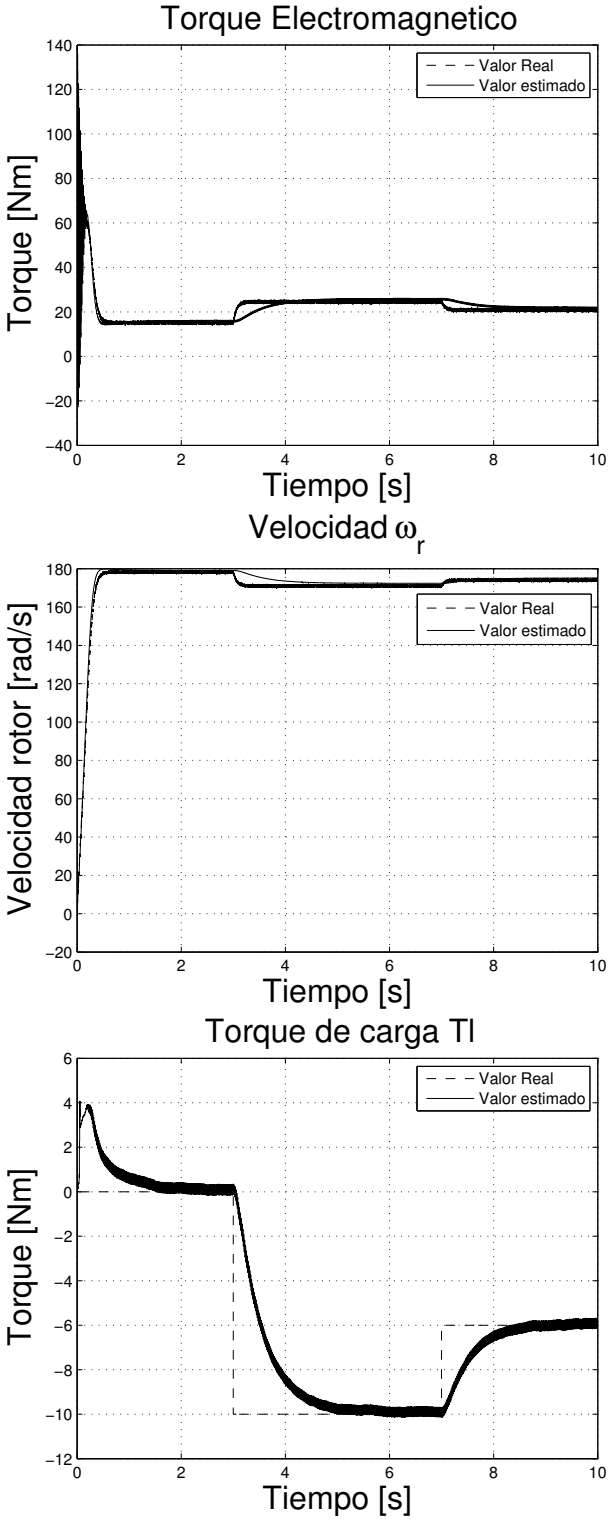


Figura 3.31: Filtro H_∞ con $L = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ con perturbación

En las gráficas de torque electromagnético y velocidad del rotor se observa que la respuesta del sistema es lenta y al ocurrir una perturbación la variable estimada no reacciona con la misma rapidez que la variable real. Sin embargo, luego de un tiempo relativamente corto en estado estable las dos señales concuerdan perfectamente. De igual manera en el caso del torque de carga se observa que la señal tiene una reacción lenta, comparable con un sistema de segundo orden sobreamortiguado, ya que tampoco presenta picos u oscilaciones. En estado estable se observa ruido producto de las oscilaciones alrededor del punto real de referencia.

3.2.8. Filtro con $L = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$-Vista detallada de los efectos de la perturbación

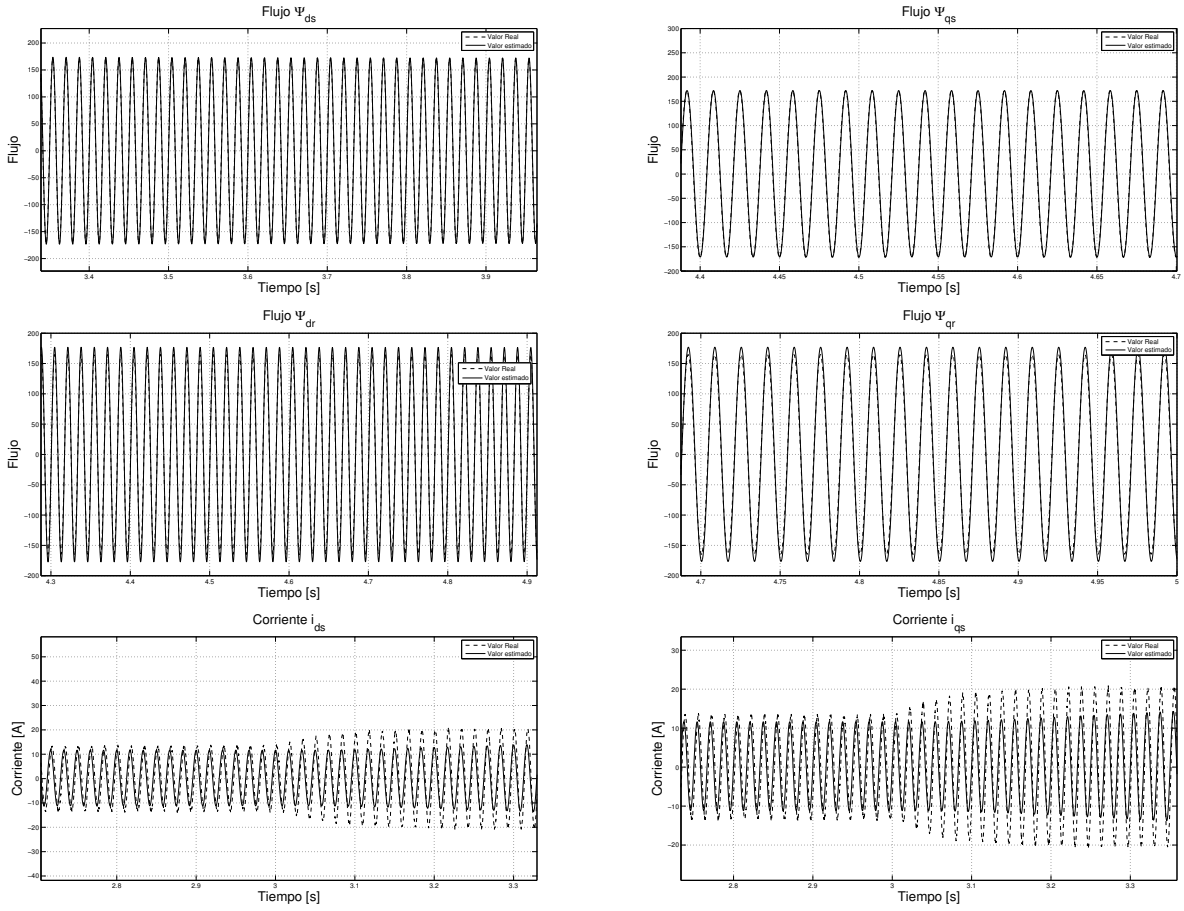


Figura 3.32: Filtro H_∞ con $L = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ con perturbación- Vista detallada de los efectos de la perturbación

Se aprecia que los flujos de la máquina son perfectamente estimados. Las gráficas reales y estimadas prácticamente se sobreponen en todo momento, incluso cuando ocurre la perturbación en $t = 3s$. Por otra parte en las corrientes se observa que las señales estimadas no

responden igual que las señales reales. Las corrientes estimadas tienen menor amplitud de las reales, sin embargo se puede apreciar también que las señales estimadas mantienen la tendencia a aumentar hasta igualarse a las señales reales. Su respuesta dinámica es más lenta.

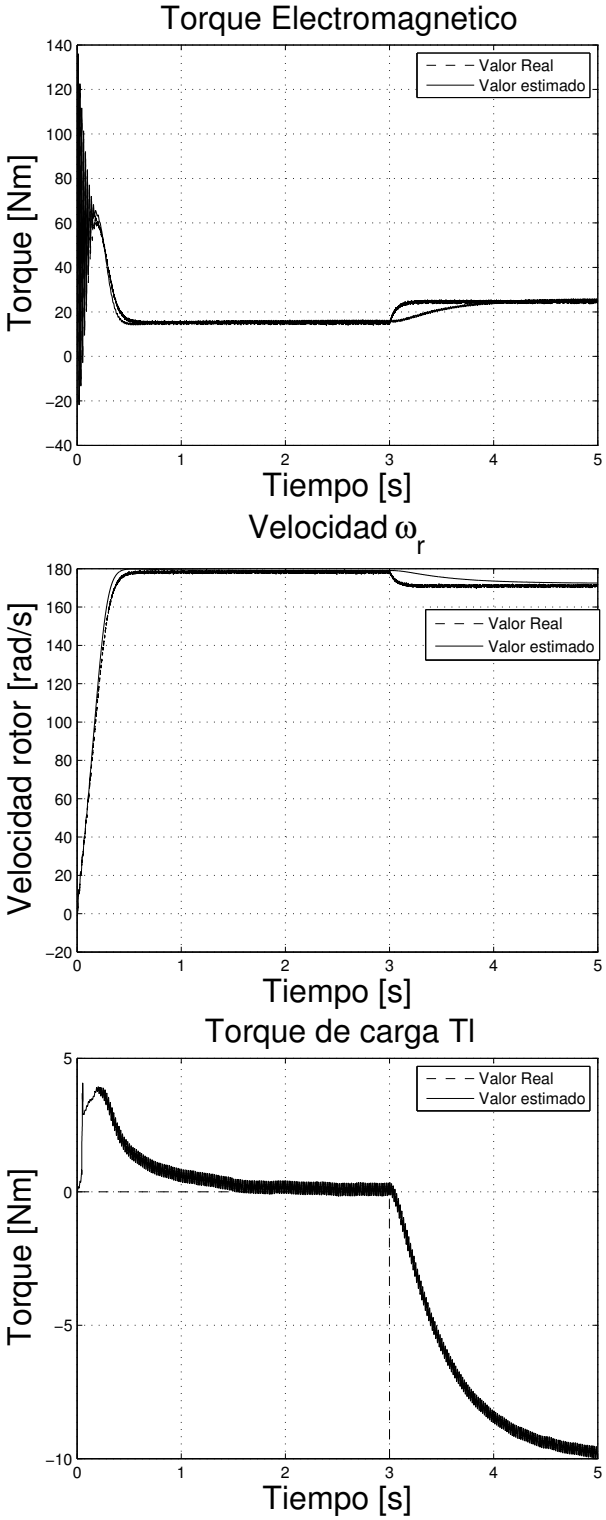


Figura 3.33: Filtro H_∞ con $L = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ con perturbación-Vista detallada de los efectos de la perturbación

En las gráficas de torque electromagnético y velocidad del rotor se observa como el filtro reacciona de forma lenta a las perturbaciones pero tiende a estabilizarse al final una vez que se alcanza el estado estable. Esta diferencia se puede apreciar como una reacción sobreamortiguada de un sistema de segundo orden. En el torque de carga se observa un comportamiento similar. Al inicio se observa un pico que se atenua y al momento de la perturbación el sistema responde de forma lenta (aproximadamente 2s antes de alcanzar el valor final) Sin embargo, no se observan picos u oscilaciones amortiguadas. En su lugar se observan pequeñas oscilaciones sostenidas incluso una vez que se alcanza el valor final de referencia.

De los escenarios simulados anteriormente: arranque del motor, estado estable y perturbaciones se concluye que la respuesta del filtro es favorable en los dos casos. En lo referente estimación de flujos y corrientes prácticamente no hay diferencia entre una y otra estimación. Las dos como se observan tienen excelentes estimaciones en los flujos pero respuestas lentas en el caso de la estimación de corrientes. En lo referente a estimación de torque electromagnético y velocidad del rotor se observa que los dos filtros tienen igual comportamiento. No se observa desviación en estado estable pero la respuesta transitoria luce lenta. En el caso del torque de carga las dos respuestas son muy similares, se tiene un pico al inicio de la estimación, que bien puede ser producto del proceso de arranque del motor. Luego, se tienen un decaimiento hacia el valor de referencia. Sin embargo, el decaimiento mencionado se produce con pequeñas oscilaciones que hacen ver ruidosa a la señal estimada de torque. Las oscilaciones sostenidas son aproximadamente del 10 % del torque nominal por lo que pueden llegar a ser significativas aunque como ya se ha resaltado el valor medio de dichas oscilaciones es el valor real de la variable estimada (torque de carga). Esto en términos generales de la simulación. Dado que no se aprecian mayores diferencias en las simulaciones se optará por el modelo del filtro con $L_k = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ para la etapa de implementación. Esto debido a la menor carga computacional que requiere y a los no indicios de disminución significativa en el desempeño del filtro.

Al igual que en el caso del filtro Kalman, los resultados de simulación muestran que no hay mayor diferencia entre las dos condiciones y se opta por la más sencilla a la hora de la implementación. No obstante, se debe tener en cuenta un detalle importante. En el primer caso, con $L_k = I_{6\times 6}$, el filtro tiene $\gamma = 0{,}4$ es decir la función de costo (2.51) es menor a $\gamma^2 = 0{,}16$. En el segundo caso, con $L_k = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$, se utiliza $\gamma = 0{,}2$ de donde se desprende que la función de costo debe ser menor a $\gamma^2 = 0{,}04$. Se aprecia que en el segundo caso el error en términos absolutos es menor ya que la condición de estimación es mucho más relajada en el sentido que el desempeño se centra solo en 2 de los 6 estados totales. Al reducir la complejidad del filtro se puede puede mejorar la condición sobre el error. Se hicieron ensayos con $\gamma < 0{,}4$ en el primer caso pero la simulación empezaba a generar problemas numéricos sobretodo al momento de invertir la matriz R_w (2.85). Por esta razón se conservó el mejor resultado posible. Una dinámica similar se dio con el segundo filtro, pero en este caso el mínimo alcanzado fue $\gamma = 0{,}2$

Hasta ahora, la simulación muestra una buena estimación de los estados del modelo i.e. flujos $qd0$ de estator y rotor, velocidad del rotor y torque de carga tanto con el filtro Kalman extendido como con el filtro H_∞ extendido. Cada uno de los filtros tiene sus particularidades, pros y contras sin embargo se debe mantener siempre presente que solo se trata de un resultado

simulado. Por ejemplo, al incluir parámetros sin incertidumbre en el modelo de la máquina se deja de lado una de las fortalezas atribuidas al filtro H_∞, su robustez ante incertidumbre en los parámetros. De igual manera en la simulación se utilizaron fuentes de ruido blanco en los dos casos. El ruido blanco es una de las hipótesis que se tienen al momento de proponer un filtro Kalman. El filtro H_∞ por su parte no requiere que el ruido satisfaga una condición en particular, excepto tener energía finita.

El comentario previo se debe a que en simulación el filtro Kalman extendido parece tener un mejor desempeño que el filtro H_∞ extendido sobretodo en presencia de perturbaciones. Sin embargo, como ya se puntualizó los resultados de simulación deben ser considerados como un punto de partida válido que fija ideas y orienta en la determinación de los parámetros, i.e matrices Q_k, y R_k en el caso de Kalman y los parámetros γ y G_k en el caso de H_∞ pero no como un resultado concluyente ya que existen una serie de eventos y condiciones de operación que no se consideran a la hora de programar y ejecutar una simulación. Se esperará tener los resultados de implementación real para determinar definitivamente cual de los dos filtros tiene mejor desempeño.

Para finalizar, es importante comentar acerca de la estimación de torque de carga. Esta variable, que se modela como una perturbación de acuerdo a (2.41), donde se la asume constante en el tiempo fue correctamente estimada en los casos simulados. Pese a lo simple de su modelamiento esta variable respondió bastante bien lo que indica que tanto el diseño del filtro como los algoritmos utilizados estan trabajando bien y son sumamente poderosos para estimación. Adicionalmente, se resalta que la estimación de torque muestra una respuesta muy similar a la de un sistema de segundo orden, con respuestas subamortiguadas en el caso del filtro Kalman y respuestas sobreamortiguadas en el caso del filtro H_∞. No obstante, se debe tener presente que en sistemas reales, el torque de carga no puede cambiar como una función escalón ya que al ser más bien una variable mecánica tiene una dinámica más lenta. Por esta razón muy probablemente no se observe el mismo comportamiento en el torque estimado por un filtro real y el torque estimado por filtros simulados.

De los resultados obtenidos en simulación se decide que los filtros a implementarse serán:

- Filtro Kalman extendido con base en la aproximación lineal en T_s debido a su menor complejidad computacional y alto desempeño.
- Filtro H_∞ extendido con $L_k = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ como la matriz de combinación lineal de estados debido a la considerable reducción de cálculos que se requieren sobretodo en la inversión de la matriz R_w y a la no disminución en su desempeño respecto al filtro H_∞ con $L_k = I_{6\times 6}$.

4. Implementación, pruebas de campo y sintonización de parámetros.

4.1. Generalidades sobre el software VisSim, DSP F2808 y modulo de potencia de Francecol

En este capítulo se revisa la implementación de los filtros simulados en la sección anterior. Se comienza mencionando los aspectos más importantes tanto del hardware como el software utilizado, es decir el módulo de potencia, el DSP de control, el diseño de placas de acondicionamiento de señales para muestreo de voltaje y el software de programación y monitoreo VisSim. Luego se discute la programación e implementación del algoritmo en hardware antes de proceder a las pruebas reales. En esta sección también se detalla la secuencia que se debe seguir en VisSim y en Code Composer Studio para poder grabar el DSP y mantener el programa ejecutándose por tiempo indefinido. Una vez establecidos todos los parámetros y requerimientos de diseño se proceden a las pruebas con el motor. Se generan los datos de estimación para diversas condiciones de trabajo a saber, alta frecuencia en el voltaje de alimentación (más de 40 Hz), frecuencia media (25-40Hz) y baja frecuencia (menos de 25Hz). En las tres condiciones dadas se realizan pruebas con perturbaciones. Se aumenta y disminuye la carga en el motor. Adicionalmente se prueba el funcionamiento de los filtros cuando la perturbación es una variación en la frecuencia de alimentación. Se registran las estimaciones de flujos $qd0$ de rotor y estator, velocidad del rotor, torque de carga y además se registran las señales de voltajes y corrientes $qd0$ en el estator muestreadas por el DSP durante la operación de los filtros.

Se compara el desempeño de los dos filtros considerando no solo la calidad de la estimación sino la robustez del algoritmo, el nivel de oscilación, la respuesta a perturbaciones, la complejidad de los cálculos efectuados, la carga computacional entre otros.

4.1.1. VisSim

VisSim es un software de programación, simulación y modelamiento basado en diagramas de bloques. Su interfaz gráfica es muy similar a la Simulink de Mathworks [24]. Entre sus librerías cuenta con bloques para anotaciones (enrutamiento), bloques generadores (sources), bloques de presentación (sinks), bloques aritméticos, booleanos, de operaciones matriciales, no lineales, optimización, de valores aleatorios, pero el de más utilidad es el de punto fijo (fixed point). Esta última librería permite programar algoritmos y secuencias en punto fijo. Dentro de la librería mencionada existen las siguientes operaciones básicas:

- Funciones trigonométricas: seno, coseno y tangente.
- Operaciones básicas: suma, multiplicación, multiplicación por una constante, valor absoluto, signo, división.
- Funciones lógicas: “y”, “o” “not”, “or”, “xor”.
- Comparadores lógicos: mayor que, menor que, igual, distinto de, negación.
- Controladores: regulador PI, regulador PID, filtros mediante función de transferencia.

- Funciones no lineales: saturador, retraso unitario, merge (posibilidad de elegir la salida que se quiere), convertidor (permiten mover el punto fijo)

Mediante la combinación de los recursos disponibles en la librería Fixed Point y algunos de los recursos de las demás librerías es posible desarrollar gráficamente programas que controlen un proceso. Estos programas pueden ser de todo tipo desde complejos accionamientos de motores de inducción hasta sencillos filtros digitales de audio. Dichos programas posteriormente se pueden compilar y descargar en un DSP de Texas Instruments.

Por otra parte, las librerías adicionales mencionadas de VisSim permiten desarrollar modelos para simulación de diversos procesos físicos. Se pueden modelar complejos sistemas desde controles de niveles en tanques hasta motores y bandas transportadoras.

La principal característica de VisSim es la interacción que tiene con el programa Code Composer Studio de Texas Instruments. Debido a esta cualidad es posible programar secuencias o algoritmos en diagramas de bloques y el software genera un archivo de código C que es posible compilarlo y descargarlo en un microprocesador o DSP de Texas Instruments. En este proyecto se utiliza el microprocesador en tiempo real de 32 bits y punto fijo F2808. Dependiendo de la complejidad y espacio de memoria que ocupa el algoritmo programado en VisSim y de la capacidad de memoria del dispositivo se tiene una mayor o menor complejidad en la descarga del programa hacia el dispositivo. Una vez que se compila y se descarga el código al microprocesador es posible monitorear las variables internas del DSP en el computador. Para ello se utiliza el archivo .OUT que genera VisSim. En este caso se monitorean las seis variables de estado (flujos qd0 del estator y rotor de la máquina, velocidad del rotor y torque de carga). Además se observan las señales de voltaje y corriente del estator muestreadas por el dispositivo. La amplitud y frecuencia del voltaje de alimentación se controlan mediante potenciómetros que generan señales análogas de control. Adicionalmente se observa el porcentaje de uso del DSP que se requiere para ejecutar el algoritmo.

VisSim permite observar en tiempo real (vía conección USB entre el DSP y el computador) las variables del sistema que están siendo procesadas por el DSP, además, dichos datos pueden ser adquiridos en archivos .dat para su posterior análisis y procesamiento. Para lleva a cabo esto es necesario configurar el archivo para mantener corriendo el programa por tiempo indefinido. Esto se hace de la siguiente manera:

En el menú System selecione System Properties y en la pestaña Range marque las tres viñetas como se muestran la Figura 4.1. El campo Time Step permite definir el tiempo de muestreo $T_s = 1/F_s$. En la Figura 4.1 se observa $T_s = 2ms = 0{,}002s$. El procedimiento señalado es importante ya que uno de los errores más comunes al momento de usar VisSim es no configurar el archivo de esta manera y observar como el programa se ejecuta una sola vez y se detiene automáticamente.

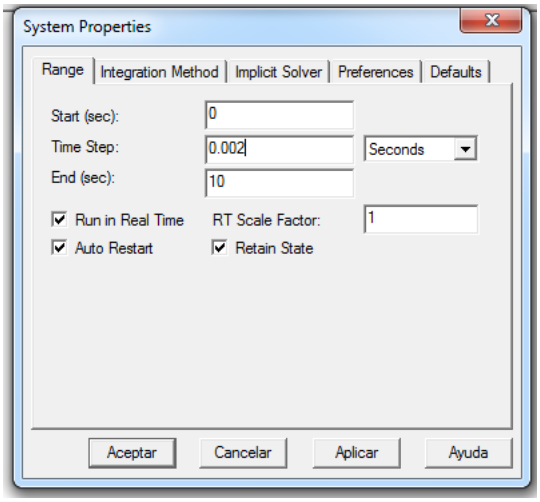


Figura 4.1: Configuración para ejecutar por tiempo indefinido un programa en VisSim

En cuanto a la programación en punto fijo, VisSim permite elegir el número de bits de precisión y enteros. En el caso presente se trabajó según la siguiente tabla:

Filtro	Bits de resolución	Enteros	Mínimo	Máximo
Kalman	32	14	-8192	8191.9999961853
H_∞	32	12	-2048	2047.9999990463

Cuadro 5: Precisión y decimales en la programación en punto fijo

Para el filtro H_∞ se utilizó mayor número de bits para decimales porque las entradas en las matrices involucradas en el algoritmo son pequeñas numéricamente. Esta condición es critica sobretodo en el algoritmo de inversión de la matriz R_w ya que al realizar multiplicaciones para calcular la inversa, las entradas de la matriz adjunta (necesaria en el algoritmo de inversión por calculo de adjuntos) son incluso más pequeñas numéricamente por lo que se requiere mejor resolución. En los ensayos con una resolución de 14 bits para enteros se encontró que algunas de las entradas se volvían cero por falta de precisión y el algoritmo no respondía como se espera en esta etapa.

Más adelante en la sección de implementación en hardware del algoritmo se mostrará como descargar el programa al DSP cuando su tamaño supera la capacidad de la memoria RAM disponible en el dispositivo. También se dará detalles sobre el proceso de compilación y guardado de los archivos de ejercución.

4.1.2. DSP TMS320F2808 de Texas Instruments

Este DSP genera las señales de control de la tarjeta de potencia de Francecol, muestreada y escala las señales de corriente y voltaje de las tres fases, establece la comunicación con el computador y además ejecuta todo el algoritmo de estimación. Sus características más sobresalientes son las siguientes [25]:

- Reloj interno de 100 MHz.

- 3.3V para alimentación
- CPU de 32 bits.
- 64K X 16 Flash, 18K X 16 SARAM.
- 16 salidas PWM.
- 16 canales ADC de 12 bits (entradas 0-3.3V).
- Tasa de conversión: 6.25 MSPS.
- Soporte para Code Composer Studio.
- Interfase para comunicación serial.
- Librerías para control digital de motores y potencia digital.
- Modos de ahorro de energía.

En el caso de los algoritmos implementados se utiliza una frecuencia de muestreo de 5kHz con un porcentaje de utilización del dispositivo del 60 % aproximadamente. En los dos filtros se utiliza el hecho que algunas de las matrices involucradas en los algoritmos son simétricas (matrices P y R_w básicamente) y se calcula solo una parte de los términos, ya que el resto de la matriz se completa a partir de esta propiedad. De esta forma se reduce significativamente el número de operaciones realizadas. Sin embargo, el porcentaje total del uso del dispositivo es alto si se considera que se ha optimizado el modelo, el algoritmo e incluso los métodos de calculo.

4.1.3. Módulo de potencia V4 de Francecol

Este módulo contiene tanto la estación de control como una estación de potencia. A la derecha de la Figura 4.2 se observa la estación de control y los conectores para entradas análogas, mientras que en la parte izquierda esta todo lo referente a la etapa de potencia [26, 27, 28, 29].

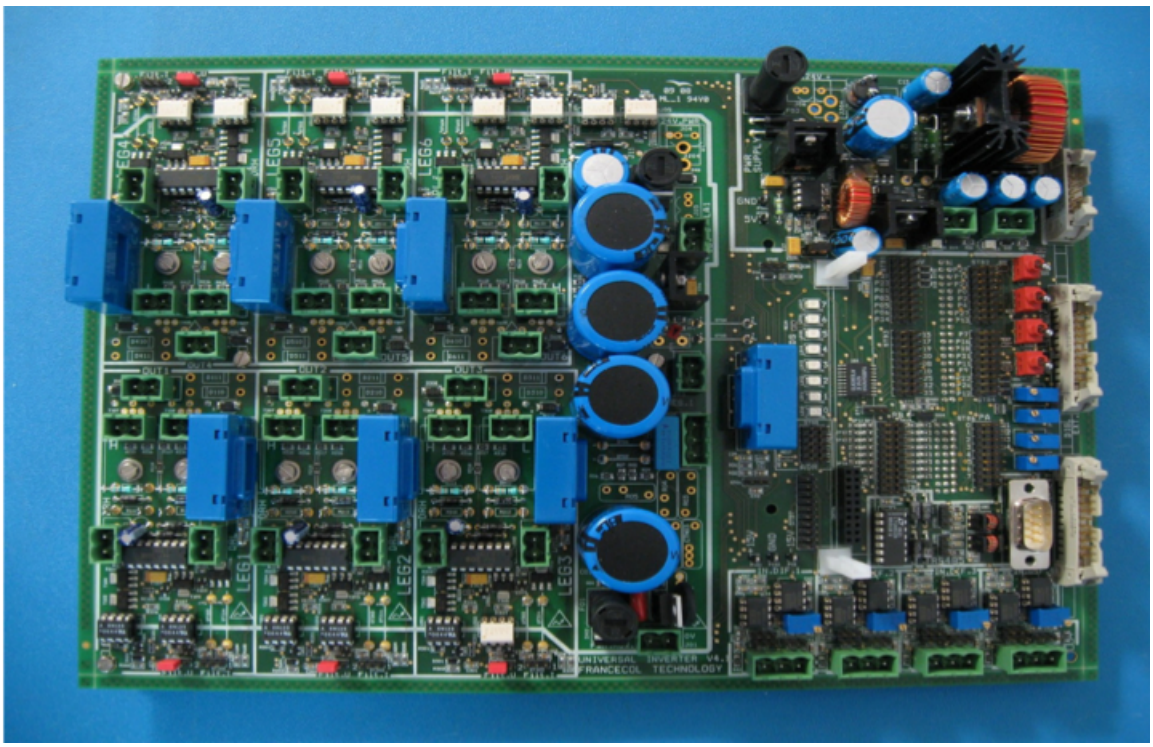


Figura 4.2: Módulo de potencia V4 de Francecol

En la sección correspondiente a control se destacan las extensiones de los pines más importantes del DSP, es decir; las entradas y salidas digitales, salidas PWM y entradas ADC. Las siete primeras entradas ADC estan conectadas internamente a los sensores de corriente (6 para corrientes AC y uno para corriente DC), las cuatro siguientes están accesibles externamente para conectar señales externas y así poder procesarlas en el DSP. Finalmente, las entradas restantes están conectadas a potenciometros integrados en la placa (potenciometros azules en la parte media derecha de la Figura 4.2), que permiten manipular manualmente dichas entradas al DSP. En este caso se diseñó una adaptación externa para poder muestrear las voltajes de fase del motor a través de tres de las entradas disponibles para señales externas. Previamente, los voltajes del motor tuvieron que ser acondicionados a los requerimientos de las entradas del DSP, es decir reducir su amplitud y aumentar la componente de DC para evitar voltajes negativos. Adicionalmente, a la entrada de los cuatro puertos ADC disponibles (borneras verdes en la parte inferior derecha de la Figura 4.2) existe una etapa de amplificación que puede ser controlada mediante jumpers. La ganancia se fija en 0, 20, 40, 60 y 80 dB según el requerimiento. En este caso se utilizó una configuración de 0dBs. No obstante para futura referencia se muestra la tabla de configuración 6. La x significa conección en el jumper.

Jumper	0dB	20dB	40dB	40dB	60dB	80dB
1		x			x	
2			x			x
3				x	x	x

Cuadro 6: Configuración de jumpers para fijar ganancia en entradas ADC

En la misma etapa de control existen conectores accesibles a las fuentes de 0, 5, 15 y -15V en caso de requerirlas.

De igual manera, las salidas PWM del DSP estan conectadas a la tarjeta de control y son las que comandan la operación de los conmutadores de potencia. En este módulo existen conectores que permiten conección serial RS-232, conección de entradas análogas y conección para encoder, en caso de requerirse hacer control en lazo cerrrado con realimentación de velocidad.

En la sección de potencia de la tarjeta existe una entrada de voltaje de potencia (voltaje DC). La potencia máxima del circuito es 600V y 40A. Seis de las salidas de voltaje son AC y una salida de voltaje es DC. Para los commutadores se utilizan IGBTs comandados como ya se mencionó por medio de las salidas PWM del DSP. Toda la etapa de potencia está aislada de la de control mediante optoacopladores. En la tarjeta de potencia existen conectores para monitoreo de las señales de control.

Así también, en la etapa de potencia están los sensores de corriente. Éstos no son más que transformadores, que mediante inducción generan un voltaje proporcional a la corriente que circula por el cable que se debe enrrollar a su alrededor y al número de vueltas que se dan. En este caso se enrrollaron tres vueltas alrededor de cada transformador. El voltaje inducido se muestrea con las primeras siete entradas ADC del DSP. En la Figura 4.3 se muestra una imagen de la configuración para medir corriente.

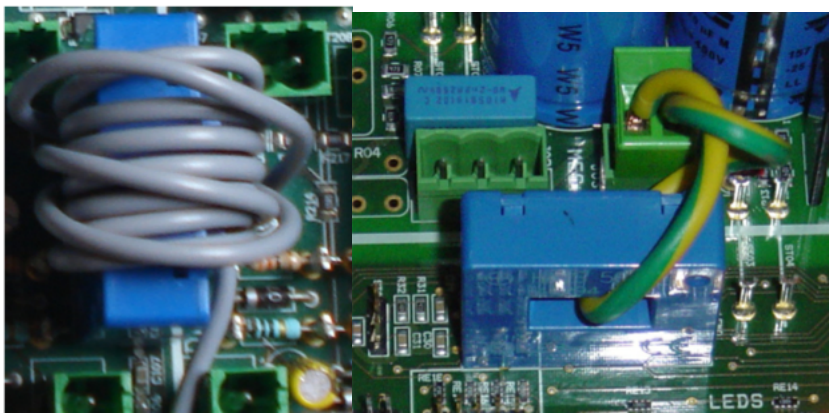


Figura 4.3: Configuración para medición de corriente

En cuanto a las entradas ADC accesibles del equipo se debe mencionar un cambio que se realizó en la placa original. Inicialmente se muestreaba la señal, pero esta mostraba una gran cantidad de ruido de alta frecuencia (950 kHz aproximadamente). El ruido era comparable a la

amplitud de la señal análoga . Después de realizar el seguimiento necesario de las señales desde la etapa amplificación (en la placa) hasta los espadines disponibles para monitoreo (antes de la entrada al DSP) se notó que el ruido provenía de los amplificadores no conectados a tierra. Para resolver este problema se procedió a conectar un filtro pasa bajas (RC) justo a la salida de los amplificadores; entre la salida de los mismos y tierra. De esta manera el ruido se filtraba y pasaba la señal acondicionada. Debido a que la señal muestreada es de baja frecuencia (menos de 100Hz) no existe mayor alteración de la misma. La frecuencia de corte del filtro es de $1kHz$, con $R = 10k\Omega$ y $C = 0{,}1\mu F$.

Adicionalmente, para este proyecto se requirió diseñar una pequeña tarjeta de acondicionamiento de señales. Se midió directamente el voltaje de cada una de las fases del motor. Dicho voltaje (70Vrms) pasó por tranformadores que reducían la señal (a 3.5Vrms). Dicho voltaje se reducía y se le sumaba un valor de voltaje DC de forma tal que la señal oscile entre 0 y 3V antes de conectarla a una entrada ADC del DSP.

4.1.4. Hardware adicional

Adicionalmente, para este proyecto se requirió diseñar una pequeña tarjeta de acondicionamiento de señales. Se tomaba una muestra directamente el voltaje de cada una de las fases del motor. Dicho voltaje (aproximadamente 70Vrms) pasaba por transformadores que no solo reducían la señal (a 3.5Vrms) sino que también aislaban todo la etapa de potencia de la etapa de control. Dicho voltaje se atenuaba nuevamente mediante amplificadores operacionales calibrados para el efecto y se le sumaba análogamente un valor de voltaje DC de forma tal que la señal oscile entre 0 y 3V antes de conectarla a una entrada ADC del DSP. Como protección, las salidas acondicionadas tenía un diodo zener de 3.3V para evitar cualquier sobrevoltaje que pueda dañar las entradas del DSP. Se utilizaron amplificadores operacionales TL-084 de Texas Instruments debido a su alto desempeño y bajo costo. Para calibrar las ganancias y valores de desviación se utilizaron potenciómetros de precisión que permitan un ajuste fino particular para cada una de las fases. El detalle del diseño se da en la sección de Anexos. Las señales de voltaje y también de corriente muestreadas en el DSP son centradas alrededor de cero y reescaladas digitalmente dentro del mismo dispositivo. Los valores de escalamiento son calculados experimentalmente de forma tal que el valor rms medido y muestreado en el DSP sean los mismos. Para ello se tomaron muestras de datos del voltaje y corriente muestrado y se procesaron hasta que el valor rms de dichas muestras coincidan el valor rms medido. Con esto se terminó el acondicionamiento de la señal para utilizar

4.2. Implementación en hardware de filtros Kalman extendidos y $\mathbf{H}_\infty$

Inicialmente se programaron los filtros utilizando la librería de operaciones matriciales de VisSim. Sin embargo, no se consideró que ésta estaba más orientada a aplicaciones de simulación ya que el formato era punto flotante. Al momento de descargar y ejecutar el filtro en el DSP se tuvo una serie de inconvenientes, ya que el algoritmo de inversión de matrices de VisSim era poco adecuado cuando se tiene matrices cuyas entradas muy pequeñas numéricamente. Una vez solucionado esto mediante una actualización de software (el fabricante actualizó su programación) se logró compilar y descargar el filtro. Sin embargo al momento de la ejecución el dispositivo no fue capaz de ejecutar las operaciones requeridas y la frecuencia máxima de trabajo fue de 400Hz, muy por debajo de los 5000Hz esperados. La razón de este

mal rendimiento era las múltiples conversiones que debía hacer el DSP de punto flotante a punto fijo durante las iteraciones. Ésto limitaba seriamente el desempeño del DSP. Para solucionar este problema se optó por descomponer cada una de las operaciones matriciales en operaciones escalares entrada por entrada y se programó todo el algoritmo de esta manera. Para ello se utilizaron los bloques disponibles en la librería de punto fijo, en especial sumas y mutiplicaciones. Con esto se redujo la carga computacional pero complicó ampliamente la programación ya que las operaciones se volvieron tediosas y largas. Esto a su vez aumentó el tamaño del código, por lo que no fue posible descargar directamente el código a la memoria RAM del DSP. Fue necesario descargarlo a la memoria FLASH. Siendo la diferencia entre éstas dos que la memoria FLASH tiene mayor capacidad y no se borra al desenergizar el dispositivo. Por otra parte todo lo grabado en la memoria RAM se borra al apagar el equipo. Pese a esta diferencia relativamente sencilla, el procedimiento para compilar y descargar un programa a la memoria FLASH del DSP es más largo y tedioso que para descargarlo en la memoria RAM. A continuación se detalla el procedimiento a seguir:

- Una vez terminado el programa se procede a comprimirlo en un solo bloque final por medio de la opción Compound Block. Se selecciona el bloque comprimido y en el menú Tools se escoge la opción Code Gen como se muestra en la Figura 4.4

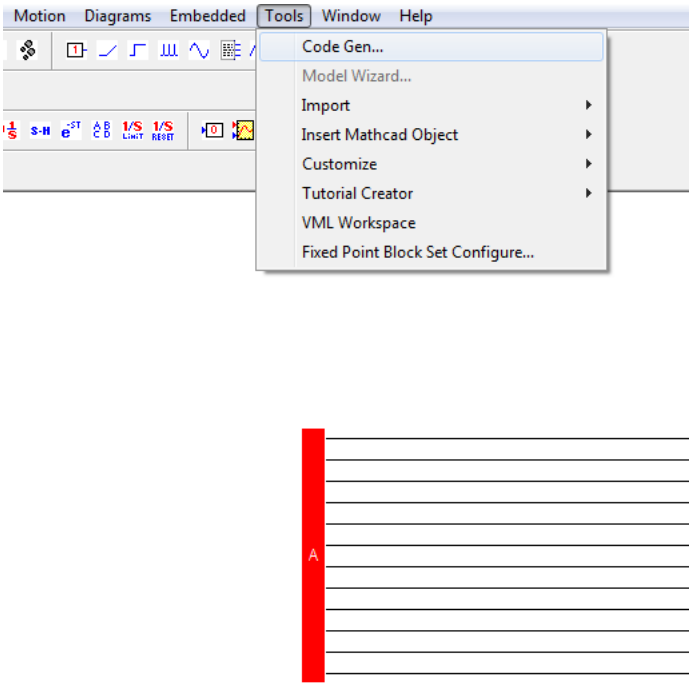


Figura 4.4: Generación de código

- En el cuadro que se abre a continuación se debe seleccionar el tipo de dispositivo y marcar las opciones Include VisSim Communication Interface y Target FLASH. En los campos Stack Size y Heap Size se recomienda utilizar 512 y 1024 respectivamente. Terminada la configuración dé click en la opción Code Gen y luego en Compile. Se abre una

consola negra donde se muestra el proceso. Espere que aparezca un mensaje indicando la finalización del mismo. La Figura 4.5 indica este procedimiento.

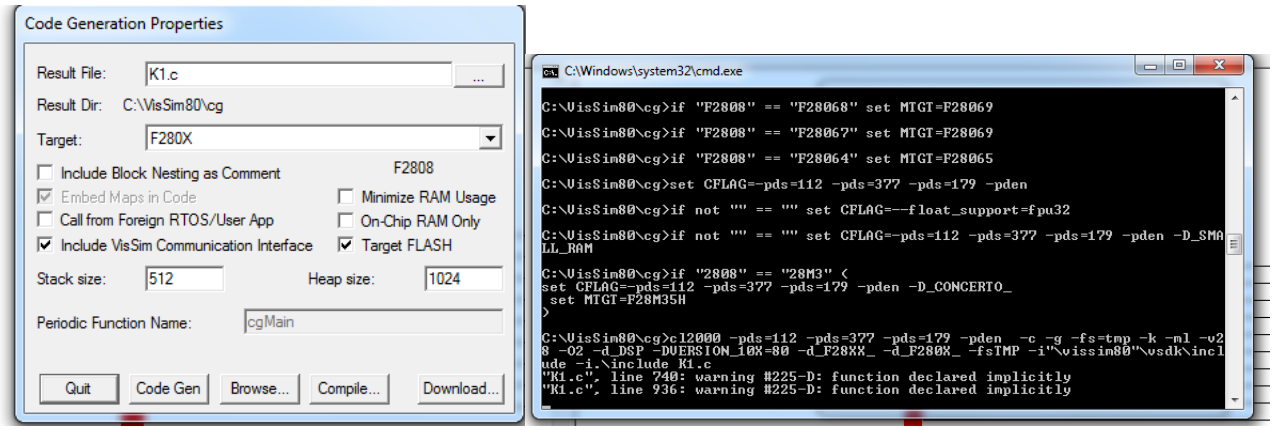


Figura 4.5: Configuración antes de compilar el código de un programa desarrollado en VisSim.

- A continuación se debe crear un nuevo archivo de VisSim. En el menú Embbeded seleccione el submenú F280x y la opción targetInterface. Se crea un bloque con el nombre del bloque compuesto que se compiló seguido de la terminación .OUT. A este bloque se conectan los sinks (salidas) que se requieran y/o deseen para visualizar las variables seleccionadas como salidas en el programa. En la Figura 4.6 se muestra un ejemplo.

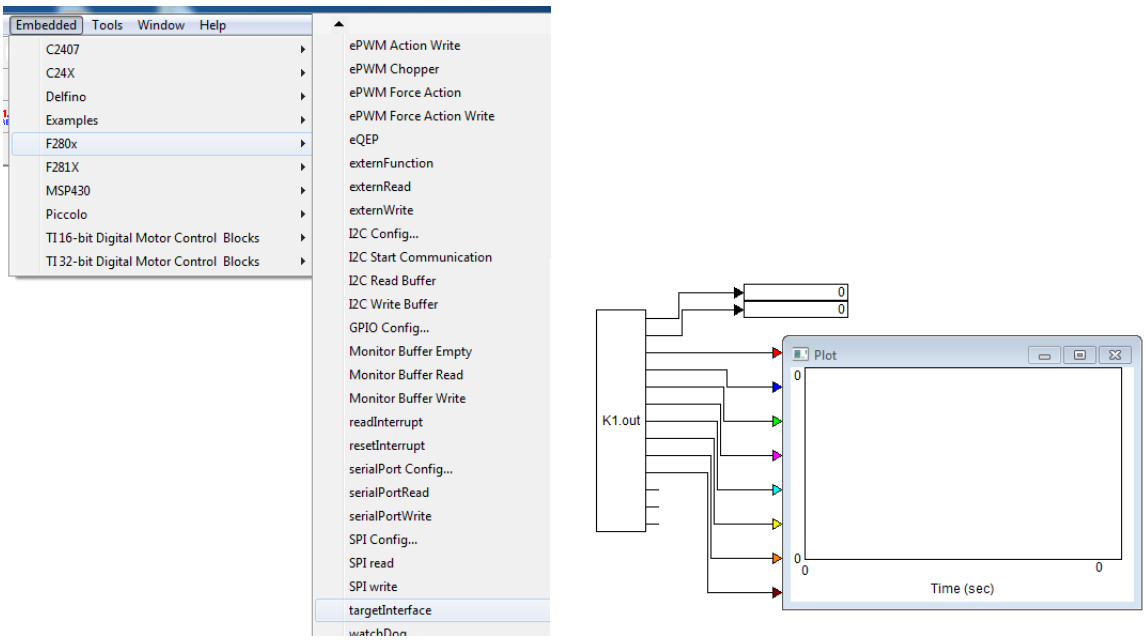


Figura 4.6: Creación de archivo VisSim para monitoreo de variables

- Vaya al menú System, opción System Properties y configúrelo como se explicó en la Figura 4.1. Adicionalmente, al dar doble click sobre el bloque .OUT asegúrese que en el cuadro Sample Rate (Hz) esté correctamente configurado según los requerimientos del programa. En este caso se va a trabajar con $F_s = 5000Hz$. La Figura 4.7 muestra este procedimiento. Además en el cuadro desplegado se muestra un resumen del comportamiento del bloque, el nombre del programa que ejecuta y el número de entradas y salidas. Guarde el archivo con el mismo nombre que el programa original pero con la terminación "-d" incluida. Por ejemplo si el programa original se llama filtro1, el programa para monitoreo deberá llamarse filtro1-d.

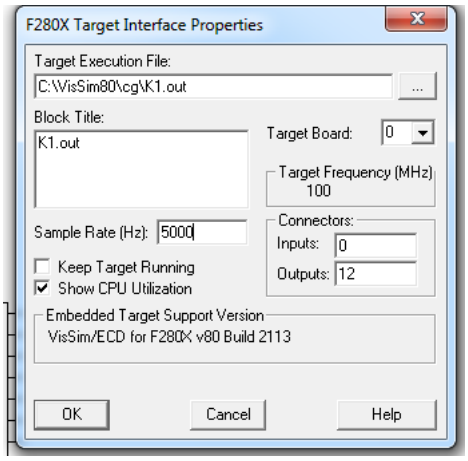


Figura 4.7: Configuración de archivo VisSim para monitoreo de variables

De esta manera se termina la compilación y configuración del programa de monitoreo en VisSim. Ahora se explicará como descargar el código a la memoria FLASH del DSP utilizando Code Composer Studio.

- Una vez iniciado el programa en el menú Target seleccione la opción New Target Configuration. En el cuadro que se abre seleccione Finish. A continuación en la esquina inferior izquierda del programa se abre un nuevo cuadro de diálogo. En éste seleccione la opción Texas Instruments XDS100v2 USB Emulator y TMS320F2808. Guarde la configuración. El procedimiento descrito se muesta en la Figura 4.8.

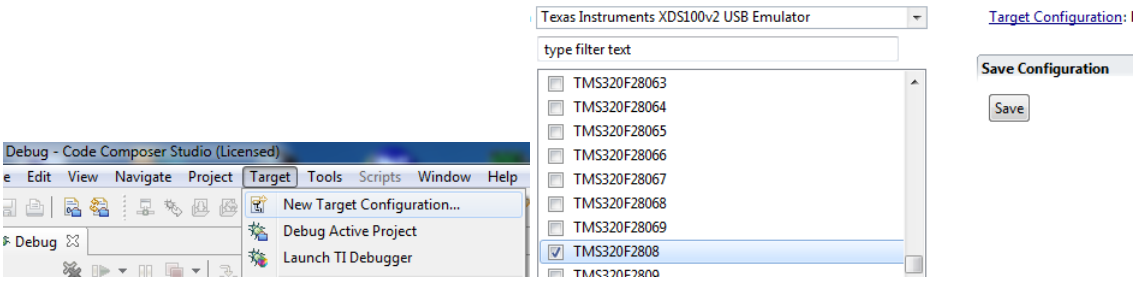


Figura 4.8: Configuración de programa en Code Composer Studio

- Finalizada la configuración de comunicación del programa con el dispositivo vaya nuevamente al menú Target y seleccione Launch TI Debuger. Una vez que se despliega la pantalla correspondiente, en el mismo menú Target seleccione Connect Target y espere confirmación. A continuación seleccione Load program y en el menú que se despliega busque la carpeta donde se creó el archivo .OUT previamente compilado con VisSim y seleccione el archivo correspondiente. Por default la carpeta requerida es VisSim80\cg en el disco C. Finalmente, dé click en Download y espere confirmación de descarga del archivo. Antes de salir seleccione la opción Disconnect Target para evitar conflictos posteriores con VisSim. La Figura 4.9 muestra lo explicado en el párrafo anterior.

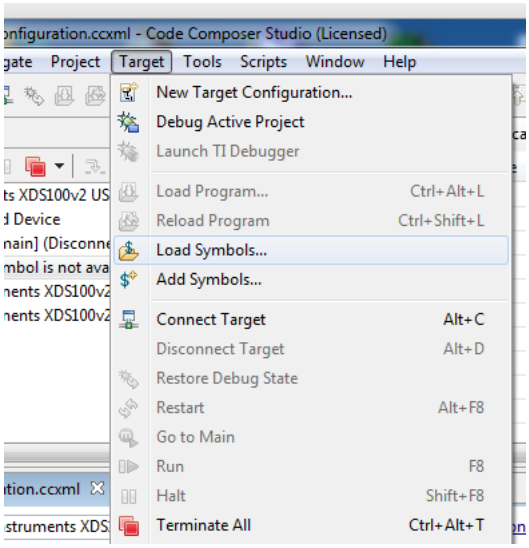


Figura 4.9: Descarga del programa en el DSP

En VisSim ejecute el programa mediante la opción Play.

4.2.1. Filtro Kalman extendido

Para la implementación física del filtro Kalman extendido se utilizó un DSP F2808 de punto fijo fabricado por Texas Instruments. Este DSP además de correr todos los algoritmos de estimación se encargaba del control de alimentación del motor mediante la generación de señales PWM a una frecuencia y amplitud definida por el usuario mediante dos potenciometros destinados para el efecto (accesibles en la tarjeta de control de Francecol). Así también, el DSP realiza el muestreo de las señales tanto de corriente como de voltaje. Estas señales son escaladas digitalmente a los valores reales en el motor dentro del mismo DSP.

Toda esta programación como ya se indicó se desarrolló en el entorno gráfico de VisSim mediante diagramas de bloques adecuados para los requerimientos de programación. En la sección anexos se dan detalles del programa para generar señales PWM y acondicionamiento de las señales de corriente y voltaje. Esta sección se concentrará en la programación del algoritmo del filtro.

El algoritmo descrito en las ecuaciones (2.31), (2.32), (2.33), (2.34), (2.1.3), (2.35) se implementó en VisSim. Sin embargo, debido a la caracteristica de punto fijo del DSP todas las operaciones matriciales debieron ser implementadas como operaciones escalares independientes. Esto porque VisSim no tiene una librería de operaciones matriciales en punto fijo, solo en punto flotante.

Inicialmente se programó el algoritmo en punto flotante (operaciones matriciales) debido a la simplicidad de la programación. Básicamente se hizo diagramas de bloques de las ecuaciones del algoritmo ya mencionado. Sin embargo, este tipo de programación se requería una gran cantidad de conversiones internas en el DSP (de punto flotante a punto fijo) que hacían que la frecuencia máxima de operación bordeara los 400Hz con el 95 % de la capacidad del DSP en uso. La frecuencia de operación mínima estimada era de 1kHz, mientras que las simulaciones se realizaron a una frecuencia de 5kHz, considerado un valor óptimo. Por las razones expuestas se decidió programar en punto fijo y de forma matricial todas las operaciones. Para la programación en punto fijo se utilizó 32 bits de precisión con 14 de ellos destinados a la parte entera del número. Las operaciones matriciales se vieron reducidas a sumas y multiplicaciones. Si bien esto aumentó la complejidad de la programación gráfica (las sumas y mutiplicaciones se implementan mediante bloques y no código), al obviar las conversiones internas se simplificó enormente el algoritmo y se mejoró su desempeño. Con este cambio se logró una frecuencia máxima de muestreo de 8kHz al 95 % de la capacidad del DSP o 5kHz (frecuencia de simulación) a poco más del 65 % de la capacidad del dispositivo. Se optó por trabajar a 5kHz para así poder realizar comparaciones equitativas con el filtro H_∞ dado que este al tener una carga computacional más pesada, su frecuencia se trabajo se esperaba sea menor. Adicionalmente, para fijar las condiciones iniciales tanto de la matriz P como de los estados x se utilizaron bloques de retraso unitario, que en el entorno de VisSim permiten fijar condiciones iniciales para el algoritmo.

A continuación se muestran los bloques más significativos del algoritmo. Se aprovechó la capacidad que ofrece el software de comprimir varios bloques en un solo bloque para mantener el orden en el programa. Sin embargo, no se debe abusar de esta característica ya que disminuye el desempeño final del programa según lo menciona la guía de manual del usuario de VisSim [24].

En la Figura 4.10 se muestra el bloque compuesto del filtro Kalman extendido. Las trece salidas son las siguientes:

1. Frecuencia de voltaje de alimentación.
2. Amplitud (normalizado a 1) de voltaje de alimentación.
3. Voltaje del estator en el eje de cuadratura (v_{qs}).
4. Voltaje del estator en el eje directo (v_{ds}).
5. Corriente del estator en el eje de cuadratura (i_{qs}).
6. Corriente del estator en el eje directo (i_{ds}).
7. Flujo magnético del estator en el eje de cuadratura (ψ_{qs}).
8. Flujo magnético del estator en el eje directo (ψ_{ds}).

9. Flujo magnético del rotor en el eje de cuadratura (ψ_{qr}).
10. Flujo magnético del rotor en el eje directo (ψ_{dr}).
11. Velocidad del rotor (ω_r).
12. Torque de carga (T_l).
13. Porcentaje de uso del DSP.

Figura 4.10: Bloque final del filtro Kalman extendido

En la Figura 4.11 se muestran los bloques más importantes del algoritmo. Se observa primero la etapa de generación de la señal PWM (bloques open loop tri sinus generator, scaling y PWM) y de las salidas de frecuencia y amplitud del voltaje de alimentación. A continuación se aprecia la etapa de muestreo, cambio de ejes de referencia a $qd0$, acondicionamiento y escalamiento tanto de las señales de voltaje como corriente (bloques voltaje, corriente y Clarke Transform32).

Los siguientes bloques que se aprecian son los requeridos en el algoritmo descrito por las ecuaciones (2.31), (2.32), (2.33), (2.34), (2.1.3), (2.35).

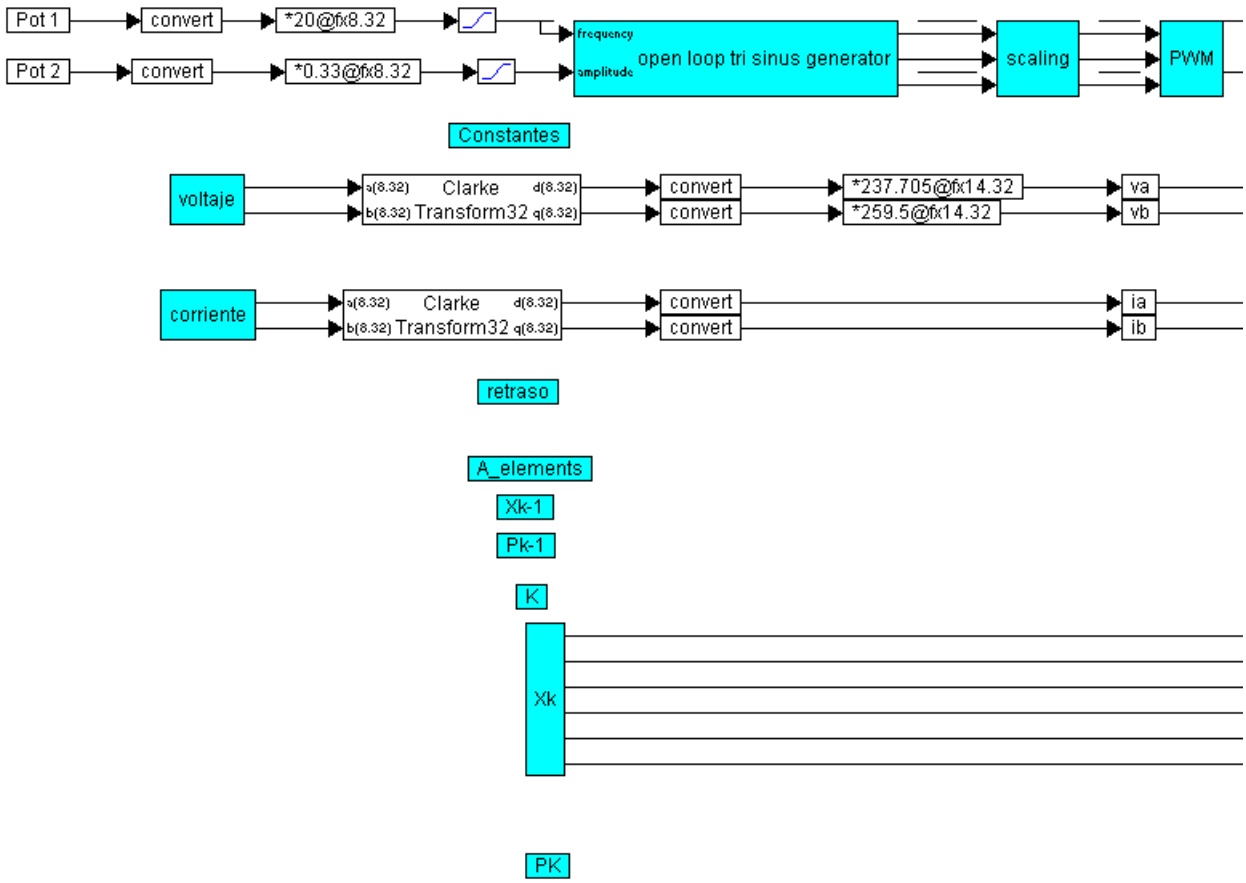


Figura 4.11: Detalle de los bloques más importantes en el Kalman extendido

En la Figura 4.15 se aprecia la interfaz de VisSim utilizada para monitorear el desempeño del filtro en tiempo real. Se aprecian cada una de las variables de interés ya sea mediante paneles numéricos o ejes cartesianos para mostrar evolución temporal.

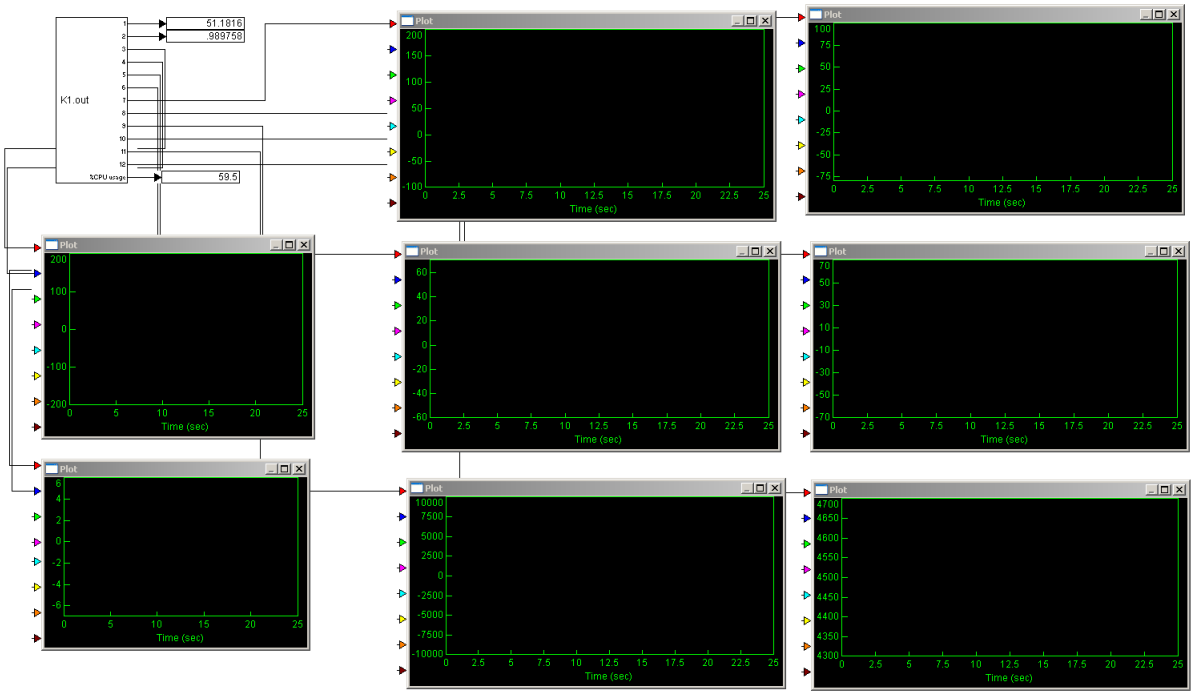


Figura 4.12: Interfaz de monitoreo del filtro Kalman extendido

4.2.2. Filtro H_∞ extendido

Para la implementación del filtro H_∞ extendido se utilizó un DSP de punto fijo F2808 de Texas Instruments. Al igual que en el filtro Kalman, el DSP era el encargado del control de alimentación del motor mediante la generación de señales PWM a una frecuencia y amplitud definida por el usuario mediante dos potenciometros destinados para el efecto (accesibles en la tarjeta de control de Francecol). Así también, el DSP realiza el muestreo de las señales tanto de corriente como de voltaje. Estas señales son escaladas digitalmente a los valores reales en el motor dentro del mismo DSP.

El algoritmo descrito en las ecuaciones (2.80), (2.81), (2.82), (2.83), (2.84) y (2.85) se implementó en VisSim. Sin embargo, debido a la caracteristica de punto fijo del DSP todas las operaciones matriciales debieron ser implementadas como operaciones escalares independientes. Esto porque VisSim no tiene una librería de operaciones matriciales en punto fijo, solo en punto flotante. Si se programaba el filtro H_∞ con la librería matricial el número de conversiones internas para pasar de punto fijo a punto flotante habría hecho que el filtro se ejecute a una velocidad extremadamente baja. Debido a la experiencia del filtro Kalman no se intentó siquiera programar el algoritmo en operaciones matriciales.

No obstante, se aprovechó el hecho que las matrices definidas en (2.81), (2.84) y (2.85) son simétricas para así calcular solo una parte de todas las entradas. Por ejemplo, en el caso de P_k (2.84) se calcularon 21 de las 36 entradas totales lo que significa más de 40 % menos cálculos. De esta forma se optimizó el código y se redujo la carga computacional sobretodo en la sección en la que se calcula la inversa de R_w. Dado que el algoritmo de inversión de matrices utilizado es mediante cálculo de adjuntos; para poder invertir una matriz R_w de dimensiones

4×4 se requieren calcular 16 determinantes 3×3. Sin embargo, al aprovechar la simetría de R_w tan sólo se calculan 10 determinates. Mediante esta técnica fue posible utilizar el 60 % de la capacidad del dispositivo a 5kHz, es decir prácticamente las mismas condiciones de trabajo que el filtro Kalman pese a ser un algoritmo más complejo y extenso computacionalmente.

A continuación se muestran los bloques más significativos del algoritmo. Se aprovechó la capacidad que ofrece el software de comprimir varios bloques en un solo bloque para mantener el orden en el programa. Sin embargo, no se debe abusar de esta característica ya que disminuye el desempeño final del programa según lo menciona la guía de manual del usuario de VisSim [24].

En la Figura 4.13 se muestra el bloque compuesto del filtro H_∞ extendido. Las trece salidas son las mismas que las descritas para el filtro Kalman extendido.

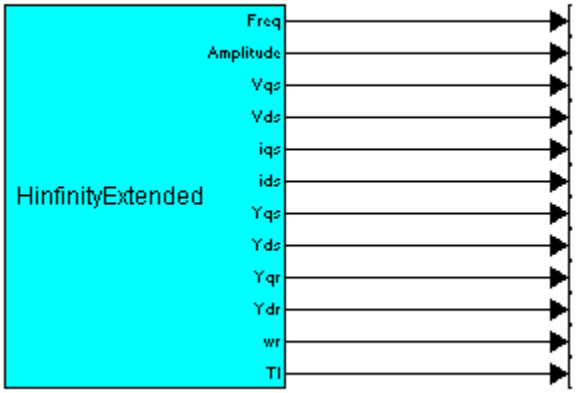


Figura 4.13: Bloque final del filtro H_∞ extendido

De manera análoga en la Figura 4.14 se muestra el detalle del filtro H_∞. Las primeras etapas son la mismas que las del filtro Kalman, es decir generación de señales PWM y muestreo y acondicionamiento de señales de voltaje y corriente. Los siguientes bloques son los necesarios para implementar el algoritmo dado por las ecuaciones (2.80), (2.81), (2.82), (2.83), (2.84) y (2.85). En este grupo resalta el cálculo de la matriz R_w y su respectiva inversa.

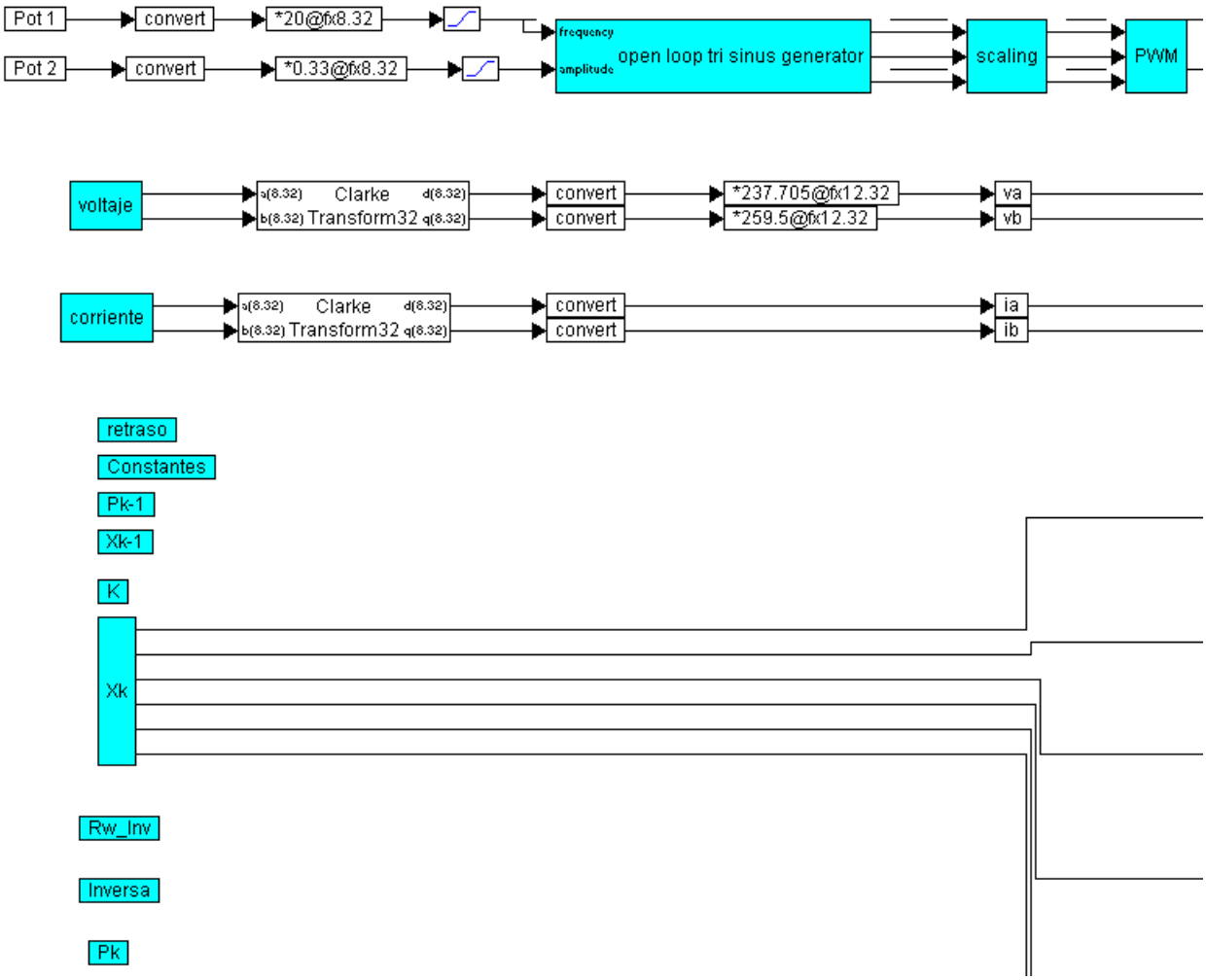


Figura 4.14: Detalle de los bloques más importantes en el filtro H_∞ extendido

En la Figura 4.15 se aprecia la interfaz de VisSim utilizada para monitorear el desempeño del filtro en tiempo real. Se aprecian cada una de las variables de interés ya sea mediante paneles numéricos o ejes cartesianos para mostrar evolución temporal.

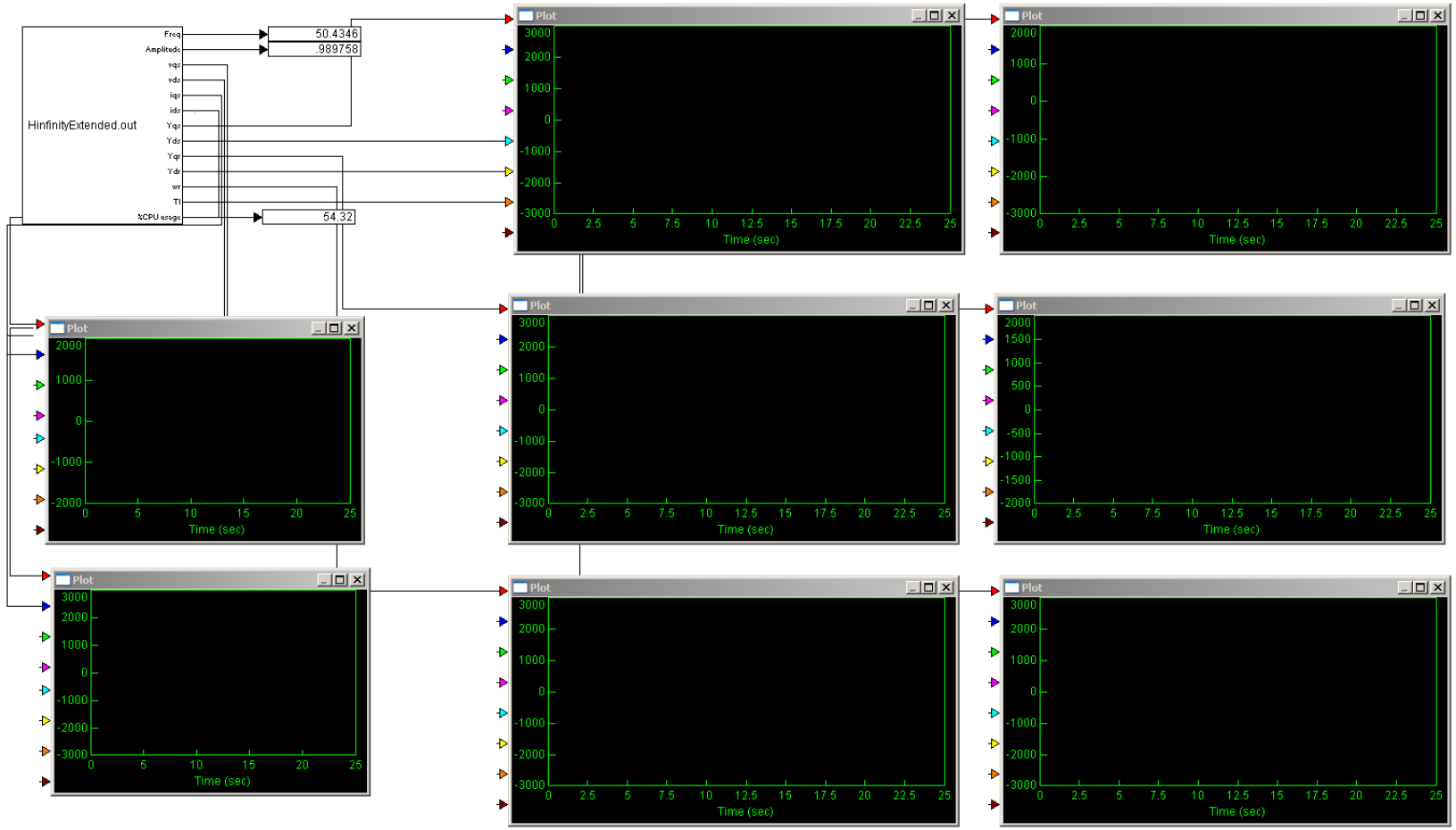


Figura 4.15: Interfaz de monitoreo del filtro H_∞ extendido

En ninguno de los dos casos, filtro Kalman extendido o filtro H_∞ extendido se muestra el detalle propiamente dicho de la programación en bloques debido al gran tamaño que ocupa. En su lugar, se incluye como anexo el código C que genera VisSim para ser compilado en Code Composer Studio.

4.3. Resultados de pruebas en tiempo real de filtros Kalman extendido y H_∞ extendido

A continuación se muestran diversas pruebas realizadas para los filtros Kalman Extendido y H_∞ extendido. En los dos casos se hicieron las mismas pruebas para tener una base sobre la cual comparar. Se dividió el rango de frecuencia del voltaje de alimentación de la máquina de inducción de la siguiente manera:

- Alta Frecuencia: alimentación entre 40 y 60Hz.
- Frecuencia intermedia: alimentación entre 25 y 40Hz.
- Baja Frecuencia: alimentación entre 0 y 25Hz.

Esto con el afán de observar el comportamiento del filtro en diversas condiciones de trabajo, dentro de cada segmento de frecuencia se realizaron las 8 pruebas que se detallan:

1. Estimación al arranque del motor sin carga.
2. Estimación en estado estable sin carga.
3. Estimación cuando existe cambio en la frecuencia de voltaje de alimentación de un valor inicial hacia uno mayor.
4. Estimación cuando existe cambio en la frecuencia de voltaje de alimentación de un valor inicial hacia uno menor.
5. Estimación cuando la carga frena completamente al motor.
6. Estimación cuando se frena y acelera el motor por aumento y disminución de carga.
7. Estimación cuando se realizan aceleraciones y frenados múltiples.
8. Estimación cuando el motor arranca con carga.
9. Estimación cuando el motor tiene una carga inicial alta y carga final baja.

Las pruebas se realizaron a frecuencias de operación de 50Hz, 30Hz y 20Hz para los rangos de alta, media y baja frecuencia respectivamente.

4.3.1. Filtro Kalman extendido-Alta frecuencia (40-60 Hz)

Las matrices Q_k y R_k utilizadas en la implementación del filtro Kalman extendido fueron:

$$Q_k = \begin{bmatrix} 1{,}5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1{,}5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1{,}5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1{,}5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0{,}01 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0{,}0001 \end{bmatrix}$$

$$R_k = \begin{bmatrix} 0{,}1 & 0 \\ 0 & 0{,}1 \end{bmatrix}$$

La matriz G_k y el parámetro γ

Se realizaron las pruebas detalladas en la introducción de esta sección con una frecuencia de alimentación de 50Hz. A continuación se muestran los resultados. Como ya se mencionó se muestran los voltajes y corrientes $dq0$ del estator (medidos vía muestreo en el DSP) y las variables estimadas: flujos $dq0$ de estator y rotor, velocidad del rotor y torque de carga.

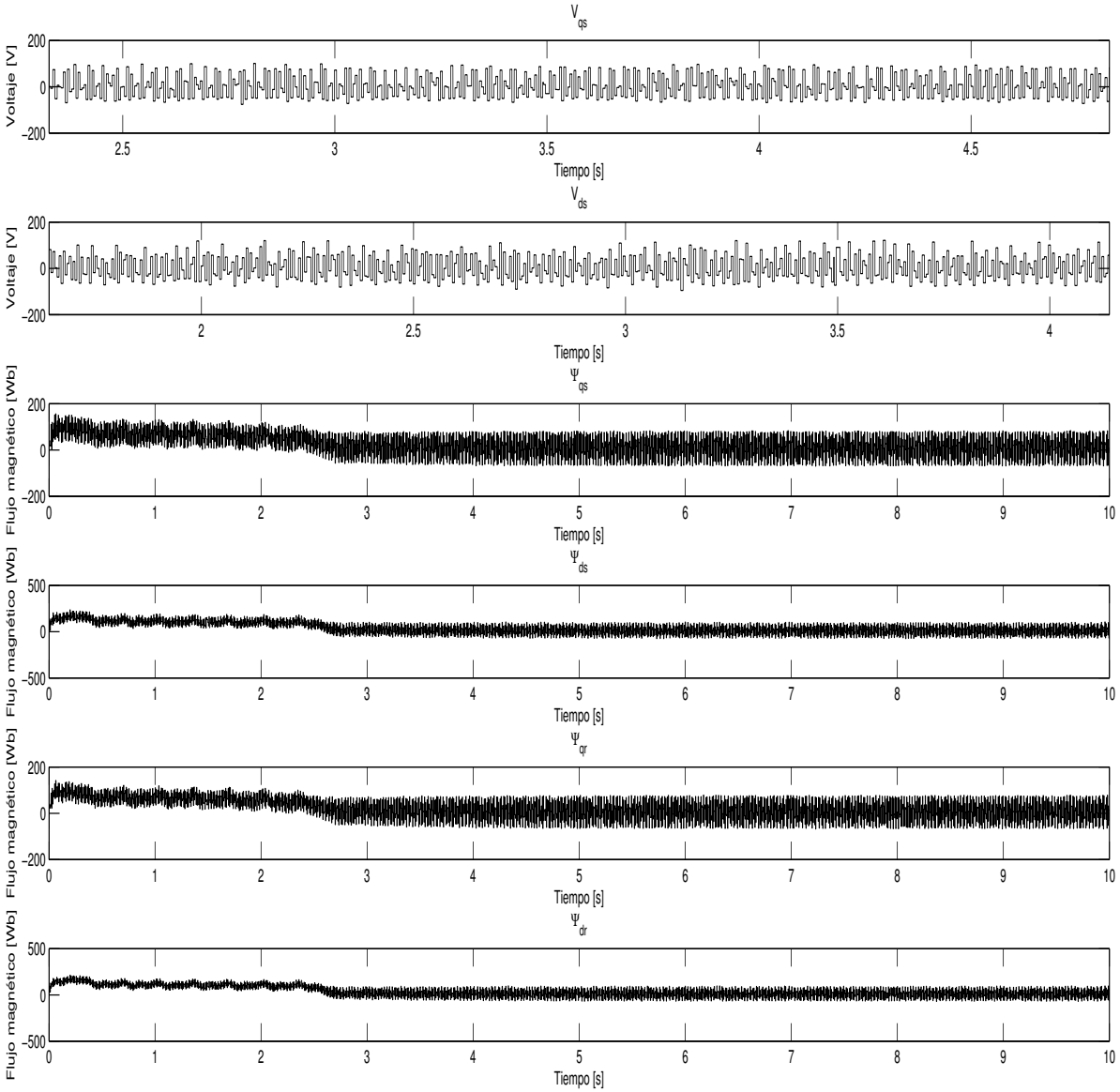


Figura 4.16: Estimación Kalman a 50Hz en el arranque sin carga (Voltajes y Flujos)

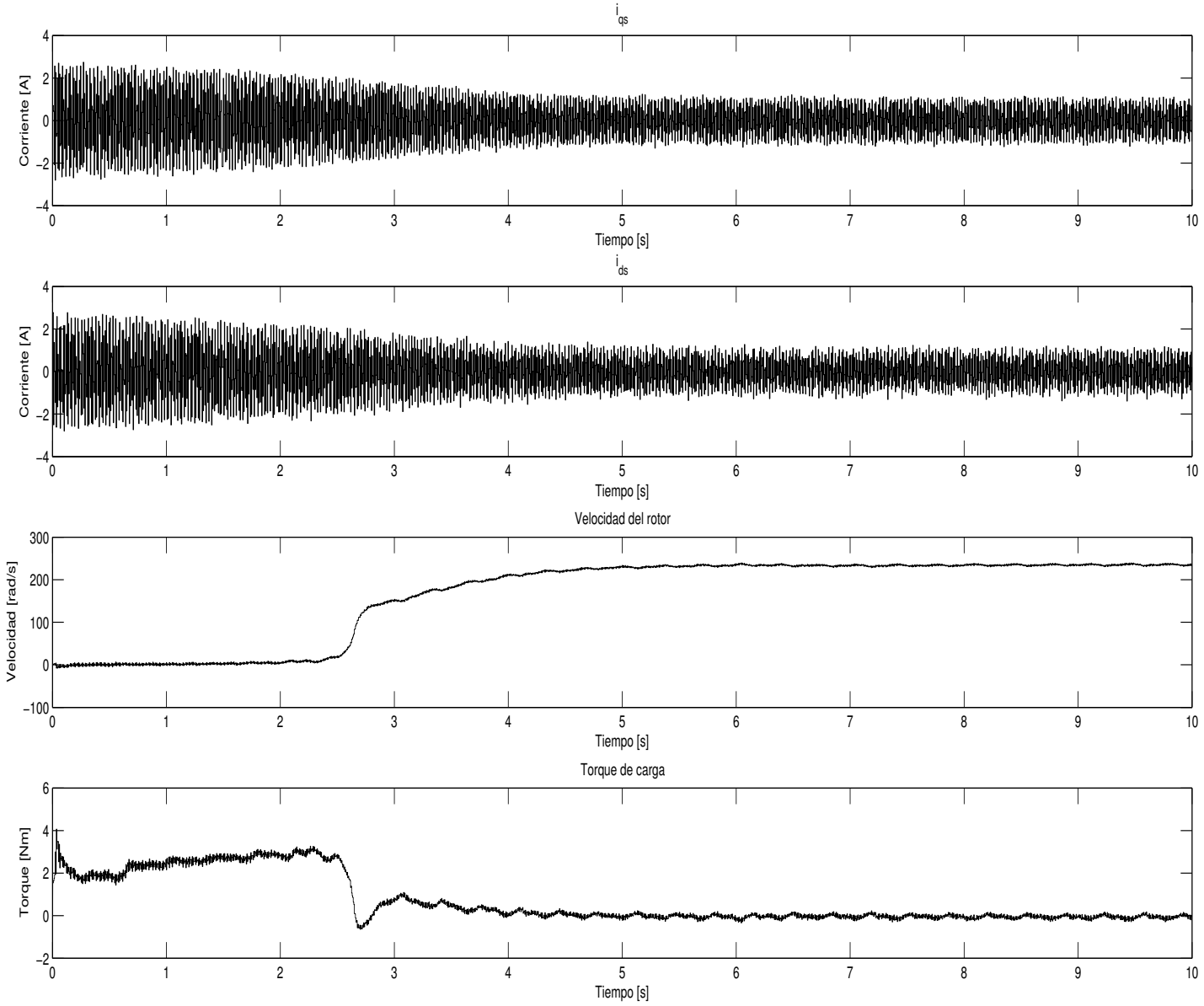


Figura 4.17: Estimación Kalman a 50Hz en el arranque sin carga (Corriente, velocidad, torque)

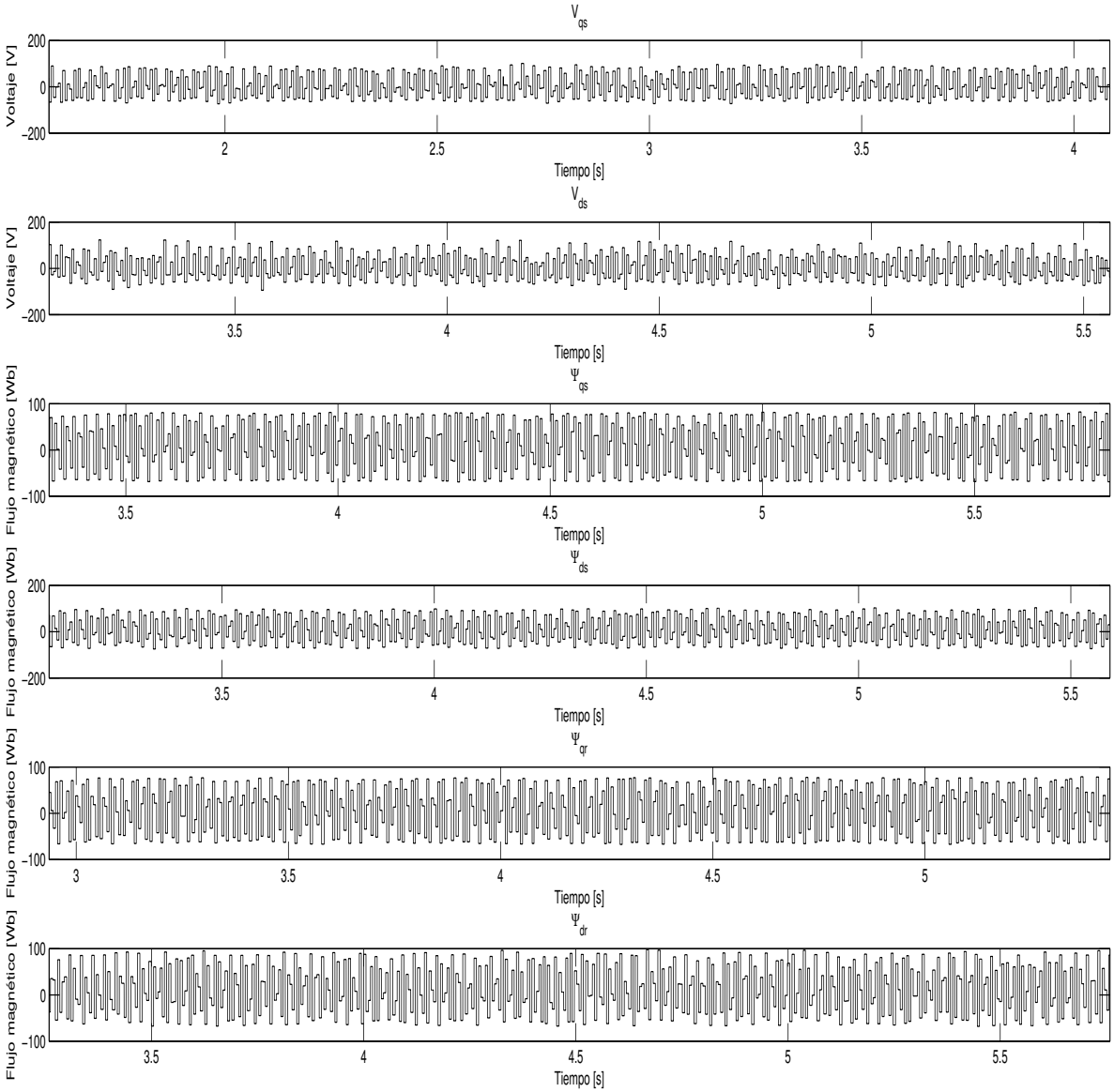


Figura 4.18: Estimación Kalman a 50Hz en estado estable (Voltajes y Flujos)

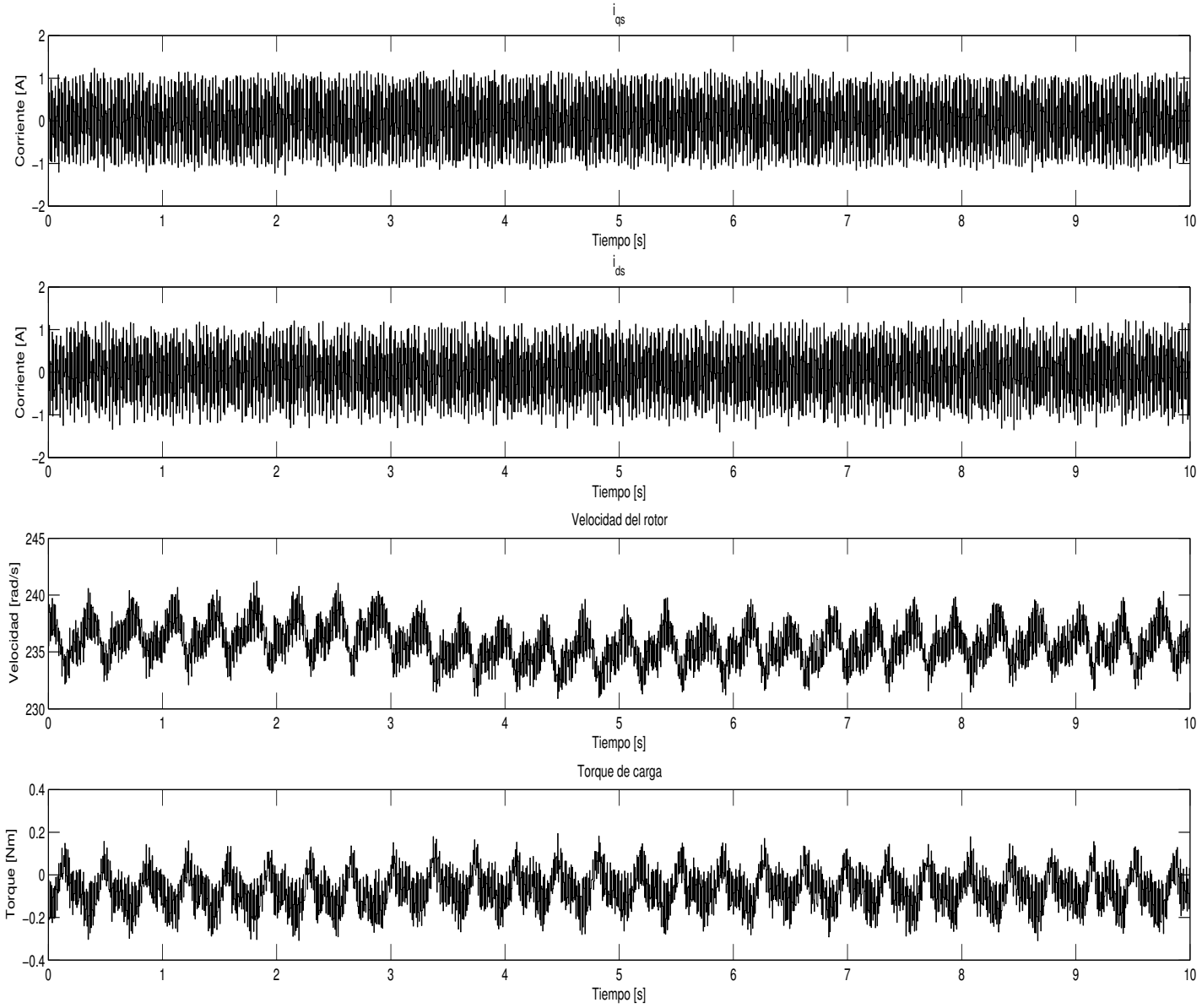


Figura 4.19: Estimación Kalman a 50Hz en estado estable (Corriente, velocidad, torque)

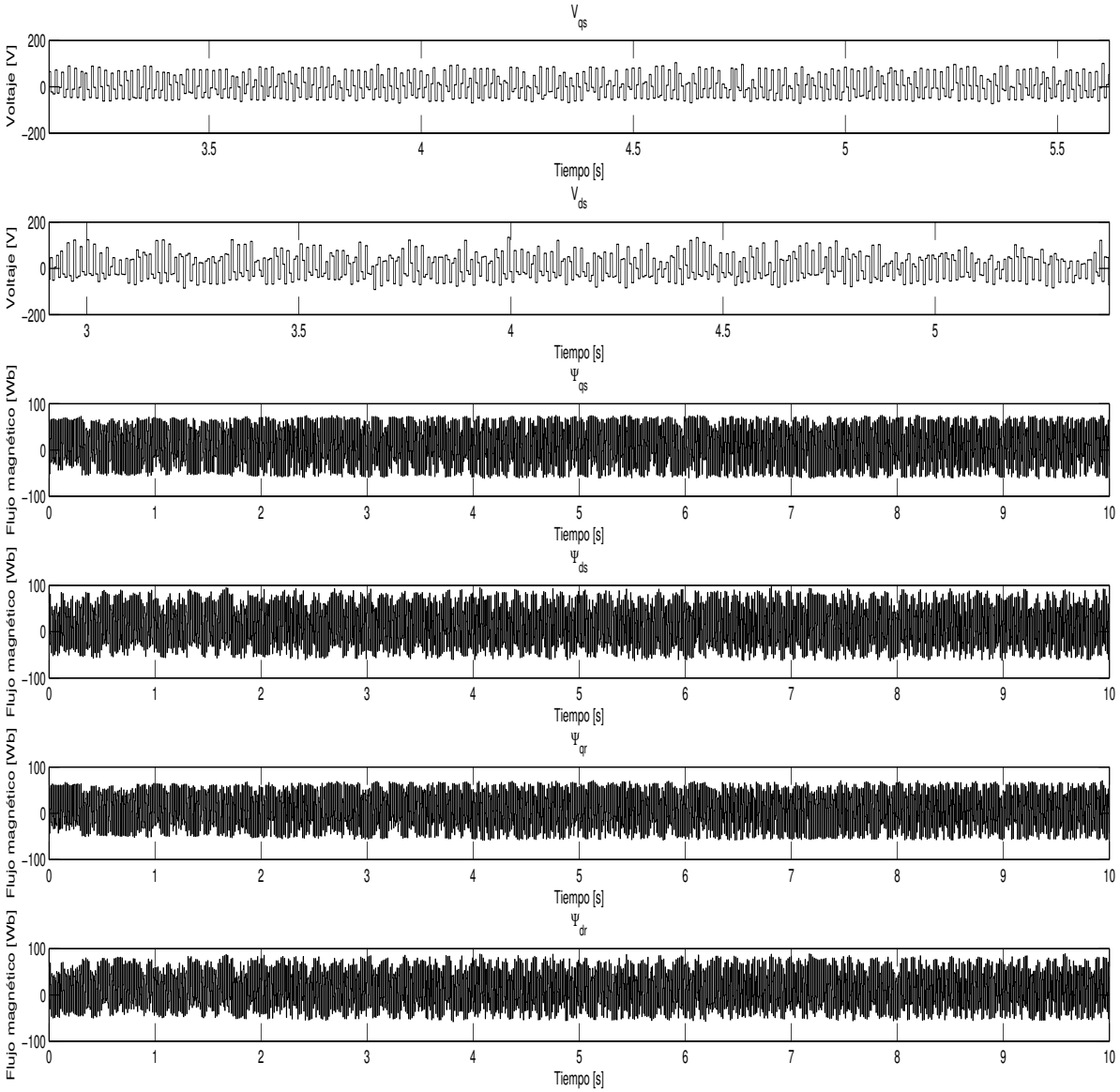


Figura 4.20: Estimación Kalman con cambio de 50Hz a 56Hz (Voltajes y Flujos)

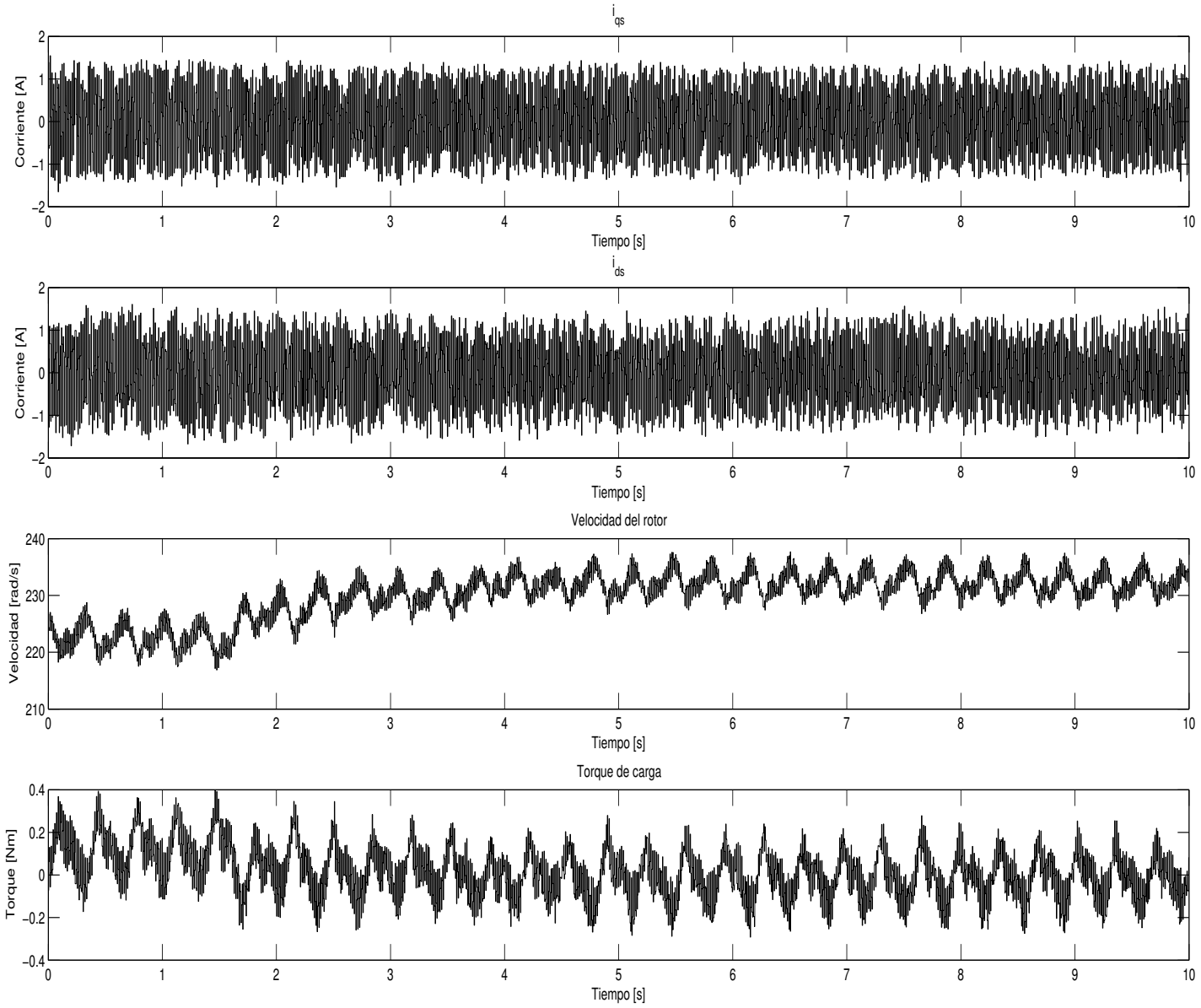


Figura 4.21: Estimación Kalman con cambio de 50Hz a 56Hz (Corrientes, Velocidad y Torque)

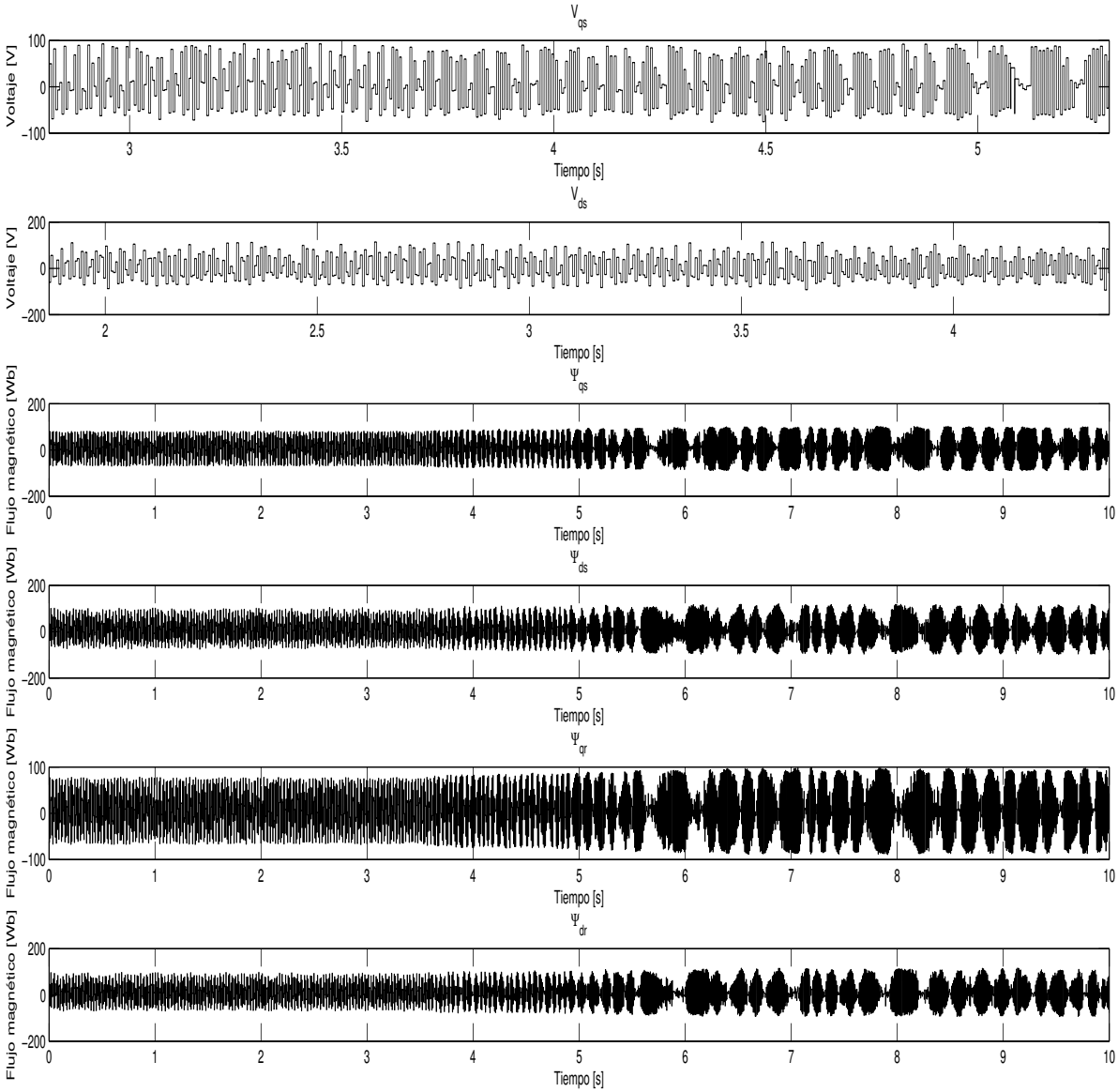


Figura 4.22: Estimación Kalman con cambio de 50Hz a 41Hz (Voltajes y Flujos)

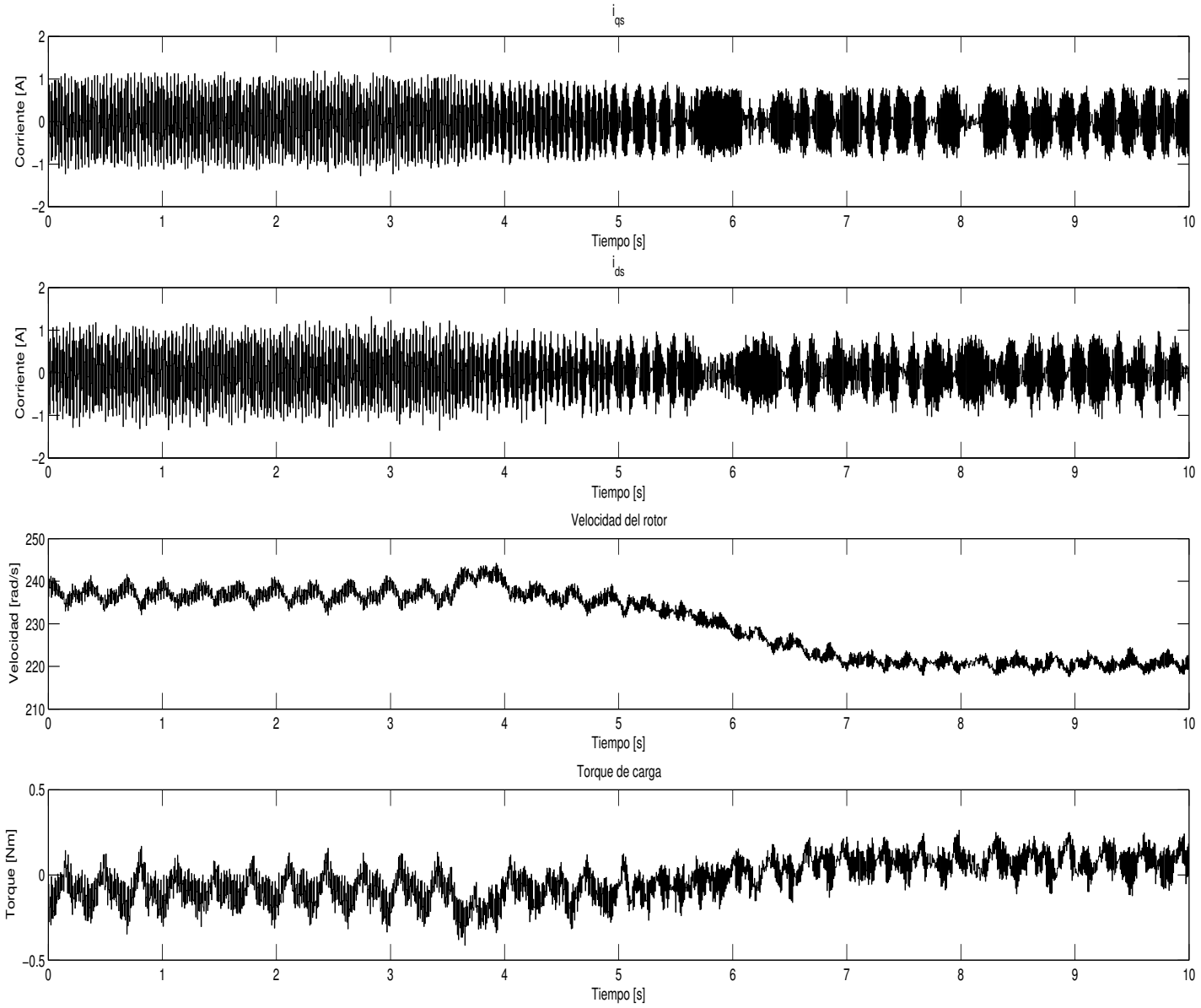


Figura 4.23: Estimación Kalman con cambio de 50Hz a 41Hz (Corrientes, velocidad y torque)

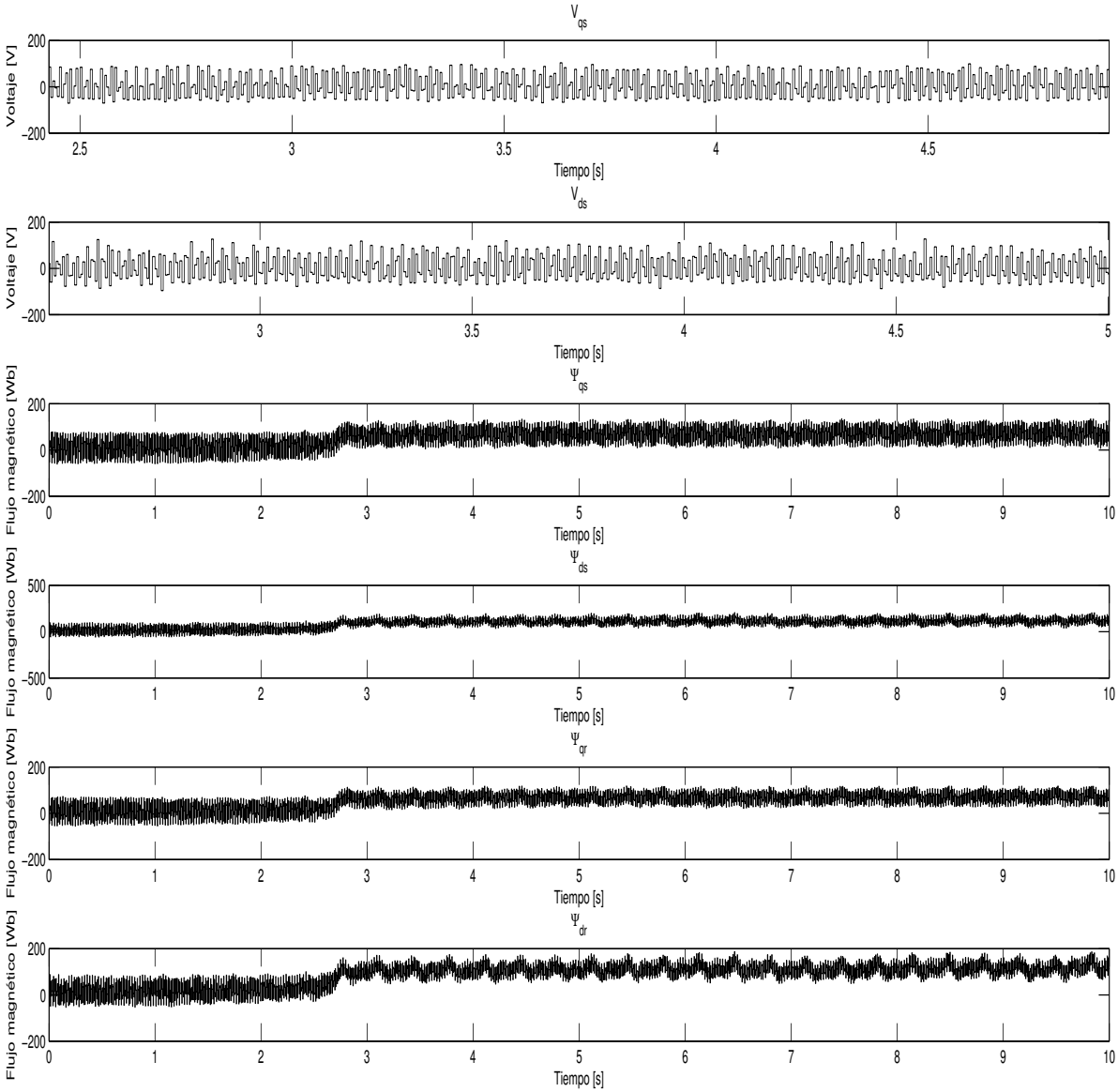


Figura 4.24: Estimación Kalman con frenado total (Voltajes y Flujos)

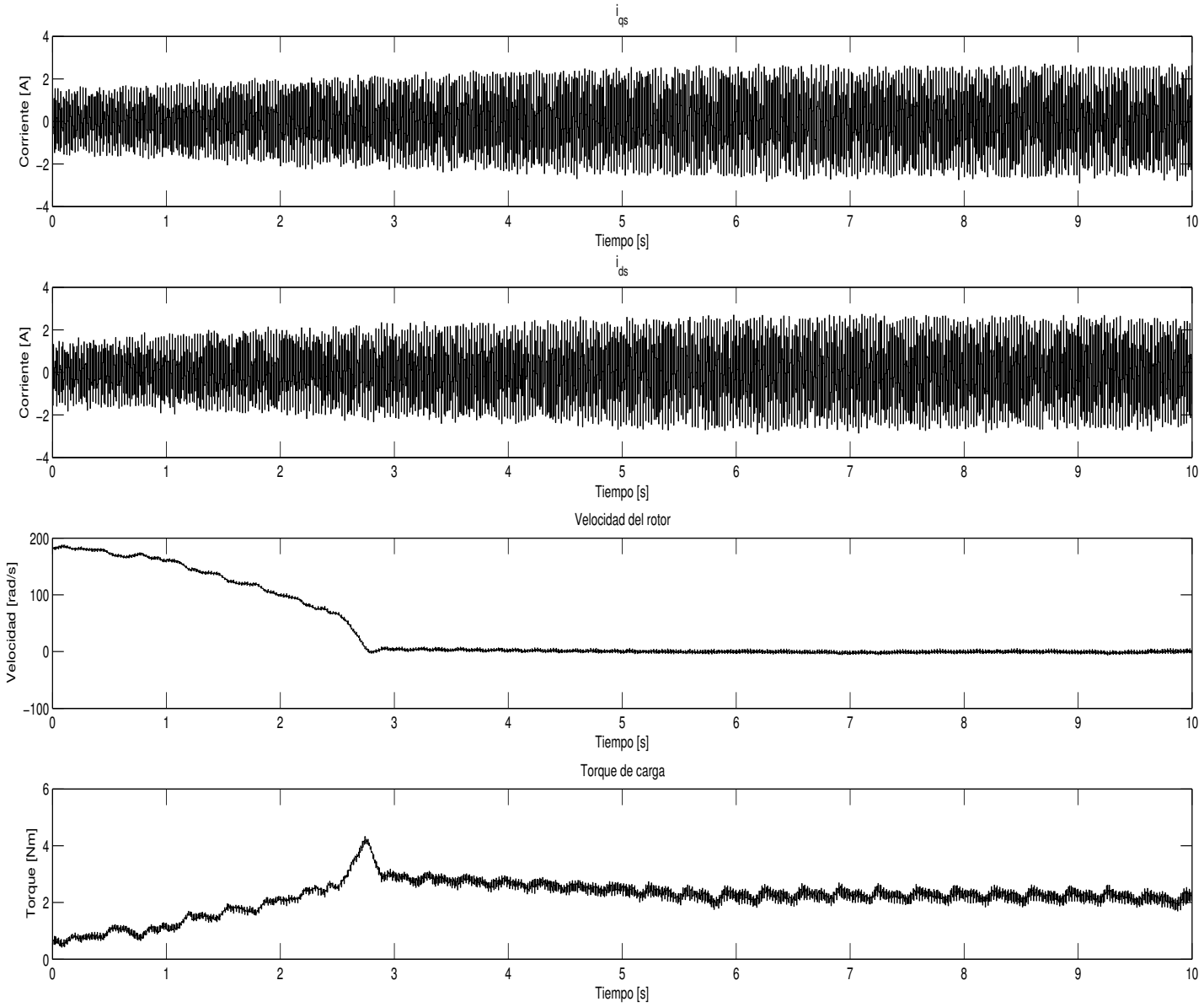


Figura 4.25: Estimación Kalman con frenado total (Corrientes, velocidad y torque)

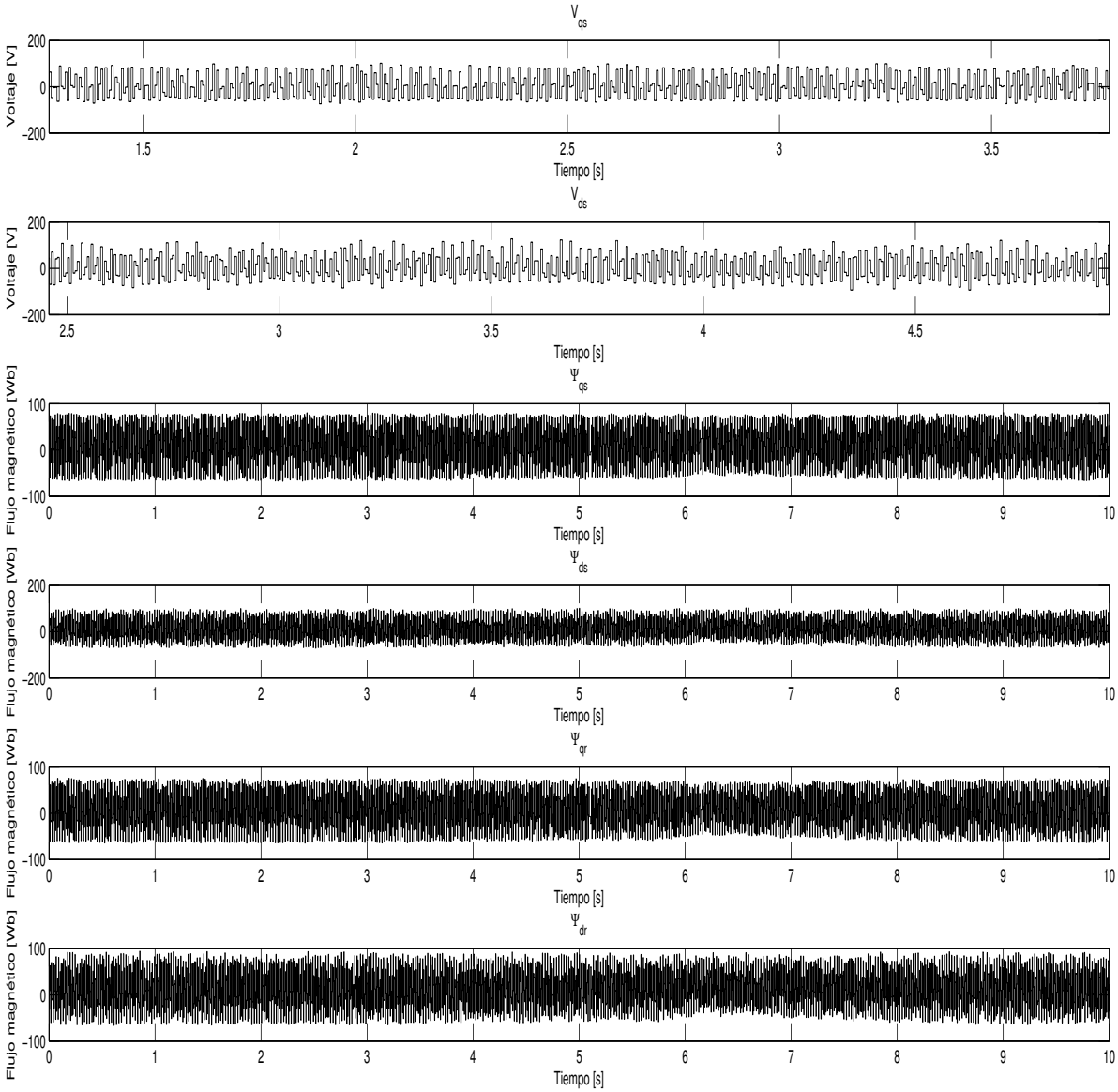


Figura 4.26: Estimación Kalman con cambios de carga (Voltajes y Flujos)

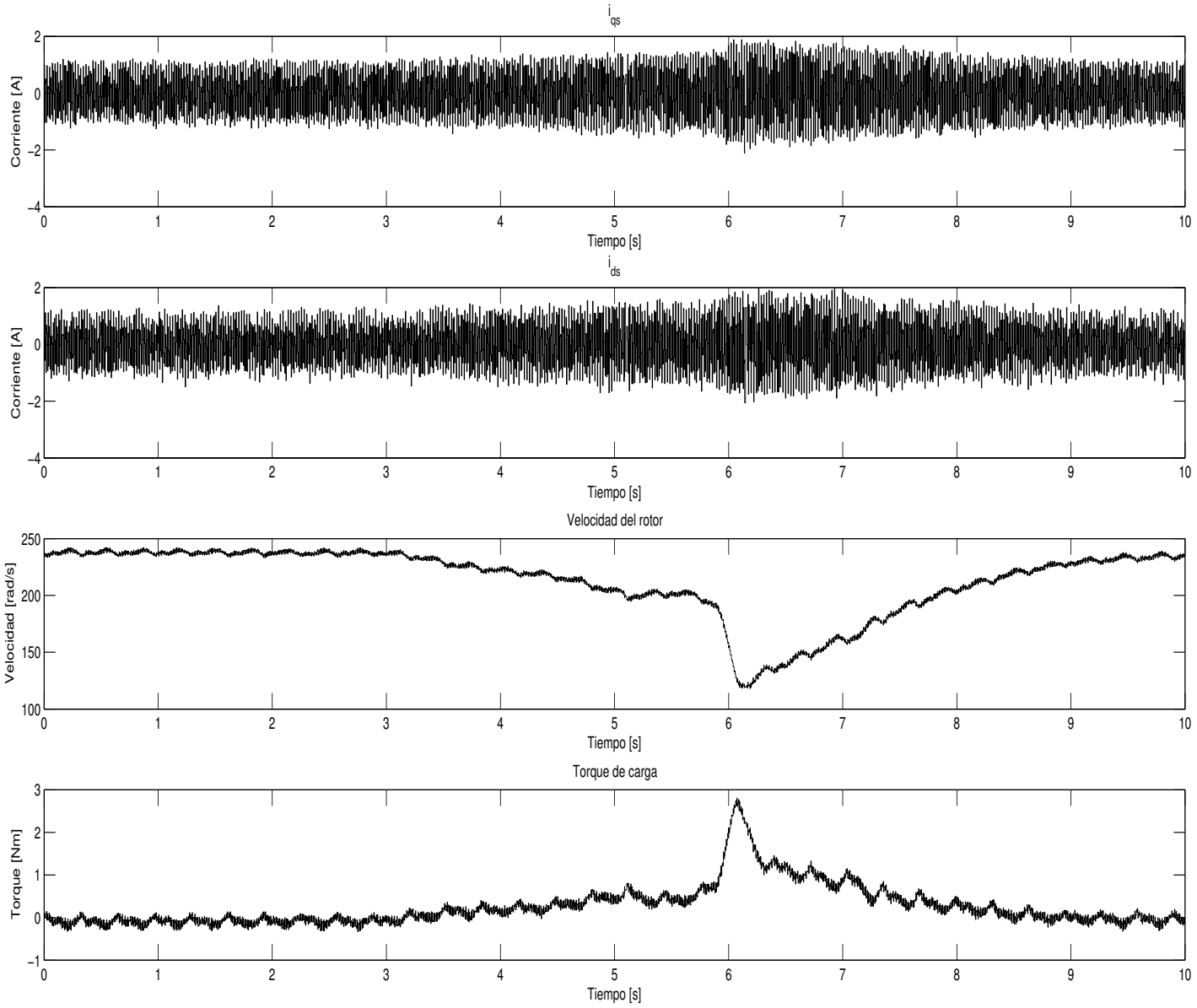


Figura 4.27: Estimación Kalman con cambios de carga (Corrientes, velocidad y torque)

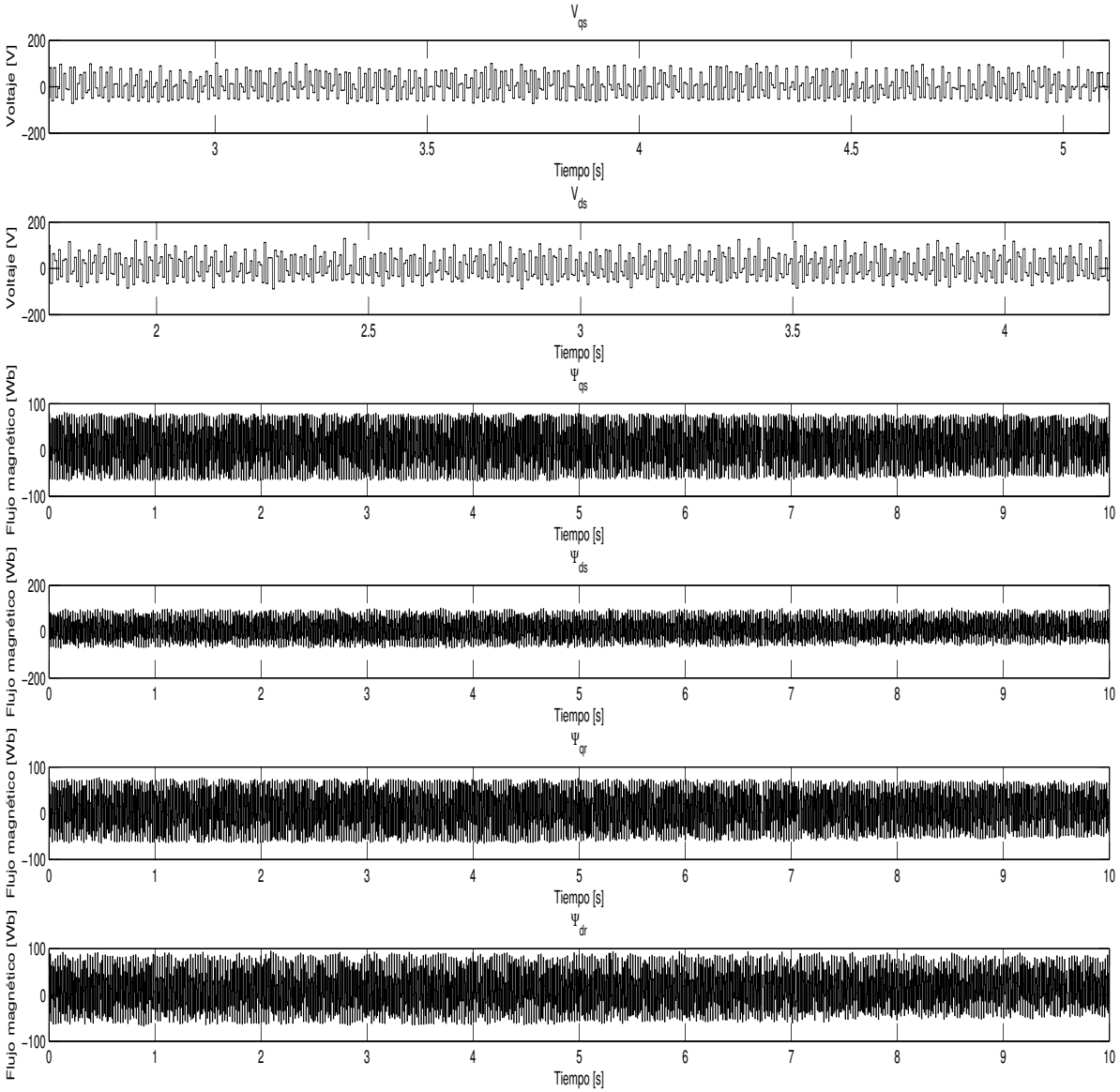


Figura 4.28: Estimación Kalman con cambios de carga múltiples (Voltajes y Flujos)

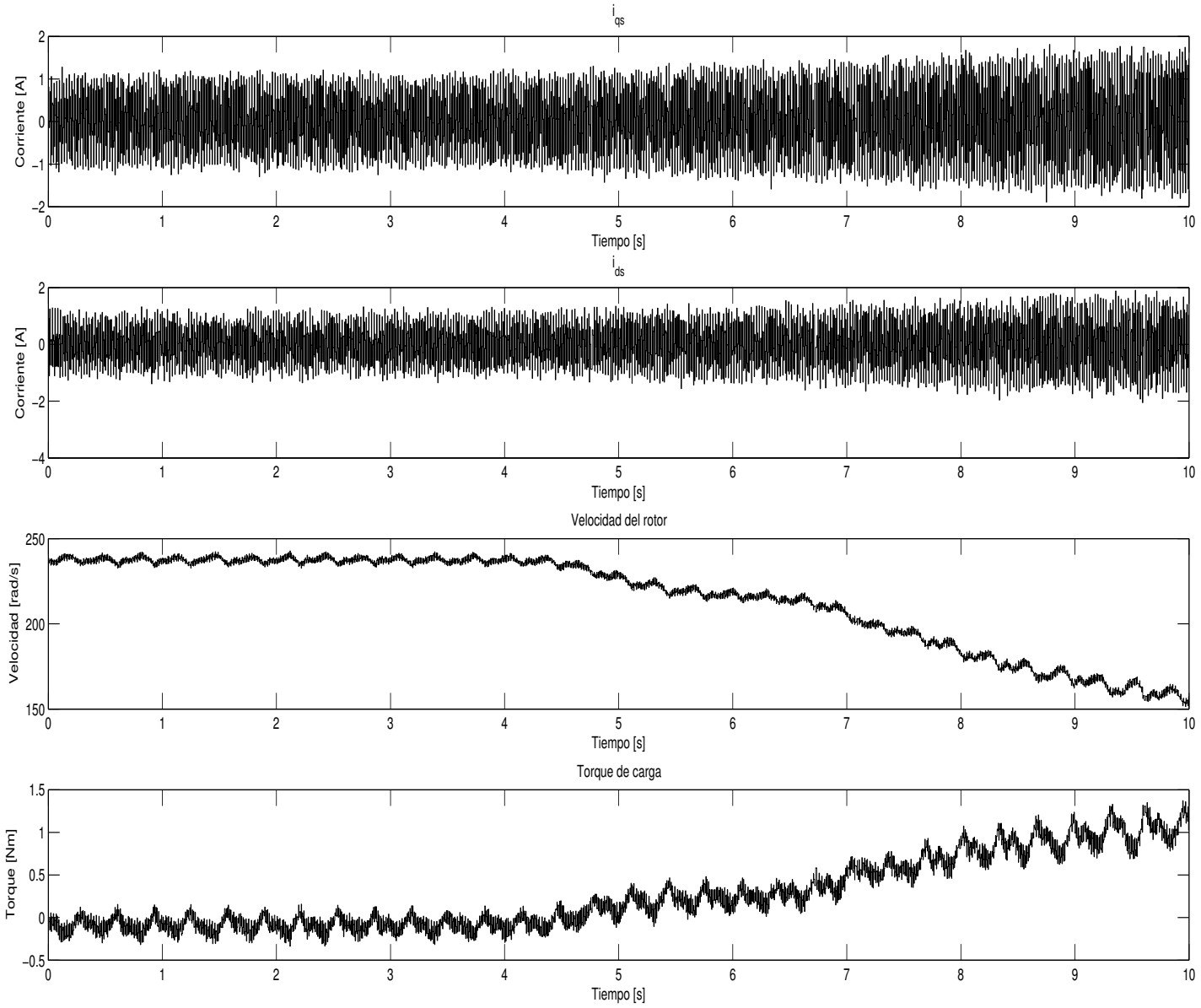


Figura 4.29: Estimación Kalman con cambios de carga múltiples. (Corrientes, velocidad y torque)

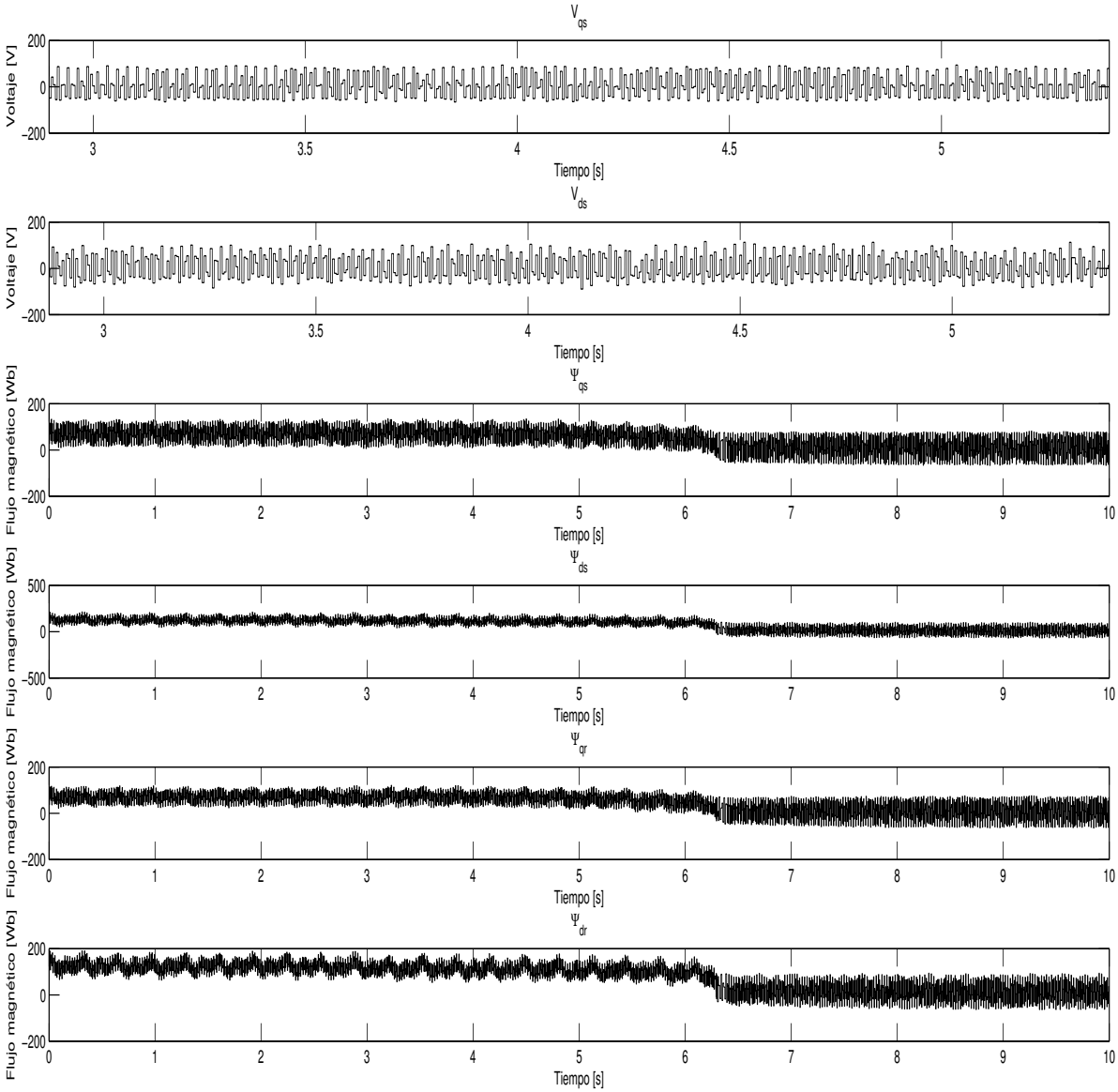


Figura 4.30: Estimación Kalman con carga total al arranque (Voltajes y Flujos)

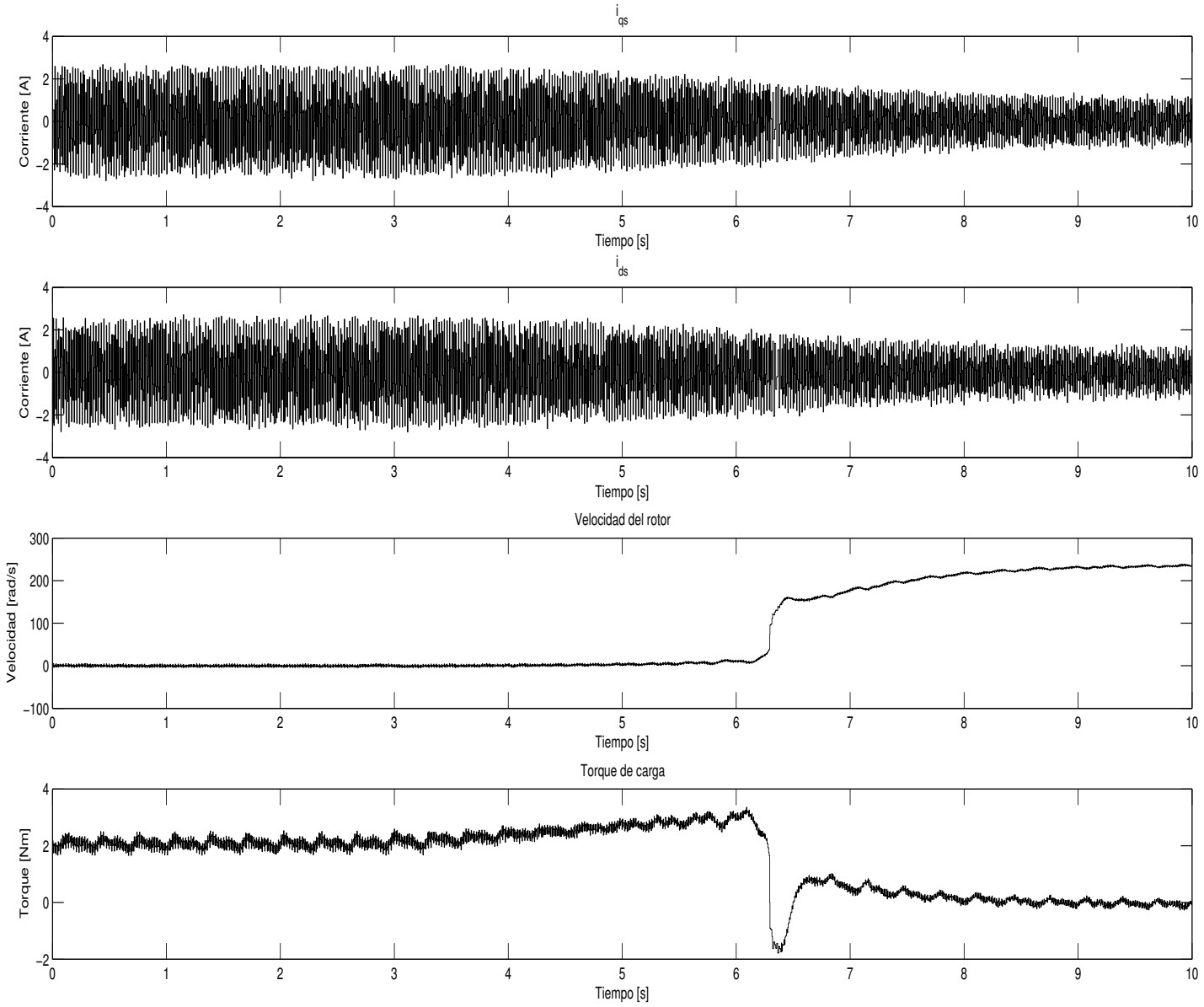


Figura 4.31: Estimación Kalman con carga total al arranque (Corrientes, velocidad y torque)

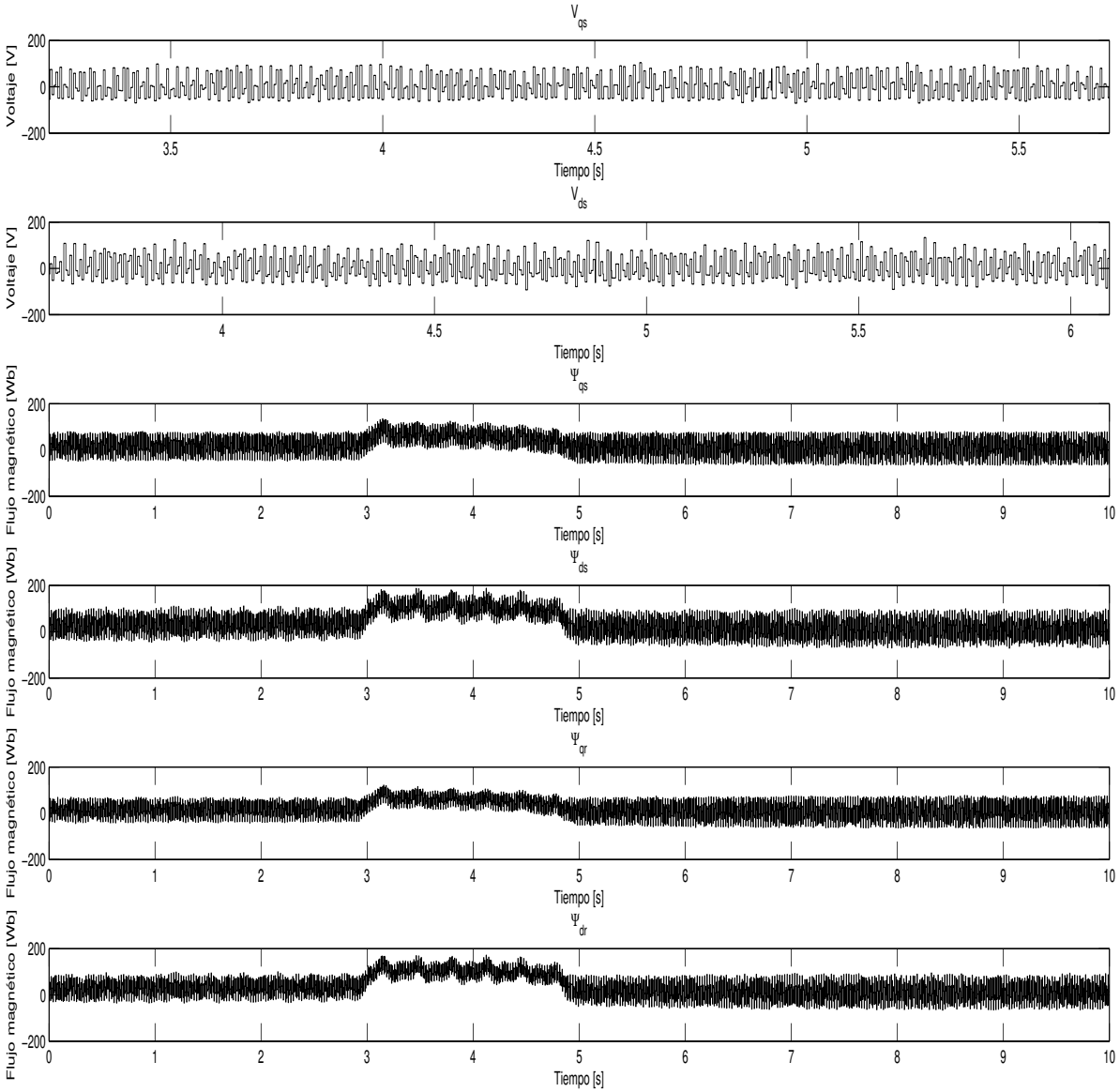


Figura 4.32: Estimación Kalman con carga y posterior liberación de carga (Voltajes y Flujos)

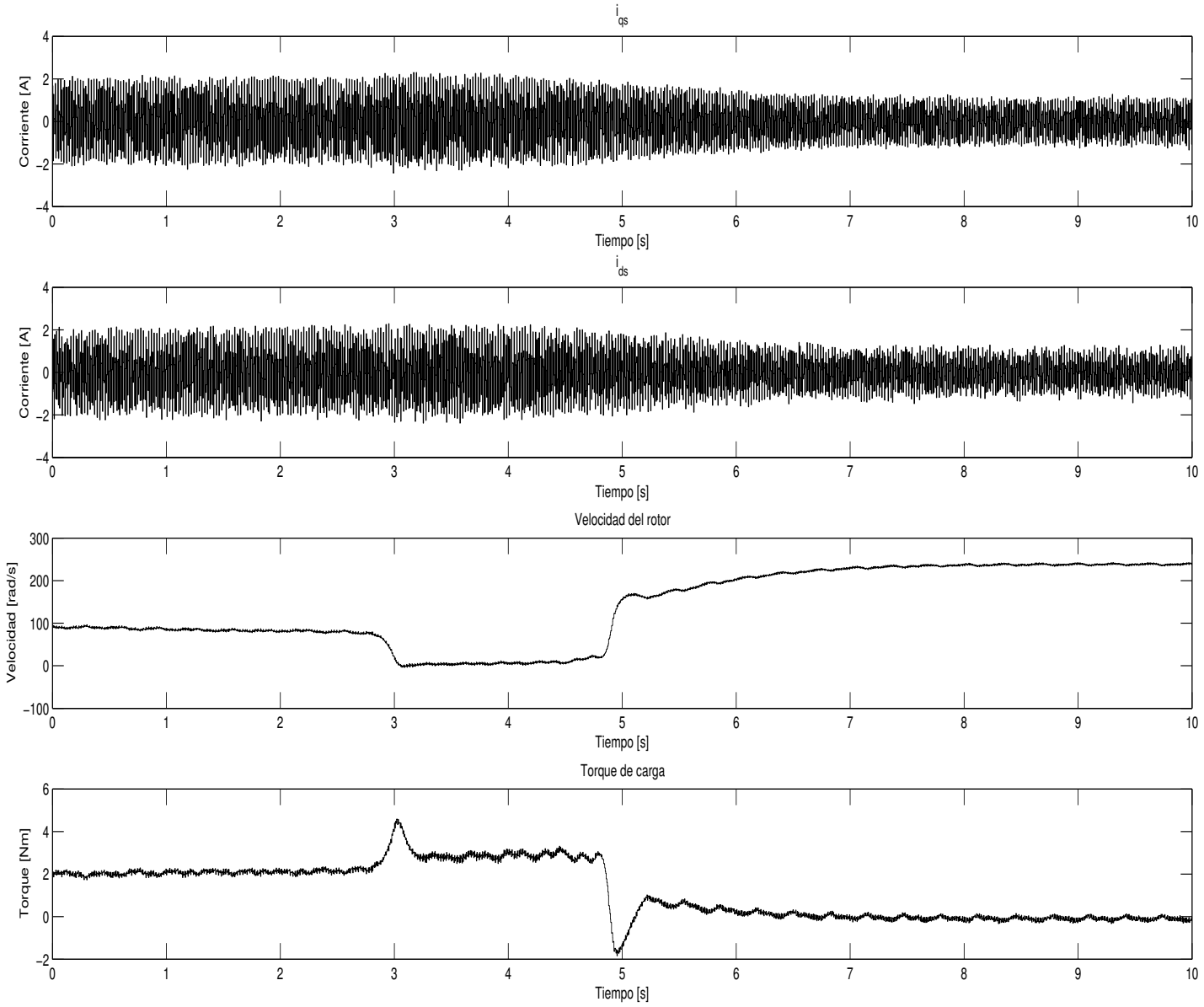


Figura 4.33: Estimación Kalman con carga y posterior liberación de carga (Corrientes, velocidad y torque)

En las Figuras 4.18 y 4.17, las curvas observadas son muy similares a las que se obtuvieron en simulación. Se observan estimaciones bastante limpias, sin oscilaciones grandes. Sim embargo, se observa que la velocidad final alcanzada no corresponde al valor ideal (314 rad/s). La velocidad es mucho menor, aproximadamente 250rad/s. Esto se debe principalmente a los

parámetros eléctricos medidos de la máquina y a la medición simplificada de los parámetros mecánicos. Se comprobó que el motor no era totalmente simétrico en sus fases por lo que el modelo de Krause es una buena aproximación pero se deberían incluir más elementos para mejorar el modelamiento, aunque se complique más el modelo. No obstante, la dinámica del filtro es consistente con lo que se espera para este sistema.

En las Figuras 4.18 y 4.19 correspondiente al estado estable se observan oscilaciones de aproximadamente 5rad/s en el caso de la velocidad y 0.2Nm en el caso del torque. En ambas estimaciones las oscilaciones son mínimas y dan cuenta de la calidad de las variables estimadas.

En las Figuras 4.22 y 4.21 se observa como aumenta la velocidad del rotor al aumentar la frecuencia del voltaje de alimentación del motor. Sin embargo, el estimador de torque de carga responde a la perturbación de frecuencia como si se tratase de un cambio carga. Al acelerarse el motor y aumentar su velocidad disminuye el torque de carga estimado.

Al disminuirse la frecuencia del voltaje de alimentación de motor se observa en las Figuras 4.22 y 4.23 que la velocidad del rotor baja pero también se puede apreciar que el torque de velocidad aumenta. Es decir el estimador asume como perturbación de carga a la perturbación en frecuencia. Esto tiene sentido ya que la reducción de frecuencia ocasiona una desaceleración del motor, el mismo efecto que una carga aplicada.

En las Figuras 4.24 y 4.25 se aplica una carga de forma tal de detener totalmente al motor. En este caso se observa como los flujos reaccionan de forma significativa, la corriente aumenta, la velocidad decae hasta 0 rad/s rápidamente y sin ruido y la señal de torque de carga se estabiliza en aproximadamente 2Nm. Esta viene a ser una primera estimación y medición del torque real de la máquina.

Cuando se efectuan cambios de carga en las Figuras 4.26 y 4.27 se observa que el filtro responde correctamente a estas perturbaciones, mostrando picos de torque de carga en los mínimos de velocidad del rotor. Posteriormente al liberar la carga se observa como el motor se acelera.

En las Figuras 4.28 y 4.29 se muestra el caso de perturbaciones de carga múltiples se ve que el estimador realiza una buena estimación de las variables de interés. No obstante, también se observa que existen oscilaciones durante el proceso de convergencia aunque éstas no sean muy grandes.

Las Figuras 4.30 y 4.31 muestran el comportamiento de un motor con 100 % de carga. Inicialmente el motor tiene una carga tal que no puede arrancar (2 Nm aproximadamente) como ya se observó en la prueba de frenado total expuesta anteriormente. Luego, al liberar la carga se observa un aumento rápido en la velocidad del rotor y en simultáneo un descenso del torque de carga estimado. El filtro responde como se espera para este sistema.

En las Figuras 4.32 y 4.33 se observa la estimación de velocidad de rotor y torque de carga en el caso en que el motor tiene carga y posteriormente se la libera. Se deja una carga baja, luego se aumenta la carga, para finalmente liberar al sistema totalmente. Se observa que el filtro responde exactamente según la dinámica escrita. Los picos que se observan en los momentos de la pertubación son muy similares a los que se observan el sistemas de segundo orden subamortiguados.

4.3.2. Filtro Kalman extendido- Frecuencia media (25-40 Hz)

Se realizaron las pruebas detalladas en la introducción de esta sección con una frecuencia de alimentación de 30Hz. A continuación se muestran los resultados. Como ya se mencionó se muestran los voltajes y corrientes $dq0$ del estator (medidos vía muestreo en el DSP) y las variables estimadas: flujos $dq0$ de estator y rotor, velocidad del rotor y torque de carga.

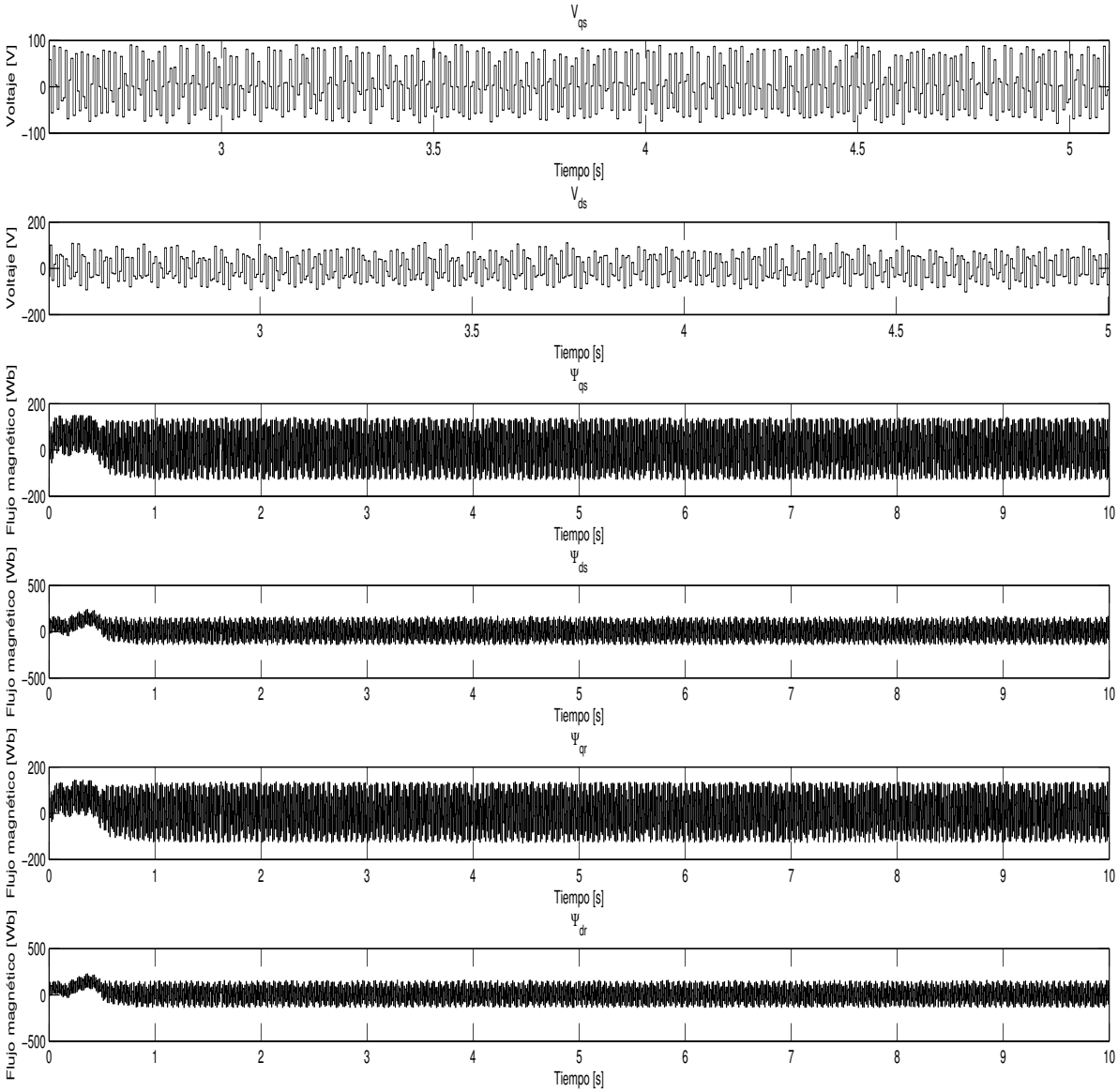


Figura 4.34: Estimación Kalman a 30Hz en el arranque sin carga (Voltajes y Flujos)

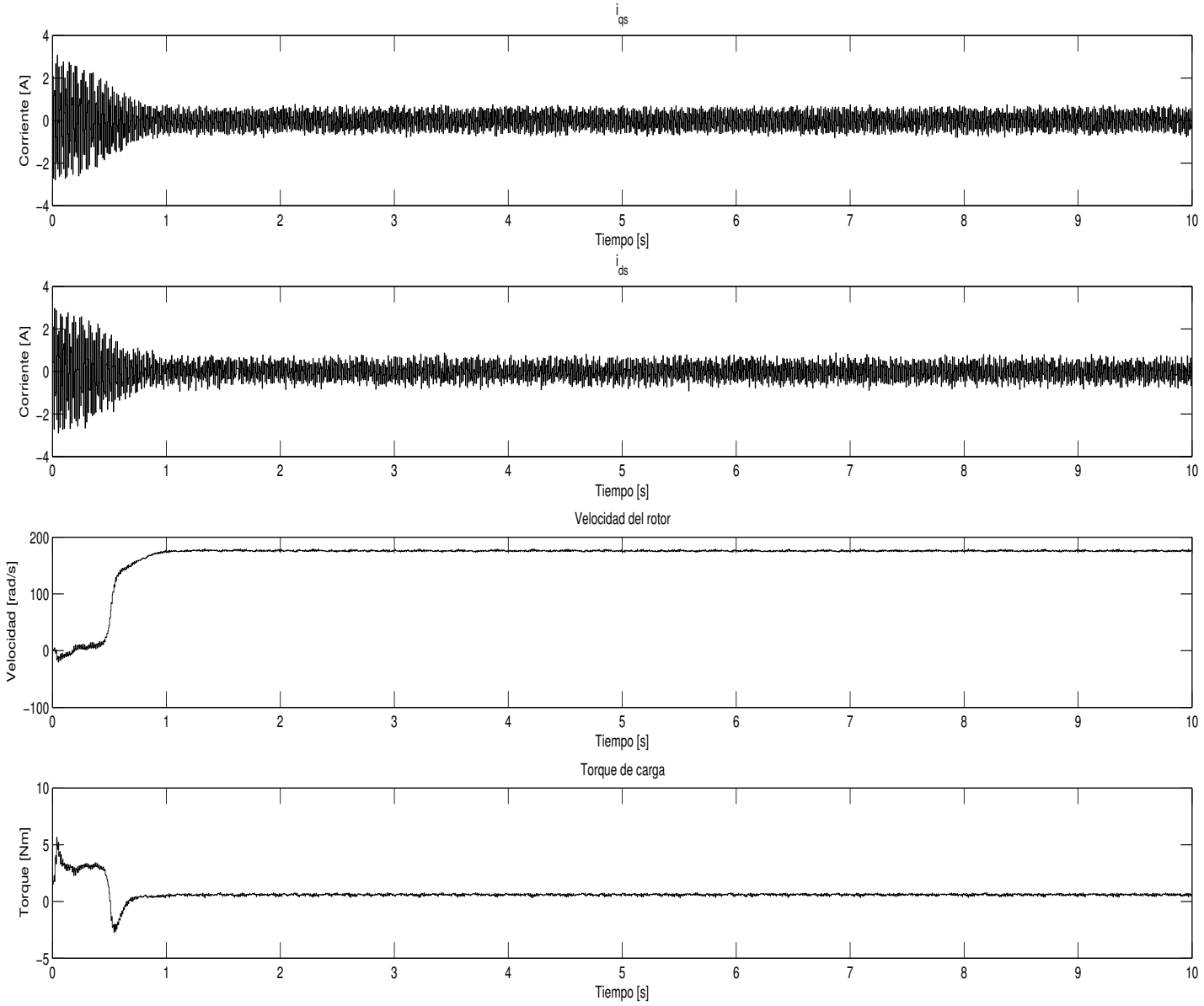


Figura 4.35: Estimación Kalman a 30Hz en el arranque sin carga (Corrientes, velocidad y torque)

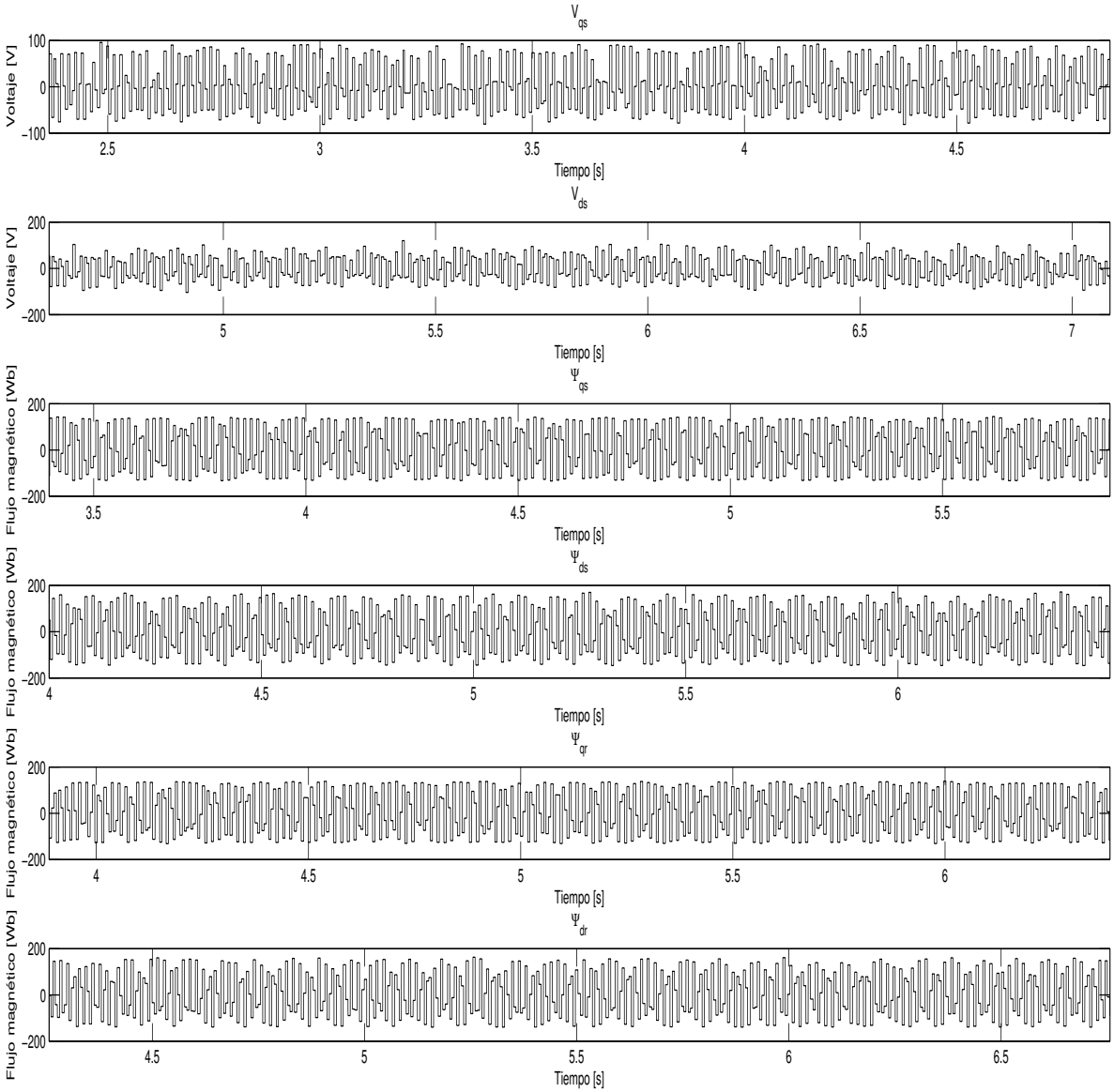


Figura 4.36: Estimación Kalman a 30Hz en estado estable sin carga (Voltajes y Flujos)

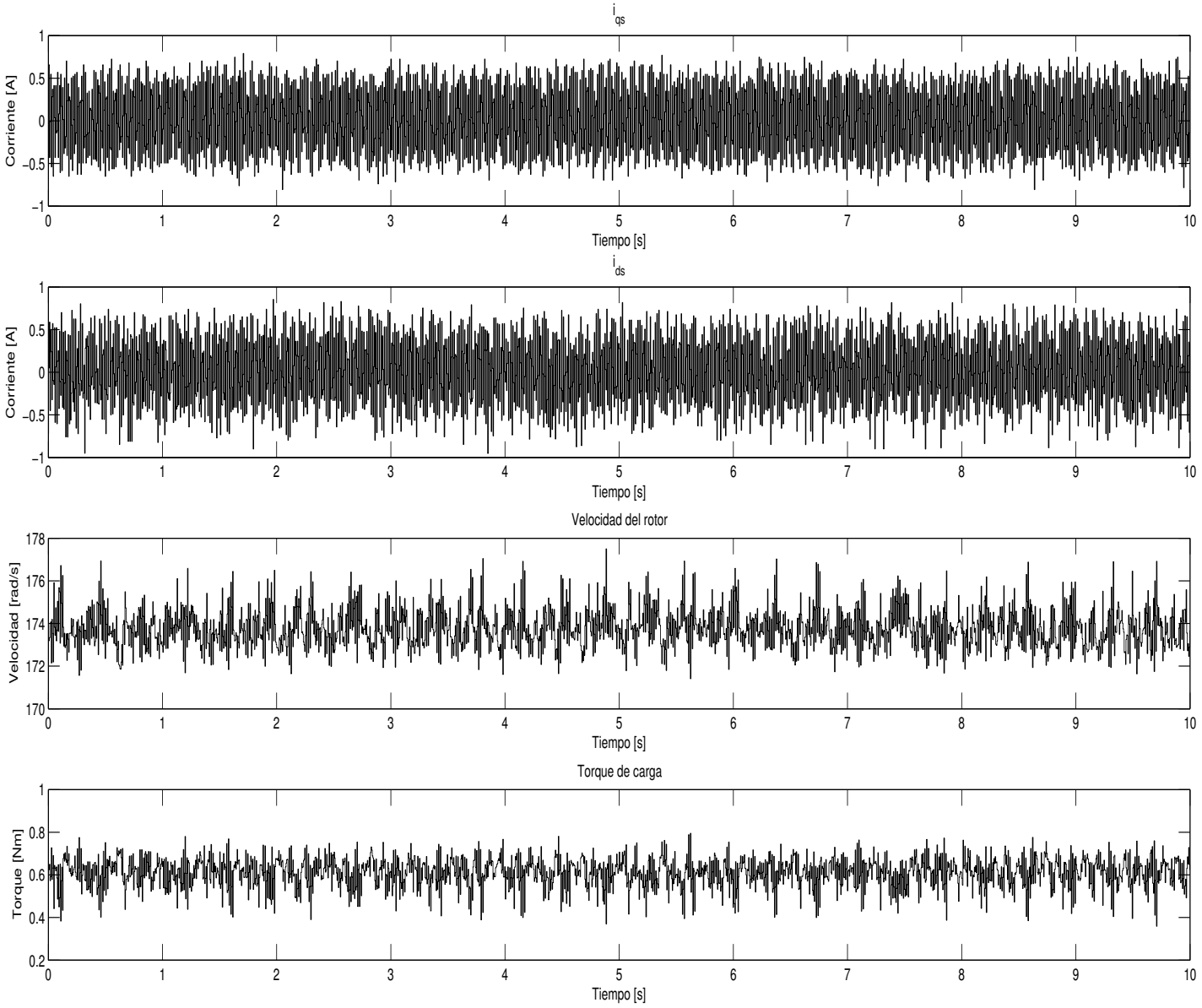


Figura 4.37: Estimación Kalman a 30Hz en estado estable sin carga (Corrientes, velocidad y torque)

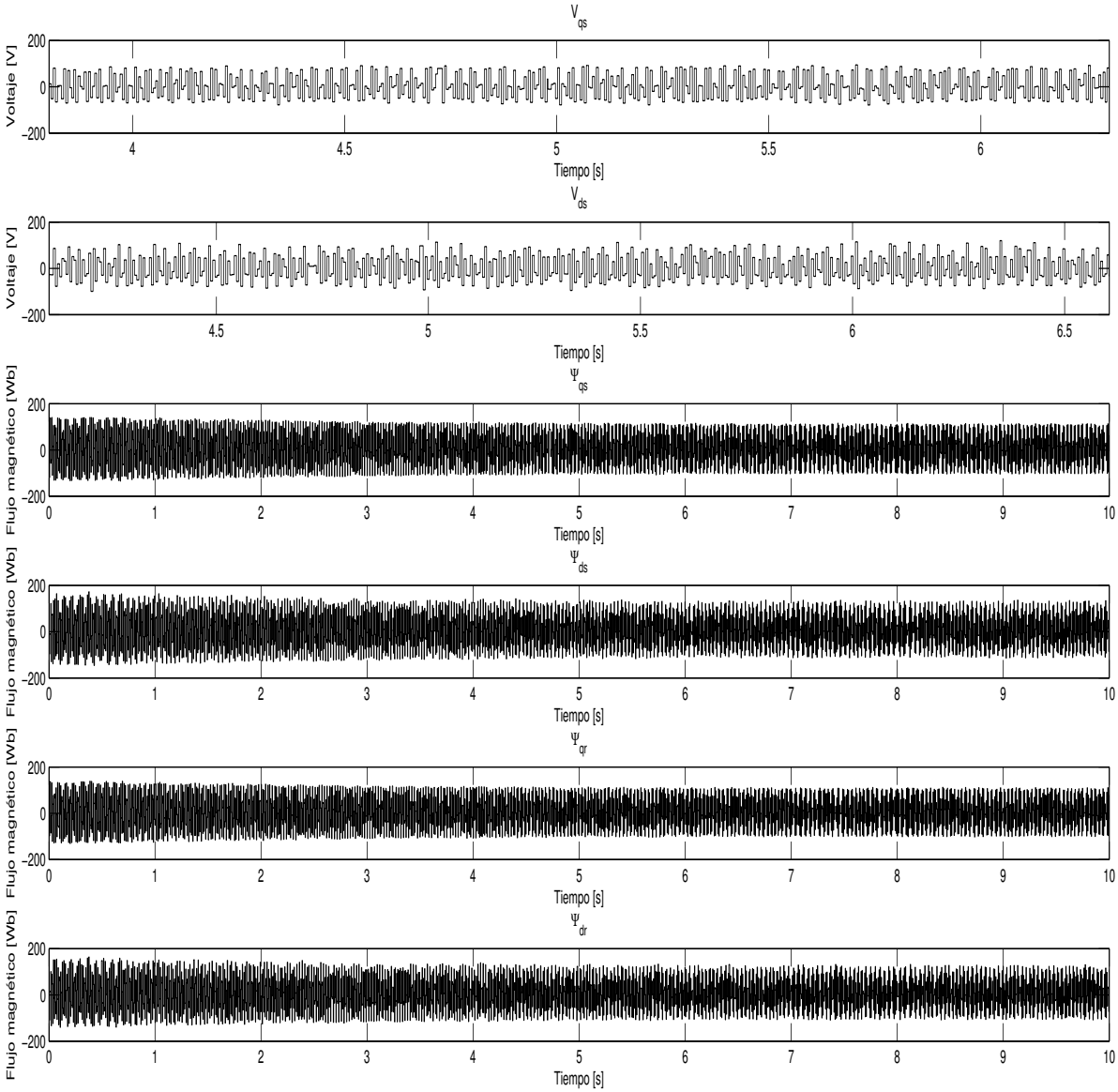


Figura 4.38: Estimación Kalman con cambio de 30Hz a 36.5Hz (Voltajes y Flujos)

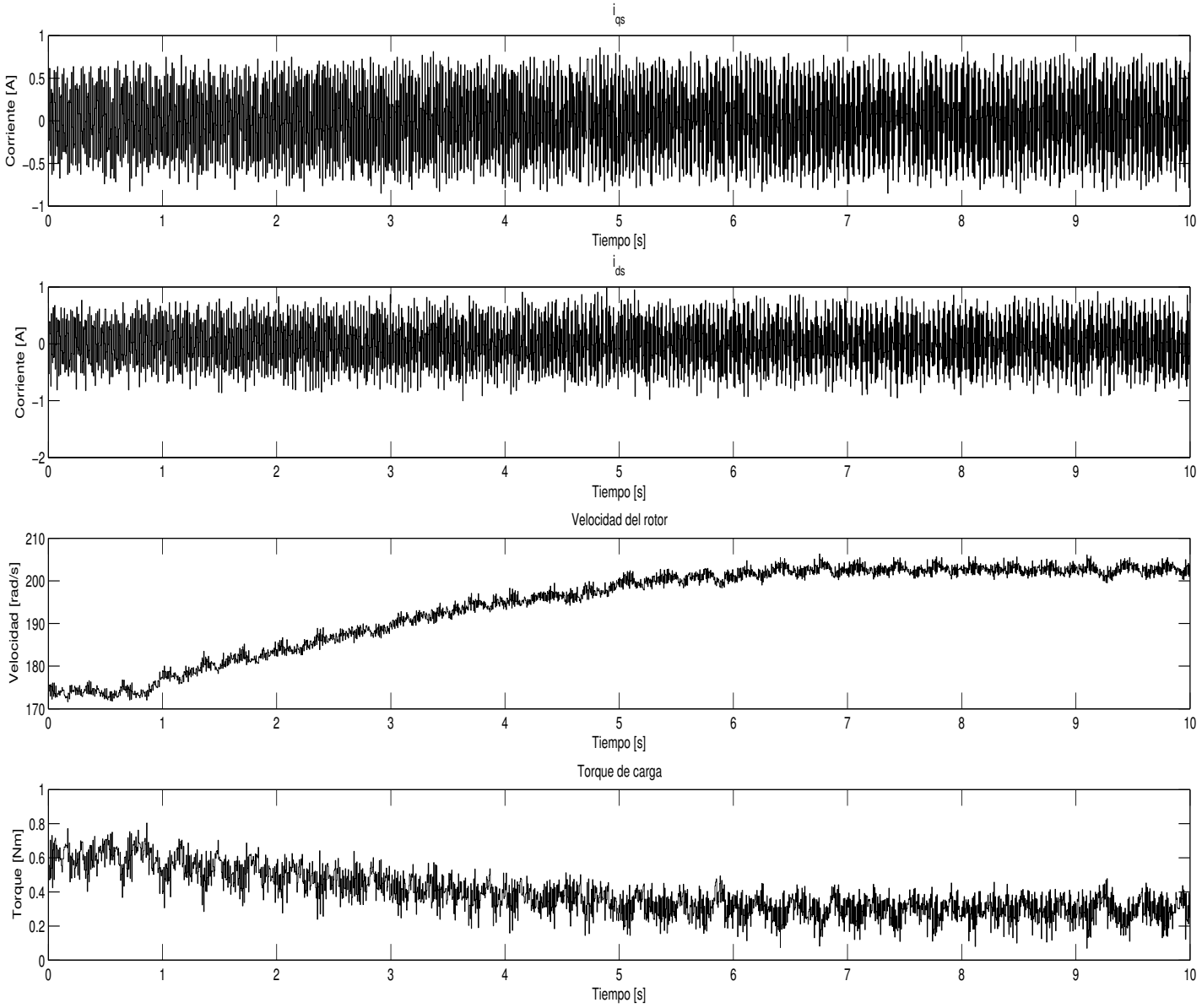


Figura 4.39: Estimación Kalman con cambio de 30Hz a 36.5Hz (Corrientes, velocidad y torque)

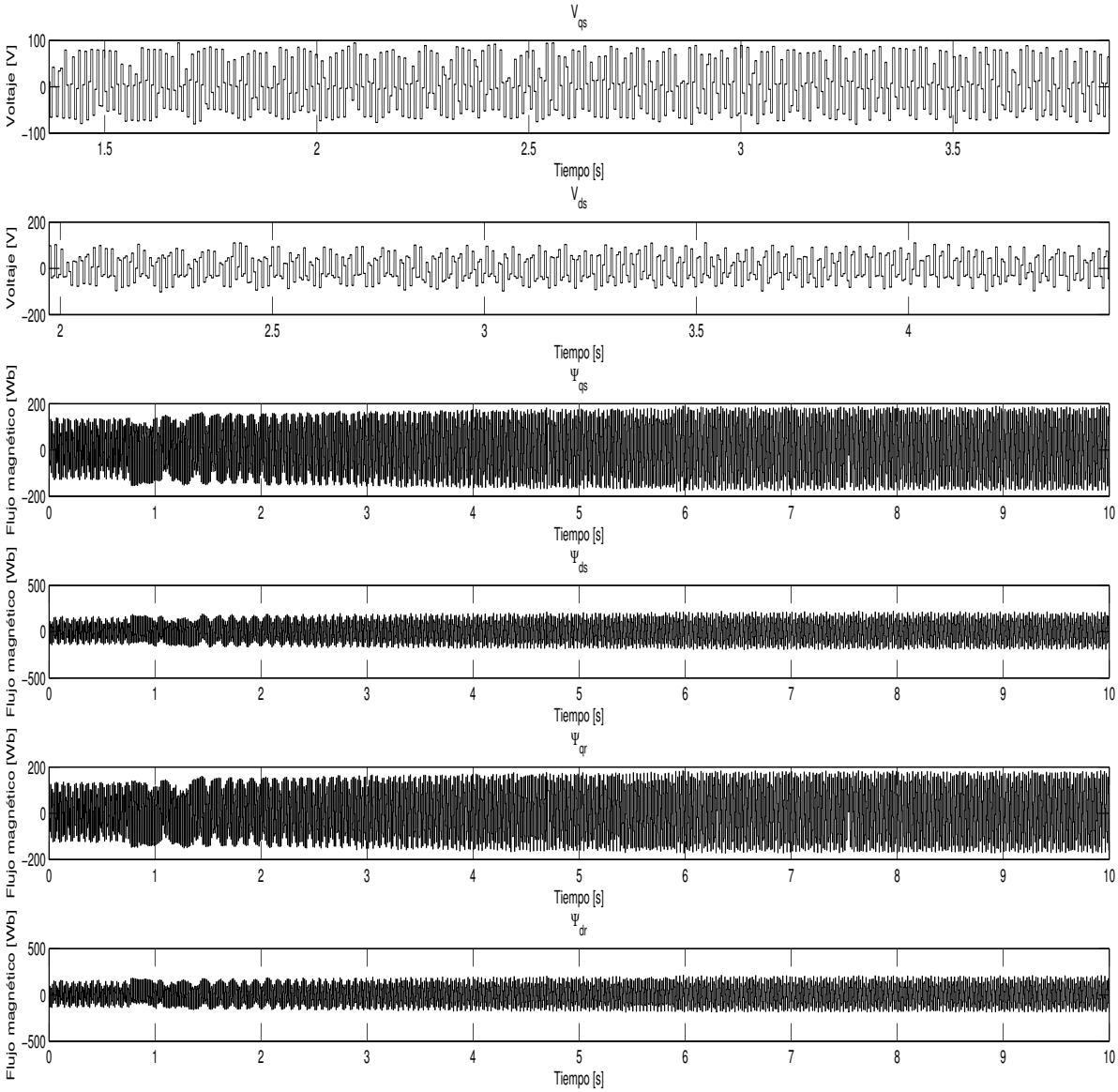


Figura 4.40: Estimación Kalman con cambio de 30Hz a 24 Hz (Voltajes y Flujos)

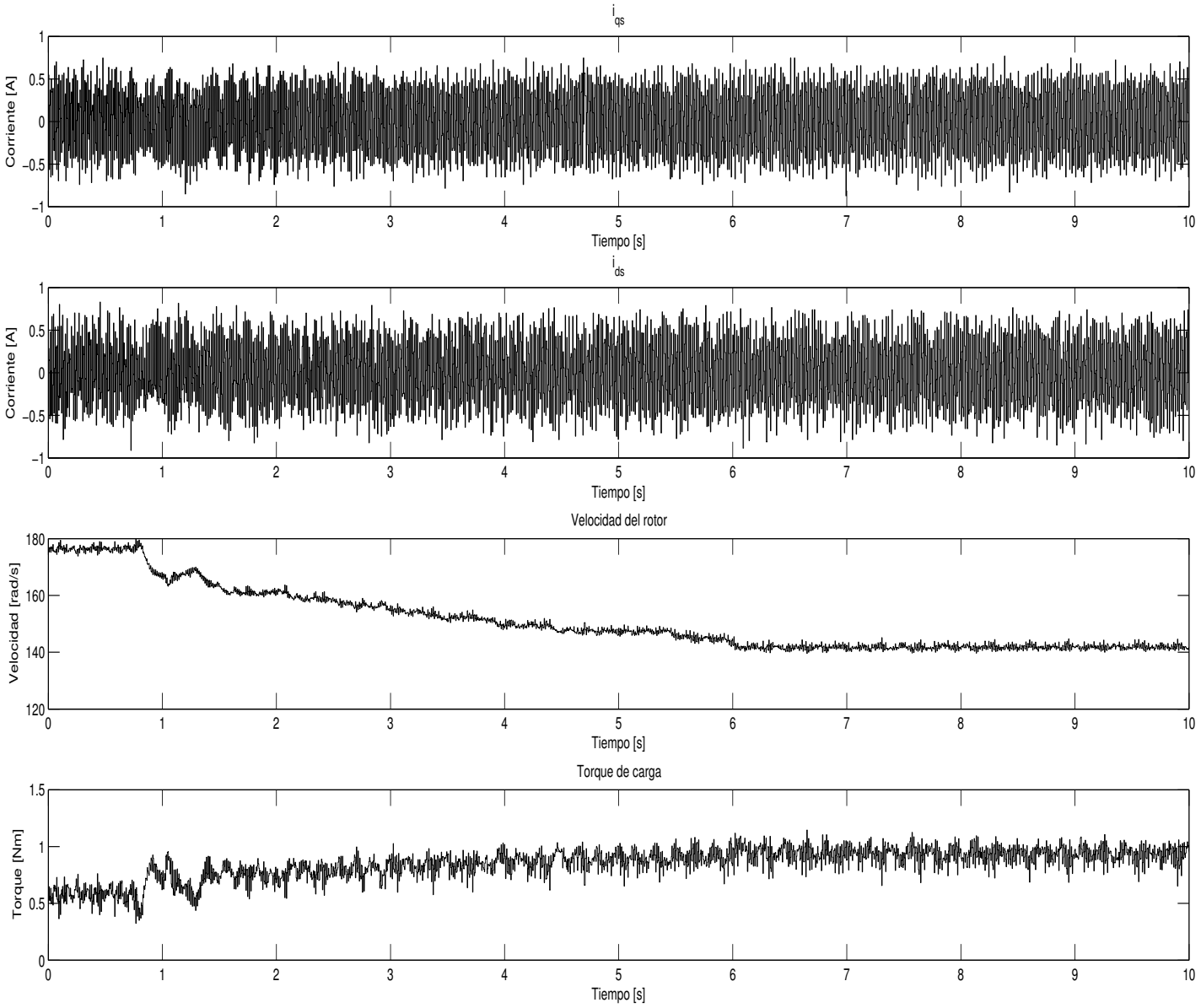


Figura 4.41: Estimación Kalman con cambio de 30Hz a 24 Hz (Corrientes, velocidad y torque)

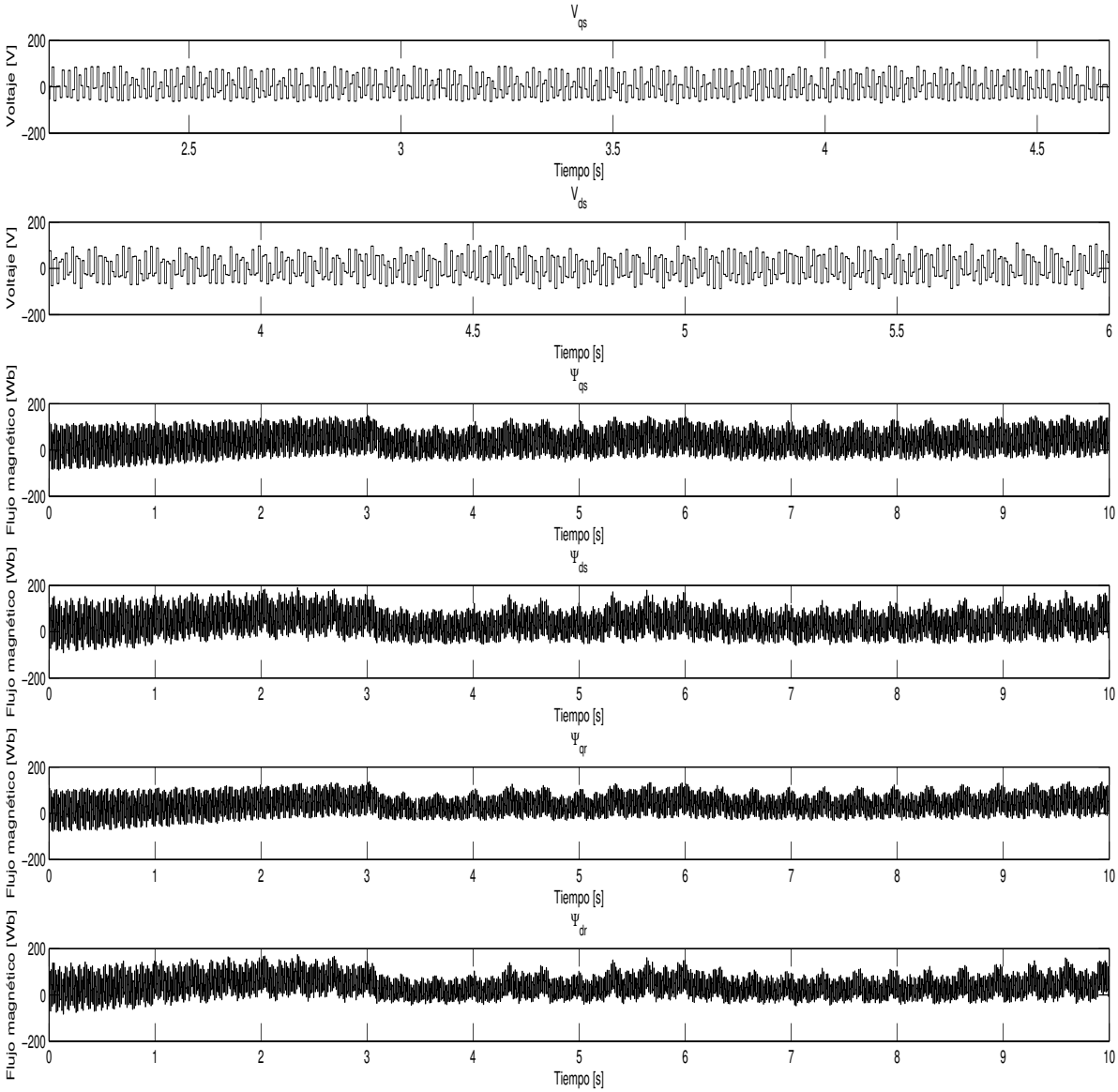


Figura 4.42: Estimación Kalman con frenado total (Voltajes y Flujos)

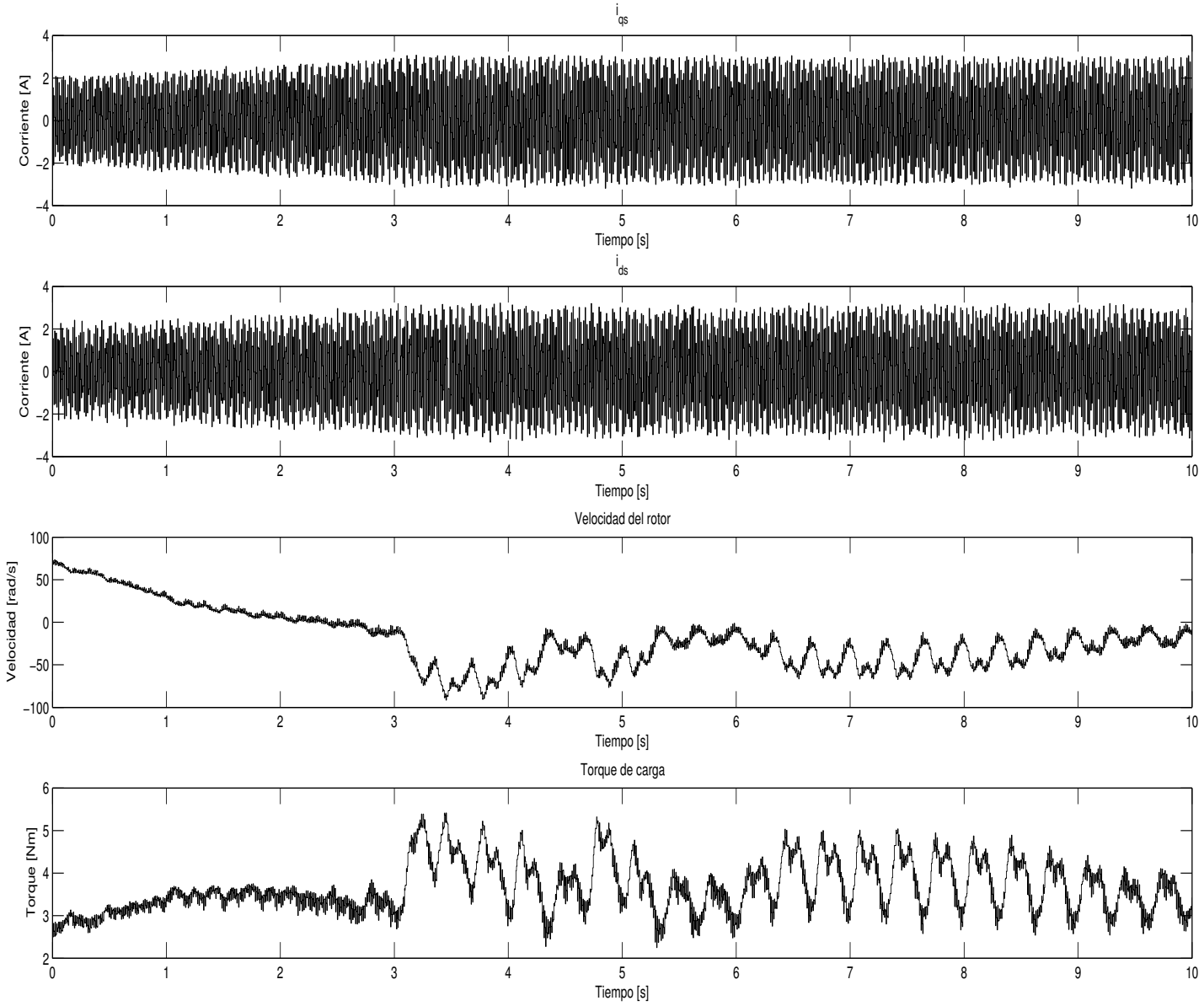


Figura 4.43: Estimación Kalman con frenado total (Corrientes, velocidad y torque)

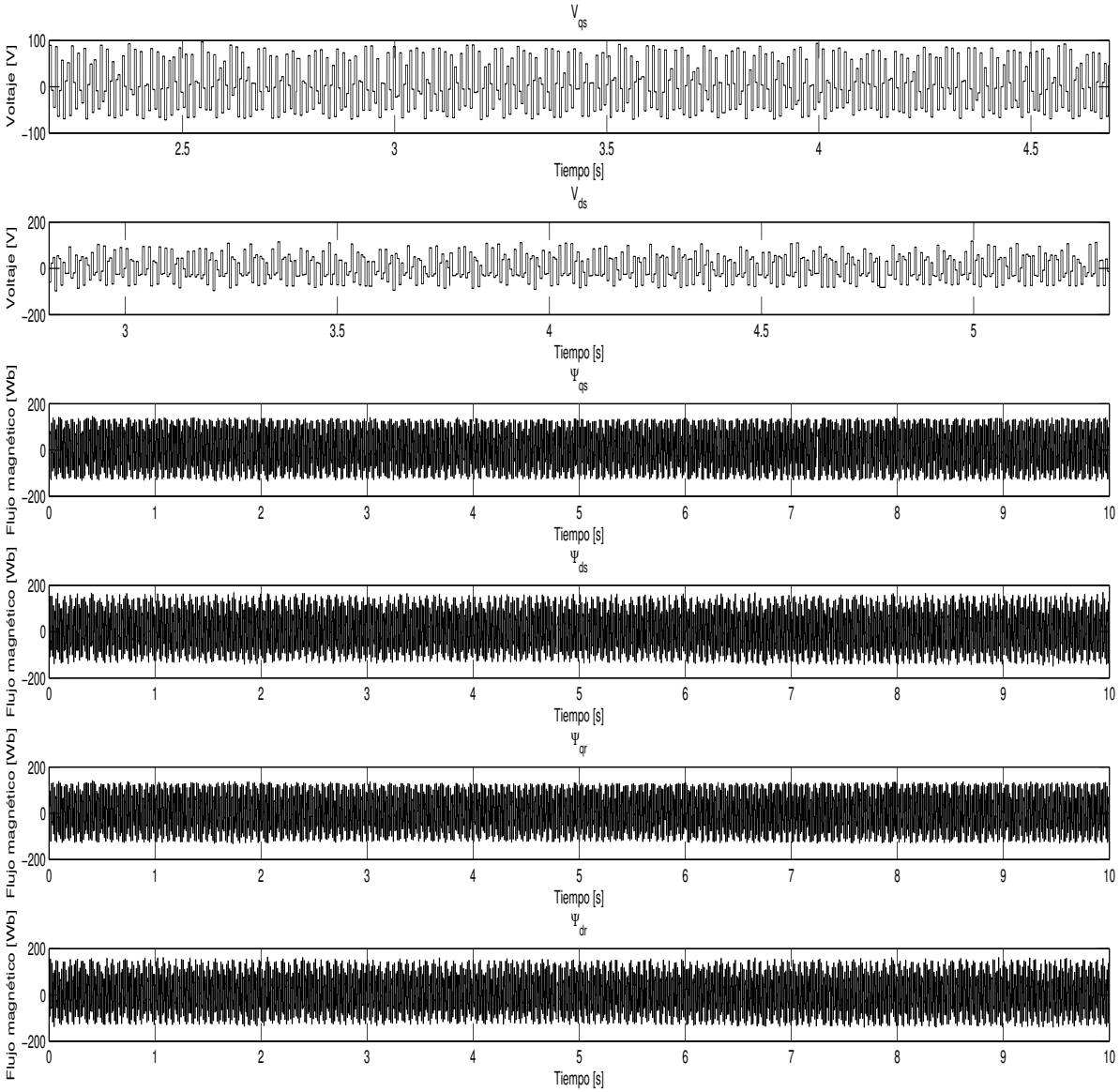


Figura 4.44: Estimación Kalman con cambios de carga (Voltajes y Flujos)

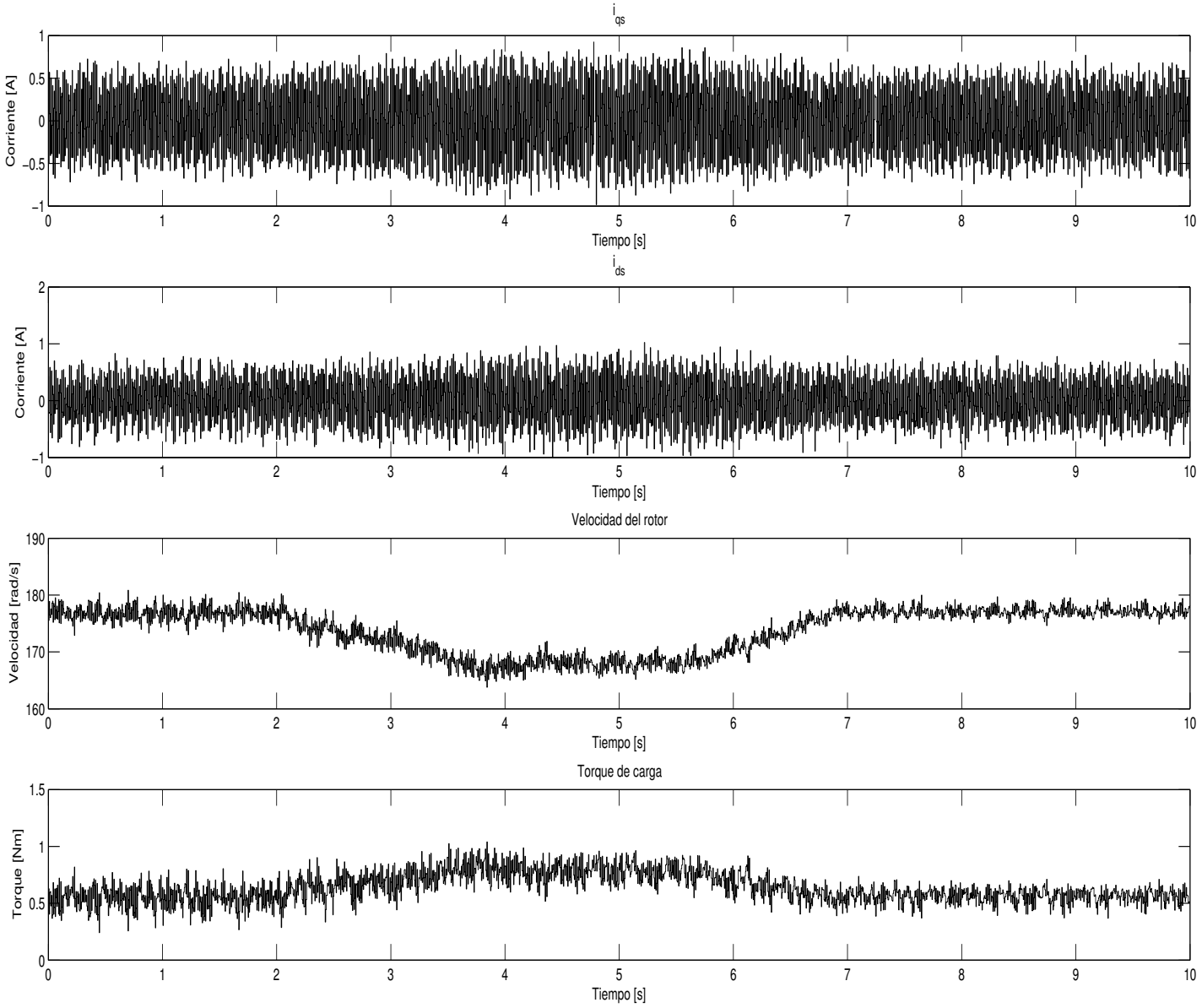


Figura 4.45: Estimación Kalman con cambios de carga (Corrientes, velocidad y torque)

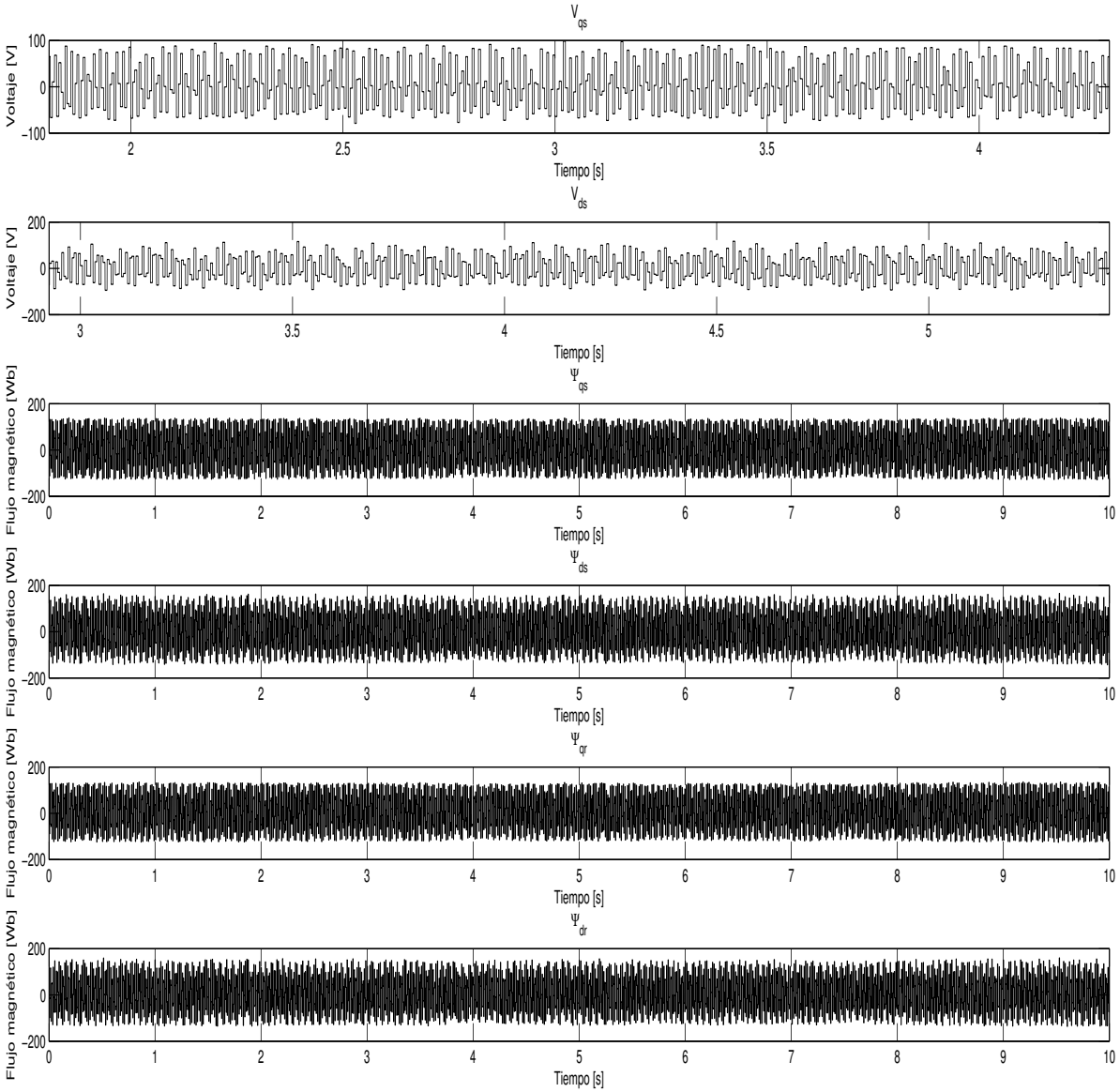


Figura 4.46: Estimación Kalman con cambios de carga múltiples (Voltajes y Flujos)

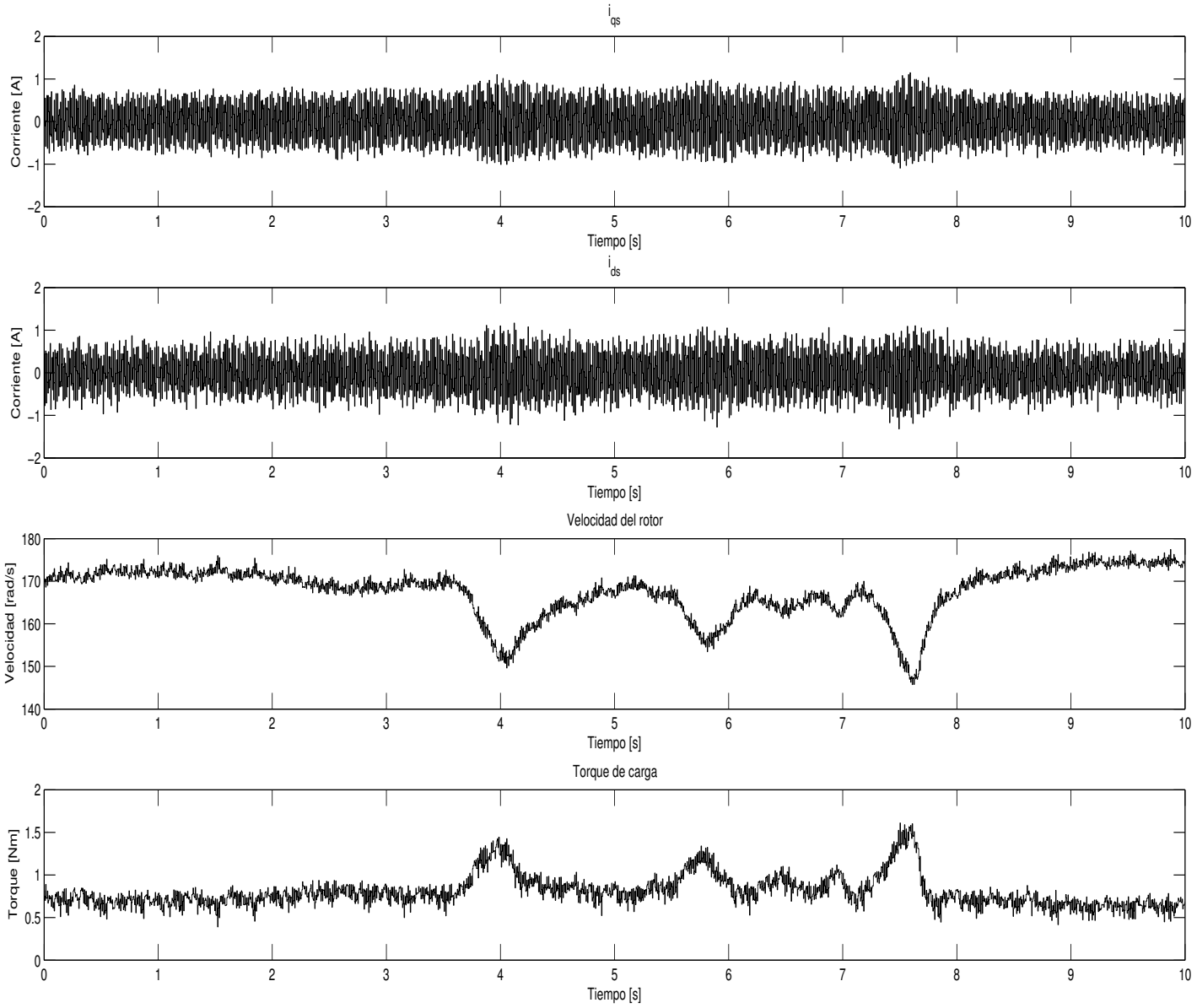


Figura 4.47: Estimación Kalman con cambios de carga múltiples (Corrientes, velocidad y torque)

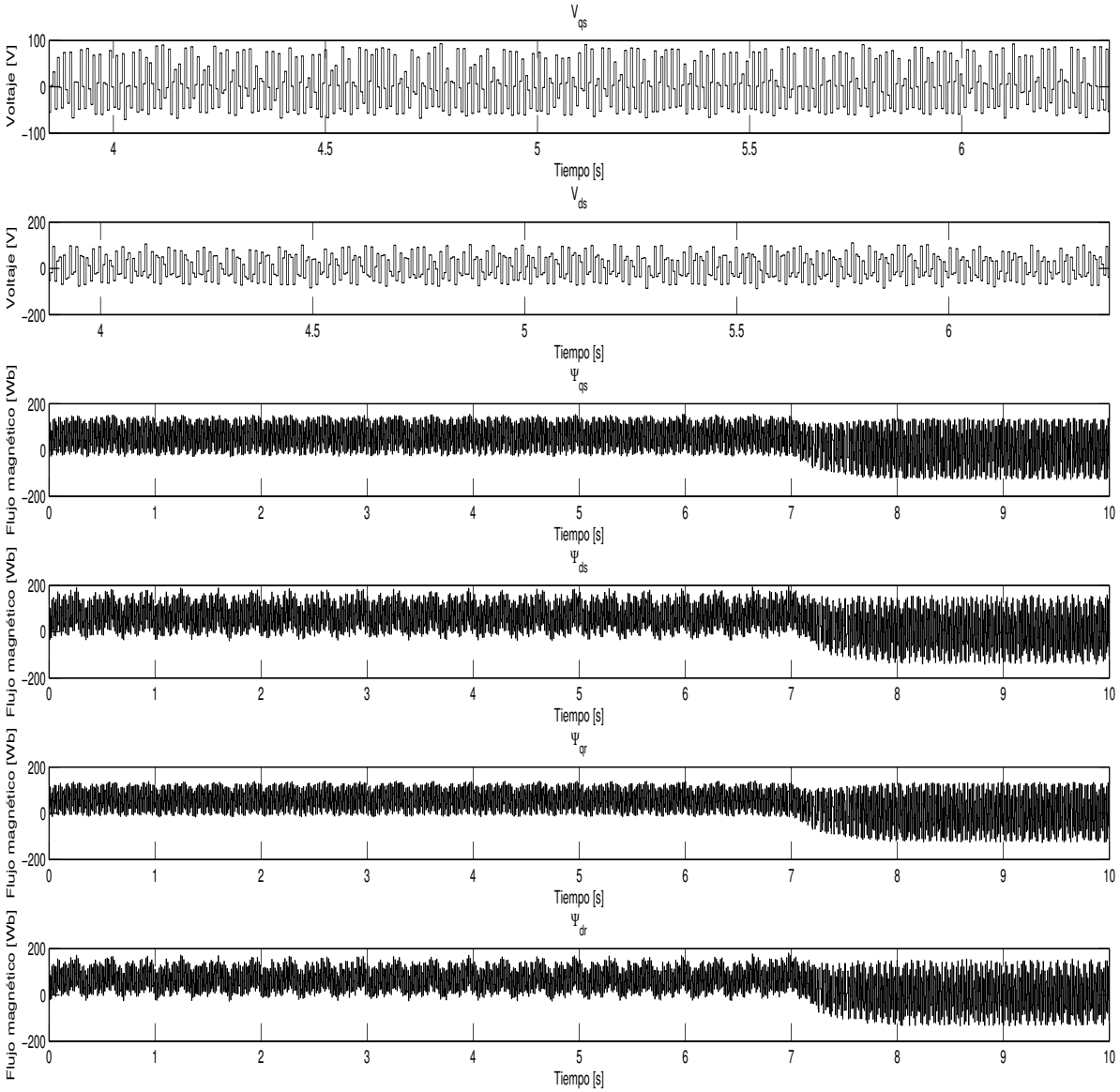


Figura 4.48: Estimación Kalman con carga total al arranque (Voltajes y Flujos)

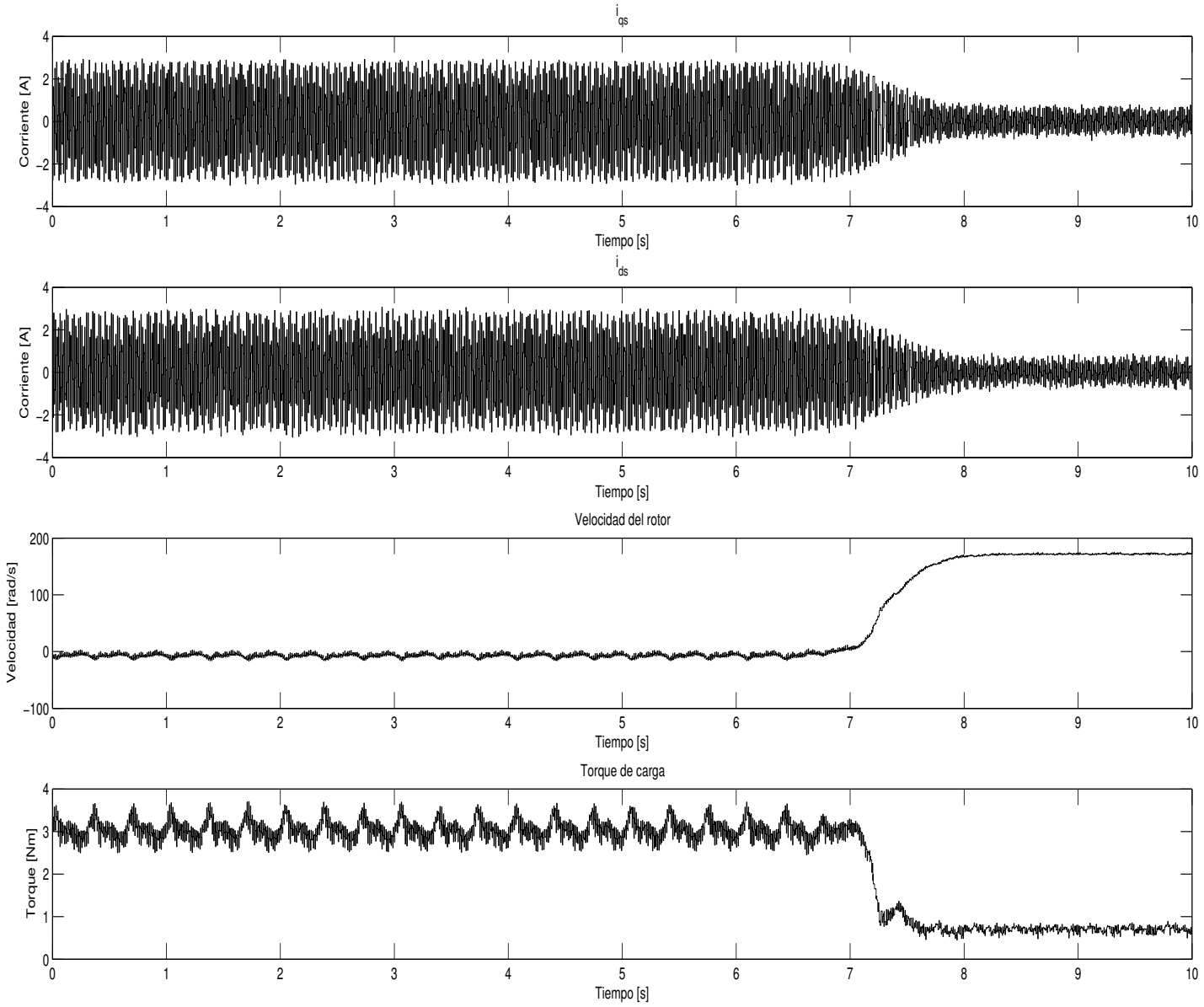


Figura 4.49: Estimación Kalman con carga total al arranque (Corrientes, velocidad y torque)

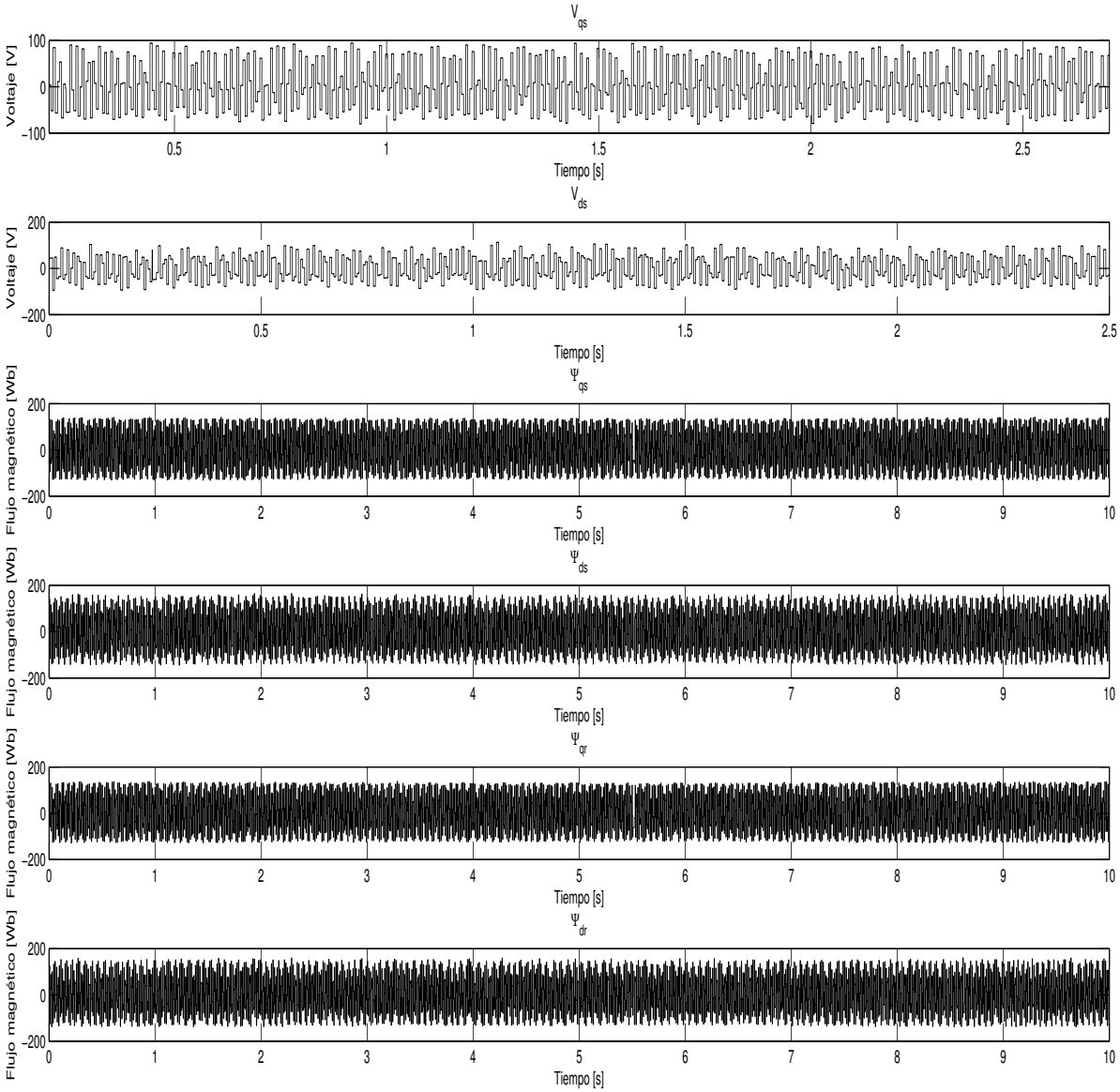


Figura 4.50: Estimación Kalman con carga y posterior liberación de carga (Voltajes y Flujos)

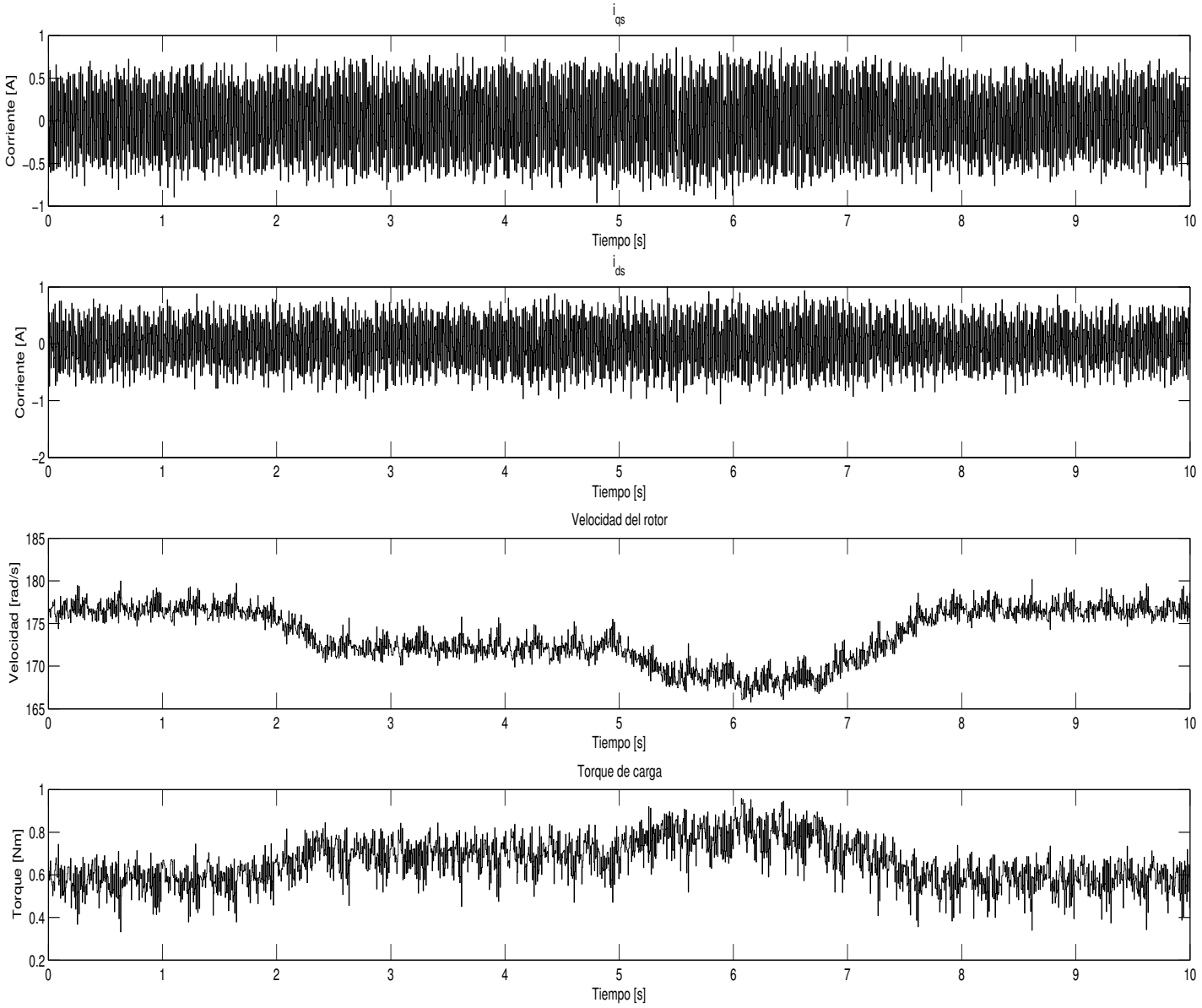


Figura 4.51: Estimación Kalman con carga y posterior liberación de carga (Corrientes, velocidad y torque)

Las Figuras 4.34 y 4.35 muestran el comportamiento del filtro y las variables estimadas al arranque con una frecuencia del voltaje de alimentación de 30Hz. Se observan las transcientes en los flujos y el arranque del motor. En el caso de la velocidad se aprecia la curva de arranque tal como se esperaría y el sistema se estabiliza a aproximadamente 175rad/s. En el caso

del torque de carga se tiene un pico inicial y posteriormente el sistema se estabiliza. Como ya se mencionó en las pruebas anteriores la estimación trata la reducción de frecuencia de alimentación como una perturbación de carga. En este caso, por operar a 30Hz, el filtro estima un torque de 0.6Nm incluso al no tener carga inicial.

En las Figuras 4.36 y 4.37 se muestra el comportamiento del sistema en estado estable con una frecuencia de alimentación de 30Hz. Las oscilaciones en estado estable son de aproximadamente 4rad/s en el caso de la velocidad del rotor y 0.4Nm en el caso del torque, aunque el valor medio es esta variable es de 0.6Nm. El estimador no distingue desaceleración del motor por variación de frecuencia o aumento de carga y la variable torque de carga se ve estimada como si fuera una perturbación de carga.

La respuesta del filtro a un cambio de frecuencia de 30 a 36.5 Hz se observa en las Figuras 4.38 y 4.39. La velocidad del rotor aumenta debido al aumento de frecuencia y como se esperaba, la estimación de torque disminuye ya que el filtro ve el cambio de frecuencia como si fuera un proceso de liberación de carga. En los dos casos se ven pequeñas oscilaciones sostenidas en la estimación.

Si se varía la frecuencia de operación de 30 a 24Hz se tiene la respuesta mostrada en las Figuras 4.40 y 4.41. Se observa que la velocidad del rotor disminuye hasta estabilizarse, mientras que el torque de carga aumenta ya que el sistema no distigue entre perturbaciones de frecuencia y perturbaciones de carga. La oscilación en velocidad durante la transición es baja. En el caso del torque la oscilación es mayor pero tiende a estabilizarse.

La respuesta del filtro a una carga tal que produzca frenado total se observa en las Figuras 4.42 y 4.43. Se aprecia que el filtro no converge rápidamente a 0rad/s como en el caso de la operación a 50Hz. El filtro primero estima velocidades negativas y posteriormente oscilaciones amortiguadas con tendencia de convergencia hacia cero. La convergencia es lenta. En el caso del torque de carga igualmente se observa que el filtro converge hacia un valor estable de 2.6 Nm. En el caso de 50Hz convergía a 2Nm, sin embargo se debe considerar también los 0.6Nm de desviación por efectos de frecuencia de operación. No obstante, al igual que la velocidad, el torque oscila fuertemente llegando a picos de 5Nm, aunque estos se atenúan y se observa una tendencia a converger.

Las Figuras 4.44 y 4.45 muestran los efectos de aumento y disminución de carga en el motor. La velocidad rotórica disminuye para posteriormente aumentar al liberarse la carga. No hay mayor oscilación durante la operación. De igual manera en el caso del torque de carga se observa la respuesta a las perturbaciones de carga con pequeñas oscilaciones en la respuesta estimada.

Las Figuras 4.46 y 4.47 muestran la respuesta de la estimación cuando se realizan varias perturbaciones de carga en un periodo corto de tiempo. Se observa como las aceleraciones y frenados con carga se traducen en picos tanto en la estimación de velocidad del rotor como en el torque de carga. Con esto se prueba que el filtro sigue la dinámica del proceso de buena forma.

Se muestra la respuesta del filtro cuando el motor arranca al 100 % de su carga en las Figuras 4.48 y 4.49. En el caso de los flujos magnéticos se observa claramente como varía la estimación y cómo estas variables cambian en el tiempo al liberarse la carga. De igual manera se aprecia claramente el cambio significativo en las corrientes del estator medidas. En el caso

de la velocidad, se observa como el filtro oscila alrededor de cero y al momento de producirse la liberación de carga arranca hasta la velocidad de 175rad/s establecida por la frecuencia de operación. De igual manera el torque de carga disminuye al valor final de 0.6Nm que es la desviación estimada por operar a 30Hz.

Las Figuras 4.50 y 4.51 muestran la respuesta del filtro cuando existe un proceso de carga, aumento de carga y posterior liberación pero de forma no instantánea. Se aprecian las oscilaciones pequeñas de estimación durante los transitorios, pero en general la respuesta del filtro es la esperada.

4.3.3. Filtro Kalman extendido-Baja frecuencia (0-25 Hz)

Se realizaron las pruebas detalladas en la introducción de esta sección con una frecuencia de alimentación de 20Hz. A continuación se muestran los resultados. Como ya se mencionó se muestran los voltajes y corrientes $dq0$ del estator (medidos vía muestreo en el DSP) y las variables estimadas: flujos $dq0$ de estator y rotor, velocidad del rotor y torque de carga.

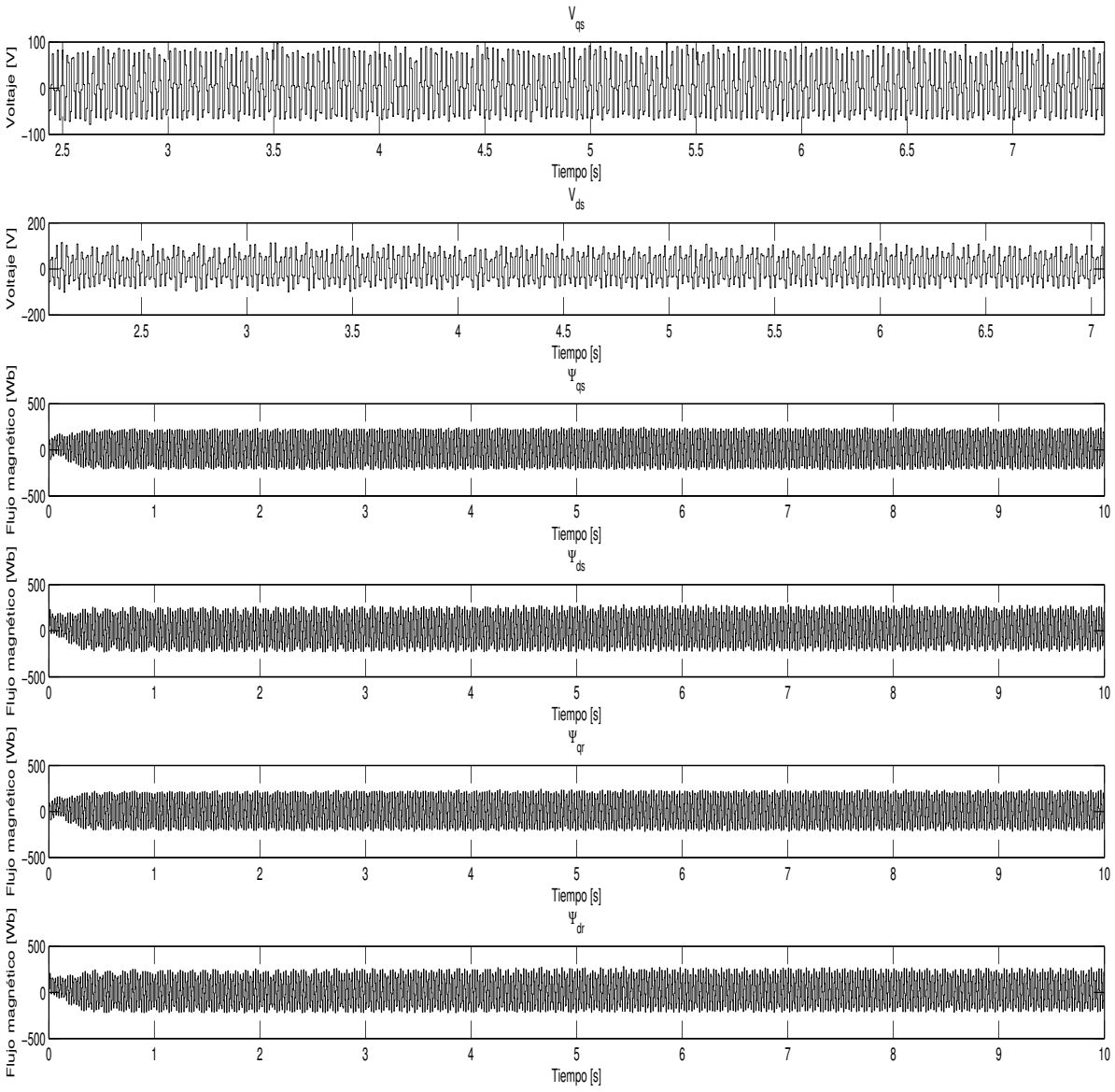


Figura 4.52: Estimación Kalman a 20Hz en el arranque sin carga(Voltajes y Flujos)

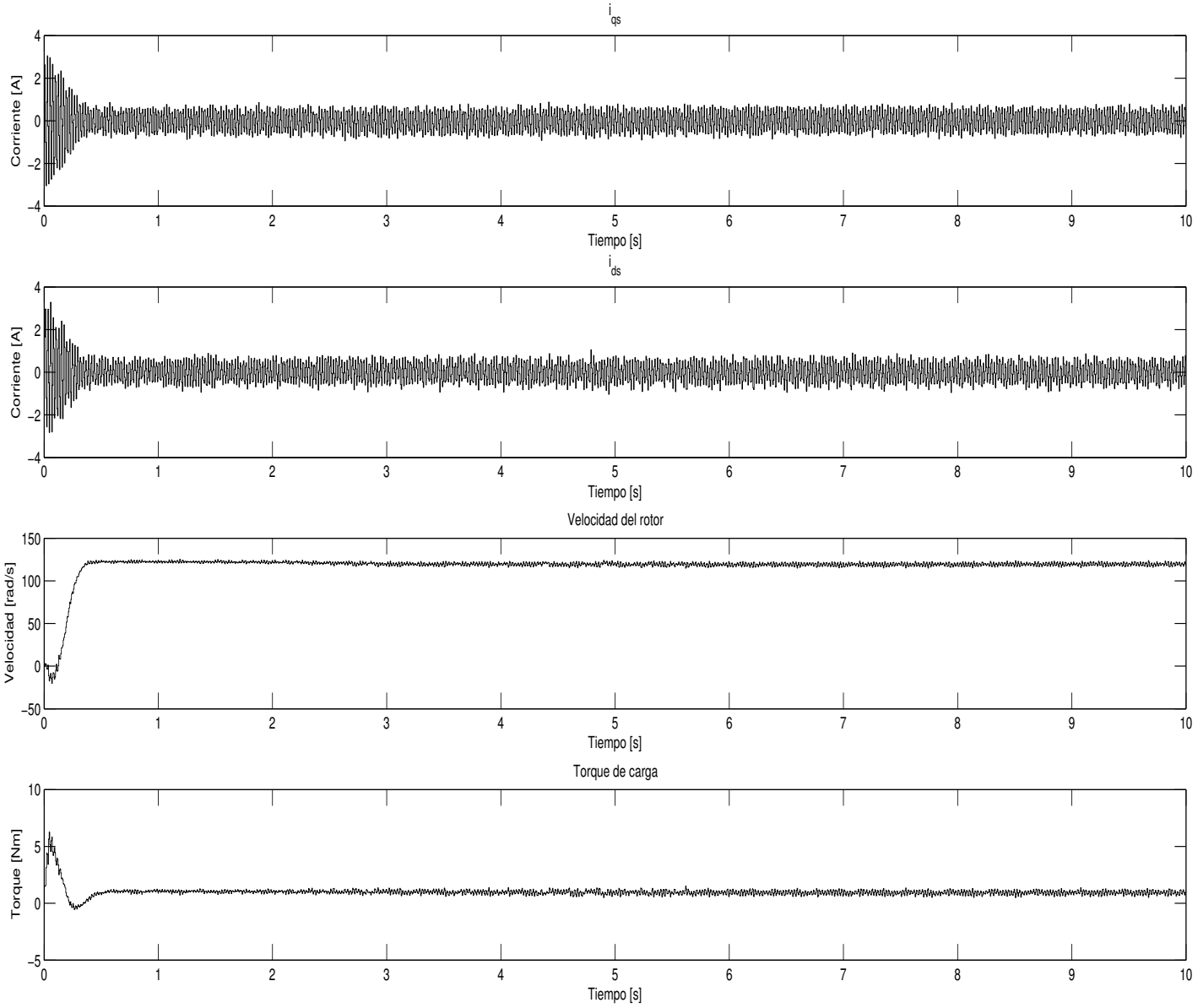


Figura 4.53: Estimación Kalman a 20Hz en el arranque sin carga(Corrientes, velocidad y torque)

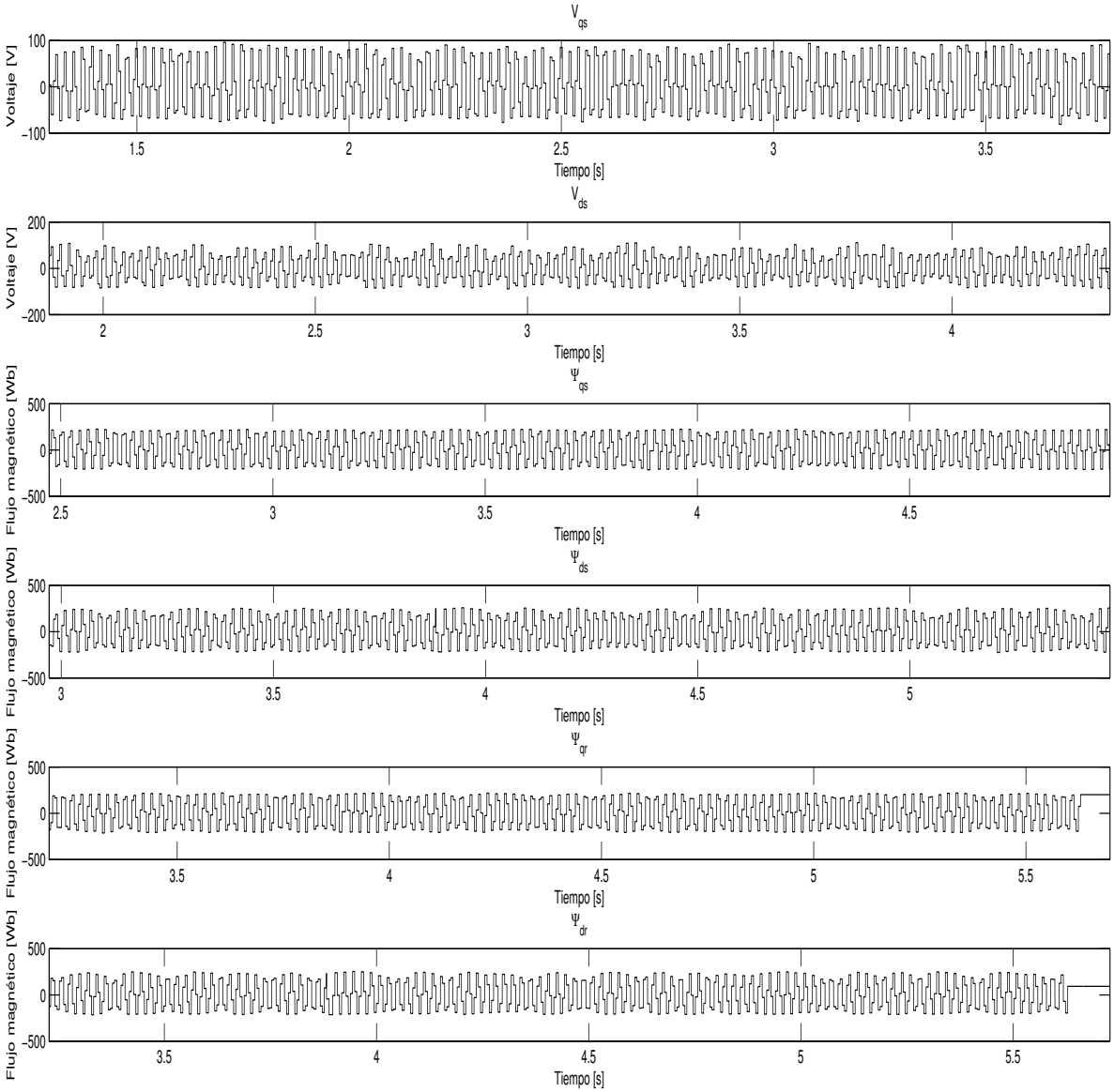


Figura 4.54: Estimación Kalman a 20Hz en estado estable sin carga (Voltajes y Flujos)

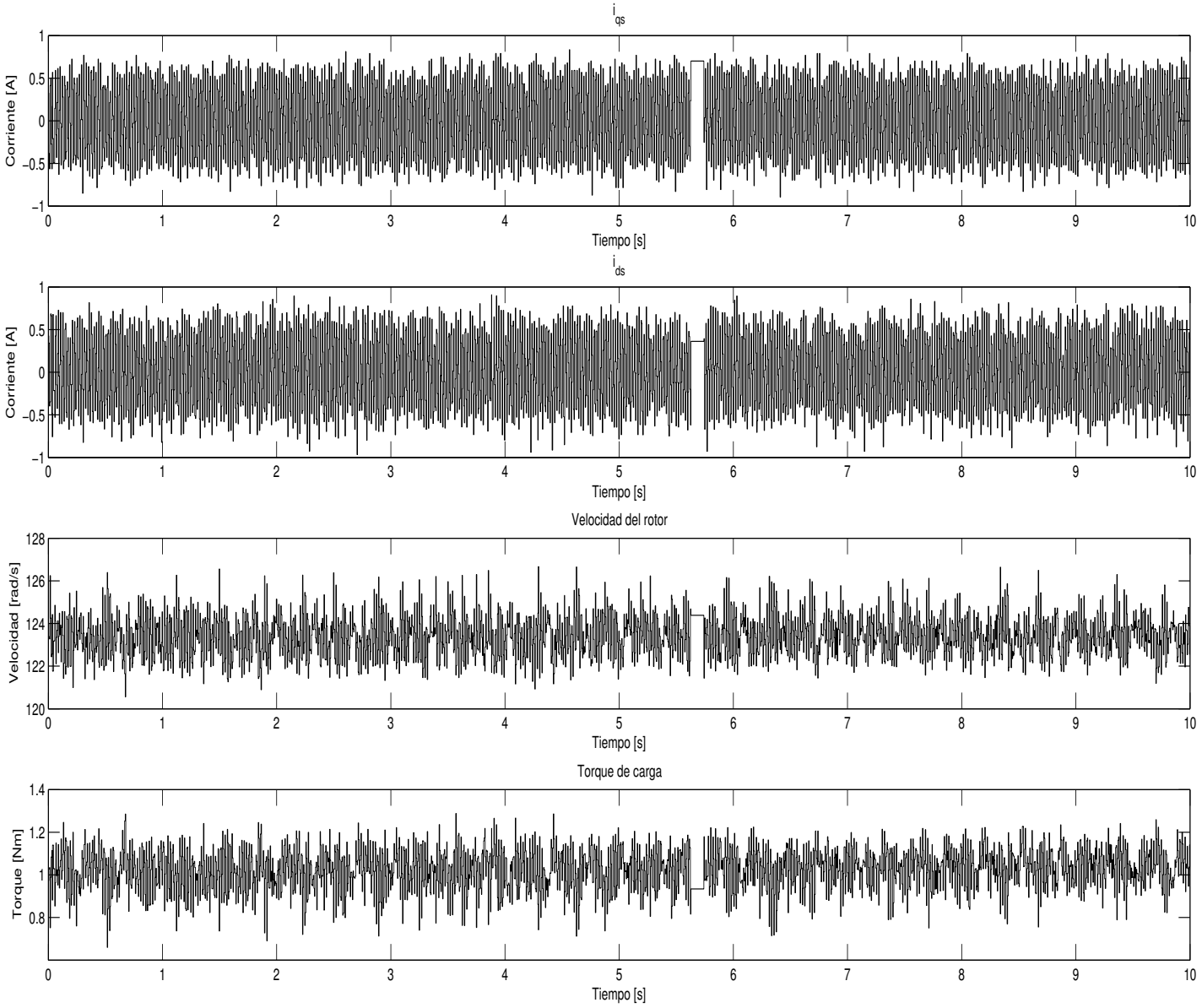


Figura 4.55: Estimación Kalman a 20Hz en estado estable sin carga (Corrientes, velocidad y torque)

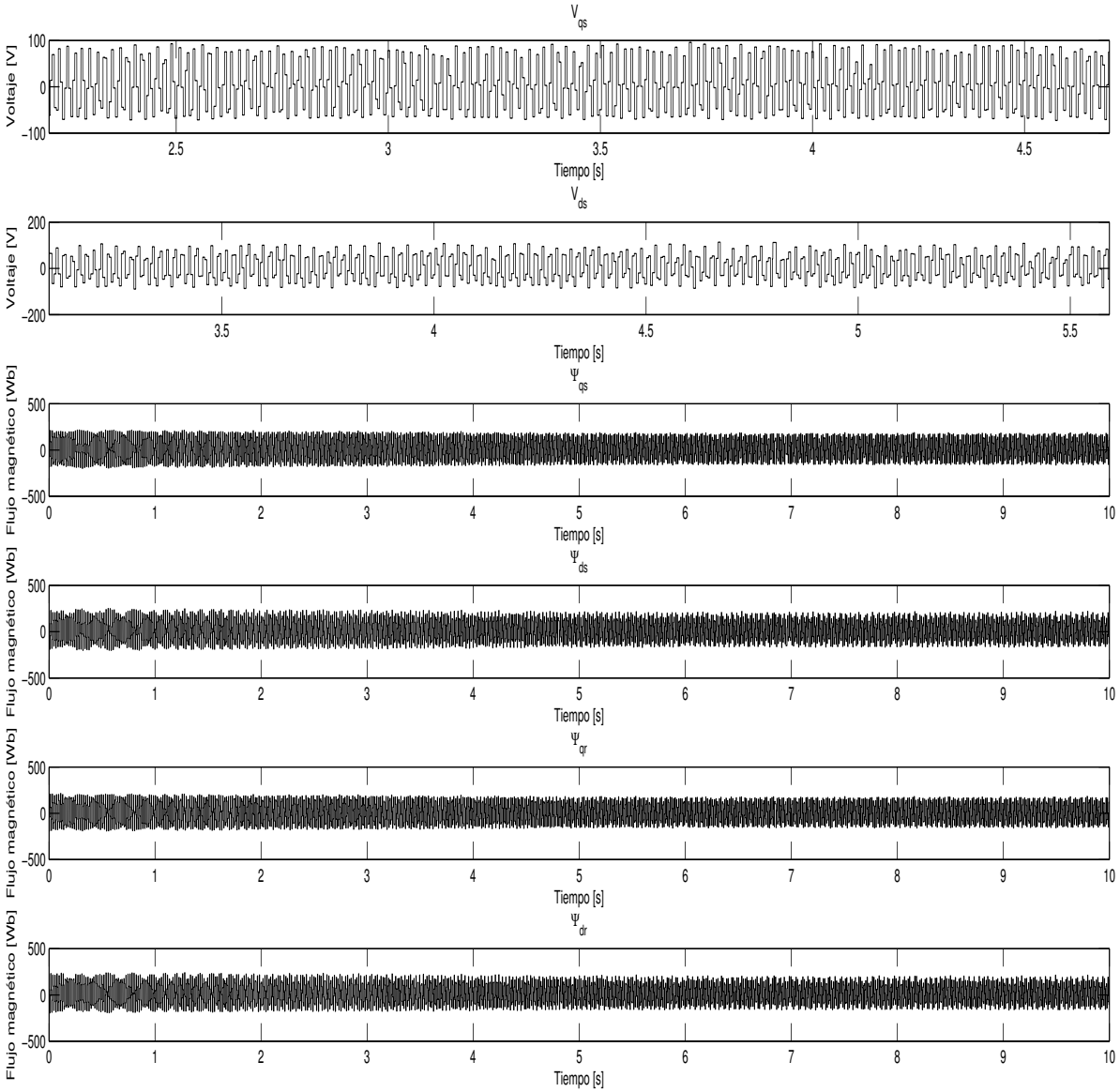


Figura 4.56: Estimación Kalman con cambio de 20Hz a 26Hz (Voltajes y Flujos)

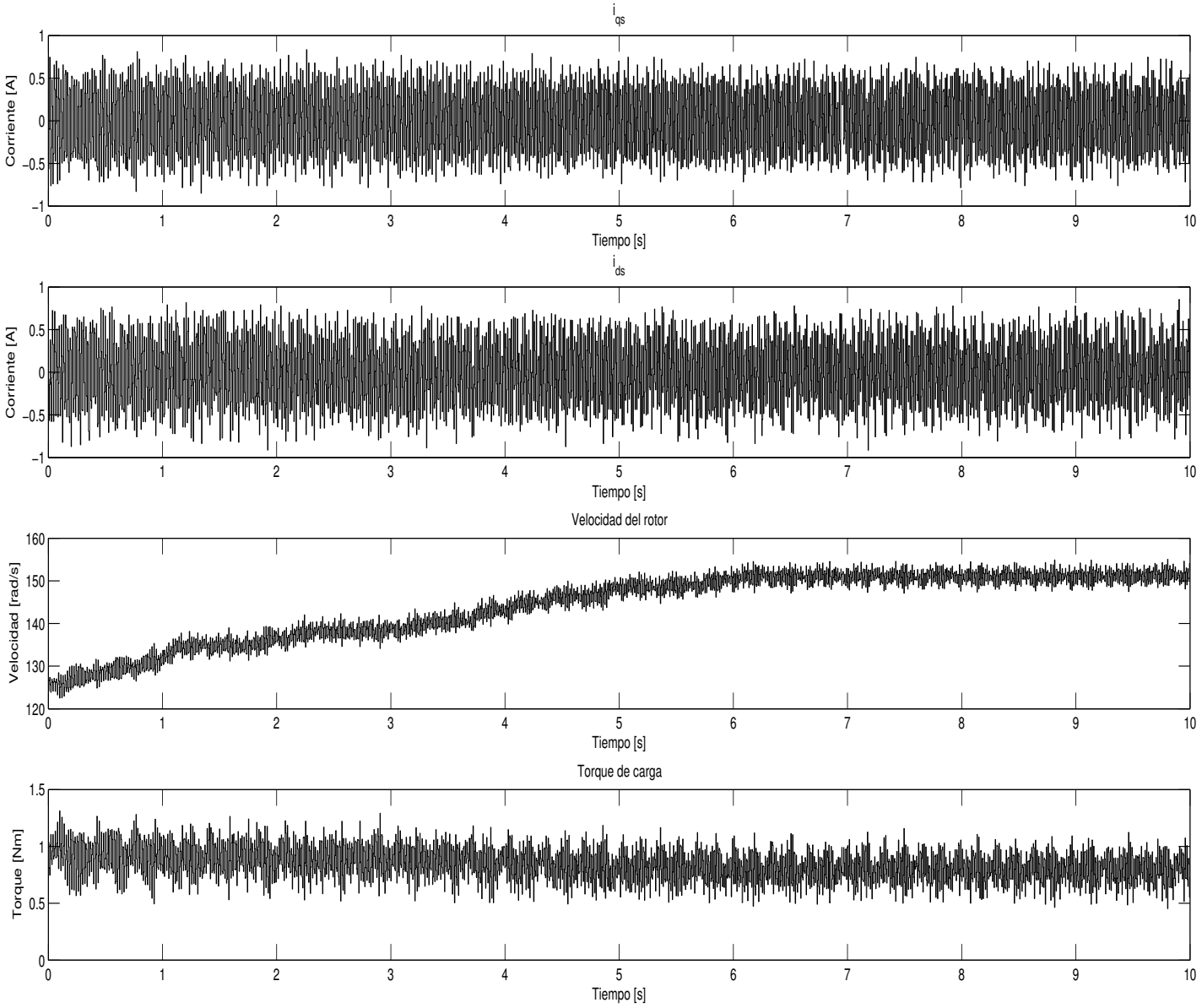


Figura 4.57: Estimación Kalman con cambio de 20Hz a 26Hz (Corrientes, velocidad y torque)

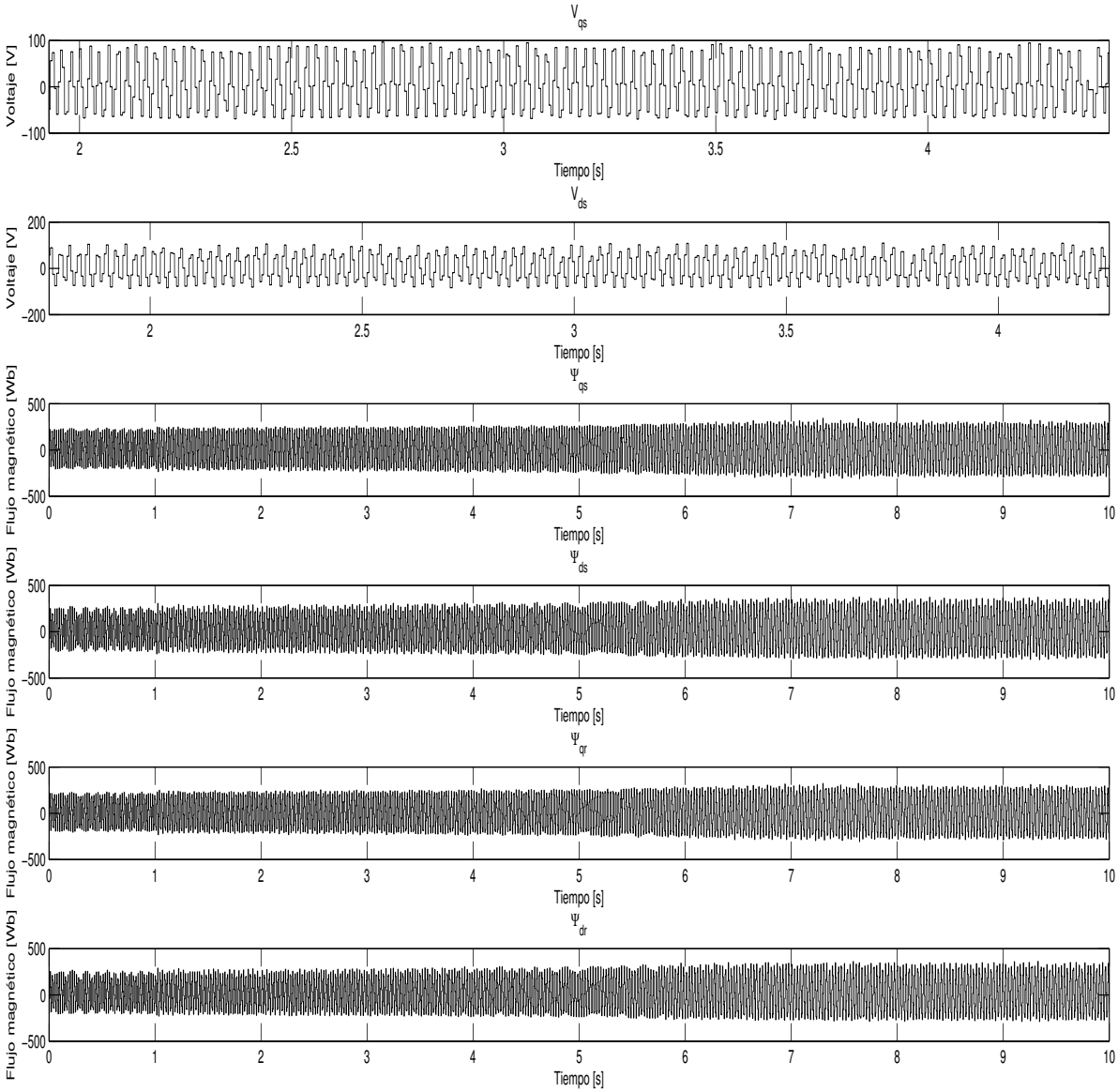


Figura 4.58: Estimación Kalman con cambio de 20Hz a 15 Hz (Voltajes y Flujos)

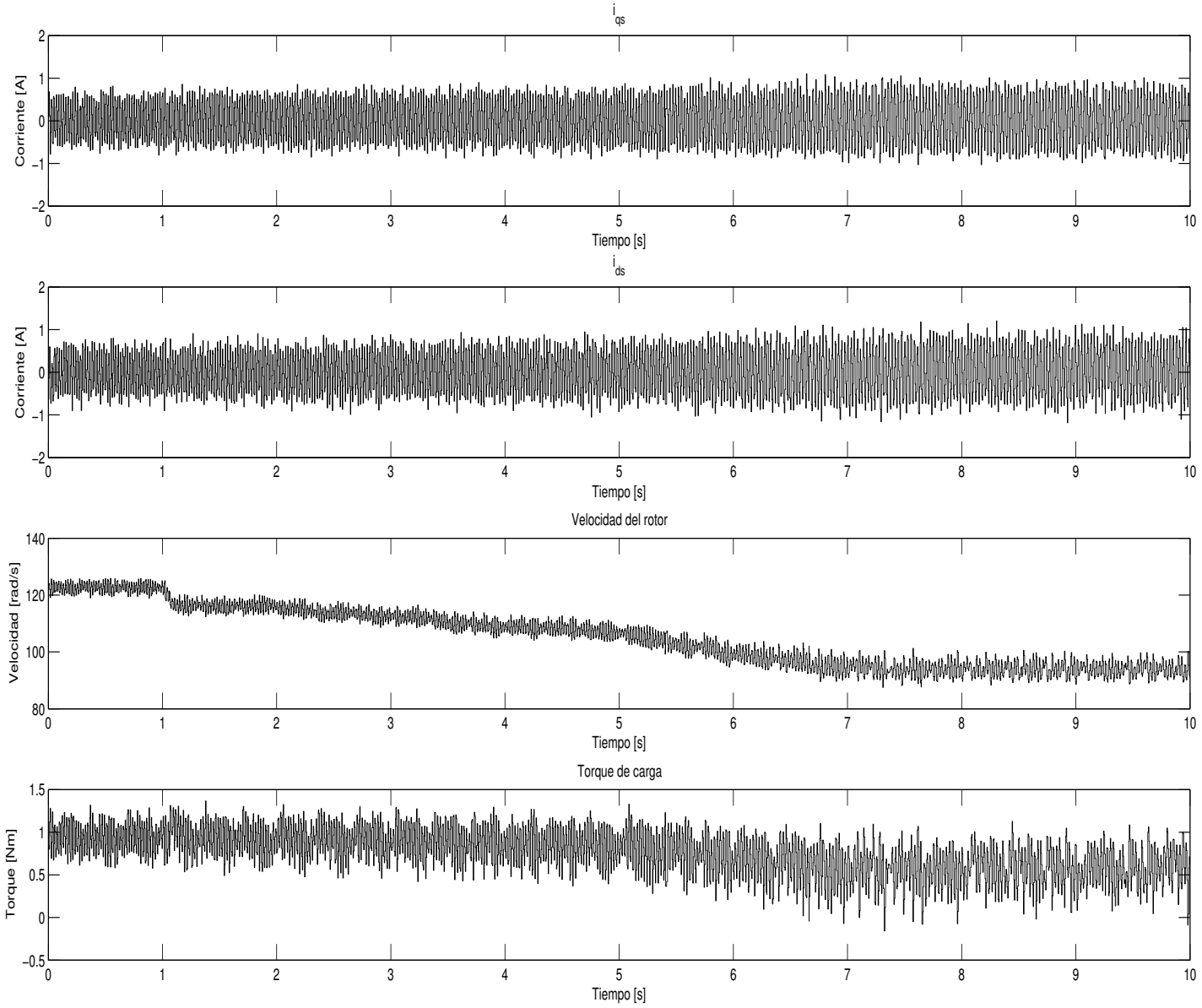


Figura 4.59: Estimación Kalman con cambio de 20Hz a 15 Hz (Corrientes, velocidad y torque)

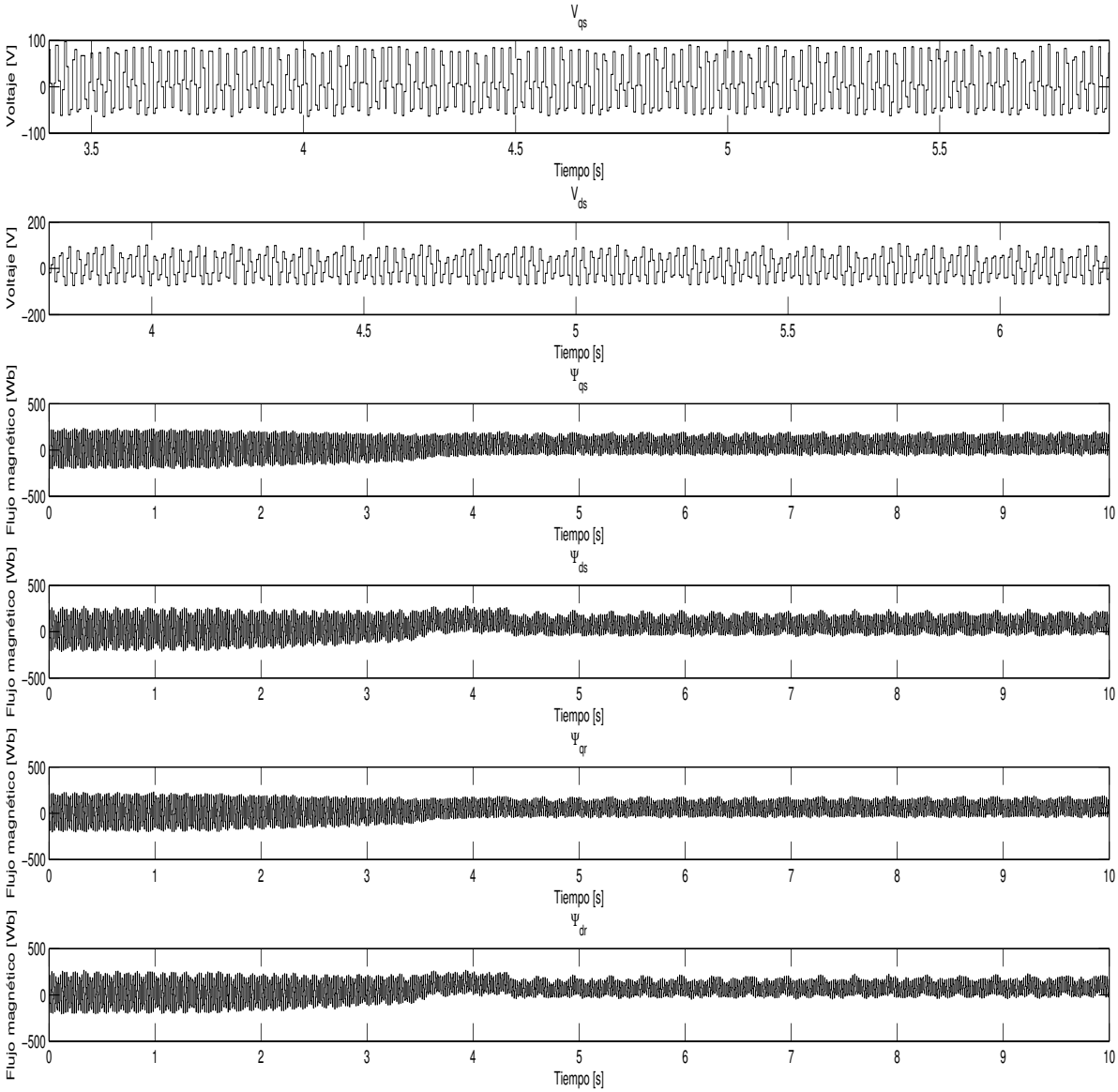


Figura 4.60: Estimación Kalman con frenado total (Voltaje y Flujos)

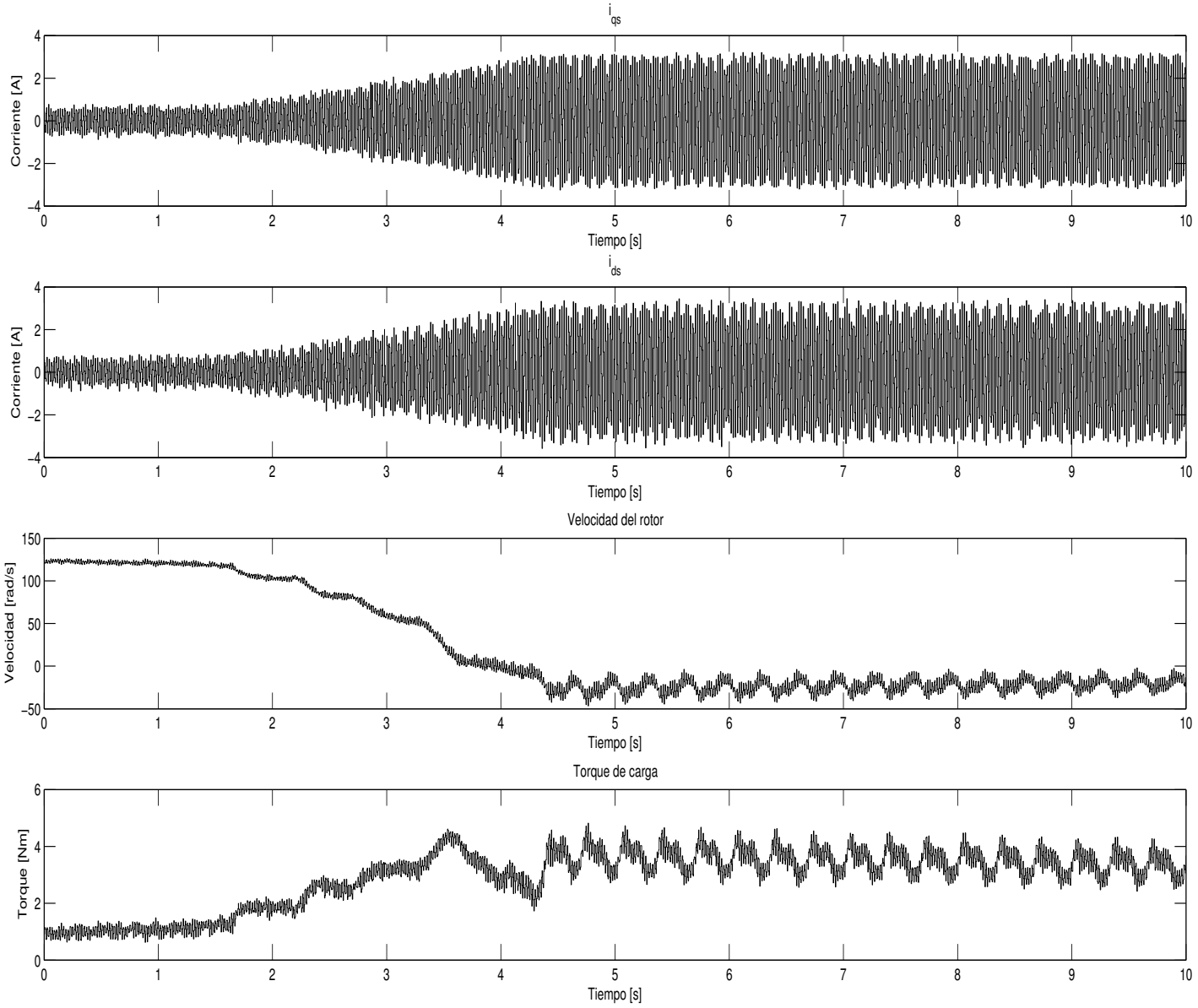


Figura 4.61: Estimación Kalman con frenado total (Corrientes, velocidad y torque)

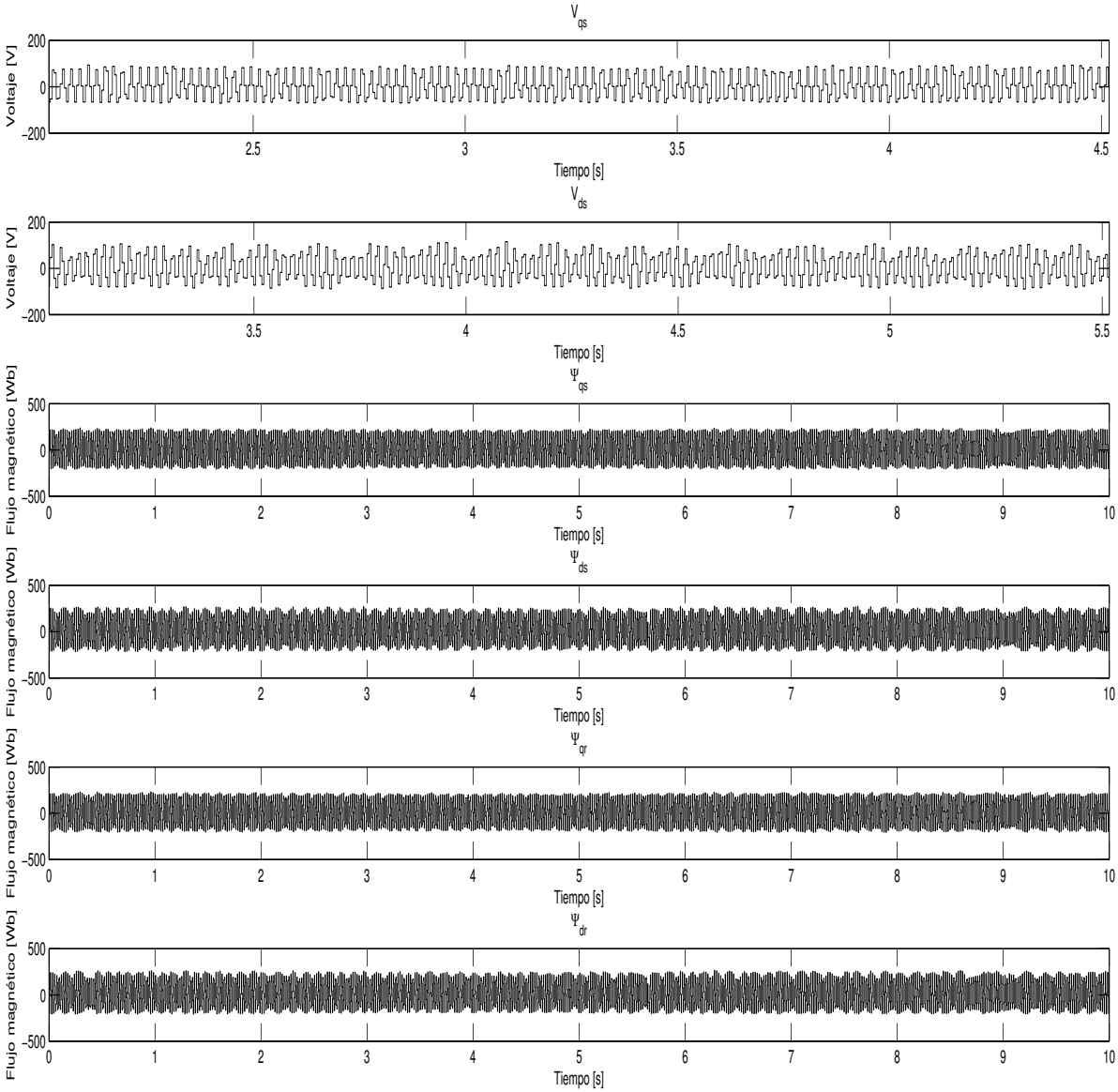


Figura 4.62: Estimación Kalman con cambios de carga (Voltajes y Flujos)

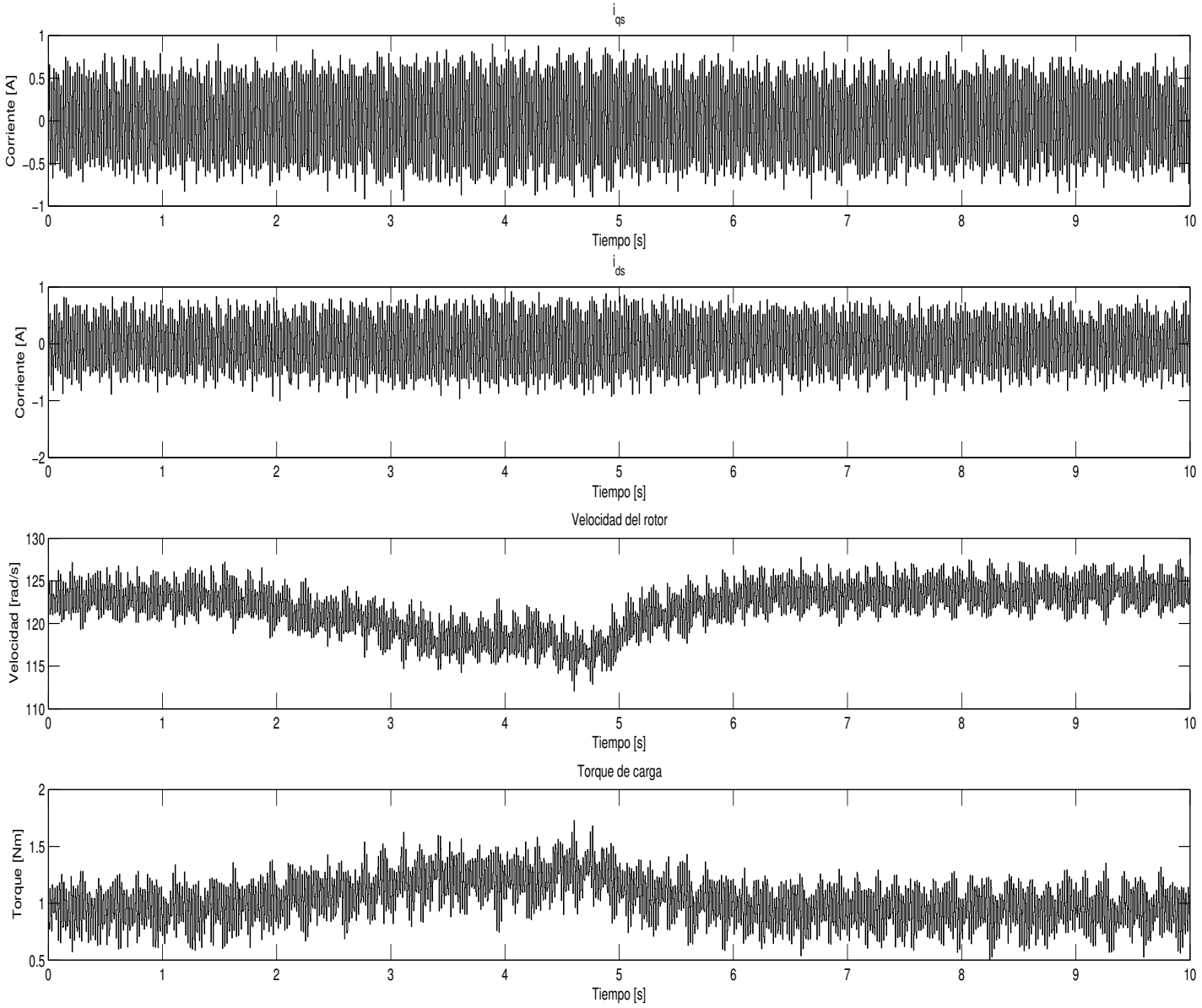


Figura 4.63: Estimación Kalman con cambios de carga (Corrientes, velocidad y torque)

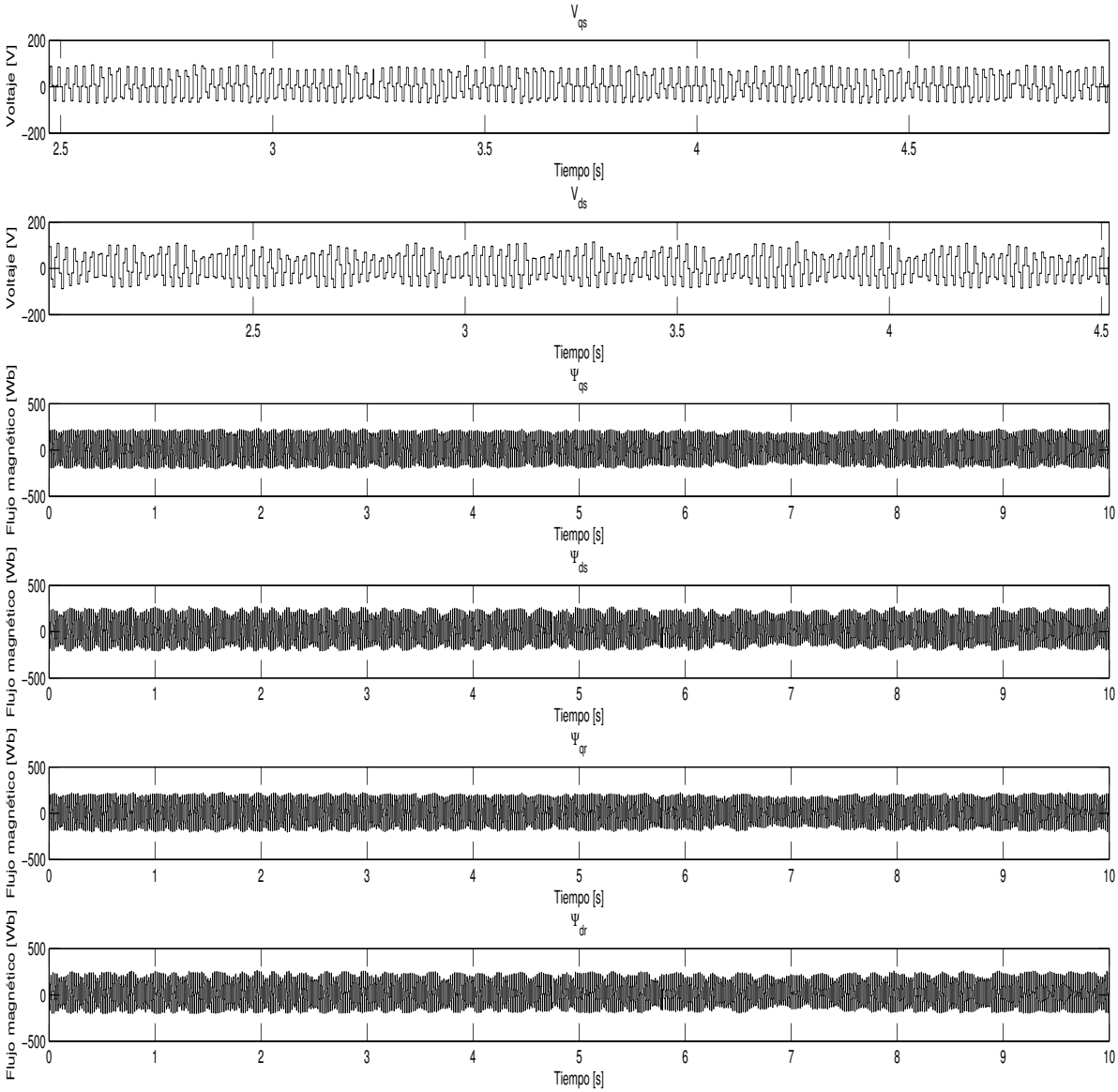


Figura 4.64: Estimación Kalman con cambios de carga múltiples (Voltajes y Flujos)

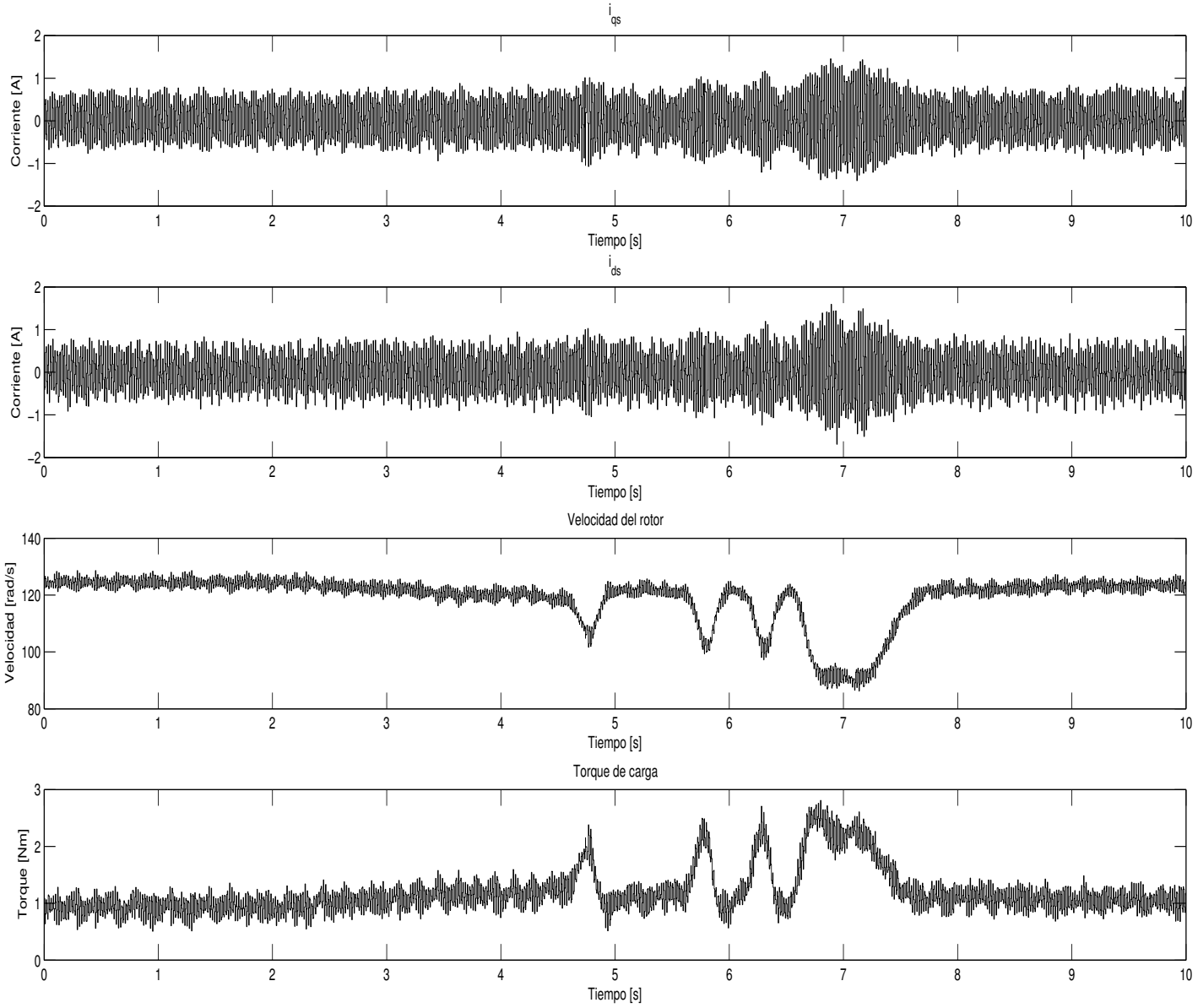


Figura 4.65: Estimación Kalman con cambios de carga múltiples (Corrientes, velocidad y torque)

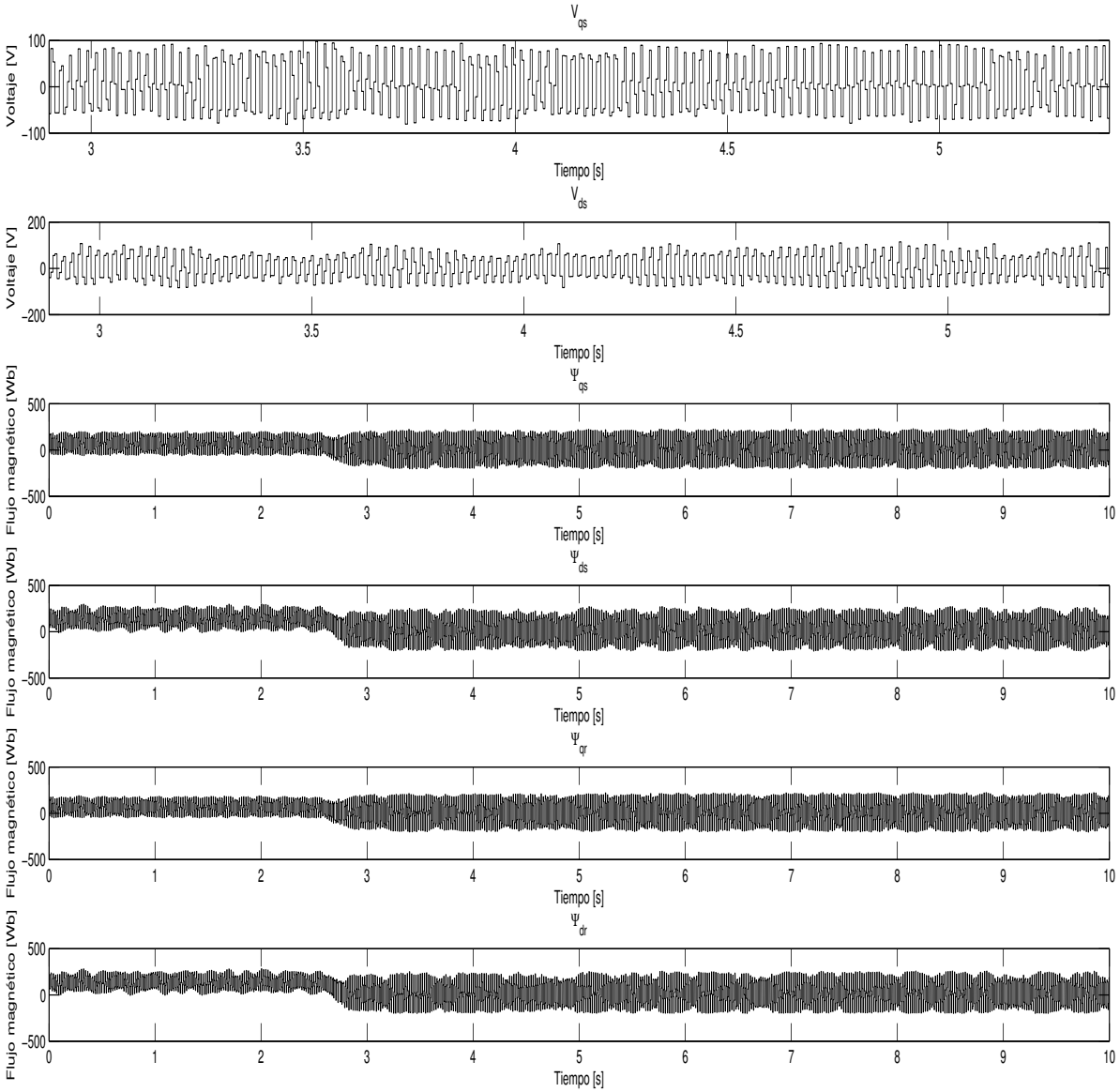


Figura 4.66: Estimación Kalman con carga total al arranque (Voltajes y Flujos)

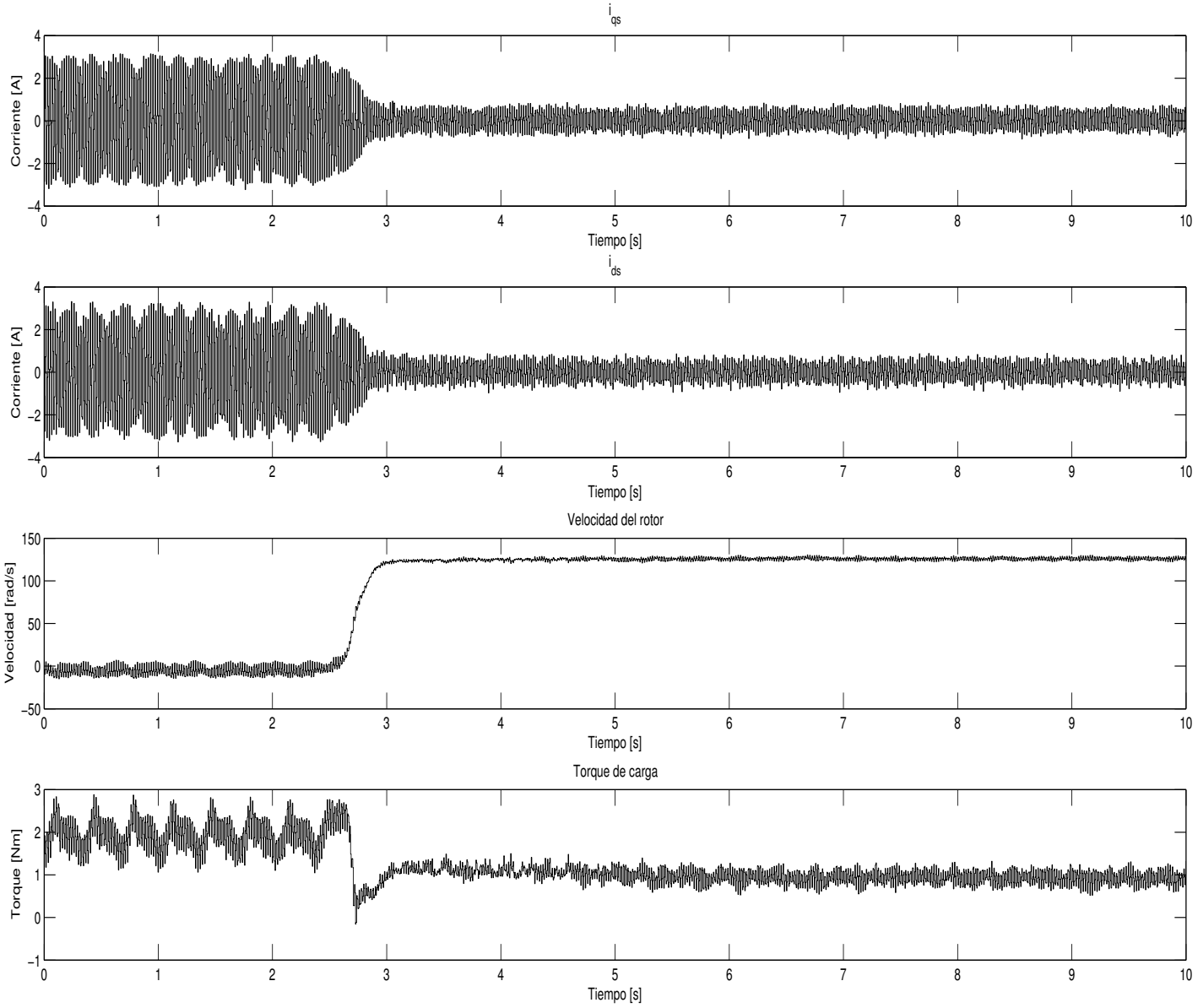


Figura 4.67: Estimación Kalman con carga total al arranque (Corrientes, velocidad y torque)

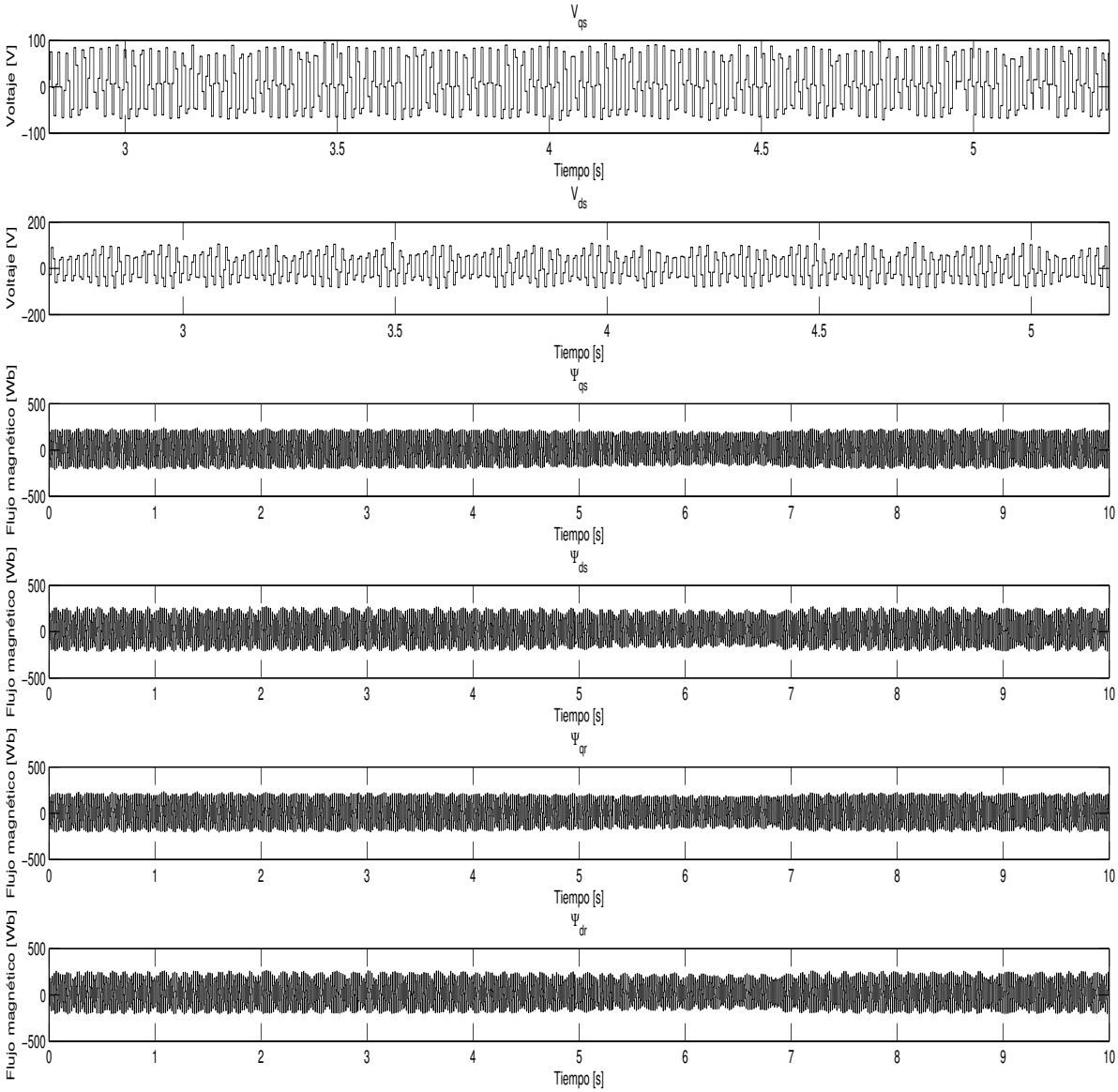


Figura 4.68: Estimación Kalman con carga y posterior liberación de carga (Voltajes y Flujos)

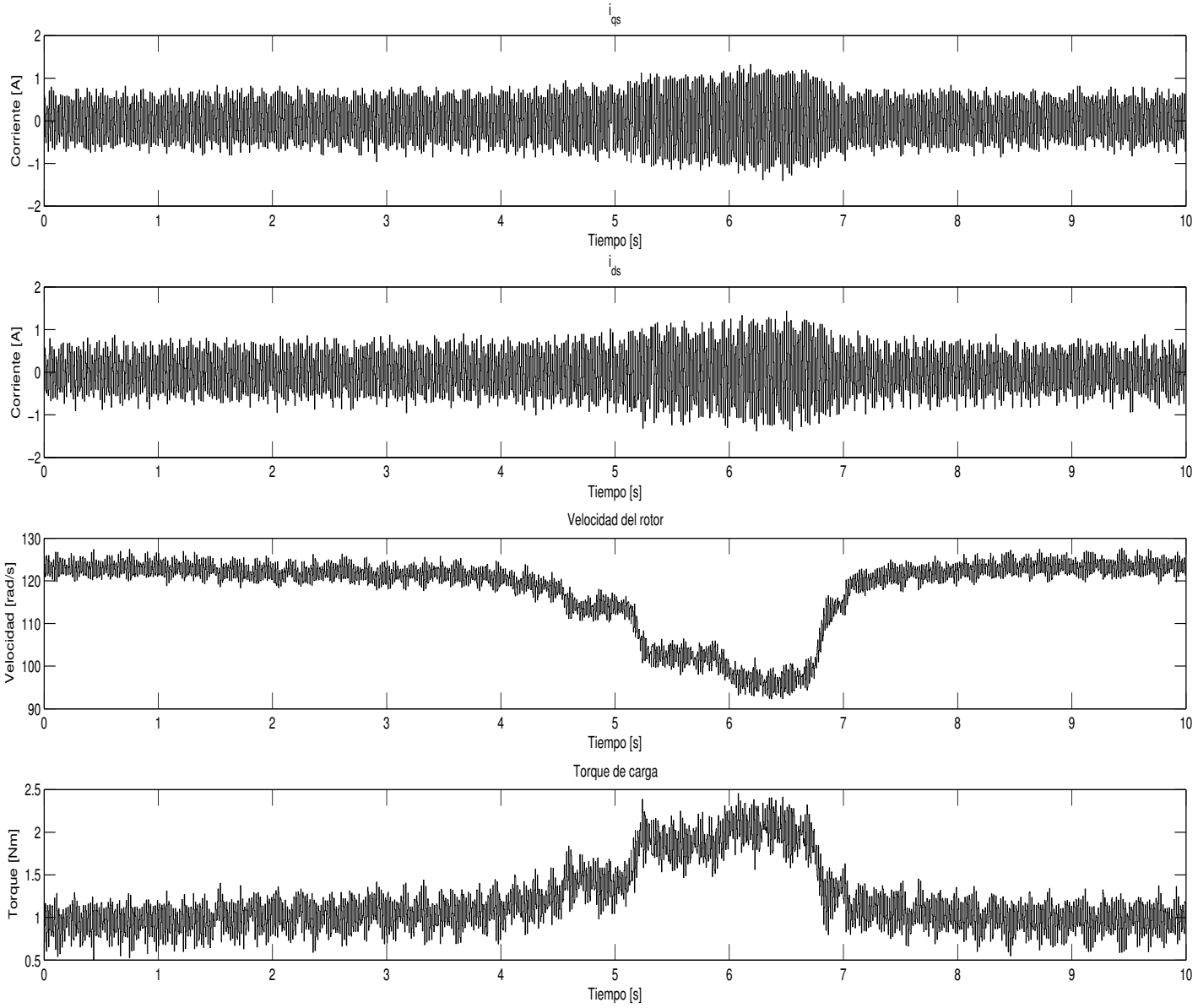


Figura 4.69: Estimación Kalman con carga y posterior liberación de carga (Corrientes, velocidad y torque)

En las Figuras 4.52 y 4.53 se aprecia el arranque del motor tanto en las gráficas de flujo como en las de velocidad. Es clara la etapa transitoria en las estimaciones de flujos seguida del estado estable en el mismo. En el caso de la velocidad del rotor se observa el arranque del motorsin oscilaciones considerables durante la estimación. El torque de carga presenta un

pico inicial seguido de una condición de operación estable alrededor de 1Nm. Al igual que en el caso de la operación a 30Hz, se presenta un desviación en el torque producto de que el estimador ve a la operación a baja frecuencia como una carga que frena al motor y no como una perturbación en baja frecuencia.

La respuesta en estado estable cuando el motor se alimenta con voltaje a 20Hz se observa en las Figuras 4.54 y 4.55. Se aprecia que la velocidad del motor oscila en aproximadamente 4rad/s alrededor del punto de operación estable. Esta oscilación sigue siendo tolerable. Es importante notar que en estado estable en las tres frecuencias de operación consideradas la amplitud de oscilación ha estado siempre en el orden de 4rad/s. En el caso del torque también se observa oscilaciones de aproximadamente 0.4Nm. Las oscilaciones ocurren alrededor de 1Nm que es el valor estimado por el filtro debido a la operación a 20Hz. El estimador no diferencia entre freno del motor por frecuencia de alimentación o por carga mecánica.

La respuesta a un cambio de frecuencia de 20Hz a 26Hz se observa en las Figuras 4.56 y 4.57. En este caso se ve como aumenta la velocidad del rotor al aumentar la frecuencia de alimentación. Si bien el estimador oscila, estas variaciones siguen sin ser demasiado significativas. En el torque de carga se observa que el valor estimado disminuye, ya que al acelerarse el motor por aumento en la frecuencia, el estimador considera que esto es producto de una reducción en la carga.

La respuesta a un cambio de frecuencia de 20Hz a 15Hz se observa en las Figuras 4.58 y 4.59. La velocidad del rotor disminuye de forma progresiva hasta llegar a un valor estable final. La oscilación está dentro de los rangos tolerables durante los transitorios. En tanto, la estimación del torque de carga presenta un comportamiento anómalo ya que disminuye. Se esperaría según lo discutido antes que aumente, ya que el disminuir la frecuencia de alimentación el filtro reaccionaba como si se aumentase carga. Este es el primer problema serio que se observa en la estimación a baja frecuencia.

Las Figuras 4.60 y 4.61 muestran la estimación cuando la carga es tal que frena completamente al motor. En el caso de la velocidad se observa que el filtro estima velocidades negativas de aproximadamente -30rad/s por un largo tiempo y con oscilaciones sostenidas que no parecen converger a 0rad/s. De igual manera el torque presenta oscilaciones grandes sostenidas alrededor de 3.5Nm. No se aprecia una tendencia que indique que este escenario va a cambiar y que la variable estimada se estabilice. El filtro presenta problemas para estimar la velocidad y el torque en el caso de baja frecuencia de operación.

Las Figuras 4.62 y 4.63 muestran la respuesta del filtro a un cambio de carga sencillo. Inicialmente se aplica carga que hace que el motor se frene como se puede observa en la respuesta del filtro. Posteriormente, al liberar carga el motor nuevamente se acelera hasta alcanzar el valor final. Las oscilaciones sostenidas durante este periodo son de aproximadamente 5rad/s. En el caso del torque la reacción es consistente con los resultados de velocidad y se observa oscilaciones sostenidas alrededor de 1Nm.

En las Figuras 4.64 y 4.65 se aprecia la respuesta del filtro a cambios múltiples de carga. Esta prueba se hace con el objetivo de probar la respuesta del filtro cuando ocurren múltiples perturbaciones en poco tiempo. El filtro responde bien, ya que se pueden apreciar los picos durante las perturbaciones. Estos picos indican la buena respuesta dinámica del sistema tanto en la estimación de velocidad rotórica como en el torque de carga. Las oscilaciones sostenidas son tolerables.

El arranque del motor con el 100 % de su carga se aprecia en las Figuras 4.66 y 4.67. Se ve que el filtro tiene problemas para estimar la velocidad cero al arranque. En el caso del filtro Kalman a baja frecuencia se debe resaltar este comportamiento como una primera respuesta limitada del estimador. Se puede apreciar las oscilaciones iniciales con el rotor bloqueado. Es un caso similar con el torque de carga que presenta fuertes oscilaciones en la estimación. Una vez liberado el rotor el filtro regresa a su modo de trabajo normal y la estimación recupera las características descritas anteriormente con una estimación bastante limpia de las variables de interés.

Las Figuras 4.68 y 4.69 muestran la respuesta del filtro cuando se aplican procesos de carga y liberación de carga sostenidos. En este caso se aplica una carga pequeña, se aumenta la carga y finalmente se libera completamente al motor pero sin cambios bruscos. Como se espera, se tienen cambios suaves en las variables estimadas, pero con el respectivo proceso oscilatorio durante las transientes principalmente.

4.3.4. Filtro H_∞ extendido-Alta frecuencia (40-60 Hz)

La matriz G_k y el parámetro γ utilizados en la implementación del filtro H_∞ fueron:

$$G_k = \begin{bmatrix} 4 & 4 & 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 & 4 & 4 \end{bmatrix}$$

$$\gamma = 0{,}4$$

Se realizaron las pruebas detalladas en la introducción de esta sección con una frecuencia de alimentación de 50Hz. A continuación se muestran los resultados. Como ya se mencionó se muestran los voltajes y corrientes $dq0$ del estator (medidos vía muestreo en el DSP) y las variables estimadas: flujos $dq0$ de estator y rotor, velocidad del rotor y torque de carga.

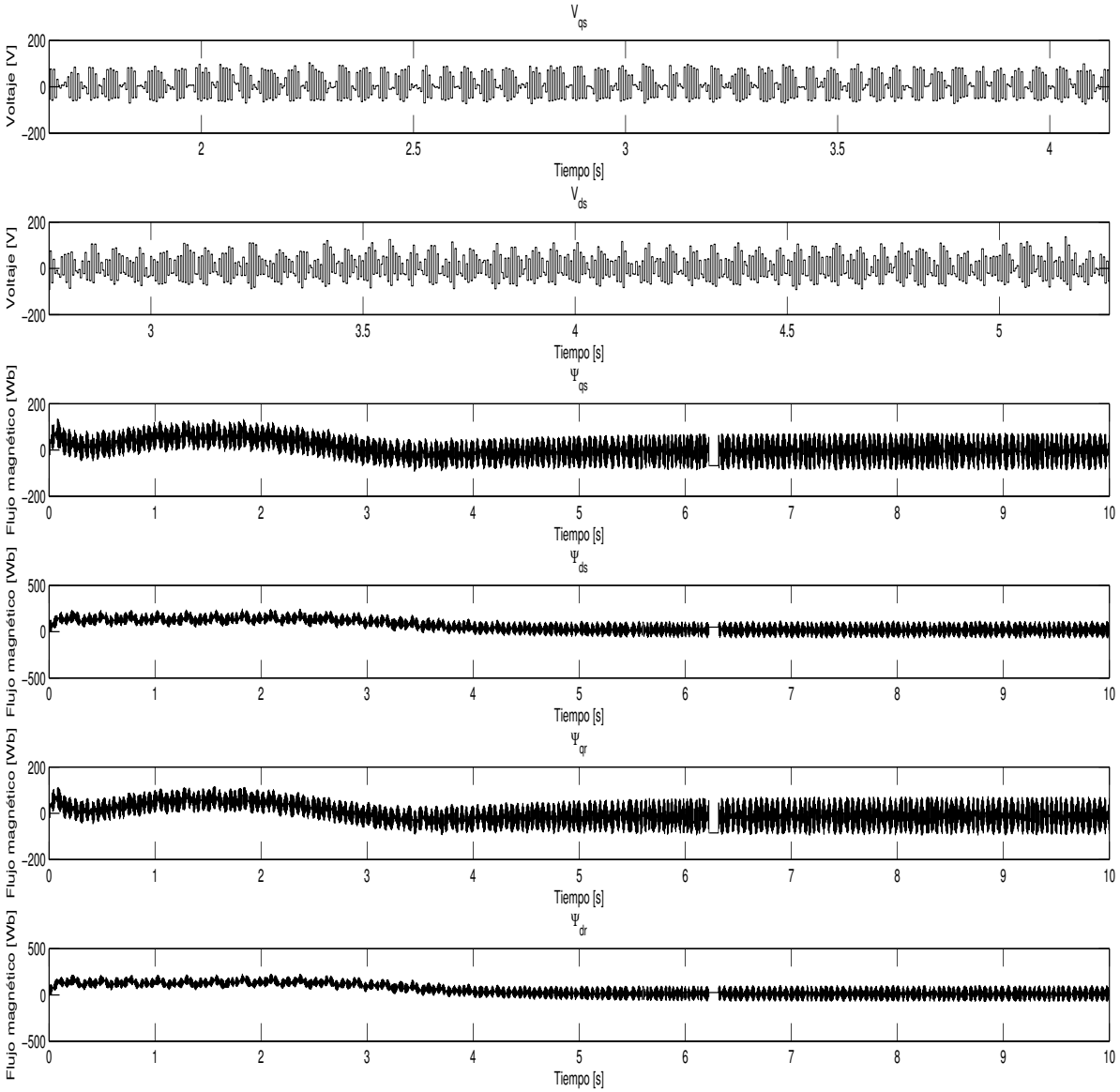


Figura 4.70: Estimación H_∞ a 50Hz en el arranque sin carga (Voltajes y Flujos)

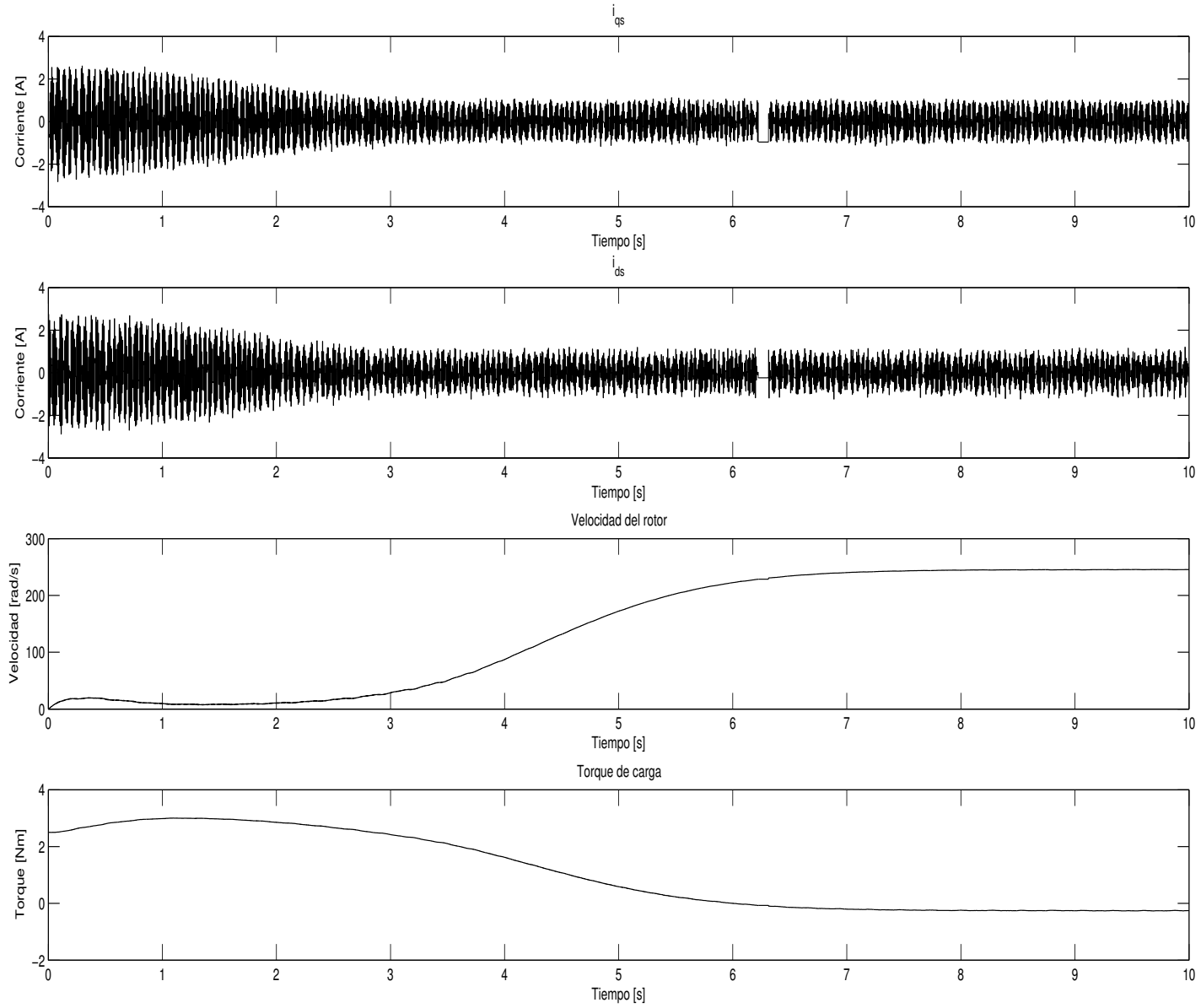


Figura 4.71: Estimación H_∞ a 50Hz en el arranque sin carga (Corrientes, velocidad y torque)

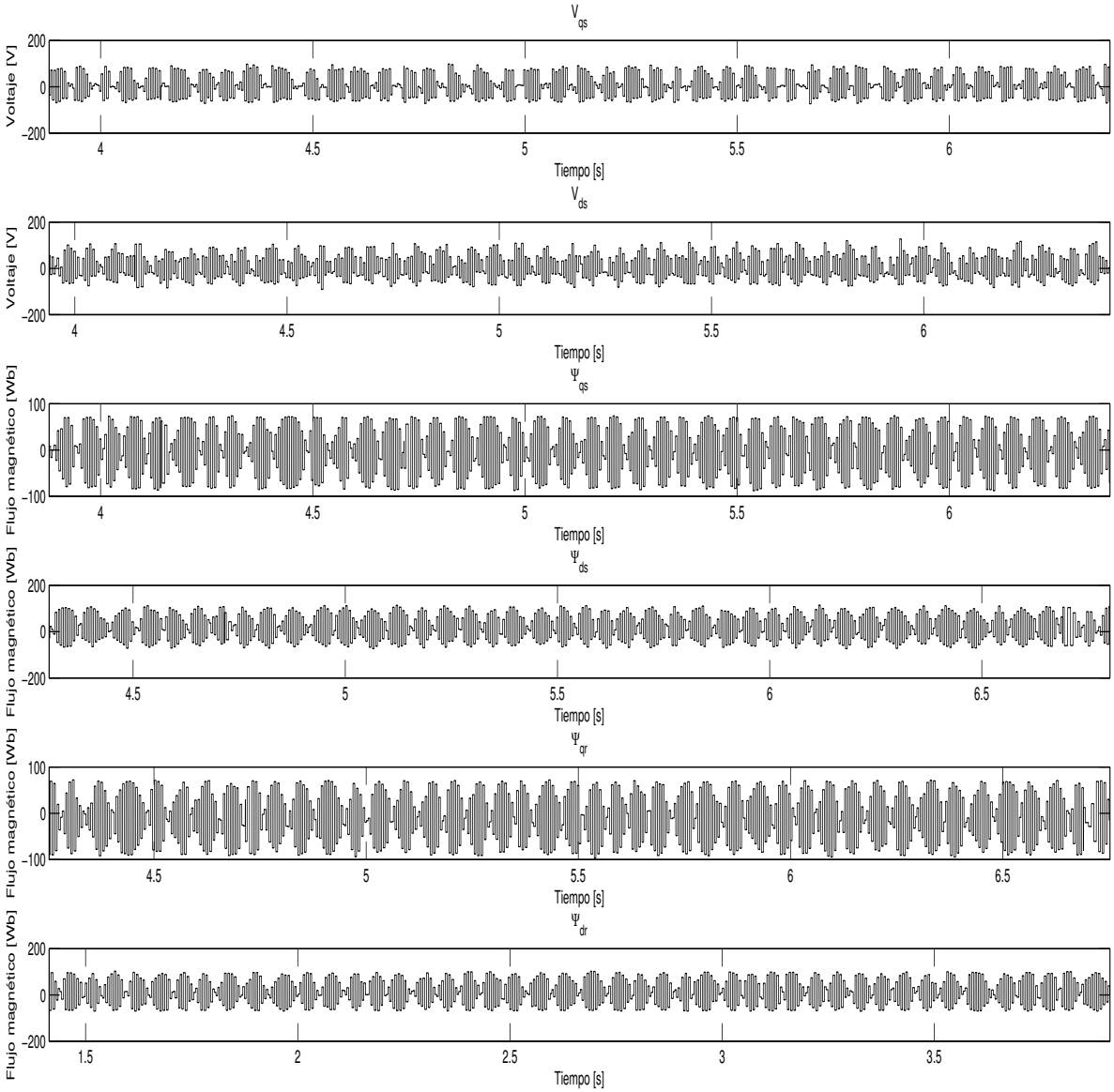


Figura 4.72: Estimación H_∞ a 50Hz en estado estable sin carga (Voltajes y Flujos)

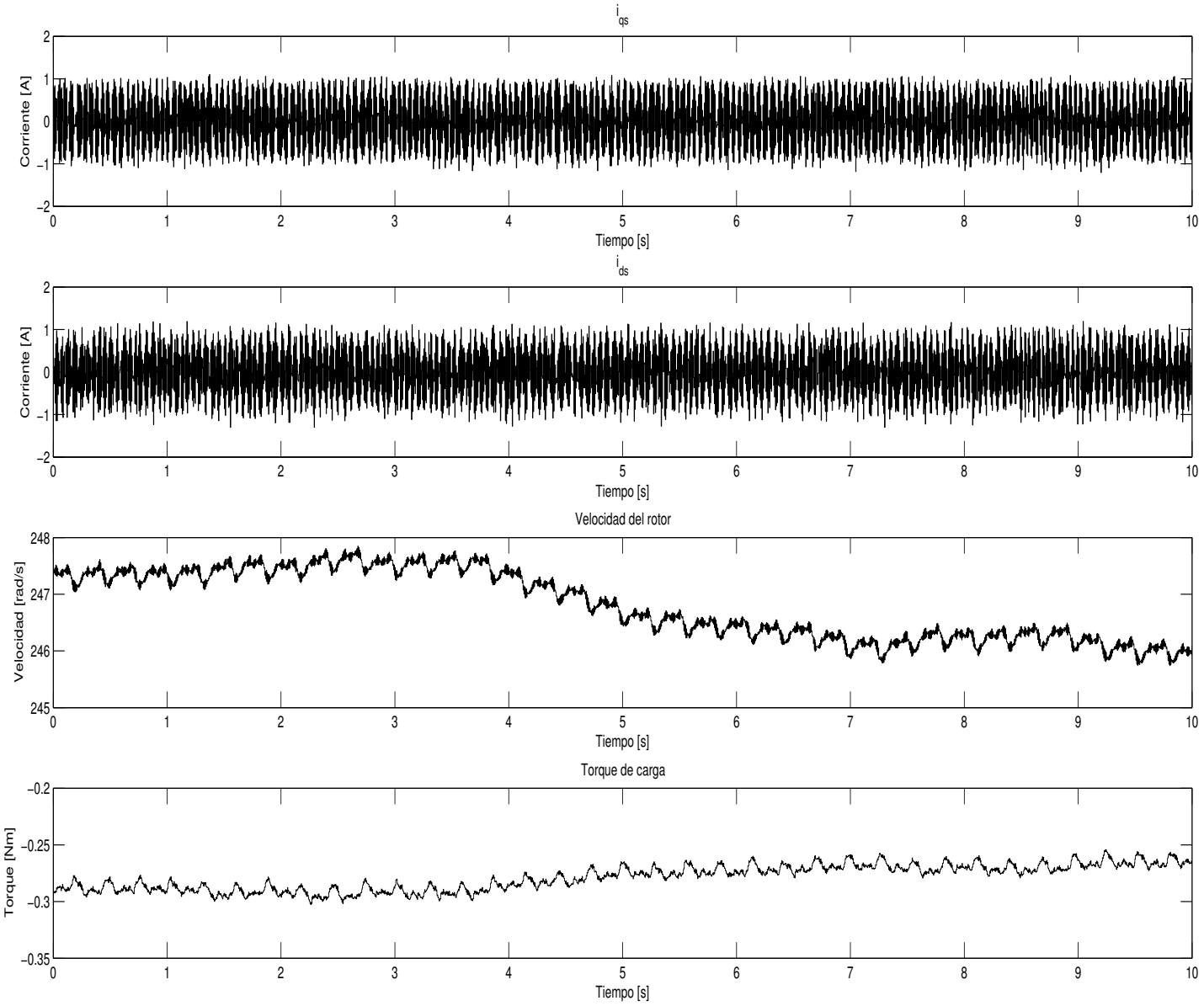


Figura 4.73: Estimación H_∞ a 50Hz en estado estable sin carga(Corrientes, velocidad y torque)

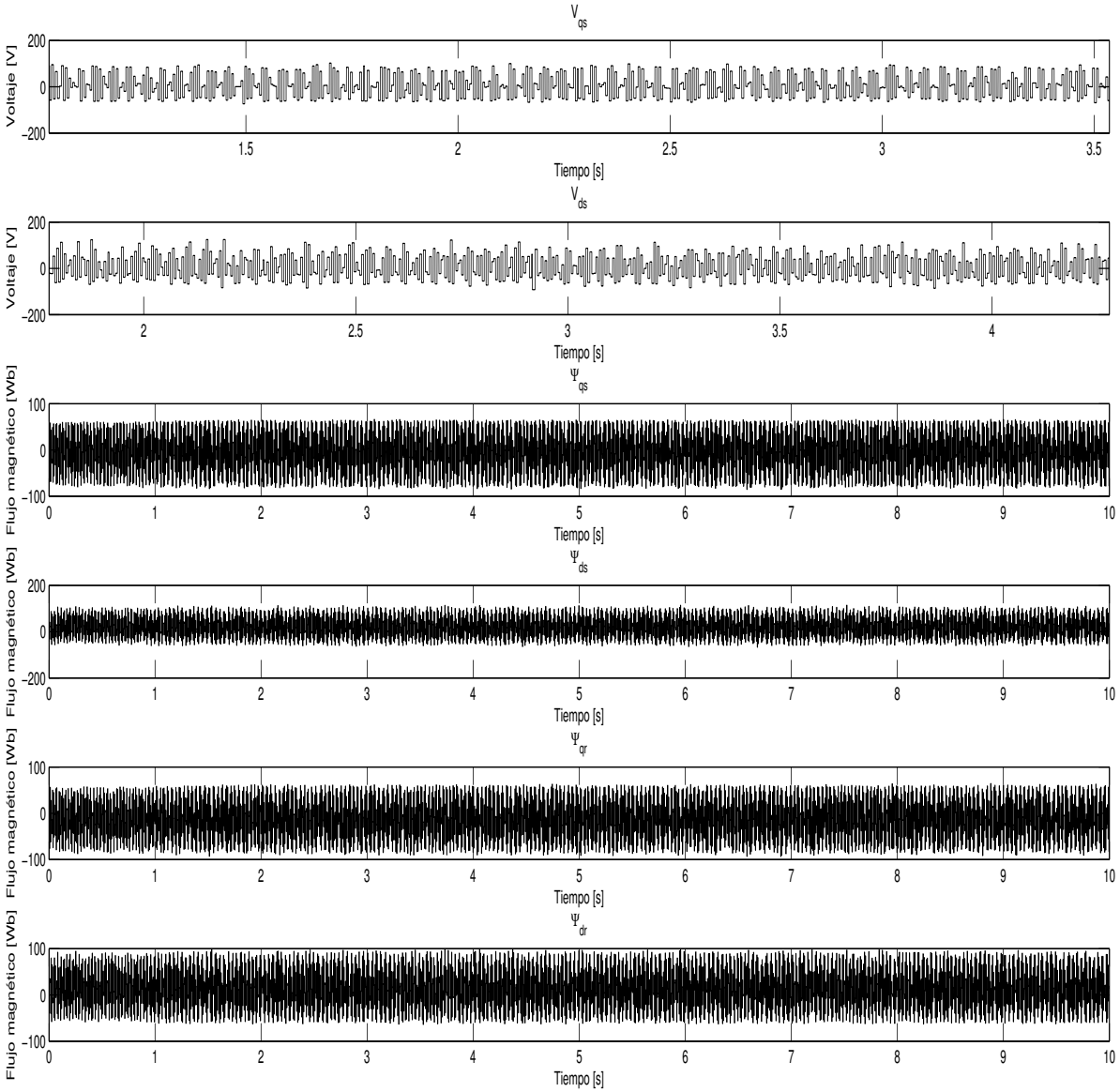


Figura 4.74: Estimación H_∞ con cambio de 48Hz a 52Hz (Voltajes y Flujos)

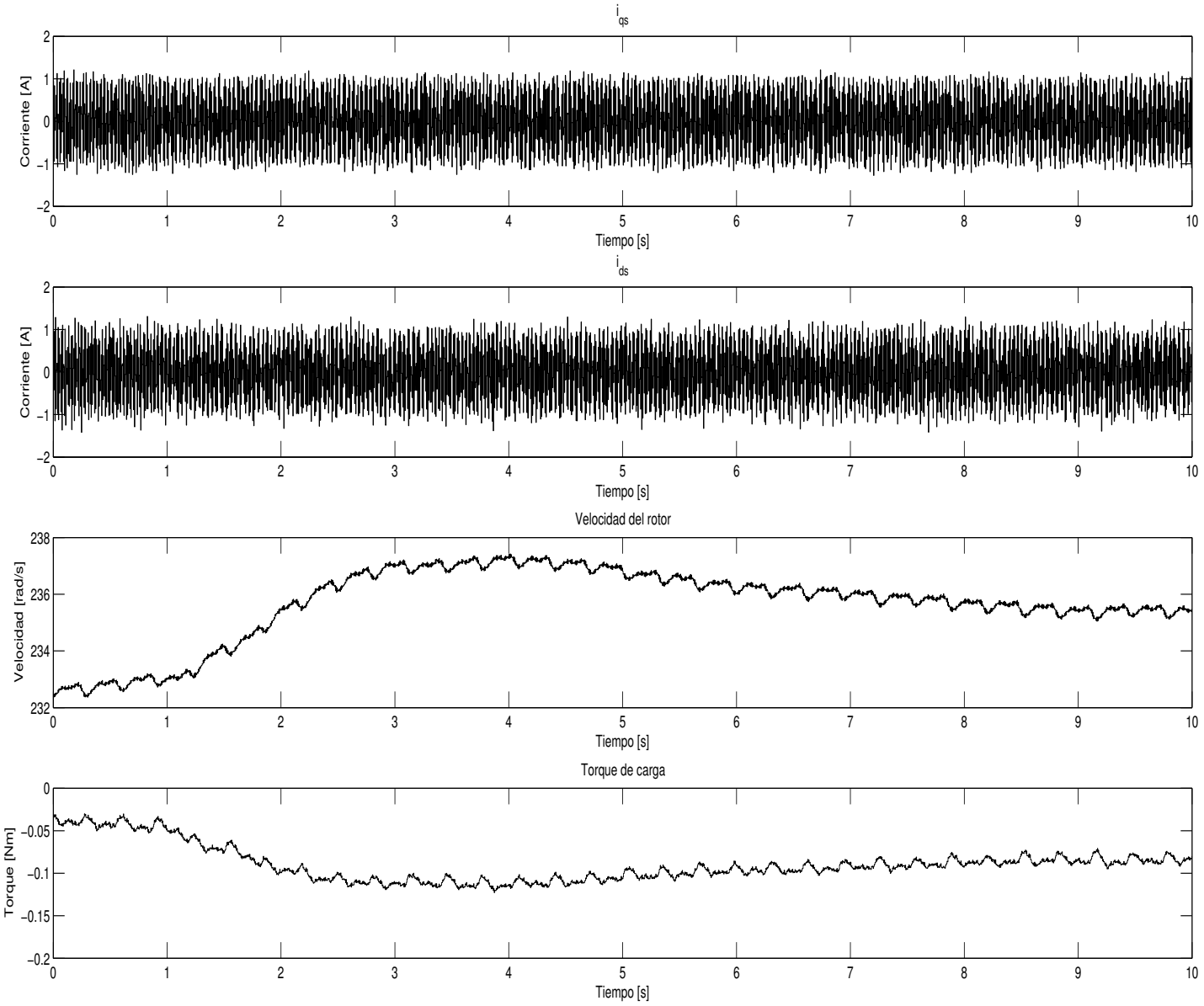


Figura 4.75: Estimación H_∞ con cambio de 48Hz a 52Hz (Corrientes, velocidad y torque)

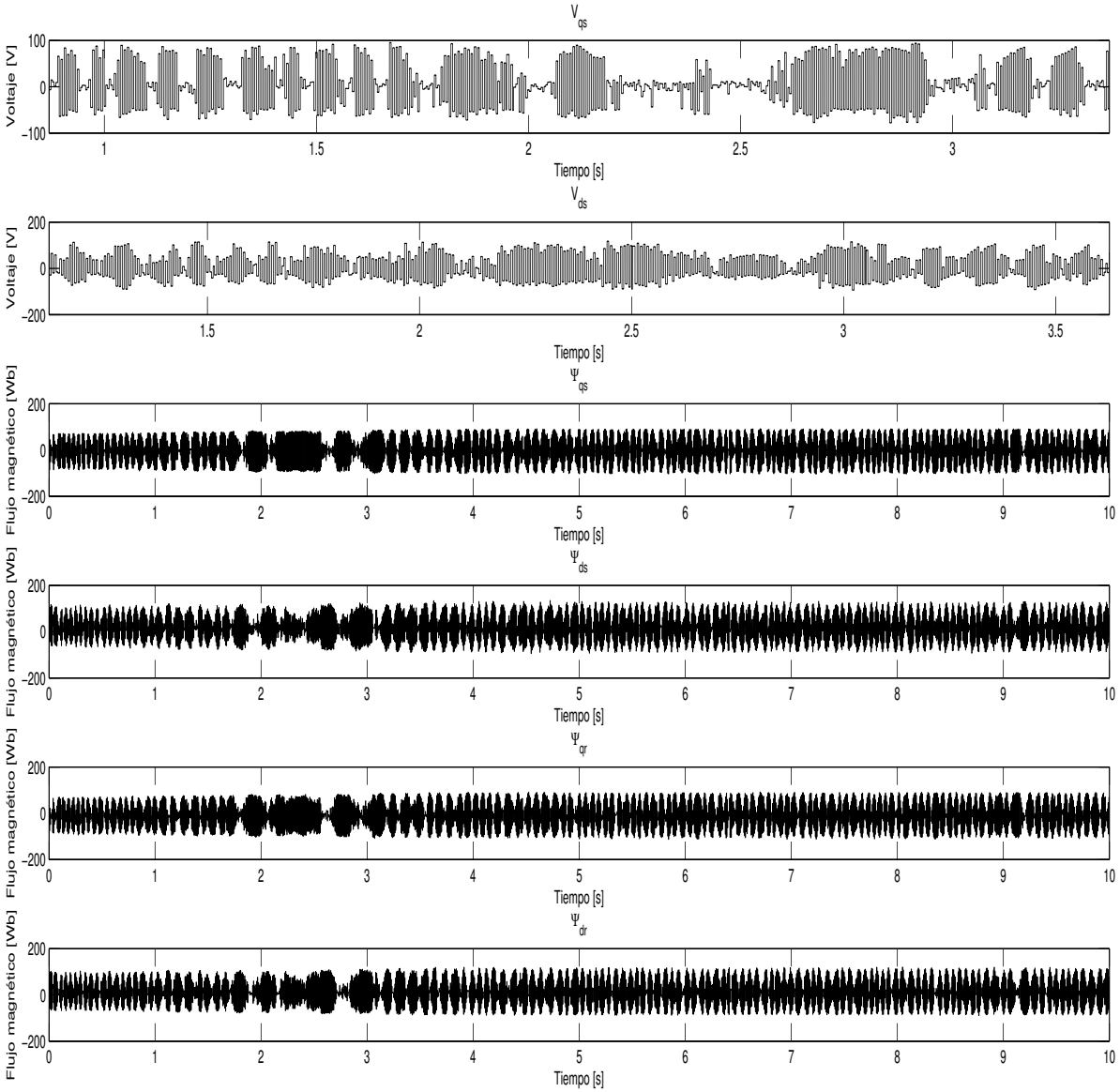


Figura 4.76: Estimación H_∞ con cambio de 50Hz a 42Hz (Voltajes y Flujos)

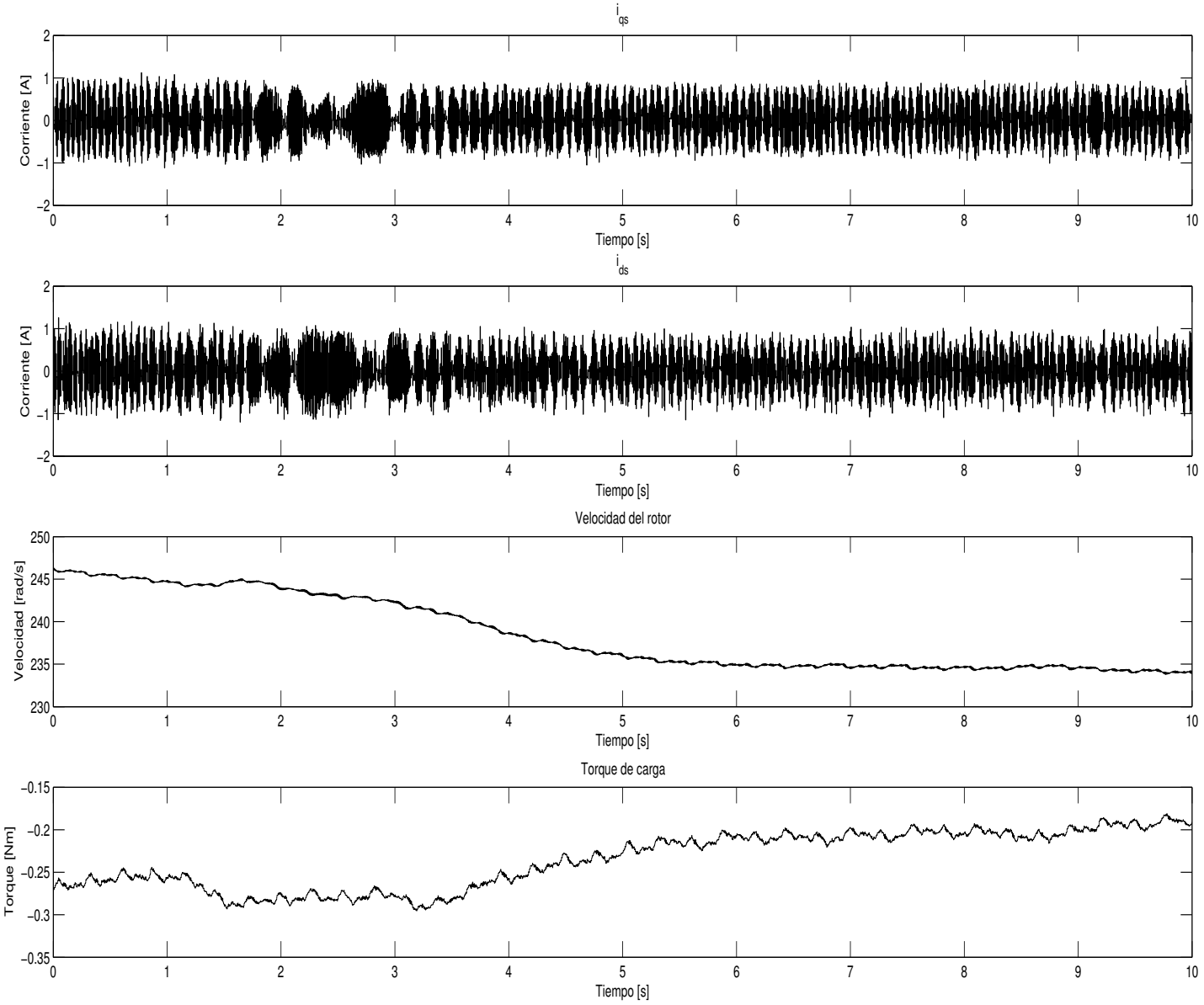


Figura 4.77: Estimación H_∞ con cambio de 50Hz a 42Hz (Corrientes, velocidad y torque)

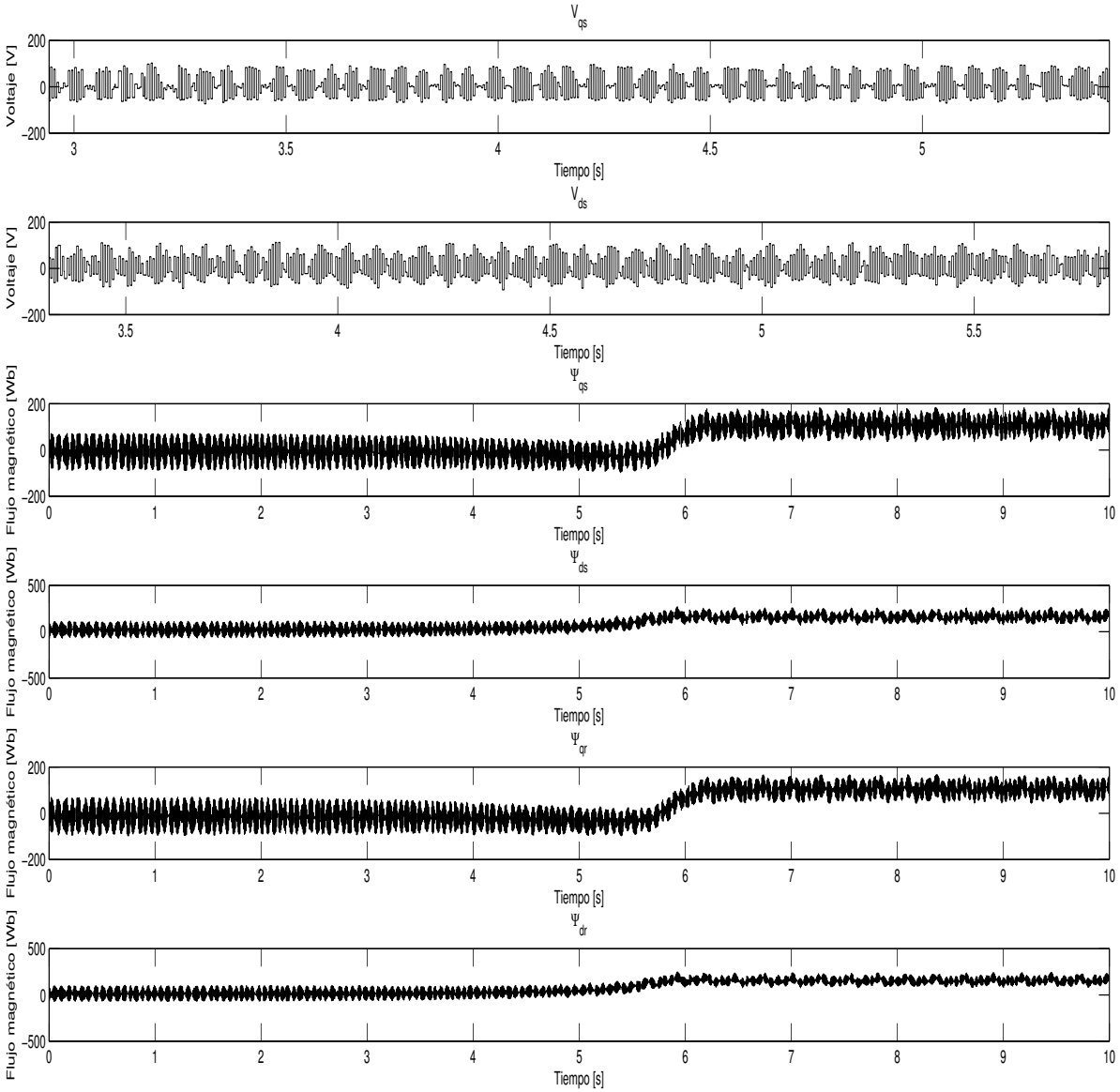


Figura 4.78: Estimación H_∞ con frenado total (Voltajes y Flujos)

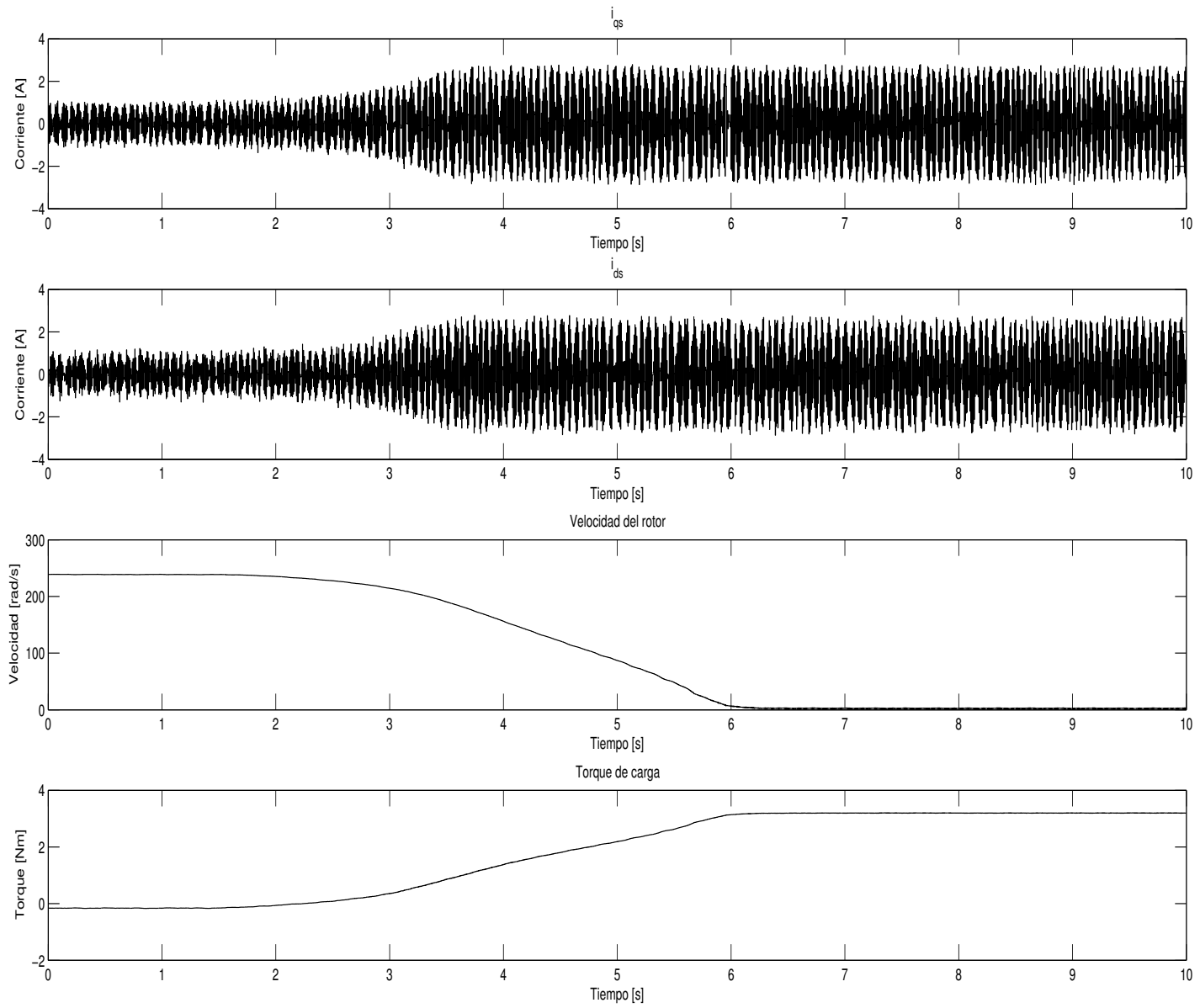


Figura 4.79: Estimación H_∞ con frenado total (Corrientes, velocidad y torque)

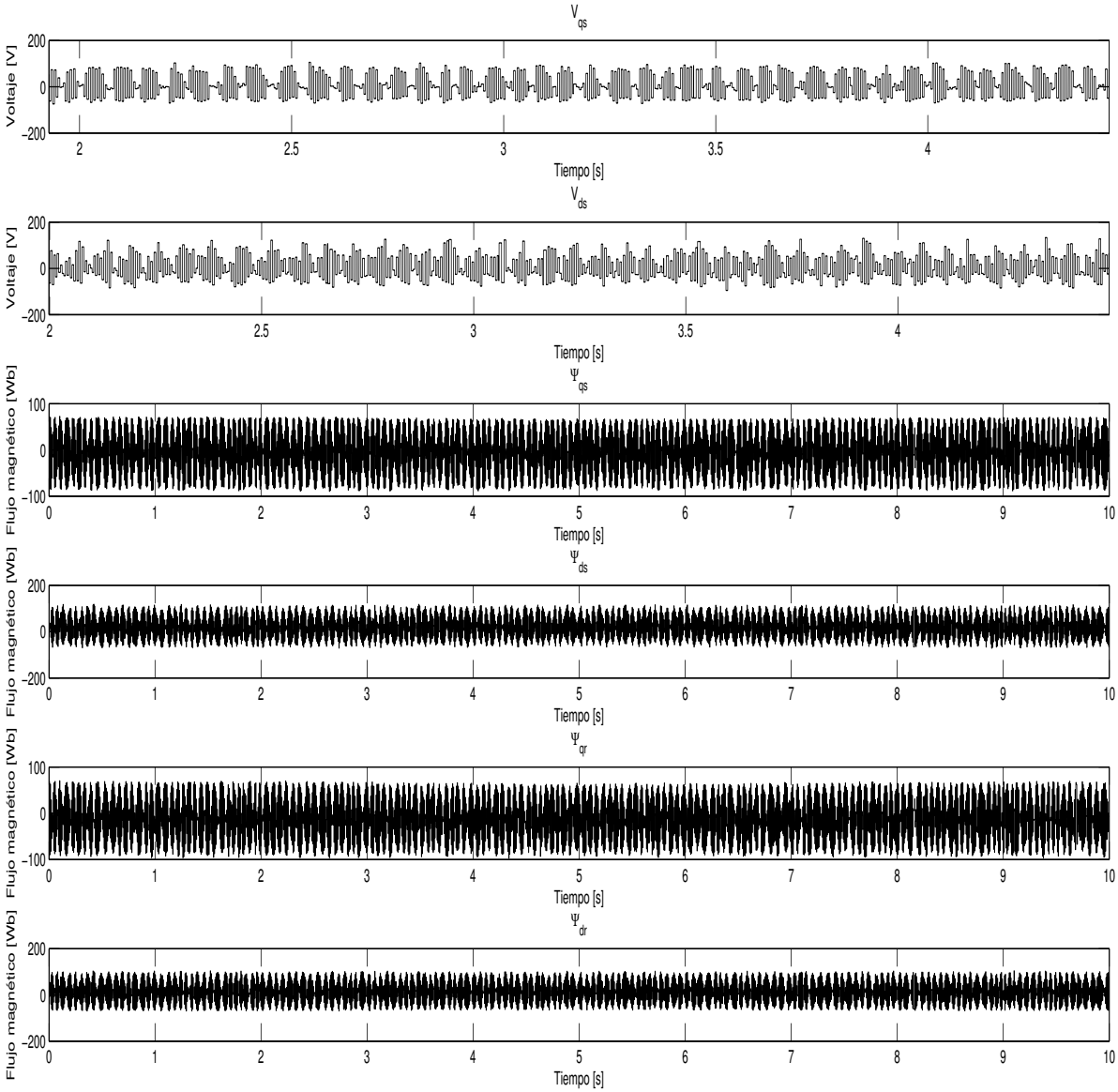


Figura 4.80: Estimación H_∞ con cambios de carga (Voltajes y Flujos)

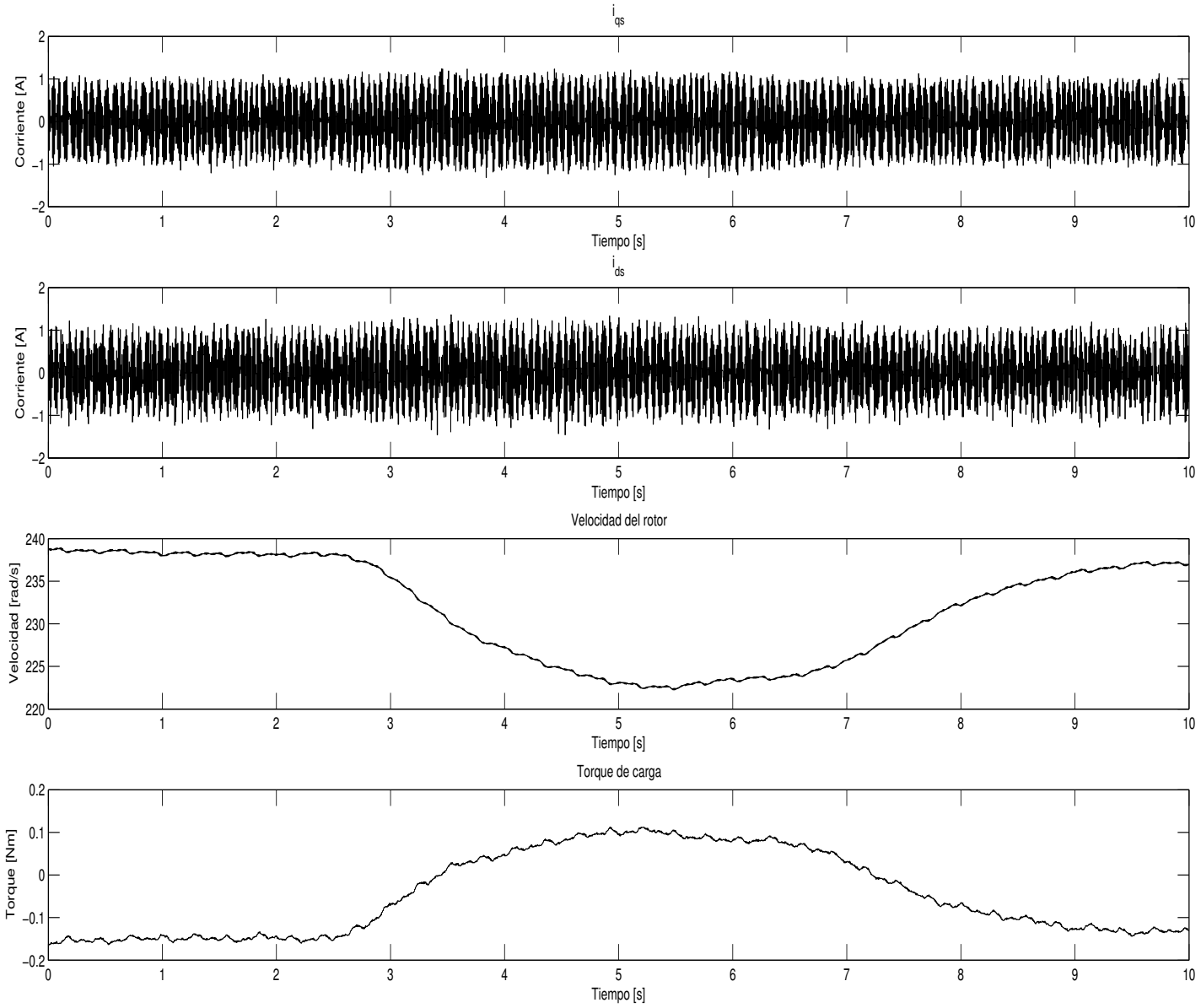


Figura 4.81: Estimación H_∞ con cambios de carga (Corrientes, velocidad y torque)

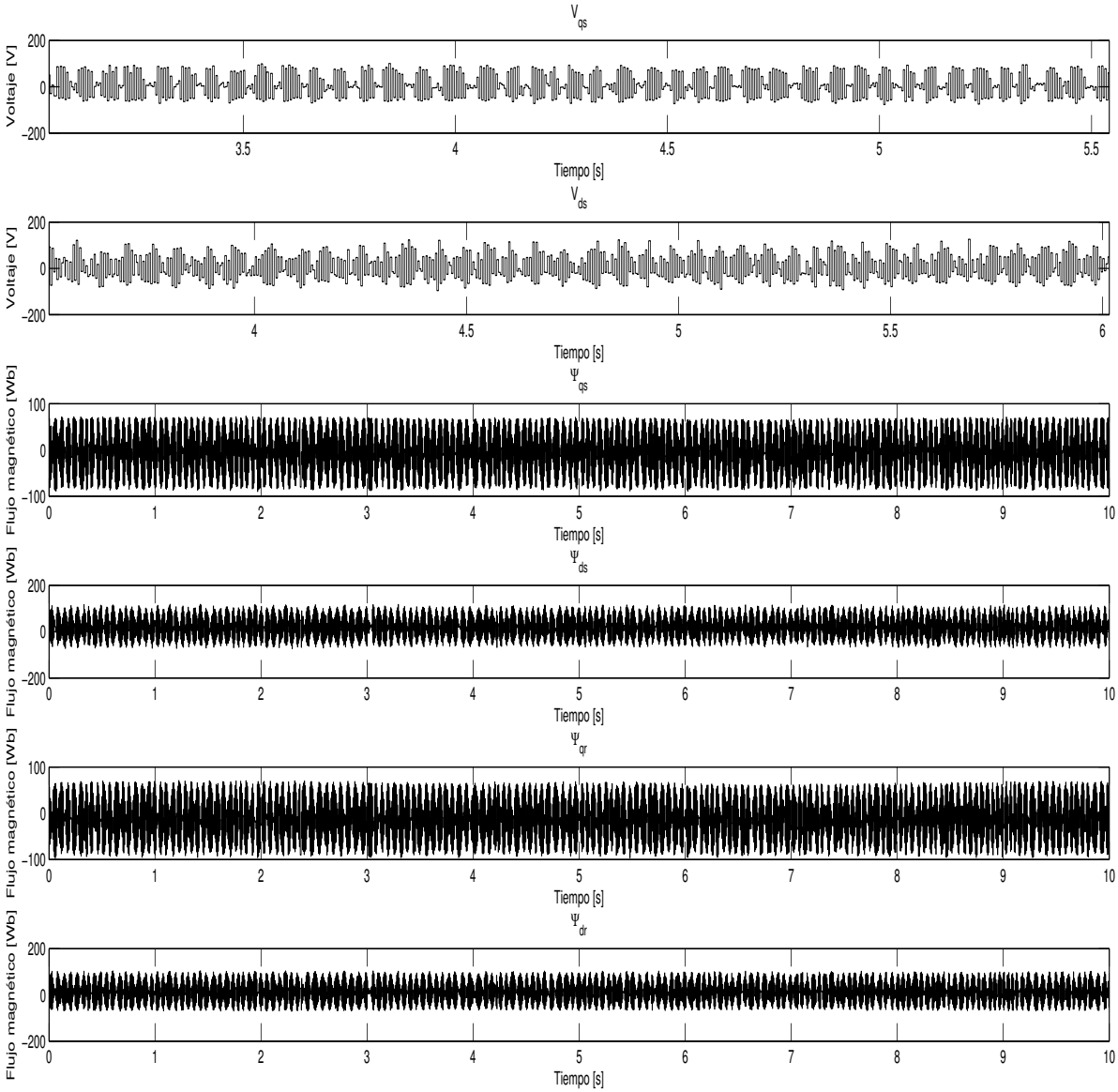


Figura 4.82: Estimación H_∞ con cambios de carga múltiples (Voltajes y Flujos)

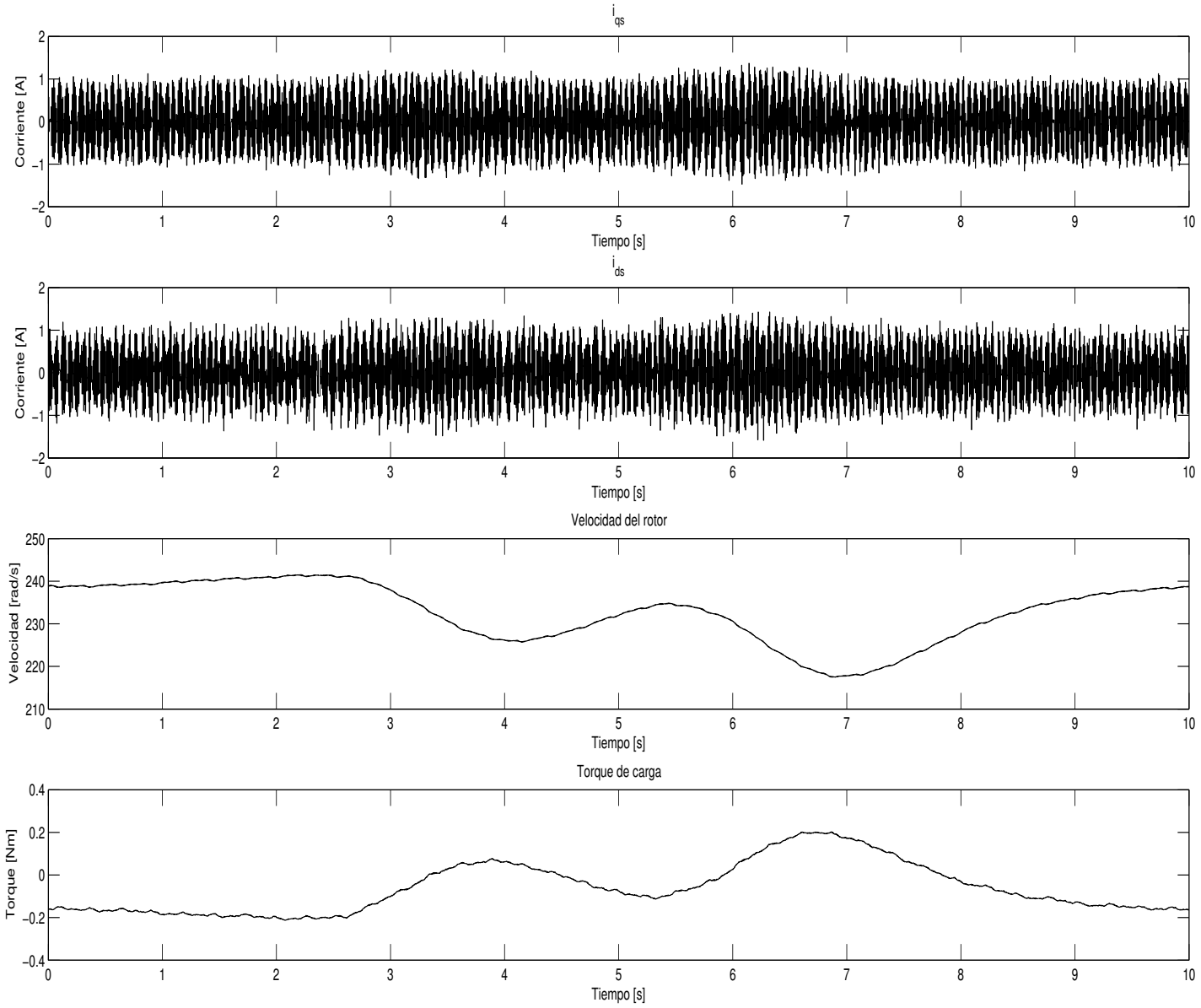


Figura 4.83: Estimación H_∞ con cambios de carga múltiples (Corrientes, velocidad y torque)

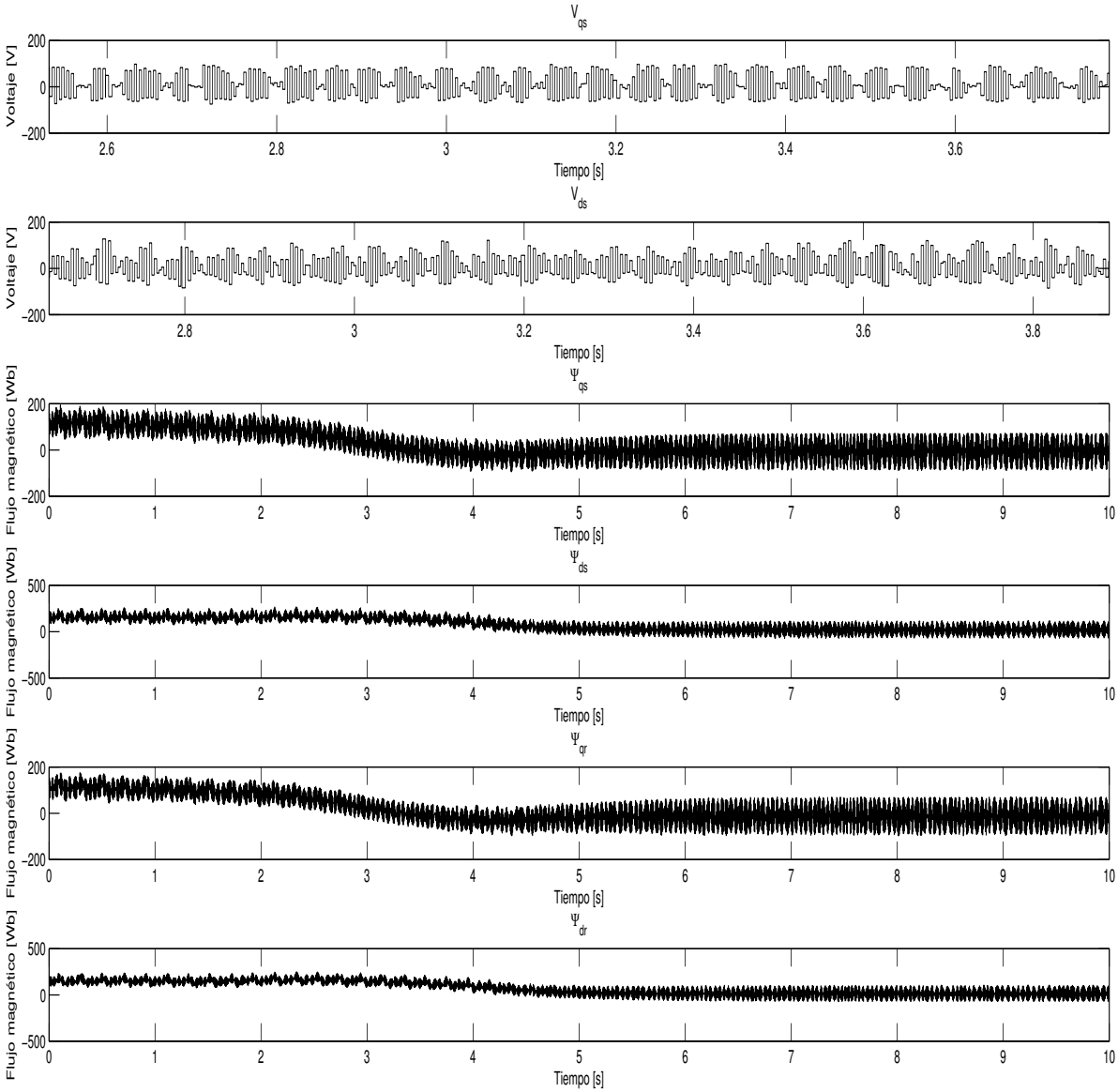


Figura 4.84: Estimación H_∞ con carga total al arranque (Voltajes y Flujos)

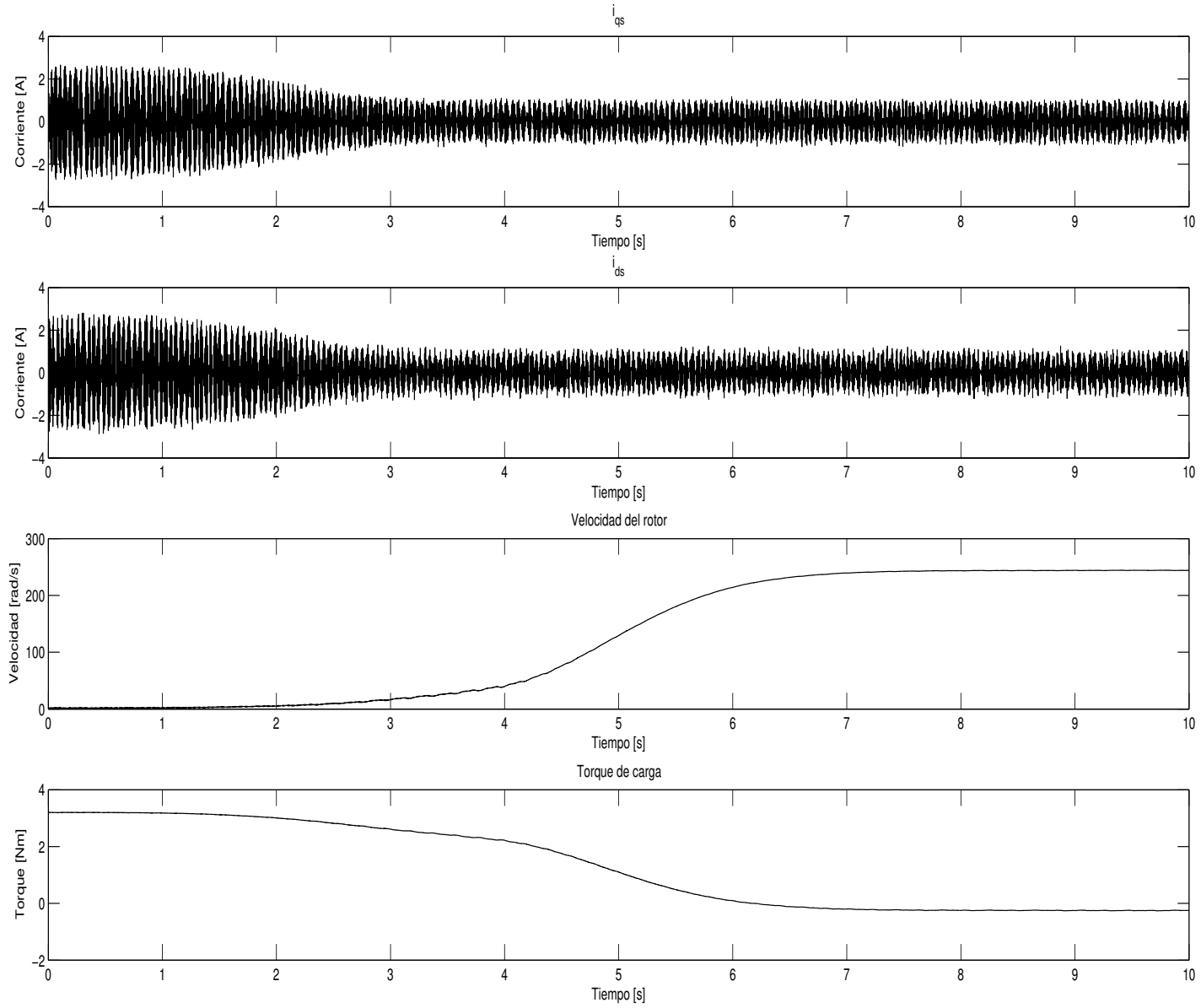


Figura 4.85: Estimación H_∞ con carga total al arranque (Corrientes, velocidad y torque)

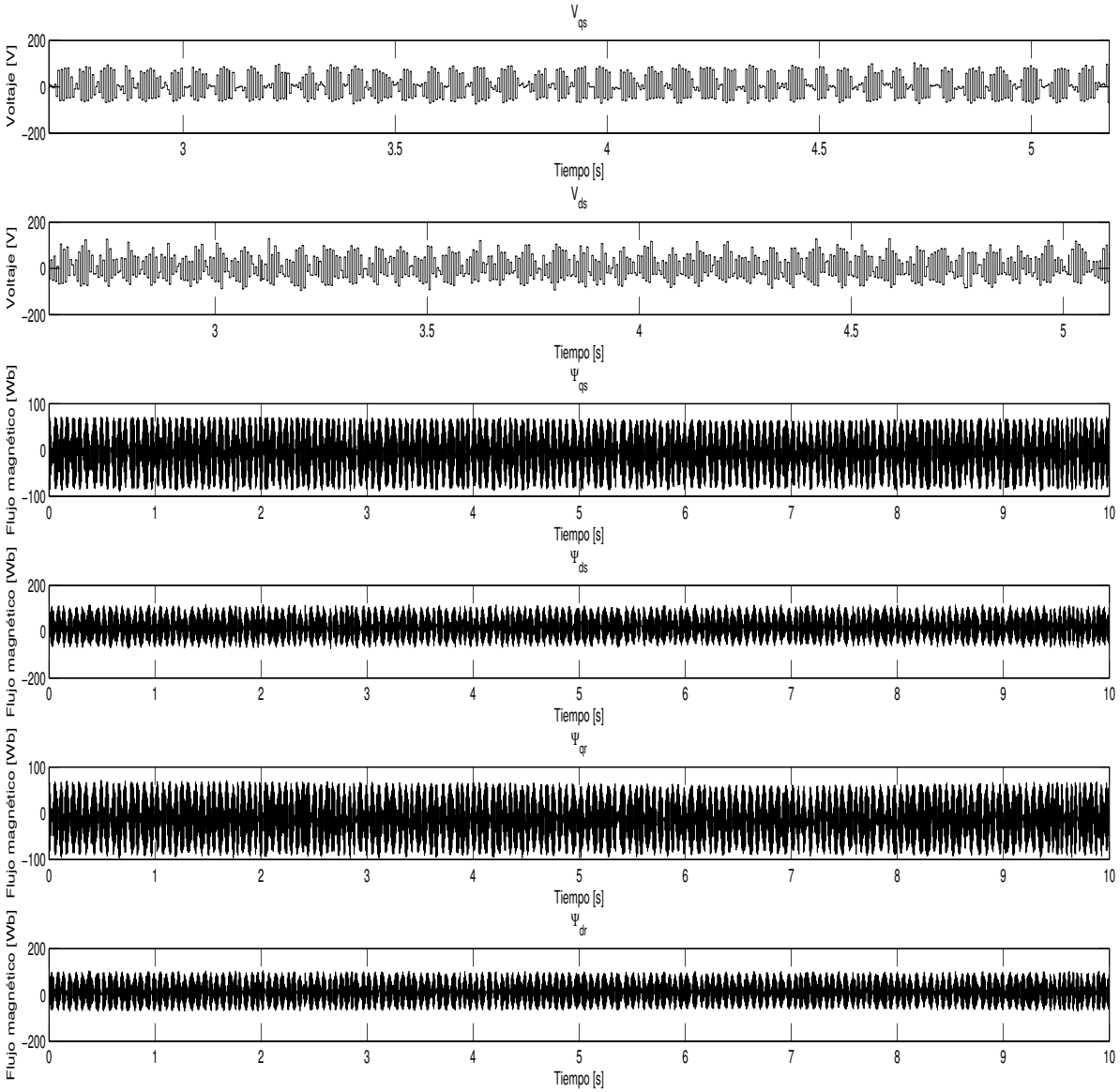


Figura 4.86: Estimación H_∞ con carga y posterior liberación de carga (Voltajes y Flujos)

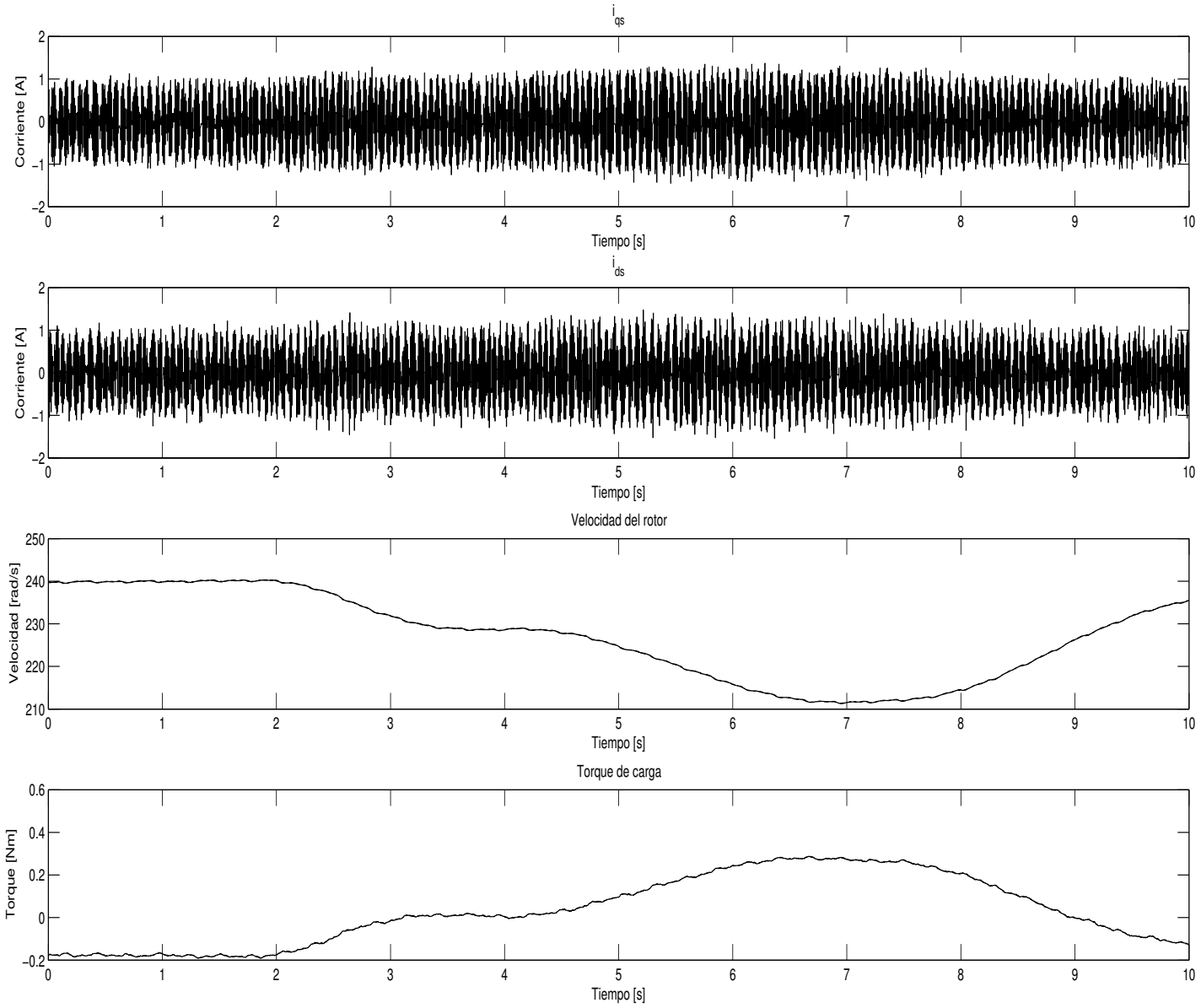


Figura 4.87: Estimación H_∞ con carga y posterior liberación de carga (Corrientes, velocidad y torque)

En las Figuras 4.72 y 4.71 se observa la estimación del filtro para una frecuencia de alimentación de voltaje de 50Hz. Se puede apreciar que la estimación de velocidad es muy limpia, sin oscilaciones apreciables. No obstante, la mejora en la calidad de la estimación tiene una contraparte en la baja velocidad de reacción del filtro. Se puede apreciar que la respuesta es

más lenta que la que se tenía para el filtro Kalman. En el caso del torque de carga se observa la misma característica, una estimación de la variable muy limpia pero lenta en su dinámica. No se observa un pico corto y pronunciado antes de la convergencia a 0Nm sino más bien una curva suave.

La estimación en estado estable con alimentación de voltaje a 50Hz se observa en las Figuras 4.72 y 4.73. La oscilación en velocidad es mínima, de menos de 0.5 rad/s en las oscilaciones más pequeñas, pero se observan oscilaciones más grandes de aproximadamente 2rad/s. Estas oscilaciones son mucho más pequeñas que las registradas en el filtro Kalman. De igual manera la estimación de torque de carga tiene oscilaciones más pequeñas, de aproximadamente 0.05Nm. Sin embargo, se observa un valor de desviación en la variable estimada.

En las Figuras 4.76 y 4.75 se aprecia la respuesta del filtro cuando existe una perturbación de cambio de frecuencia de operación de 48Hz a 52Hz. En este caso la velocidad del rotor aumenta, sin embargo, se puede apreciar que la respuesta del filtro es lenta, ya que hay un largo y poco pronunciado sobreimpulso, que luego disminuye hasta que el sistema se estabiliza. El torque de carga muestra una respuesta similar. Inicialmente disminuye por efecto del aumento de velocidad del motor y luego al bajar la velocidad nuevamente aumenta. Es importante mecionar que la estimación de torque es sensible a cambios de frecuencia y estima los efectos de éstos como si fueran cambios de carga.

Las Figuras 4.76 y 4.77 muestran la estimación en el caso de producirse una perturbación de cambio de frecuencia de 50 a 42Hz. En este caso se observa como disminuye la velocidad del rotor, prácticamente con nada de oscilaciones pero con una respuesta lenta. De igual manera el torque de carga cambia con poco rizado pero tiene una dinámica lenta. Nuevamente el filtro reacciona al cambio de frecuencia como si se tratase de un cambio en la carga.

Los resultados de las pruebas con carga hasta deterner completamente el motor se muestran en las Figuras 4.78 y 4.79. En estas gráficas se puede apreciar que la estimación en torque y velocidad prácticamente no tiene rizado. No hay oscilaciones de consideración. La respuesta de las dos oscilaciones es lenta pero con excelente convergencia a 0rad/s cuando el rotor esta detenido completamente.

En las Figuras 4.80 y 4.81 se observa la respuesta del filtro a dos perturbaciones de carga sencillas. Primero se carga y luego se libera la carga del motor. Se observa una respuesta lenta pero consistente con la dinámica del sistema. Las variables estimadas siguen una dinámica muy parecida a las de un sistema de segundo orden sobreamortiguado durante las perturbaciones de carga.

Las Figuras 4.82 y 4.83 muestran la respuesta del filtro cuando existen perturbaciones múltiples en un rango limitado de tiempo. El objetivo de esta prueba es estudiar la respuesta del filtro a cambios repentinos y continuos. El filtro responde de una manera muy adecuada, con cambios sin oscilaciones importantes durante los transitorios pero con una respuesta un tanto lenta, que hace que no se observen picos pronunciados sino más bien curvas suaves.

En las Figuras 4.84 y 4.85 se observa la estimación cuando el motor arranca con el rotor bloqueado. Se observa que no hay oscilaciones alrededor de 0rad/s en el estado inicial y posteriormente al liberar el rotor la velocidad aumenta hasta alcanzar el estado estable. Sin embargo este proceso ocurre de manera lenta como se puede observar. En el caso del torque de

carga la señal estimada empieza en 2.5Nm aproximadamente y disminuye hasta poco menos de 0Nm de forma paulatina pero sin oscilaciones sostenidas de importancia.

En la última prueba se aplica carga al motor, luego se aumenta dicha carga para finalmente liberarlo. Este procedimiento y su respectiva respuesta del filtro se observa en las Figuras 4.86 y 4.87 donde se ve que el sistema responde acorde a lo que se espera pero de forma sobreamortiguada de decirlo de alguna manera. No se aprecian picos pronunciados sino más bien respuestas suaves a las perturbaciones. Este tipo de comportamiento ya se había observado en simulación, donde durante las perturbaciones de carga el sistema respondía de forma lenta hasta converger en el valor final.

4.3.5. Filtro H_∞ extendido- Frecuencia media (25-40 Hz)

Se realizaron las pruebas detalladas en la introducción de esta sección con una frecuencia de alimentación de 30Hz. A continuación se muestran los resultados. Como ya se mencionó se muestran los voltajes y corrientes $dq0$ del estator (medidos vía muestreo en el DSP) y las variables estimadas: flujos $dq0$ de estator y rotor, velocidad del rotor y torque de carga.

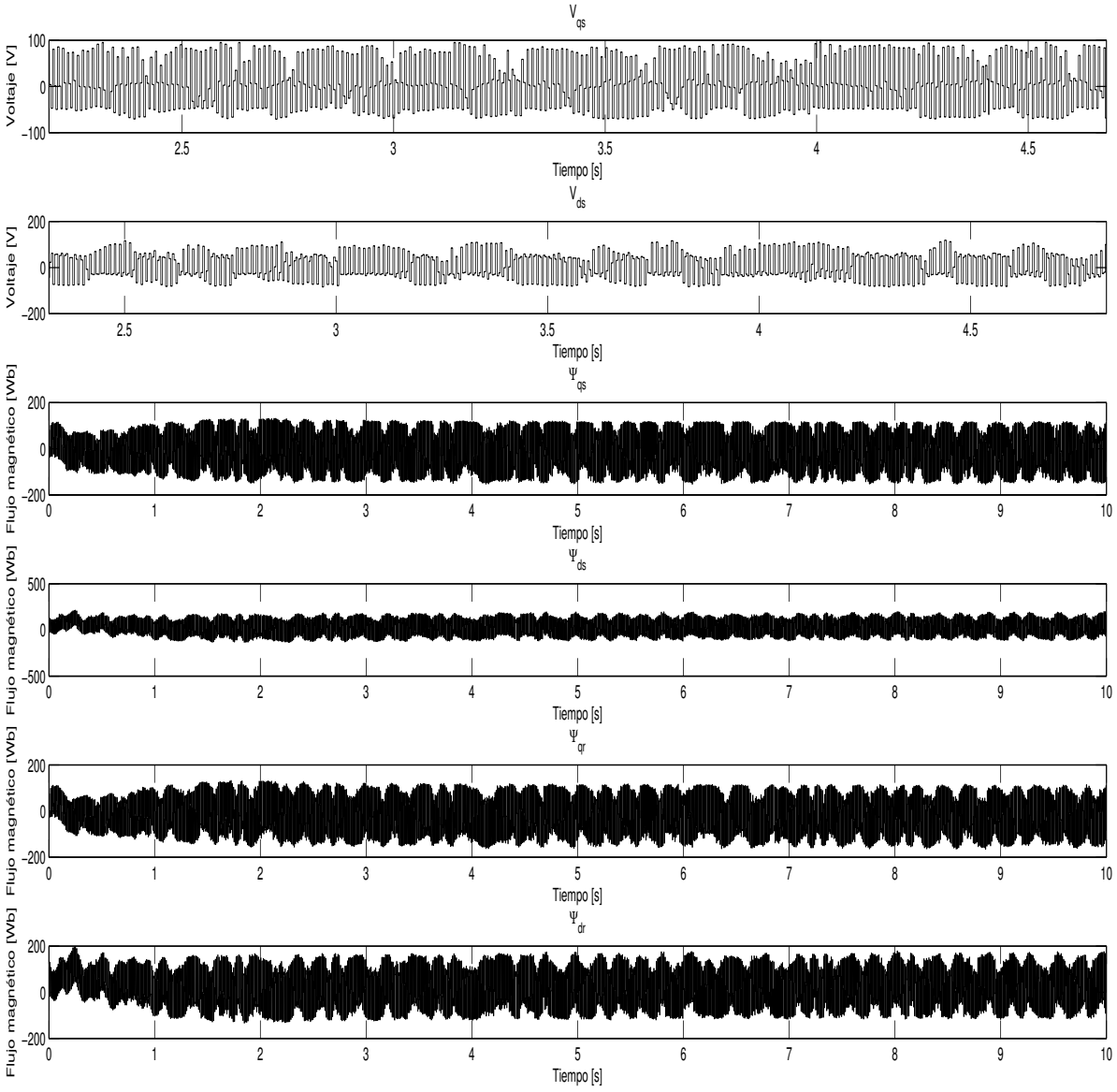


Figura 4.88: Estimación H_∞ a 30Hz en el arranque sin carga (Voltajes y Flujos)

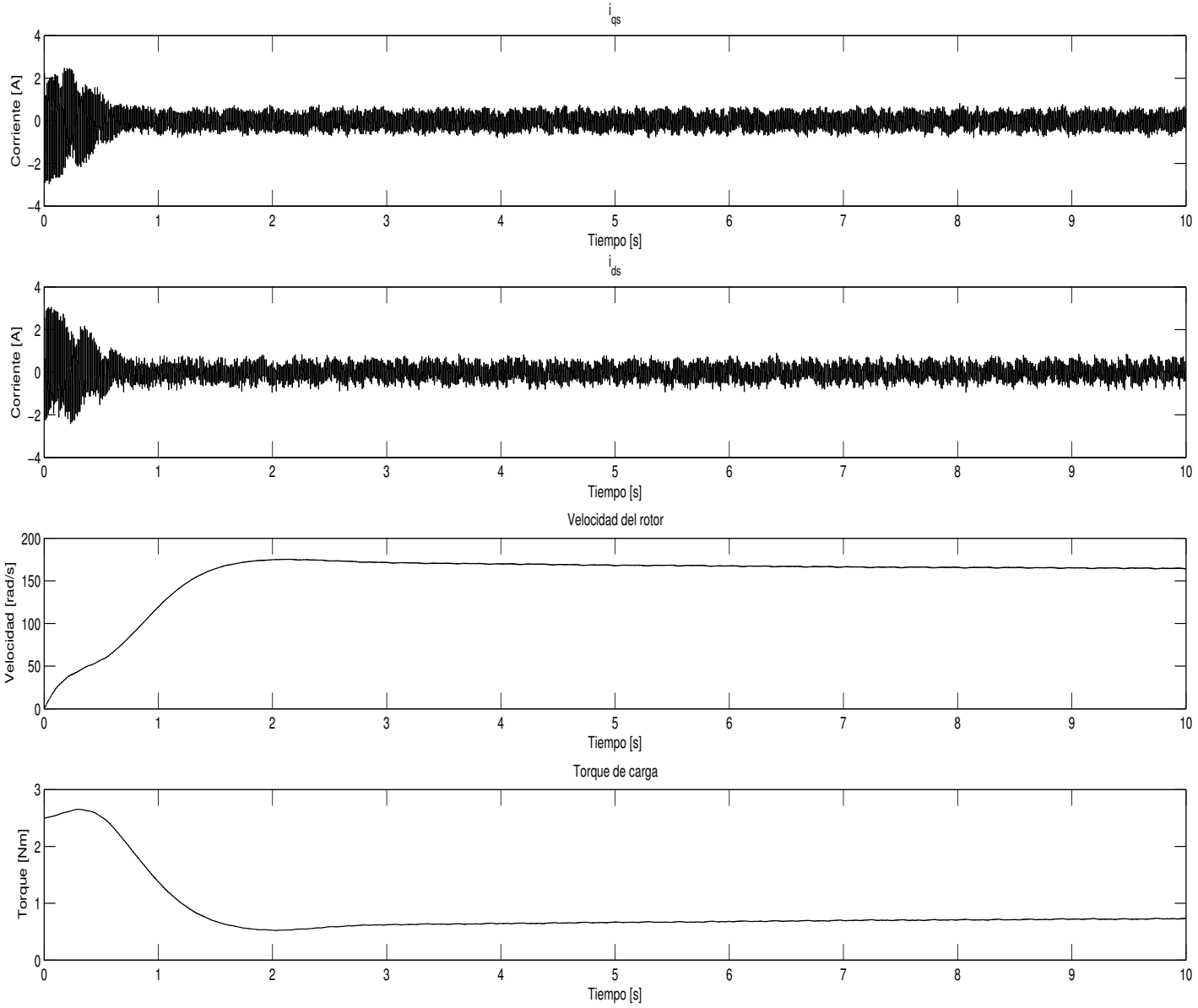


Figura 4.89: Estimación H_∞ a 30Hz en el arranque sin carga (Corrientes, velocidad y torque)

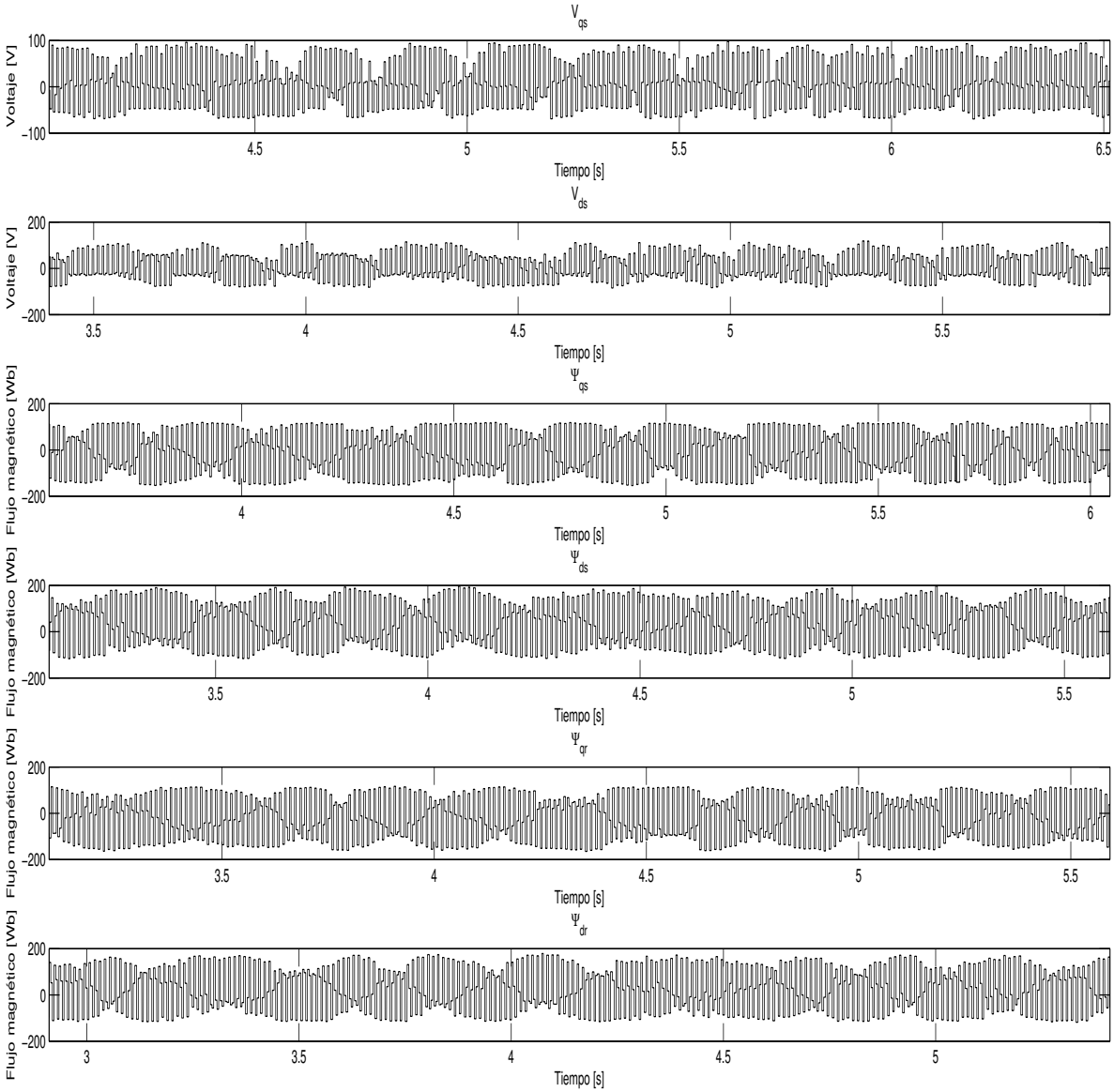


Figura 4.90: Estimación H_∞ a 30Hz en estado estable sin carga (Voltajes y Flujos)

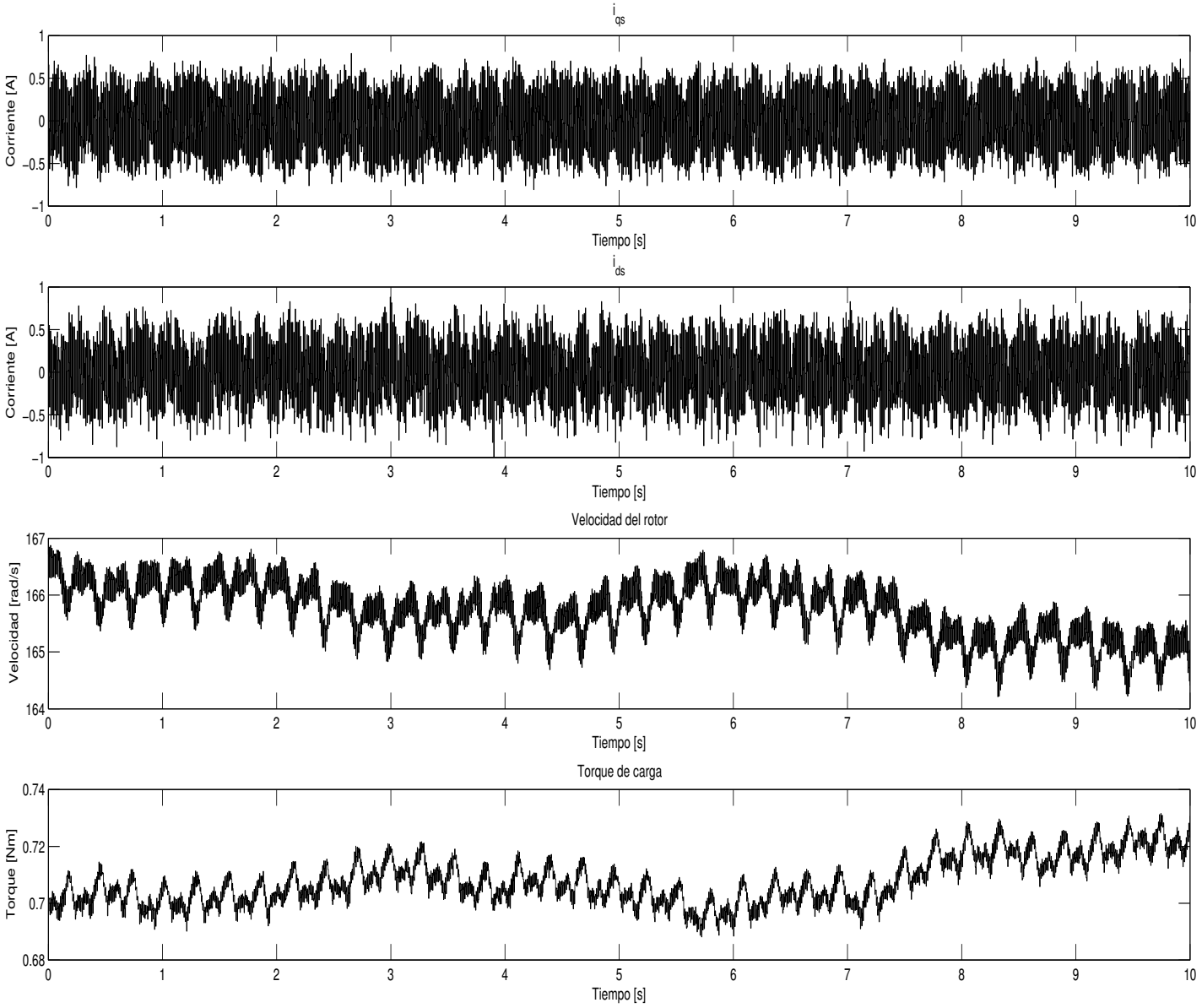


Figura 4.91: Estimación H_∞ a 30Hz en estado estable sin carga (Corrientes, velocidad y torque)

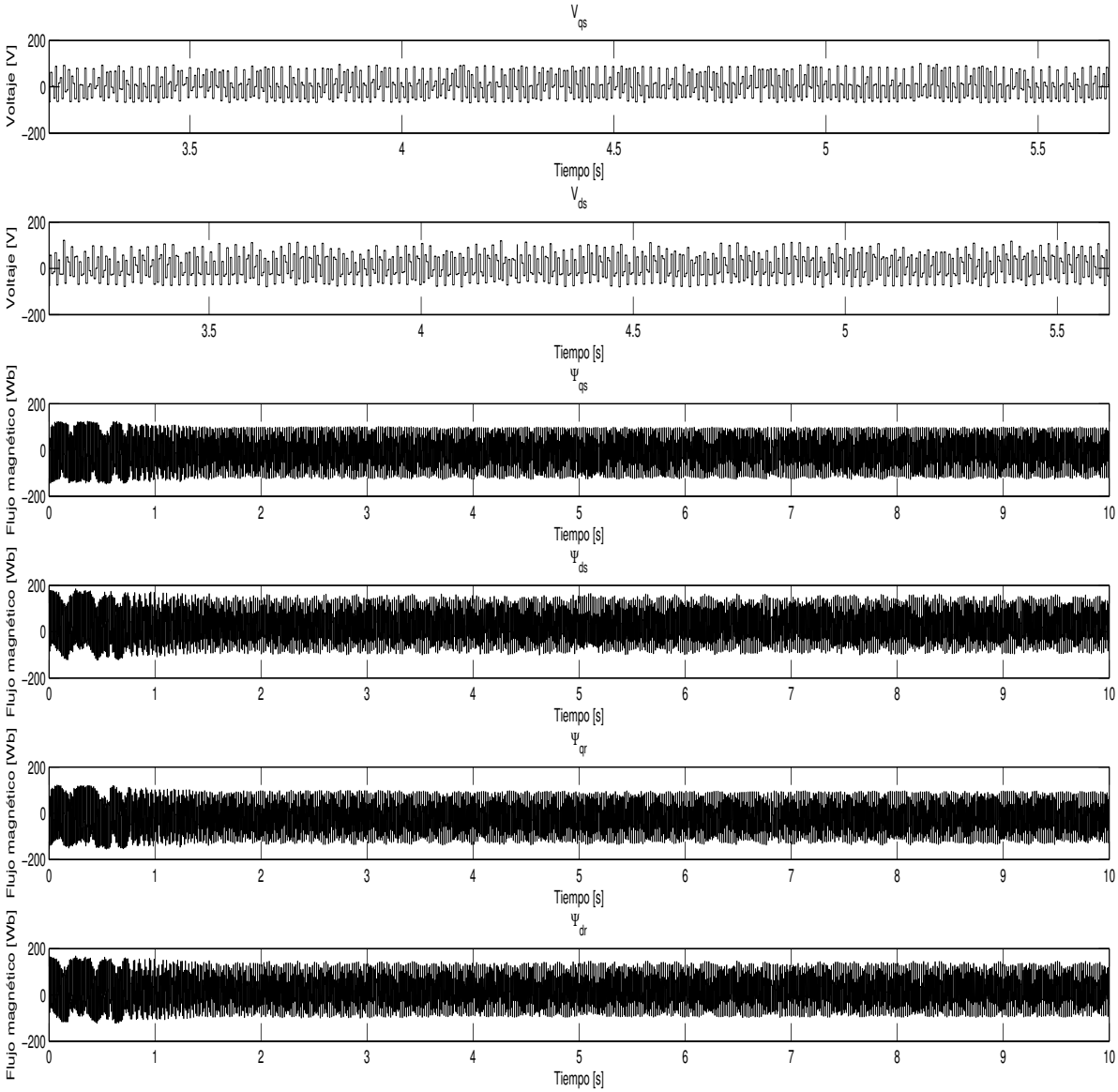


Figura 4.92: Estimación H_∞ con cambio de 30Hz a 35Hz (Voltajes y Flujos)

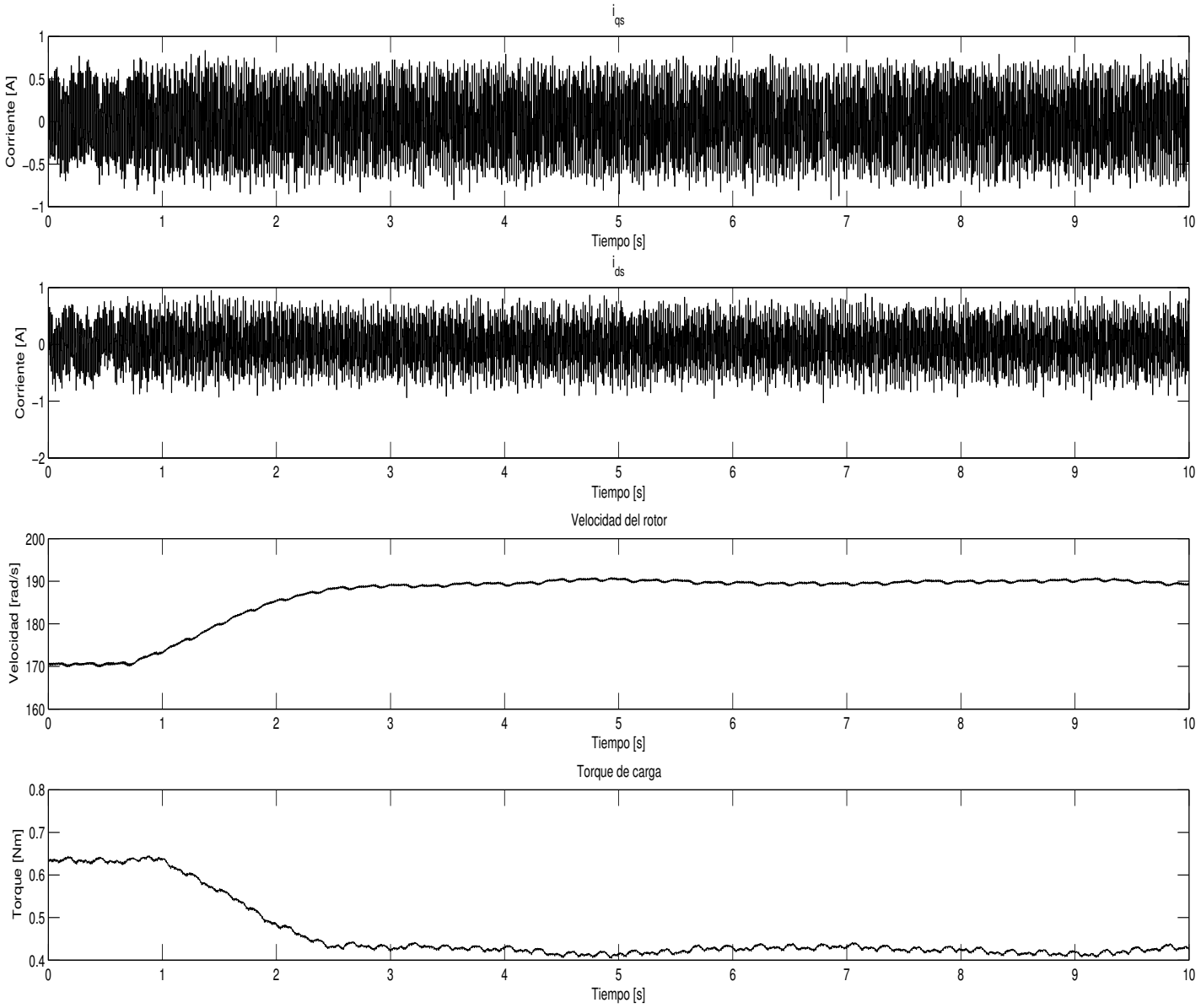


Figura 4.93: Estimación H_∞ con cambio de 30Hz a 35Hz (Corrientes, velocidad y torque)

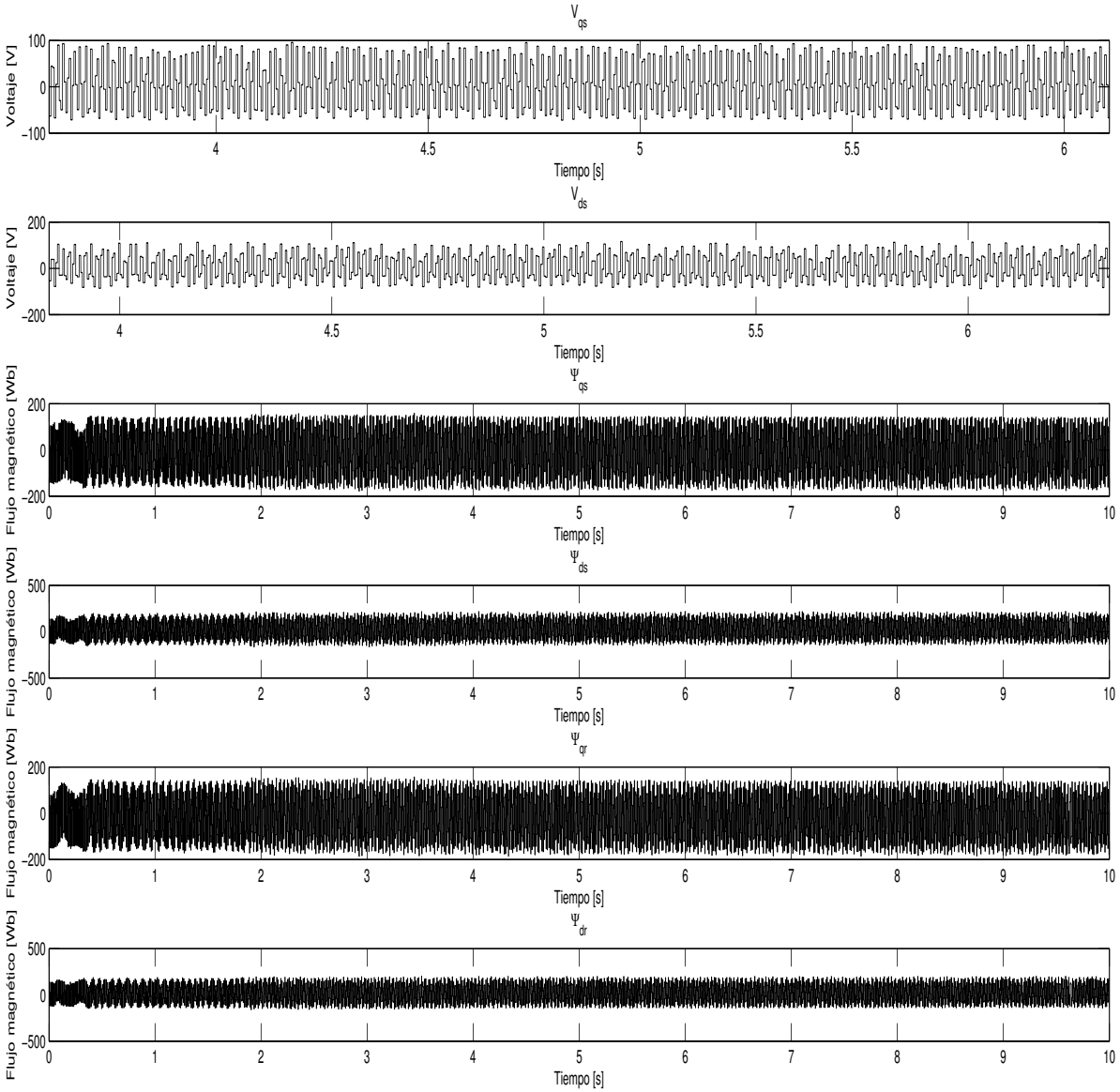


Figura 4.94: Estimación H_∞ con cambio de 30Hz a 26 Hz (Voltajes y Flujos)

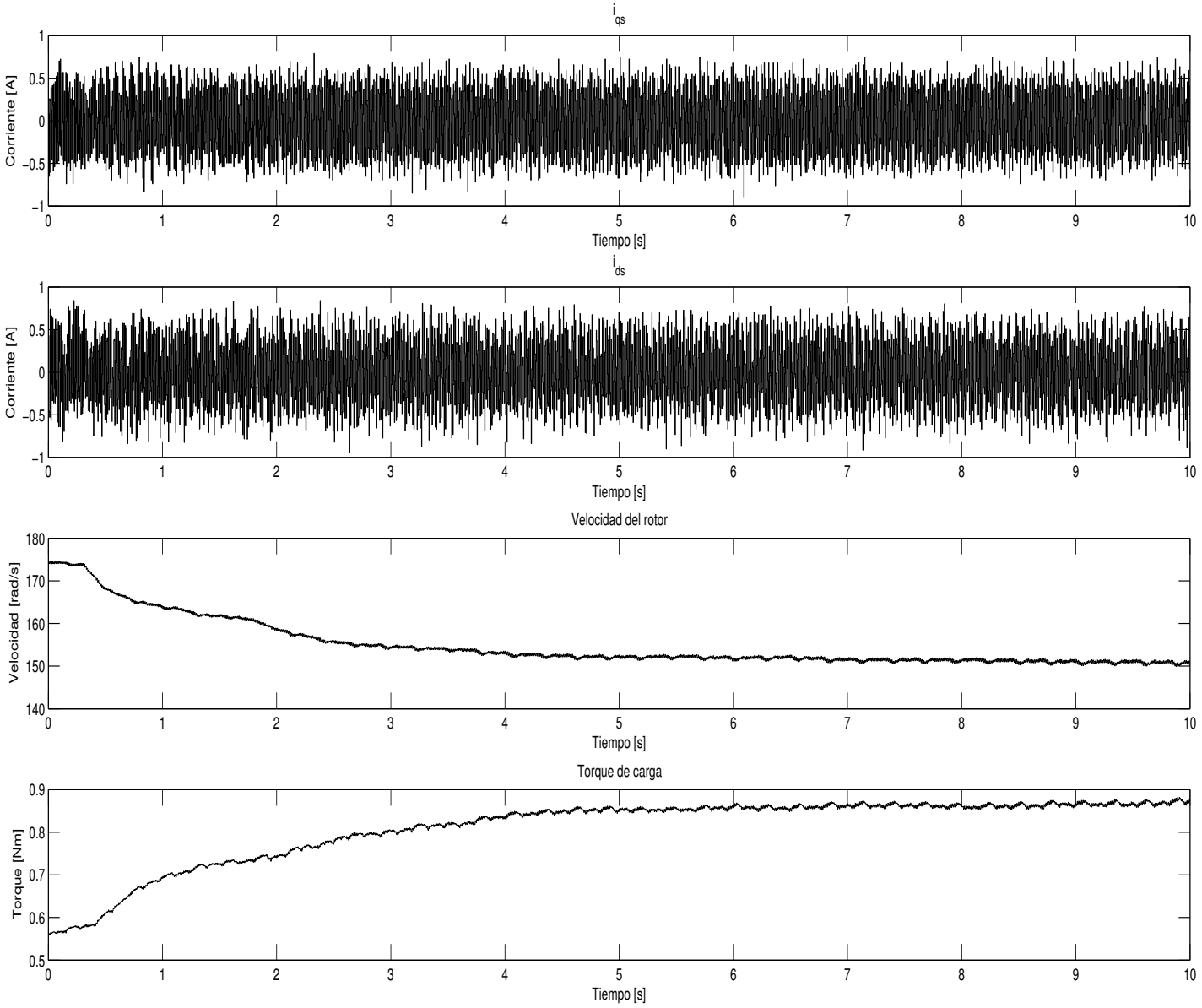


Figura 4.95: Estimación H_∞ con cambio de 30Hz a 26 Hz (Corrientes, velocidad y torque)

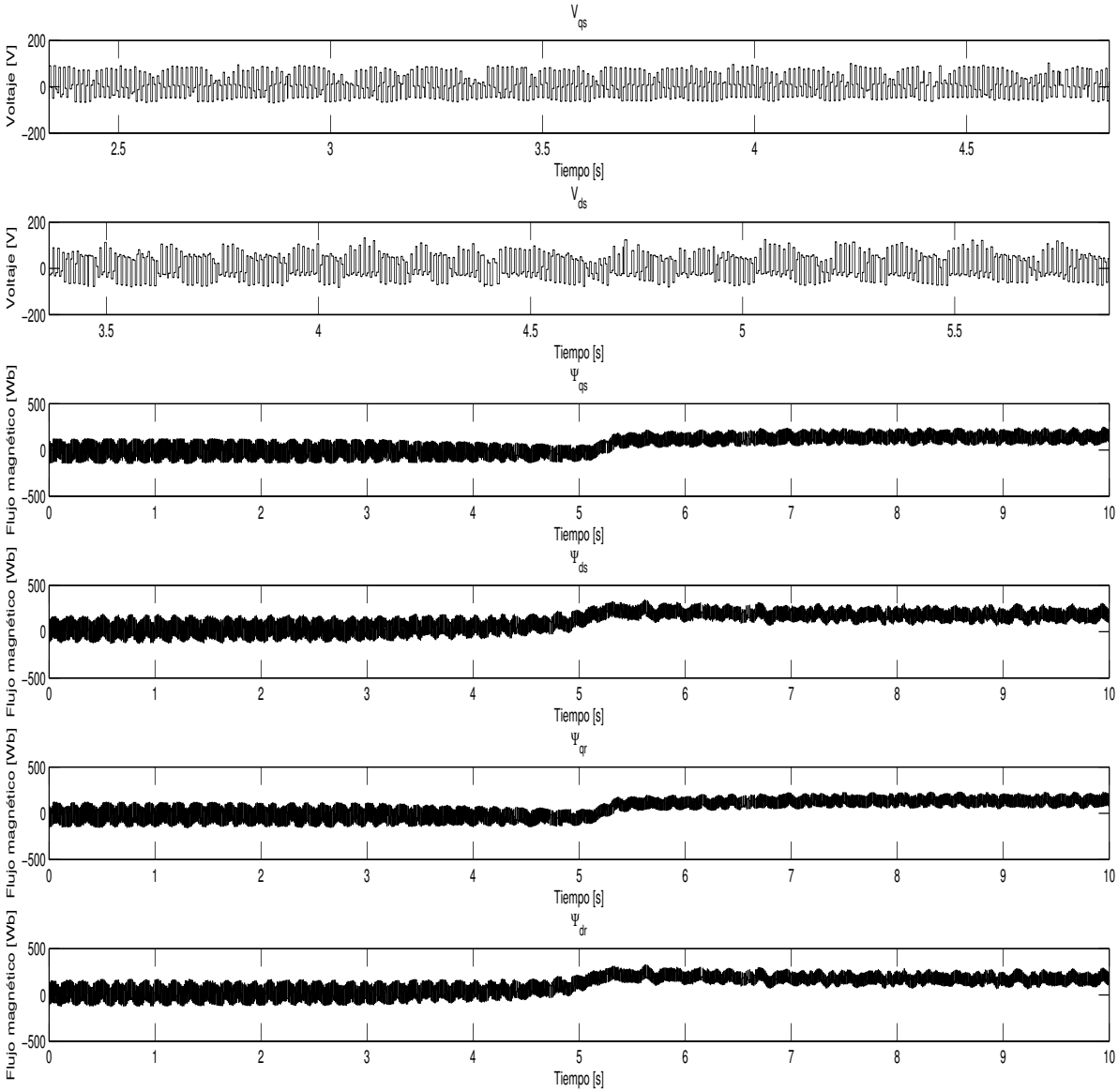


Figura 4.96: Estimación H_∞ con frenado total (Voltajes y Flujos)

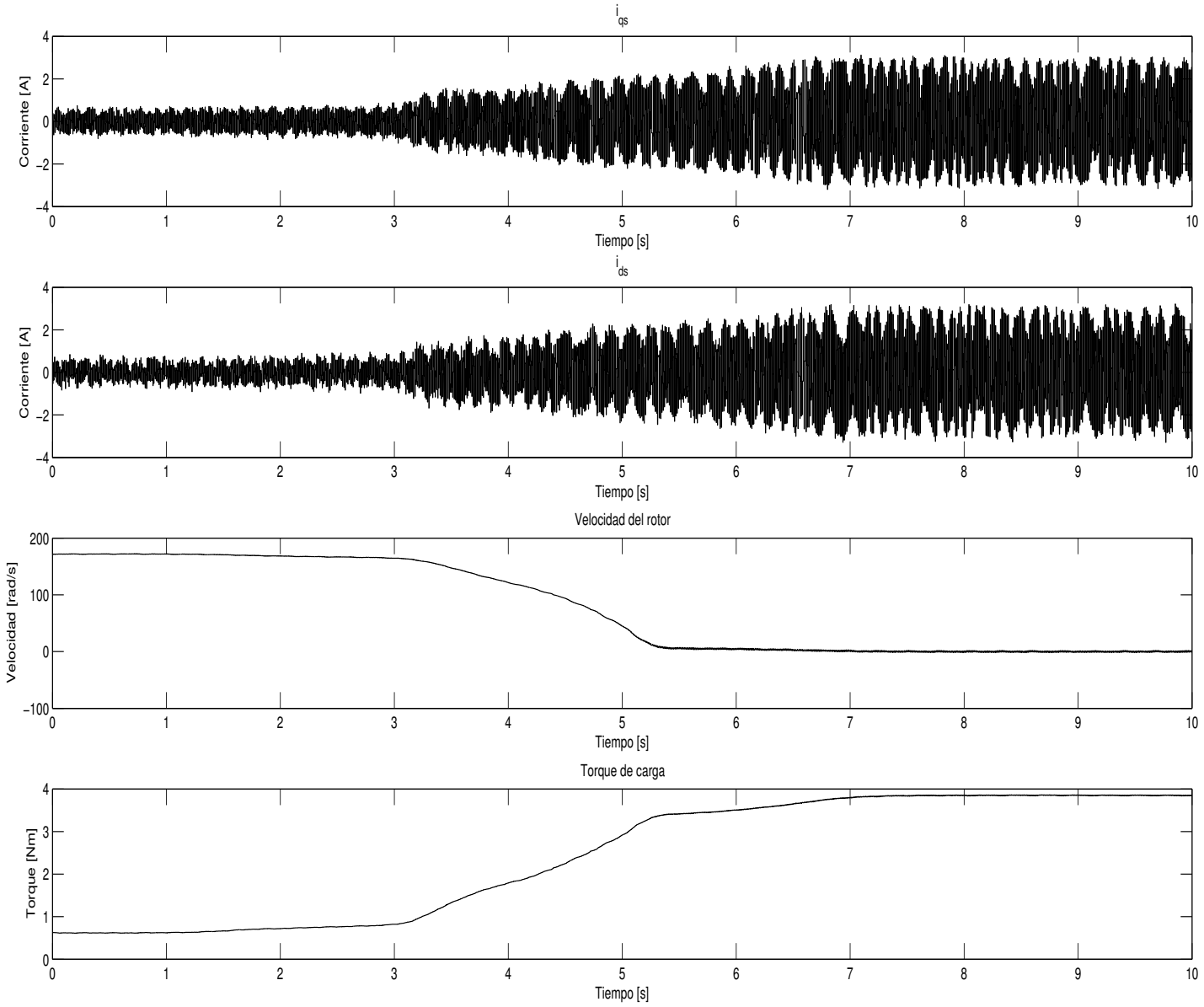


Figura 4.97: Estimación H_∞ con frenado total (Corrientes, velocidad y torque)

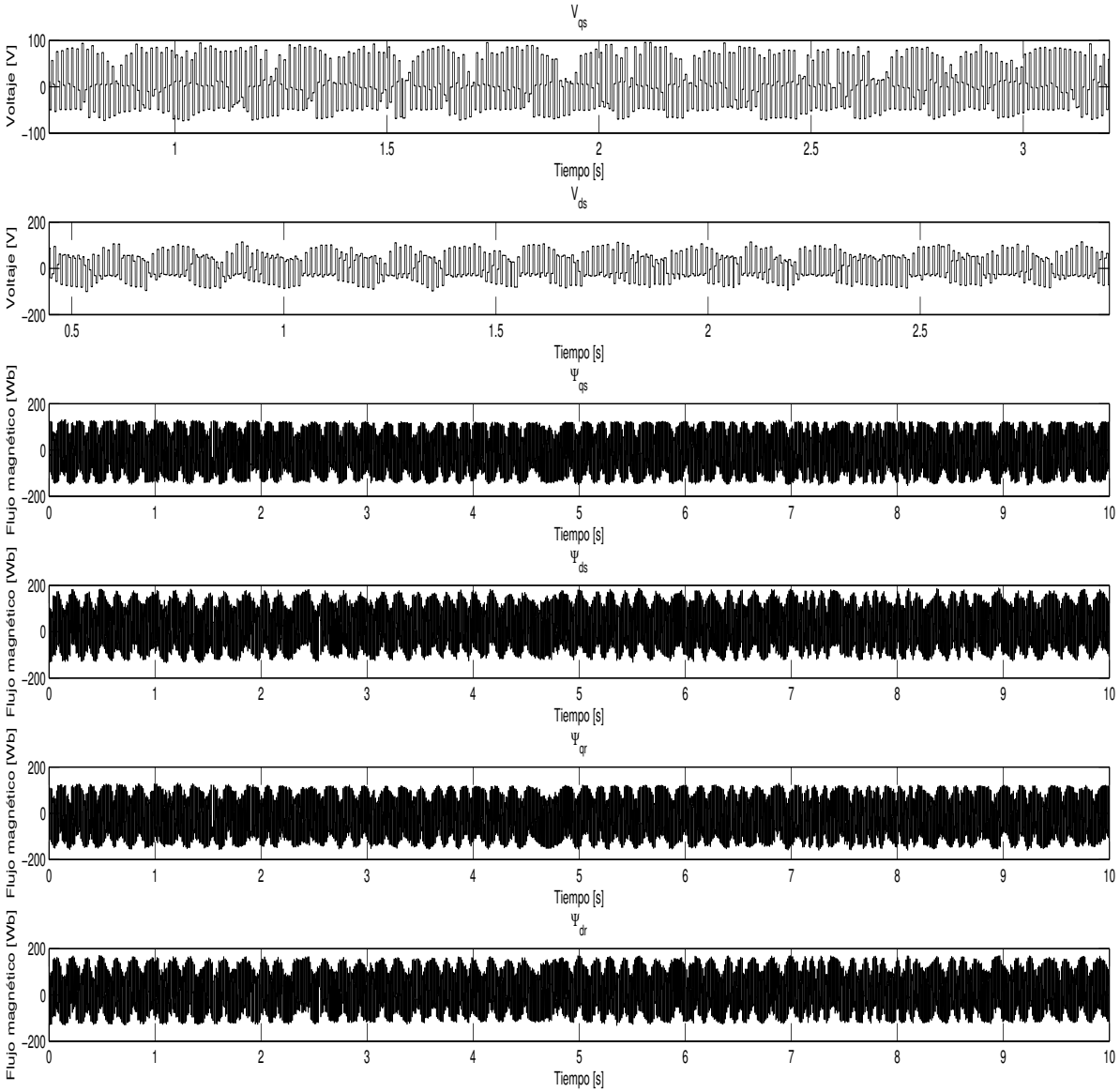


Figura 4.98: Estimación H_∞ con cambios de carga (Voltajes y Flujos)

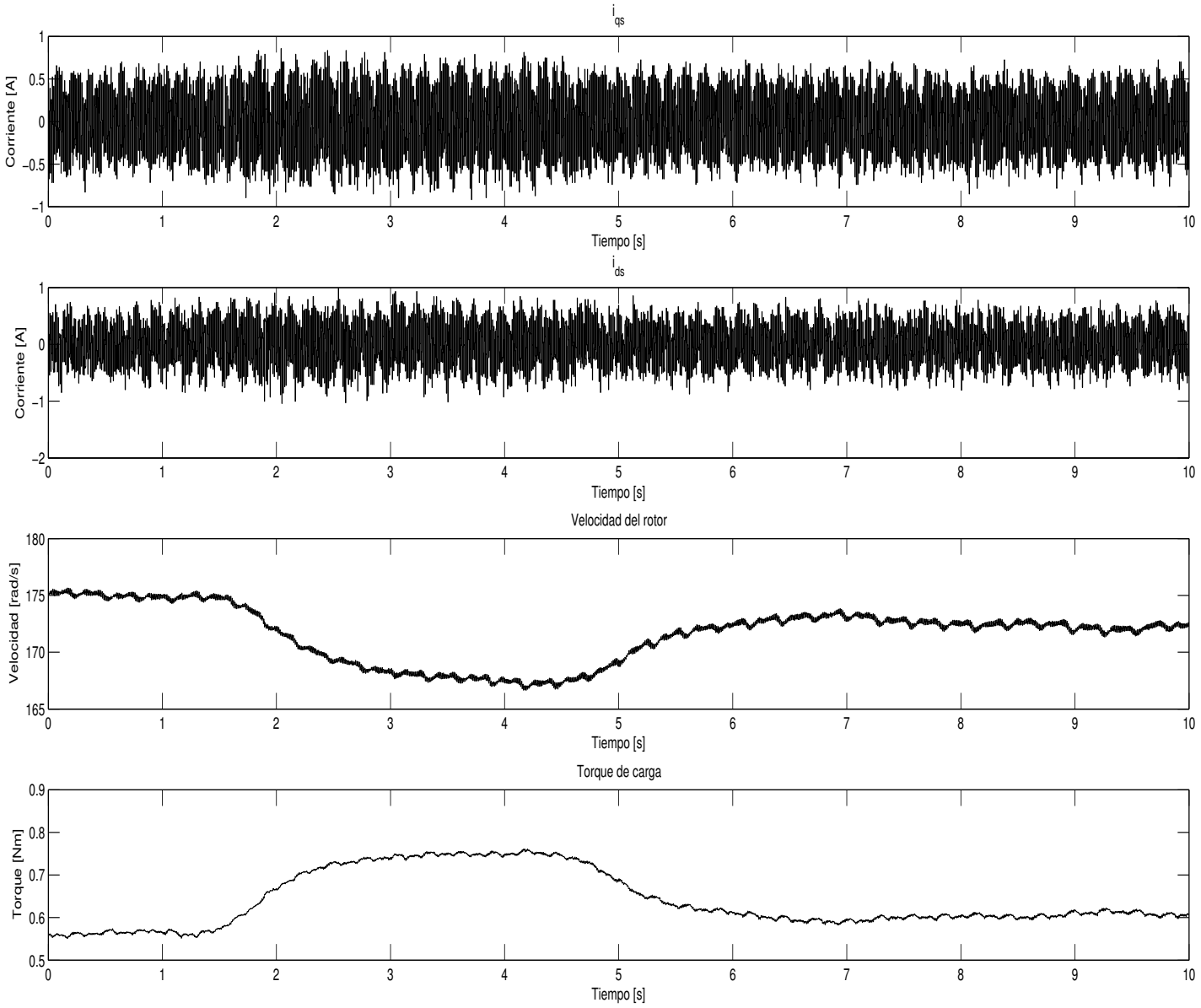


Figura 4.99: Estimación H_∞ con cambios de carga (Corrientes, velocidad y torque)

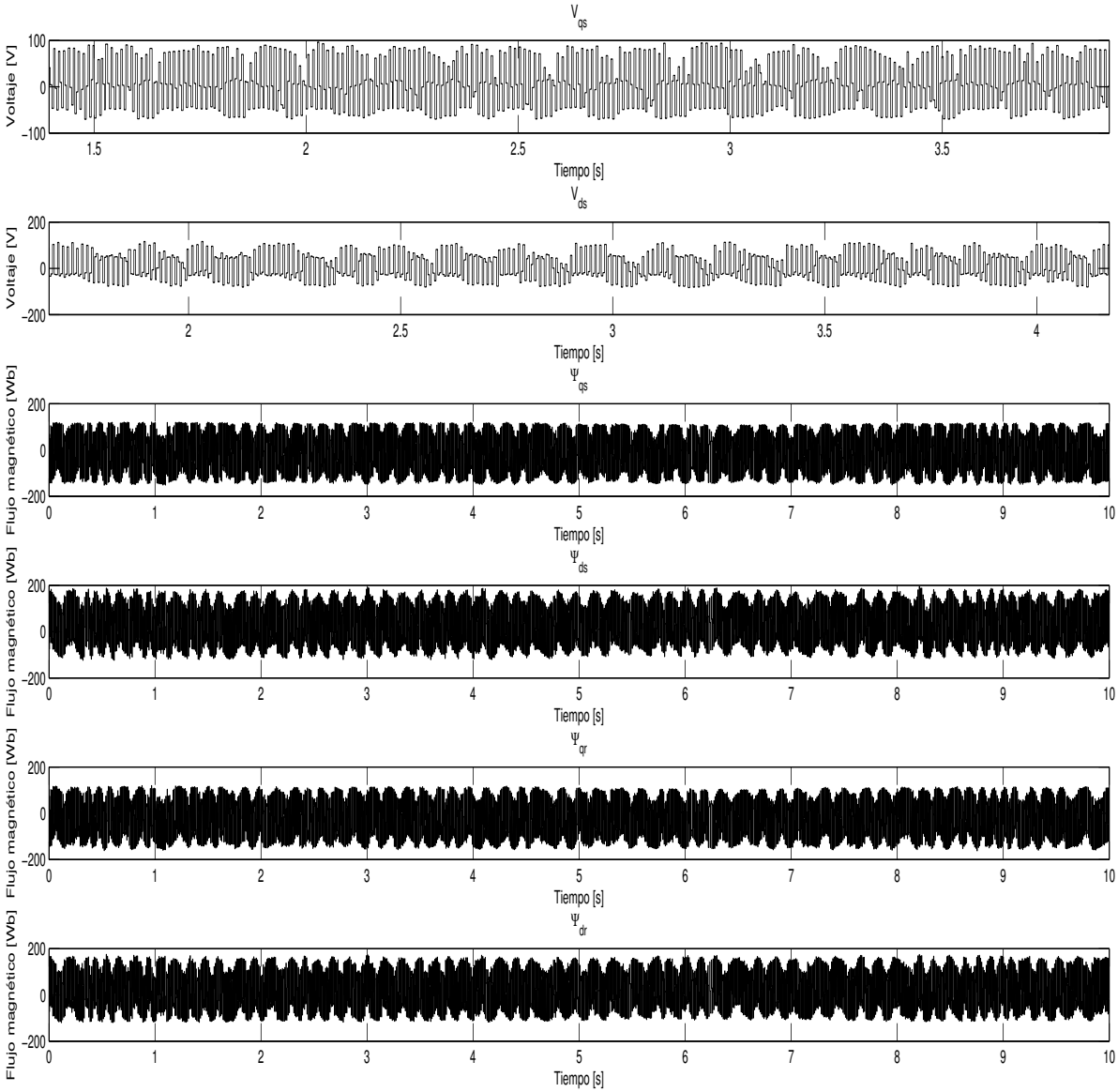


Figura 4.100: Estimación H_∞ con cambios de carga múltiples (Voltajes y Flujos)

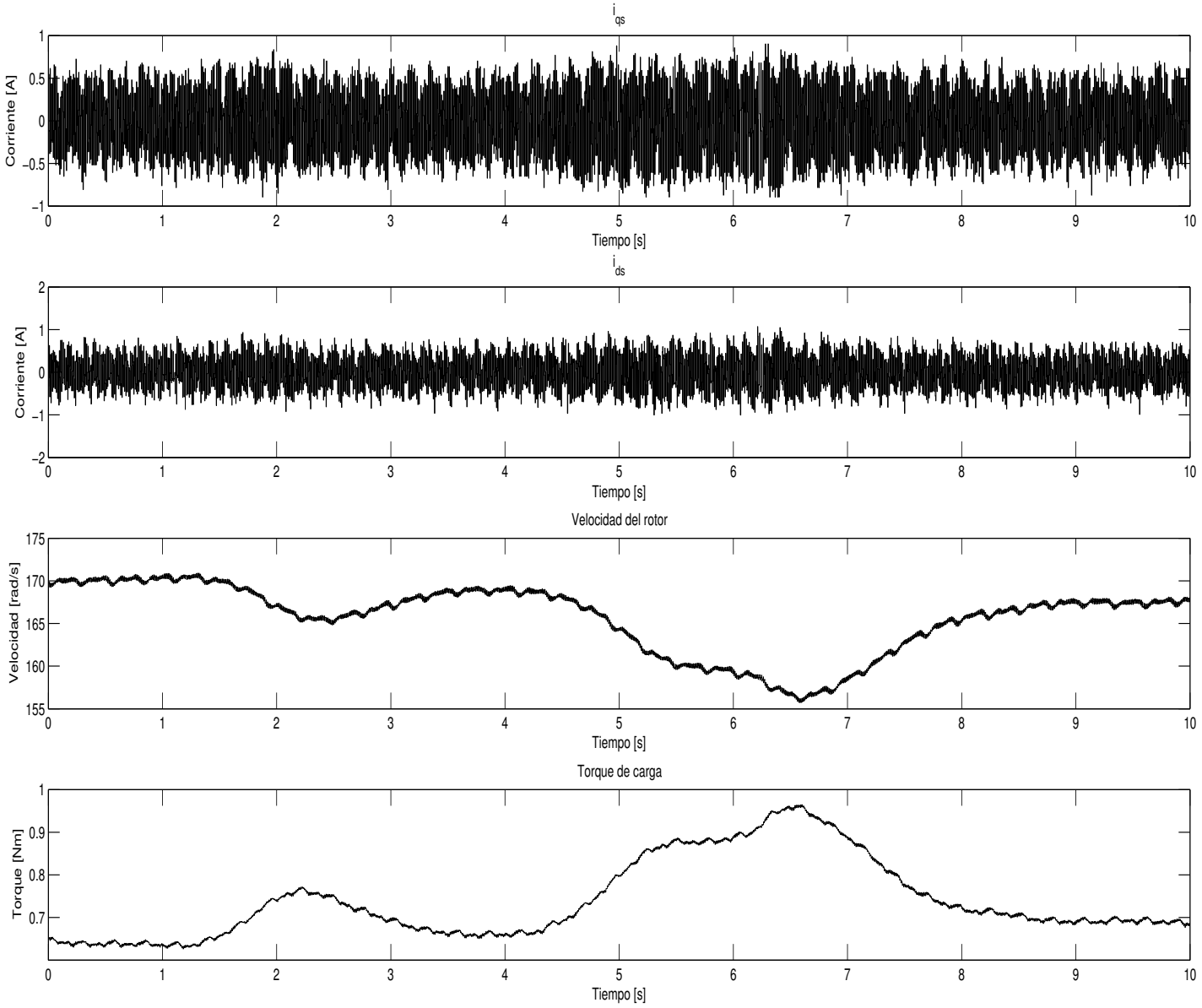


Figura 4.101: Estimación H_∞ con cambios de carga múltiples (Corrientes, velocidad y torque)

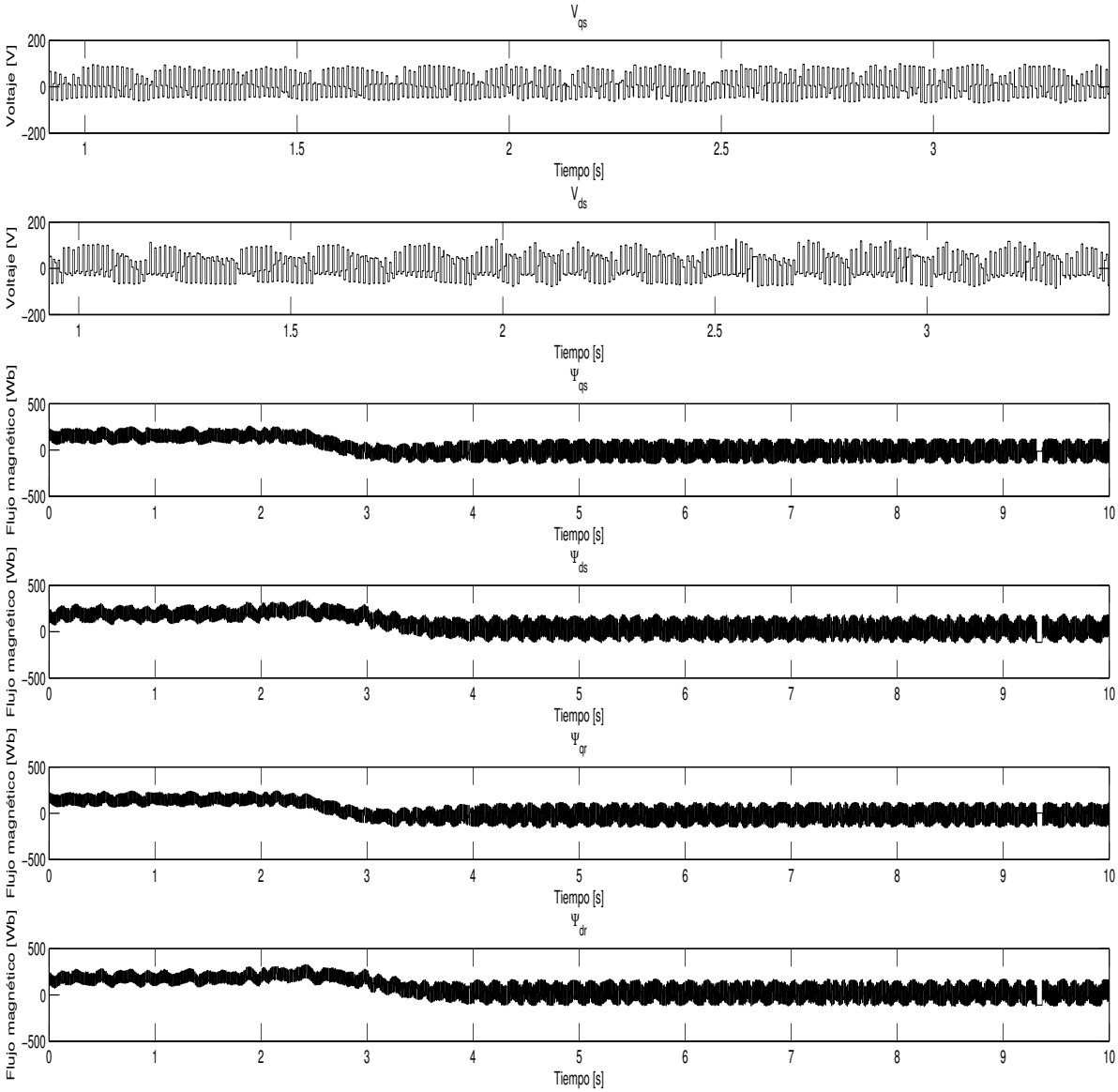


Figura 4.102: Estimación H_∞ con carga total al arranque (Voltajes y Flujos)

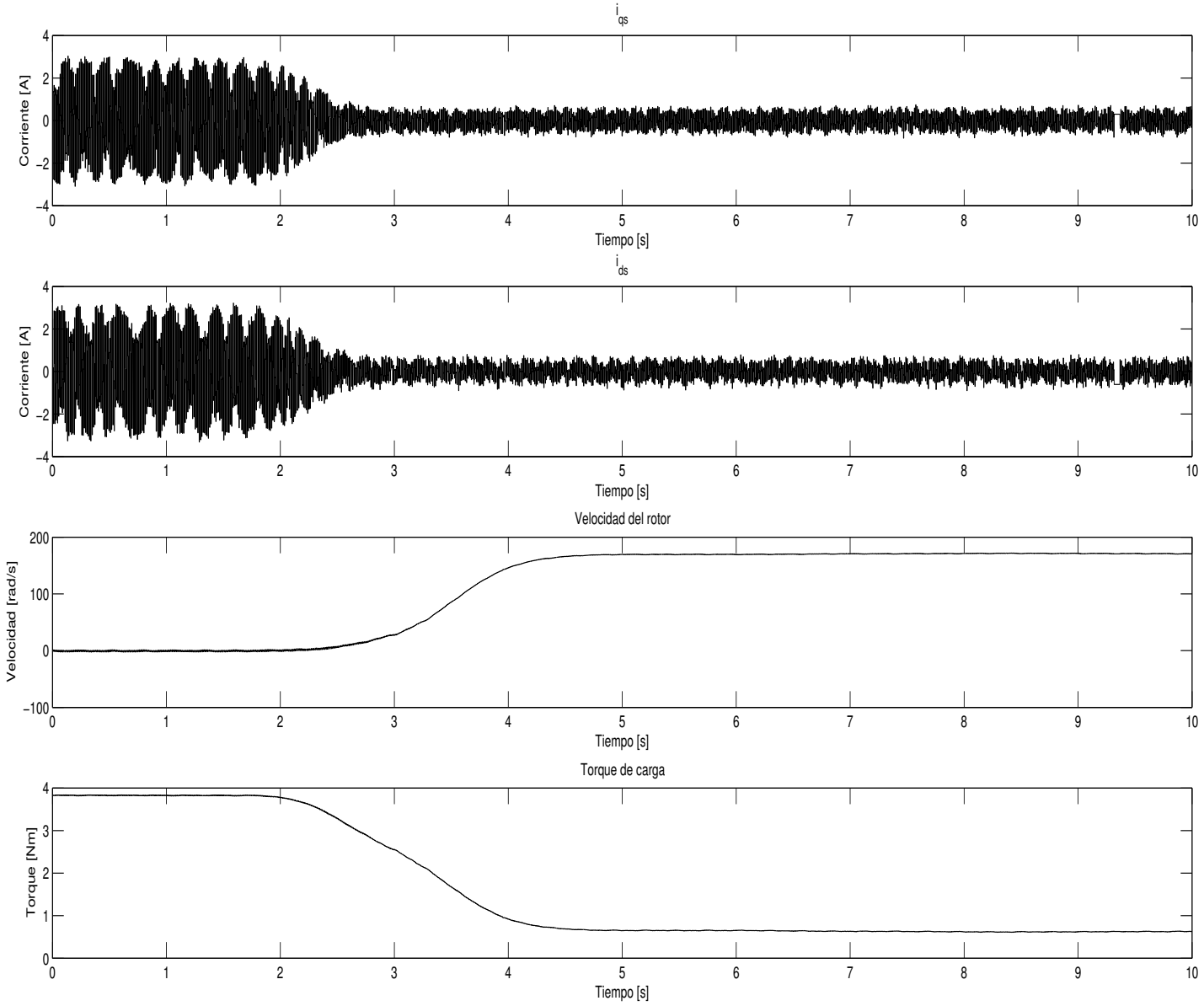


Figura 4.103: Estimación H_∞ con carga total al arranque (Corrientes, velocidad y torque)

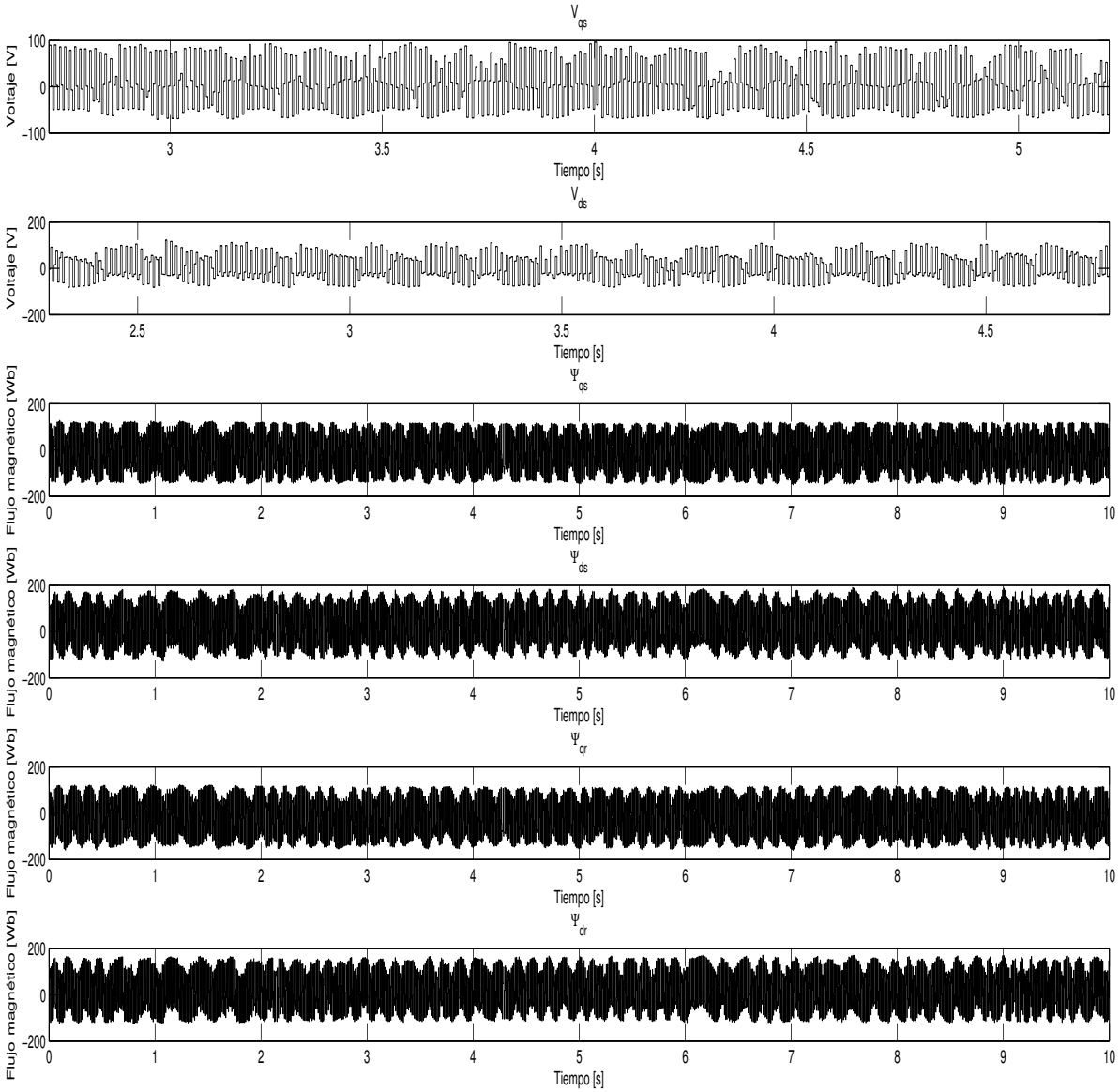


Figura 4.104: Estimación H_∞ con carga y posterior liberación de carga (Voltajes y Flujos)

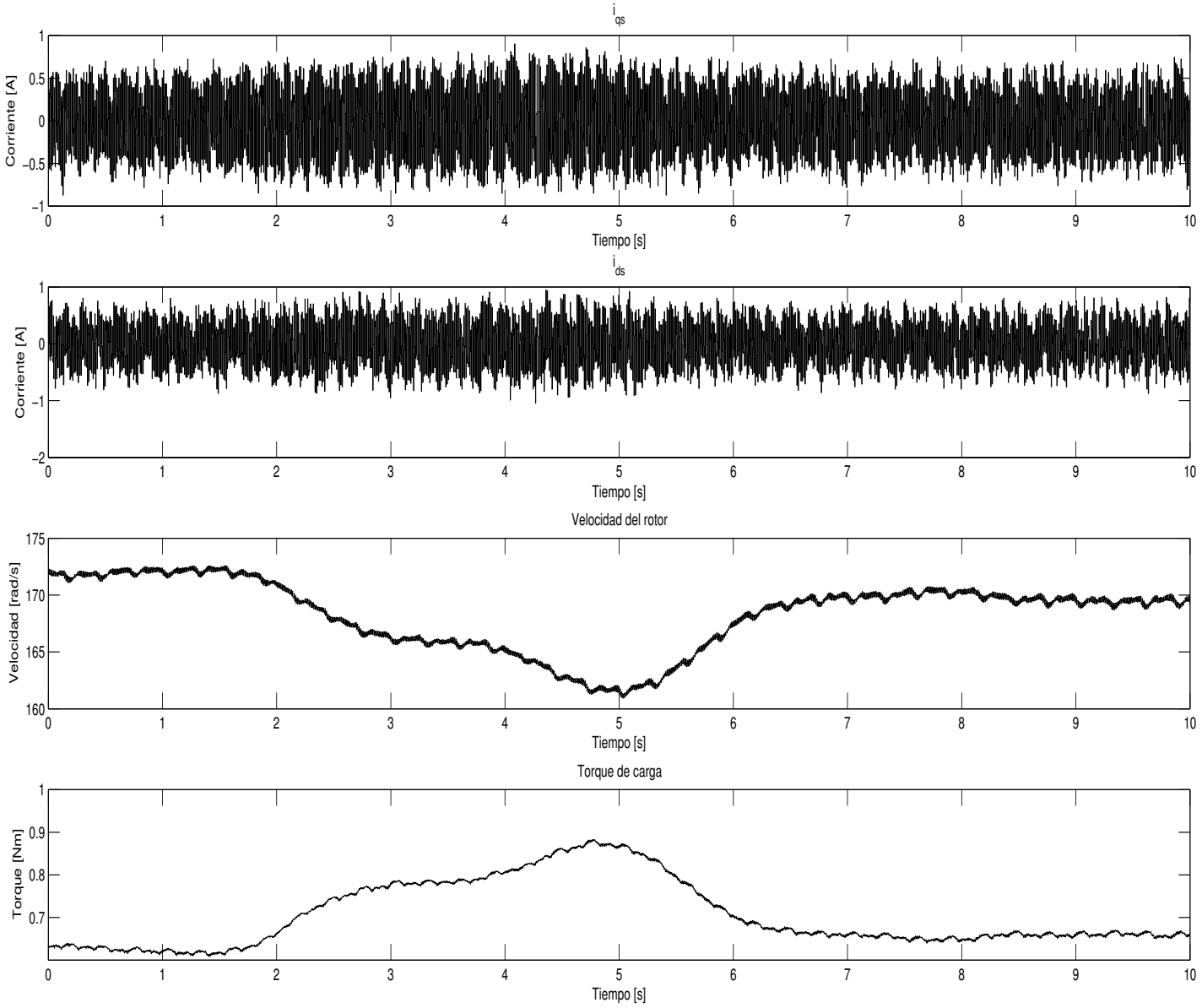


Figura 4.105: Estimación H_∞ con carga y posterior liberación de carga (Corrientes, velocidad y torque)

En las Figuras 4.88 y 4.89 se observa el arranque del motor con una frecuencia de alimentación de voltaje de 30Hz. En el caso de la velocidad el sistema converge a una velocidad de aproximadamente a 170rad/s. La estimación es libre de oscilaciones durante el transitorio, no obstante, la respuesta estimada luce lenta. En cuanto al torque de carga, la estimación

muestra un valor de 0.7Nm iniciales, que se pueden considerar como una desviación debido a la operación a 30Hz. La estimación de torque no diferencia entre perturbaciones debido a carga y perturbaciones por frecuencia. El filtro se comporta como si la menor velocidad del rotor estuviera dada por carga y no por la frecuencia reducida en la alimentación. La curva de torque no presenta un pico pronunciado sino más bien una curva suave.

Las Figuras 4.90 y 4.91 muestran el comportamiento del filtro en la operación en estado estable. Se aprecia que la velocidad tiene una oscilación en estado estable de menos de 2rad/s Es decir es una estimación muy limpia, con poco rizado. Además se puede apreciar una segunda oscilación de mayor periodo, que de igual manera no varía más allá de los 2rad/s. En el caso del torque de carga se observa oscilaciones de 0.03Nm alrededor de 0.7Nm que actua como un valor de desviación en este caso. El filtro esta modelando el cambio frecuencia de operación como si fuera una carga mecánica en el motor. No obstante, la dinámica del filtro es consecuente con la dinámica del motor.

La respuesta del filtro a un cambio de frecuencia de operación de 30 a 35Hz se observa en las Figuras 4.92 y 4.93. Se aprecia el aumento de la velocidad del rotor según lo esperado sin oscilaciones sostenidas de importancia. De igual forma la respuesta del sistema se aprecia más rápida en este caso que en los casos observados al operar a 50Hz. En el caso del torque de carga se ve que éste reacciona como si el aumento de frecuencia se tratáse de una disminución de carga. Debido a que el motor gira más rápido, el sistema ve una menor carga.

La respuesta del filtro a un cambio de frecuencia de operación de 30 a 26Hz se muestra en las Figuras 4.94 y 4.95. La perturbación disminuye la velocidad del rotor, pero como se observa en la gráfica correspondiente, la respuesta del filtro a este cambio es lenta hasta converger a su valor final. Durante la convergencia no se observan oscilaciones significativas. Por otra parte, el torque estimado aumenta ya que el filtro no diferencia esta perturbación en frecuencia de una perturbación en carga. Esta estimación también se observa libre de oscilaciones significativas durante el estado transitorio. El rizado es mínimo.

Las Figuras 4.96 y 4.97 muestran la respuesta del filtro cuando la carga aplicada detiene totalmente al rotor. En este caso se observa como la velocidad estimada disminuye un tanto lenta en su respuesta dinámica, pero no se observan grandes oscilaciones alrededor de velocidades negativas como se observó en el filtro Kalman. La respuesta converge a 0rad/s. Es claro que hay oscilaciones alrededor de 0rad/s pero éstas siguen siendo tolerables. En el caso del torque de carga se aprecia el aumento del mismo hasta aproximadamente 3.8Nm. Se debe tener en cuenta que el torque inicial es 0.8Nm producto de la operación a 30Hz (el filtro considera esta condición como carga que desacelera el motor). La respuesta del filtro es lenta pero sin oscilaciones significativas como fue en el caso de Kalman.

En las Figuras 4.98 y 4.99 se observa la respuesta del filtro a cambios sencillos de carga. Se aumenta la carga inicialmente, para luego liberar completamente al motor. Se puede observar que la respuesta dinámica es lenta. No obstante, el comportamiento en general del sistema es el esperado y con poco rizado durante el proceso.

Para investigar la respuesta del estimador cuando existen múltiples perturbaciones en un intervalo corto de tiempo se realizan las pruebas documentadas en las Figuras 4.100 y 4.101. Se aprecia que el filtro efectivamente responde a las perturbaciones aplicadas aunque de forma lenta. No existen picos pronunciados como se esperaría sino más bien una respuesta suave

durante la estimación. Tanto la velocidad como el torque reaccionan sin oscilaciones sostenidas grandes.

Se realizan pruebas con el rotor totalmente bloqueado al principio de la operación; y posteriormente se lo libera. Los resultados se muestran en las Figuras 4.102 y 4.103 Se aprecia que la estimación inicial de velocidad es de 0rad/s libre de oscilaciones significativas. Una vez que se libera el rotor el motor aumenta su velocidad hasta llegar al estado estable. La respuesta dinámica es lenta pero se compensa con las no oscilaciones que se observan con el rotor bloqueado. En el caso del torque se aprecia un valor inicial de 3.8Nm muy estable, sin oscilaciones, que converge hasta 0.7Nm que es el valor esperado por la desviación de operación a 30Hz. La estimación en torque tampoco tiene rizado de importancia.

Las Figuras 4.104 y 4.105 muestran la estimación en el caso en que se aplica una carga inicial, luego ésta se aumenta y finalmente se libera el motor. No se aprecian picos durante la estimación sino más bien curvas suaves propias de la reacción a una perturbación de un sistema de segundo orden sobreamotiguado. Las oscilaciones durante la estimación son pequeñas y el sistema reacciona de forma lenta a las perturbaciones. Al momento de liberar completamente al rotor se puede apreciar que el filtro tarda en regresar completamente al valor inicial de velocidad y torque aunque la tendencia indica que es el punto de convergencia final.

4.3.6. Filtro H_∞ extendido-Baja frecuencia (0-25 Hz)

Se realizaron las pruebas detalladas en la introducción de esta sección con una frecuencia de alimentación de 20Hz. A continuación se muestran los resultados. Como ya se mencionó se muestran los voltajes y corrientes $dq0$ del estator (medidos vía muestreo en el DSP) y las variables estimadas: flujos $dq0$ de estator y rotor, velocidad del rotor y torque de carga.

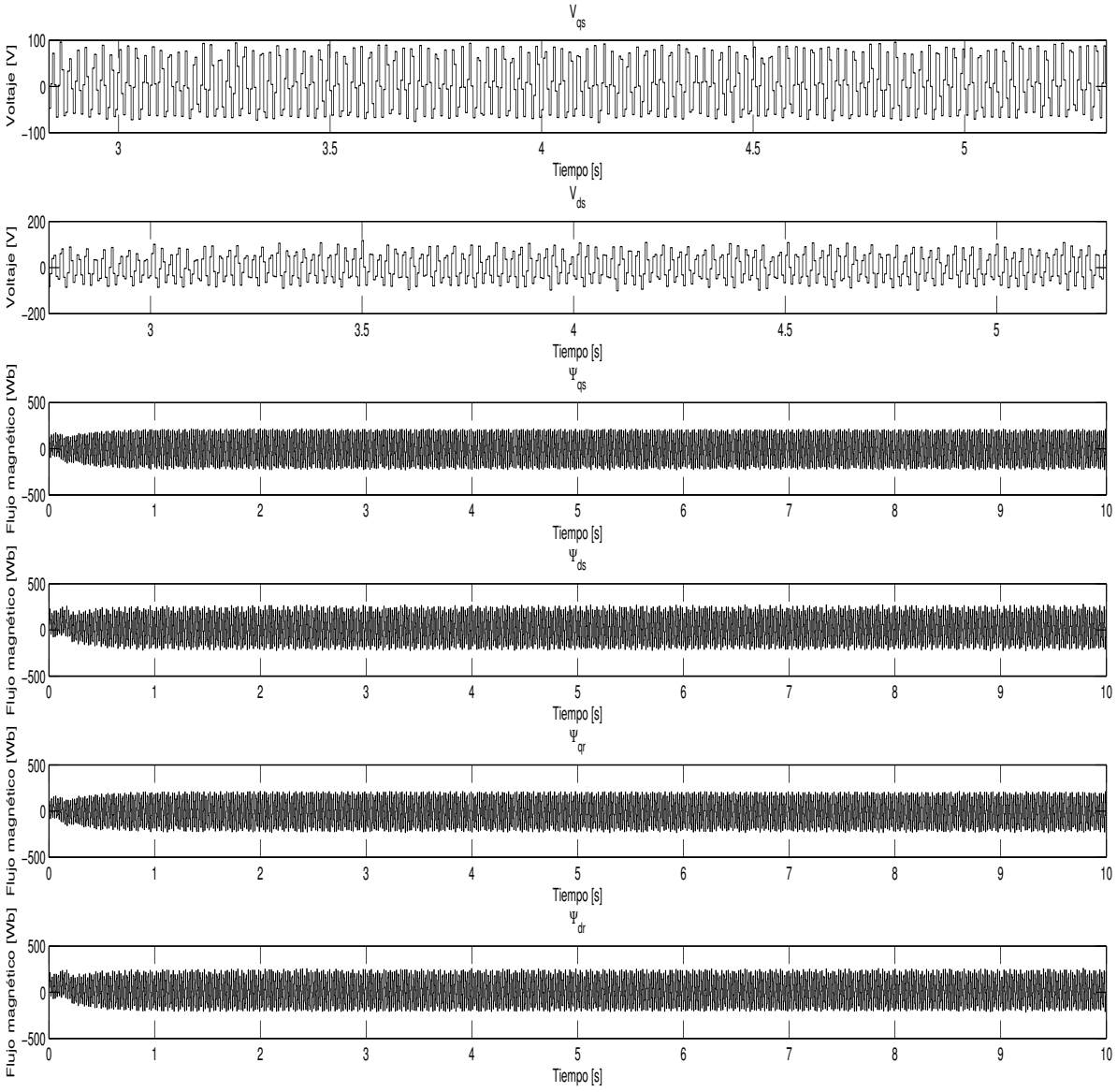


Figura 4.106: Estimación H_∞ a 20Hz en el arranque sin carga (Voltajes y Flujos)

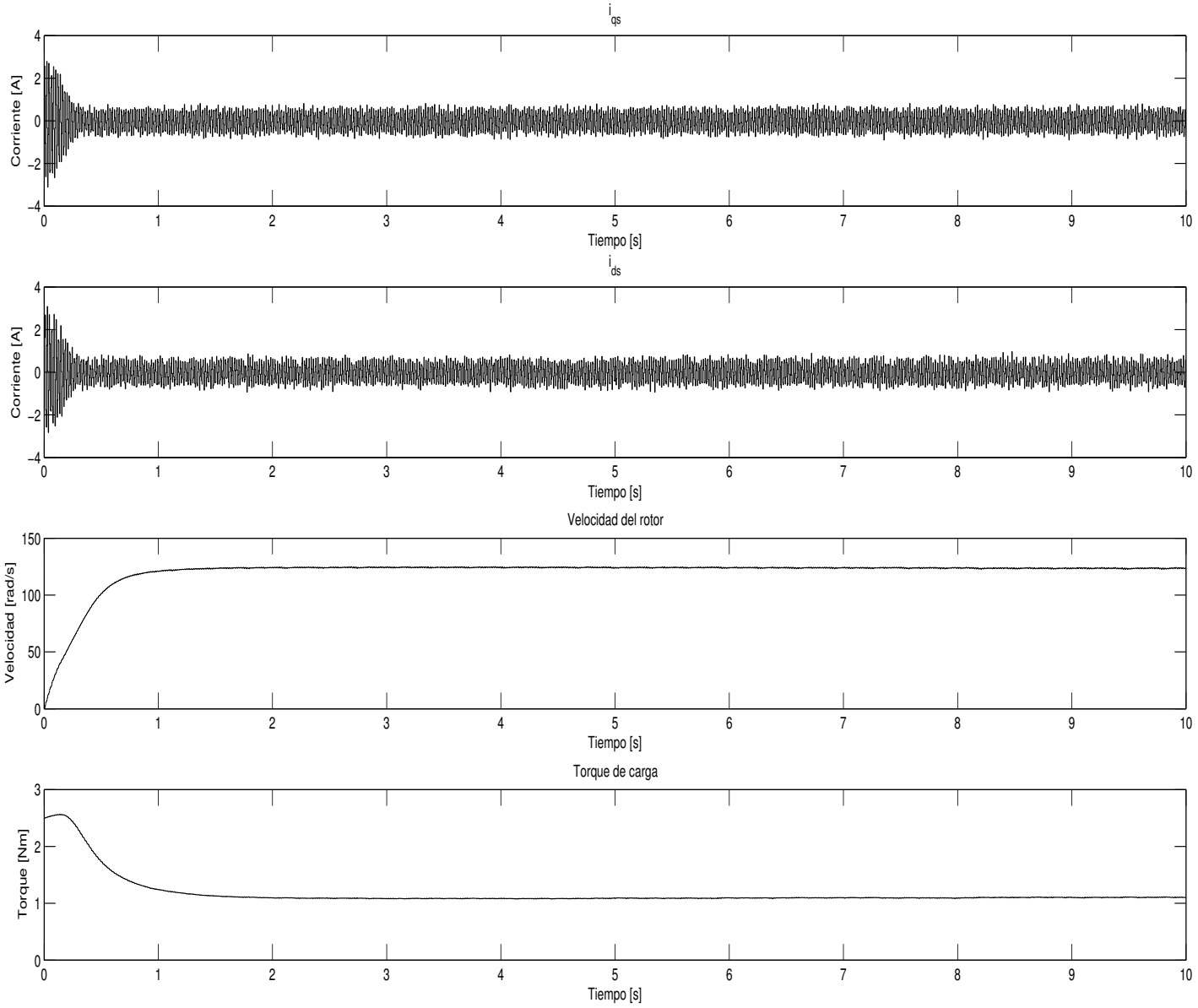


Figura 4.107: Estimación H_∞ a 20Hz en el arranque sin carga (Corrientes, velocidad y torque)

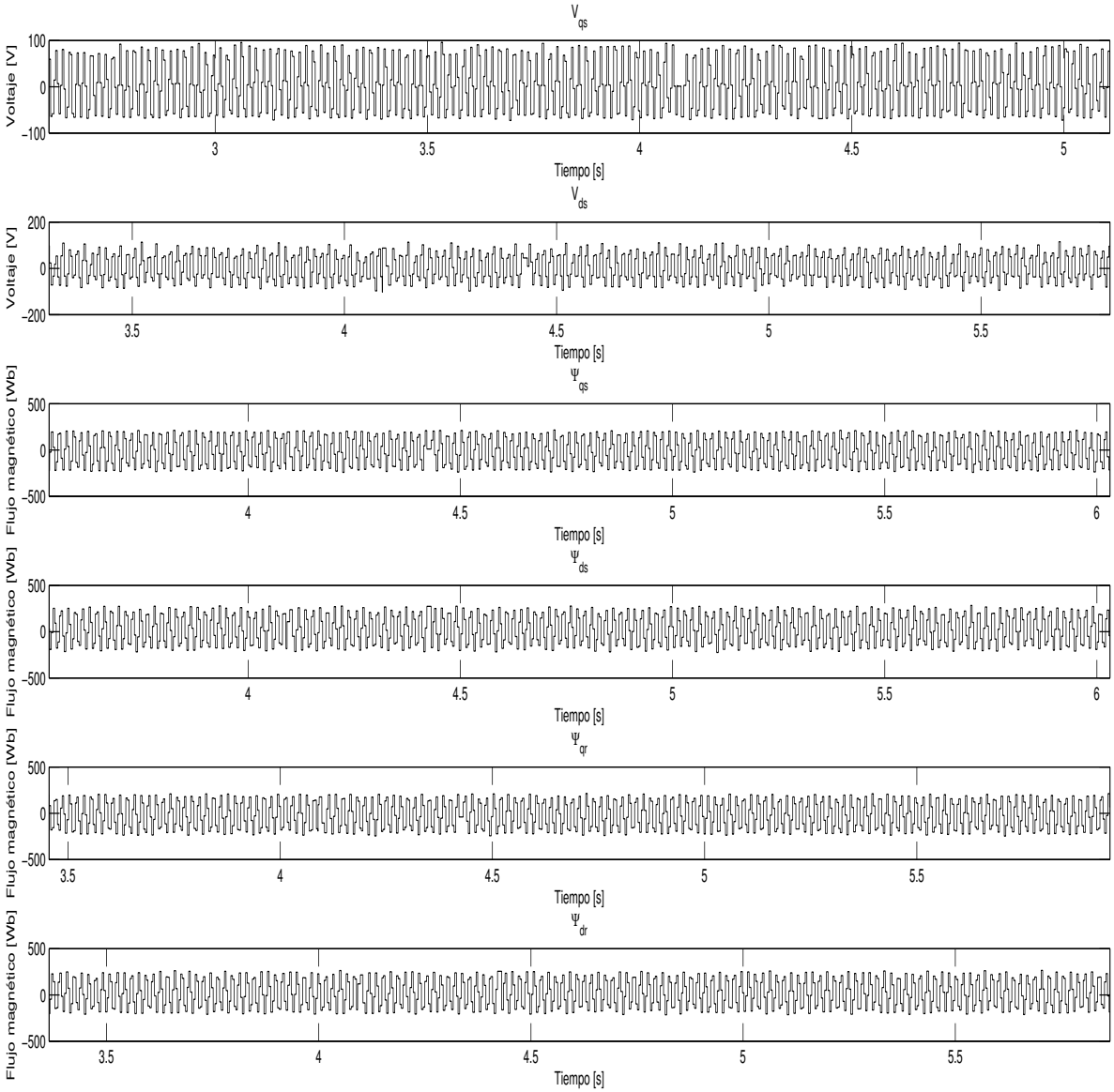


Figura 4.108: Estimación H_∞ a 20Hz en estado estable sin carga (Voltajes y Flujos)

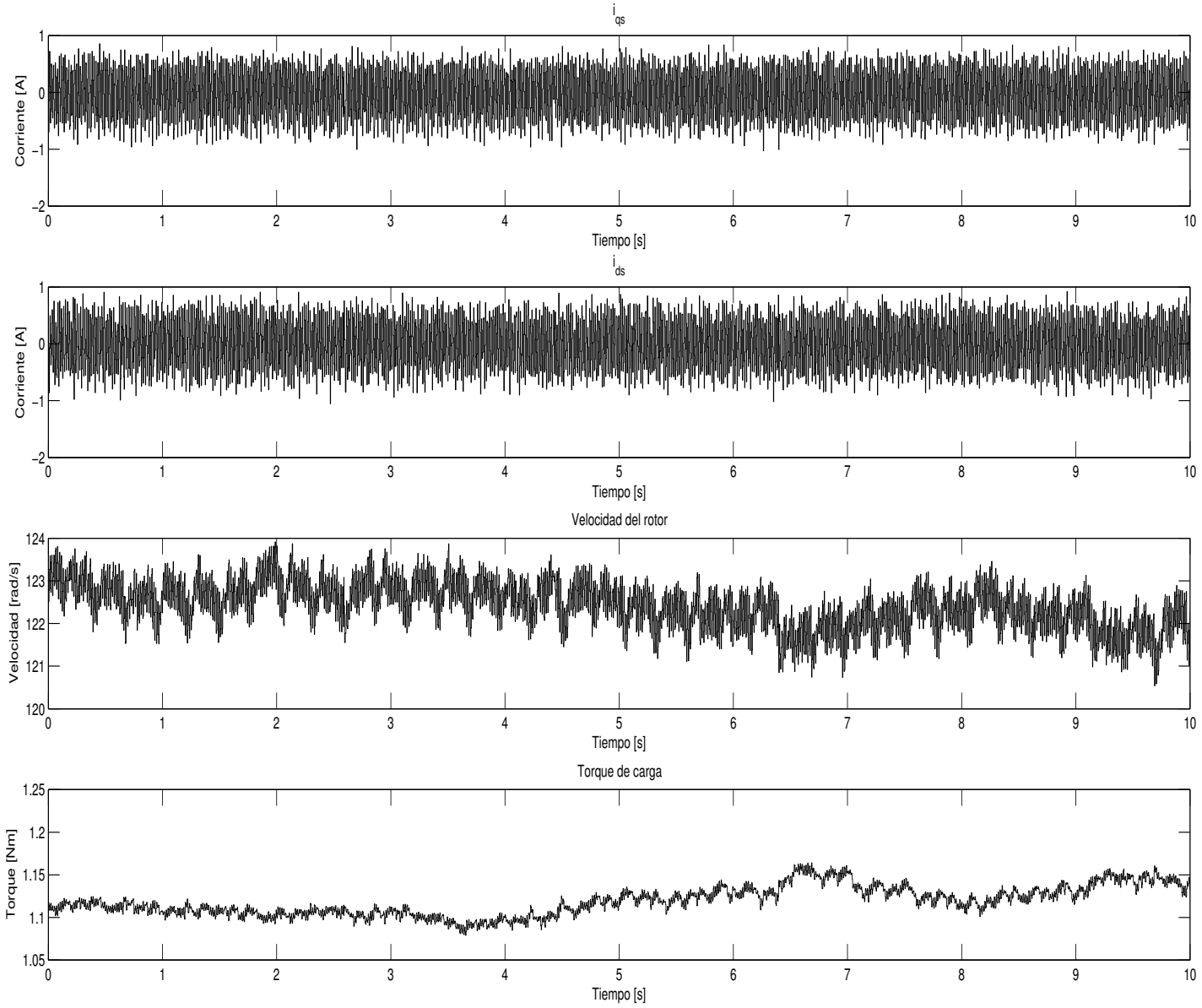


Figura 4.109: Estimación H_∞ a 20Hz en estado estable sin carga (Corrientes, velocidad y torque)

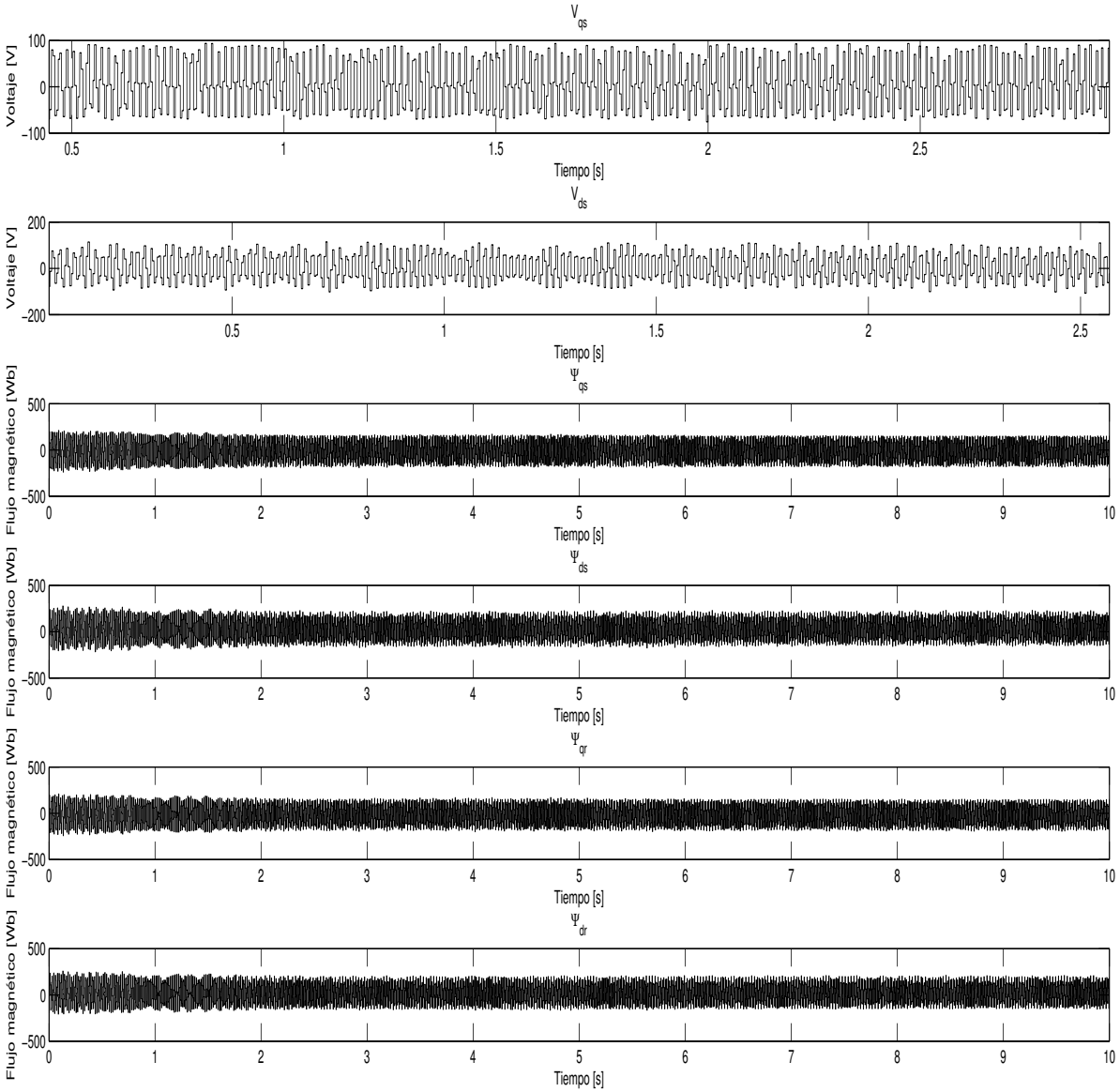


Figura 4.110: Estimación H_∞ con cambio de 20Hz a 25Hz (Voltajes y Flujos)

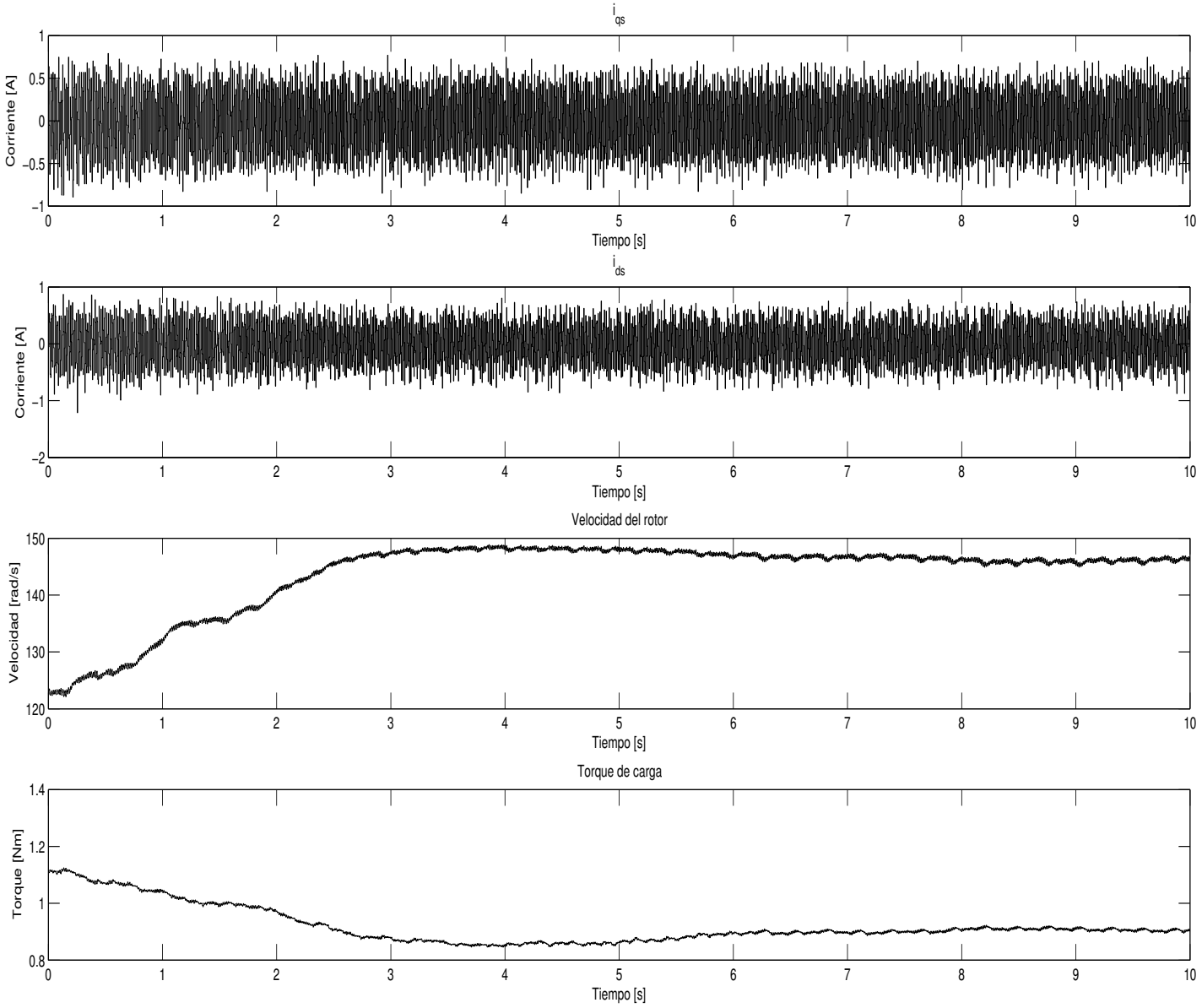


Figura 4.111: Estimación H_∞ con cambio de 20Hz a 25Hz (Corrientes, velocidad y torque)

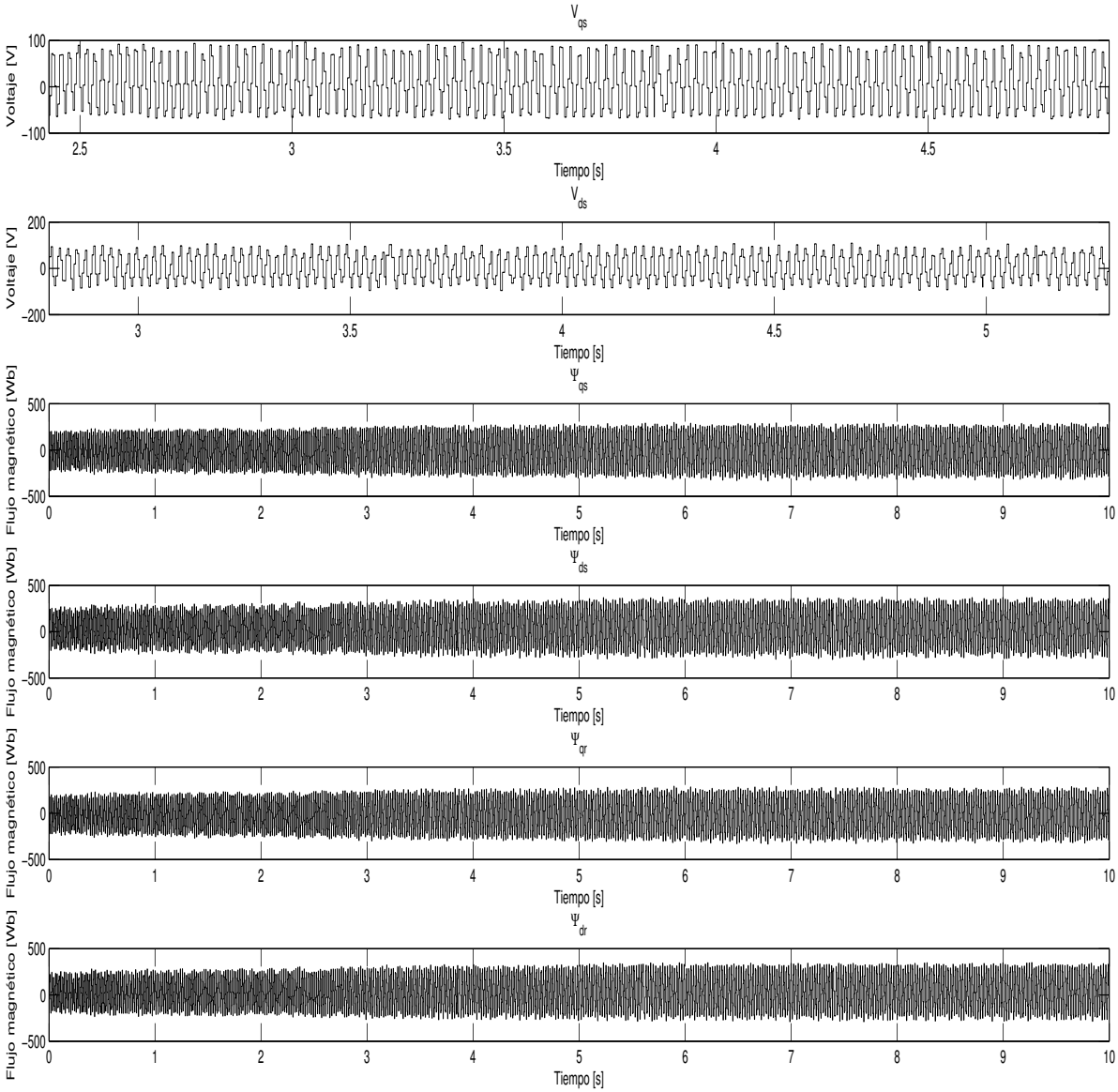


Figura 4.112: Estimación H_∞ con cambio de 20Hz a 15 Hz (Voltajes y Flujos)

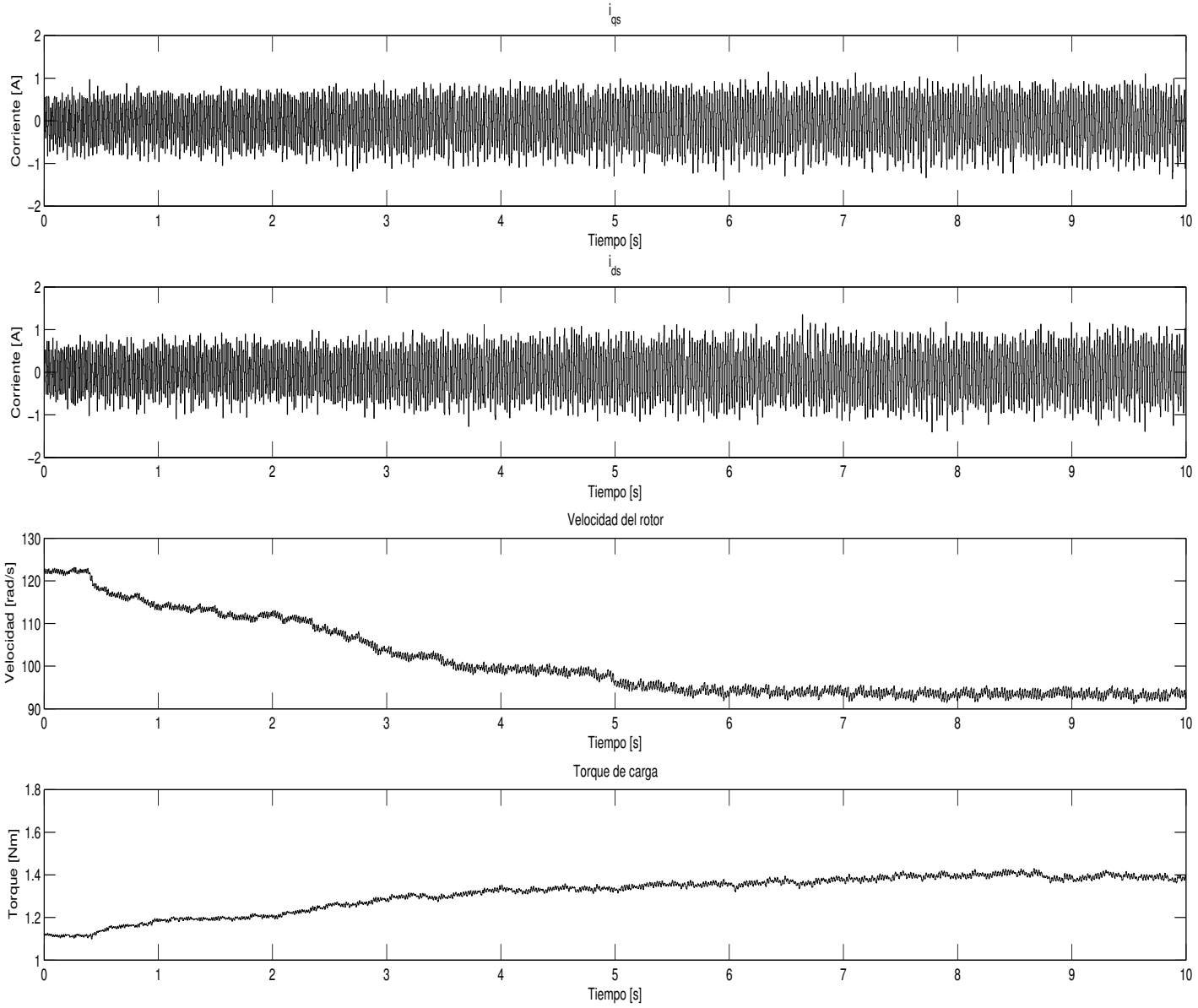


Figura 4.113: Estimación H_∞ con cambio de 20Hz a 15 Hz (Corrientes, velocidad y torque)

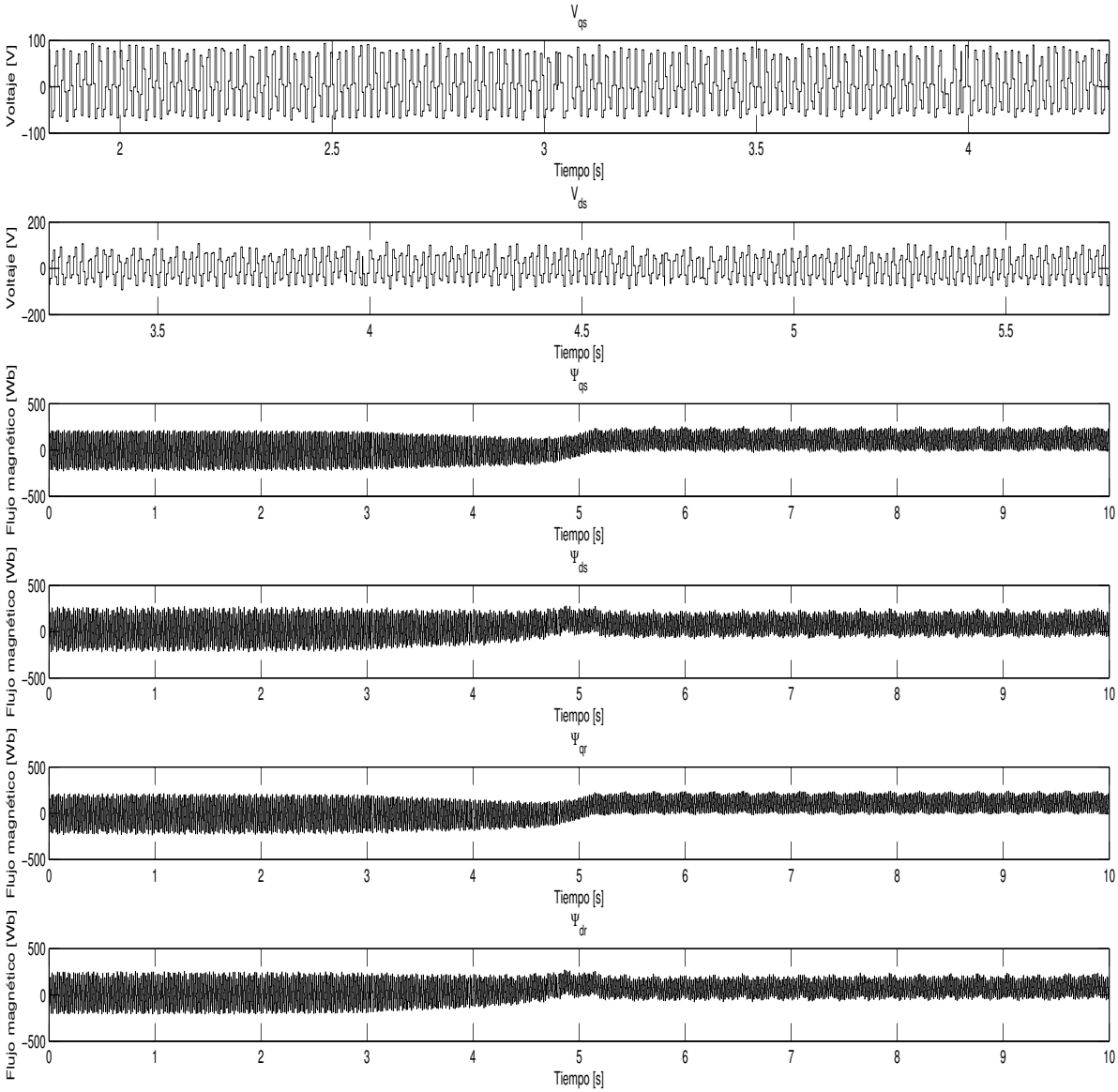


Figura 4.114: Estimación H_∞ con frenado total (Voltajes y Flujos)

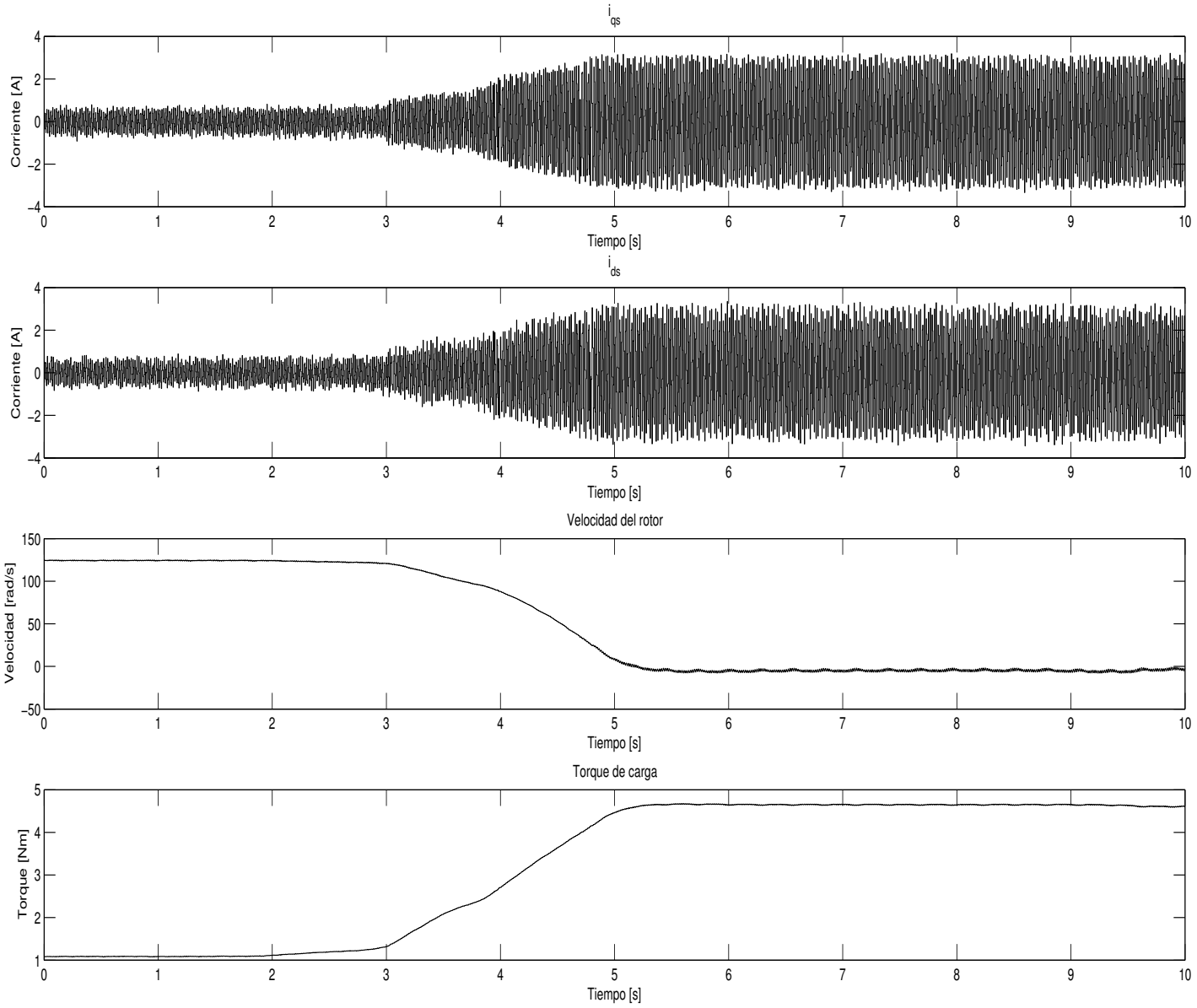


Figura 4.115: Estimación H_∞ con frenado total (Corrientes, velocidad y torque)

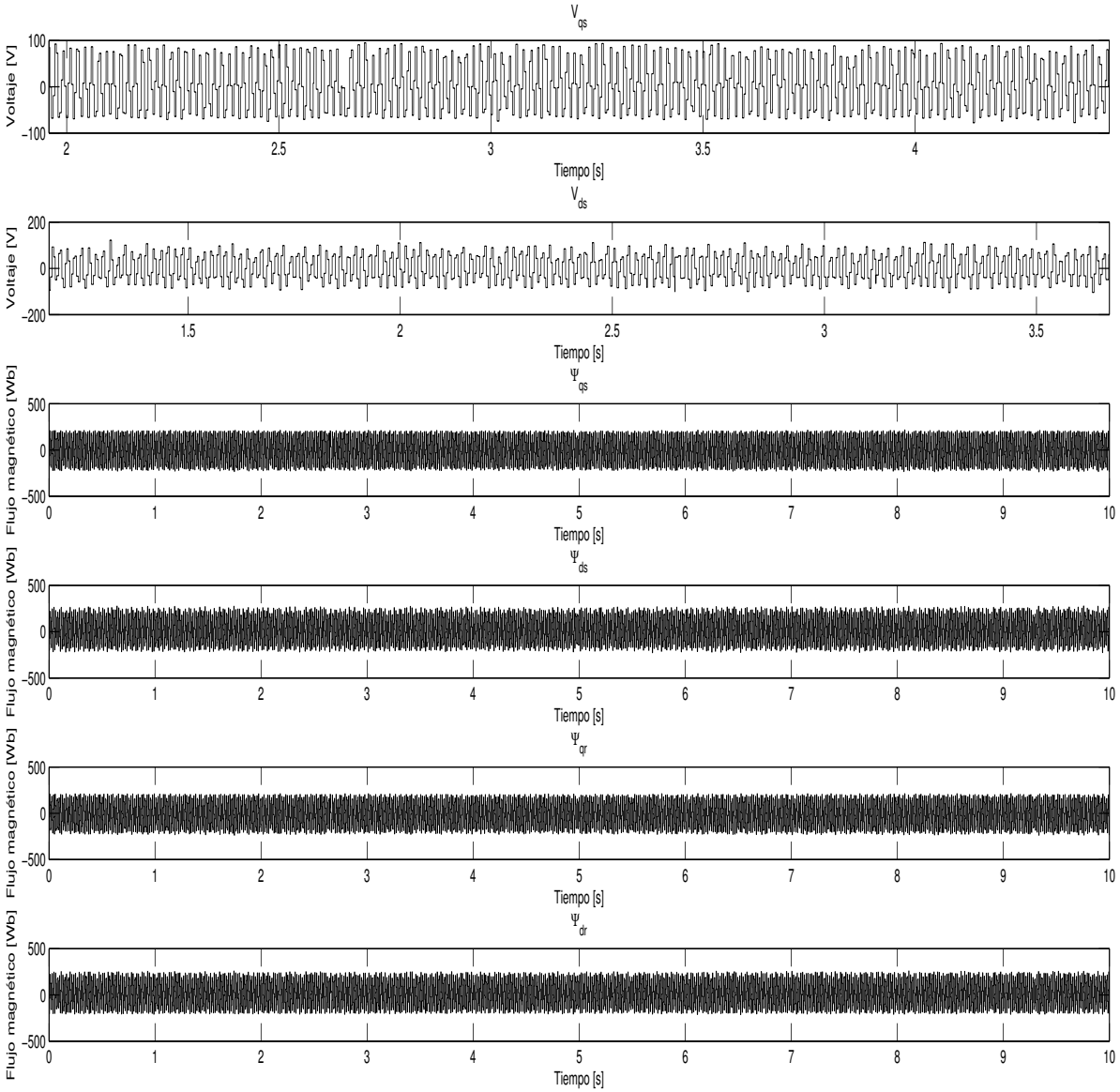


Figura 4.116: Estimación H_∞ con cambios de carga (Voltajes y Flujos)

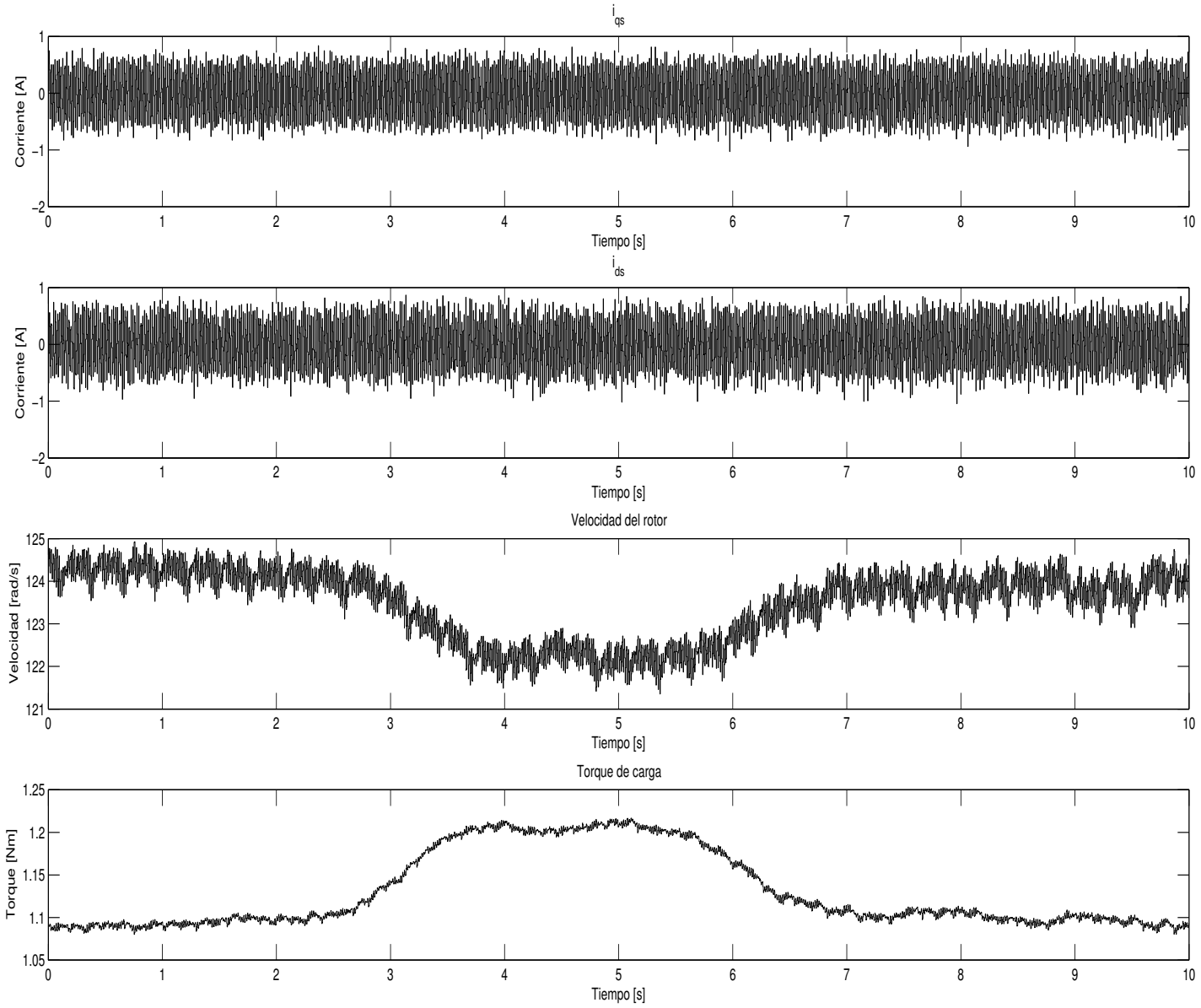


Figura 4.117: Estimación H_∞ con cambios de carga (Corrientes, velocidad y torque)

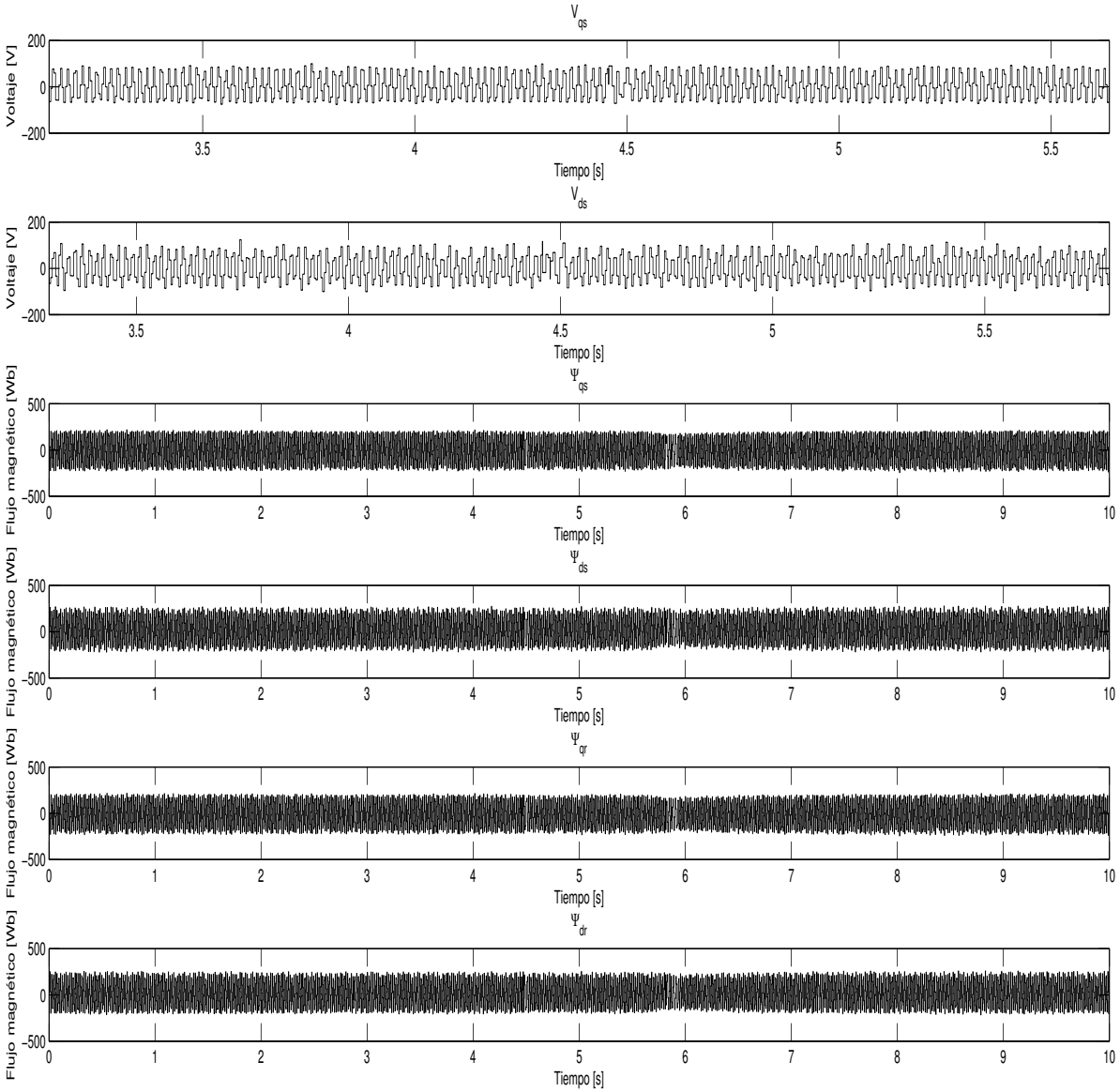


Figura 4.118: Estimación H_∞ con cambios de carga múltiples (Voltajes y Flujos)

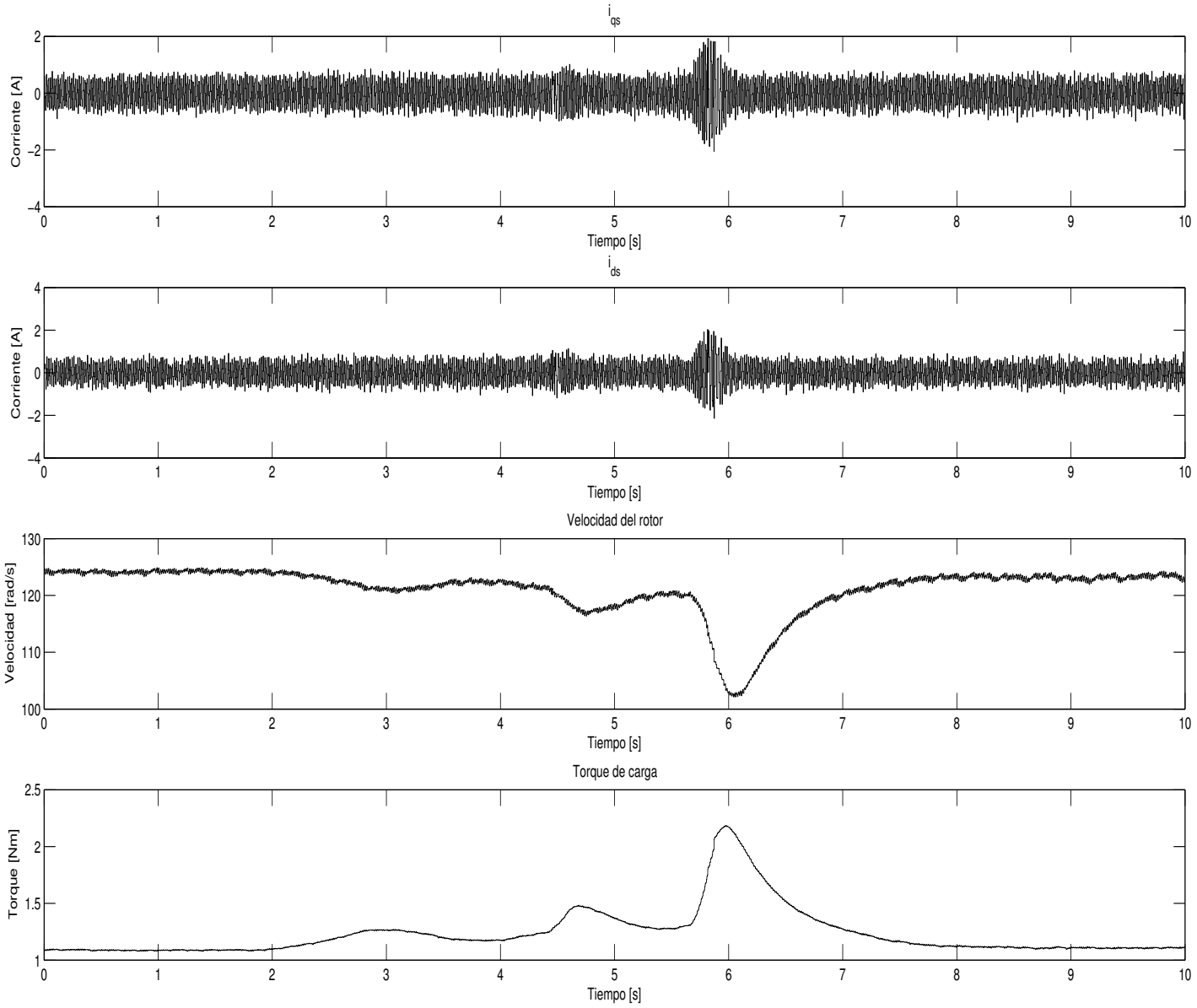


Figura 4.119: Estimación H_∞ con cambios de carga múltiples (Corrientes, velocidad y torque)

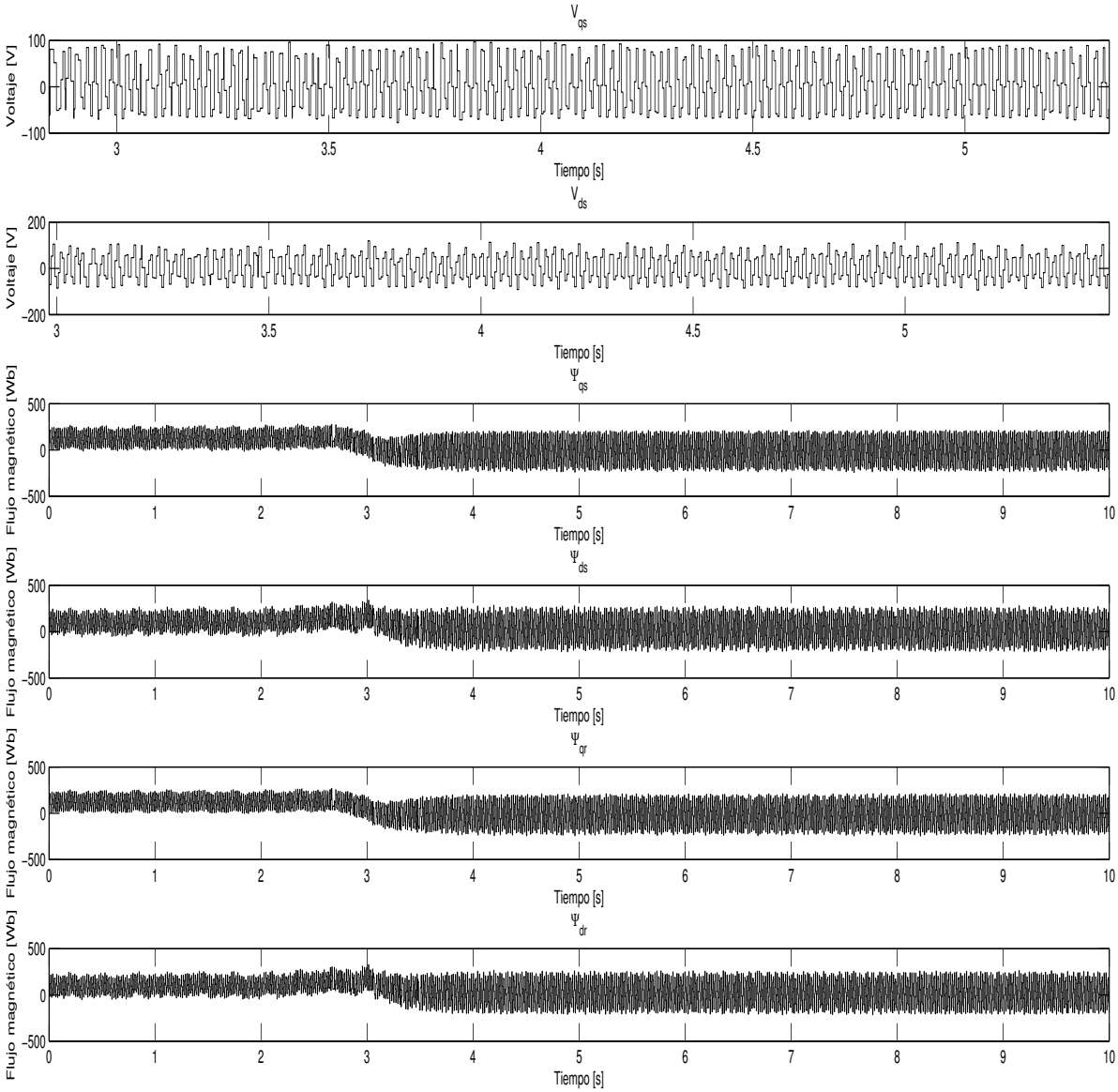


Figura 4.120: Estimación H_∞ con carga total al arranque (Voltajes y Flujos)

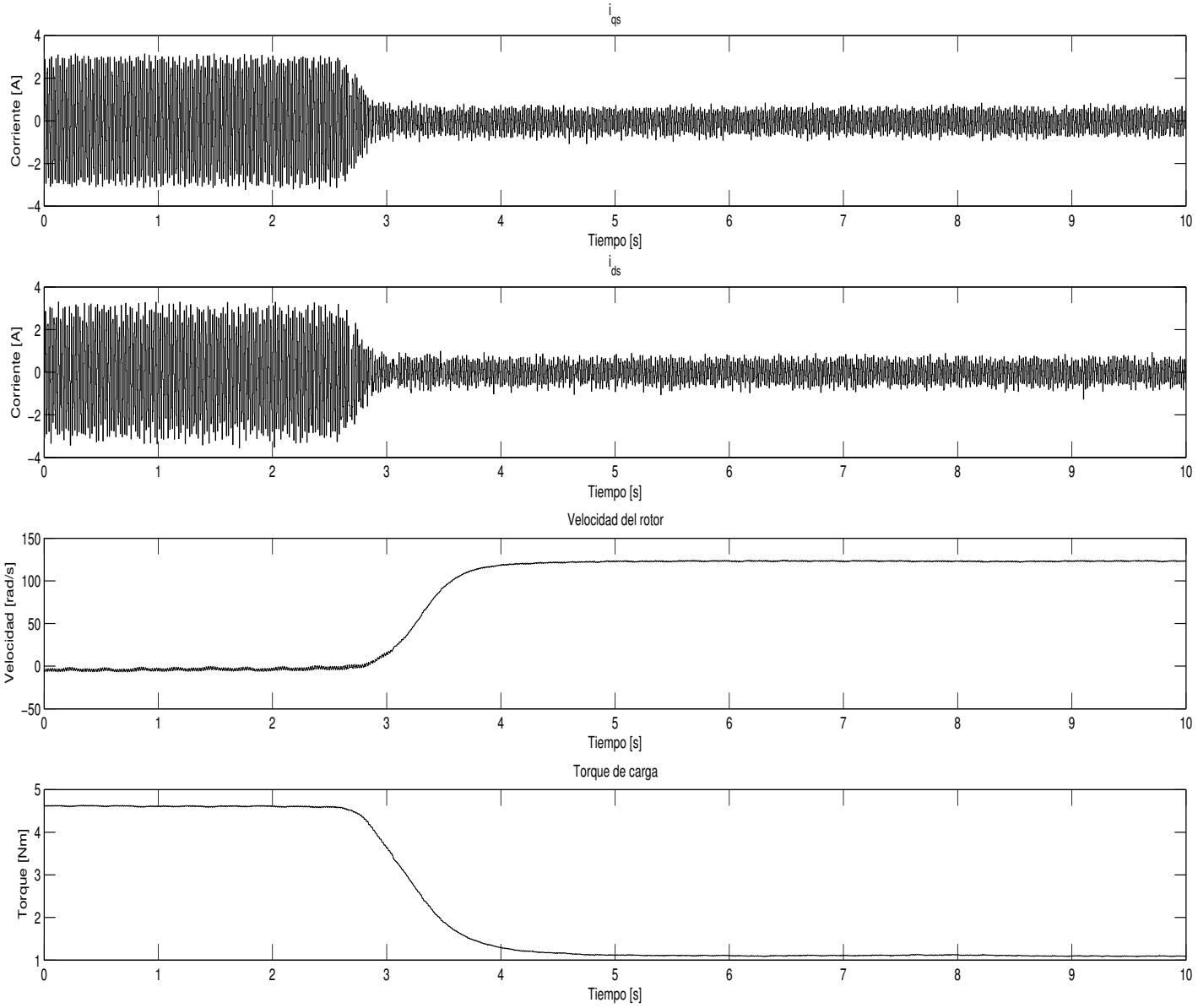


Figura 4.121: Estimación H_∞ con carga total al arranque (Corrientes, velocidad y torque)

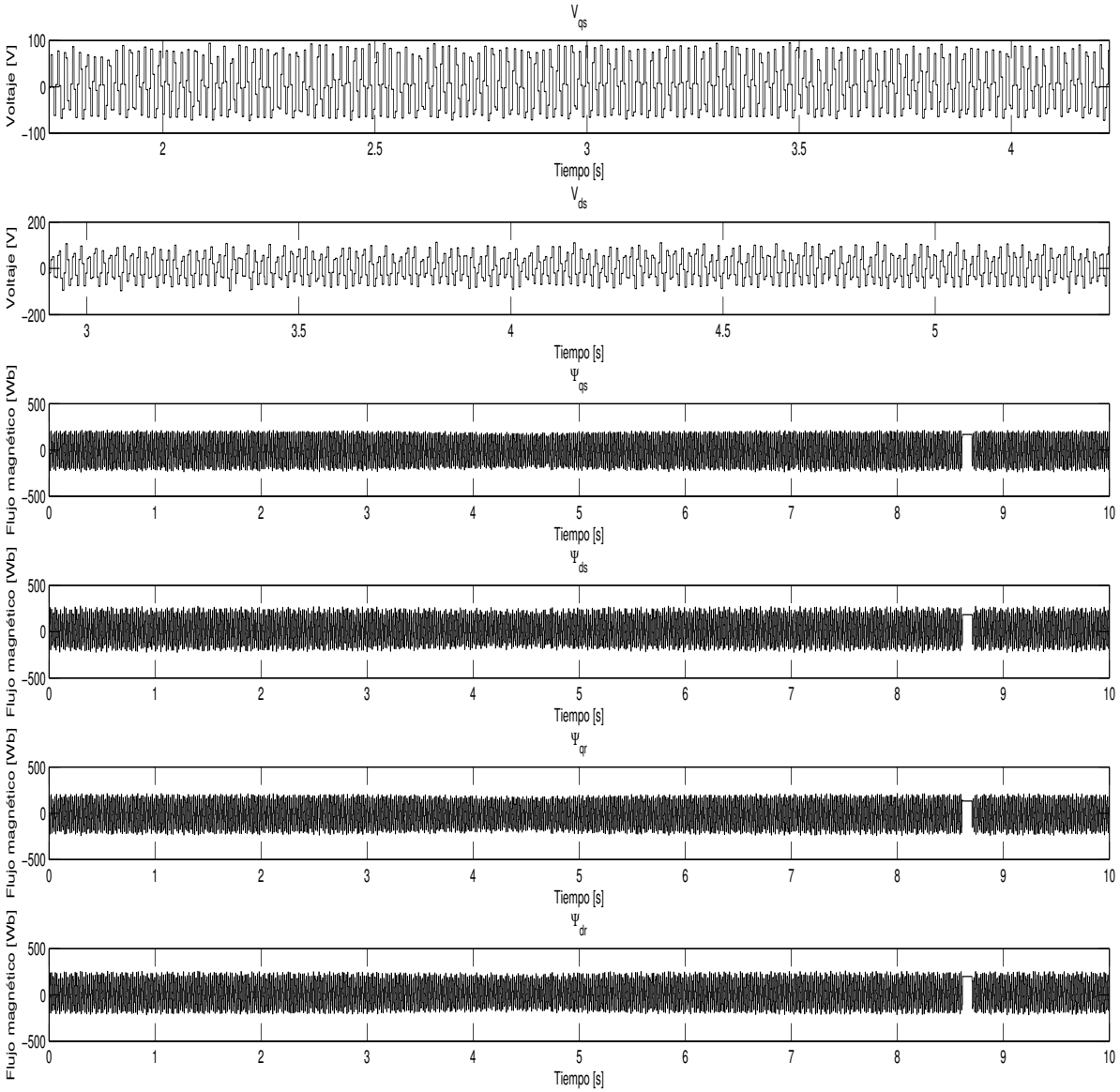


Figura 4.122: Estimación H_∞ con carga y posterior liberación de carga (Voltajes y Flujos)

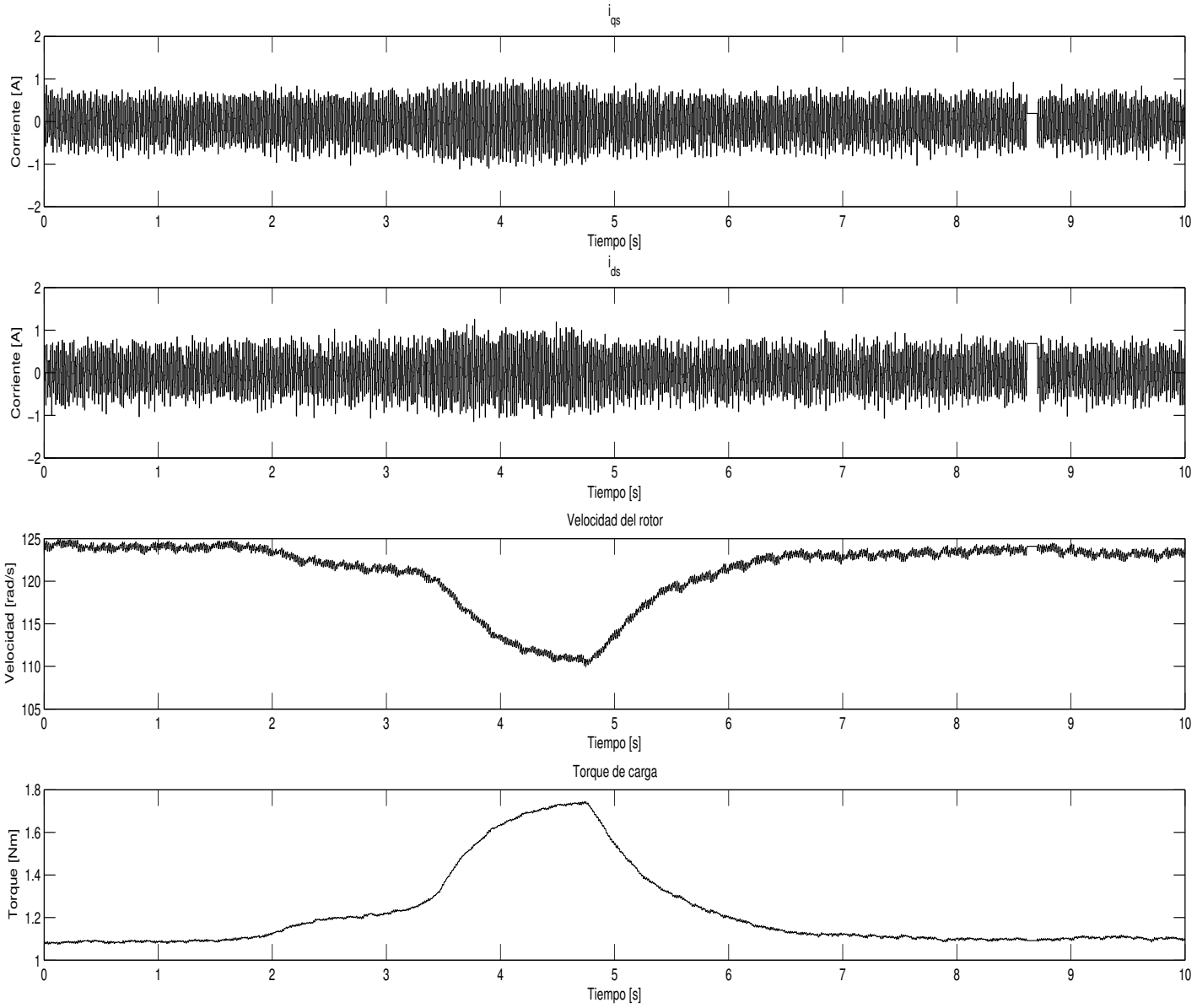


Figura 4.123: Estimación H_∞ con carga y posterior liberación de carga (Corrientes, velocidad y torque)

Las Figuras 4.106 y 4.107 muestra la estimación para el arranque del motor cuando la frecuencia de voltaje de alimentación es 20Hz. La respuesta de la estimación de velocidad se observa más rápida que en los casos anteriores (frecuencia de alimentación de 30 y 50Hz). En el estado estable converge a 122rad/s. No se ven oscilaciones sostenidas durante el estado tran-

sitorio. El torque por otro lado tiene un ligero sobreimpulso antes de converger hacia 1.1Nm. Al igual que en los casos anteriores, la estimación de carga no distingue entre perturbaciones de carga y cambios en la frecuencia de operación. Al operar a baja frecuencia, el motor gira a menor velocidad lo que en el filtro se ve como un efecto de carga que produce desviación a la estimación.

La estimación en estado estable a una frecuencia de alimentación de 20Hz se muestra en las Figuras 4.108 y 4.109. La velocidad en 122rad/s con oscilaciones sostenidas de no más de 2rad/s. La respuesta del filtro en estado estable es sumamente buena, superando ampliamente a la respuesta del filtro Kalman. Por otra parte, la respuesta de torque converge a 1.1Nm. En este caso las oscilaciones me mayor periodo son menores a 0.05Nm mientras que las oscilaciones de alta frecuencia son del orden de 0.005Nm.

La respuesta del filtro cuando ocurre una perturbación en frecuencia que cambia de 20 a 25Hz se muestra en las Figuras 4.110 y 4.111. La velocidad del rotor aumenta de forma gradual y con una dinámica lenta, sin embargo, la respuesta se aprecia más rápida que en los casos analizados previamente. El torque disminuye de su valor de desviación debido a la frecuencia de operación. Esto se debe a que la estimación ve el aumento de frecuencia como una liberación de carga en el motor. En los dos casos no se observan oscilaciones significativas durante el transitorio debido a la perturbación.

La respuesta del filtro cuando ocurre una perturbación en frecuencia que cambia de 20 a 15Hz se muestra en las Figuras 4.112 y 4.113. La velocidad del rotor disminuye según lo esperado pero nuevamente se observa una respuesta lenta a este cambio. La convergencia no tiene oscilaciones sostenidas significativas pero existe una gran descompensación en velocidad de reacción. En el caso del torque se oberva que este aumenta ya que la disminución de la frecuencia provoca una desaceleración en el motor que el filtro estima como si fuese un efecto de carga.

En las Figuras 4.114 y 4.115 se muestra la estimación para el caso en que la carga aplicada frena totalmente al motor. Se observa que el cambio en la velocidad es lento pero por otra parte no existen oscilaciones grandes alrededor de velocidades negativas como ocurría en el caso del filtro Kalman. La convergencia hacia 0 es más directa y también más estable. En el caso del torque de carga, la estimación converge hacia 4Nm (considerando que la desviación por operar a 20Hz es 1.1Nm). Tampoco existen fuertes oscilaciones de esta variable sino más bien una rápida convergencia hacia el estado estable. En estos casos el filtro muestra tener alto rendimiento al superar los efectos adversos observados en la estimación Kalman.

Las Figuras 4.116 y 4.117 muestran la estimación en el caso que se aplican perturbaciones de carga sencillas. Se ve como la velocidad disminuye y luego aumenta debido a los cambios en la carga. La dinámica de la respuesta se observa lenta. Sin embargo, la estimación es consistente con lo esperado. Igual en el caso del torque hay un cambio pero no observan picos u oscilaciones grandes. La respuesta más bien parece ser sobreamotiguada.

Las Figuras 4.118 y 4.119 muestran la respuesta del filtro cuando se aplican una serie de perturbaciones de carga durante un corto periodo de tiempo. Este análisis da una idea de la rapidez con la que responde el filtro en caso de cambios en las condiciones de operación. Se observa como la respuesta del filtro en las variables velocidad y torque resulta lenta ya que la estimación aun no converge y debe responde a una nueva perturbación. Esto se observa

también en la ausencia de sobreimpulsos que muestren rapidez en la respuesta. La estimación en sí reacciona como un sistema sobreamortiguado. Sin embargo, es importante resaltar que no hay oscilaciones sostenidas o rizado significativo durante los transitorios.

En las Figuras 4.120 y 4.121 se observa la estimación para el caso en que el rotor inicialmente esta completamente bloqueado y posteriormente se libera la carga. Se ve una pequeña desviación diferente de 0rad/s en el arranque, pero este es mucho menor y más estable que el observado en el caso del filtro Kalman. La estimación de velocidad se ve rápida en su respuesta hasta converger en 122rad/s. En el caso del torque se observa un comportamiento similar en la medida en la que no se tienen oscilaciones iniciales grandes, la respuesta dinámica es relativamente rápida y no hay oscilaciones importantes en el estado estable.

Inicialmente se tiene carga en el motor, luego se aumenta dicha carga y finalmente se libera el rotor. Este estudio se documenta en las Figuras 4.122 y 4.123. La velocidad disminuye por efecto del aumento de carga hasta que se disminuye la carga y el motor se acelera. Sin embargo, la estimación se ve un tanto lenta y más ruidosa que en las pruebas anteriores. En el caso del torque se tiene un respuesta con menos ruido y oscilaciones sostenidas. Sin embargo, es evidente que estimación tiene una dinámica lenta. Se observa un pico no muy pronunciado y una convergencia en estado estable hacia 1.1Nm tal como se espera de los casos precedentes.

4.4. Comparación del desempeño real de los estimadores.

Una vez realizada una serie de pruebas con perturbaciones de carga y frecuencia a diversas frecuencias de operación se puede comparar el desempeño de los filtros. En alta frecuencia los dos filtros muestran un comportamiento similar pero el filtro Kalman presenta mayores oscilaciones en estado estable. En el caso del filtro H_∞ estas oscilaciones son más pequeñas y durante los transitorios de las diversas perturbaciones aplicadas también se aprecian oscilaciones más pequeñas. Sin embargo, es importante considerar que la respuesta dinámica del filtro Kalman es más rápida que la que se muestra en el filtro H_∞. En el primer caso el filtro se comporta aproximadamente como un sistema de segundo orden subamortiguado, mientras que el segundo filtro asemeja más a un sistema de segundo orden sobreamortiguado. Este comportamiento era esperado por los resultados que se observaron en el capítulo de simulación.

En frecuencias intermedias los dos filtros muestran una limitante. Al operar a 30Hz el motor gira a una menor velocidad. Sin embargo esta baja velocidad es vista por el estimador como si fuera debido a efectos de carga en el motor. Es decir se estima un torque aparente que produce una desviación en todas las demás estimaciones de torque. Al ocurrir perturbaciones en carga, el torque varía sobre esta carga aparente. El torque aparente estimado en el sistema por efectos de frecuencia es de 0.5Nm en el Filtro Kalman y 0.7Nm en el filtro H_∞. Considerando esta observación se procede a la comparación del desempeño de los filtros. En el caso Kalman, se observan nuevamente mayores oscilaciones sostenidas, mientras que el filtro H_∞ tiene oscilaciones menores, que en el peor de los casos alcanzan el 50 % de las observadas en Kalman. Sin embargo, nuevamente se observa que el filtro Kalman tiene una respuesta dinámica más rápida. En el filtro H_∞ no se observan cambios pronunciados sino más bien curvas suaves durante las perturbaciones aplicadas. Una de las pruebas donde mejor debería verse los picos de estimación es en el cambio múltiple de carga. Sin embargo, en el caso H_∞ las curvas son redondeadas y se observa que el sistema todavía no converge a su valor final cuando se aplica la siguiente perturbación. En Kalman, el cambio es más rápido y se observan

curvas con cambios más pronunciados. La respuesta a perturbaciones de frecuencia tiene un comportamiento similar. Cambios muy poco oscilantes pero lentos en el caso H_∞ y cambios rápidos, pero con alto rizado en el caso Kalman. En los dos casos el torque de carga cambia con perturbaciones de frecuencia. Cuando aumenta la frecuencia, el motor gira más rápido y la estimación de torque disminuye como si se hubiera liberado carga. El efecto contrario ocurre cuando la frecuencia disminuye.

La prueba que mayor contraste da en el comportamiento de los dos filtros es la de carga en motor hasta que el rotor se detenga completamente. En el caso Kalman, se observa que la velocidad disminuye rápidamente, pero va por debajo de los 0rad/s y empieza a oscilar considerablemente alrededor de -30 rad/s aproximadamente. Este comportamiento se mantiene por un largo tiempo y la convergencia hacia 0rad/s es demorada. De igual manera el torque empieza a oscilar considerablemente alrededor de 4Nm. La estimación tarda tiempo en atenuar las oscilaciones y estabilizarse alrededor de un valor final. Por otra parte, el filtro H_∞ muestra un descenso de la velocidad hasta 0rad/s más lento que el filtro Kalman pero no se tienen oscilaciones alrededor de un valor negativo. El filtro rápidamente se estabiliza alrededor de 0rad/s. En este sentido se tiene una estimación más robusta ya que filtro no varía su comportamiento en estimación de velocidad debido a los cambios de frecuencia. En el caso Kalman se nota una respuesta totalmente diferente a la prueba de carga total cuando se cambió la frecuencia de operación. El torque estimado por el filtro H_∞ también tiene un comportamiento más robuto al exhibir menos oscilaciones y una convergencia más rápida. En el caso Kalman, las oscilaciones antes de la convergencia final son grandes y el proceso en sí es lento.

A bajas frecuencias de operación se muestra una operación muy similar que la observada en frecuencias intermedias. Nuevamente se observa que la estimación de torque reacciona a cambios de frecuencia como si se trataran de cambios de carga. A mayor frecuencia en la operación la estimación de torque es menor, mientras que al disminuir la frecuencia a la que se opera el torque estimado aumenta. En el caso Kalman el torque aparente por frecuencia es de 1Nm, mientras que en H_∞ el torque aparente es 1.1Nm. Sobre estos valores se producen las variaciones de torque debido a perturbaciones de carga.

Al aplicar perturbaciones de frecuencia al sistema se observa como la estimación de torque cambia como si se tratase de cambios en la carga. A menor frecuencia, el motor gira más lento y el estimador ve una mayor carga. A mayor frecuencia el motor se acelera y el estimador ve una menor carga. En estas pruebas se mantuvo el comportamiento ya expuesto, en el que la estimación Kalman es más rápida pero también presenta mayores oscilaciones durante la operación. La respuesta H_∞ es más lenta pero tiene oscilaciones mucho más pequeñas.

En cuanto a las respuestas a perturbaciones de torque se observa que el filtro H_∞ mejoró un poco en su velocidad de reacción pero sigue estando por debajo de Kalman en ese aspecto. Se mantiene la tendencia de menores oscilaciones sostenidas durante las estimaciones en el caso H_∞. En la prueba de perturbaciones de carga múltiples se observó claramente que el filtro H_∞ no convergía a su valor final luego de una perturbación y debía reaccionar a una nueva. Esto provoca una curva un tanto suave debido donde no se ven picos pronunciados. Por otra parte, el filtro Kalman reaccionaba más rápido y se ven cambios más pronunciados durante las perturbaciones. Incluso el filtro converge a su valor final antes de ocurrir la siguiente perturbación.

Nuevamente una prueba determinante en la respuesta de los filtros es la de carga total hasta detener el rotor. En el caso del filtro Kalman se observa un rápido descenso pero el filtro empieza a estimar velocidades negativas. Se presentan oscilaciones que se atenúan lentamente y no tienden a cero. De igual manera la estimación de torque se muestra oscilante sin tendencia a converger hacia el valor final. En el filtro H_∞ el descenso a 0rad/s es rápido y una vez llegado a este valor se observa una rápida estabilización. Las oscilaciones son pequeñas comparadas con las oscilaciones que presenta el filtro Kalman. La estimación H_∞ se muestra mucho más robusta en este escenario. En baja frecuencia la ventaja de un filtro H_∞ es evidente.

En base al análisis anterior se concluye que a alta frecuencia de operación el rendimiento de los filtros es comparable y no se observa mayor diferenciación. La menor oscilación en el caso H_∞ tiene la limitante de una menor velocidad de reacción. Como se indicó este comportamiento es similar al observado en las simulaciones de este filtro. Por otra parte, a alta frecuencia, el filtro Kalman tiene más oscilaciones y de mayor amplitud, pero su respuesta a perturbaciones es más rápida. Dependiendo de la velocidad del motor que se tenga se podría optar por cualquiera de estos dos métodos. En frecuencias intermedias y bajas frecuencias el escenario es tanto diferente. El filtro H_∞ mejora un poco en su velocidad y mantiene su característica de poca oscilación. De igual manera el filtro Kalman resulta más veloz pero también con mayores oscilaciones. Sin embargo, a baja velocidad y baja frecuencia de operación el filtro Kalman muestra oscilaciones significativas y problemas para converger a un valor final mientras que el filtro H_∞ converge mucho más rápido, sin oscilaciones significativas. Su comportamiento es más robusto en estas condiciones de operación. De esta manera para estimación a frecuencia intermedia o baja frecuencia es recomendable utilizar un filtro H_∞.

En lo referente a la implementación como tal los dos filtros trabajaron a $5kHz$, con un porcentaje de utilización de la capacidad del DSP de 60 % aproximadamente. Sin embargo, debe considerarse que en el caso H_∞ se explotaron las propiedades matemáticas (simetría en las matrices involucradas) del algoritmo de estimación para reducir así el número de cálculos. Por ejemplo la matriz R_w de 4x4 para ser invertida requiere del cálculo de 16 determinantes 3x3. Sin embargo, al ser una matriz simétrica se calcularon únicamente 10 determinates. Esto significa un ahorro en operaciones de aproximadamente 38 %. Similar tratamiento se dió a las demás matrices. Esto se traduce en un ahorro de aproximadamente el 30 % de las operaciones. En base a lo anterior resulta evidente que la carga computacional es al menos un 30 % mayor en el filtro H_∞ en relación al filtro Kalman. Además se debe considerar que el filtro H_∞ se concentró en las variables velocidad del rotor (ω_r) y torque de carga (T_l) para la estimación. En el caso de un filtro concentrado en las 6 variables involucradas la carga computacional era mucho mayor, incluso explotando las propiedades de simetría ya mencionadas.

Los parámetros de sintonización del filtro Kalman son las matrices Q_k y R_k que describen los ruidos de proceso y medición respectivamente. La condición sobre estas matrices es que sean diagonales. En las pruebas realizadas se pudo apreciar que el filtro depende fuertemente de estos parámetros. Una elección incorrecta (en general valores altos para las entradas de estas matrices) hacían que el filtro se desborde numéricamente o que sé no trabaje correctamente. Partiendo de los valores utilizados en simulación se procedió a sintonizar el filtro hasta lograr los mejores resultados. En el filtro H_∞ los parámetros que se pueden sintonizar son G_k y γ siendo el último más un parámetro de rendimiento del sistema (cota superior en la función de costo) que un parámetros sintonizable. Por otra parte G_k debe ser simétrica pero no necesariamente diagonal. En las pruebas realizadas se observó que el filtro prácticamente no varía al cambiar

estos parámetros. Se ensayaron diversos valores. El único problema observado fue que para valores $\gamma > 0{,}8$ existían problemas numéricos por desbordamiento en la resolución del DSP pero no problemas asociados a la estimación como tal. Esto da una idea que el filtro H_∞ es más robusto y adaptado más naturalmente al sistema. En el caso Kalman, al tener que sintonizar parámetros se tiene la impresión que se adapta el filtro al sistema pero este no lo hace naturalmente.

Otro aspecto a tener en cuenta es la estabilidad numérica del algoritmo. En el caso Kalman se utilizaron 32 bits de resolución con 14 enteros. Esta configuración fue sufiente para manejar los picos grandes que se producen durante la estimación en las variables y además tener el suficiente número de decimales para el manejo de las operaciones en el algoritmo como tal, es decir las operaciones matriciales. El algoritmo no tiene números muy grandes ni muy pequeños. Por otra parte, en el filtro H_∞ se utilizó 32 bits de resolución con 12 enteros. Durante las pruebas con 14 enteros se notó que algunos de los números involucrados en el algoritmo, especialmente en la inversión de la matriz R_w eran demasiado pequeños y el DSP los consideraba como 0, afectando al correcto desarrollo de la estimación. Debido a este problema se optó por cambiar a un formato de 12 enteros. Esto fue posible porque al no tener picos tan pronunciados como en el filtro Kalman, no se requerían 14 enteros sin tener desborde numérico. Es indica que el algoritmo H_∞ es más sensible numéricamente y requiere un mejor tratamiento ya que errores en estimación pueden no ser producto del algoritmo en sí sino debido a fallas numéricas debido a las características del dispositivo utilizado.

Una visión global de los filtros Kalman y H_∞ muestran que la robustez que muestra el segundo sobretodo a bajas frecuencias de operación requiere de una contraparte tanto numérica como en velocidad de reacción. El filtro H_∞ luce más lento en reaccionar a perturbaciones, pero lo hace de mejor manera, con menos oscilaciones o rizado incluso en situaciones críticas. Así también requiere mayor carga computacional, aproximadamente 30 % más. En el aspecto numérico, las operaciones que se realizan involucran números pequeños que merecen especial atención. Por otra parte, en la implementación de campo se notó que el rendimiento del filtro H_∞ prácticamente no varía con los parámetros G_k y γ que se elijan, mientras que el filtro Kalman requirió de diversas pruebas de sintonización antes que la estimación sea la mejor. Se observa nuevamente la robustez del filtro H_∞.

5. Conclusiones y Recomendaciones

5.1. Conclusiones

- Modelo de la máquina de inducción.

El modelo utilizado para la máquina de inducción se basó en el propuesto por Krause [2], donde las variables de estado son los flujos magnéticos en el estator y rotor de la máquina. Todo esto en coordenadas $dq0$ ya que al ser (o asumirse) un sistema trifásico balanceado la componente 0 de las variables $qd0$ del sistema es nula y se puede reducir a un sistema bifásico en lugar del sistema trifásico original. Así también la transformación de coordenadas permite tener inductancias independientes de la posición relativa entre estator y rotor y de la velocidad a la que gira el rotor de la máquina. De esta manera se logra un modelo de la máquina de inducción sencillo. Este modelo fue discretizado y adaptado a las necesidades de estimación del presente problema. Sin lugar a dudas, la elección adecuada del modelo permite tener un algoritmo de estimación sólido pero a su vez simplificado en el número de variables que deben calcularse en cada iteración. De igual manera en la simulación y en la implementación se observa que el modelo elegido describe de forma muy aproximada la dinámica real del sistema estudiado.

- Simulación del filtro Kalman extendido y filtro H_∞ extendido.

A partir del modelo del motor ya descrito, su versión en tiempo dicreto y los algoritmos de estimación tratados a lo largo de este trabajo se realizaron varias simulaciones con el objetivo de validar los algoritmos y modelos utilizados y comparar diferentes variaciones del modelo discreto del motor o del filtro de cara a la implementación. En el caso del filtro Kalman extendido se simularon situaciones con perturbación externa (torque de carga) y sin perturbación externa (torque de carga) considerando una aproximación lineal y cuadrática en T_s. Los resultados de simulación mostraron que no existe una diferencia significativa entre los dos modelos por lo que se escoge el modelo lineal por su menor complejidad en programación y carga computacional. En este caso las variables estimadas siguen el comportamiento esperado, sobretodo el torque de carga, ya que al ser la variable cuyo modelamiento en el sistema era más rudimentario (se la consideró constante) se esperaba sea un factor crítico a la hora de evaluar el desempeño del filtro. La respuesta fue favorable con una estimación muy buena y estable.

En simulación también se pudo observar el efecto de las matrices diagonales Q_k y R_k en el algoritmo. Una elección acertada de estas variables mejora considerablemente la estimación. Así también el filtro dependía fuertemente de la matriz de covarianza inicial P_0 (para valores iniciales altos de esta variable el sistema convergía rápidamente al valor final pero la estimación era de mala calidad en los transitorios) y en menor medida de los estados iniciales x_0. Si bien los parámetros Q_k, R_k y P_0 pueden ser vistos como parámetros de diseño flexibles que permiten adaptar el filtro a una aplicación específica no dejan de ser una limitante ya en situaciones reales. Como se observa, la respuesta del filtro cambia con estos parámetros y si la dinámica del sistema real cambia en el tiempo o los parámetros del modelo varían, el filtro no tiene la misma respuesta. Esto hace que el filtro Kalman no sea robusto. Sin embargo, una buena sintonización del filtro en general da bueno resultados.

En lo referente a la simulación del filtro H_∞ extendido se consideraron dos casos. En el uno se tomó la matriz $L_k = I_{6\times 6}$ y en el otro $L_k = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$. En vista de los resultados obtenidos con la simulación del filtro Kalman resultaba poco ventajoso comparar nuevamente el modelo lineal con el modelo cuadrático. Por esta razón se opta por comparar la respuesta en base a la elección de la matriz L_k. El objetivo final era ver si el rendimiento del filtro cambiaba substancialmente ya que la elección de $L_k = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ resultaba en un algoritmo mucho más ligero computacionalmente hablando. Tras observar los resultados se nota que no existen mayores diferencias entre los modelos por lo que se descarta utilizar $L_k = I_{6\times 6}$ en la implementación. La calidad de las estimaciones es muy buena, casi sin mayor diferenciación entre las señales reales y las calculadas incluso en presencia de perturbaciones. De igual manera el torque de carga (variable estimada crítica) mostraba una buena dinámica, aunque se apreciaron también oscilaciones pequeñas alrededor del punto de equilibrio.

En el caso del filtro H_∞ extendido se tienen solo dos parámetros libres de diseño, la matriz G_k que describe el ruido en el sistema y el parámetro γ que representa una cota para el rendimiento del filtro. Sin embargo, la matriz G_k es mucho más flexible que las matrices Q_k y R_k al no tener que satisfacer la condición de diagonalidad por lo tanto su modelamiento puede ser más cercano a la realidad de la planta. En las pruebas de simulación y sintonización del filtro H_∞ extendido prácticamente no se observaron variaciones considerables al variar estos parámetros. Tampoco existe dependencia fuerte con la matriz de covarianza inicial P_0 o el vector de estados inicial x_0. A diferencia del filtro Kalman, este filtro se presentaba robusto a los parámetros mencionados. Sin embargo, se debe decir que para valores mayores a 1 de γ el filtro tenía problemas númericos que hacían que el algoritmo falle por mal condicionamiento de las matrices necesarias para la estimación.

- Implementación del filtro Kalman extendido y filtro H_∞ extendido.

Como se menciona en la sección correspondiente a este literal, la implementación fue un punto crítico de este proyecto. No sólo porque de una buena implementación depende el funcionamiento de la estimación sino también por la gran cantidad de trabajo que significa. Debido a la programación en punto fijo de los dos algoritmos fue necesario programar toda las operaciones matriciales como operaciones escalares entrada por entrada. Esto no solo significa mayor tamaño en el programa sino también más complejidad a la hora de programar. Se debe tener sumo cuidado al momento de generar cada uno de los bloques de programación. No obstante, mediante la reducción de operaciones a problemas escalares fue posible alcanzar una frecuencia de trabajo alta ($5kHz$), la misma que se utilizó en simulación, con un porcentaje de utilización del DSP relativamente bajo, esto es 60 % aproximadamente. Si se tiene en cuenta que las primeras pruebas se realizaron a 250Hz con 95 % de utilización cuando se programaron los filtros con la librería matricial de VisSim la mejora es sustancial.

En cuanto al hardware de implementación, más allá de los problemas iniciales que se tuvieron con la placa de control debido a configuraciones de fábrica o problemas de ruido no existen mayor inconveniente. El hardware de Francecol trabajó según las especificaciones y la placa adicional que se diseño para acondicionamiento de la señal del voltaje tampoco presentó inconvenientes.

- Desempeño en tiempo real del filtro Kalman extendido y filtro H_∞ extendido.

En alimentación de voltaje a alta frecuencia, el desempeño del filtro Kalman y el filtro H_∞ es comparable, por lo que considerando la mayor carga computacional y menor velocidad de respuesta a perturbaciones del segundo se debe optar por un filtro Kalman. Sin embargo, en operaciones de frecuencia media y baja frecuencia el filtro H_∞ muestra una respuesta mucho más estable, menos oscilante, mejora su velocidad y se aprecia de mejor manera la robustez del algoritmo. Las ventajas anteriores son suficientes para justificar incluso la carga computacional extra. En los dos casos el porcentaje de ocupación del DSP fue de aproximadamente 60 %, pero considerando que el código del filtro H_∞ está optimizado en al menos 30 %. De no mediar este procedimiento posiblemente no hubiera sido posible operar a $5kHz$.

- Acerca de los dos modelos

El filtro H_∞ definitivamente es la mejor opción si se va a trabajar en un amplio rango de frecuencia y bajo diversas condiciones de operación, pero debe tener en cuenta que su implementación es más compleja y debe ser estudiada con más deteminiento que la implementación del filtro Kalman. En este trabajo se aprovechó todas las propiedades matemáticas del algoritmo y además se centró el desempeño de estimación únicamente en las variables velocidad del rotor y torque de carga. Caso contrario el algoritmo resultaba ser sumamente pesado y prácticamente imposible de implementar en el DSP F2808 disponible.

En condiciones normales de operación, sin perturbaciones grandes el filtro Kalman sigue siendo una gran opción tanto por su desempeño como por su menor complejidad teórica y computacional. Sin embargo, la elección del filtro siempre queda determinada también por el hardware disponible para implementación. En el caso presente, el DSP F2808 tiene un alto desempeño, pero si se quiere mejor el algoritmo o probar con un filtro H_∞ centrado en la estimación de las seis variables de estado se debería pensar en cambiar de tecnología hacia un FPGA o un DSP de mayor velocidad y memoria.

En lo referente al modelo teórico de los filtros se debe explorar modelos que incluyan incertidumbre en los parámetros como los propuestos por [20, 21] pero aplicado al caso de modelos no lineales como el de la máquina de inducción. Estos modelos pueden ser estudiados y probados con el objetivo de determinar si la complejidad extra que se requiere tanto en teoría como en implementación, recursos y programación se justifican mediante una mejora significativa en los resultados de la estimación.

Con los filtros probados durante este trabajo se abre la puerta para una serie de trabajos ya sean complementarios o posteriores. Entre los posibles proyectos que complementen a éste se pueden mencionar un estudio más profundo para determinar los parámetros eléctricos y mecánicos del motor, el incluir incertidumbre en el modelo simulado del motor para estudiar la respuesta de los filtros en esos casos y el modificar el modelo en espacio de estados para diferenciar perturbaciones de carga y frecuencia en la estimación de torque de carga. En cuanto a los trabajos posteriores se debe mencionar sin duda el utilizar los filtros desarrollados para avanzar hacia el control sensorless de la máquina de inducción. Este trabajo se centró enteramente en desarrollar estimadores pero deja la puerta abierta para que las estimaciones desarrolladas sean utilizadas para implementar técnicas de control vectorial de motores.

5.2. Recomendaciones

- Modelo de la máquina de inducción.

Para futuros trabajos se recomendienda realizar un estudio que permita incluir efectos no lineales en el modelo, por ejemplo la saturación del núcleo magnético por corrientes muy altas a voltaje de baja frecuencia, modelamiento de la variación de los parámetros internos de la máquina, en especial las resistencias ya que éstas cambian con temperatura debido a la operación misma del motor. Así también se debería poner mayor énfasis en las pérdidas mecánicas del sistema, ya que los rozamientos, deslizamientos y amortiguamientos en la máquina afectan su funcionamiento y pueden provocar diferencias significativas entre las variables reales y las variables estimadas. Finalmente, es recomendable hacer un estudio más profundo de la determinación experimental de los parámetros del motor. En este caso se utilizó el método propuesto por [1], pero en futuros estudios se puede considerar otros métodos o perfeccionar el utilizado.

- Simulación del filtro Kalman extendido y filtro H_∞ extendido.

Para futuros estudios se recomienda adaptar el modelo del motor a un modelo de parámetros variables, es decir con incertidumbre. De esta manera se puede observar la respuesta de los dos filtros en caso de cambios en los parámetros. Además, es recomendable probar con distintas fuentes de ruido en la simulación. En el caso presente se simuló con fuentes de ruido gaussiano. Sin embargo, esta condición es necesaria solo en el filtro Kalman. Hubiese sido importante probar la respuesta del filtro Kalman con otro tipo de ruido en simulación y de paso verificar la robustez del filtro H_∞ en dichos casos. Igualmente en el algoritmo del filtro H_∞ se pueden hacer las modificaciones propuestas por [20, 21] para tener en cuenta incluso la incertidumbre en los parámetros. No obstante, no se debe complicar demasiado el algoritmo hasta aumentar considerablemente su carga computacional.

- Implementación del filtro Kalman extendido y filtro H_∞ extendido.

Se sugiere en futuros estudios trabajar en variables por unidad para la estimación. De esta manera es posible utilizar la configuración óptima de bits de precisión que sugiere Texas Instruments [25], 32 bits de precisión y 8 enteros. Los filtros del presente trabajo no se plantearon en variables por unidad por lo que los 8 enteros (± 128 en representación numérica) no eran suficientes para el valor numérico de las variables involucradas. Se requirió trabajar en 12 enteros en el caso del filtro H_∞ y 14 enteros en el caso del filtro Kalman. Adicionalmente, se sugiere para próximos trabajos involucrarse directamente con el programa Code Composer Studio y evitar el paso intermedio por VisSim. Esto debido a que una vez familiarizado con la interfaz de Code Composer Studio se tendría mayor flexibilidad a la hora de programar y generar subrutinas. Principalmente se debe considerar que algunas técnicas de programación como lazos o bucles no resulta evidente en la interfaz VisSim.

- Desempeño en tiempo real del filtro Kalman extendido y filtro H_∞ extendido.

Una vez más se hace recomendaciones en lo referente a utilizar variables por unidad. Durante las pruebas se tuvieron problemas de falta de resolución por lo que fue necesario cambiar de 14 enteros a un formato con 12 enteros. Esto se puede evitar si se trabaja como sugiere Texas Instruments [25] con 8 bits de enteros y 32 de resolución. El formato sugerido de Texas no se pudo aplicar en el modelo implementado porque habían variables que se desbordaban irremediablemente. Otra recomendación es ampliar el estudio del filtro H_∞ y explorar las razones por las que luce más lento en su respuesta dinámica respecto del filtro Kalman. Se sugiere un estudio más profundo del modelo matemático y algoritmo de estimación utilizado.

Uno de los aspectos más particulares del desempeño real fue la respuesta de los dos filtros a los cambios de frecuencia. En dichas perturbaciones la estimación de torque reaccionó como si se tratara de una perturbación de carga. Al aumentar la frecuencia el motor acelera y el torque estimado disminuye. Al disminuir la frecuencia el motor se frena y el torque estimado aumenta. Al trabajar a una frecuencia intermedia (30Hz) o baja (20Hz) se observa que existe un torque base o de desviación en la estimación. Con frecuencia intermedia éste es de aproximadamente 0.5Nm y en baja frecuencia es de 1Nm. Sobre esta base se aprecian los cambios de torque cuando ocurren las pertubaciones de prueba, ya sean éstas de carga o frecuencia. Se recomienda profundizar en el desarrollo del modelo del motor en espacio de estado para modelar de mejor manera el torque de carga (no como una perturbación constante en el tiempo) y así lograr diferenciación entre perturbaciones de frecuencia y carga. En el modelo actual se observa que la estimación de carga esta ligada fuertemente a la velocidad del rotor. La relación es inversamente proporcional.

En la prueba de bloqueo total del rotor el filtro Kalman estimó un torque máximo de 2Nm mientras que el filtro H_∞ estimó un máximo de 3Nm. Se recomienda medir mecánicamente (o mediante un sensor) el torque aplicado para así determinar cual de los dos valores es el correcto. De esta manera se puede tener otro elemento de comparación del desempeño del filtro. Así también se puede profundizar en el estudio de "torques aparentes", es decir los torques de carga que se estiman por efectos de frecuencia.

Referencias

[1] Fitzgerald A, Kingsley C. & Kusko A. "Máquinas Eléctricas" McGraw Hill.

[2] Krause Paul, Wasynczuk Oleg & Sudhoff Scott. "Analysis of Electric Machinery and Drive Systems. IEEE Press. 2002.

[3] Rigatos Gerasimos, Siano Pierluigi. "Sensorless Control of Electric Motors with Kalman Filters: Applications to Robotic and Industrial Systems". International Journal of Advance Robotic Systems. 2011.

[4] IEEE Standard Test Procedure for Polyphase Induction Motors and Generators. Institute of Electrical and Electronics Engineers. 2004

[5] Grimble Michael "Robust Industrial Control Systems: Optimal Design Approach for Polynomial Systems" John Wiley & Sons, Ltd. 2006.

[6] Wang Xingbo "Target Tracking based on the extended H-infinity filter in wireless sensor networks". Journal of Control Theory and Applications, vol 9, no. 4, pp. 479-486. November 2011.

[7] Simon Dan "Optimal State Estimation. Kalman, H_∞ and Nonlinear Approaches" Wiley Interscience. 2006.

[8] Hassibi Babak, Sayed Ali, Kailath Thomas. "Indefinite Quadratic Estimation and Control . A Unified Approach to H_2 and H_∞ Theories. SIAM Studies in Applied and Numerical Mathematics. 1999.

[9] Lewis Frank, Xie Lihua, Popa Dan. " Optimal and Robust Estimation. With an introduction to Stochastic Control Theory". CRC Press. 2008

[10] Welch Greg, Bishop Gary. "An Introduction to the Kalman Filter" University of North Carolina at Chapel Hill. 2006.

[11] Atkinson DJ, Acarnley PP & Finch JW. "Application of estimation techniques in vector-controlled induction motor drives. Power Electronics and Variable-Speed Drives, 1991., Fourth International Conference on. pp. 358-363. July 1990.

[12] Kim Young-Real, Sul Seung-Ki & Park Min-Ho. "Speed Sensorless Vector Control of Induction Motor Using Extended Kalman Filter. IEEE Transactions on Industry Applications, vol 30, no. 5, pp. 1225-1233. September/October 1994.

[13] Shi K., Wong Y. & Ho S. "Speed Estimation of and Induction Motor Drive Using an Optimized Extended Kalman Filter". IEEE Transactions on Industry Applications, vol 49, no. 1, pp. 124-133. February1994.

[14] Hilairet Mickael, Auger François & Berthelot Eric. "Speed and rotor flux estimation of induction machines using two-stage extended Kalman filter". Automatica, vol 45, no. 8, pp. 1819-1827. August, 2009.

[15] Acarnley PP, Atkinson DJ & Finch JW. "Observers for induction motor rotor current estimator". Industry Applications Society Annual Meeting, 1990. vol 1, pp. 399-403. October 1990.

[16] Abu-Rub Haitham, Guzinski Jaroslaw. "Simple Observer for Induction Motor Speed Sensorless Control". IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society, pp. 2024-2029. November 2011.

[17] Aksoy Saadettin, Mühürcü Aydin & Kizmaz Hakan. "State and Parameter Estimation in Inductiom Motor Using the Extended Kalman Filtering Algorithm". Modern Electric Power Systems (MEPS), 2010 Proceedings of the International Symposium, pp. 1-5. September 2010.

[18] Finch JW, Acarnley PP & Atkinson DJ. "Full-order estimator for induction motor states and parameters". IEEE Electr. Power Appl, vol 30, no. 3, pp. 169-179. May 1998.

[19] Vas, P. "Sensorless vector and direct torque control" Oxford University Press. 1998.

[20] de Souza, C. E., Shaked, U., and Fu, M. "Robust H_∞ filtering for continuous time varying uncertain systems with deterministic input signal." IEEE Transactions on Signal Processing, vol 43, pp. 709-719. 1995.

[21] Hung Y. S., Yang Fuwen. "Robust H_∞ filtering for discrete time-varying incertain systems with a known deterministic input." International Journal of Control, vol 75, pp. 1159-1169. 2002.

[22] Chee-Mun Ong. "Dynamic Simulation of Electric Machinery Using MATLAB/Simulink". Prentice Hall 1998.

[23] Dorato Peter. "Robust control: A historical review." Proceedings of 25th Conference on Decision and Control. pp. 346-349. 1985.

[24] Visual Solutions. "VisSim/Embedded Controls Developer User's Guide". Versión 8. 2010.

[25] Texas Instruments. "Digital Signal Processors TMS320F280x. Data manual. Marzo 2011.

[26] Francecol Technology. "Variador universal pre-caracterizado. Manual del usuario". Referencia: DT10230

[27] Francecol Technology. "Variador universal pre-caracterizado. Control de motores con un DSP F2808". Referencia: DT10060

[28] Francecol Technology. "Tarjeta base variador universal versión Starter Kit". Referencia: DT10210

[29] Francecol Technology. "Módulo de control DSP del software VisSim". Referencia: DT10010

Anexos

Modelo dinámico del motor en Simulink.

```
/*
 * async1.c qdo model of the induction machine in general reference frame
 * Author: Alberto Sanchez-Danilo Llano
 * 20-07-2012
 *
 * ver 2.0
 *
 * Model Description in this version.
 * Note: All variables refered to the stator.
 *
 * States:  x = [Psi_qs Psi_ds Psi_qr Psi_dr wr/wb]
 * Inputs:  u = [v_qs v_ds Tmech Tdamp]
 * Outputs: y = [Tem wr i_qs i_ds Psi_qs Psi_ds Psi_qr Psi_dr]
 *
 * PAR = [xls xm xlr rs rr J P]
 * REF = [w wb]
 *
 */

#define S_FUNCTION_NAME async1

#include "simstruc.h"
#include <math.h>

#define XINIT   ssGetArg(S,0)
#define PAR     ssGetArg(S,1)
#define REF     ssGetArg(S,2)

/*
 * mdlInitializeSizes - initialize the sizes array
 */
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumContStates(S, 5);       /* number of continuous states           */
    ssSetNumDiscStates(S, 0);       /* number of discrete states             */
    ssSetNumInputs(S, 4);           /* number of inputs                      */
    ssSetNumOutputs(S, 8);          /* number of outputs                     */
    ssSetDirectFeedThrough(S, 1);   /* direct feedthrough flag               */
    ssSetNumSampleTimes(S, 1);      /* number of sample times                */
    ssSetNumSFcnParams(S, 3);       /* number of input arguments             */
    ssSetNumRWork(S, 0);            /* number of real work vector elements   */
    ssSetNumIWork(S, 0);            /* number of integer work vector elements*/
    ssSetNumPWork(S, 0);            /* number of pointer work vector elements*/
}

/*
 * mdlInitializeSampleTimes - initialize the sample times array
 */
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
```

```
}


/*
 * mdlInitializeConditions - initialize the states
 */
static void mdlInitializeConditions(double *x0, SimStruct *S)
{
int i;

for (i = 0; i < 5; i++) {
   x0[i] = mxGetPr(XINIT)[i];
}
}

/*
 * mdlOutputs - compute the outputs
 */

static void mdlOutputs(double *y, double *x, double *u, SimStruct *S, int tid)
{
double xls, xm, xlr, rs, rr, J, P, wb, w;
double Tem, i_qs, i_ds, XM, Psi_mq, Psi_md;

xls = mxGetPr(PAR)[0];
xm = mxGetPr(PAR)[1];
xlr = mxGetPr(PAR)[2];
rs = mxGetPr(PAR)[3];
rr = mxGetPr(PAR)[4];
J = mxGetPr(PAR)[5];
P = mxGetPr(PAR)[6];
w = mxGetPr(REF)[0];
wb = mxGetPr(REF)[1];

XM=(xlr*xls*xm)/(xlr*xls+xm*xlr+xm*xls);
Psi_mq=XM*((x[0]/xls)+(x[2]/xlr));
Psi_md=XM*((x[1]/xls)+(x[3]/xlr));
i_qs=(x[0]-Psi_mq)/xls;
i_ds=(x[1]-Psi_md)/xls;
Tem=((3*P)/(4*wb))*(x[1]*i_qs-x[0]*i_ds);

y[0]= Tem;
y[1]= x[4]*wb*2/P;
y[2]= i_qs;
y[3]= i_ds;
y[4]= x[0];
y[5]= x[1];
y[6]= x[2];
y[7]= x[3];
}

/*
 * mdlUpdate - perform action at major integration time step
 */

static void mdlUpdate(double *x, double *u, SimStruct *S, int tid)
```

```
{
}

/*
 * mdlDerivatives - compute the derivatives
 */
static void mdlDerivatives(double *dx, double *x, double *u, SimStruct *S, int tid)
{

double xls, xm, xlr, rs, rr, J, P, wb, w;
double Tem, i_qs, i_ds, XM, Psi_mq, Psi_md;

xls = mxGetPr(PAR)[0];
xm = mxGetPr(PAR)[1];
xlr = mxGetPr(PAR)[2];
rs = mxGetPr(PAR)[3];
rr = mxGetPr(PAR)[4];
J = mxGetPr(PAR)[5];
P = mxGetPr(PAR)[6];
w = mxGetPr(REF)[0];
wb = mxGetPr(REF)[1];

XM=(xlr*xls*xm)/(xlr*xls+xm*xlr+xm*xls);
Psi_mq=XM*(x[0]/xls+x[2]/xlr);
Psi_md=XM*(x[1]/xls+x[3]/xlr);
i_qs=(x[0]-Psi_mq)/xls;
i_ds=(x[1]-Psi_md)/xls;
Tem=((3*P)/(4*wb))*(x[1]*i_qs-x[0]*i_ds);

dx[0]= wb*(u[0]-(w/wb)*x[1]+(rs/xls)*(Psi_mq-x[0]));
dx[1]= wb*(u[1]+(w/wb)*x[0]+(rs/xls)*(Psi_md-x[1]));
dx[2]= wb*(-(w/wb-x[4])*x[3]+(rr/xlr)*(Psi_mq-x[2]));
dx[3]= wb*((w/wb-x[4])*x[2]+(rr/xlr)*(Psi_md-x[3]));
dx[4]=(P)/(2*J*wb)*(Tem+u[2]-u[3]-0.085*x[4]*wb*(2/P));
}


/*
 * mdlTerminate - called when the simulation is terminated.
 */
static void mdlTerminate(SimStruct *S)
{
}

#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

Modelo del filtro Kalman extendido en Simulink y VisSim.

Filtro Kalman lineal en T_s -Simulink

```
function [sys,x0,str,ts] = exkalman4(t,x,u,flag)
%  EXKALMAN Extenden Kalman filter for the induction machine
%
```

```
% Danilo Llano-Alberto SÃ¡nchez
%  Copyright 2012
%
%
% States:  x = [Psi_qs Psi_ds Psi_qr Psi_dr wr Tl]
%          P = [p11 p22 p33 p44 p55]
% Inputs:  u = [v_qs v_ds i_qs i_ds]
% Outputs: y = [Tem i_qs i_ds Psi_qs Psi_ds Psi_qr Psi_dr wr Tl]
%
%  Aproximacion lineal Filtro Kalman Extendido
%
%
%Kalman covariances: Q and R
%

%speed and rotor flux estimation vof induction machines using
%a two-stage extended kalman filter
%
%200us
Q =diag([.02  .02  0.02  0.02  1  0.001]);
%200us
R =0.45*diag([1  1]);

%Provide Machine Parameters


PAR1=[0.754  26.13  0.754  0.435  0.816  0.089  4];
%PAR1 = [0.302  13.08  0.302  0.087  0.228  1.662  4];
%PAR1=[1.206  54.02  1.206  0.262  0.187  11.06  4];
%PAR1=[0.226  13.04  0.226  0.029  0.022  63.87  4];



xls =PAR1(1);
xm = PAR1(2);
xlr =PAR1(3);
rs =PAR1(4);
rr =PAR1(5);
J =PAR1(6);
P =PAR1(7);
w = 0;
wb = 2*pi*60;
Ts = 200e-6;
B=0.085;
z1=exp((-B*Ts)/J);


XM = (xlr*xls*xm)/(xlr*xls + xm*xlr + xm*xls);

as1 = (wb*rs/xls) * (XM/xls - 1);
as2 = (wb*rs/(xls*xlr)) * XM;
ar2 = (wb*rr/(xls*xlr)) * XM;
ar1 = (wb*rr/xlr) * (XM/xlr - 1);

a13 = (3*P*P/(8*wb*J))*(XM/(xls*xlr));
```

```
b = wb;

k1 = (3*P/(4*wb));
cs1 = (1/xls)-XM/(xls*xls);
cs2 = -XM/(xls*xlr);

Gamma=2*k1*(1-z1)/J;


switch flag,

  %%%%%%%%%%%%%%%%%%
  % Initialization %
  %%%%%%%%%%%%%%%%%%
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes(Ts);

  %%%%%%%%%%
  % Update %
  %%%%%%%%%%
  case 2,
    sys = mdlUpdate(t,x,u,as1,as2,ar1,ar2,b,cs1,cs2,a13,k1,Ts,Q,R,J,z1,Gamma,B,P);

  %%%%%%%%%%
  % Output %
  %%%%%%%%%%
  case 3,
    sys = mdlOutputs(t,x,u,cs1,cs2,a13,k1);

  %%%%%%%%%%%%
  % Terminate %
  %%%%%%%%%%%%
  case 9,
    sys = []; % do nothing

  %%%%%%%%%%%%%%%%%%%%
  % Unexpected flags %
  %%%%%%%%%%%%%%%%%%%%
  otherwise
    error(['unhandled flag = ',num2str(flag)]);
end

%end dsfunc

%
%=============================================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function.
%=============================================================================
%
function [sys,x0,str,ts] = mdlInitializeSizes(Ts)

sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 42;
```

```
sizes.NumOutputs     = 9;
sizes.NumInputs      = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);

%condiciones iniciales de:
%[Psi_qs Psi_ds Psi_qr Psi_dr wr] y [p11 p22 p33 p44 p55]
x0(1:6)= [zeros(4,1);.15;0];
x0(7:42)=ones(6,6);
str = [];
ts  = [Ts 0];

%end mdlInitializeSizes

%
%=============================================================
%mdlUpdate
%Handle discrete state updates, sample time hits, and major time step
%requirements.
%=============================================================
%
function sys = mdlUpdate(~,x,u,as1,as2,ar1,ar2,b,cs1,cs2,~,~,Ts,Q,R,~,z1,Gamma,B,P)

xh_minus = x(1:6);
P_minus = [x(7:12)' ; x(13:18)'; x(19:24)'; x(25:30)';x(31:36)'; x(37:42)'];
z = u(3:4);

Hrow1 = [cs1 0 cs2 0 0 0];
Hrow2 = [0 cs1 0 cs2 0 0];
H =[Hrow1;Hrow2];

Phirow1 = [(1+as1*Ts) 0 as2*Ts 0 0 0];
Phirow2 = [0 (1+as1*Ts) 0 as2*Ts 0 0];
Phirow3 = [ar2*Ts 0 (1+ar1*Ts) Ts*xh_minus(5) Ts*xh_minus(4) 0];
Phirow4 = [0 ar2*Ts -Ts*xh_minus(5) (1+ar1*Ts) -Ts*xh_minus(3) 0];
Phirow5 = [-Gamma*cs2*xh_minus(4) Gamma*cs2*xh_minus(3) Gamma*cs2*xh_minus(2) -
Gamma*cs2*xh_minus(1) z1 ((z1-1)/B)];
Phirow6=[0 0 0 0 0 1];


Phi = [Phirow1; Phirow2; Phirow3; Phirow4; Phirow5; Phirow6];

K = P_minus*H'/(H*P_minus*H'+R);
xh_plus = xh_minus+K*(z-H*xh_minus);

P_plus = P_minus-K*H*P_minus;
P_plus = 0.5*(P_plus+P_plus');

P_minus = Phi*P_plus*Phi'+Q;

xh_temp = zeros(6,1);
xh_temp(1) = (1+as1*Ts)*xh_minus(1)+as2*Ts*xh_minus(3)+b*Ts*u(1);
xh_temp(2) = (1+as1*Ts)*xh_minus(2)+as2*Ts*xh_minus(4)+b*Ts*u(2);
xh_temp(3) = (1+ar1*Ts)*xh_minus(3)+ar2*Ts*xh_minus(1)+Ts*xh_minus(4)*xh_minus(5);
```

```
xh_temp(4) = ar2*Ts*xh_minus(2)-Ts*xh_minus(3)*xh_minus(5)+(1+ar1*Ts)*xh_minus(4);
xh_temp(5) =Gamma*cs2*xh_minus(2)*xh_minus(3)-
Gamma*cs2*xh_minus(1)*xh_minus(4)+z1*xh_minus(5)+(P/2)*((z1-1)/B)*xh_minus(6);
xh_temp(6)=xh_plus(6);
sys(1:6)=xh_temp;
sys(7:42)=P_minus;
%sys = [xh_temp; diag(P_minus)];

%end mdlUpdate

%
%=============================================================================
% mdlOutputs
% Return the output vector for the S-function
%=============================================================================
%
function sys = mdlOutputs(t,x,u,cs1,cs2,a13,k1)

% x = [Psi_qs Psi_ds Psi_qr Psi_dr wr]
% Tem = k1*(x(1)*x(4)-x(2)*x(3));
iqs = cs1*x(1)+cs2*x(3);
ids = cs1*x(2)+cs2*x(4);
Tem = k1*(x(2)*iqs-x(1)*ids);
sys = [Tem; iqs; ids; x(1:4); .5*x(5); -x(6)];

%end mdlOutputs
```

Filtro Kalman cuadrático en T_s -Simulink

```
function [sys,x0,str,ts] = exkalman5(t,x,u,flag)
%  EXKALMAN Extenden Kalman filter for the induction machine
%
% Autor: Danilo Llano-Alberto SÃ¡nchez
%   Copyright 2012
%
% States:  x = [Psi_qs Psi_ds Psi_qr Psi_dr wr Tl]
%          P = [p11 p22 p33 p44 p55]
% Inputs:  u = [v_qs v_ds i_qs i_ds]
% Outputs: y = [Tem i_qs i_ds Psi_qs Psi_ds Psi_qr Psi_dr wr Tl]
%
%
% Aproximacion cuadratica Filtro Kalman
%
% Kalman covariances: Q and R
%

% speed and rotor flux estimation vof induction machines using a
% two-stage extended kalman filter
%
% 500us
%Q=diag([0.01 0.01 .001 .001 .1 .001]);
% 500us
%R=0.5*diag([1 1]);
% 200us
Q =diag([.02 .02 0.02 0.02 1 0.001]);
%200us
R =0.45*diag([1 1]);
```

```
% Provide Machine Parameters
%PAR = [0.95 31.93 0.95 0.531 0.408 0.1 4];

PAR1=[0.754 26.13 0.754 0.435 0.816 0.089 4];
%PAR1 = [0.302 13.08 0.302 0.087 0.228 1.662 4];
%PAR1=[1.206 54.02 1.206 0.262 0.187 11.06 4];
%PAR1=[0.226 13.04 0.226 0.029 0.022 63.87 4];



xls =PAR1(1);
xm = PAR1(2);
xlr =PAR1(3);
rs =PAR1(4);
rr =PAR1(5);
J =PAR1(6);
P =PAR1(7);
w = 0;
wb = 2*pi*60;
Ts = 500e-6;
B=0.085;
z1=exp((-B*Ts)/J);


XM = (xlr*xls*xm)/(xlr*xls + xm*xlr + xm*xls);

as1 = (wb*rs/xls) * (XM/xls - 1);
as2 = (wb*rs/(xls*xlr)) * XM;
ar2 = (wb*rr/(xls*xlr)) * XM;
ar1 = (wb*rr/xlr) * (XM/xlr - 1);

a13 = (3*P*P/(8*wb*J))*(XM/(xls*xlr));

b = wb;

k1 = (3*P/(4*wb));
cs1 = (1/xls)-XM/(xls*xls);
cs2 = -XM/(xls*xlr);

Gamma=2*k1*(1-z1)/J;


switch flag,

  %%%%%%%%%%%%%%%%%%
  % Initialization %
  %%%%%%%%%%%%%%%%%%
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes(Ts);

  %%%%%%%%%%
  % Update %
  %%%%%%%%%%
  case 2,
    sys = mdlUpdate(t,x,u,as1,as2,ar1,ar2,b,cs1,cs2,a13,k1,Ts,Q,R,J,z1,Gamma,B,P);
```

```
  %%%%%%%%%%
  % Output %
  %%%%%%%%%%
  case 3,
    sys = mdlOutputs(t,x,u,cs1,cs2,a13,k1);

  %%%%%%%%%%%%
  % Terminate %
  %%%%%%%%%%%%
  case 9,
    sys = []; % do nothing

  %%%%%%%%%%%%%%%%%%%%
  % Unexpected flags %
  %%%%%%%%%%%%%%%%%%%%
  otherwise
    error(['unhandled flag = ',num2str(flag)]);
end

%end dsfunc

%
%=============================================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function.
%=============================================================================
%
function [sys,x0,str,ts] = mdlInitializeSizes(Ts)

sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 42;
sizes.NumOutputs     = 9;
sizes.NumInputs      = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);

% condiciones iniciales de:
% [Psi_qs Psi_ds Psi_qr Psi_dr wr] y [p11 p22 p33 p44 p55]
x0(1:6)= [zeros(4,1);.15;0];
x0(7:42)=ones(6,6);
str = [];
ts  = [Ts 0];

% end mdlInitializeSizes

%
%=============================================================================
% mdlUpdate
% Handle discrete state updates, sample time hits, and major time step
% requirements.
%=============================================================================
%
```

```
function sys = mdlUpdate(~,x,u,as1,as2,ar1,ar2,b,cs1,cs2,~,~,Ts,Q,R,~,z1,Gamma,B,P)

xh_minus = x(1:6);
P_minus = [x(7:12)' ; x(13:18)'; x(19:24)'; x(25:30)';x(31:36)'; x(37:42)'];
z = u(3:4);

Hrow1 = [cs1 0 cs2 0 0 0];
Hrow2 = [0 cs1 0 cs2 0 0];
H =[Hrow1;Hrow2];

Phirow1 = [(1+as1*Ts+.5*Ts*Ts*(as1*as1+as2*ar2)) 0 (as2*Ts+.5*Ts*Ts*(as1*as2+as2*ar1))
(.5*Ts*Ts*as2*xh_minus(5)) (.5*as2*Ts*Ts*xh_minus(4)) 0];
Phirow2 = [0 (1+as1*Ts+.5*Ts*Ts*(as1*as1+as2*ar2)) (-.5*Ts*Ts*as2*xh_minus(5))
 (as2*Ts+.5*Ts*Ts*(as1*as2+as2*ar1)) (-.5*as2*Ts*Ts*xh_minus(3)) 0];
Phirow3 = [(ar2*Ts+.5*Ts*Ts*(ar2*as1+ar1*ar2)) (.5*Ts*Ts*ar2*xh_minus(5))
(1+ar1*Ts+.5*Ts*Ts*(ar2*as2+ar1*ar1-xh_minus(5)*xh_minus(5)))
(Ts*xh_minus(5)+ar1*Ts*Ts*xh_minus(5)) (ar2*.5*Ts*Ts*xh_minus(2)-
Ts*Ts*xh_minus(3)*xh_minus(5)+(Ts+Ts*Ts*ar1)*xh_minus(4)) 0];

Phirow4 = [(-.5*Ts*Ts*ar2*xh_minus(5)) (ar2*Ts+.5*Ts*Ts*(as1*ar2+ar1*ar2))
(-Ts*xh_minus(5)-Ts*Ts*ar1*xh_minus(5))
 (1+ar1*Ts+.5*Ts*Ts*(as2*ar2+ar1*ar1-xh_minus(5)*xh_minus(5)))
 (-ar2*.5*Ts*Ts*xh_minus(1)*xh_minus(1)-Ts*Ts*xh_minus(4)*xh_minus(5)
-(Ts+Ts*Ts*ar1)*xh_minus(3)) 0];
Phirow5 = [-Gamma*cs2*xh_minus(4) Gamma*cs2*xh_minus(3) Gamma*cs2*xh_minus(2)
 -Gamma*cs2*xh_minus(1) z1 ((z1-1)/B)];
Phirow6=[0 0 0 0 0 1];


Phi = [Phirow1; Phirow2; Phirow3; Phirow4; Phirow5; Phirow6];

K = P_minus*H'/(H*P_minus*H'+R);
xh_plus = xh_minus+K*(z-H*xh_minus);

P_plus = P_minus-K*H*P_minus;
P_plus = 0.5*(P_plus+P_plus');

P_minus = Phi*P_plus*Phi'+Q;

xh_temp = zeros(6,1);
xh_temp(1) = (1+as1*Ts+.5*Ts*Ts*(as1*as1+as2*ar2))*xh_minus(1)+
(as2*Ts+.5*Ts*Ts*(as1*as2+as2*ar1))*xh_minus(3)+
(.5*Ts*Ts*as2)*xh_minus(5)*xh_minus(4)+(b*Ts+as1*b*.5*Ts*Ts)*u(1);
xh_temp(2) = (1+as1*Ts+.5*Ts*Ts*(as1*as1+as2*ar2))*xh_minus(2)
-.5*Ts*Ts*as2*xh_minus(5)*xh_minus(3)
+(as2*Ts+.5*Ts*Ts*(as1*as2+as2*ar1))*xh_minus(4)+(b*Ts+as1*b*.5*Ts*Ts)*u(2);
xh_temp(3) = (1+ar1*Ts+.5*Ts*Ts*(ar2*as2+ar1*ar1-xh_minus(5)*xh_minus(5)))*xh_minus(3)
+.5*Ts*Ts*ar2*xh_minus(5)*xh_minus(2)+
(ar2*Ts+.5*Ts*Ts*(ar2*as1+ar1*ar2))*xh_minus(1)
+(1+Ts*ar1)*Ts*xh_minus(4)*xh_minus(5)+u(1)*ar2*.5*b*Ts*Ts;
xh_temp(4) = -.5*Ts*Ts*ar2*xh_minus(5)*xh_minus(1)
+(ar2*Ts+.5*Ts*Ts*(as1*ar2+ar1*ar2))*xh_minus(2)
-(Ts+ar1*Ts*Ts)*xh_minus(3)*xh_minus(5)+
(1+ar1*Ts+.5*Ts*Ts*(as2*ar2-xh_minus(5)*xh_minus(5)+ar1*ar1))*xh_minus(4)+
u(2)*ar2*b*.5*Ts*Ts;
xh_temp(5) =Gamma*cs2*xh_minus(2)*xh_minus(3)-Gamma*cs2*xh_minus(1)*xh_minus(4)
```

```
+z1*xh_minus(5)+(P/2)*((z1-1)/B)*xh_minus(6);
xh_temp(6)=xh_plus(6);
sys(1:6)=xh_temp;
sys(7:42)=P_minus;
%sys = [xh_temp; diag(P_minus)];

%end mdlUpdate

%
%==============================================================
% mdlOutputs
% Return the output vector for the S-function
%==============================================================
%
function sys = mdlOutputs(t,x,u,cs1,cs2,a13,k1)

%x = [Psi_qs Psi_ds Psi_qr Psi_dr wr]
%Tem = k1*(x(1)*x(4)-x(2)*x(3));
iqs = cs1*x(1)+cs2*x(3);
ids = cs1*x(2)+cs2*x(4);
Tem = k1*(x(2)*iqs-x(1)*ids);
sys = [Tem; iqs; ids; x(1:4); .5*x(5); -x(6)];

%end mdlOutputs
```

Filtro Kalman extendido- Código VisSim

```
/*** VisSim Automatic C Code Generator Version 8.0B10 ***/
/*  Output for K1 at Mon Jul 30 12:05:11 2012 */


#include "math.h"
#include "cgen.h"
#include "cgendll.h"
#include "c2000.h"
#include "DMC32.h"
#include "DMC32.h"
int _maxAnalogInChan=23;
CLARKE clarke3377 = CLARKE_DEFAULTS;
CLARKE clarke3378 = CLARKE_DEFAULTS;
INV_CLARKE invClarke22 = {0};
static long beta1;
static long K11;
static long K12;
static long beta2;
static long x6_k1;
static long x6;
static long K9;
static long K10;
static long x5_k1;
static long x5;
static long x4;
static long x4_k1;
static long K8;
static long K7;
static long x3;
static long x3_k1;
static long K6;
static long K5;
static long K3;
static long K4;
static long x2_k1;
static long x2;
static long x1;
static long x1_k1;
static long K2;
static long K1;
static long h1;
static long h2;
static long i2;
static long i1;
static long as1;
static long Ts;
static long A1;
static long as2;
static long A3;
static long A6;
static long A8;
static long A9;
static long ar2;
static long A11;
static long ar1;
static long A12;
static long A14;
static long A16;
static long A15;
static long A56;
```

```
static long z1;
static long Gamma;
static long cs2;
static long A53;
static long A54;
static long B1;
static long b;
static long B4;
static long cs1;
static long v1;
static long v2;
static long P13;
static long P1;
static long Pka1;
static long Q1;
static long P3;
static long P15;
static long P7;
static long P19;
static long Pka7;
static long P21;
static long P9;
static long f3;
static long P27;
static long P25;
static long Pka13;
static long f4;
static long Pka19;
static long P33;
static long P31;
static long Pka31;
static long Pka25;
static long P16;
static long P4;
static long Pka2;
static long P2;
static long P14;
static long P22;
static long P10;
static long Pka8;
static long P8;
static long P20;
static long P28;
static long P26;
static long Pka14;
static long Pka20;
static long P32;
static long Pka26;
static long P34;
static long Pka32;
static long Q2;
static long Pka3;
static long P5;
static long P17;
static long P23;
static long P11;
static long Pka9;
static long Pka15;
static long P29;
static long Pka21;
static long Q3;
static long Pka4;
```

```
static long Pka10;
static long Pka16;
static long Pka22;
static long Q4;
static long Pka5;
static long P6;
static long P18;
static long Pka11;
static long P24;
static long P12;
static long Pka17;
static long Pka23;
static long P30;
static long Pka6;
static long Pka12;
static long Pka18;
static long Pka24;
static long Pka30;
static long P36;
static long Q6;
static long Pka36;
static long P35;
static long Pka34;
static long Pka28;
static long Pka35;
static long Q5;
static long Pka29;
static long R1;
static long L1;
static long L2;
static long L3;
static long L4;
static long R2;
static long det;
static long alpha1;
static long alpha4;
static long alpha2;
static long alpha3;
static long Pka27;
static long Pka33;
static long va;
static long vb;
static long ib;
static long ia;

extern CGDOUBLE Zed;

DLL_SIG_DECL_M(0,12)
INTERRUPT void cgMain();
static ARG_DESCR outArgInfo3393[]={
  { T_SCALED_INT,0,8,32},
  { T_SCALED_INT,0,8,32},
  { T_SCALED_INT,0,14,32},
  { T_SCALED_INT,0,14,32},
  { T_SCALED_INT,0,14,32},
  { T_SCALED_INT,0,14,32},
  { T_SCALED_INT,0,14,32},
  { T_SCALED_INT,0,14,32},
  { T_SCALED_INT,0,14,32},
  { T_SCALED_INT,0,14,32},
  { T_SCALED_INT,0,14,32},
  { T_SCALED_INT,0,14,32},
```

```
};
static ARG_DESCR inArgInfo3393[]={
0};
static SIM_STATE tSim={0, 0.0002, 1,0,0.0002,0,0,0,0,0,0,0,0
,outArgInfo3393, inArgInfo3393,0,12,0,0,0,cgMain,0,0,0,0,0,0,1};
SIM_STATE *sim=&tSim;

/* A */
INTERRUPT void cgMain()
{
static long _delayOutBuf1010=262144;
static long _delayOutBuf486=262144;
static long _delayOutBuf533=262144;
static long _delayOutBuf532=262144;
static long _delayOutBuf954=262144;
static long _delayOutBuf953=262144;
static long _delayOutBuf969=262144;
static long _delayOutBuf970=262144;
static long _delayOutBuf1092=262144;
static long _delayOutBuf626=262144;
static long _delayOutBuf625=262144;
static long _delayOutBuf637=262144;
static long _delayOutBuf3728=0;
static long _delayOutBuf206=262144;
static long _delayOutBuf207=262144;
static long _delayOutBuf3742=0;
static long _delayOutBuf210=262144;
static long _delayOutBuf211=262144;
static long _delayOutBuf404=262144;
static long _delayOutBuf309=262144;
static long _delayOutBuf876=262144;
static long _delayOutBuf877=262144;
static long _delayOutBuf865=262144;
static long _delayOutBuf1247=262144;
static long _delayOutBuf1248=262144;
static long _delayOutBuf1232=262144;
static long _delayOutBuf1231=262144;
static long _delayOutBuf1260=262144;
static long _delayOutBuf1261=262144;
static long _delayOutBuf1219=262144;
static long _delayOutBuf1218=262144;
static long _delayOutBuf176=262144;
static long _delayOutBuf1631=262144;
static long _delayOutBuf3400=262144;
static long _delayOutBuf1699=262144;
static long _delayOutBuf1644=262144;
static long _delayOutBuf1678=262144;
static long _delayOutBuf1677=262144;
static long _delayOutBuf1669=262144;
static long _delayOutBuf1686=262144;
static long _delayOutBuf1643=262144;
static long _delayOutBuf1700=262144;
static long _delayOutBuf1708=262144;
static long _delayOutBuf1691=262144;
static long _delayOutBuf1713=262144;
static long _delayOutBuf1723=262144;
static long _delayOutBuf3474=262144;
static long _delayOutBuf714=262144;
static long _delayOutBuf715=262144;
static long _delayOutBuf1739=262144;
static long _delayOutBuf1738=262144;
static long _delayOutBuf1747=262144;
```

```
static long _delayOutBuf1905=262144;
static long _delayOutBuf1906=262144;
static long _delayOutBuf1897=262144;
static long _delayOutBuf1914=262144;
static long _delayOutBuf1892=262144;
static long _delayOutBuf1887=262144;
static long _delayOutBuf1397=262144;
static long _delayOutBuf1396=262144;
static long _delayOutBuf1408=262144;
static long _delayOutBuf3694=262144;
static long _delayOutBuf1422=262144;
static long _delayOutBuf1423=262144;
static long _delayOutBuf3525=262144;
static long _delayOutBuf1325=262144;
static long _delayOutBuf1324=262144;
static long _delayOutBuf1340=262144;
static long _delayOutBuf1341=262144;
static long _delayOutBuf1312=262144;
static long _delayOutBuf1311=262144;
static long _delayOutBuf1299=262144;
static long _delayOutBuf1300=262144;
static long _delayOutBuf1353=262144;
static long _delayOutBuf1354=262144;
static long _delayOutBuf1522=262144;
static long _delayOutBuf1523=262144;
static long _delayOutBuf1511=262144;
static long _delayOutBuf1531=262144;
static long _delayOutBuf1544=262144;
static long _delayOutBuf1543=262144;
static long _delayOutBuf1553=262144;
static long _delayOutBuf1535=262144;
static long _delayOutBuf1567=262144;
static long _delayOutBuf1568=262144;
static long _delayOutBuf1558=262144;
static long _delayOutBuf1576=262144;
static long _delayOutBuf1496=262144;
static long _delayOutBuf1495=262144;
static long _delayOutBuf1505=262144;
static long _delayOutBuf652=262144;
static long _delayOutBuf651=262144;
static long _delayOutBuf663=262144;
static long _delayOutBuf643=262144;
static long _delayOutBuf668=262144;
static long _delayOutBuf673=262144;
static long _delayOutBuf697=262144;
static long _delayOutBuf698=262144;
static long _delayOutBuf688=262144;
static long _delayOutBuf706=262144;
static long _delayOutBuf683=262144;
static long _delayOutBuf678=262144;
static long _delayOutBuf1487=262144;
static long _delayOutBuf1590=262144;
static long _delayOutBuf1589=262144;
static long _delayOutBuf1599=262144;
static long _delayOutBuf1581=262144;
static long _delayOutBuf1613=262144;
static long _delayOutBuf1614=262144;
static long _delayOutBuf1604=262144;
static long _delayOutBuf2028=262144;
static long _delayOutBuf2027=262144;
static long _delayOutBuf2039=262144;
static long _delayOutBuf2019=262144;
```

```
static long _delayOutBuf2044=262144;
static long _delayOutBuf2049=262144;
static long _delayOutBuf2073=262144;
static long _delayOutBuf2074=262144;
static long _delayOutBuf2064=262144;
static long _delayOutBuf2082=262144;
static long _delayOutBuf2059=262144;
static long _delayOutBuf2054=262144;
static long _delayOutBuf2005=262144;
static long _delayOutBuf2006=262144;
static long _delayOutBuf1996=262144;
static long _delayOutBuf2014=262144;
static long _delayOutBuf1991=262144;
static long _delayOutBuf1986=262144;
static long _delayOutBuf2096=262144;
static long _delayOutBuf2095=262144;
static long _delayOutBuf2105=262144;
static long _delayOutBuf2087=262144;
static long _delayOutBuf2110=262144;
static long _delayOutBuf2115=262144;
static long _delayOutBuf1962=262144;
static long _delayOutBuf1961=262144;
static long _delayOutBuf1971=262144;
static long _delayOutBuf1953=262144;
static long _delayOutBuf1976=262144;
static long _delayOutBuf1981=262144;
static long _delayOutBuf1939=262144;
static long _delayOutBuf1940=262144;
static long _delayOutBuf1930=262144;
static long _delayOutBuf1948=262144;
static long _delayOutBuf1925=262144;
static long _delayOutBuf1920=262144;
static long _delayOutBuf1622=262144;
static long _delayOutBuf1366=262144;
static long _delayOutBuf1365=262144;
static long _delayOutBuf3524=262144;
static long _delayOutBuf3536=262144;
static long _delayOutBuf3516=262144;
static long _delayOutBuf3541=262144;
static long _delayOutBuf3546=262144;
static long _delayOutBuf3570=262144;
static long _delayOutBuf3571=262144;
static long _delayOutBuf3561=262144;
static long _delayOutBuf3579=262144;
static long _delayOutBuf3556=262144;
static long _delayOutBuf3551=262144;
static long _delayOutBuf3502=262144;
static long _delayOutBuf3503=262144;
static long _delayOutBuf3493=262144;
static long _delayOutBuf3511=262144;
static long _delayOutBuf3488=262144;
static long _delayOutBuf3483=262144;
static long _delayOutBuf3645=262144;
static long _delayOutBuf3644=262144;
static long _delayOutBuf3654=262144;
static long _delayOutBuf3636=262144;
static long _delayOutBuf3659=262144;
static long _delayOutBuf3664=262144;
static long _delayOutBuf1413=262144;
static long _delayOutBuf3699=262144;
static long _delayOutBuf1435=262144;
static long _delayOutBuf1434=262144;
```

```
static long _delayOutBuf1444=262144;
static long _delayOutBuf3704=262144;
static long _delayOutBuf1384=262144;
static long _delayOutBuf1385=262144;
static long _delayOutBuf1375=262144;
static long _delayOutBuf3709=262144;
static long _delayOutBuf1458=262144;
static long _delayOutBuf1459=262144;
static long _delayOutBuf1449=262144;
static long _delayOutBuf3714=262144;
static long _delayOutBuf1471=262144;
static long _delayOutBuf1470=262144;
static long _delayOutBuf1480=262144;
static long _delayOutBuf3719=262144;
static long _delayOutBuf1730=262144;
static long _delayOutBuf3475=262144;
static long _delayOutBuf3466=262144;
static long _delayOutBuf3669=262144;
static long _delayOutBuf424=262144;
static long _delayOutBuf423=262144;
static long _delayOutBuf1718=262144;
static long _delayOutBuf1655=262144;
static long _delayOutBuf1656=262144;
static long _delayOutBuf1664=262144;
static long _delayOutBuf3724=262144;
static long _delayOutBuf1632=262144;
static long _delayOutBuf1273=262144;
static long _delayOutBuf1274=262144;
static long _delayOutBuf1286=262144;
static long _delayOutBuf1287=262144;
static long _delayOutBuf1839=262144;
static long _delayOutBuf1840=262144;
static long _delayOutBuf1828=262144;
static long _delayOutBuf1848=262144;
static long _delayOutBuf1823=262144;
static long _delayOutBuf1818=262144;
static long _delayOutBuf1794=262144;
static long _delayOutBuf1793=262144;
static long _delayOutBuf1803=262144;
static long _delayOutBuf1785=262144;
static long _delayOutBuf1808=262144;
static long _delayOutBuf1813=262144;
static long _delayOutBuf1862=262144;
static long _delayOutBuf1861=262144;
static long _delayOutBuf1871=262144;
static long _delayOutBuf1853=262144;
static long _delayOutBuf1876=262144;
static long _delayOutBuf1881=262144;
static long _delayOutBuf1771=262144;
static long _delayOutBuf1772=262144;
static long _delayOutBuf1762=262144;
static long _delayOutBuf1780=262144;
static long _delayOutBuf1757=262144;
static long _delayOutBuf1752=262144;
static long _delayOutBuf885=262144;
static long _delayOutBuf898=262144;
static long _delayOutBuf897=262144;
static long _delayOutBuf907=262144;
static long _delayOutBuf889=262144;
static long _delayOutBuf921=262144;
static long _delayOutBuf922=262144;
static long _delayOutBuf912=262144;
```

```
static long _delayOutBuf930=262144;
static long _delayOutBuf3621=262144;
static long _delayOutBuf3620=262144;
static long _delayOutBuf3630=262144;
static long _delayOutBuf3612=262144;
static long _delayOutBuf308=262144;
static long _delayOutBuf324=262144;
static long _delayOutBuf325=262144;
static long _delayOutBuf403=262144;
static long _delayOutBuf415=262144;
static long _delayOutBuf395=262144;
static long _delayOutBuf381=262144;
static long _delayOutBuf382=262144;
static long _delayOutBuf372=262144;
static long _delayOutBuf390=262144;
static long _delayOutBuf237=262144;
static long _delayOutBuf236=262144;
static long _delayOutBuf235=262144;
static long _delayOutBuf3734=0;
static long _delayOutBuf208=262144;
static long _delayOutBuf209=262144;
static long _delayOutBuf3737=0;
static long _delayOutBuf212=262144;
static long _delayOutBuf3395=262144;
static long _delayOutBuf617=262144;
static long _delayOutBuf604=262144;
static long _delayOutBuf605=262144;
static long _delayOutBuf595=262144;
static long _delayOutBuf613=262144;
static long _delayOutBuf1093=262144;
static long _delayOutBuf1081=262144;
static long _delayOutBuf3674=262144;
static long _delayOutBuf1067=262144;
static long _delayOutBuf1066=262144;
static long _delayOutBuf1076=262144;
static long _delayOutBuf3679=262144;
static long _delayOutBuf1054=262144;
static long _delayOutBuf1055=262144;
static long _delayOutBuf1045=262144;
static long _delayOutBuf3684=262144;
static long _delayOutBuf1105=262144;
static long _delayOutBuf1104=262144;
static long _delayOutBuf1114=262144;
static long _delayOutBuf3689=262144;
static long _delayOutBuf941=262144;
static long _delayOutBuf940=262144;
static long _delayOutBuf982=262144;
static long _delayOutBuf983=262144;
static long _delayOutBuf586=262144;
static long _delayOutBuf587=262144;
static long _delayOutBuf575=262144;
static long _delayOutBuf3420=262144;
static long _delayOutBuf561=262144;
static long _delayOutBuf560=262144;
static long _delayOutBuf570=262144;
static long _delayOutBuf3425=262144;
static long _delayOutBuf520=262144;
static long _delayOutBuf521=262144;
static long _delayOutBuf505=262144;
static long _delayOutBuf504=262144;
static long _delayOutBuf548=262144;
static long _delayOutBuf549=262144;
```

```
static long _delayOutBuf175=262144;
static long _delayOutBuf487=262144;
static long _delayOutBuf475=262144;
static long _delayOutBuf495=262144;
static long _delayOutBuf470=262144;
static long _delayOutBuf465=262144;
static long _delayOutBuf441=262144;
static long _delayOutBuf440=262144;
static long _delayOutBuf450=262144;
static long _delayOutBuf432=262144;
static long _delayOutBuf455=262144;
static long _delayOutBuf460=262144;
static long _delayOutBuf253=262144;
static long _delayOutBuf254=262144;
static long _delayOutBuf269=262144;
static long _delayOutBuf255=262144;
static long _delayOutBuf268=262144;
static long _delayOutBuf3394=262144;
static long _delayOutBuf1173=262144;
static long _delayOutBuf1172=262144;
static long _delayOutBuf1184=262144;
static long _delayOutBuf1164=262144;
static long _delayOutBuf1151=262144;
static long _delayOutBuf1152=262144;
static long _delayOutBuf1142=262144;
static long _delayOutBuf1160=262144;
static long _delayOutBuf1128=262144;
static long _delayOutBuf1127=262144;
static long _delayOutBuf1137=262144;
static long _delayOutBuf1119=262144;
static long _delayOutBuf1199=262144;
static long _delayOutBuf1200=262144;
static long _delayOutBuf1190=262144;
static long _delayOutBuf1208=262144;
static long _delayOutBuf1011=262144;
static long _delayOutBuf995=262144;
static long _delayOutBuf994=262144;
static long _delayOutBuf1023=262144;
static long _delayOutBuf1024=262144;
static long _delayOutBuf1036=262144;
static long _delayOutBuf1035=262144;
static long _delayOutBuf783=262144;
static long _delayOutBuf782=262144;
static long _delayOutBuf798=262144;
static long _delayOutBuf799=262144;
static long _delayOutBuf3447=262144;
static long _delayOutBuf3446=262144;
static long _delayOutBuf770=262144;
static long _delayOutBuf769=262144;
static long _delayOutBuf744=262144;
static long _delayOutBuf745=262144;
static long _delayOutBuf729=262144;
static long _delayOutBuf728=262144;
static long _delayOutBuf3434=262144;
static long _delayOutBuf3435=262144;
static long _delayOutBuf757=262144;
static long _delayOutBuf758=262144;
static long _delayOutBuf811=262144;
static long _delayOutBuf810=262144;
static long _delayOutBuf822=262144;
static long _delayOutBuf3456=262144;
static long _delayOutBuf836=262144;
```

```
static long _delayOutBuf837=262144;
static long _delayOutBuf827=262144;
static long _delayOutBuf3461=262144;
static long _delayOutBuf849=262144;
static long _delayOutBuf848=262144;
static long _delayOutBuf858=262144;
static long _delayOutBuf3584=262144;
static long _delayOutBuf3603=262144;
static long _delayOutBuf3604=262144;
static long _delayOutBuf3594=262144;
static long _delayOutBuf3589=262144;
static long _delayOutBuf362=262144;
static long _delayOutBuf363=262144;
static long _delayOutBuf351=262144;
static long _delayOutBuf3410=262144;
static long _delayOutBuf337=262144;
static long _delayOutBuf336=262144;
static long _delayOutBuf346=262144;
static long _delayOutBuf3415=262144;
static long _delayOutBuf280=262144;
static long _delayOutBuf279=262144;
static long _delayOutBuf296=262144;
static long _delayOutBuf297=262144;
  long t120;
  long t123;
  long t135;
  long t132;
  long t167;
  long t59;
  long t3344;
  long t31;
  long t29;
static long _sampBuf27=0;
  long t8;
  long t24;
  long t3341;
  long t12;
  int t10;
  long t13;
  int t9;
  long t14;
  int t11;
  ADCTRL2 |= 0x4001;  // Reset ADC Seq
  ADCTRL2 |= 0x2001;  // Trigger ADC
  GET_MAX_STACK_USED();
  as1 = -88826472 /* -338.846@fx14.32 */;
  Ts = 52 /* 0.0002@fx14.32 */;
  t120 = (MUL_SHIFT32( as1 , Ts ,18)+262144 /* 1@fx14.32 */);
  A6 =  t120;
  as2 = 86174623 /* 328.73@fx14.32 */;
  t123 = MUL_SHIFT32( as2 , Ts ,18);
  A8 =  t123;
  ar1 = -119718071 /* -456.688@fx14.32 */;
  t135 = (MUL_SHIFT32( ar1 , Ts ,18)+262144 /* 1@fx14.32 */);
  A16 =  t135;
  ar2 = 116143973 /* 443.054@fx14.32 */;
  t132 = MUL_SHIFT32( ar2 , Ts ,18);
  A14 =  t132;
  A12 = MUL_SHIFT32( Ts , _delayOutBuf3394,18);
  A15 = MUL_SHIFT32( A12 ,-1073741824L,30)/* -1@fx14.32 */;
  f4 = MUL_SHIFT32( _delayOutBuf3395,MUL_SHIFT32( Ts ,-1073741824L,30)/* -1@fx14.32 */,18);
  Pka10 = (MUL_SHIFT32((MUL_SHIFT32( A6 , _delayOutBuf1010,18)+MUL_SHIFT32( A8 , _delayOutBuf1011,18)), A16 ,18)+MUL_SHIFT32((MUL_SHIFT32(
```

```
A8 , _delayOutBuf995,18)+MUL_SHIFT32( A6 , _delayOutBuf994,18)), A14 ,18)+MUL_SHIFT32((MUL_SHIFT32( A6 , _delayOutBuf1023,18)+
MUL_SHIFT32( A8 , _delayOutBuf1024,18)), A15 ,18)+MUL_SHIFT32((MUL_SHIFT32( A6 , _delayOutBuf1036,18)+MUL_SHIFT32( A8 , _delayOutBuf1035,18)),
f4 ,18));
A1 = t120;
A3 = t123;
Pka4 = (MUL_SHIFT32((MUL_SHIFT32( A1 , _delayOutBuf954,18)+MUL_SHIFT32( A3 , _delayOutBuf953,18)), A14 ,18)+MUL_SHIFT32((MUL_SHIFT32(
A1 , _delayOutBuf969,18)+MUL_SHIFT32( A3 , _delayOutBuf970,18)), A15 ,18)+MUL_SHIFT32((MUL_SHIFT32( A1 , _delayOutBuf941,18)+MUL_SHIFT32(
A3 , _delayOutBuf940,18)), A16 ,18)+MUL_SHIFT32((MUL_SHIFT32( A1 , _delayOutBuf982,18)+MUL_SHIFT32( A3 , _delayOutBuf983,18)),
f4 ,18));
cs1 = 20892 /* 0.0797@fx14.32 */;
h1 = cs1 ;
Q2 = 393216 /* 1.5@fx14.32 */;
Pka8 = (MUL_SHIFT32((MUL_SHIFT32( A6 , _delayOutBuf533,18)+MUL_SHIFT32( A8 , _delayOutBuf532,18)), A8 ,18)+MUL_SHIFT32((MUL_SHIFT32(
A6 , _delayOutBuf548,18)+MUL_SHIFT32( A8 , _delayOutBuf549,18)), A6 ,18)+ Q2 );
cs2 = -20892 /* -0.0797@fx14.32 */;
h2 = cs2 ;
Pka7 = (MUL_SHIFT32((MUL_SHIFT32( A8 , _delayOutBuf309,18)+MUL_SHIFT32( A6 , _delayOutBuf308,18)), A1 ,18)+MUL_SHIFT32((MUL_SHIFT32(
A8 , _delayOutBuf324,18)+MUL_SHIFT32( A6 , _delayOutBuf325,18)), A3 ,18));
Pka19 = (MUL_SHIFT32((MUL_SHIFT32( A15 , _delayOutBuf404,18)+MUL_SHIFT32( A16 , _delayOutBuf403,18)+MUL_SHIFT32( A14 , _delayOutBuf415,18)+
MUL_SHIFT32( f4 , _delayOutBuf395,18)), A1 ,18)+MUL_SHIFT32((MUL_SHIFT32( A15 , _delayOutBuf381,18)+MUL_SHIFT32( A16 , _delayOutBuf382,18)+
MUL_SHIFT32( A14 , _delayOutBuf372,18)+MUL_SHIFT32( f4 , _delayOutBuf390,18)), A3 ,18));
A11 = t135;
A9 = t132;
f3 = MUL_SHIFT32( _delayOutBuf3400, Ts ,18);
Pka9 = (MUL_SHIFT32((MUL_SHIFT32( A6 , _delayOutBuf783,18)+MUL_SHIFT32( A8 , _delayOutBuf782,18)), A12 ,18)+MUL_SHIFT32((MUL_SHIFT32(
A8 , _delayOutBuf798,18)+MUL_SHIFT32( A6 , _delayOutBuf799,18)), A11 ,18)+MUL_SHIFT32((MUL_SHIFT32( A8 , _delayOutBuf3447,18)+
MUL_SHIFT32( A6 , _delayOutBuf3446,18)), A9 ,18)+MUL_SHIFT32((MUL_SHIFT32( A6 , _delayOutBuf770,18)+MUL_SHIFT32( A8 , _delayOutBuf769,18)),
f3 ,18));
Pka21 = (MUL_SHIFT32((MUL_SHIFT32( A15 , _delayOutBuf876,18)+MUL_SHIFT32( A16 , _delayOutBuf877,18)+MUL_SHIFT32( A14 , _delayOutBuf865,18)+
MUL_SHIFT32( f4 , _delayOutBuf885,18)), A12 ,18)+MUL_SHIFT32((MUL_SHIFT32( A15 , _delayOutBuf898,18)+MUL_SHIFT32( A16 , _delayOutBuf897,18)+
MUL_SHIFT32( A14 , _delayOutBuf907,18)+MUL_SHIFT32( f4 , _delayOutBuf889,18)), A11 ,18)+MUL_SHIFT32((MUL_SHIFT32( A15 , _delayOutBuf921,18)+
MUL_SHIFT32( A16 , _delayOutBuf922,18)+MUL_SHIFT32( A14 , _delayOutBuf912,18)+MUL_SHIFT32( f4 , _delayOutBuf930,18)), f3 ,18)+MUL_SHIFT32(
(MUL_SHIFT32( A15 , _delayOutBuf3621,18)+MUL_SHIFT32( A16 , _delayOutBuf3620,18)+MUL_SHIFT32( A14 , _delayOutBuf3630,18)+MUL_SHIFT32(
f4 , _delayOutBuf3612,18)), A9 ,18));
L3 = (MUL_SHIFT32((MUL_SHIFT32( Pka7 , h1 ,18)+MUL_SHIFT32( Pka19 , h2 ,18)), h1 ,18)+MUL_SHIFT32((MUL_SHIFT32( Pka9 , h1 ,18)+
MUL_SHIFT32( Pka21 , h2 ,18)), h2 ,18));
Q1 = 393216 /* 1.5@fx14.32 */;
Pka1 = (MUL_SHIFT32((MUL_SHIFT32( A1 , _delayOutBuf280,18)+MUL_SHIFT32( A3 , _delayOutBuf279,18)), A1 ,18)+MUL_SHIFT32((MUL_SHIFT32(
A1 , _delayOutBuf296,18)+MUL_SHIFT32( A3 , _delayOutBuf297,18)), A3 ,18)+ Q1 );
Pka13 = (MUL_SHIFT32((MUL_SHIFT32( A11 , _delayOutBuf362,18)+MUL_SHIFT32( A12 , _delayOutBuf363,18)+MUL_SHIFT32( f3 , _delayOutBuf351,18)+
MUL_SHIFT32( A9 , _delayOutBuf3410,18)), A1 ,18)+MUL_SHIFT32((MUL_SHIFT32( A11 , _delayOutBuf337,18)+MUL_SHIFT32( A12 , _delayOutBuf336,18)+
MUL_SHIFT32( f3 , _delayOutBuf346,18)+MUL_SHIFT32( A9 , _delayOutBuf3415,18)), A3 ,18));
R1 = 26214 /* 0.1@fx14.32 */;
Pka3 = (MUL_SHIFT32((MUL_SHIFT32( A1 , _delayOutBuf744,18)+MUL_SHIFT32( A3 , _delayOutBuf745,18)), A11 ,18)+MUL_SHIFT32((MUL_SHIFT32(
A1 , _delayOutBuf729,18)+MUL_SHIFT32( A3 , _delayOutBuf728,18)), A12 ,18)+MUL_SHIFT32((MUL_SHIFT32( A1 , _delayOutBuf3434,18)+
MUL_SHIFT32( A3 , _delayOutBuf3435,18)), A9 ,18)+MUL_SHIFT32((MUL_SHIFT32( A1 , _delayOutBuf757,18)+MUL_SHIFT32( A3 , _delayOutBuf758,18)),
f3 ,18));
Q3 = 393216 /* 1.5@fx14.32 */;
Pka15 = (MUL_SHIFT32((MUL_SHIFT32( A11 , _delayOutBuf811,18)+MUL_SHIFT32( A12 , _delayOutBuf810,18)+MUL_SHIFT32( f3 , _delayOutBuf822,18)+
MUL_SHIFT32( A9 , _delayOutBuf3456,18)), A11 ,18)+MUL_SHIFT32((MUL_SHIFT32( A11 , _delayOutBuf836,18)+MUL_SHIFT32( A12 , _delayOutBuf837,18)+
MUL_SHIFT32( f3 , _delayOutBuf827,18)+MUL_SHIFT32( A9 , _delayOutBuf3461,18)), A12 ,18)+MUL_SHIFT32((MUL_SHIFT32( A11 , _delayOutBuf849,18)+
MUL_SHIFT32( A12 , _delayOutBuf848,18)+MUL_SHIFT32( f3 , _delayOutBuf858,18)+MUL_SHIFT32( A9 , _delayOutBuf3584,18)), f3 ,18)+ Q3 +
MUL_SHIFT32((MUL_SHIFT32( A11 , _delayOutBuf3603,18)+MUL_SHIFT32( A12 , _delayOutBuf3604,18)+MUL_SHIFT32( f3 , _delayOutBuf3594,18)+
MUL_SHIFT32( A9 , _delayOutBuf3589,18)), A9 ,18));
L1 = (MUL_SHIFT32((MUL_SHIFT32( Pka1 , h1 ,18)+MUL_SHIFT32( Pka13 , h2 ,18)), h1 ,18)+ R1 +MUL_SHIFT32((MUL_SHIFT32( Pka3 , h1 ,18)+
MUL_SHIFT32( Pka15 , h2 ,18)), h2 ,18));
Pka20 = (MUL_SHIFT32((MUL_SHIFT32( A15 , _delayOutBuf626,18)+MUL_SHIFT32( A16 , _delayOutBuf625,18)+MUL_SHIFT32( A14 , _delayOutBuf637,18)+
MUL_SHIFT32( f4 , _delayOutBuf617,18)), A6 ,18)+MUL_SHIFT32((MUL_SHIFT32( A15 , _delayOutBuf604,18)+MUL_SHIFT32( A16 , _delayOutBuf605,18)+
MUL_SHIFT32( A14 , _delayOutBuf595,18)+MUL_SHIFT32( f4 , _delayOutBuf613,18)), A8 ,18));
R2 = 26214 /* 0.1@fx14.32 */;
Q4 = 393216 /* 1.5@fx14.32 */;
Pka22 = (MUL_SHIFT32((MUL_SHIFT32( A15 , _delayOutBuf1173,18)+MUL_SHIFT32( A16 , _delayOutBuf1172,18)+MUL_SHIFT32( A14 , _delayOutBuf1184,18)+
```

```
      MUL_SHIFT32( f4 , _delayOutBuf1164,18)), A16 ,18)+MUL_SHIFT32((MUL_SHIFT32( A15 , _delayOutBuf1151,18)+MUL_SHIFT32( A16 , _delayOutBuf1152,18)+
      MUL_SHIFT32( A14 , _delayOutBuf1142,18)+MUL_SHIFT32( f4 , _delayOutBuf1160,18)), A14 ,18)+MUL_SHIFT32((MUL_SHIFT32( A15 , _delayOutBuf1128,18)+
      MUL_SHIFT32( A16 , _delayOutBuf1127,18)+MUL_SHIFT32( A14 , _delayOutBuf1137,18)+MUL_SHIFT32( f4 , _delayOutBuf1119,18)), A15 ,18)+
      MUL_SHIFT32((MUL_SHIFT32( A15 , _delayOutBuf1199,18)+MUL_SHIFT32( A16 , _delayOutBuf1200,18)+MUL_SHIFT32( A14 , _delayOutBuf1190,18)+
      MUL_SHIFT32( f4 , _delayOutBuf1208,18)), f4 ,18)+ Q4 );
L4 = (MUL_SHIFT32((MUL_SHIFT32( Pka8 , h1 ,18)+MUL_SHIFT32( Pka20 , h2 ,18)), h1 ,18)+ R2 +MUL_SHIFT32((MUL_SHIFT32( Pka10 , h1 ,18)+
      MUL_SHIFT32( Pka22 , h2 ,18)), h2 ,18));
Pka2 = (MUL_SHIFT32((MUL_SHIFT32( A1 , _delayOutBuf520,18)+MUL_SHIFT32( A3 , _delayOutBuf521,18)), A6 ,18)+MUL_SHIFT32((MUL_SHIFT32(
       A1 , _delayOutBuf505,18)+MUL_SHIFT32( A3 , _delayOutBuf504,18)), A8 ,18));
Pka14 = (MUL_SHIFT32((MUL_SHIFT32( A11 , _delayOutBuf586,18)+MUL_SHIFT32( A12 , _delayOutBuf587,18)+MUL_SHIFT32( f3 , _delayOutBuf575,18)+
      MUL_SHIFT32( A9 , _delayOutBuf3420,18)), A6 ,18)+MUL_SHIFT32((MUL_SHIFT32( A11 , _delayOutBuf561,18)+MUL_SHIFT32( A12 , _delayOutBuf560,18)+
      MUL_SHIFT32( f3 , _delayOutBuf570,18)+MUL_SHIFT32( A9 , _delayOutBuf3425,18)), A8 ,18));
Pka16 = (MUL_SHIFT32((MUL_SHIFT32( A11 , _delayOutBuf1092,18)+MUL_SHIFT32( A12 , _delayOutBuf1093,18)+MUL_SHIFT32( f3 , _delayOutBuf1081,18)+
      MUL_SHIFT32( A9 , _delayOutBuf3674,18)), A14 ,18)+MUL_SHIFT32((MUL_SHIFT32( A11 , _delayOutBuf1067,18)+MUL_SHIFT32( A12 , _delayOutBuf1066,18)+
      MUL_SHIFT32( f3 , _delayOutBuf1076,18)+MUL_SHIFT32( A9 , _delayOutBuf3679,18)), A15 ,18)+MUL_SHIFT32((MUL_SHIFT32( A11 , _delayOutBuf1054,18)+
      MUL_SHIFT32( A12 , _delayOutBuf1055,18)+MUL_SHIFT32( f3 , _delayOutBuf1045,18)+MUL_SHIFT32( A9 , _delayOutBuf3684,18)), A16 ,18)+
      MUL_SHIFT32((MUL_SHIFT32( A11 , _delayOutBuf1105,18)+MUL_SHIFT32( A12 , _delayOutBuf1104,18)+MUL_SHIFT32( f3 , _delayOutBuf1114,18)+
      MUL_SHIFT32( A9 , _delayOutBuf3689,18)), f4 ,18));
L2 = (MUL_SHIFT32((MUL_SHIFT32( Pka2 , h1 ,18)+MUL_SHIFT32( Pka14 , h2 ,18)), h1 ,18)+MUL_SHIFT32((MUL_SHIFT32( Pka4 , h1 ,18)+
      MUL_SHIFT32( Pka16 , h2 ,18)), h2 ,18));
det = (MUL_SHIFT32( L1 , L4 ,18)- MUL_SHIFT32( L2 , L3 ,18));
alpha3 = DIV_SHIFT32(MUL_SHIFT32( L3 ,-1073741824L,30)/* -1@fx14.32 */, det ,18,18);
alpha1 = DIV_SHIFT32( L4 , det ,18,18);
K3 = (MUL_SHIFT32((MUL_SHIFT32( Pka8 , h1 ,18)+MUL_SHIFT32( Pka10 , h2 ,18)), alpha3 ,18)+MUL_SHIFT32((MUL_SHIFT32( Pka7 , h1 ,18)+
      MUL_SHIFT32( Pka9 , h2 ,18)), alpha1 ,18));
alpha4 = DIV_SHIFT32( L1 , det ,18,18);
alpha2 = DIV_SHIFT32(MUL_SHIFT32( L2 ,-1073741824L,30)/* -1@fx14.32 */, det ,18,18);
K4 = (MUL_SHIFT32((MUL_SHIFT32( Pka8 , h1 ,18)+MUL_SHIFT32( Pka10 , h2 ,18)), alpha4 ,18)+MUL_SHIFT32((MUL_SHIFT32( Pka7 , h1 ,18)+
      MUL_SHIFT32( Pka9 , h2 ,18)), alpha2 ,18));
P10 = ( Pka10 - MUL_SHIFT32(MUL_SHIFT32( Pka4 , h1 ,18), K3 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka16 , h2 ,18), K3 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka10 , h1 ,18), K4 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka22 , h2 ,18), K4 ,18));
K1 = (MUL_SHIFT32((MUL_SHIFT32( Pka1 , h1 ,18)+MUL_SHIFT32( Pka3 , h2 ,18)), alpha1 ,18)+MUL_SHIFT32((MUL_SHIFT32( Pka2 , h1 ,18)+
      MUL_SHIFT32( Pka4 , h2 ,18)), alpha3 ,18));
K2 = (MUL_SHIFT32((MUL_SHIFT32( Pka1 , h1 ,18)+MUL_SHIFT32( Pka3 , h2 ,18)), alpha2 ,18)+MUL_SHIFT32((MUL_SHIFT32( Pka2 , h1 ,18)+
      MUL_SHIFT32( Pka4 , h2 ,18)), alpha4 ,18));
P1 = ( Pka1 - MUL_SHIFT32(MUL_SHIFT32( Pka1 , h1 ,18), K1 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka13 , h2 ,18), K1 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka7 , h1 ,18), K2 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka19 , h2 ,18), K2 ,18));
K7 = (MUL_SHIFT32((MUL_SHIFT32( Pka19 , h1 ,18)+MUL_SHIFT32( Pka21 , h2 ,18)), alpha1 ,18)+MUL_SHIFT32((MUL_SHIFT32( Pka20 , h1 ,18)+
      MUL_SHIFT32( Pka22 , h2 ,18)), alpha3 ,18));
K8 = (MUL_SHIFT32((MUL_SHIFT32( Pka19 , h1 ,18)+MUL_SHIFT32( Pka21 , h2 ,18)), alpha2 ,18)+MUL_SHIFT32((MUL_SHIFT32( Pka20 , h1 ,18)+
      MUL_SHIFT32( Pka22 , h2 ,18)), alpha4 ,18));
P22 = ( Pka22 - MUL_SHIFT32(MUL_SHIFT32( Pka4 , h1 ,18), K7 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka16 , h2 ,18), K7 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka10 , h1 ,18), K8 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka22 , h2 ,18), K8 ,18));
P2 = ( Pka2 - MUL_SHIFT32(MUL_SHIFT32( Pka2 , h1 ,18), K1 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka14 , h2 ,18), K1 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka8 , h1 ,18), K2 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka20 , h2 ,18), K2 ,18));
K5 = (MUL_SHIFT32((MUL_SHIFT32( Pka13 , h1 ,18)+MUL_SHIFT32( Pka15 , h2 ,18)), alpha1 ,18)+MUL_SHIFT32((MUL_SHIFT32( Pka14 , h1 ,18)+
      MUL_SHIFT32( Pka16 , h2 ,18)), alpha3 ,18));
K6 = (MUL_SHIFT32((MUL_SHIFT32( Pka13 , h1 ,18)+MUL_SHIFT32( Pka15 , h2 ,18)), alpha2 ,18)+MUL_SHIFT32((MUL_SHIFT32( Pka14 , h1 ,18)+
      MUL_SHIFT32( Pka16 , h2 ,18)), alpha4 ,18));
P14 = ( Pka14 - MUL_SHIFT32(MUL_SHIFT32( Pka2 , h1 ,18), K5 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka14 , h2 ,18), K5 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka8 , h1 ,18), K6 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka20 , h2 ,18), K6 ,18));
P3 = ( Pka3 - MUL_SHIFT32(MUL_SHIFT32( Pka3 , h1 ,18), K1 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka15 , h2 ,18), K1 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka9 , h1 ,18), K2 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka21 , h2 ,18), K2 ,18));
P15 = ( Pka15 - MUL_SHIFT32(MUL_SHIFT32( Pka3 , h1 ,18), K5 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka15 , h2 ,18), K5 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka9 , h1 ,18), K6 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka21 , h2 ,18), K6 ,18));
P20 = ( Pka20 - MUL_SHIFT32(MUL_SHIFT32( Pka2 , h1 ,18), K7 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka14 , h2 ,18), K7 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka8 , h1 ,18), K8 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka20 , h2 ,18), K8 ,18));
P8 = ( Pka8 - MUL_SHIFT32(MUL_SHIFT32( Pka2 , h1 ,18), K3 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka14 , h2 ,18), K3 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka8 , h1 ,18), K4 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka20 , h2 ,18), K4 ,18));
clarke3377.as = MUL_SHIFT32((((long)(f280xReadAnalogVal(8))<<12)+-28487712 /* -1.698@fx8.32 */),2030043136L,26)/* 30.25@fx8.32 */;
clarke3377.bs = MUL_SHIFT32((((long)(f280xReadAnalogVal(9))<<12)+-28123647 /* -1.6763@fx8.32 */),2049504706L,26)/* 30.54@fx8.32 */;
clarke_calc(&clarke3377);
```

```
ia = (( clarke3377.ds)>>6);
i1 =  _delayOutBuf3728;
b = 98825928 /* 376.991@fx14.32 */;
t167 = MUL_SHIFT32( b , Ts ,18);
B1 =  t167;
v1 =  _delayOutBuf3742;
x1_k1 = (MUL_SHIFT32( A1 , _delayOutBuf206,18)+MUL_SHIFT32( A3 , _delayOutBuf207,18)+MUL_SHIFT32( B1 , v1 ,18));
x3_k1 = (MUL_SHIFT32( A9 , _delayOutBuf210,18)+MUL_SHIFT32( A11 , _delayOutBuf211,18)+MUL_SHIFT32( A12 , _delayOutBuf212,18));
beta1 = ( i1 - MUL_SHIFT32( x1_k1 , h1 ,18)- MUL_SHIFT32( x3_k1 , h2 ,18));
i2 =  _delayOutBuf3734;
B4 =  t167;
v2 =  _delayOutBuf3737;
x2_k1 = (MUL_SHIFT32( A6 , _delayOutBuf208,18)+MUL_SHIFT32( A8 , _delayOutBuf209,18)+MUL_SHIFT32( B4 , v2 ,18));
x4_k1 = (MUL_SHIFT32( A14 , _delayOutBuf237,18)+MUL_SHIFT32( A15 , _delayOutBuf236,18)+MUL_SHIFT32( A16 , _delayOutBuf235,18));
beta2 = ( i2 - MUL_SHIFT32( x2_k1 , h1 ,18)- MUL_SHIFT32( x4_k1 , h2 ,18));
x1 = (MUL_SHIFT32( K1 , beta1 ,18)+ x1_k1 +MUL_SHIFT32( K2 , beta2 ,18));
x3 = (MUL_SHIFT32( K5 , beta1 ,18)+ x3_k1 +MUL_SHIFT32( K6 , beta2 ,18));
clarke3378.as = (((long)(f280xReadAnalogVal(0))<<12)+-19250177 /* -1.1474@fx8.32 */);
clarke3378.bs = (((long)(f280xReadAnalogVal(1))<<12)+-20181313 /* -1.2029@fx8.32 */);
clarke_calc(&clarke3378);
va = MUL_SHIFT32((( clarke3378.ds)>>6),1994014064L,23)/* 237.705@fx14.32 */;
P13 = ( Pka13 - MUL_SHIFT32(MUL_SHIFT32( Pka1 , h1 ,18), K5 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka13 , h2 ,18), K5 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka7 , h1 ,18), K6 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka19 , h2 ,18), K6 ,18));
P19 = ( Pka19 - MUL_SHIFT32(MUL_SHIFT32( Pka1 , h1 ,18), K7 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka13 , h2 ,18), K7 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka7 , h1 ,18), K8 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka19 , h2 ,18), K8 ,18));
P16 = ( Pka16 - MUL_SHIFT32(MUL_SHIFT32( Pka4 , h1 ,18), K5 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka16 , h2 ,18), K5 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka10 , h1 ,18), K6 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka22 , h2 ,18), K6 ,18));
P4 = ( Pka4 - MUL_SHIFT32(MUL_SHIFT32( Pka4 , h1 ,18), K1 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka16 , h2 ,18), K1 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka10 , h1 ,18), K2 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka22 , h2 ,18), K2 ,18));
Pka6 = (MUL_SHIFT32( A1 , _delayOutBuf1631,18)+MUL_SHIFT32( A3 , _delayOutBuf1632,18));
Pka18 = (MUL_SHIFT32( f3 , _delayOutBuf1655,18)+MUL_SHIFT32( A11 , _delayOutBuf1656,18)+MUL_SHIFT32( A12 , _delayOutBuf1664,18)+
      MUL_SHIFT32( A9 , _delayOutBuf3724,18));
Pka12 = (MUL_SHIFT32( A6 , _delayOutBuf1644,18)+MUL_SHIFT32( A8 , _delayOutBuf1643,18));
Pka24 = (MUL_SHIFT32( f4 , _delayOutBuf1678,18)+MUL_SHIFT32( A14 , _delayOutBuf1677,18)+MUL_SHIFT32( A15 , _delayOutBuf1669,18)+
      MUL_SHIFT32( A16 , _delayOutBuf1686,18));
P6 = ( Pka6 - MUL_SHIFT32(MUL_SHIFT32( Pka6 , h1 ,18), K1 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka18 , h2 ,18), K1 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka12 , h1 ,18), K2 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka24 , h2 ,18), K2 ,18));
x4 = (MUL_SHIFT32( K7 , beta1 ,18)+ x4_k1 +MUL_SHIFT32( K8 , beta2 ,18));
z1 = 262091 /* 0.9998@fx14.32 */;
Gamma = 23 /* 8.84e-005@fx14.32 */;
A53 = MUL_SHIFT32(MUL_SHIFT32( Gamma , cs2 ,18), _delayOutBuf175,18);
A54 = MUL_SHIFT32(MUL_SHIFT32( Gamma , cs2 ,18),MUL_SHIFT32( _delayOutBuf176,-1073741824L,30)/* -1@fx14.32 */,18);
A56 = MUL_SHIFT32(( z1 +-262144 /* -1@fx14.32 */),20971520 /* 80@fx14.32 */,18);
Pka30 = (MUL_SHIFT32( z1 , _delayOutBuf1699,18)+MUL_SHIFT32((0.*0x40000), _delayOutBuf1700,18)+MUL_SHIFT32( A53 , _delayOutBuf1708,18)+
      MUL_SHIFT32( A54 , _delayOutBuf1691,18)+MUL_SHIFT32((0.*0x40000), _delayOutBuf1713,18)+MUL_SHIFT32( A56 , _delayOutBuf1718,18));
Pka25 = (MUL_SHIFT32((MUL_SHIFT32((0.*0x40000), _delayOutBuf486,18)+MUL_SHIFT32( A53 , _delayOutBuf487,18)+MUL_SHIFT32( A54 , _delayOutBuf475,18)+
      MUL_SHIFT32( A56 , _delayOutBuf495,18)+MUL_SHIFT32((0.*0x40000), _delayOutBuf470,18)+MUL_SHIFT32( z1 , _delayOutBuf465,18)), A1 ,18)+
      MUL_SHIFT32((MUL_SHIFT32((0.*0x40000), _delayOutBuf441,18)+MUL_SHIFT32( A53 , _delayOutBuf440,18)+MUL_SHIFT32( A54 , _delayOutBuf450,18)+
      MUL_SHIFT32( A56 , _delayOutBuf432,18)+MUL_SHIFT32((0.*0x40000), _delayOutBuf455,18)+MUL_SHIFT32( z1 , _delayOutBuf460,18)), A3 ,18));
Pka27 = (MUL_SHIFT32((MUL_SHIFT32((0.*0x40000), _delayOutBuf3525,18)+MUL_SHIFT32( A53 , _delayOutBuf3524,18)+MUL_SHIFT32( A54 ,
       _delayOutBuf3536,18)+MUL_SHIFT32( A56 , _delayOutBuf3516,18)+MUL_SHIFT32((0.*0x40000), _delayOutBuf3541,18)+MUL_SHIFT32( z1 , _delayOutBuf3546,18)),
       f3 ,18)+MUL_SHIFT32((MUL_SHIFT32((0.*0x40000), _delayOutBuf3570,18)+MUL_SHIFT32( A53 , _delayOutBuf3571,18)+MUL_SHIFT32( A54 ,
       _delayOutBuf3561,18)+MUL_SHIFT32( A56 , _delayOutBuf3579,18)+MUL_SHIFT32((0.*0x40000), _delayOutBuf3556,18)+MUL_SHIFT32( z1 , _delayOutBuf3551,18)),
       A12 ,18)+MUL_SHIFT32((MUL_SHIFT32((0.*0x40000), _delayOutBuf3502,18)+MUL_SHIFT32( A53 , _delayOutBuf3503,18)+MUL_SHIFT32( A54 ,
       _delayOutBuf3493,18)+MUL_SHIFT32( A56 , _delayOutBuf3511,18)+MUL_SHIFT32((0.*0x40000), _delayOutBuf3488,18)+MUL_SHIFT32( z1 , _delayOutBuf3483,18)),
       A11 ,18)+MUL_SHIFT32((MUL_SHIFT32((0.*0x40000), _delayOutBuf3645,18)+MUL_SHIFT32( A53 , _delayOutBuf3644,18)+MUL_SHIFT32( A54 ,
       _delayOutBuf3654,18)+MUL_SHIFT32( A56 , _delayOutBuf3636,18)+MUL_SHIFT32((0.*0x40000), _delayOutBuf3659,18)+MUL_SHIFT32( z1 , _delayOutBuf3664,18)),
       A9 ,18));
Pka26 = (MUL_SHIFT32((MUL_SHIFT32((0.*0x40000), _delayOutBuf652,18)+MUL_SHIFT32( A53 , _delayOutBuf651,18)+MUL_SHIFT32( A54 , _delayOutBuf663,18)+
      MUL_SHIFT32( A56 , _delayOutBuf643,18)+MUL_SHIFT32((0.*0x40000), _delayOutBuf668,18)+MUL_SHIFT32( z1 , _delayOutBuf673,18)), A6 ,18)+
      MUL_SHIFT32((MUL_SHIFT32((0.*0x40000), _delayOutBuf697,18)+MUL_SHIFT32( A53 , _delayOutBuf698,18)+MUL_SHIFT32( A54 , _delayOutBuf688,18)+
      MUL_SHIFT32( A56 , _delayOutBuf706,18)+MUL_SHIFT32((0.*0x40000), _delayOutBuf683,18)+MUL_SHIFT32( z1 , _delayOutBuf678,18)), A8 ,18));
```

```
Pka28 = (MUL_SHIFT32((MUL_SHIFT32((0.*0x40000), _delayOutBuf1839,18)+MUL_SHIFT32( A53 , _delayOutBuf1840,18)+MUL_SHIFT32( A54 ,
        _delayOutBuf1828,18)+MUL_SHIFT32( A56 , _delayOutBuf1848,18)+MUL_SHIFT32((0.*0x40000), _delayOutBuf1823,18)+MUL_SHIFT32( z1 , _delayOutBuf1818,18)),
        f4 ,18)+MUL_SHIFT32((MUL_SHIFT32((0.*0x40000), _delayOutBuf1794,18)+MUL_SHIFT32( A53 , _delayOutBuf1793,18)+MUL_SHIFT32( A54 ,
        _delayOutBuf1803,18)+MUL_SHIFT32( A56 , _delayOutBuf1785,18)+MUL_SHIFT32((0.*0x40000), _delayOutBuf1808,18)+MUL_SHIFT32( z1 , _delayOutBuf1813,18)),
        A16 ,18)+MUL_SHIFT32((MUL_SHIFT32((0.*0x40000), _delayOutBuf1862,18)+MUL_SHIFT32( A53 , _delayOutBuf1861,18)+MUL_SHIFT32( A54 ,
        _delayOutBuf1871,18)+MUL_SHIFT32( A56 , _delayOutBuf1853,18)+MUL_SHIFT32((0.*0x40000), _delayOutBuf1876,18)+MUL_SHIFT32( z1 , _delayOutBuf1881,18)),
        A14 ,18)+MUL_SHIFT32((MUL_SHIFT32((0.*0x40000), _delayOutBuf1771,18)+MUL_SHIFT32( A53 , _delayOutBuf1772,18)+MUL_SHIFT32( A54 ,
        _delayOutBuf1762,18)+MUL_SHIFT32( A56 , _delayOutBuf1780,18)+MUL_SHIFT32((0.*0x40000), _delayOutBuf1757,18)+MUL_SHIFT32( z1 , _delayOutBuf1752,18)),
        A15 ,18));
K9 = (MUL_SHIFT32((MUL_SHIFT32( Pka25 , h1 ,18)+MUL_SHIFT32( Pka27 , h2 ,18)), alpha1 ,18)+MUL_SHIFT32((MUL_SHIFT32( Pka26 , h1 ,18)+
     MUL_SHIFT32( Pka28 , h2 ,18)), alpha3 ,18));
K10 = (MUL_SHIFT32((MUL_SHIFT32( Pka25 , h1 ,18)+MUL_SHIFT32( Pka27 , h2 ,18)), alpha2 ,18)+MUL_SHIFT32((MUL_SHIFT32( Pka26 , h1 ,18)+
     MUL_SHIFT32( Pka28 , h2 ,18)), alpha4 ,18));
P30 = ( Pka30 - MUL_SHIFT32(MUL_SHIFT32( Pka6 , h1 ,18), K9 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka18 , h2 ,18), K9 ,18)- MUL_SHIFT32(
     MUL_SHIFT32( Pka12 , h1 ,18), K10 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka24 , h2 ,18), K10 ,18));
P12 = ( Pka12 - MUL_SHIFT32(MUL_SHIFT32( Pka6 , h1 ,18), K3 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka18 , h2 ,18), K3 ,18)- MUL_SHIFT32(
     MUL_SHIFT32( Pka12 , h1 ,18), K4 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka24 , h2 ,18), K4 ,18));
P18 = ( Pka18 - MUL_SHIFT32(MUL_SHIFT32( Pka6 , h1 ,18), K5 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka18 , h2 ,18), K5 ,18)- MUL_SHIFT32(
     MUL_SHIFT32( Pka12 , h1 ,18), K6 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka24 , h2 ,18), K6 ,18));
P24 = ( Pka24 - MUL_SHIFT32(MUL_SHIFT32( Pka6 , h1 ,18), K7 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka18 , h2 ,18), K7 ,18)- MUL_SHIFT32(
     MUL_SHIFT32( Pka12 , h1 ,18), K8 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka24 , h2 ,18), K8 ,18));
Q6 = 26 /* 0.0001@fx14.32 */;
Pka36 = ( _delayOutBuf1723+ Q6 );
Pka31 = (MUL_SHIFT32( A1 , _delayOutBuf424,18)+MUL_SHIFT32( A3 , _delayOutBuf423,18));
Pka33 = (MUL_SHIFT32( A11 , _delayOutBuf3474,18)+MUL_SHIFT32( A12 , _delayOutBuf3475,18)+MUL_SHIFT32( f3 , _delayOutBuf3466,18)+
     MUL_SHIFT32( A9 , _delayOutBuf3669,18));
Pka32 = (MUL_SHIFT32( A6 , _delayOutBuf714,18)+MUL_SHIFT32( A8 , _delayOutBuf715,18));
Pka34 = (MUL_SHIFT32( A14 , _delayOutBuf1739,18)+MUL_SHIFT32( A15 , _delayOutBuf1738,18)+MUL_SHIFT32( A16 , _delayOutBuf1747,18)+
     MUL_SHIFT32( f4 , _delayOutBuf1730,18));
K11 = (MUL_SHIFT32((MUL_SHIFT32( Pka31 , h1 ,18)+MUL_SHIFT32( Pka33 , h2 ,18)), alpha1 ,18)+MUL_SHIFT32((MUL_SHIFT32( Pka32 , h1 ,18)+
     MUL_SHIFT32( Pka34 , h2 ,18)), alpha3 ,18));
K12 = (MUL_SHIFT32((MUL_SHIFT32( Pka31 , h1 ,18)+MUL_SHIFT32( Pka33 , h2 ,18)), alpha2 ,18)+MUL_SHIFT32((MUL_SHIFT32( Pka32 , h1 ,18)+
     MUL_SHIFT32( Pka34 , h2 ,18)), alpha4 ,18));
P36 = ( Pka36 - MUL_SHIFT32(MUL_SHIFT32( Pka6 , h1 ,18), K11 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka18 , h2 ,18), K11 ,18)- MUL_SHIFT32(
     MUL_SHIFT32( Pka12 , h1 ,18), K12 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka24 , h2 ,18), K12 ,18));
P33 = ( Pka33 - MUL_SHIFT32(MUL_SHIFT32( Pka3 , h1 ,18), K11 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka15 , h2 ,18), K11 ,18)- MUL_SHIFT32(
     MUL_SHIFT32( Pka9 , h1 ,18), K12 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka21 , h2 ,18), K12 ,18));
P32 = ( Pka32 - MUL_SHIFT32(MUL_SHIFT32( Pka2 , h1 ,18), K11 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka14 , h2 ,18), K11 ,18)- MUL_SHIFT32(
     MUL_SHIFT32( Pka8 , h1 ,18), K12 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka20 , h2 ,18), K12 ,18));
P34 = ( Pka34 - MUL_SHIFT32(MUL_SHIFT32( Pka4 , h1 ,18), K11 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka16 , h2 ,18), K11 ,18)- MUL_SHIFT32(
     MUL_SHIFT32( Pka10 , h1 ,18), K12 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka22 , h2 ,18), K12 ,18));
P31 = ( Pka31 - MUL_SHIFT32(MUL_SHIFT32( Pka1 , h1 ,18), K11 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka13 , h2 ,18), K11 ,18)- MUL_SHIFT32(
     MUL_SHIFT32( Pka7 , h1 ,18), K12 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka19 , h2 ,18), K12 ,18));
Pka35 = (MUL_SHIFT32((0.*0x40000), _delayOutBuf1905,18)+MUL_SHIFT32((0.*0x40000), _delayOutBuf1906,18)+MUL_SHIFT32( A53 , _delayOutBuf1897,18)+
     MUL_SHIFT32( A54 , _delayOutBuf1914,18)+MUL_SHIFT32( A56 , _delayOutBuf1892,18)+MUL_SHIFT32( z1 , _delayOutBuf1887,18));
Pka5 = (MUL_SHIFT32((MUL_SHIFT32( A1 , _delayOutBuf1247,18)+MUL_SHIFT32( A3 , _delayOutBuf1248,18)),(0.*0x40000),18)+MUL_SHIFT32(
     (MUL_SHIFT32( A1 , _delayOutBuf1232,18)+MUL_SHIFT32( A3 , _delayOutBuf1231,18)),(0.*0x40000),18)+MUL_SHIFT32((MUL_SHIFT32( A1 ,
      _delayOutBuf1260,18)+MUL_SHIFT32( A3 , _delayOutBuf1261,18)), A53 ,18)+MUL_SHIFT32((MUL_SHIFT32( A1 , _delayOutBuf1219,18)+MUL_SHIFT32(
      A3 , _delayOutBuf1218,18)), A54 ,18)+MUL_SHIFT32((MUL_SHIFT32( A1 , _delayOutBuf1273,18)+MUL_SHIFT32( A3 , _delayOutBuf1274,18)),
      A56 ,18)+MUL_SHIFT32((MUL_SHIFT32( A1 , _delayOutBuf1286,18)+MUL_SHIFT32( A3 , _delayOutBuf1287,18)), z1 ,18));
Pka17 = (MUL_SHIFT32((MUL_SHIFT32( A11 , _delayOutBuf1397,18)+MUL_SHIFT32( A12 , _delayOutBuf1396,18)+MUL_SHIFT32( A9 , _delayOutBuf1408,18)+
     MUL_SHIFT32( f3 , _delayOutBuf3694,18)), A56 ,18)+MUL_SHIFT32((MUL_SHIFT32( A11 , _delayOutBuf1422,18)+MUL_SHIFT32( A12 , _delayOutBuf1423,18)+
     MUL_SHIFT32( f3 , _delayOutBuf1413,18)+MUL_SHIFT32( A9 , _delayOutBuf3699,18)),(0.*0x40000),18)+MUL_SHIFT32((MUL_SHIFT32( A11 ,
      _delayOutBuf1435,18)+MUL_SHIFT32( A12 , _delayOutBuf1434,18)+MUL_SHIFT32( f3 , _delayOutBuf1444,18)
     +MUL_SHIFT32( A9 , _delayOutBuf3704,18)),(0.*0x40000),18)+
     MUL_SHIFT32((MUL_SHIFT32( A11 , _delayOutBuf1384,18)+MUL_SHIFT32( A12 , _delayOutBuf1385,18)+MUL_SHIFT32( f3 , _delayOutBuf1375,18)+
     MUL_SHIFT32( A9 , _delayOutBuf3709,18)), A53 ,18)+MUL_SHIFT32((MUL_SHIFT32( A11 , _delayOutBuf1458,18)+MUL_SHIFT32( A12 , _delayOutBuf1459,18)+
     MUL_SHIFT32( f3 , _delayOutBuf1449,18)+MUL_SHIFT32( A9 , _delayOutBuf3714,18)), A54 ,18)+MUL_SHIFT32((MUL_SHIFT32( A11 , _delayOutBuf1471,18)+
     MUL_SHIFT32( A12 , _delayOutBuf1470,18)+MUL_SHIFT32( f3 , _delayOutBuf1480,18)+MUL_SHIFT32( A9 , _delayOutBuf3719,18)), z1 ,18));
Pka11 = (MUL_SHIFT32((MUL_SHIFT32( A6 , _delayOutBuf1325,18)+MUL_SHIFT32( A8 , _delayOutBuf1324,18)), A54 ,18)+MUL_SHIFT32((MUL_SHIFT32(
      A8 , _delayOutBuf1340,18)+MUL_SHIFT32( A6 , _delayOutBuf1341,18)), A56 ,18)+MUL_SHIFT32((MUL_SHIFT32( A6 , _delayOutBuf1312,18)+
     MUL_SHIFT32( A8 , _delayOutBuf1311,18)),(0.*0x40000),18)+MUL_SHIFT32((MUL_SHIFT32( A6 , _delayOutBuf1299,18)
```

```
      +MUL_SHIFT32( A8 , _delayOutBuf1300,18)),(0.*0x40000),18)+
      MUL_SHIFT32((MUL_SHIFT32( A6 , _delayOutBuf1353,18)+MUL_SHIFT32( A8 , _delayOutBuf1354,18)), A53 ,18)+MUL_SHIFT32((MUL_SHIFT32(
       A6 , _delayOutBuf1366,18)+MUL_SHIFT32( A8 , _delayOutBuf1365,18)), z1 ,18));
Pka23 = (MUL_SHIFT32((MUL_SHIFT32( A15 , _delayOutBuf1522,18)+MUL_SHIFT32( A16 , _delayOutBuf1523,18)+MUL_SHIFT32( A14 , _delayOutBuf1511,18)+
      MUL_SHIFT32( f4 , _delayOutBuf1531,18)), A56 ,18)+MUL_SHIFT32((MUL_SHIFT32( A15 , _delayOutBuf1544,18)+MUL_SHIFT32( A16 , _delayOutBuf1543,18)+
      MUL_SHIFT32( A14 , _delayOutBuf1553,18)+MUL_SHIFT32( f4 , _delayOutBuf1535,18)), A54 ,18)+MUL_SHIFT32((MUL_SHIFT32( A15 , _delayOutBuf1567,18)+
      MUL_SHIFT32( A16 , _delayOutBuf1568,18)+MUL_SHIFT32( A14 , _delayOutBuf1558,18)+MUL_SHIFT32( f4 , _delayOutBuf1576,18)),(0.*0x40000),18)+
      MUL_SHIFT32((MUL_SHIFT32( A15 , _delayOutBuf1496,18)+MUL_SHIFT32( A16 , _delayOutBuf1495,18)+MUL_SHIFT32( A14 , _delayOutBuf1505,18)+
      MUL_SHIFT32( f4 , _delayOutBuf1487,18)),(0.*0x40000),18)+MUL_SHIFT32((MUL_SHIFT32( A15 , _delayOutBuf1590,18)+MUL_SHIFT32( A16 ,
       _delayOutBuf1589,18)+MUL_SHIFT32( A14 , _delayOutBuf1599,18)+MUL_SHIFT32( f4 , _delayOutBuf1581,18)), A53 ,18)+MUL_SHIFT32((MUL_SHIFT32(
       A15 , _delayOutBuf1613,18)+MUL_SHIFT32( A16 , _delayOutBuf1614,18)+MUL_SHIFT32( A14 , _delayOutBuf1604,18)+MUL_SHIFT32( f4 , _delayOutBuf1622,18)),
       z1 ,18));
P35 = ( Pka35 - MUL_SHIFT32(MUL_SHIFT32( Pka5 , h1 ,18), K11 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka17 , h2 ,18), K11 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka11 , h1 ,18), K12 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka23 , h2 ,18), K12 ,18));
P5 = ( Pka5 - MUL_SHIFT32(MUL_SHIFT32( Pka5 , h1 ,18), K1 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka17 , h2 ,18), K1 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka11 , h1 ,18), K2 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka23 , h2 ,18), K2 ,18));
P7 = ( Pka7 - MUL_SHIFT32(MUL_SHIFT32( Pka1 , h1 ,18), K3 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka13 , h2 ,18), K3 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka7 , h1 ,18), K4 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka19 , h2 ,18), K4 ,18));
P9 = ( Pka9 - MUL_SHIFT32(MUL_SHIFT32( Pka3 , h1 ,18), K3 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka15 , h2 ,18), K3 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka9 , h1 ,18), K4 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka21 , h2 ,18), K4 ,18));
P21 = ( Pka21 - MUL_SHIFT32(MUL_SHIFT32( Pka3 , h1 ,18), K7 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka15 , h2 ,18), K7 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka9 , h1 ,18), K8 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka21 , h2 ,18), K8 ,18));
P28 = ( Pka28 - MUL_SHIFT32(MUL_SHIFT32( Pka4 , h1 ,18), K9 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka16 , h2 ,18), K9 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka10 , h1 ,18), K10 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka22 , h2 ,18), K10 ,18));
P25 = ( Pka25 - MUL_SHIFT32(MUL_SHIFT32( Pka1 , h1 ,18), K9 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka13 , h2 ,18), K9 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka7 , h1 ,18), K10 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka19 , h2 ,18), K10 ,18));
P26 = ( Pka26 - MUL_SHIFT32(MUL_SHIFT32( Pka2 , h1 ,18), K9 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka14 , h2 ,18), K9 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka8 , h1 ,18), K10 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka20 , h2 ,18), K10 ,18));
P27 = ( Pka27 - MUL_SHIFT32(MUL_SHIFT32( Pka3 , h1 ,18), K9 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka15 , h2 ,18), K9 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka9 , h1 ,18), K10 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka21 , h2 ,18), K10 ,18));
P17 = ( Pka17 - MUL_SHIFT32(MUL_SHIFT32( Pka5 , h1 ,18), K5 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka17 , h2 ,18), K5 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka11 , h1 ,18), K6 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka23 , h2 ,18), K6 ,18));
P23 = ( Pka23 - MUL_SHIFT32(MUL_SHIFT32( Pka5 , h1 ,18), K7 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka17 , h2 ,18), K7 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka11 , h1 ,18), K8 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka23 , h2 ,18), K8 ,18));
P11 = ( Pka11 - MUL_SHIFT32(MUL_SHIFT32( Pka5 , h1 ,18), K3 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka17 , h2 ,18), K3 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka11 , h1 ,18), K4 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka23 , h2 ,18), K4 ,18));
Q5 = 2621 /* 0.01@fx14.32 */;
Pka29 = (MUL_SHIFT32((MUL_SHIFT32((0.*0x40000), _delayOutBuf2028,18)+MUL_SHIFT32( A53 , _delayOutBuf2027,18)+MUL_SHIFT32( A54 ,
       _delayOutBuf2039,18)+MUL_SHIFT32( A56 , _delayOutBuf2019,18)+MUL_SHIFT32((0.*0x40000), _delayOutBuf2044,18)+MUL_SHIFT32( z1 , _delayOutBuf2049,18)),
       A56 ,18)+MUL_SHIFT32((MUL_SHIFT32((0.*0x40000), _delayOutBuf2073,18)+MUL_SHIFT32( A53 , _delayOutBuf2074,18)+MUL_SHIFT32( A54 ,
       _delayOutBuf2064,18)+MUL_SHIFT32( A56 , _delayOutBuf2082,18)+MUL_SHIFT32((0.*0x40000), _delayOutBuf2059,18)+MUL_SHIFT32( z1 , _delayOutBuf2054,18)),
       A54 ,18)+MUL_SHIFT32((MUL_SHIFT32((0.*0x40000), _delayOutBuf2005,18)+MUL_SHIFT32( A53 , _delayOutBuf2006,18)+MUL_SHIFT32( A54 ,
       _delayOutBuf1996,18)+MUL_SHIFT32( A56 , _delayOutBuf2014,18)+MUL_SHIFT32((0.*0x40000), _delayOutBuf1991,18)
      +MUL_SHIFT32( z1 , _delayOutBuf1986,18)),(0.*0x40000),18)+
      MUL_SHIFT32((MUL_SHIFT32((0.*0x40000), _delayOutBuf2096,18)+MUL_SHIFT32( A53 , _delayOutBuf2095,18)+MUL_SHIFT32( A54 , _delayOutBuf2105,18)+
      MUL_SHIFT32( A56 , _delayOutBuf2087,18)+MUL_SHIFT32((0.*0x40000), _delayOutBuf2110,18)+MUL_SHIFT32( z1 , _delayOutBuf2115,18)),(0.*0x40000),18)+
      MUL_SHIFT32((MUL_SHIFT32((0.*0x40000), _delayOutBuf1962,18)+MUL_SHIFT32( A53 , _delayOutBuf1961,18)+MUL_SHIFT32( A54 , _delayOutBuf1971,18)+
      MUL_SHIFT32( A56 , _delayOutBuf1953,18)+MUL_SHIFT32((0.*0x40000), _delayOutBuf1976,18)+MUL_SHIFT32( z1 , _delayOutBuf1981,18)),
       A53 ,18)+MUL_SHIFT32((MUL_SHIFT32((0.*0x40000), _delayOutBuf1939,18)+MUL_SHIFT32( A53 , _delayOutBuf1940,18)+MUL_SHIFT32( A54 ,
       _delayOutBuf1930,18)+MUL_SHIFT32( A56 , _delayOutBuf1948,18)+MUL_SHIFT32((0.*0x40000), _delayOutBuf1925,18)+MUL_SHIFT32( z1 , _delayOutBuf1920,18)),
       z1 ,18)+ Q5 );
P29 = ( Pka29 - MUL_SHIFT32(MUL_SHIFT32( Pka5 , h1 ,18), K9 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka17 , h2 ,18), K9 ,18)- MUL_SHIFT32(
      MUL_SHIFT32( Pka11 , h1 ,18), K10 ,18)- MUL_SHIFT32(MUL_SHIFT32( Pka23 , h2 ,18), K10 ,18));
x2 = (MUL_SHIFT32( K3 , beta1 ,18)+ x2_k1 +MUL_SHIFT32( K4 , beta2 ,18));
ib = (( clarke3377.qs)>>6);
vb = MUL_SHIFT32((( clarke3378.qs)>>6),1088421888L,22)/* 259.5@fx14.32 */;
x6_k1 =  _delayOutBuf269;
x6 = (MUL_SHIFT32( K11 , beta1 ,18)+ x6_k1 +MUL_SHIFT32( K12 , beta2 ,18));
x5_k1 = (MUL_SHIFT32( A53 , _delayOutBuf253,18)+MUL_SHIFT32( A54 , _delayOutBuf254,18)+MUL_SHIFT32( A56 , _delayOutBuf255,18)+
      MUL_SHIFT32( z1 , _delayOutBuf268,18));
t59 = (MUL_SHIFT32( K9 , beta1 ,18)+ x5_k1 +MUL_SHIFT32( K10 , beta2 ,18));
x5 =  t59;
```

```
  t3344 = MUL_SHIFT32(((long)(f280xReadAnalogVal(7))<<12),1342177280L,26)/* 20@fx8.32 */;
  t3344 = MIN(1090519040L,t3344);
  t3344 = MAX(t3344,0);
;
  t31 = ( _sampBuf27+MUL_SHIFT32(3355 /* 0.0002@fx8.32 */, t3344,24));
  t29 = ((16777216 /* 1@fx8.32 */< t31)?0 /* 0@fx8.32 */: t31);
  t8 = 8388608 /* 0.5@fx8.32 */;
  t24 = (( _sampBuf27)<<7);
  invClarke22.ds = ((fxCos32( t24))>>7);
  invClarke22.qs = ((fxSin32( t24))>>7);
  invclark32(&invClarke22);
  t3341 = MUL_SHIFT32(((long)(f280xReadAnalogVal(4))<<12),354334801L,30)/* 0.33@fx8.32 */;
  t3341 = MIN(16777216L,t3341);
  t3341 = MAX(t3341,0);
;
  t12 = ( t8+MUL_SHIFT32(MUL_SHIFT32( invClarke22.as, t3341,24),536870912L,30)/* 0.5@fx8.32 */);
  t12 = MIN(15938355L,t12);
  t12 = MAX(t12,838860L);
;
  t10 = (int)(( t12)>>9);
  t13 = ( t8+MUL_SHIFT32(MUL_SHIFT32( invClarke22.bs, t3341,24),536870912L,30)/* 0.5@fx8.32 */);
  t13 = MIN(15938355L,t13);
  t13 = MAX(t13,838860L);
;
  t9 = (int)(( t13)>>9);
  t14 = ( t8+MUL_SHIFT32(MUL_SHIFT32( invClarke22.cs, t3341,24),536870912L,30)/* 0.5@fx8.32 */);
  t14 = MIN(15938355L,t14);
  t14 = MAX(t14,838860L);
;
  t11 = (int)(( t14)>>9);
  { long _duty32 = (long) t10*2000;
    CMPA1 = (int)(_duty32>>15);
  }
  CMPB1 = (int)(((long) t10*2000)>>15);
  { long _duty32 = (long) t9*2000;
    CMPA2 = (int)(_duty32>>15);
  }
  CMPB2 = (int)(((long) t9*2000)>>15);
  { long _duty32 = (long) t11*2000;
    CMPA3 = (int)(_duty32>>15);
  }
  CMPB3 = (int)(((long) t11*2000)>>15);
  sim->outSigS[0].u.scaledInt.val =  t3344;
  sim->outSigS[1].u.scaledInt.val =  t3341;
  sim->outSigS[2].u.scaledInt.val =  va ;
  sim->outSigS[3].u.scaledInt.val =  vb ;
  sim->outSigS[4].u.scaledInt.val =  ia ;
  sim->outSigS[5].u.scaledInt.val =  ib ;
  sim->outSigS[6].u.scaledInt.val =  x1 ;
  sim->outSigS[7].u.scaledInt.val =  x2 ;
  sim->outSigS[8].u.scaledInt.val =  x3 ;
  sim->outSigS[9].u.scaledInt.val =  x4 ;
  sim->outSigS[10].u.scaledInt.val = MUL_SHIFT32( t59,-1073741824L,30)/* -1@fx14.32 */;
  sim->outSigS[11].u.scaledInt.val =  x6 ;

  _delayOutBuf1010= P10 ;
  _delayOutBuf486= P1 ;
  _delayOutBuf533= P10 ;
  _delayOutBuf532= P22 ;
  _delayOutBuf954= P2 ;
  _delayOutBuf953= P14 ;
  _delayOutBuf969= P3 ;
```

```
_delayOutBuf970= P15 ;
_delayOutBuf1092= P14 ;
_delayOutBuf626= P14 ;
_delayOutBuf625= P20 ;
_delayOutBuf637= P8 ;
_delayOutBuf3728= ia ;
_delayOutBuf206= x1 ;
_delayOutBuf207= x3 ;
_delayOutBuf3742= va ;
_delayOutBuf210= x1 ;
_delayOutBuf211= x3 ;
_delayOutBuf404= P13 ;
_delayOutBuf309= P19 ;
_delayOutBuf876= P16 ;
_delayOutBuf877= P22 ;
_delayOutBuf865= P10 ;
_delayOutBuf1247= P1 ;
_delayOutBuf1248= P13 ;
_delayOutBuf1232= P2 ;
_delayOutBuf1231= P14 ;
_delayOutBuf1260= P3 ;
_delayOutBuf1261= P15 ;
_delayOutBuf1219= P4 ;
_delayOutBuf1218= P16 ;
_delayOutBuf176= x1 ;
_delayOutBuf1631= P6 ;
_delayOutBuf3400= x4 ;
_delayOutBuf1699= P30 ;
_delayOutBuf1644= P12 ;
_delayOutBuf1678= P30 ;
_delayOutBuf1677= P12 ;
_delayOutBuf1669= P18 ;
_delayOutBuf1686= P24 ;
_delayOutBuf1643= P24 ;
_delayOutBuf1700= P12 ;
_delayOutBuf1708= P18 ;
_delayOutBuf1691= P24 ;
_delayOutBuf1713= P6 ;
_delayOutBuf1723= P36 ;
_delayOutBuf3474= P33 ;
_delayOutBuf714= P32 ;
_delayOutBuf715= P34 ;
_delayOutBuf1739= P32 ;
_delayOutBuf1738= P33 ;
_delayOutBuf1747= P34 ;
_delayOutBuf1905= P31 ;
_delayOutBuf1906= P32 ;
_delayOutBuf1897= P33 ;
_delayOutBuf1914= P34 ;
_delayOutBuf1892= P36 ;
_delayOutBuf1887= P35 ;
_delayOutBuf1397= P18 ;
_delayOutBuf1396= P24 ;
_delayOutBuf1408= P6 ;
_delayOutBuf3694= P30 ;
_delayOutBuf1422= P13 ;
_delayOutBuf1423= P19 ;
_delayOutBuf3525= P5 ;
_delayOutBuf1325= P10 ;
_delayOutBuf1324= P22 ;
_delayOutBuf1340= P24 ;
_delayOutBuf1341= P12 ;
```

```
_delayOutBuf1312= P7 ;
_delayOutBuf1311= P19 ;
_delayOutBuf1299= P8 ;
_delayOutBuf1300= P20 ;
_delayOutBuf1353= P9 ;
_delayOutBuf1354= P21 ;
_delayOutBuf1522= P18 ;
_delayOutBuf1523= P24 ;
_delayOutBuf1511= P12 ;
_delayOutBuf1531= P30 ;
_delayOutBuf1544= P16 ;
_delayOutBuf1543= P22 ;
_delayOutBuf1553= P10 ;
_delayOutBuf1535= P28 ;
_delayOutBuf1567= P13 ;
_delayOutBuf1568= P19 ;
_delayOutBuf1558= P7 ;
_delayOutBuf1576= P25 ;
_delayOutBuf1496= P14 ;
_delayOutBuf1495= P20 ;
_delayOutBuf1505= P8 ;
_delayOutBuf652= P2 ;
_delayOutBuf651= P14 ;
_delayOutBuf663= P20 ;
_delayOutBuf643= P32 ;
_delayOutBuf668= P8 ;
_delayOutBuf673= P26 ;
_delayOutBuf697= P4 ;
_delayOutBuf698= P16 ;
_delayOutBuf688= P22 ;
_delayOutBuf706= P34 ;
_delayOutBuf683= P10 ;
_delayOutBuf678= P28 ;
_delayOutBuf1487= P26 ;
_delayOutBuf1590= P15 ;
_delayOutBuf1589= P21 ;
_delayOutBuf1599= P9 ;
_delayOutBuf1581= P27 ;
_delayOutBuf1613= P17 ;
_delayOutBuf1614= P23 ;
_delayOutBuf1604= P11 ;
_delayOutBuf2028= P6 ;
_delayOutBuf2027= P18 ;
_delayOutBuf2039= P24 ;
_delayOutBuf2019= P36 ;
_delayOutBuf2044= P12 ;
_delayOutBuf2049= P30 ;
_delayOutBuf2073= P4 ;
_delayOutBuf2074= P16 ;
_delayOutBuf2064= P22 ;
_delayOutBuf2082= P34 ;
_delayOutBuf2059= P10 ;
_delayOutBuf2054= P28 ;
_delayOutBuf2005= P1 ;
_delayOutBuf2006= P13 ;
_delayOutBuf1996= P19 ;
_delayOutBuf2014= P31 ;
_delayOutBuf1991= P7 ;
_delayOutBuf1986= P25 ;
_delayOutBuf2096= P2 ;
_delayOutBuf2095= P14 ;
_delayOutBuf2105= P20 ;
```

```
_delayOutBuf2087= P32 ;
_delayOutBuf2110= P8 ;
_delayOutBuf2115= P26 ;
_delayOutBuf1962= P3 ;
_delayOutBuf1961= P15 ;
_delayOutBuf1971= P21 ;
_delayOutBuf1953= P33 ;
_delayOutBuf1976= P9 ;
_delayOutBuf1981= P27 ;
_delayOutBuf1939= P5 ;
_delayOutBuf1940= P17 ;
_delayOutBuf1930= P23 ;
_delayOutBuf1948= P35 ;
_delayOutBuf1925= P11 ;
_delayOutBuf1920= P29 ;
_delayOutBuf1622= P29 ;
_delayOutBuf1366= P11 ;
_delayOutBuf1365= P23 ;
_delayOutBuf3524= P17 ;
_delayOutBuf3536= P23 ;
_delayOutBuf3516= P35 ;
_delayOutBuf3541= P11 ;
_delayOutBuf3546= P29 ;
_delayOutBuf3570= P4 ;
_delayOutBuf3571= P16 ;
_delayOutBuf3561= P22 ;
_delayOutBuf3579= P34 ;
_delayOutBuf3556= P10 ;
_delayOutBuf3551= P28 ;
_delayOutBuf3502= P3 ;
_delayOutBuf3503= P15 ;
_delayOutBuf3493= P21 ;
_delayOutBuf3511= P33 ;
_delayOutBuf3488= P9 ;
_delayOutBuf3483= P27 ;
_delayOutBuf3645= P1 ;
_delayOutBuf3644= P13 ;
_delayOutBuf3654= P19 ;
_delayOutBuf3636= P31 ;
_delayOutBuf3659= P7 ;
_delayOutBuf3664= P25 ;
_delayOutBuf1413= P25 ;
_delayOutBuf3699= P1 ;
_delayOutBuf1435= P14 ;
_delayOutBuf1434= P20 ;
_delayOutBuf1444= P26 ;
_delayOutBuf3704= P2 ;
_delayOutBuf1384= P15 ;
_delayOutBuf1385= P21 ;
_delayOutBuf1375= P27 ;
_delayOutBuf3709= P3 ;
_delayOutBuf1458= P16 ;
_delayOutBuf1459= P22 ;
_delayOutBuf1449= P28 ;
_delayOutBuf3714= P4 ;
_delayOutBuf1471= P17 ;
_delayOutBuf1470= P23 ;
_delayOutBuf1480= P27 ;
_delayOutBuf3719= P5 ;
_delayOutBuf1730= P35 ;
_delayOutBuf3475= P34 ;
_delayOutBuf3466= P35 ;
```

```
_delayOutBuf3669= P31 ;
_delayOutBuf424= P31 ;
_delayOutBuf423= P33 ;
_delayOutBuf1718= P36 ;
_delayOutBuf1655= P30 ;
_delayOutBuf1656= P18 ;
_delayOutBuf1664= P24 ;
_delayOutBuf3724= P6 ;
_delayOutBuf1632= P18 ;
_delayOutBuf1273= P6 ;
_delayOutBuf1274= P18 ;
_delayOutBuf1286= P5 ;
_delayOutBuf1287= P17 ;
_delayOutBuf1839= P5 ;
_delayOutBuf1840= P17 ;
_delayOutBuf1828= P23 ;
_delayOutBuf1848= P35 ;
_delayOutBuf1823= P11 ;
_delayOutBuf1818= P29 ;
_delayOutBuf1794= P4 ;
_delayOutBuf1793= P16 ;
_delayOutBuf1803= P22 ;
_delayOutBuf1785= P34 ;
_delayOutBuf1808= P10 ;
_delayOutBuf1813= P28 ;
_delayOutBuf1862= P2 ;
_delayOutBuf1861= P14 ;
_delayOutBuf1871= P20 ;
_delayOutBuf1853= P32 ;
_delayOutBuf1876= P8 ;
_delayOutBuf1881= P26 ;
_delayOutBuf1771= P3 ;
_delayOutBuf1772= P15 ;
_delayOutBuf1762= P21 ;
_delayOutBuf1780= P33 ;
_delayOutBuf1757= P9 ;
_delayOutBuf1752= P27 ;
_delayOutBuf885= P28 ;
_delayOutBuf898= P15 ;
_delayOutBuf897= P21 ;
_delayOutBuf907= P9 ;
_delayOutBuf889= P27 ;
_delayOutBuf921= P17 ;
_delayOutBuf922= P23 ;
_delayOutBuf912= P11 ;
_delayOutBuf930= P29 ;
_delayOutBuf3621= P13 ;
_delayOutBuf3620= P19 ;
_delayOutBuf3630= P7 ;
_delayOutBuf3612= P25 ;
_delayOutBuf308= P7 ;
_delayOutBuf324= P21 ;
_delayOutBuf325= P9 ;
_delayOutBuf403= P19 ;
_delayOutBuf415= P7 ;
_delayOutBuf395= P25 ;
_delayOutBuf381= P15 ;
_delayOutBuf382= P21 ;
_delayOutBuf372= P9 ;
_delayOutBuf390= P27 ;
_delayOutBuf237= x2 ;
_delayOutBuf236= x3 ;
```

```
_delayOutBuf235= x4 ;
_delayOutBuf3734= ib ;
_delayOutBuf208= x2 ;
_delayOutBuf209= x4 ;
_delayOutBuf3737= vb ;
_delayOutBuf212= x4 ;
_delayOutBuf3395= x3 ;
_delayOutBuf617= P26 ;
_delayOutBuf604= P16 ;
_delayOutBuf605= P22 ;
_delayOutBuf595= P10 ;
_delayOutBuf613= P28 ;
_delayOutBuf1093= P20 ;
_delayOutBuf1081= P26 ;
_delayOutBuf3674= P2 ;
_delayOutBuf1067= P15 ;
_delayOutBuf1066= P21 ;
_delayOutBuf1076= P27 ;
_delayOutBuf3679= P3 ;
_delayOutBuf1054= P16 ;
_delayOutBuf1055= P22 ;
_delayOutBuf1045= P28 ;
_delayOutBuf3684= P4 ;
_delayOutBuf1105= P17 ;
_delayOutBuf1104= P23 ;
_delayOutBuf1114= P29 ;
_delayOutBuf3689= P5 ;
_delayOutBuf941= P4 ;
_delayOutBuf940= P16 ;
_delayOutBuf982= P5 ;
_delayOutBuf983= P17 ;
_delayOutBuf586= P14 ;
_delayOutBuf587= P20 ;
_delayOutBuf575= P26 ;
_delayOutBuf3420= P2 ;
_delayOutBuf561= P16 ;
_delayOutBuf560= P22 ;
_delayOutBuf570= P28 ;
_delayOutBuf3425= P4 ;
_delayOutBuf520= P2 ;
_delayOutBuf521= P14 ;
_delayOutBuf505= P4 ;
_delayOutBuf504= P16 ;
_delayOutBuf548= P8 ;
_delayOutBuf549= P20 ;
_delayOutBuf175= x2 ;
_delayOutBuf487= P13 ;
_delayOutBuf475= P19 ;
_delayOutBuf495= P31 ;
_delayOutBuf470= P7 ;
_delayOutBuf465= P25 ;
_delayOutBuf441= P3 ;
_delayOutBuf440= P15 ;
_delayOutBuf450= P21 ;
_delayOutBuf432= P33 ;
_delayOutBuf455= P9 ;
_delayOutBuf460= P27 ;
_delayOutBuf253= x3 ;
_delayOutBuf254= x4 ;
_delayOutBuf269= x6 ;
_delayOutBuf255= x6 ;
_delayOutBuf268= x5 ;
```

```
_delayOutBuf3394= x5 ;
_delayOutBuf1173= P16 ;
_delayOutBuf1172= P22 ;
_delayOutBuf1184= P10 ;
_delayOutBuf1164= P28 ;
_delayOutBuf1151= P14 ;
_delayOutBuf1152= P20 ;
_delayOutBuf1142= P8 ;
_delayOutBuf1160= P26 ;
_delayOutBuf1128= P15 ;
_delayOutBuf1127= P21 ;
_delayOutBuf1137= P9 ;
_delayOutBuf1119= P27 ;
_delayOutBuf1199= P17 ;
_delayOutBuf1200= P23 ;
_delayOutBuf1190= P11 ;
_delayOutBuf1208= P29 ;
_delayOutBuf1011= P22 ;
_delayOutBuf995= P20 ;
_delayOutBuf994= P8 ;
_delayOutBuf1023= P9 ;
_delayOutBuf1024= P21 ;
_delayOutBuf1036= P11 ;
_delayOutBuf1035= P23 ;
_delayOutBuf783= P10 ;
_delayOutBuf782= P22 ;
_delayOutBuf798= P21 ;
_delayOutBuf799= P9 ;
_delayOutBuf3447= P19 ;
_delayOutBuf3446= P7 ;
_delayOutBuf770= P11 ;
_delayOutBuf769= P23 ;
_delayOutBuf744= P3 ;
_delayOutBuf745= P15 ;
_delayOutBuf729= P4 ;
_delayOutBuf728= P16 ;
_delayOutBuf3434= P1 ;
_delayOutBuf3435= P13 ;
_delayOutBuf757= P5 ;
_delayOutBuf758= P17 ;
_delayOutBuf811= P15 ;
_delayOutBuf810= P21 ;
_delayOutBuf822= P27 ;
_delayOutBuf3456= P3 ;
_delayOutBuf836= P16 ;
_delayOutBuf837= P22 ;
_delayOutBuf827= P28 ;
_delayOutBuf3461= P4 ;
_delayOutBuf849= P17 ;
_delayOutBuf848= P23 ;
_delayOutBuf858= P27 ;
_delayOutBuf3584= P5 ;
_delayOutBuf3603= P13 ;
_delayOutBuf3604= P19 ;
_delayOutBuf3594= P25 ;
_delayOutBuf3589= P1 ;
_delayOutBuf362= P13 ;
_delayOutBuf363= P19 ;
_delayOutBuf351= P25 ;
_delayOutBuf3410= P1 ;
_delayOutBuf337= P15 ;
_delayOutBuf336= P21 ;
```

```
   _delayOutBuf346= P27 ;
   _delayOutBuf3415= P3 ;
   _delayOutBuf280= P1 ;
   _delayOutBuf279= P13 ;
   _delayOutBuf296= P3 ;
   _delayOutBuf297= P15 ;
   _sampBuf27 =  t29;
   endOfSampleCount = TIMER2TIM;

}

main()
{
   noIntegrationUsed = 1;
   EALLOW;
   PLLSTS = 0x10; // reset clk check
   WDCR=0x00ef;  // Disable Watchdog
   asm(" clrc DBGM");
   if (!(PLLSTS&8))       // Skip PLL set if OSC failure
     { PLLSTS = 0x40;     //Disable OSC check
       PLLCR = 0xa;       // set PLL to 5xOSC = 100 MHZ;
       PLLSTS = 0x0;      //Enable OSC check (&F283xx /2 mode)
     }
   PCLKCR |= 0xc;
   HISPCP = 0x0;     // HCLK  = 100 MHZ
   PCLKCR1 = 0x7;
   PCLKCR3 = 0x400;
   EDIS;
   TBPRD1 = 0x7d0;
   AQCTLA1 = 0x90;
   AQCTLB1 = 0x600;
   ETSEL1 = 0x9000;
   ETPS1 = 0x1100;
   DBCTL1 = 0xb;
   DBRED1 = 0xfa;
   DBFED1 = 0xfa;
   EALLOW;
   TZCTL1 = 0x0;
   TZSEL1 = 0x0;
   EDIS;
   TBPRD2 = 0x7d0;
   AQCTLA2 = 0x90;
   AQCTLB2 = 0x600;
   ETSEL2 = 0x9000;
   ETPS2 = 0x1100;
   DBCTL2 = 0xb;
   DBRED2 = 0xfa;
   DBFED2 = 0xfa;
   EALLOW;
   TZCTL2 = 0x0;
   TZSEL2 = 0x0;
   EDIS;
   TBPRD3 = 0x7d0;
   AQCTLA3 = 0x90;
   AQCTLB3 = 0x600;
   ETSEL3 = 0x9000;
   ETPS3 = 0x1100;
   DBCTL3 = 0xb;
   DBRED3 = 0xfa;
   DBFED3 = 0xfa;
   EALLOW;
   TZCTL3 = 0x0;
```

```
  TZSEL3 = 0x0;
  EDIS;
  ADCTRL1_VAL = 0x300;
  ADCTRL2_VAL = 0x1;
  ADCTRL3_VAL = 0x6;
  EALLOW;
  GPAMUX1 = 0x555;
  GPBMUX1 = 0x0;
  EDIS;
  ADCTRL3 = 0xE0;  // Power up ADC
  ADCCHSELSEQ1 = 0x3210;
  ADCCHSELSEQ2 = 0x7654;
  ADCCHSELSEQ3 = 0xba98;
  ADCCHSELSEQ4 = 0xfedc;
  ADCMAX_CONV = 0x17;
 simInit( &tSim );
  startSimDsp();
  installInterruptVec(-2,7,cgMain);
  TIMER2PRD = 0x4e20; // 32-bit Timer Period Low
  TIMER2PRDH = 0x0; // 32-bit Timer Period High
  TIMER2TCR |= 0x4020; //Interrupt enable, Timer Reset
  EALLOW;
  PIECTRL = 1; // Enable PIE Interrupts
  EDIS;
  IER |= 0x2000; //CPU Interrupt enable
  resetInterrupts();
  disable_interrupts();  // Disable interrupts until PC handshake complete
  TBCTL1 = 0x412; // Start timer
  TBCTL2 = 0x412; // Start timer
  TBCTL3 = 0x412; // Start timer
  dspWait();
}
```

Modelo del filtro H_∞ en Simulink y VisSim.

Filtro H_∞ con $L = I_{6\times6}$- Simulink

```
function [sys,x0,str,ts] = hinfinito4(t,x,u,flag)
%  EXKALMAN Extenden Kalman filter for the induction machine
%
%Autor: Danilo Llano
%   Copyright 2012
%
%States:  x = [Psi_qs Psi_ds Psi_qr Psi_dr wr Tl]
%         P = [p11 p22 p33 p44 p55]
%Inputs:  u = [v_qs v_ds i_qs i_ds]
%Outputs: y = [Tem i_qs i_ds Psi_qs Psi_ds Psi_qr Psi_dr wr Tl]
%
% Aproximacion lineal Filtro H infinito
%
%
%Kalman covariances: Q and R
%

%speed and rotor flux estimation of induction machines using a
%two-stage extended H infinity filter
%
%500us
%Q=diag([.06 .06 .5 .5 .01 .005])
%500us
%R=0.05*diag([1 1])
%200us
Q =diag([.02 .02 0.02 0.02 1 0.001]);
%200us
R =0.45*diag([1 1]);

%Provide Machine Parameters
%PAR = [0.95 31.93 0.95 0.531 0.408 0.1 4];

PAR1=[0.754 26.13 0.754 0.435 0.816 0.089 4];
%PAR1 = [0.302 13.08 0.302 0.087 0.228 1.662 4];
%PAR1=[1.206 54.02 1.206 0.262 0.187 11.06 4];
%PAR1=[0.226 13.04 0.226 0.029 0.022 63.87 4];



xls =PAR1(1);
xm = PAR1(2);
xlr =PAR1(3);
rs =PAR1(4);
rr =PAR1(5);
J =PAR1(6);
P =PAR1(7);
w = 0;
wb = 2*pi*60;
Ts = 200e-6;
B=0.085;
z1=exp((-B*Ts)/J);
```

```
XM = (xlr*xls*xm)/(xlr*xls + xm*xlr + xm*xls);

as1 = (wb*rs/xls) * (XM/xls - 1);
as2 = (wb*rs/(xls*xlr)) * XM;
ar2 = (wb*rr/(xls*xlr)) * XM;
ar1 = (wb*rr/xlr) * (XM/xlr - 1);

a13 = (3*P*P/(8*wb*J))*(XM/(xls*xlr));

b = wb;

k1 = (3*P/(4*wb));
cs1 = (1/xls)-XM/(xls*xls);
cs2 = -XM/(xls*xlr);

Gamma=2*k1*(1-z1)/J;


switch flag,

  %%%%%%%%%%%%%%%%%%
  % Initialization %
  %%%%%%%%%%%%%%%%%%
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes(Ts);

  %%%%%%%%%%
  % Update %
  %%%%%%%%%%
  case 2,
    sys = mdlUpdate(t,x,u,as1,as2,ar1,ar2,b,cs1,cs2,a13,k1,Ts,Q,R,J,z1,Gamma,B,P);

  %%%%%%%%%%
  % Output %
  %%%%%%%%%%
  case 3,
    sys = mdlOutputs(t,x,u,cs1,cs2,a13,k1);

  %%%%%%%%%%%%%
  % Terminate %
  %%%%%%%%%%%%%
  case 9,
    sys = []; % do nothing

  %%%%%%%%%%%%%%%%%%%%%%
  % Unexpected flags %
  %%%%%%%%%%%%%%%%%%%%%%
  otherwise
    error(['unhandled flag = ',num2str(flag)]);
end

%end dsfunc

%
%=============================================================================
```

```
%mdlInitializeSizes
%Return the sizes, initial conditions, and sample times for the S-function.
%=============================================================================
%
function [sys,x0,str,ts] = mdlInitializeSizes(Ts)

sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 42;
sizes.NumOutputs     = 9;
sizes.NumInputs      = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);

%condiciones iniciales de:
%[Psi_qs Psi_ds Psi_qr Psi_dr wr] y [p11 p22 p33 p44 p55]
x0(1:6)= [zeros(4,1);0;0];
x0(7:42)=eye(6);
str = [];
ts  = [Ts 0];

%end mdlInitializeSizes

%
%=============================================================================
%mdlUpdate
%Handle discrete state updates, sample time hits, and major time step
%requirements.
%=============================================================================
%
function sys = mdlUpdate(~,x,u,as1,as2,ar1,ar2,b,cs1,cs2,~,~,Ts,Q,R,~,z1,Gamma,B,P)

xh_minus = x(1:6);
P_minus = [x(7:12)' ; x(13:18)'; x(19:24)'; x(25:30)';x(31:36)'; x(37:42)'];
z = u(3:4);

Hrow1 = [cs1 0 cs2 0 0 0];
Hrow2 = [0 cs1 0 cs2 0 0];
H =[Hrow1;Hrow2];

Phirow1 = [(1+as1*Ts) 0 as2*Ts 0 0 0];
Phirow2 = [0 (1+as1*Ts) 0 as2*Ts 0 0];
Phirow3 = [ar2*Ts 0 (1+ar1*Ts) Ts*xh_minus(5) Ts*xh_minus(4) 0];
Phirow4 = [0 ar2*Ts -Ts*xh_minus(5) (1+ar1*Ts) -Ts*xh_minus(3) 0];
Phirow5 = [-Gamma*cs2*xh_minus(4) Gamma*cs2*xh_minus(3) Gamma*cs2*xh_minus(2)
-Gamma*cs2*xh_minus(1) z1 ((z1-1)/B)];
Phirow6=[0 0 0 0 0 1];


Phi = [Phirow1; Phirow2; Phirow3; Phirow4; Phirow5; Phirow6];

K = P_minus*H'/(H*P_minus*H'+eye(2));
xh_plus = xh_minus+K*(z-H*xh_minus);
```

```
Hr=[H; eye(6)];
error=.4;
Rw=[eye(2)+H*P_minus*H',H*P_minus; P_minus*H',-error^2*eye(6)+P_minus];

P_plus = P_minus-P_minus*Hr'*inv(Rw)*Hr*P_minus;

G=.2*ones(6);
P_minus = Phi*P_plus*Phi'+G*G;

xh_temp = zeros(6,1);
xh_temp(1) = (1+as1*Ts)*xh_minus(1)+as2*Ts*xh_minus(3)+b*Ts*u(1);
xh_temp(2) = (1+as1*Ts)*xh_minus(2)+as2*Ts*xh_minus(4)+b*Ts*u(2);
xh_temp(3) = (1+ar1*Ts)*xh_minus(3)+ar2*Ts*xh_minus(1)+Ts*xh_minus(4)*xh_minus(5);
xh_temp(4) = ar2*Ts*xh_minus(2)-Ts*xh_minus(3)*xh_minus(5)+(1+ar1*Ts)*xh_minus(4);
xh_temp(5) =Gamma*cs2*xh_minus(2)*xh_minus(3)
-Gamma*cs2*xh_minus(1)*xh_minus(4)+z1*xh_minus(5)+(P/2)*((z1-1)/B)*xh_minus(6);
xh_temp(6)=xh_plus(6);
sys(1:6)=xh_temp;
sys(7:42)=P_minus;
%sys = [xh_temp; diag(P_minus)];

%end mdlUpdate

%
%=============================================================
% mdlOutputs
% Return the output vector for the S-function
%=============================================================
%
function sys = mdlOutputs(t,x,u,cs1,cs2,a13,k1)

%x = [Psi_qs Psi_ds Psi_qr Psi_dr wr]
%Tem = k1*(x(1)*x(4)-x(2)*x(3));
iqs = cs1*x(1)+cs2*x(3);
ids = cs1*x(2)+cs2*x(4);
Tem = k1*(x(2)*iqs-x(1)*ids);
sys = [Tem; iqs; ids; x(1:4); .5*x(5); -x(6)];

%end mdlOutputs
```

Filtro H_∞ con $L = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ - Simulink

```
function [sys,x0,str,ts] = hinfinito5(t,x,u,flag)
%  EXKALMAN Extenden Kalman filter for the induction machine
%
%   Copyright 2012
% Danilo Llano
%
% States:  x = [Psi_qs Psi_ds Psi_qr Psi_dr wr Tl]
%          P = [p11 p22 p33 p44 p55]
% Inputs:  u = [v_qs v_ds i_qs i_ds]
% Outputs: y = [Tem i_qs i_ds Psi_qs Psi_ds Psi_qr Psi_dr wr Tl]
%
%  Aproximacion lineal
```

```
%
%
%Kalman covariances: Q and R
%

%speed and rotor flux estimation vof induction machines using a
%two-stage extended H infinity filter
%
%500us
%Q=diag([.06 .06 .5 .5 .01 .005])
%500us
%R=0.05*diag([1 1])
%200us
Q =diag([.02 .02 0.02 0.02 1 0.001]);
%200us
R =0.45*diag([1 1]);

%Provide Machine Parameters
%PAR = [0.95 31.93 0.95 0.531 0.408 0.1 4];

PAR1=[0.754 26.13 0.754 0.435 0.816 0.089 4];
%PAR1 = [0.302 13.08 0.302 0.087 0.228 1.662 4];
%PAR1=[1.206 54.02 1.206 0.262 0.187 11.06 4];
%PAR1=[0.226 13.04 0.226 0.029 0.022 63.87 4];



xls =PAR1(1);
xm = PAR1(2);
xlr =PAR1(3);
rs =PAR1(4);
rr =PAR1(5);
J =PAR1(6);
P =PAR1(7);
w = 0;
wb = 2*pi*60;
Ts = 200e-6;
B=0.085;
z1=exp((-B*Ts)/J);


XM = (xlr*xls*xm)/(xlr*xls + xm*xlr + xm*xls);

as1 = (wb*rs/xls) * (XM/xls - 1);
as2 = (wb*rs/(xls*xlr)) * XM;
ar2 = (wb*rr/(xls*xlr)) * XM;
ar1 = (wb*rr/xlr) * (XM/xlr - 1);

a13 = (3*P*P/(8*wb*J))*(XM/(xls*xlr));

b = wb;
```

```
k1 = (3*P/(4*wb));
cs1 = (1/xls)-XM/(xls*xls);
cs2 = -XM/(xls*xlr);

Gamma=2*k1*(1-z1)/J;


switch flag,

  %%%%%%%%%%%%%%%%%%
  % Initialization %
  %%%%%%%%%%%%%%%%%%
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes(Ts);

  %%%%%%%%%%
  % Update %
  %%%%%%%%%%
  case 2,
    sys = mdlUpdate(t,x,u,as1,as2,ar1,ar2,b,cs1,cs2,a13,k1,Ts,Q,R,J,z1,Gamma,B,P);

  %%%%%%%%%%
  % Output %
  %%%%%%%%%%
  case 3,
    sys = mdlOutputs(t,x,u,cs1,cs2,a13,k1);

  %%%%%%%%%%%%
  % Terminate %
  %%%%%%%%%%%%
  case 9,
    sys = []; % do nothing

  %%%%%%%%%%%%%%%%%%%%
  % Unexpected flags %
  %%%%%%%%%%%%%%%%%%%%
  otherwise
    error(['unhandled flag = ',num2str(flag)]);
end

%end dsfunc

%
%=============================================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function.
%=============================================================================
%
function [sys,x0,str,ts] = mdlInitializeSizes(Ts)

sizes = simsizes;
sizes.NumContStates  = 0;
```

```
sizes.NumDiscStates  = 42;
sizes.NumOutputs     = 9;
sizes.NumInputs      = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);

%condiciones iniciales de:
%[Psi_qs Psi_ds Psi_qr Psi_dr wr] y [p11 p22 p33 p44 p55]
x0(1:6)= [zeros(4,1);0;0];
x0(7:42)=eye(6);
str = [];
ts  = [Ts 0];

%end mdlInitializeSizes

%
%=============================================================================
%mdlUpdate
%Handle discrete state updates, sample time hits, and major time step
%requirements.
%=============================================================================
%
function sys = mdlUpdate(~,x,u,as1,as2,ar1,ar2,b,cs1,cs2,~,~,Ts,Q,R,~,z1,Gamma,B,P)

xh_minus = x(1:6);
P_minus = [x(7:12)' ; x(13:18)'; x(19:24)'; x(25:30)';x(31:36)'; x(37:42)'];
z = u(3:4);

Hrow1 = [cs1 0 cs2 0 0 0];
Hrow2 = [0 cs1 0 cs2 0 0];
H =[Hrow1;Hrow2];

Phirow1 = [(1+as1*Ts) 0 as2*Ts 0 0 0];
Phirow2 = [0 (1+as1*Ts) 0 as2*Ts 0 0];
Phirow3 = [ar2*Ts 0 (1+ar1*Ts) Ts*xh_minus(5) Ts*xh_minus(4) 0];
Phirow4 = [0 ar2*Ts -Ts*xh_minus(5) (1+ar1*Ts) -Ts*xh_minus(3) 0];
Phirow5 = [-Gamma*cs2*xh_minus(4) Gamma*cs2*xh_minus(3) Gamma*cs2*xh_minus(2)
-Gamma*cs2*xh_minus(1) z1 ((z1-1)/B)];
Phirow6=[0 0 0 0 0 1];
%esta ecuacion tiene una magia q no se!

Phi = [Phirow1; Phirow2; Phirow3; Phirow4; Phirow5; Phirow6];

K = P_minus*H'/(H*P_minus*H'+eye(2));
xh_plus = xh_minus+K*(z-H*xh_minus);
L=[0,0,0,0,1,0;
     0,0,0,0,0,1];

Hr=[H; L];
error=.2;
```

```
Rw=[eye(2)+H*P_minus*H',H*P_minus*L'; L*P_minus*H',-error^2*eye(2)+L*P_minus*L'];

P_plus = P_minus-P_minus*Hr'*inv(Rw)*Hr*P_minus;

G=.1*ones(6);
P_minus = Phi*P_plus*Phi'+G;

xh_temp = zeros(6,1);
xh_temp(1) = (1+as1*Ts)*xh_minus(1)+as2*Ts*xh_minus(3)+b*Ts*u(1);
xh_temp(2) = (1+as1*Ts)*xh_minus(2)+as2*Ts*xh_minus(4)+b*Ts*u(2);
xh_temp(3) = (1+ar1*Ts)*xh_minus(3)+ar2*Ts*xh_minus(1)+Ts*xh_minus(4)*xh_minus(5);
xh_temp(4) = ar2*Ts*xh_minus(2)-Ts*xh_minus(3)*xh_minus(5)+(1+ar1*Ts)*xh_minus(4);
xh_temp(5) =Gamma*cs2*xh_minus(2)*xh_minus(3)-Gamma*cs2*xh_minus(1)*xh_minus(4)
+z1*xh_minus(5)+(P/2)*((z1-1)/B)*xh_minus(6);
xh_temp(6)=xh_plus(6);
sys(1:6)=xh_temp;
sys(7:42)=P_minus;
%sys = [xh_temp; diag(P_minus)];

%end mdlUpdate

%
%=============================================================================
% mdlOutputs
% Return the output vector for the S-function
%=============================================================================
%
function sys = mdlOutputs(t,x,u,cs1,cs2,a13,k1)

%x = [Psi_qs Psi_ds Psi_qr Psi_dr wr]
%Tem = k1*(x(1)*x(4)-x(2)*x(3));
iqs = cs1*x(1)+cs2*x(3);
ids = cs1*x(2)+cs2*x(4);
Tem = k1*(x(2)*iqs-x(1)*ids);
sys = [Tem; iqs; ids; x(1:4); .5*x(5); -x(6)];

%end mdlOutputs
```

Filtro H_∞ extendido- Código VisSim

```
/*** VisSim Automatic C Code Generator Version 8.0B10 ***/
/*  Output for HinfinityExtended at Mon Jul 30 12:04:36 2012 */


#include "math.h"
#include "cgen.h"
#include "cgendll.h"
#include "c2000.h"
#include "DMC32.h"
#include "DMC32.h"
int _maxAnalogInChan=23;
CLARKE clarke3381 = CLARKE_DEFAULTS;
CLARKE clarke3380 = CLARKE_DEFAULTS;
INV_CLARKE invClarke3338 = {0};
static long n4;
static long s7;
static long s6;
static long s2;
static long s4;
static long s8;
static long s3;
static long s12;
static long s11;
static long n3;
static long s16;
static long n2;
static long n1;
static long det1;
static long n7;
static long w7;
static long w12;
static long n12;
static long w4;
static long w2;
static long n16;
static long w16;
static long w11;
static long n11;
static long n6;
static long w6;
static long w8;
static long n8;
static long w3;
static long w1;
static long s1;
static long Pka5;
static long Pka6;
static long h2;
static long Pka3;
static long Pka1;
static long h1;
static long Pka2;
static long Pka4;
static long TL1;
static long TL2;
static long TL4;
static long TL3;
static long P1;
static long Pka8;
static long Pka10;
```

```
static long Pka9;
static long Pka11;
static long Pka12;
static long P2;
static long P3;
static long Pka18;
static long Pka17;
static long Pka15;
static long Pka16;
static long Pka22;
static long Pka23;
static long Pka24;
static long P4;
static long P6;
static long Pka36;
static long Pka30;
static long Pka29;
static long P5;
static long TL5;
static long TL6;
static long TL7;
static long TL8;
static long P8;
static long P9;
static long P10;
static long P12;
static long P11;
static long TL9;
static long TL10;
static long TL11;
static long TL12;
static long P15;
static long P16;
static long P17;
static long P18;
static long TL13;
static long TL14;
static long TL15;
static long TL16;
static long P22;
static long P23;
static long P24;
static long TL17;
static long TL18;
static long TL19;
static long TL20;
static long P29;
static long P30;
static long TL21;
static long TL22;
static long TL23;
static long TL24;
static long P36;
static long ml;
static long as1;
static long b;
static long g1;
static long Gamma;
static long as2;
static long ar1;
static long ar2;
static long cs1;
```

```
static long cs2;
static long Ts;
static long z1;
static long f3;
static long f4;
static long x3a;
static long x5a;
static long x1a;
static long x2a;
static long B4;
static long B1;
static long A54;
static long A53;
static long A56;
static long A15;
static long A12;
static long A16;
static long A14;
static long A11;
static long A9;
static long A8;
static long A6;
static long A3;
static long A1;
static long x4a;
static long A52;
static long A51;
static long ia;
static long i1;
static long i2;
static long ib;
static long vb;
static long v2;
static long v1;
static long va;
static long x1;
static long x2;
static long x4;
static long x3;
static long x6a;
static long x6;
static long x5;
static long Pa1;
static long Pa2;
static long Pa4;
static long Pa3;
static long Pa5;
static long Pa6;
static long Pa12;
static long Pa10;
static long Pa11;
static long Pa9;
static long Pa8;
static long Pa15;
static long Pa16;
static long Pa18;
static long Pa17;
static long Pa24;
static long Pa23;
static long Pa22;
static long Pa29;
static long Pa30;
```

```
static long Pa36;
static long x1_k1;
static long x2_k1;
static long x3_k1;
static long x4_k1;
static long x5_k1;
static long x6_k1;
static long L1;
static long L2;
static long L3;
static long L4;
static long det;
static long alpha1;
static long alpha4;
static long alpha2;
static long alpha3;
static long K1;
static long K3;
static long K5;
static long K7;
static long K9;
static long K11;
static long K2;
static long K4;
static long K6;
static long K8;
static long K10;
static long K12;
static long beta1;
static long beta2;

extern CGDOUBLE Zed;

DLL_SIG_DECL_M(0,12)
INTERRUPT void cgMain();
static ARG_DESCR outArgInfo3410[]={
  { T_SCALED_INT,0,8,32},
  { T_SCALED_INT,0,8,32},
  { T_SCALED_INT,0,12,32},
  { T_SCALED_INT,0,12,32},
  { T_SCALED_INT,0,12,32},
  { T_SCALED_INT,0,12,32},
  { T_SCALED_INT,0,12,32},
  { T_SCALED_INT,0,12,32},
  { T_SCALED_INT,0,12,32},
  { T_SCALED_INT,0,12,32},
  { T_SCALED_INT,0,12,32},
  { T_SCALED_INT,0,12,32},
};
static ARG_DESCR inArgInfo3410[]={
0};
static SIM_STATE tSim={0, 0.0002, 10,0,0.0002,0,0,0,0,0,0,0,0
,outArgInfo3410, inArgInfo3410,0,12,0,0,0,cgMain,0,0,0,0,0,0,1};
SIM_STATE *sim=&tSim;

/* HinfinityExtended */
INTERRUPT void cgMain()
{
static long _delayOutBuf1825=0;
static long _delayOutBuf1873=0;
static long _delayOutBuf1912=0;
static long _delayOutBuf1856=0;
```

```
static long _delayOutBuf1864=0;
static long _delayOutBuf1905=0;
static long _delayOutBuf1808=0;
static long _delayOutBuf1821=0;
static long _delayOutBuf1816=0;
static long _delayOutBuf1813=0;
static long _delayOutBuf1828=0;
static long _delayOutBuf1920=1048576;
static long _delayOutBuf1869=0;
static long _delayOutBuf1925=0;
static long _delayOutBuf1929=1048576;
static long _delayOutBuf1888=1048576;
static long _delayOutBuf1836=0;
static long _delayOutBuf1908=0;
static long _delayOutBuf1880=0;
static long _delayOutBuf1897=0;
static long _delayOutBuf1917=1048576;
static long _delayOutBuf1875=0;
static long _delayOutBuf1883=0;
static long _delayOutBuf1853=0;
static long _delayOutBuf1833=0;
static long _delayOutBuf1900=0;
static long _delayOutBuf1841=0;
static long _delayOutBuf1844=0;
static long _delayOutBuf1892=1048576;
static long _delayOutBuf1861=0;
static long _delayOutBuf1848=1048576;
  long t1788;
  long t1785;
  long t1773;
  long t1776;
  long t1741;
  long t2358;
  long t3332;
  long t3346;
  long t3343;
static long _sampBuf3342=0;
  long t3370;
  long t3340;
  long t3335;
  long t3374;
  int t3372;
  long t3375;
  int t3371;
  long t3376;
  int t3373;
  ADCTRL2 |= 0x4001;  // Reset ADC Seq
  ADCTRL2 |= 0x2001;  // Trigger ADC
  GET_MAX_STACK_USED();
  as1 = -355305888 /* -338.846@fx12.32 */;
  Ts = 209 /* 0.0002@fx12.32 */;
  t1788 = (MUL_SHIFT32( as1 , Ts ,20)+1048576 /* 1@fx12.32 */);
  A1 =  t1788;
  Pa1 =  _delayOutBuf1848;
  as2 = 344698493 /* 328.73@fx12.32 */;
  t1785 = MUL_SHIFT32( as2 , Ts ,20);
  A3 =  t1785;
  Pa3 =  _delayOutBuf1861;
  g1 = 4194304 /* 4@fx12.32 */;
  Pa15 =  _delayOutBuf1892;
  Pka1 = (MUL_SHIFT32((MUL_SHIFT32( A1 , Pa1 ,20)+MUL_SHIFT32( A3 , Pa3 ,20)), A1 ,20)+ g1 +MUL_SHIFT32((MUL_SHIFT32( A1 , Pa3 ,20)+
       MUL_SHIFT32( A3 , Pa15 ,20)), A3 ,20));
```

```
cs1 = 83571 /* 0.0797@fx12.32 */;
h1 =  cs1 ;
ar1 = -478872286 /* -456.688@fx12.32 */;
t1773 = (MUL_SHIFT32( ar1 , Ts ,20)+1048576 /* 1@fx12.32 */);
A11 =  t1773;
Pa4 =  _delayOutBuf1856;
Pa16 =  _delayOutBuf1897;
x5a =  _delayOutBuf1844;
A12 = MUL_SHIFT32( Ts , x5a ,20);
ar2 = 464575895 /* 443.054@fx12.32 */;
t1776 = MUL_SHIFT32( ar2 , Ts ,20);
A9 =  t1776;
Pa5 =  _delayOutBuf1864;
Pa17 =  _delayOutBuf1905;
x4a =  _delayOutBuf1833;
f3 = MUL_SHIFT32( x4a , Ts ,20);
Pka3 = (MUL_SHIFT32((MUL_SHIFT32( A1 , Pa3 ,20)+MUL_SHIFT32( A3 , Pa15 ,20)), A11 ,20)+MUL_SHIFT32((MUL_SHIFT32( A1 , Pa4 ,20)+
       MUL_SHIFT32( A3 , Pa16 ,20)), A12 ,20)+ g1 +MUL_SHIFT32((MUL_SHIFT32( A1 , Pa1 ,20)+MUL_SHIFT32( A3 , Pa3 ,20)), A9 ,20)+MUL_SHIFT32(
       (MUL_SHIFT32( A1 , Pa5 ,20)+MUL_SHIFT32( A3 , Pa17 ,20)), f3 ,20));
cs2 = -83571 /* -0.0797@fx12.32 */;
h2 =  cs2 ;
A6 =  t1788;
Pa10 =  _delayOutBuf1875;
A8 =  t1785;
Pa22 =  _delayOutBuf1917;
Pa8 =  _delayOutBuf1888;
Pka8 = (MUL_SHIFT32((MUL_SHIFT32( A6 , Pa10 ,20)+MUL_SHIFT32( A8 , Pa22 ,20)), A8 ,20)+ g1 +MUL_SHIFT32((MUL_SHIFT32( A6 , Pa8 ,20)+
       MUL_SHIFT32( A8 , Pa10 ,20)), A6 ,20));
A16 =  t1773;
A14 =  t1776;
Pa9 =  _delayOutBuf1883;
A15 = MUL_SHIFT32( A12 ,-1073741824L,30)/* -1@fx12.32 */;
Pa11 =  _delayOutBuf1880;
Pa23 =  _delayOutBuf1912;
x3a =  _delayOutBuf1836;
f4 = MUL_SHIFT32( x3a ,MUL_SHIFT32( Ts ,-1073741824L,30)/* -1@fx12.32 */,20);
Pka10 = (MUL_SHIFT32((MUL_SHIFT32( A6 , Pa10 ,20)+MUL_SHIFT32( A8 , Pa22 ,20)), A16 ,20)+MUL_SHIFT32((MUL_SHIFT32( A8 , Pa10 ,20)+
       MUL_SHIFT32( A6 , Pa8 ,20)), A14 ,20)+ g1 +MUL_SHIFT32((MUL_SHIFT32( A6 , Pa9 ,20)+MUL_SHIFT32( A8 , Pa16 ,20)), A15 ,20)+MUL_SHIFT32(
       (MUL_SHIFT32( A6 , Pa11 ,20)+MUL_SHIFT32( A8 , Pa23 ,20)), f4 ,20));
Pa29 =  _delayOutBuf1920;
Pka22 = (MUL_SHIFT32((MUL_SHIFT32( A15 , Pa16 ,20)+MUL_SHIFT32( A16 , Pa22 ,20)+MUL_SHIFT32( A14 , Pa10 ,20)+MUL_SHIFT32( f4 ,
        Pa23 ,20)), A16 ,20)+MUL_SHIFT32((MUL_SHIFT32( A15 , Pa9 ,20)+MUL_SHIFT32( A16 , Pa10 ,20)+MUL_SHIFT32( A14 , Pa8 ,20)+MUL_SHIFT32(
        f4 , Pa11 ,20)), A14 ,20)+ g1 +MUL_SHIFT32((MUL_SHIFT32( A15 , Pa15 ,20)+MUL_SHIFT32( A16 , Pa16 ,20)+MUL_SHIFT32( A14 , Pa9 ,20)+
       MUL_SHIFT32( f4 , Pa17 ,20)), A15 ,20)+MUL_SHIFT32((MUL_SHIFT32( A15 , Pa17 ,20)+MUL_SHIFT32( A16 , Pa23 ,20)+MUL_SHIFT32( A14 ,
        Pa11 ,20)+MUL_SHIFT32( f4 , Pa29 ,20)), f4 ,20));
L4 = (MUL_SHIFT32((MUL_SHIFT32( Pka8 , h1 ,20)+MUL_SHIFT32( Pka10 , h2 ,20)), h1 ,20)+1048576 /* 1@fx12.32 */+MUL_SHIFT32((MUL_SHIFT32(
        Pka10 , h1 ,20)+MUL_SHIFT32( Pka22 , h2 ,20)), h2 ,20));
Pka15 = (MUL_SHIFT32((MUL_SHIFT32( A11 , Pa15 ,20)+MUL_SHIFT32( A12 , Pa16 ,20)+MUL_SHIFT32( f3 , Pa17 ,20)+MUL_SHIFT32( A9 , Pa3 ,20)),
        A11 ,20)+MUL_SHIFT32((MUL_SHIFT32( A11 , Pa16 ,20)+MUL_SHIFT32( A12 , Pa22 ,20)+MUL_SHIFT32( f3 , Pa23 ,20)+MUL_SHIFT32( A9 , Pa4 ,20)),
        A12 ,20)+ g1 +MUL_SHIFT32((MUL_SHIFT32( A11 , Pa17 ,20)+MUL_SHIFT32( A12 , Pa23 ,20)+MUL_SHIFT32( f3 , Pa29 ,20)+MUL_SHIFT32( A9 ,
        Pa5 ,20)), f3 ,20)+MUL_SHIFT32((MUL_SHIFT32( A11 , Pa3 ,20)+MUL_SHIFT32( A12 , Pa4 ,20)+MUL_SHIFT32( f3 , Pa5 ,20)+MUL_SHIFT32(
        A9 , Pa1 ,20)), A9 ,20));
L1 = (MUL_SHIFT32((MUL_SHIFT32( Pka1 , h1 ,20)+MUL_SHIFT32( Pka3 , h2 ,20)), h1 ,20)+1048576 /* 1@fx12.32 */+MUL_SHIFT32((MUL_SHIFT32(
        Pka3 , h1 ,20)+MUL_SHIFT32( Pka15 , h2 ,20)), h2 ,20));
Pa2 =  _delayOutBuf1853;
Pka2 = (MUL_SHIFT32((MUL_SHIFT32( A1 , Pa2 ,20)+MUL_SHIFT32( A3 , Pa9 ,20)), A6 ,20)+ g1 +MUL_SHIFT32((MUL_SHIFT32( A1 , Pa4 ,20)+
       MUL_SHIFT32( A3 , Pa16 ,20)), A8 ,20));
Pka9 = (MUL_SHIFT32((MUL_SHIFT32( A6 , Pa10 ,20)+MUL_SHIFT32( A8 , Pa22 ,20)), A12 ,20)+MUL_SHIFT32((MUL_SHIFT32( A8 , Pa16 ,20)+
       MUL_SHIFT32( A6 , Pa9 ,20)), A11 ,20)+ g1 +MUL_SHIFT32((MUL_SHIFT32( A8 , Pa4 ,20)+MUL_SHIFT32( A6 , Pa2 ,20)), A9 ,20)+MUL_SHIFT32(
       (MUL_SHIFT32( A6 , Pa11 ,20)+MUL_SHIFT32( A8 , Pa23 ,20)), f3 ,20));
Pka4 = (MUL_SHIFT32((MUL_SHIFT32( A1 , Pa2 ,20)+MUL_SHIFT32( A3 , Pa9 ,20)), A14 ,20)+MUL_SHIFT32((MUL_SHIFT32( A1 , Pa3 ,20)+
       MUL_SHIFT32( A3 , Pa15 ,20)), A15 ,20)+ g1 +MUL_SHIFT32((MUL_SHIFT32( A1 , Pa4 ,20)+MUL_SHIFT32( A3 , Pa16 ,20)), A16 ,20)+MUL_SHIFT32(
```

```
      (MUL_SHIFT32( A1 , Pa5 ,20)+MUL_SHIFT32( A3 , Pa17 ,20)), f4 ,20));
Pka16 = (MUL_SHIFT32((MUL_SHIFT32( A11 , Pa9 ,20)+MUL_SHIFT32( A12 , Pa10 ,20)+MUL_SHIFT32( f3 , Pa11 ,20)+MUL_SHIFT32( A9 , Pa2 ,20)),
       A14 ,20)+MUL_SHIFT32((MUL_SHIFT32( A11 , Pa15 ,20)+MUL_SHIFT32( A12 , Pa16 ,20)+MUL_SHIFT32( f3 , Pa17 ,20)+MUL_SHIFT32( A9 , Pa3 ,20)),
       A15 ,20)+ g1 +MUL_SHIFT32((MUL_SHIFT32( A11 , Pa16 ,20)+MUL_SHIFT32( A12 , Pa22 ,20)+MUL_SHIFT32( f3 , Pa23 ,20)+MUL_SHIFT32( A9 ,
       Pa4 ,20)), A16 ,20)+MUL_SHIFT32((MUL_SHIFT32( A11 , Pa17 ,20)+MUL_SHIFT32( A12 , Pa23 ,20)+MUL_SHIFT32( f3 , Pa29 ,20)+MUL_SHIFT32(
       A9 , Pa5 ,20)), f4 ,20));
L2 = (MUL_SHIFT32((MUL_SHIFT32( Pka2 , h1 ,20)+MUL_SHIFT32( Pka9 , h2 ,20)), h1 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka4 , h1 ,20)+
      MUL_SHIFT32( Pka16 , h2 ,20)), h2 ,20));
L3 = (MUL_SHIFT32((MUL_SHIFT32( Pka2 , h1 ,20)+MUL_SHIFT32( Pka4 , h2 ,20)), h1 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka9 , h1 ,20)+
      MUL_SHIFT32( Pka16 , h2 ,20)), h2 ,20));
det = (MUL_SHIFT32( L1 , L4 ,20)- MUL_SHIFT32( L2 , L3 ,20));
alpha1 = DIV_SHIFT32( L4 , det ,20,20);
alpha3 = DIV_SHIFT32(MUL_SHIFT32( L3 ,-1073741824L,30)/* -1@fx12.32 */, det ,20,20);
K1 = (MUL_SHIFT32((MUL_SHIFT32( Pka1 , h1 ,20)+MUL_SHIFT32( Pka3 , h2 ,20)), alpha1 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka2 , h1 ,20)+
      MUL_SHIFT32( Pka4 , h2 ,20)), alpha3 ,20));
i1 =  _delayOutBuf1808;
x1a =  _delayOutBuf1825;
b = 395303714 /* 376.991@fx12.32 */;
t1741 = MUL_SHIFT32( b , Ts ,20);
B1 =  t1741;
v1 =  _delayOutBuf1821;
x1_k1 = (MUL_SHIFT32( A1 , x1a ,20)+MUL_SHIFT32( A3 , x3a ,20)+MUL_SHIFT32( B1 , v1 ,20));
x3_k1 = (MUL_SHIFT32( A9 , x1a ,20)+MUL_SHIFT32( A11 , x3a ,20)+MUL_SHIFT32( A12 , x4a ,20));
beta1 = ( i1 - MUL_SHIFT32( x1_k1 , h1 ,20)- MUL_SHIFT32( x3_k1 , h2 ,20));
alpha2 = DIV_SHIFT32(MUL_SHIFT32( L2 ,-1073741824L,30)/* -1@fx12.32 */, det ,20,20);
alpha4 = DIV_SHIFT32( L1 , det ,20,20);
K2 = (MUL_SHIFT32((MUL_SHIFT32( Pka1 , h1 ,20)+MUL_SHIFT32( Pka3 , h2 ,20)), alpha2 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka2 , h1 ,20)+
      MUL_SHIFT32( Pka4 , h2 ,20)), alpha4 ,20));
i2 =  _delayOutBuf1813;
x2a =  _delayOutBuf1828;
B4 =  t1741;
v2 =  _delayOutBuf1816;
x2_k1 = (MUL_SHIFT32( A6 , x2a ,20)+MUL_SHIFT32( A8 , x4a ,20)+MUL_SHIFT32( B4 , v2 ,20));
x4_k1 = (MUL_SHIFT32( A14 , x2a ,20)+MUL_SHIFT32( A15 , x3a ,20)+MUL_SHIFT32( A16 , x4a ,20));
beta2 = ( i2 - MUL_SHIFT32( x2_k1 , h1 ,20)- MUL_SHIFT32( x4_k1 , h2 ,20));
x1 = (MUL_SHIFT32( K1 , beta1 ,20)+ x1_k1 +MUL_SHIFT32( K2 , beta2 ,20));
Pa12 =  _delayOutBuf1873;
Pa24 =  _delayOutBuf1908;
Pka12 = (MUL_SHIFT32( A6 , Pa12 ,20)+ g1 +MUL_SHIFT32( A8 , Pa24 ,20));
z1 = 1048366 /* 0.9998@fx12.32 */;
Pa30 =  _delayOutBuf1925;
Gamma = 92 /* 8.84e-005@fx12.32 */;
A52 = MUL_SHIFT32(MUL_SHIFT32( Gamma , cs2 ,20), x3a ,20);
A53 = MUL_SHIFT32(MUL_SHIFT32( Gamma , cs2 ,20), x2a ,20);
Pa18 =  _delayOutBuf1900;
A54 = MUL_SHIFT32(MUL_SHIFT32( Gamma , cs2 ,20),MUL_SHIFT32( x1a ,-1073741824L,30)/* -1@fx12.32 */,20);
A51 = MUL_SHIFT32(MUL_SHIFT32( Gamma , cs2 ,20),MUL_SHIFT32( x4a ,-1073741824L,30)/* -1@fx12.32 */,20);
Pa6 =  _delayOutBuf1869;
A56 = MUL_SHIFT32(( z1 +-1048576 /* -1@fx12.32 */),83886080 /* 80@fx12.32 */,20);
Pa36 =  _delayOutBuf1929;
Pka30 = (MUL_SHIFT32( z1 , Pa30 ,20)+MUL_SHIFT32( A52 , Pa12 ,20)+MUL_SHIFT32( A53 , Pa18 ,20)+ g1 +MUL_SHIFT32( A54 , Pa24 ,20)+
      MUL_SHIFT32( A51 , Pa6 ,20)+MUL_SHIFT32( A56 , Pa36 ,20));
Pka11 = (MUL_SHIFT32((MUL_SHIFT32( A6 , Pa10 ,20)+MUL_SHIFT32( A8 , Pa22 ,20)), A54 ,20)+MUL_SHIFT32((MUL_SHIFT32( A8 , Pa24 ,20)+
      MUL_SHIFT32( A6 , Pa12 ,20)), A56 ,20)+MUL_SHIFT32((MUL_SHIFT32( A6 , Pa2 ,20)+MUL_SHIFT32( A8 , Pa4 ,20)), A51 ,20)+ g1 +MUL_SHIFT32(
       (MUL_SHIFT32( A6 , Pa8 ,20)+MUL_SHIFT32( A8 , Pa10 ,20)), A52 ,20)+MUL_SHIFT32((MUL_SHIFT32( A6 , Pa9 ,20)+MUL_SHIFT32( A8 , Pa16 ,20)),
       A53 ,20)+MUL_SHIFT32((MUL_SHIFT32( A6 , Pa11 ,20)+MUL_SHIFT32( A8 , Pa23 ,20)), z1 ,20));
s1 = (MUL_SHIFT32((MUL_SHIFT32( Pka1 , h1 ,20)+MUL_SHIFT32( Pka3 , h2 ,20)), h1 ,20)+1048576 /* 1@fx12.32 */+MUL_SHIFT32((MUL_SHIFT32(
       Pka3 , h1 ,20)+MUL_SHIFT32( Pka15 , h2 ,20)), h2 ,20));
s6 = (MUL_SHIFT32((MUL_SHIFT32( Pka8 , h1 ,20)+MUL_SHIFT32( Pka10 , h2 ,20)), h1 ,20)+1048576 /* 1@fx12.32 */+MUL_SHIFT32((MUL_SHIFT32(
       Pka10 , h1 ,20)+MUL_SHIFT32( Pka22 , h2 ,20)), h2 ,20));
Pka36 = ( Pa36 + g1 );
m1 = 167772 /* 0.16@fx12.32 */;
```

```
s16 = ( Pka36 - m1 );
Pka24 = (MUL_SHIFT32( f4 , Pa30 ,20)+MUL_SHIFT32( A14 , Pa12 ,20)+ g1 +MUL_SHIFT32( A15 , Pa18 ,20)+MUL_SHIFT32( A16 , Pa24 ,20));
s8 = (MUL_SHIFT32( Pka12 , h1 ,20)+MUL_SHIFT32( Pka24 , h2 ,20));
s2 = (MUL_SHIFT32((MUL_SHIFT32( Pka2 , h1 ,20)+MUL_SHIFT32( Pka9 , h2 ,20)), h1 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka4 , h1 ,20)+
     MUL_SHIFT32( Pka16 , h2 ,20)), h2 ,20));
Pka6 = (MUL_SHIFT32( A1 , Pa6 ,20)+ g1 +MUL_SHIFT32( A3 , Pa18 ,20));
Pka18 = (MUL_SHIFT32( f3 , Pa30 ,20)+MUL_SHIFT32( A11 , Pa18 ,20)+ g1 +MUL_SHIFT32( A12 , Pa24 ,20)+MUL_SHIFT32( A9 , Pa6 ,20));
s4 = (MUL_SHIFT32( Pka6 , h1 ,20)+MUL_SHIFT32( Pka18 , h2 ,20));
n11 = (MUL_SHIFT32( s1 ,(MUL_SHIFT32( s6 , s16 ,20)- MUL_SHIFT32( s8 , s8 ,20)),20)- MUL_SHIFT32( s2 ,(MUL_SHIFT32( s2 , s16 ,20)-
     MUL_SHIFT32( s8 , s4 ,20)),20)+MUL_SHIFT32( s4 ,(MUL_SHIFT32( s2 , s8 ,20)- MUL_SHIFT32( s6 , s4 ,20)),20));
Pka29 = (MUL_SHIFT32((MUL_SHIFT32( A51 , Pa6 ,20)+MUL_SHIFT32( A53 , Pa18 ,20)+MUL_SHIFT32( A54 , Pa24 ,20)+MUL_SHIFT32( A56 ,
      Pa36 ,20)+MUL_SHIFT32( A52 , Pa12 ,20)+MUL_SHIFT32( z1 , Pa30 ,20)), A56 ,20)+MUL_SHIFT32((MUL_SHIFT32( A51 , Pa4 ,20)+MUL_SHIFT32(
      A53 , Pa16 ,20)+MUL_SHIFT32( A54 , Pa22 ,20)+MUL_SHIFT32( A56 , Pa24 ,20)+MUL_SHIFT32( A52 , Pa10 ,20)+MUL_SHIFT32( z1 , Pa23 ,20)),
      A54 ,20)+MUL_SHIFT32((MUL_SHIFT32( A51 , Pa1 ,20)+MUL_SHIFT32( A53 , Pa3 ,20)+MUL_SHIFT32( A54 , Pa4 ,20)+MUL_SHIFT32( A56 , Pa6 ,20)+
     MUL_SHIFT32( A52 , Pa2 ,20)+MUL_SHIFT32( z1 , Pa5 ,20)), A51 ,20)+ g1 +MUL_SHIFT32((MUL_SHIFT32( A51 , Pa2 ,20)+MUL_SHIFT32( A53 ,
      Pa9 ,20)+MUL_SHIFT32( A54 , Pa10 ,20)+MUL_SHIFT32( A56 , Pa12 ,20)+MUL_SHIFT32( A52 , Pa8 ,20)+MUL_SHIFT32( z1 , Pa11 ,20)), A52 ,20)+
     MUL_SHIFT32((MUL_SHIFT32( A51 , Pa3 ,20)+MUL_SHIFT32( A53 , Pa15 ,20)+MUL_SHIFT32( A54 , Pa16 ,20)+MUL_SHIFT32( A56 , Pa18 ,20)+
     MUL_SHIFT32( A52 , Pa9 ,20)+MUL_SHIFT32( z1 , Pa17 ,20)), A53 ,20)+MUL_SHIFT32((MUL_SHIFT32( A51 , Pa5 ,20)+MUL_SHIFT32( A53 , Pa17 ,20)+
     MUL_SHIFT32( A54 , Pa23 ,20)+MUL_SHIFT32( A56 , Pa30 ,20)+MUL_SHIFT32( A52 , Pa11 ,20)+MUL_SHIFT32( z1 , Pa29 ,20)), z1 ,20));
s11 = ( Pka29 - m1 );
s12 =  Pka30 ;
Pka23 = (MUL_SHIFT32((MUL_SHIFT32( A15 , Pa18 ,20)+MUL_SHIFT32( A16 , Pa24 ,20)+MUL_SHIFT32( A14 , Pa12 ,20)+MUL_SHIFT32( f4 ,
      Pa30 ,20)), A56 ,20)+MUL_SHIFT32((MUL_SHIFT32( A15 , Pa16 ,20)+MUL_SHIFT32( A16 , Pa22 ,20)+MUL_SHIFT32( A14 , Pa10 ,20)+MUL_SHIFT32(
      f4 , Pa23 ,20)), A54 ,20)+MUL_SHIFT32((MUL_SHIFT32( A15 , Pa3 ,20)+MUL_SHIFT32( A16 , Pa4 ,20)+MUL_SHIFT32( A14 , Pa2 ,20)+MUL_SHIFT32(
      f4 , Pa5 ,20)), A51 ,20)+ g1 +MUL_SHIFT32((MUL_SHIFT32( A15 , Pa9 ,20)+MUL_SHIFT32( A16 , Pa10 ,20)+MUL_SHIFT32( A14 , Pa8 ,20)+
     MUL_SHIFT32( f4 , Pa11 ,20)), A52 ,20)+MUL_SHIFT32((MUL_SHIFT32( A15 , Pa15 ,20)+MUL_SHIFT32( A16 , Pa16 ,20)+MUL_SHIFT32( A14 ,
      Pa9 ,20)+MUL_SHIFT32( f4 , Pa17 ,20)), A53 ,20)+MUL_SHIFT32((MUL_SHIFT32( A15 , Pa17 ,20)+MUL_SHIFT32( A16 , Pa23 ,20)+MUL_SHIFT32(
      A14 , Pa11 ,20)+MUL_SHIFT32( f4 , Pa29 ,20)), z1 ,20));
s7 = (MUL_SHIFT32( Pka11 , h1 ,20)+MUL_SHIFT32( Pka23 , h2 ,20));
n1 = (MUL_SHIFT32( s6 ,(MUL_SHIFT32( s11 , s16 ,20)- MUL_SHIFT32( s12 , s12 ,20)),20)- MUL_SHIFT32( s7 ,(MUL_SHIFT32( s7 , s16 ,20)-
     MUL_SHIFT32( s12 , s8 ,20)),20)+MUL_SHIFT32( s8 ,(MUL_SHIFT32( s7 , s12 ,20)- MUL_SHIFT32( s11 , s8 ,20)),20));
Pka5 = (MUL_SHIFT32((MUL_SHIFT32( A1 , Pa1 ,20)+MUL_SHIFT32( A3 , Pa3 ,20)), A51 ,20)+MUL_SHIFT32((MUL_SHIFT32( A1 , Pa2 ,20)+
     MUL_SHIFT32( A3 , Pa9 ,20)), A52 ,20)+MUL_SHIFT32((MUL_SHIFT32( A1 , Pa3 ,20)+MUL_SHIFT32( A3 , Pa15 ,20)), A53 ,20)+ g1 +MUL_SHIFT32(
      (MUL_SHIFT32( A1 , Pa4 ,20)+MUL_SHIFT32( A3 , Pa16 ,20)), A54 ,20)+MUL_SHIFT32((MUL_SHIFT32( A1 , Pa6 ,20)+MUL_SHIFT32( A3 , Pa18 ,20)),
      A56 ,20)+MUL_SHIFT32((MUL_SHIFT32( A1 , Pa5 ,20)+MUL_SHIFT32( A3 , Pa17 ,20)), z1 ,20));
Pka17 = (MUL_SHIFT32((MUL_SHIFT32( A11 , Pa18 ,20)+MUL_SHIFT32( A12 , Pa24 ,20)+MUL_SHIFT32( A9 , Pa6 ,20)+MUL_SHIFT32( f3 , Pa30 ,20)),
      A56 ,20)+MUL_SHIFT32((MUL_SHIFT32( A11 , Pa3 ,20)+MUL_SHIFT32( A12 , Pa4 ,20)+MUL_SHIFT32( f3 , Pa5 ,20)+MUL_SHIFT32( A9 , Pa1 ,20)),
      A51 ,20)+MUL_SHIFT32((MUL_SHIFT32( A11 , Pa9 ,20)+MUL_SHIFT32( A12 , Pa10 ,20)+MUL_SHIFT32( f3 , Pa11 ,20)+MUL_SHIFT32( A9 , Pa2 ,20)),
      A52 ,20)+ g1 +MUL_SHIFT32((MUL_SHIFT32( A11 , Pa15 ,20)+MUL_SHIFT32( A12 , Pa16 ,20)+MUL_SHIFT32( f3 , Pa17 ,20)+MUL_SHIFT32( A9 ,
      Pa3 ,20)), A53 ,20)+MUL_SHIFT32((MUL_SHIFT32( A11 , Pa16 ,20)+MUL_SHIFT32( A12 , Pa22 ,20)+MUL_SHIFT32( f3 , Pa23 ,20)+MUL_SHIFT32(
      A9 , Pa4 ,20)), A54 ,20)+MUL_SHIFT32((MUL_SHIFT32( A11 , Pa17 ,20)+MUL_SHIFT32( A12 , Pa23 ,20)+MUL_SHIFT32( f3 , Pa29 ,20)+MUL_SHIFT32(
      A9 , Pa5 ,20)), z1 ,20));
s3 = (MUL_SHIFT32( Pka5 , h1 ,20)+MUL_SHIFT32( Pka17 , h2 ,20));
n2 = (MUL_SHIFT32( s2 ,(MUL_SHIFT32( s11 , s16 ,20)- MUL_SHIFT32( s12 , s12 ,20)),20)- MUL_SHIFT32( s7 ,(MUL_SHIFT32( s3 , s16 ,20)-
     MUL_SHIFT32( s12 , s4 ,20)),20)+MUL_SHIFT32( s8 ,(MUL_SHIFT32( s3 , s12 ,20)- MUL_SHIFT32( s11 , s4 ,20)),20));
n3 = (MUL_SHIFT32( s2 ,(MUL_SHIFT32( s7 , s16 ,20)- MUL_SHIFT32( s12 , s8 ,20)),20)- MUL_SHIFT32( s6 ,(MUL_SHIFT32( s3 , s16 ,20)-
     MUL_SHIFT32( s12 , s4 ,20)),20)+MUL_SHIFT32( s8 ,(MUL_SHIFT32( s3 , s8 ,20)- MUL_SHIFT32( s7 , s4 ,20)),20));
n4 = (MUL_SHIFT32( s2 ,(MUL_SHIFT32( s7 , s12 ,20)- MUL_SHIFT32( s11 , s8 ,20)),20)- MUL_SHIFT32( s6 ,(MUL_SHIFT32( s3 , s12 ,20)-
     MUL_SHIFT32( s11 , s4 ,20)),20)+MUL_SHIFT32( s7 ,(MUL_SHIFT32( s3 , s8 ,20)- MUL_SHIFT32( s7 , s4 ,20)),20));
det1 = (MUL_SHIFT32( s1 , n1 ,20)- MUL_SHIFT32( s2 , n2 ,20)+MUL_SHIFT32( s3 , n3 ,20)- MUL_SHIFT32( s4 , n4 ,20));
w11 = DIV_SHIFT32( n11 , det1 ,20,20);
n7 = (MUL_SHIFT32( s1 ,(MUL_SHIFT32( s7 , s16 ,20)- MUL_SHIFT32( s12 , s8 ,20)),20)- MUL_SHIFT32( s2 ,(MUL_SHIFT32( s3 , s16 ,20)-
     MUL_SHIFT32( s12 , s4 ,20)),20)+MUL_SHIFT32( s4 ,(MUL_SHIFT32( s3 , s8 ,20)- MUL_SHIFT32( s7 , s4 ,20)),20));
n12 = (MUL_SHIFT32( s1 ,(MUL_SHIFT32( s6 , s12 ,20)- MUL_SHIFT32( s7 , s8 ,20)),20)- MUL_SHIFT32( s2 ,(MUL_SHIFT32( s2 , s12 ,20)-
     MUL_SHIFT32( s7 , n7 ,20)),20)+MUL_SHIFT32( s3 ,(MUL_SHIFT32( s2 , s8 ,20)- MUL_SHIFT32( s6 , n7 ,20)),20));
w12 = DIV_SHIFT32(MUL_SHIFT32( n12 ,-1073741824L,30)/* -1@fx12.32 */, det1 ,20,20);
w3 = DIV_SHIFT32( n3 , det1 ,20,20);
w7 = DIV_SHIFT32(MUL_SHIFT32( n7 ,-1073741824L,30)/* -1@fx12.32 */, det1 ,20,20);
TL5 = (MUL_SHIFT32( Pka11 , w11 ,20)+MUL_SHIFT32( Pka12 , w12 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka2 , h1 ,20)+MUL_SHIFT32( Pka9 ,
      h2 ,20)), w3 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka8 , h1 ,20)+MUL_SHIFT32( Pka10 , h2 ,20)), w7 ,20));
n16 = (MUL_SHIFT32( s1 ,(MUL_SHIFT32( s6 , s11 ,20)- MUL_SHIFT32( s7 , s7 ,20)),20)- MUL_SHIFT32( s2 ,(MUL_SHIFT32( s2 , s11 ,20)-
     MUL_SHIFT32( s7 , s3 ,20)),20)+MUL_SHIFT32( s3 ,(MUL_SHIFT32( s2 , s7 ,20)- MUL_SHIFT32( s6 , s3 ,20)),20));
```

```
w16 = DIV_SHIFT32( n16 , det1 ,20,20);
w4 = DIV_SHIFT32(MUL_SHIFT32( n4 ,-1073741824L,30)/* -1@fx12.32 */, det1 ,20,20);
n8 = (MUL_SHIFT32( s1 ,(MUL_SHIFT32( s7 , s12 ,20)- MUL_SHIFT32( s11 , s8 ,20)),20)- MUL_SHIFT32( s2 ,(MUL_SHIFT32( s3 , s12 ,20)-
      MUL_SHIFT32( s11 , s4 ,20)),20)+MUL_SHIFT32( s3 ,(MUL_SHIFT32( s3 , s8 ,20)- MUL_SHIFT32( s7 , s4 ,20)),20));
w8 = DIV_SHIFT32( n8 , det1 ,20,20);
TL6 = (MUL_SHIFT32( Pka11 , w12 ,20)+MUL_SHIFT32( Pka12 , w16 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka2 , h1 ,20)+MUL_SHIFT32( Pka9 ,
      h2 ,20)), w4 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka8 , h1 ,20)+MUL_SHIFT32( Pka10 , h2 ,20)), w8 ,20));
w1 = DIV_SHIFT32( n1 , det1 ,20,20);
w2 = DIV_SHIFT32(MUL_SHIFT32( n2 ,-1073741824L,30)/* -1@fx12.32 */, det1 ,20,20);
TL7 = (MUL_SHIFT32( Pka11 , w3 ,20)+MUL_SHIFT32( Pka12 , w4 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka2 , h1 ,20)+MUL_SHIFT32( Pka9 , h2 ,20)),
      w1 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka8 , h1 ,20)+MUL_SHIFT32( Pka10 , h2 ,20)), w2 ,20));
n6 = (MUL_SHIFT32( s1 ,(MUL_SHIFT32( s11 , s16 ,20)- MUL_SHIFT32( s12 , s12 ,20)),20)- MUL_SHIFT32( s3 ,(MUL_SHIFT32( s3 , s16 ,20)-
      MUL_SHIFT32( s12 , s4 ,20)),20)+MUL_SHIFT32( s4 ,(MUL_SHIFT32( s3 , s12 ,20)- MUL_SHIFT32( s11 , s4 ,20)),20));
w6 = DIV_SHIFT32( n6 , det1 ,20,20);
TL8 = (MUL_SHIFT32( Pka11 , w7 ,20)+MUL_SHIFT32( Pka12 , w8 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka2 , h1 ,20)+MUL_SHIFT32( Pka9 , h2 ,20)),
      w2 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka8 , h1 ,20)+MUL_SHIFT32( Pka10 , h2 ,20)), w6 ,20));
P12 = ( Pka12 - MUL_SHIFT32( Pka30 , TL5 ,20)- MUL_SHIFT32( Pka36 , TL6 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka6 , h1 ,20), TL7 ,20)-
      MUL_SHIFT32(MUL_SHIFT32( Pka18 , h2 ,20), TL7 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka12 , h1 ,20), TL8 ,20)- MUL_SHIFT32(MUL_SHIFT32(
      Pka24 , h2 ,20), TL8 ,20));
TL13 = (MUL_SHIFT32( Pka23 , w11 ,20)+MUL_SHIFT32( Pka24 , w12 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka4 , h1 ,20)+MUL_SHIFT32( Pka16 ,
       h2 ,20)), w3 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka10 , h1 ,20)+MUL_SHIFT32( Pka22 , h2 ,20)), w7 ,20));
TL14 = (MUL_SHIFT32( Pka23 , w12 ,20)+MUL_SHIFT32( Pka24 , w16 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka4 , h1 ,20)+MUL_SHIFT32( Pka16 ,
       h2 ,20)), w4 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka10 , h1 ,20)+MUL_SHIFT32( Pka22 , h2 ,20)), w8 ,20));
TL15 = (MUL_SHIFT32( Pka23 , w3 ,20)+MUL_SHIFT32( Pka24 , w4 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka4 , h1 ,20)+MUL_SHIFT32( Pka16 ,
       h2 ,20)), w1 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka10 , h1 ,20)+MUL_SHIFT32( Pka22 , h2 ,20)), w2 ,20));
TL16 = (MUL_SHIFT32( Pka23 , w7 ,20)+MUL_SHIFT32( Pka24 , w8 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka4 , h1 ,20)+MUL_SHIFT32( Pka16 ,
       h2 ,20)), w2 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka10 , h1 ,20)+MUL_SHIFT32( Pka22 , h2 ,20)), w6 ,20));
P23 = ( Pka23 - MUL_SHIFT32( Pka29 , TL13 ,20)- MUL_SHIFT32( Pka30 , TL14 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka5 , h1 ,20), TL15 ,20)-
      MUL_SHIFT32(MUL_SHIFT32( Pka17 , h2 ,20), TL15 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka11 , h1 ,20), TL16 ,20)- MUL_SHIFT32(MUL_SHIFT32(
      Pka23 , h2 ,20), TL16 ,20));
TL1 = (MUL_SHIFT32( Pka5 , w11 ,20)+MUL_SHIFT32( Pka6 , w12 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka1 , h1 ,20)+MUL_SHIFT32( Pka3 , h2 ,20)),
      w3 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka2 , h1 ,20)+MUL_SHIFT32( Pka4 , h2 ,20)), w7 ,20));
TL2 = (MUL_SHIFT32( Pka5 , w12 ,20)+MUL_SHIFT32( Pka6 , w16 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka1 , h1 ,20)+MUL_SHIFT32( Pka3 , h2 ,20)),
      w4 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka2 , h1 ,20)+MUL_SHIFT32( Pka4 , h2 ,20)), w8 ,20));
TL3 = (MUL_SHIFT32( Pka5 , w3 ,20)+MUL_SHIFT32( Pka6 , w4 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka1 , h1 ,20)+MUL_SHIFT32( Pka3 , h2 ,20)),
      w1 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka2 , h1 ,20)+MUL_SHIFT32( Pka4 , h2 ,20)), w2 ,20));
TL4 = (MUL_SHIFT32( Pka5 , w7 ,20)+MUL_SHIFT32( Pka6 , w8 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka1 , h1 ,20)+MUL_SHIFT32( Pka3 , h2 ,20)),
      w2 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka2 , h1 ,20)+MUL_SHIFT32( Pka4 , h2 ,20)), w6 ,20));
P4 = ( Pka4 - MUL_SHIFT32( Pka23 , TL1 ,20)- MUL_SHIFT32( Pka24 , TL2 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka4 , h1 ,20), TL3 ,20)-
     MUL_SHIFT32(MUL_SHIFT32( Pka16 , h2 ,20), TL3 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka10 , h1 ,20), TL4 ,20)- MUL_SHIFT32(MUL_SHIFT32(
      Pka22 , h2 ,20), TL4 ,20));
P5 = ( Pka5 - MUL_SHIFT32( Pka29 , TL1 ,20)- MUL_SHIFT32( Pka30 , TL2 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka5 , h1 ,20), TL3 ,20)-
     MUL_SHIFT32(MUL_SHIFT32( Pka17 , h2 ,20), TL3 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka11 , h1 ,20), TL4 ,20)- MUL_SHIFT32(MUL_SHIFT32(
      Pka23 , h2 ,20), TL4 ,20));
TL9 = (MUL_SHIFT32( Pka17 , w11 ,20)+MUL_SHIFT32( Pka18 , w12 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka3 , h1 ,20)+MUL_SHIFT32( Pka15 ,
      h2 ,20)), w3 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka9 , h1 ,20)+MUL_SHIFT32( Pka16 , h2 ,20)), w7 ,20));
TL10 = (MUL_SHIFT32( Pka17 , w12 ,20)+MUL_SHIFT32( Pka18 , w16 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka3 , h1 ,20)+MUL_SHIFT32( Pka15 ,
       h2 ,20)), w4 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka9 , h1 ,20)+MUL_SHIFT32( Pka16 , h2 ,20)), w8 ,20));
TL11 = (MUL_SHIFT32( Pka17 , w3 ,20)+MUL_SHIFT32( Pka18 , w4 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka3 , h1 ,20)+MUL_SHIFT32( Pka15 ,
       h2 ,20)), w1 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka9 , h1 ,20)+MUL_SHIFT32( Pka16 , h2 ,20)), w2 ,20));
TL12 = (MUL_SHIFT32( Pka17 , w7 ,20)+MUL_SHIFT32( Pka18 , w8 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka3 , h1 ,20)+MUL_SHIFT32( Pka15 ,
       h2 ,20)), w2 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka9 , h1 ,20)+MUL_SHIFT32( Pka16 , h2 ,20)), w6 ,20));
P17 = ( Pka17 - MUL_SHIFT32( Pka29 , TL9 ,20)- MUL_SHIFT32( Pka30 , TL10 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka5 , h1 ,20), TL11 ,20)-
      MUL_SHIFT32(MUL_SHIFT32( Pka17 , h2 ,20), TL11 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka11 , h1 ,20), TL12 ,20)- MUL_SHIFT32(MUL_SHIFT32(
      Pka23 , h2 ,20), TL12 ,20));
clarke3381.as = MUL_SHIFT32((((long)(f280xReadAnalogVal(8))<<12)+-28487712 /* -1.698@fx8.32 */),2030043136L,26)/* 30.25@fx8.32 */;
clarke3381.bs = MUL_SHIFT32((((long)(f280xReadAnalogVal(9))<<12)+-28123647 /* -1.6763@fx8.32 */),2049504706L,26)/* 30.54@fx8.32 */;
clarke_calc(&clarke3381);
ia = (( clarke3381.ds)>>4);
clarke3380.as = (((long)(f280xReadAnalogVal(0))<<12)+-19250177 /* -1.1474@fx8.32 */);
clarke3380.bs = (((long)(f280xReadAnalogVal(1))<<12)+-20181313 /* -1.2029@fx8.32 */);
clarke_calc(&clarke3380);
```

```
va = MUL_SHIFT32(((  clarke3380.ds)>>4),1994014064L,23)/* 237.705@fx12.32 */;
vb = MUL_SHIFT32(((  clarke3380.qs)>>4),1088421888L,22)/* 259.5@fx12.32 */;
ib = ((  clarke3381.qs)>>4);
K3 = (MUL_SHIFT32((MUL_SHIFT32( Pka8 , h1 ,20)+MUL_SHIFT32( Pka10 , h2 ,20)), alpha3 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka2 , h1 ,20)+
     MUL_SHIFT32( Pka9 , h2 ,20)), alpha1 ,20));
K4 = (MUL_SHIFT32((MUL_SHIFT32( Pka8 , h1 ,20)+MUL_SHIFT32( Pka10 , h2 ,20)), alpha4 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka2 , h1 ,20)+
     MUL_SHIFT32( Pka9 , h2 ,20)), alpha2 ,20));
x2 = (MUL_SHIFT32( K3 , beta1 ,20)+ x2_k1 +MUL_SHIFT32( K4 , beta2 ,20));
TL17 = (MUL_SHIFT32( Pka29 , w11 ,20)+MUL_SHIFT32( Pka30 , w12 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka5 , h1 ,20)+MUL_SHIFT32( Pka17 ,
     h2 ,20)), w3 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka11 , h1 ,20)+MUL_SHIFT32( Pka23 , h2 ,20)), w7 ,20));
TL18 = (MUL_SHIFT32( Pka29 , w12 ,20)+MUL_SHIFT32( Pka30 , w16 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka5 , h1 ,20)+MUL_SHIFT32( Pka17 ,
     h2 ,20)), w4 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka11 , h1 ,20)+MUL_SHIFT32( Pka23 , h2 ,20)), w8 ,20));
TL19 = (MUL_SHIFT32( Pka29 , w3 ,20)+MUL_SHIFT32( Pka30 , w4 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka5 , h1 ,20)+MUL_SHIFT32( Pka17 ,
     h2 ,20)), w1 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka11 , h1 ,20)+MUL_SHIFT32( Pka23 , h2 ,20)), w2 ,20));
TL20 = (MUL_SHIFT32( Pka29 , w7 ,20)+MUL_SHIFT32( Pka30 , w8 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka5 , h1 ,20)+MUL_SHIFT32( Pka17 ,
     h2 ,20)), w2 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka11 , h1 ,20)+MUL_SHIFT32( Pka23 , h2 ,20)), w6 ,20));
P29 = ( Pka29 - MUL_SHIFT32( Pka29 , TL17 ,20)- MUL_SHIFT32( Pka30 , TL18 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka5 , h1 ,20), TL19 ,20)-
     MUL_SHIFT32(MUL_SHIFT32( Pka17 , h2 ,20), TL19 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka11 , h1 ,20), TL20 ,20)- MUL_SHIFT32(MUL_SHIFT32(
      Pka23 , h2 ,20), TL20 ,20));
P6 = ( Pka6 - MUL_SHIFT32( Pka30 , TL1 ,20)- MUL_SHIFT32( Pka36 , TL2 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka6 , h1 ,20), TL3 ,20)-
     MUL_SHIFT32(MUL_SHIFT32( Pka18 , h2 ,20), TL3 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka12 , h1 ,20), TL4 ,20)- MUL_SHIFT32(MUL_SHIFT32(
      Pka24 , h2 ,20), TL4 ,20));
P30 = ( Pka30 - MUL_SHIFT32( Pka30 , TL17 ,20)- MUL_SHIFT32( Pka36 , TL18 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka6 , h1 ,20), TL19 ,20)-
     MUL_SHIFT32(MUL_SHIFT32( Pka18 , h2 ,20), TL19 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka12 , h1 ,20), TL20 ,20)- MUL_SHIFT32(MUL_SHIFT32(
      Pka24 , h2 ,20), TL20 ,20));
TL21 = (MUL_SHIFT32( Pka30 , w11 ,20)+MUL_SHIFT32( Pka36 , w12 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka6 , h1 ,20)+MUL_SHIFT32( Pka18 ,
     h2 ,20)), w3 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka12 , h1 ,20)+MUL_SHIFT32( Pka24 , h2 ,20)), w7 ,20));
TL22 = (MUL_SHIFT32( Pka30 , w12 ,20)+MUL_SHIFT32( Pka36 , w16 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka6 , h1 ,20)+MUL_SHIFT32( Pka18 ,
     h2 ,20)), w4 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka12 , h1 ,20)+MUL_SHIFT32( Pka24 , h2 ,20)), w8 ,20));
TL23 = (MUL_SHIFT32( Pka30 , w3 ,20)+MUL_SHIFT32( Pka36 , w4 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka6 , h1 ,20)+MUL_SHIFT32( Pka18 ,
     h2 ,20)), w1 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka12 , h1 ,20)+MUL_SHIFT32( Pka24 , h2 ,20)), w2 ,20));
TL24 = (MUL_SHIFT32( Pka30 , w7 ,20)+MUL_SHIFT32( Pka36 , w8 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka6 , h1 ,20)+MUL_SHIFT32( Pka18 ,
     h2 ,20)), w2 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka12 , h1 ,20)+MUL_SHIFT32( Pka24 , h2 ,20)), w6 ,20));
P36 = ( Pka36 - MUL_SHIFT32( Pka30 , TL21 ,20)- MUL_SHIFT32( Pka36 , TL22 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka6 , h1 ,20), TL23 ,20)-
     MUL_SHIFT32(MUL_SHIFT32( Pka18 , h2 ,20), TL23 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka12 , h1 ,20), TL24 ,20)- MUL_SHIFT32(MUL_SHIFT32(
      Pka24 , h2 ,20), TL24 ,20));
P8 = ( Pka8 - MUL_SHIFT32( Pka11 , TL5 ,20)- MUL_SHIFT32( Pka12 , TL6 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka2 , h1 ,20), TL7 ,20)-
     MUL_SHIFT32(MUL_SHIFT32( Pka9 , h2 ,20), TL7 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka8 , h1 ,20), TL8 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka10 ,
     h2 ,20), TL8 ,20));
K5 = (MUL_SHIFT32((MUL_SHIFT32( Pka3 , h1 ,20)+MUL_SHIFT32( Pka15 , h2 ,20)), alpha1 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka9 , h1 ,20)+
     MUL_SHIFT32( Pka16 , h2 ,20)), alpha3 ,20));
K6 = (MUL_SHIFT32((MUL_SHIFT32( Pka3 , h1 ,20)+MUL_SHIFT32( Pka15 , h2 ,20)), alpha2 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka9 , h1 ,20)+
     MUL_SHIFT32( Pka16 , h2 ,20)), alpha4 ,20));
x3 = (MUL_SHIFT32( K5 , beta1 ,20)+ x3_k1 +MUL_SHIFT32( K6 , beta2 ,20));
P24 = ( Pka24 - MUL_SHIFT32( Pka30 , TL13 ,20)- MUL_SHIFT32( Pka36 , TL14 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka6 , h1 ,20), TL15 ,20)-
     MUL_SHIFT32(MUL_SHIFT32( Pka18 , h2 ,20), TL15 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka12 , h1 ,20), TL16 ,20)- MUL_SHIFT32(MUL_SHIFT32(
      Pka24 , h2 ,20), TL16 ,20));
P11 = ( Pka11 - MUL_SHIFT32( Pka29 , TL5 ,20)- MUL_SHIFT32( Pka30 , TL6 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka5 , h1 ,20), TL7 ,20)-
     MUL_SHIFT32(MUL_SHIFT32( Pka17 , h2 ,20), TL7 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka11 , h1 ,20), TL8 ,20)- MUL_SHIFT32(MUL_SHIFT32(
      Pka23 , h2 ,20), TL8 ,20));
P16 = ( Pka16 - MUL_SHIFT32( Pka23 , TL9 ,20)- MUL_SHIFT32( Pka24 , TL10 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka4 , h1 ,20), TL11 ,20)-
     MUL_SHIFT32(MUL_SHIFT32( Pka16 , h2 ,20), TL11 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka10 , h1 ,20), TL12 ,20)- MUL_SHIFT32(MUL_SHIFT32(
      Pka22 , h2 ,20), TL12 ,20));
P22 = ( Pka22 - MUL_SHIFT32( Pka23 , TL13 ,20)- MUL_SHIFT32( Pka24 , TL14 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka4 , h1 ,20), TL15 ,20)-
     MUL_SHIFT32(MUL_SHIFT32( Pka16 , h2 ,20), TL15 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka10 , h1 ,20), TL16 ,20)- MUL_SHIFT32(MUL_SHIFT32(
      Pka22 , h2 ,20), TL16 ,20));
P10 = ( Pka10 - MUL_SHIFT32( Pka23 , TL5 ,20)- MUL_SHIFT32( Pka24 , TL6 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka4 , h1 ,20), TL7 ,20)-
     MUL_SHIFT32(MUL_SHIFT32( Pka16 , h2 ,20), TL7 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka10 , h1 ,20), TL8 ,20)- MUL_SHIFT32(MUL_SHIFT32(
      Pka22 , h2 ,20), TL8 ,20));
P9 = ( Pka9 - MUL_SHIFT32( Pka17 , TL5 ,20)- MUL_SHIFT32( Pka18 , TL6 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka3 , h1 ,20), TL7 ,20)-
     MUL_SHIFT32(MUL_SHIFT32( Pka15 , h2 ,20), TL7 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka9 , h1 ,20), TL8 ,20)- MUL_SHIFT32(MUL_SHIFT32(
      Pka16 , h2 ,20), TL8 ,20));
```

```
  P2 = ( Pka2 - MUL_SHIFT32( Pka11 , TL1 ,20)- MUL_SHIFT32( Pka12 , TL2 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka2 , h1 ,20), TL3 ,20)-
       MUL_SHIFT32(MUL_SHIFT32( Pka9 , h2 ,20), TL3 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka8 , h1 ,20), TL4 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka10 ,
        h2 ,20), TL4 ,20));
  K7 = (MUL_SHIFT32((MUL_SHIFT32( Pka4 , h1 ,20)+MUL_SHIFT32( Pka16 , h2 ,20)), alpha1 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka10 , h1 ,20)+
       MUL_SHIFT32( Pka22 , h2 ,20)), alpha3 ,20));
  K8 = (MUL_SHIFT32((MUL_SHIFT32( Pka4 , h1 ,20)+MUL_SHIFT32( Pka16 , h2 ,20)), alpha2 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka10 , h1 ,20)+
       MUL_SHIFT32( Pka22 , h2 ,20)), alpha4 ,20));
  x4 = (MUL_SHIFT32( K7 , beta1 ,20)+ x4_k1 +MUL_SHIFT32( K8 , beta2 ,20));
  P18 = ( Pka18 - MUL_SHIFT32( Pka30 , TL9 ,20)- MUL_SHIFT32( Pka36 , TL10 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka6 , h1 ,20), TL11 ,20)-
       MUL_SHIFT32(MUL_SHIFT32( Pka18 , h2 ,20), TL11 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka12 , h1 ,20), TL12 ,20)- MUL_SHIFT32(MUL_SHIFT32(
        Pka24 , h2 ,20), TL12 ,20));
  K11 = (MUL_SHIFT32((MUL_SHIFT32( Pka6 , h1 ,20)+MUL_SHIFT32( Pka18 , h2 ,20)), alpha1 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka12 , h1 ,20)+
       MUL_SHIFT32( Pka24 , h2 ,20)), alpha3 ,20));
  x6a =  _delayOutBuf1841;
  x6_k1 =  x6a ;
  K12 = (MUL_SHIFT32((MUL_SHIFT32( Pka6 , h1 ,20)+MUL_SHIFT32( Pka18 , h2 ,20)), alpha2 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka12 , h1 ,20)+((
       MUL_SHIFT32( Pka24 , h2 ,22))<<2)), alpha4 ,20));
  x6 = (MUL_SHIFT32( K11 , beta1 ,20)+ x6_k1 +MUL_SHIFT32( K12 , beta2 ,20));
  K9 = (MUL_SHIFT32((MUL_SHIFT32( Pka5 , h1 ,20)+MUL_SHIFT32( Pka17 , h2 ,20)), alpha1 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka11 , h1 ,20)+
       MUL_SHIFT32( Pka23 , h2 ,20)), alpha3 ,20));
  x5_k1 = (MUL_SHIFT32( A53 , x3a ,20)+MUL_SHIFT32( A54 , x4a ,20)+MUL_SHIFT32( A56 , x6a ,20)+MUL_SHIFT32( z1 , x5a ,20));
  K10 = (MUL_SHIFT32((MUL_SHIFT32( Pka5 , h1 ,20)+MUL_SHIFT32( Pka17 , h2 ,20)), alpha2 ,20)+MUL_SHIFT32((MUL_SHIFT32( Pka11 , h1 ,20)+
       MUL_SHIFT32( Pka23 , h2 ,20)), alpha4 ,20));
  t2358 = (MUL_SHIFT32( K9 , beta1 ,20)+ x5_k1 +MUL_SHIFT32( K10 , beta2 ,20));
  x5 =  t2358;
  P15 = ( Pka15 - MUL_SHIFT32( Pka17 , TL9 ,20)- MUL_SHIFT32( Pka18 , TL10 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka3 , h1 ,20), TL11 ,20)-
       MUL_SHIFT32(MUL_SHIFT32( Pka15 , h2 ,20), TL11 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka9 , h1 ,20), TL12 ,20)- MUL_SHIFT32(MUL_SHIFT32(
        Pka16 , h2 ,20), TL12 ,20));
  P3 = ( Pka3 - MUL_SHIFT32( Pka17 , TL1 ,20)- MUL_SHIFT32( Pka18 , TL2 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka3 , h1 ,20), TL3 ,20)-
       MUL_SHIFT32(MUL_SHIFT32( Pka15 , h2 ,20), TL3 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka9 , h1 ,20), TL4 ,20)- MUL_SHIFT32(MUL_SHIFT32(
        Pka16 , h2 ,20), TL4 ,20));
  P1 = ( Pka1 - MUL_SHIFT32( Pka5 , TL1 ,20)- MUL_SHIFT32( Pka6 , TL2 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka1 , h1 ,20), TL3 ,20)- MUL_SHIFT32(
       MUL_SHIFT32( Pka3 , h2 ,20), TL3 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka2 , h1 ,20), TL4 ,20)- MUL_SHIFT32(MUL_SHIFT32( Pka4 , h2 ,20),
        TL4 ,20));
  t3332 = MUL_SHIFT32(((long)(f280xReadAnalogVal(7))<<12),1342177280L,26)/* 20@fx8.32 */;
  t3332 = MIN(1090519040L,t3332);
  t3332 = MAX(t3332,0);
;
  t3346 = ( _sampBuf3342+MUL_SHIFT32(3355 /* 0.0002@fx8.32 */, t3332,24));
  t3343 = ((16777216 /* 1@fx8.32 */< t3346)?0 /* 0@fx8.32 */: t3346);
  t3370 = 8388608 /* 0.5@fx8.32 */;
  t3340 = (( _sampBuf3342)<<7);
  invClarke3338.ds = ((fxCos32( t3340))>>7);
  invClarke3338.qs = ((fxSin32( t3340))>>7);
  invclark32(&invClarke3338);
  t3335 = MUL_SHIFT32(((long)(f280xReadAnalogVal(4))<<12),354334801L,30)/* 0.33@fx8.32 */;
  t3335 = MIN(16777216L,t3335);
  t3335 = MAX(t3335,0);
;
  t3374 = ( t3370+MUL_SHIFT32(MUL_SHIFT32( invClarke3338.as, t3335,24),536870912L,30)/* 0.5@fx8.32 */);
  t3374 = MIN(15938355L,t3374);
  t3374 = MAX(t3374,838860L);
;
  t3372 = (int)(( t3374)>>9);
  t3375 = ( t3370+MUL_SHIFT32(MUL_SHIFT32( invClarke3338.bs, t3335,24),536870912L,30)/* 0.5@fx8.32 */);
  t3375 = MIN(15938355L,t3375);
  t3375 = MAX(t3375,838860L);
;
  t3371 = (int)(( t3375)>>9);
  t3376 = ( t3370+MUL_SHIFT32(MUL_SHIFT32( invClarke3338.cs, t3335,24),536870912L,30)/* 0.5@fx8.32 */);
  t3376 = MIN(15938355L,t3376);
  t3376 = MAX(t3376,838860L);
```

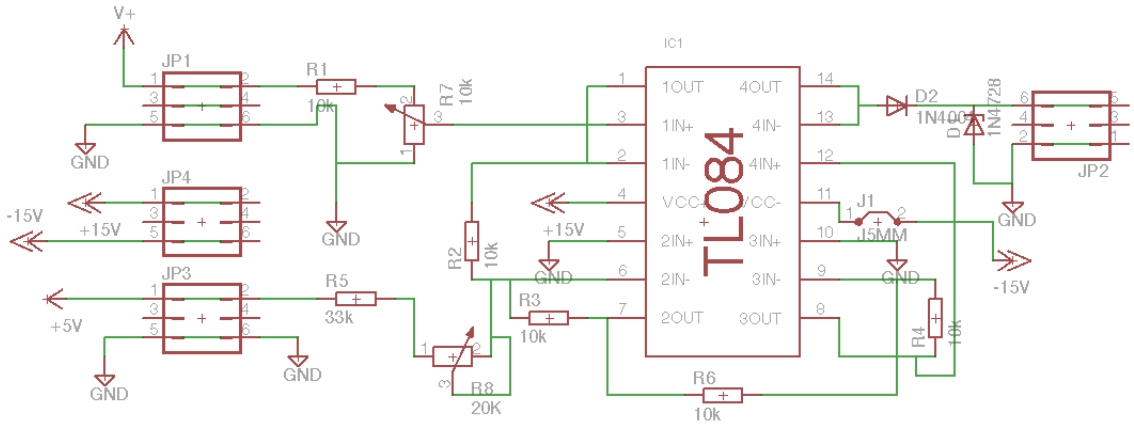
```
;
  t3373 = (int)(( t3376)>>9);
  { long _duty32 = (long) t3372*2000;
    CMPA1 = (int)(_duty32>>15);
  }
  CMPB1 = (int)(((long) t3372*2000)>>15);
  { long _duty32 = (long) t3371*2000;
    CMPA2 = (int)(_duty32>>15);
  }
  CMPB2 = (int)(((long) t3371*2000)>>15);
  { long _duty32 = (long) t3373*2000;
    CMPA3 = (int)(_duty32>>15);
  }
  CMPB3 = (int)(((long) t3373*2000)>>15);
  sim->outSigS[0].u.scaledInt.val =  t3332;
  sim->outSigS[1].u.scaledInt.val =  t3335;
  sim->outSigS[2].u.scaledInt.val =  va ;
  sim->outSigS[3].u.scaledInt.val =  vb ;
  sim->outSigS[4].u.scaledInt.val =  ia ;
  sim->outSigS[5].u.scaledInt.val =  ib ;
  sim->outSigS[6].u.scaledInt.val =  x1 ;
  sim->outSigS[7].u.scaledInt.val =  x2 ;
  sim->outSigS[8].u.scaledInt.val =  x3 ;
  sim->outSigS[9].u.scaledInt.val =  x4 ;
  sim->outSigS[10].u.scaledInt.val = MUL_SHIFT32( t2358,-1073741824L,30)/* -1@fx12.32 */;
  sim->outSigS[11].u.scaledInt.val =  x6 ;

  _delayOutBuf1825= x1 ;
  _delayOutBuf1873= P12 ;
  _delayOutBuf1912= P23 ;
  _delayOutBuf1856= P4 ;
  _delayOutBuf1864= P5 ;
  _delayOutBuf1905= P17 ;
  _delayOutBuf1808= ia ;
  _delayOutBuf1821= va ;
  _delayOutBuf1816= vb ;
  _delayOutBuf1813= ib ;
  _delayOutBuf1828= x2 ;
  _delayOutBuf1920= P29 ;
  _delayOutBuf1869= P6 ;
  _delayOutBuf1925= P30 ;
  _delayOutBuf1929= P36 ;
  _delayOutBuf1888= P8 ;
  _delayOutBuf1836= x3 ;
  _delayOutBuf1908= P24 ;
  _delayOutBuf1880= P11 ;
  _delayOutBuf1897= P16 ;
  _delayOutBuf1917= P22 ;
  _delayOutBuf1875= P10 ;
  _delayOutBuf1883= P9 ;
  _delayOutBuf1853= P2 ;
  _delayOutBuf1833= x4 ;
  _delayOutBuf1900= P18 ;
  _delayOutBuf1841= x6 ;
  _delayOutBuf1844= x5 ;
  _delayOutBuf1892= P15 ;
  _delayOutBuf1861= P3 ;
  _delayOutBuf1848= P1 ;
  _sampBuf3342 =  t3343;
  endOfSampleCount = TIMER2TIM;

}
```

```
main()
{
   noIntegrationUsed = 1;
   EALLOW;
   PLLSTS = 0x10; // reset clk check
   WDCR=0x00ef;  // Disable Watchdog
   asm(" clrc DBGM");
   if (!(PLLSTS&8))       // Skip PLL set if OSC failure
     { PLLSTS = 0x40;     //Disable OSC check
       PLLCR = 0xa;       // set PLL to 5xOSC = 100 MHZ;
       PLLSTS = 0x0;      //Enable OSC check (&F283xx /2 mode)
     }
   PCLKCR |= 0xc;
   HISPCP = 0x0;     // HCLK  = 100 MHZ
   PCLKCR1 = 0x7;
   PCLKCR3 = 0x400;
   EDIS;
   TBPRD1 = 0x7d0;
   AQCTLA1 = 0x90;
   AQCTLB1 = 0x600;
   ETSEL1 = 0x9000;
   ETPS1 = 0x1100;
   DBCTL1 = 0xb;
   DBRED1 = 0xfa;
   DBFED1 = 0xfa;
   EALLOW;
   TZCTL1 = 0x0;
   TZSEL1 = 0x0;
   EDIS;
   TBPRD2 = 0x7d0;
   AQCTLA2 = 0x90;
   AQCTLB2 = 0x600;
   ETSEL2 = 0x9000;
   ETPS2 = 0x1100;
   DBCTL2 = 0xb;
   DBRED2 = 0xfa;
   DBFED2 = 0xfa;
   EALLOW;
   TZCTL2 = 0x0;
   TZSEL2 = 0x0;
   EDIS;
   TBPRD3 = 0x7d0;
   AQCTLA3 = 0x90;
   AQCTLB3 = 0x600;
   ETSEL3 = 0x9000;
   ETPS3 = 0x1100;
   DBCTL3 = 0xb;
   DBRED3 = 0xfa;
   DBFED3 = 0xfa;
   EALLOW;
   TZCTL3 = 0x0;
   TZSEL3 = 0x0;
   EDIS;
   ADCTRL1_VAL = 0x300;
   ADCTRL2_VAL = 0x1;
   ADCTRL3_VAL = 0x6;
   EALLOW;
   GPAMUX1 = 0x555;
   GPBMUX1 = 0x0;
   EDIS;
   ADCTRL3 = 0xE0;  // Power up ADC
```

```
  ADCCHSELSEQ1 = 0x3210;
  ADCCHSELSEQ2 = 0x7654;
  ADCCHSELSEQ3 = 0xba98;
  ADCCHSELSEQ4 = 0xfedc;
  ADCMAX_CONV = 0x17;
 simInit( &tSim );
  startSimDsp();
  installInterruptVec(-2,7,cgMain);
  TIMER2PRD = 0x4e20; // 32-bit Timer Period Low
  TIMER2PRDH = 0x0; // 32-bit Timer Period High
  TIMER2TCR |= 0x4020; //Interrupt enable, Timer Reset
  EALLOW;
  PIECTRL = 1; // Enable PIE Interrupts
  EDIS;
  IER |= 0x2000; //CPU Interrupt enable
  resetInterrupts();
  disable_interrupts();  // Disable interrupts until PC handshake complete
  TBCTL1 = 0x412; // Start timer
  TBCTL2 = 0x412; // Start timer
  TBCTL3 = 0x412; // Start timer
  dspWait();
}
```

Planos de la placa para medición de voltaje.

Esquemático de la placa de medición de voltaje



Layout de la placa de medición de voltaje