UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

Estudio de los sistemas de control de la máquina síncrona de polos salientes

Memoria

Autor:
Manel Cugota

Director:
Dr. Santiago Bogarra

Proyecto para el grado en

Ingeniería Electrónica Industrial y Automática

Departamento de ingeniería eléctrica
Escola Superior d'Enginyeries Industrial Aeroespacial i Audiovisual de Terrassa (ESEIAAT)

10 de Mayo, 2019

INDICE

LISTADO DE FIGURAS

LISTADO DE TABLAS

AGRADECIMIENTOS

En primera instancia, me gustaría agradecer al director de mi proyecto, Dr. Santiago Bogarra, por la dedicación que ha tenido conmigo durante la elaboración de este proyecto, ayudándome en momentos en los que no encontraba soluciones, y guiándome para que yo mismo consiguiera los objetivos.

También es de obligada mención la ayuda que Albert Masip y Fatiha Nejjari, del departamento de control, me han brindado en determinados momentos clave.

1. INTRODUCCIÓN

El título de este proyecto es "*Estudio de los sistemas de control de la máquina síncrona de polos salientes* " y está realizado por el estudiante Manel Cugota Canals y dirigido por el Dr. Santiago Bogarra Rodríguez.

1.2. RESUMEN

El objetivo del trabajo es realizar un control sobre la excitación de una máquina síncrona de polos salientes que se amolde de manera correcta a la normativa correspondiente.

Posteriormente se ha procedido a la integración del lazo de control resultante dentro del propio bloque de la máquina síncrona, para así conseguir reducir el máximo posible el tiempo que tarda el programa en realizar el proceso de simulación.

Para ello se ha estudiado el comportamiento de la máquina síncrona y las normativas de estandarización, junto con el lenguaje requerido por el programa *Simscape* que nos ha permitido realizar la programación interna del bloque.

1.3. ANTECEDENTES

El trabajo parte de un proyecto de final de grado realizado anteriormente en la misma universidad, correspondiente al departamento de ingeniería eléctrica y dirigido por el Dr. Santiago Bogarra y con el proyectista Guillem Duarri.

Dicho proyecto consistió en estudiar muy detenidamente el funcionamiento, comportamiento y las fórmulas que describían la máquina síncrona de polos salientes, para posteriormente, realizar un modelo mediante *Simscape* que permitiera la simulación de dicha máquina con diferentes excitaciones de entrada, ya fueran eléctricas o mecánicas.

1.4. PREFACIO

Mediante este proyecto, se pretende ajustar la respuesta del modelo de máquina síncrona a una pequeña variación en la excitación de entrada mediante el añadido del sistema de control desarrollado.

Teniendo en cuenta que adquirir una máquina síncrona de polos salientes, y los componentes del lazo de control para su posterior estudio sería algo realmente costoso, este trabajo se podrá considerar como una herramienta didáctica y de estudio para futuros proyectos de investigación.

1.5. ESPECIFICACIONES BÁSICAS

Las especificaciones básicas de este proyecto se basan en disponer de un ordenador con el programa Matlab instalado con todos sus componentes secundarios (requiere que durante la instalación de Matlab se ordene instalar todos los subprogramas adyacentes, como pueden ser para citar alguna a modo ejemplo: *Simulink, Simscape*,etc.)

También es necesario mencionar, que debido a la potencia de la simulación de programa, es posible que en algunos ordenadores de potencia justa, la simulación tarde mucho más de lo esperado.

1.6. ESTRUCTURA

Con tal de facilitar al lector de la memoria el entendimiento de esta, seguidamente se relata de manera breve que estructura sigue el trabajo.

Primeramente, iniciaremos la parte teórica explicando que es una máquina síncrona de polos salientes dando a conocer las fórmulas teóricas que describen su funcionamiento, con tal de poder hacer entender al lector el comportamiento interno de esta.

Una vez se ha introducido la máquina síncrona, se plantea la necesidad de que el proyecto esté acotado por una normativa de estándares, que nos van a marcar ya de primeras algunos apartados importantes del control de esta.

Con la normativa explicada se va a proceder a la obtención de unos parámetros que encajen en las demandas de esta mediante los cálculos necesarios.

En este punto deberíamos tener ya realizado el esquema completo de lo que vendría a ser nuestro proyecto, si más no, una forma tosca que nos describa el resultado final.

Para poder pulir lo que hemos logrado, el último paso va a ser introducir en el bloque de la máquina síncrona todo el lazo de control que hemos obtenido, y que ahora se encuentra representado por bloques, a código, para así optimizar el tiempo de simulación.

Como resultado final se van a mostrar os gráficos comparativas del sistema representado mediante bloques, y el sistema pasado a código, para mostrar que su comportamiento es idéntico.

1.7. CONTEXTO

1.7.1. MÁQUINAS SÍNCRONAS

La mayoría de las fuentes de energía eléctrica en la actualidad son generadores síncronos. Si ellos son impulsados por fuerza hidráulica, vapor, aerogeneradores o motores de combustión, son los principales medios para convertir energía mecánica en energía eléctrica [1] en edificios, Infraestructuras y, por supuesto, vehículos como aviones de avión.

1.7.2. SIMSCAPE

Este entorno de simulación basado en MATLAB se puede usar dentro de Simulink, pero es diferente de él en el sentido de que funciona con el enfoque de la Red Física en lugar de trabajar con señales simples. Esto hace que el modelado de sistemas físicos sea más visual e intuitivo. [1]

2. MODELADO DE LA MÁQUINA SÍNCRONA

2.1. DESCRIPCIÓN DE LA MÁQUINA SÍNCRONA

La máquina síncrona es un convertidor electromecánico de energía con una pieza giratoria denominada rotor, cuya bobina se excita mediante la inyección de corriente continua, y una pieza fija denominada estator o armadura por cuyas bobinas circula corriente alterna. Las corrientes alternas que circulan por los devanados estatóricos producen un campo magnético rotatorio que gira en el entrehierro de la máquina con la frecuencia angular de las corrientes se armadura. El rotor debe girar a la misma velocidad que el campo magnético rotatorio producido en el estator para que el par eléctrico medio pueda ser diferente de cero. Si las velocidades angulares del campo magnético rotatorio y del rotor de la máquina síncrona son diferentes, el par eléctrico medio es nulo. Por esta razón a esta máquina se la denomina síncrona; el rotor gira mecánicamente a la misma frecuencia que el campo magnético rotatorio del estator durante la operación en régimen permanente. En la figura 1 se muestra el esquema básico de una máquina síncrona de polos salientes.

Durante la operación en régimen permanente de la máquina síncrona, la velocidad mecánica del rotor es igual a la velocidad angular del campo magnético rotatorio producido en el estator. En estas condiciones, sobre los conductores o bobinas de campo no se induce fuerza electromotriz. Para producir fuerza magneto-motriz en el rotor es necesario inyectar corriente en esta bobina mediante una fuente externa. De esta forma se obtienen dos campos magnéticos rotatorios que giran a la misma velocidad, uno producido por el estator y otro por el rotor. Estos campos interactúan produciendo par eléctrico medio y se realiza el proceso de conversión electromecánica de energía.

La bobina del rotor o campo de la máquina síncrona se alimenta mediante la inyección de corriente continua, como se ha mencionado anteriormente, con la finalidad de producir un campo magnético de magnitud constante, semejante al de un imán permanente, pero de una intensidad mucho mayor. Debido a que el rotor de la máquina gira en régimen permanente a la velocidad síncrona, el campo magnético constante producido se comporta, desde el punto de vista del estator, como un campo magnético rotatorio. Para evaluar la magnitud del par en una máquina síncrona se puede utilizar la siguiente expresión:

$$T_e = k \cdot \mathbf{F}_r \mathbf{F}_e \sin\delta \qquad (1)$$

Donde *k* es una constante de proporcionalidad que depende de la geometría de la máquina y de la disposición física de la bobinas, $\mathbf{F}_e$ es la amplitud de la distribución sinusoidal de la fuerza magneto-motriz del estator, $\mathbf{F}_r$ es la amplitud de la distribución sinusoidal de la fuerza magneto-motriz del rotor y δ es el ángulo entre las amplitudes de las dos fuerzas magneto-motrices, conocido generalmente como ángulo de carga o de par.

Las fuerzas magneto-motrices del estator $\mathbf{F}_e$, y del rotor $\mathbf{F}_r$ tienen una amplitud constante y para que en la expresión (1) el par medio resulte constante, es necesario que el ángulo δ entre las dos fuerzas magneto-motrices sea constante en el tiempo durante la operación en régimen permanente. Para lograr esto, las dos fuerzas magneto-motrices deben girar a la misma velocidad angular.

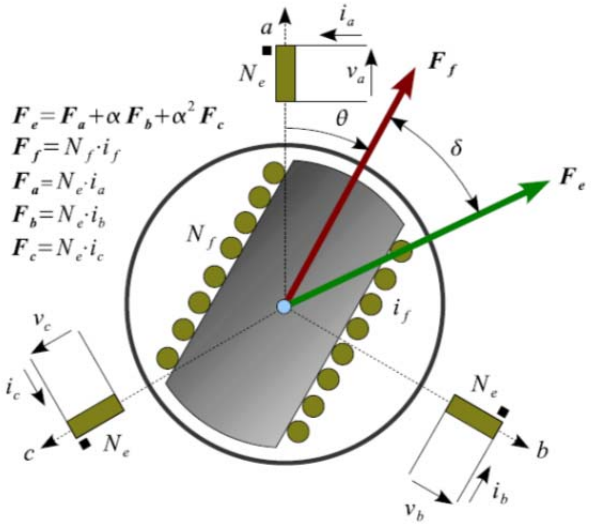

Figura 1. Esquema básico de una máquina síncrona de polos salientes. Fuente: Aller [2].

2.2. MODELO DE LA MÁQUINA SÍNCRONA

Analizando el comportamiento de los ejes eléctricos de la máquina síncrona en el sistema de coordenadas correspondiente a las bobinas reales o físicas, se satisface el siguiente sistema de ecuaciones:

$$[v_{abc,f}] = [R_{abc,f}][i_{abc,f}] + \frac{d}{dt}[\lambda_{abc,f}] \tag{2}$$

En los sistemas lineales, la relación entre las corrientes que circulan por las bobinas y los flujos concatenados vienen dados por la relación:

$$[\lambda_{abc,f}(\theta, i)] = [L_{abc,f}(\theta)][i_{abc,f}] \tag{3}$$

Sustituyendo esta relación en la expresión (2) se obtiene el resultado siguiente:

$$\begin{aligned}[v_{abc,f}] &= [R_{abc,f}][i_{abc,f}] + [L_{abc,f}]\frac{d}{dt}[i_{abc,f}] + \frac{d\theta}{dt}\frac{d}{dt}[L_{abc,f}][i_{abc,f}] = \\ &= [R_{abc,f}][i_{abc,f}] + [L_{abc,f}]p[i_{abc,f}] + \dot{\theta}[\tau_{abc,f}][i_{abc,f}]\end{aligned} \tag{4}$$

El sistema de ecuaciones diferenciales (4) representa el comportamiento dinámico de las bobinas de la máquina síncrona en coordenadas primitivas. Este sistema se expresa en forma canónica como:

$$p[i_{abc,f}] = [L_{abc,f}]^{-1}\left\{[v_{abc,f}] - \left[[R_{abc,f}] + \dot{\theta}[\tau_{abc,f}]\right][i_{abc,f}]\right\} \quad (5)$$

La matriz de inductancia $[L_{abc,f}]$ depende de la posición relativa ϑ del rotor con respecto al estator, por esta razón la matriz de transición de estado también depende de la posición angular del rotor. Si la velocidad de la máquina es constante, la posición angular del rotor es:

$$\theta = \theta_0 + \omega_m t \quad (6)$$

Durante los períodos transitorios, la velocidad angular de la máquina cambia y la posición angular del rotor es una nueva variable de estado que debe ser evaluada para determinar su dependencia temporal. En este caso es necesario incorporar una ecuación adicional al sistema (5) para determinar el comportamiento dinámico del eje mecánico de la máquina:

$$\frac{1}{2}[i_{abc,f}]^t[\tau_{abc,f}][i_{abc,f}] - T_m = J\ddot{\theta} + \alpha\dot{\theta} \quad (7)$$

Esta expresión representa el balance del par eléctrico y mecánico en el eje del rotor. El par acelerador es igual al par eléctrico menos el par resistente opuesto por la carga y por las pérdidas mecánicas. La ecuación diferencial (7) puede ser expresada mediante dos ecuaciones diferenciales de primer orden:

$$\begin{cases} \dot{\omega}_m = \frac{1}{J}\left(\frac{1}{2}[i_{abc,f}]^t[\tau_{abc,f}][i_{abc,f}] - T_m - \alpha\dot{\theta}\right) \\ \dot{\theta} = \omega_m \end{cases} \quad (8)$$

Donde J es el momento de inercia del rotor (en kg m^2), T_m es el par mecánico (en N m) y α es el coeficiente de fricción dinámica (en N m s).

El sistema de seis ecuaciones diferenciales formado por las cuatro ecuaciones del sistema (5), y las dos ecuaciones mecánicas representadas por la expresión (8), definen el comportamiento dinámico y transitorio completo de la máquina síncrona de la Figura 2. Este sistema de ecuaciones diferenciales es no lineal y los coeficientes son variables en el tiempo, por este motivo es necesario recurrir a técnicas numéricas para evaluar el comportamiento de la máquina o simplificar el problema mediante la técnica de transformación de coordenadas.

En la matriz de inductancia de la máquina síncrona, se encuentra toda la información necesaria para determinar su comportamiento. En la matriz de inductancia se resume la información sobre la disposición geométrica de las bobinas, sus acoplamientos, números de

vueltas y reluctancias de los diferentes caminos magnéticos. Una vez conocida la matriz de inductancias se puede evaluar la matriz de par calculando la derivada parcial de esta matriz con respecto a la posición angular del rotor. La matriz de inductancias de la máquina síncrona esquematizada en la figura 1 posee la siguiente estructura:

$$[L_{abc,f}(\theta)] = \begin{bmatrix} [L_{ee}(\theta)] & [L_{ef}(\theta)] \\ [L_{fe}(\theta)] & L_f \end{bmatrix} \tag{9}$$

$$[L_{ee}(\theta)] = \begin{bmatrix} L_{aa}(\theta) & M_{ab}(\theta) & M_{ac}(\theta) \\ M_{ba}(\theta) & L_{bb}(\theta) & M_{bc}(\theta) \\ M_{ca}(\theta) & M_{cb}(\theta) & L_{cc}(\theta) \end{bmatrix} \tag{10}$$

$$[L_{ef}(\theta)] = [L_{fe}(\theta)]^t = \begin{bmatrix} M_{af}(\theta) \\ M_{bf}(\theta) \\ M_{cf}(\theta) \end{bmatrix} \tag{11}$$

Donde *e* es el subíndice referido a las bobinas del estator, *f* es el subíndice referido a las bobinas de campo y *a*, *b*, *c* son los subíndices de las tres bobinas físicas del estator.

Para evaluar cada una de las inductancias definidas en las expresiones (9), (10) y (11), es necesario utilizar las expresiones 3.10 y 3.11 desarrolladas en el capítulo 3 del libro *Máquinas Eléctricas Rotativas: Introducción a la Teoría General* [2].

Cada una de las inductancias de la máquina síncrona se puede representar como una función del ángulo ϑ. Esta función es periódica porque se repite nuevamente cada vez que el rotor realiza un giro completo. Esta propiedad permite representar estas funciones mediante expansiones en series de Fourier, con el ángulo ϑ como variable.

La inductancia del rotor L_f, es independiente de la posición ϑ del rotor debido a que el estator de la máquina, despreciando el efecto de las ranuras del estator, es aproximadamente liso. El resto de las inductancias propias y mutuas depende de la posición angular ϑ, si el rotor de la máquina es de polos salientes. Las permeancias de los caminos magnéticos de las bobinas del estator y de los acoplamientos estator-rotor son dependientes de la posición angular ϑ. Cuando la pieza polar del rotor se encuentra alineada con una de las bobinas del estator, el camino magnético posee la máxima permeancia. Si la pieza polar se encuentra en cuadratura con la bobina, el entrehierro es muy grande y disminuye la permeancia. Estas inductancias se pueden representar mediante las funciones que aparecen en el capítulo 8 del libro *Máquinas Eléctricas Rotativas: Introducción a la Teoría General* [2].

El modelo de la máquina síncrona se puede obtener mediante transformaciones del sistema de coordenadas que permiten simplificar notablemente estos modelos.

2.3. TRANSFORMACIÓN DE PARK

En la máquina síncrona, el campo magnético rotatorio producido por las fuerzas magneto-motrices de los devanados estatóricos, gira a la velocidad síncrona ω_e. Por esta razón es conveniente referir las ecuaciones diferenciales que definen el comportamiento de la máquina a un sistema de coordenadas solidario con el rotor. De acuerdo con estos razonamientos se definen los siguientes ejes magnéticos:

- Eje *d*: Gira respecto al estator a la velocidad del rotor, se encuentra en la misma dirección que el eje magnético del campo.
- Eje *q*: Rota respecto al estator a la velocidad del rotor, y en todo momento se encuentra perpendicular al eje magnético del campo.
- Eje 0: Fijo en el estator y se encuentra desacoplado magnéticamente del resto de los ejes de la máquina.
- Eje *f*: Solidario con el sistema rotórico y se encuentra en la misma dirección que el eje magnético de la bobina de campo.

Aun cuando los ejes *d* y *q* giran a igual velocidad que el rotor, estos ejes representan variables del estator. El eje 0 es necesario para permitir que la transformación de coordenadas sea bidireccional, es decir, se pueda transformar variables primitivas a variables *dq*0 y viceversa. El eje 0 tiene una estrecha relación con las variables de secuencia cero de la transformación en componentes simétricas. En la práctica el eje 0 permite representar flujos de dispersión que no están acoplados con otras bobinas de la máquina. En la figura2 se ha representado el sistema de coordenadas *dq*0-*f*.

La matriz de transformación de coordenadas *dq*0-*f* a coordenadas primitivas se define mediante la relación:

$$[i_{abc,f}] = [A][i_{dq0,f}] \tag{12}$$

Si la transformación anterior se escoge de tal forma que la matriz [*A*] sea hermitiana (la inversa es igual a la transpuesta conjugada), la transformación es conservativa en potencia. Cuando la matriz es hermitiana y real, se obtiene:

$$[i_{dq0,f}] = [A]^{-1}[i_{abc,f}] = [A]^{t}[i_{abc,f}] \tag{13}$$

La matriz de transformación [*A*] se puede obtener multiplicando la transformación de coordenadas primitivas a coordenadas ortogonales *αβ*0 (transformación de Clark), por la transformación de coordenadas *αβ*0 a coordenadas *dq*0 (rotación en ϑ):

$$\begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & 0 & \frac{1}{\sqrt{2}} \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & \frac{1}{\sqrt{2}} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \\ i_0 \end{bmatrix} \tag{14}$$

$$\begin{bmatrix} i_\alpha \\ i_\beta \\ i_0 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} i_d \\ i_q \\ i_0 \end{bmatrix} \tag{15}$$

$$\begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \cos\theta & -\sin\theta & \frac{1}{\sqrt{2}} \\ \cos\left(\theta - \frac{2\pi}{3}\right) & -\sin\left(\theta - \frac{2\pi}{3}\right) & \frac{1}{\sqrt{2}} \\ \cos\left(\theta - \frac{4\pi}{3}\right) & -\sin\left(\theta - \frac{4\pi}{3}\right) & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} i_d \\ i_q \\ i_0 \end{bmatrix} \tag{16}$$

La matriz de la expresión (16) se conoce como transformación de Park. La transformación de coordenadas primitivas *abc*,*f* a coordenadas *dq*0,*f* es:

$$\begin{bmatrix} i_d \\ i_q \\ i_0 \\ i_f \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \cos\theta & \cos\left(\theta - \frac{2\pi}{3}\right) & \cos\left(\theta - \frac{4\pi}{3}\right) & 0 \\ -\sin\theta & -\sin\left(\theta - \frac{2\pi}{3}\right) & -\sin\left(\theta - \frac{4\pi}{3}\right) & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & \sqrt{\frac{3}{2}} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \\ i_f \end{bmatrix} \tag{17}$$

La transformación de Park utilizada es hermitiana y por tanto es invariante en potencia:

$$\begin{aligned} p(t) &= [v_{abc,f}]^t [i_{abc,f}] = \left[[A][v_{dq0,f}]\right]^t \left[[A][i_{dq0,f}]\right] = \\ &= [v_{dq0,f}]^t [A]^t [A][i_{dq0,f}] = [v_{dq0,f}]^t [i_{dq0,f}] = p(t) \end{aligned} \tag{18}$$

Aplicando la transformación (17), al sistema de ecuaciones (4), se obtiene:

$$[v_{dq0,f}] = [R_{dq0,f}][i_{dq0,f}] + [L_{dq0,f}]p[i_{dq0,f}] + \dot{\theta}[G_{dq0,f}][i_{dq0,f}] \tag{19}$$

Dónde:

$$[R_{dq0,f}] = [A]^t [R_{abc,f}][A] \tag{20}$$

$$[L_{dq0,f}] = [A]^t [L_{abc,f}][A] \tag{21}$$

$$[G_{dq0,f}] = [\tau_{dq0,f}] + [H_{dq0,f}] = [A]^t [\tau_{abc,f}][A] + [A]^t [R_{abc,f}] \frac{d}{d\theta}[A] \qquad (22)$$

Por otra parte, la ecuación dinámica del movimiento se puede expresar de la siguiente forma:

$$J\ddot{\theta} + \alpha\dot{\theta} = \frac{1}{2}[i_{dq0,f}]^t [\tau_{dq0,f}][i_{dq0,f}] - T_m \qquad (23)$$

Evaluando explícitamente las expresiones (20) a (22) y sustituyendo el resultado en las ecuaciones diferenciales (19) y (23) se obtiene:

$$\begin{bmatrix} v_d \\ v_q \\ v_0 \\ v_f \end{bmatrix} = \begin{bmatrix} R_e + L_d p & -\omega L_q & 0 & L_{df} p \\ \omega L_d & R_e + L_q p & 0 & \omega L_{df} \\ 0 & 0 & R_0 + L_0 p & 0 \\ L_{df} p & 0 & 0 & R_f + L_f p \end{bmatrix} \begin{bmatrix} i_d \\ i_q \\ i_0 \\ i_f \end{bmatrix} \qquad (24)$$

$$Jp\omega = \left(L_d - L_q\right) i_d i_q + L_{df} i_q i_f - \alpha\omega - T_m \qquad (25)$$

En un sistema trifásico sin neutro no circula corriente de secuencia cero, pero cuando las tres corrientes de fase encuentran un camino de retorno, es necesario considerar esta componente. La componente de secuencia cero representa la circulación de corrientes iguales y en fase por las bobinas de la máquina. Estas corrientes no producen magnetización debido a que la suma de las fuerzas magneto-motrices de las tres bobinas es cero. Sin embargo, los flujos de dispersión si poseen componente de secuencia cero. En el modelo de la máquina no existe acoplamiento magnético de esta secuencia con el resto de la bobinas. Esta componente no puede producir par eléctrico, pero influye en las pérdidas de la máquina y en las fuerzas electromotrices sobre las bobinas.

En la expresión (24) no aparecen fuerzas electromotrices de generación sobre la bobina de campo. Esto se debe a que el sistema de coordenadas *dq*0 es solidario al eje *f* del campo. Los flujos de las bobinas *d* y *q* no cruzan tangencialmente a los conductores de campo. Sin embargo, en este eje pueden aparecer fuerzas electromotrices por transformación, debido a que el flujo de la bobina del eje directo atraviesa el devanado de campo. Por el contrario, el eje cuadratura no puede producir ningún efecto sobre el campo debido a que se encuentra permanentemente en posición ortogonal.

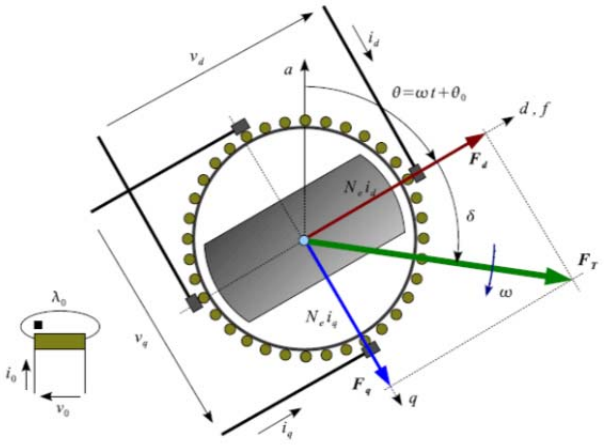

Figura 2. Modelo en coordenadas dq0-f de la máquina síncrona. Fuente: Aller[2]

La máquina síncrona puede ser representada por un modelo físico en coordenadas *dq*0-*f* como el de la figura 2 satisface las ecuaciones (24) y (25). En la máquina real, las corrientes i_d e i_q no circulan por ningún devanado físico, para determinar las corrientes reales es necesario aplicar la transformación inversa de coordenadas *dq*0-*f* a coordenadas primitivas. Si en las bobinas primitivas se inyecta un sistema equilibrado de corrientes trifásicas se obtienen las siguientes corrientes en el sistema de coordenadas *dq*0:

$$\begin{bmatrix} i_d \\ i_q \\ i_0 \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \cos\theta & \cos\left(\theta-\frac{2\pi}{3}\right) & \cos\left(\theta-\frac{4\pi}{3}\right) \\ -\sin\theta & -\sin\left(\theta-\frac{2\pi}{3}\right) & -\sin\left(\theta-\frac{4\pi}{3}\right) \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \sqrt{2} I_e \begin{bmatrix} \cos(\omega t+\alpha) \\ \cos\left(\omega t+\alpha-\frac{2\pi}{3}\right) \\ \cos\left(\omega t+\alpha-\frac{4\pi}{3}\right) \end{bmatrix} \quad (26)$$

$$\begin{bmatrix} i_d \\ i_q \\ i_0 \end{bmatrix} = \sqrt{3} I_e \begin{bmatrix} \cos(\theta-\omega t-\alpha) \\ -\sin(\theta-\omega t-\alpha) \\ 0 \end{bmatrix}$$

Si la posición angular ϑ del rotor se sincroniza con la variación angular de las corrientes ($\vartheta = \omega t + \vartheta_0$) en la expresión (26), las corrientes en las coordenadas *dq*0 son independientes del tiempo. En esta condición, los términos que dependen de las derivadas de las corrientes se anulan. Corrientes constantes en el tiempo en este sistema de coordenadas, producen fuerzas magneto-motrices constantes en las bobinas *dq*0 transformadas. Como la transformación está sincronizada con la velocidad angular de las corrientes durante el régimen permanente, el campo magnético producido por las bobinas *d* y *q*, gira con la misma velocidad y como resultado se obtiene el mismo campo magnético giratorio de la máquina síncrona en coordenadas primitivas, excitada mediante un sistema trifásico equilibrado de corrientes.

3. MODELO DE EXCITACIÓN

3.1. INTRODUCCIÓN

Para empezar este apartado debemos tener en cuenta que la función básica de un sistema de excitación es la de suministrar corriente continua al devanado de campo de una máquina síncrona.

A parte de esta, también cubre otras funciones como son la de control y protección, de mucha importancia para el correcto funcionamiento del sistema, y por lo tanto del control de corriente de excitación.

Con funciones de control, nos referimos también al control de la tensión y de la potencia reactiva, y es por este motivo que afirmamos que los sistemas del control de la excitación nos sirven para mantener el nivel de tensión deseado, controlar la potencia reactiva y mejorar la estabilidad del sistema.

Las funciones de protección por su parte, nos aseguran que los límites de capacidad del sistema de excitación y de la propia máquina síncrona no son excedidos.

3.2. ELECCIÓN DEL MODELO

El control de la tensión en bornes de la máquina se lleva a cabo mediante el *Automatic Voltage Regulator* (AVR), que actúa subministrando la corriente necesaria al devanado de campo del generador mediante la excitatriz, para así mantener el nivel de tensión deseado.

Para definir el AVR, de nuestra máquina síncrona necesitamos tomar como punto de partida un modelo de excitación que cumpla la normativa y parametrizarlo para que se comporte de la manera que nosotros deseamos.

Se decidió implementar el modelo AC8B de la versión de 2005 que se ve sobrepasado por la siguiente versión, la AC8C. Aunque esto no representa ningún problema, ya que si fuera preciso representar el resultado de este proyecto con el modelo AC8C no existiría ningún problema gracias a su similitud y, en consecuencia, compatibilidad.

Algunas de las diferencias más significativas serian, por ejemplo, que el modelo AC8C permite diferentes opciones para la conexión de los limitadores de excitación, tanto superior como inferior, de las que el modelo AC8B no dispone.

También que el modelo AC8C ofrece una mayor flexibilidad a la hora de representar las fuentes de alimentación del rectificador controlado, conectado a la bobina del campo de excitación rotativa. Y es a consecuencia de este último punto mencionado que el modelo AC8C requiere de parámetros adicionales en comparación a su precursor.

A continuación, podemos observar en la figura 3 el esquema del modelo AC8C y en la figura 4 el esquema del modelo AC8B

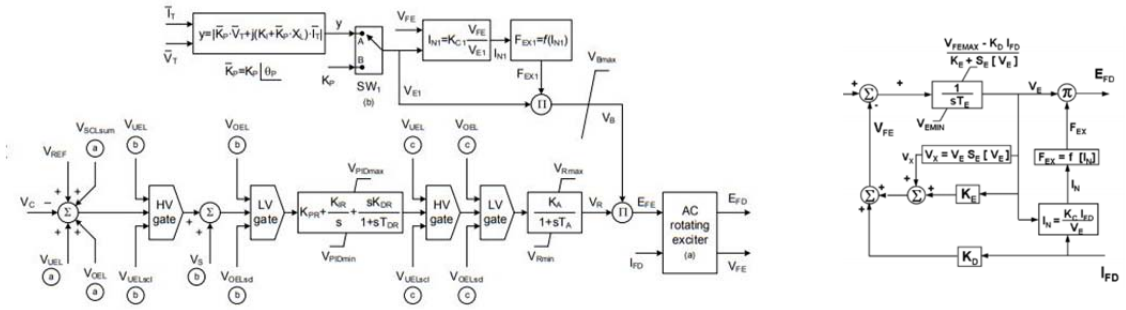

Figura 3. Diagrama de bloques del modelo de excitación AC8C

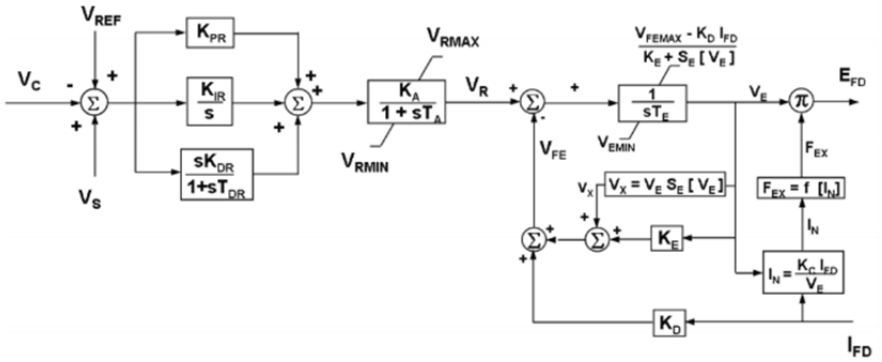

Figura 4. Diagrama de bloques del modelo de excitación AC8B

3.3. MODELO AC8B

El AVR en este modelo consiste en un control PID, con constantes separadas para la ganancia proporcional K_{PR}, la ganancia integral K_{IR} y la ganancia derivativa K_{DR}. Los valores de las ganancias del controlador PID se seleccionan para obtener el mejor funcionamiento posible para cada sistema de excitación en particular. El alternador sin escobillas se representa por medio de T_E, K_E y S_E.

El parámetro K_E es función de la resistencia de campo de la excitatriz y de la pendiente de la línea de entrehierro en la curva que relaciona la tensión de salida de la excitatriz y la corriente de campo de la excitatriz. La función de saturación S_E en por unidad se calcula

mediante la curva de saturación en vacío y la línea de entrehierro como veremos más adelante.

Además, para prevenir el sobrecalentamiento del devanado rotatorio y pérdidas de sincronismo, se utilizan limitadores de sobre y subexcitación en los controladores del sistema de excitación.

Al tratarse de un AVR tipo AC8B con control digital los parámetros K_C y K_D son 0.

El resto de parámetros, excepto las ganancias del controlador, se han aplicado los recomendados por la IEEE en [3], que se muestran en la siguiente tabla

Descripción	Parámetro	Valor	Unidades
Constante de tiempo del filtro derivativo del controlador	T_{DR}	0.03	s
Ganancia de la salida del controlador	K_A	1	pu
Constante de tiempo de la salida del controlador	T_A	0	s
Máxima salida del controlador	V_{RMAX}	35	pu
Mínima salida del controlador	V_{RMIN}	0	pu
Máxima corriente de campo	V_{FEMAX}	6	pu
Constante proporcional de la excitatriz	K_E	1.0	pu
Constante de tiempo de la excitatriz	T_E	1.2	s
Factor de rectificador de carga proporcional a la reactancia de conmutación	K_C	0	pu
Factor desmagnetizante	K_D	0	pu
Factor de saturación para E_1	$SE_{(E1)}$	0.3	-
Flujo en la excitatriz para SE_1	E_1	6.5	pu
Factor de saturación para E_2	$SE_{(E2)}$	3.0	-
Flujo en la excitatriz para SE_2	E_2	9.0	pu

Tabla 1. Parámetros para el sistema de excitación AC8B

3.3.1. CONTROLADOR PID

Un controlador PID es un dispositivo que permite controlar un sistema en lazo cerrado para que alcance el estado de salida deseado. El controlador PID está compuesto de tres elementos que proporcionan una acción Proporcional, Integral y Derivativa. Estas tres acciones son las que dan nombre al controlador PID. Ogata [4].

La entrada al controlador PID es la señal de error. Esta señal indica al controlador la diferencia que existe entre el estado que se quiere conseguir o referencia y el estado real

del sistema medido por el transductor. Si la señal de error es grande, significa que el estado del sistema se encuentra lejos del estado de referencia deseado. Si por el contrario el error es pequeño, significa que el sistema ha alcanzado el estado deseado.

3.3.1.1. ACCIÓN PROPORCIONAL

Como su nombre indica, esta acción de control es proporcional a la señal de error. Internamente la acción proporcional multiplica la señal de error por una constante *Kp*.

Esta acción de control intenta minimizar el error del sistema. Cuando el error es grande, la acción de control es grande y tiende a minimizar este error.

Aumentar la acción proporcional *Kp* tiene los siguientes efectos:

1. Aumenta la velocidad de respuesta del sistema.
2. Disminuye el error del sistema en régimen permanente.
3. Aumenta la inestabilidad del sistema.

Aunque efectos son positivos y deseables. El último efecto es negativo y hay que intentar minimizarlo. Por lo tanto al aumentar la acción proporcional existe un punto de equilibrio en el que se consigue suficiente rapidez de respuesta del sistema y reducción del error, sin que el sistema sea demasiado inestable.

3.3.1.2. ACCIÓN INTEGRAL

Esta acción de control como su nombre indica, calcula la integral de la señal de **error**. La integral se puede ver como la suma o acumulación de la señal de error. A medida que pasa el tiempo pequeños errores se van sumando para hacer que la acción integral sea cada vez mayor. Con esto se consigue reducir el error del sistema en régimen permanente. La desventaja de utilizar la acción integral consiste en que esta añade una cierta inercia al sistema y por lo tanto le hace más inestable.

Aumentar la acción integral *Ki* tiene los siguientes efectos:

1. Disminuye el error del sistema en régimen permanente.
2. Aumenta la inestabilidad del sistema.
3. Aumenta un poco la velocidad del sistema.

Esta acción de control servirá para disminuir el error en régimen permanente.

3.3.1.3. ACCIÓN DERIVATIVA

Como su nombre indica, esta acción de control es proporcional a la derivada de la señal de error. La derivada del error es otra forma de llamar a la "velocidad" del error.

Cuando la posición se encuentra por debajo la referencia, la acción de control proporcional siempre intenta aumentar la posición. El problema viene al tener en cuenta las inercias. Cuando el sistema se mueve a una velocidad alta hacia el punto de referencia, el sistema se pasará de largo debido a su inercia. Esto produce un sobrepulso y oscilaciones en torno a la referencia. Para evitar este problema, el controlador debe reconocer la velocidad a la que el sistema se acerca a la referencia para poder frenarle con antelación a medida que se acerque a la referencia deseada y evitar que la sobrepase.

Aumentar la constante de control derivativa *Kd* tiene los siguientes efectos:

1. Aumenta la estabilidad del sistema controlado.
2. Disminuye un poco la velocidad del sistema.
3. El error en régimen permanente permanecerá igual.

Esta acción de control servirá por lo tanto para estabilizar una respuesta que oscile demasiado.

3.3.1.4. RESPUESTA DINÁMICA DE UN SISTEMA

Se entiende por la respuesta dinámica de un sistema, el comportamiento de dicho sistema a una excitación.

En la figura 5 nos muestra una respuesta dinámica típica de una máquina síncrona controlada en una conexión de circuito abierto, cuando hay un cambio en forma de step en la excitación del sistema (cambio en el voltaje de referencia).

Para facilitar el entendimiento para aquellos que no estén familiarizados con las especificaciones propias del sistema de control, a continuación se explican las esenciales:

- Respuesta en régimen permanente (*Steady*-st*a*te *value*): Hace referencia al valor que obtendríamos en la salida del sistema si en algún punto del tiempo del experimento llegara a estabilizarse por completo sin ningún tipo de oscilación.
- Valor de pico (*Peak Value*): es el valor máximo al que llega la salida del sistema. Generalmente, después de alimentar el sistema con un *Step*, este reacciona con un pico en la salida, Y a medida que transcurre el tiempo, cada oscilación en la salida es menos brusca. Es por este motivo que el valor de pico lo vamos a encontrar siempre cerca del instante de tiempo en el que se empieza a excitar el sistema.
- Tiempo de retardo (*Delay time*): Como bien su propio nombre indica, hace referencia al tiempo que tarda el sistema en recibir una excitación o input necesarios para generar una respuesta u output.
- Sobre pico (*Overshoot*): Este valor corresponde a la medida del valor de pico en % respecto al valor de la respuesta permanente del sistema.

Tiempo de establecimiento (*Settling time*): Es el tiempo que tarda el sistema des de que recibe la excitación hasta ofrece una respuesta que entra en unos rangos que no superen del 2 al 5% de la respuesta en régimen permanente

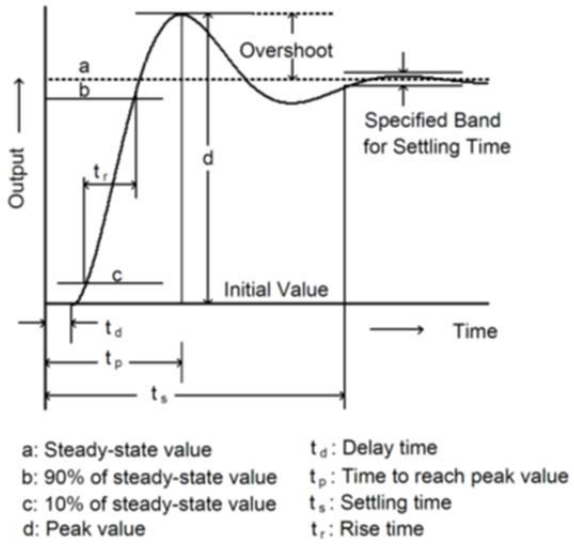

Figura 5. Respuesta dinámica típica

En la Tabla 2 podemos observar unos valores generalmente aceptados de las especificaciones de un lazo de control en respuesta a pequeñas perturbaciones.

Es necesario entender, que no es posible la optimización de todos los índices que definen el comportamiento de las respuestas de un sistema ante pequeñas perturbaciones, ya que el comportamiento de estos depende de cada sistema.

Por este mismo motivo no se puede afirmar que exista un mismo criterio de regulación aplicable para todos los sistemas. [5]

Descripción	Parámetro	Rango	Unidades
Margen de ganancia	G_m	≥ 6	dB
Marge de fase	Φ_m	≥ 40	°
Sobrepico máximo	SP_{max}	0 - 15	%
Valor de pico	M_p	0.8 - 4	dB

Tabla 2. Valores aceptados de la respuesta a pequeñas perturbaciones en un lazo de control [3]

Es necesario entender, que no es posible la optimización de todos los índices que definen el comportamiento de las respuestas de un sistema ante pequeñas perturbaciones, ya que el comportamiento de estos depende de cada sistema.

Por este mismo motivo no es posible afirmar que exista un mismo criterio de regulación aplicable para todos los sistemas. [5]

Los rangos típicos de valores de los índices que caracterizan el funcionamiento frente a pequeñas perturbaciones para sistemas de excitación son los que se muestran en la Tabla 3

Descripción	**Parámetro**	**Rango**	**Unidades**
Margen de ganancia	G_m	2 – 20	dB
Margen de fase	Φ_m	20 – 80	º
Valor de pico	M_p	0 – 12	dB
Ancho de banda	f_B	0.3 – 12	Hz
Sobrepaso máximo	SP_{max}	0 – 80	%
Tiempo de subida	t_r	0.1 – 2.5	s
Tiempo de estabilización	t_s	0.2 – 10	s

Tabla 3. Rangos típicos de parámetros en sistemas de excitación frente a pequeñas perturbaciones[3]

3.3.2. SATURACIÓN

El bloque de la saturación formaría parte de los elementos que tiene como objetivo la protección de la máquina. Dicho elemento ofrece la seguridad de que existen unos límites, tanto superiores como inferiores, para la entrada de la máquina síncrona.

Funciona de tal manera que, si este recibe como entrada un valor que está por encima del valor superior con el que se ha configurado, el bloque tendrá como salida el valor del límite estipulado, y lo mismo sucede con el límite inferior, pero en este caso, si el valor es menor que este.

Gracias a esto evitamos posibles fallos de alimentación del bloque de la máquina síncrona

3.3.3. EXCITATRIZ

La excitatriz en la encargada de suministrar la tensión y corriente continua para alimentar el rotor de un generador síncrono, y convertir a éste en un electroimán con capacidad en general para regular la intensidad del campo magnético.

La resistencia que ofrece el bobinado del rotor es fija, por lo que variando la tensión continua de alimentación se consigue variar la intensidad de la corriente que circula por el rotor, variando así la intensidad del campo magnético de acoplamiento.

El modelo del sistema de excitación IEEE tipo AC8B incluye una excitatriz AC *brushless* y queda representado mediante el diagrama de bloques de la Figura 6.

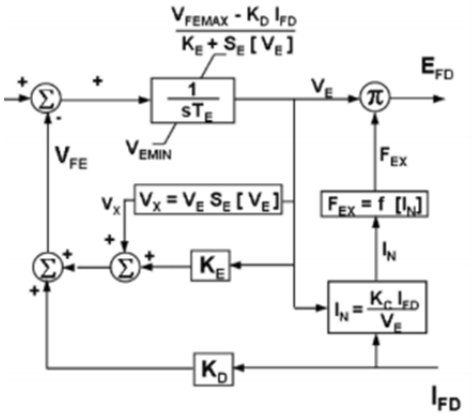

Figura 6.Modelado de la Excitatriz en el modelo AC8B [3]

Al tratarse del modelo con control digital, los parámetros K_C y K_D son 0. Entonces, la tensión de salida de la excitatriz V_E es directamente aplicada al devanado de campo de la máquina síncrona.

El resto de parámetros, excepto las ganancias del controlador, se han aplicado los recomendados por la IEEE en [3] (véase Tabla 4).

Parameter	Value	Units
T_{DR}	0.03	s
K_A	1	pu
T_A	0	s
V_{RMAX}	35	pu
V_{RMIN}	0	pu
V_{FEMAX}	6	pu
K_E	1.0	pu
T_E	1.2	s
K_C	0	pu
K_D	0	pu
$SE_{(E1)}$	0.3	-
E_1	6.9	pu
$SE_{(E2)}$	3.0	-
E_2	9.0	pu

Tabla 4. Parámetros para el caso de ejemplo

Dicho esto (K_C y K_D son 0), podemos simplificar el esquema del excitador visto anteriormente en la figura 6 en el modelo representativo del AC8B y nos quedaría tan simple como en la figura 7 (teniendo en cuenta que la rama inferior KD también desaparece).

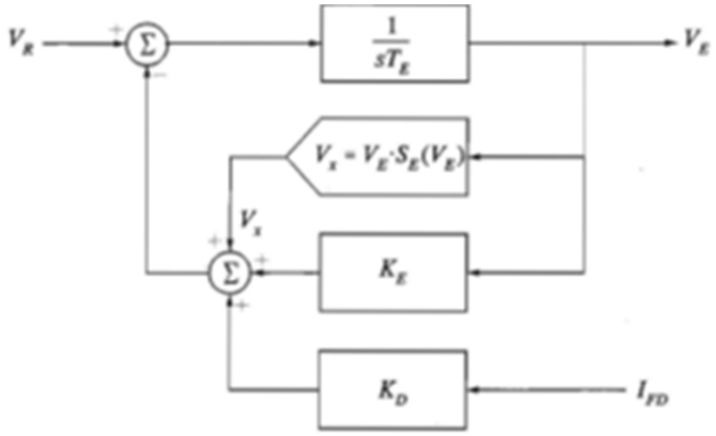

Figura 7. Diagrama de bloques de una excitatriz AC. Fuente: Kundur [5].

Dónde:

- VR: voltaje de campo del excitador controlado por el regulador.
- VE: voltaje de salida del alternador del excitador.
- IFD: corriente de campo del generador principal, esta corriente representa la carga del excitador.
- TE: constante de tiempo del excitador (datos de muestra IEEE 1.2 s).
- SE: función de saturación

4. OBTENCIÓN DE LOS PARÁMETROS DE NUESTRO MODELO

4.1. CONTROLADOR PID

Para la sintonía de nuestro controlador PID nos hemos ayudado de una herramienta que nos proporciona el propio bloque PID de *Simulink*, "*Tune…*", que encontramos al acceder a dicho bloque como observamos en la figura 8.

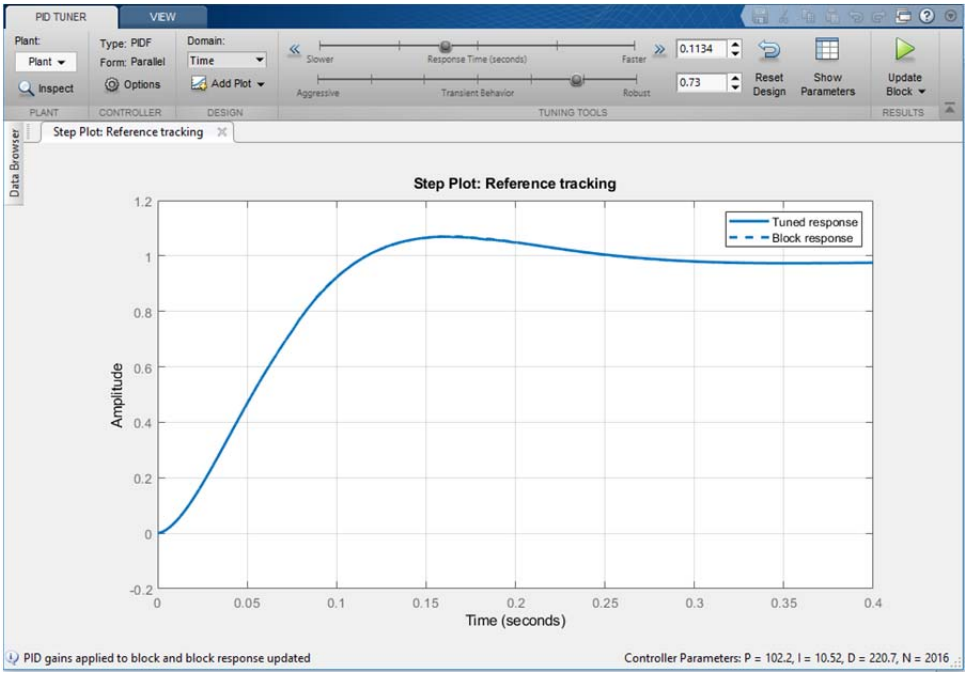

Figura 8. Pantalla principal de la herramienta Tune…

Esta herramienta realiza una simulación lineal del sistema que hemos construido, y nos permite obtener la respuesta que se adapte a nuestros requisitos. Eso lo conseguimos ajustando la acción del controlador según la rapidez que buscamos y según lo robusto que queremos que sea.

A medida que desplacemos el indicador por las barras de ajuste mencionadas, veremos como la respuesta del sistema varia, permitiéndonos así obtener el comportamiento buscado.

Otra ventaja de usar Tune es que, si estamos dudando de dos opciones diferentes de parametrización con comportamientos distintos, esta herramienta nos permite comprar las respuestas sobreponiéndolas, así como mostrando una tabla comparativa de los parámetros característicos de la respuesta.

En la Figura 9 podemos ver la comparativa de dos respuestas distintas, representada con una línea discontinua la configuración anterior, guardada en el bloque PID, y con línea continua la actual, ajustada mediante los deslizadores.

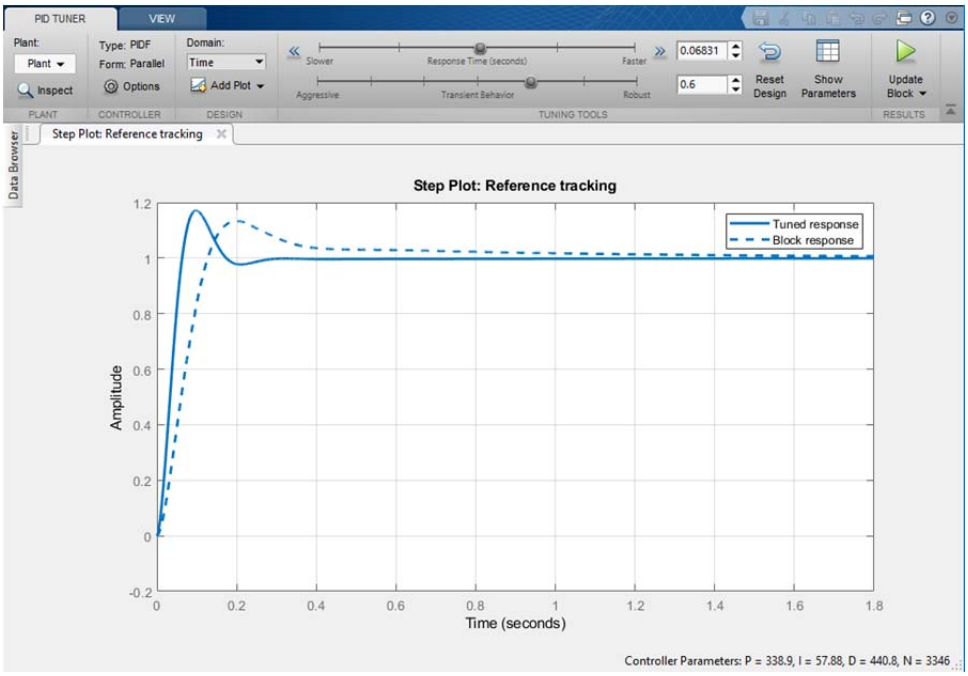

Figura 9. Representación de la comparativa entre diferentes configuraciones de parámetros en Tune…

Y mediante la opción "*Show parameters*", accedemos a la tabla comparativa de los parámetros en formato numérico de las dos configuraciones, mostrada en la figura 10, facilitando así la elección de implementar el que mejor se ajuste a nuestras necesidades.

Controller Parameters

	Tuned	Block
P	338.8672	344.3947
I	57.8799	142.5299
D	440.8108	173.2143
N	3345.9387	1676.9417

Performance and Robustness

	Tuned	Block
Rise time	0.0441 seconds	0.0889 seconds
Settling time	0.229 seconds	0.881 seconds
Overshoot	17.2 %	13.3 %
Peak	1.17	1.13
Gain margin	39 dB @ 3.08e+06 rad/s	53.1 dB @ 3.08e+06 ra...
Phase margin	52.4 deg @ 29.3 rad/s	59.9 deg @ 14.7 rad/s
Closed-loop stability	Stable	Stable

Figura 10.Ccomparativa de parámetros entre diferentes configuraciones en Tune…

Mediante esta herramienta conseguimos ajustar el controlador PID obteniendo los parámetros de las ganancias pertinentes, aunque si una vez logrado queremos ajustar por separado una de las diferentes ganancias, ya sea la proporciona, la integral o la derivativa, podemos hacerlo dando el valor concreto deseado a cada una en el menú principal del controlador. Sabiendo que tipo de comportamiento controla cada ganancia, será fácil obtener una respuesta fiel a la que aspiramos.

Los valores que nosotros hemos obtenido son los que se muestran en la figura 11.

Controller Parameters

	...	Block
P	...	107
I	...	64.7398
D	...	42.2741
N	...	200

Performance and Robustness

	...	Block
Rise time	...	0.262 seconds
Settling time	...	1.91 seconds
Overshoot	...	19.9 %
Peak	...	1.2
Gain margin	...	42.8 dB @ 128 rad/s
Phase margin	...	60.5 deg @ 4.51 rad/s
Closed-loop stability	...	Stable

Figura 11. Valores usados en el controlador PID de nuestro modelo

Como podemos observar hay valores que escapan de los rangos habituales, como pueden ser el sobre pico o el margen de ganancia. Esto es debido a que la herramienta no tenga en cuenta el código que hay escrito en el bloque de nuestra máquina síncrona, y que simplemente linealice el lazo de control exterior. Otro factor que puede influir es que en el momento de arrancar la máquina síncrona, existe un periodo en que el comportamiento de esta pasa por un estado transitorio inestable.

Como en este proyecto se pretende controlar el comportamiento de la máquina síncrona cuando se encuentra en estado de equilibrio y se aplica una variación en la excitación, obviamos ese periodo.

Por ello, aunque algunos valores salgan de los varemos usuales, hemos mantenido los valores de las ganancias del controlador obtenidas.

4.2. SATURACIÓN

En este caso usamos lo valores que se encuentran en la tabla de la figura... que son 0V para el límite inferior, y 35V para el límite superior.

4.3. EXCITATRIZ

Como se ha comentado anteriormente, al tratarse de un AVR tipo AC8B con control digital los parámetros K_C y K_D son 0, por tanto, se obtiene que V_E y E_{FD} son iguales. Además, la regulación de carga debido a la reacción de armadura es considerada utilizando la curva de saturación en circuito abierto para definir la función de saturación S_E.

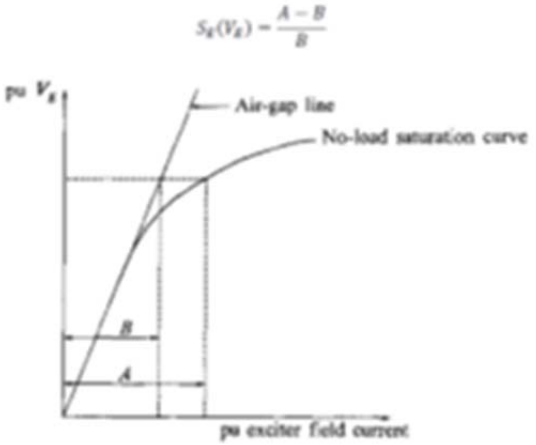

Figura 12. Curva de saturación de la excitatriz. Fuente: Kundur [6].

Observando la figura 12 podemos calcular la función de saturación SE para un valor especifico VE.

La función de saturación en por unidad viene determinada por la siguiente ecuación que encontramos en *Kundur* [6]:

$$S_E(V_E) = \frac{A-B}{B} \tag{27}$$

Para representar la función de saturación utilizaremos la siguiente función exponencial:

$$V_X = E_X S_E(E_X) = E_X A_{EX} e^{B_{EX} E_X} \tag{28}$$

En este punto tenemos:

$$SE_{(E1)} = A_{EX} e^{B_{EX} E_1} \tag{29}$$

$$SE_{(E2)} = A_{EX} e^{B_{EX} E_2} \tag{30}$$

Dividiendo la ecuación (29) por la (30) se obtiene:

$$\frac{SE_{(E1)}}{SE_{(E2)}} = e^{B_{EX}(E_1 - E_2)} \tag{31}$$

Si usamos logaritmos neperianos en ambos lados de la ecuación (31) y despejando, obtenemos:

$$B_{EX} = \frac{\ln SE_{(E1)} - \ln SE_{(E2)}}{E_1 - E_2} \quad (32)$$

A partir de las ecuaciones (29) y (30) llegamos a:

$$A_{EX} = \frac{SE_{(E1)}}{e^{B_{EX}E_1}} = \frac{SE_{(E2)}}{e^{B_{EX}E_2}} \quad (33)$$

Si se sustituyen los parámetros de la Tabla 4 en este punto, se obtienen los valores de las constantes de la función exponencial que describe la característica de saturación de la excitatriz.

Los resultados que hemos obtenido son $A_{EX} = 7.536 \cdot 10^{-4}$ y $B_{EX} = 0.921$.

4.4. TRANSDUCTOR

La función de transferencia del transductor *H(s)* según *Sheldrake* [7], viene dada por un sistema de primer orden con T_s igual a 0.02 s como constante de tiempo

$$H(s) = \frac{1}{1 + sT_s} \quad (34)$$

5. IMPLEMENTACIÓN DE CIRCUITO

Una vez parametrizados nuestros componentes procedemos a la construcción de nuestro sistema mediante los bloques que nos ofrece el propio *Simscape*.

Para este paso hay algunas cosas que necesitamos tener en cuenta.

Primeramente, tenemos que multiplicar la señal de excitación, procedente de la excitatriz por el valor de la tensión nominal de excitación, que en nuestro caso es de 6.723.

Y segundo, que estamos construyendo un circuito que irá conectado a un componente personalizado de *Simscape*. Este tipo de componentes únicamente trabajan con variables físicas, lo que implica que tendremos que implementar un conversor para que las señales con las que trabaja nuestro sistema de control puedan ser usadas por el componente de la máquina síncrona.

Esta conversión la llevaremos a cabo mediante diferentes bloques.

Primero de todo necesitamos convertir nuestra señal Simulink en señal física y viceversa, y para ello necesitamos estos bloques *Simulink-PS converter* y *PS-Simukink converter* mostrados en las figuras 13 y 14respectivamente.

Figura 13. Bloque Simulink-PS converter

Figura 14. Bloque PS-Simukink converter

Gracias a los bloques anteriores ya tenemos nuestras señales físicas, pero ahora necesitamos convertir estas señales físicas en tensión, para que nuestra máquina síncrona pueda trabajar con ella. Para ello usamos:

- *Controlled Voltage*, mostrado en la figura 15. Convierte una entrada física en una salida de tensión
- *Voltage Sensor*, mostrado en la figura 16. Nos da el valor de la tensión de entrada como variable física.

Figura 15. Bloque Controlled Voltage

Figura 16. Bloque Voltage Sensor

Una vez solucionada esta situación podemos proceder a la implementación del sistema de control, pero nos vamos a dar cuenta, que a la hora de cerrar el circuito se nos presenta otro problema.

Tenemos como salida del bloque de la máquina síncrona tres salidas diferentes de tensión, y lo que nos interesa para cerrar el circuito, es conseguir un único valor por unidad, para poder compararlo con el valor del *Step* de excitación, y que de esta comparativa salga el error en cada instante de tiempo, que alimente el controlador PID.

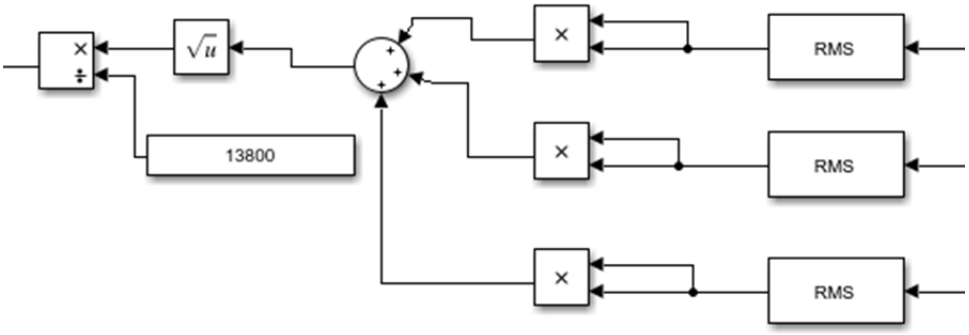

Figura 17. Circuito de conversión de la salida de nuestra máquina síncrona

Esto lo conseguimos con el circuito que se muestra en la figura 17 (Inputs en la derecha, outputs en la izquierda).

Como encontramos en el libro *Signal Processing of power quality disturbances* [8], cuando las tres formas de onda muestreadas en un sistema trifásico están disponibles como es nuestro caso, se puede realizar la siguiente aproximación:

$$V_v = \sqrt{\frac{1}{3}(V_a^2 + V_b^2 + V_c^2)}$$

Dividiendo el resultado obtenido por la tensión nominal de nuestra máquina síncrona, 13800, obtendremos el valor por unidad que buscamos.

El sistema quedaría resuelto en la figura 18

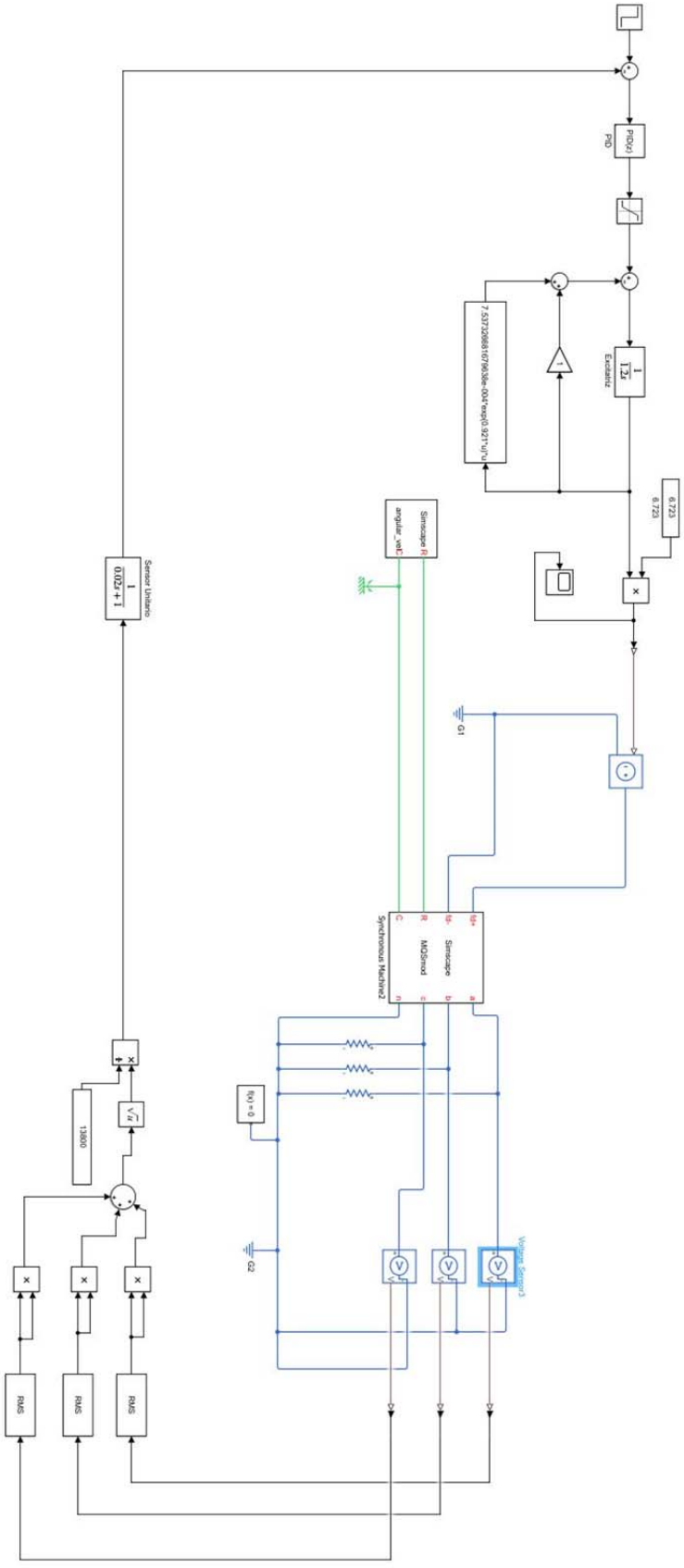

Figura 18. Modelo de excitación AC8B conectado externamente a la máquina síncrona

6. RESULTADOS DEL SISTEMA

Ahora que tenemos el sistema que sigue las directrices del modelo AC8B con los parámetros correspondientes, procedemos a ver si los resultados obtenidos encajan con los esperados, y esto lo haremos observando el comportamiento de las señales en diferentes puntos.

A la hora de interpretar las gráficas, hay que tener en cuenta que el objetivo de del proyecto es la implementación de un sistema de control para solventar un cambio en el valor de la excitación de la máquina.

Por eso es recordar en este punto, antes de mostrar ningún gráfico, que existe un periodo al inicio, cuando la máquina se pone en arranca, en el que el sistema no se comporta de manera estable, y es pasado este lapso de tiempo, cuando la respuesta de la máquina entra en régimen permanente, y podemos poner en práctica cómo se comporta el sistema de control con el cambio del valor de excitación.

Los puntos relevantes en cuanto a los resultados serían:

- El error.
- La salida del controlador PID.
- La salida de la saturación.
- La salida de la excitatriz.
- La salida del transductor.
- Las salidas de la máquina síncrona.

6.1. EL ERROR

En la figura 19 podemos ver en qué momentos la salida de la máquina síncrona no es la deseada, y por lo tanto, significa que la acción de control está trabajando para solucionarlo.

El error, al ser el resultado de la comparativa entre el *Step* que usamos a modo de excitación, y la salida del transductor procedente de la realimentación, empieza con un valor de 1. Esto se debe a que en el instante inicial no hay ninguna salida previa de la máquina con la que comparar el *Step* de excitación.

En el momento en que este error es cero, significa que tenemos el resultado deseado.

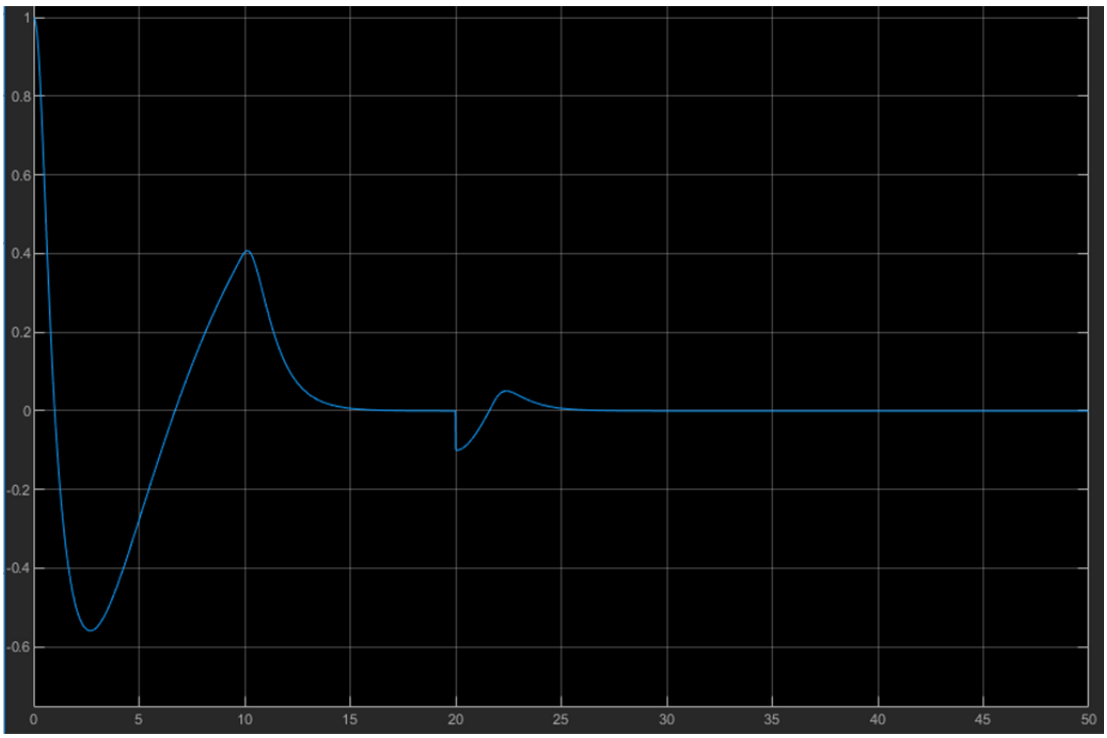

Figura 19. Representación del error del sistema respecto al tiempo

6.2. PID

En la Figura 20 encontramos la salida de nuestro controlador PID. Podemos observar como en el instante de tiempo T=20s, hay una caída muy brusca del valor de salida. Eso se debe a la aproximación que hace el sistema, pero no nos perjudica el experimento, ya que contamos con un bloque de saturación, que impide que valores no deseados, que se encuentran fuera del rango que nosotros especificamos pasen a la excitatriz.

Como se puede observar, el tiempo de asentamiento del sistema es muy pequeño, ya que a partir de que el valor de la respuesta se encuentra en un rango de entre 2 o 5% del valor de la salida en régimen permanente, ya se puede dar el valor por bueno según los cánones de control.

Por este mismo motivo, podemos afirmar que es una buena respuesta.

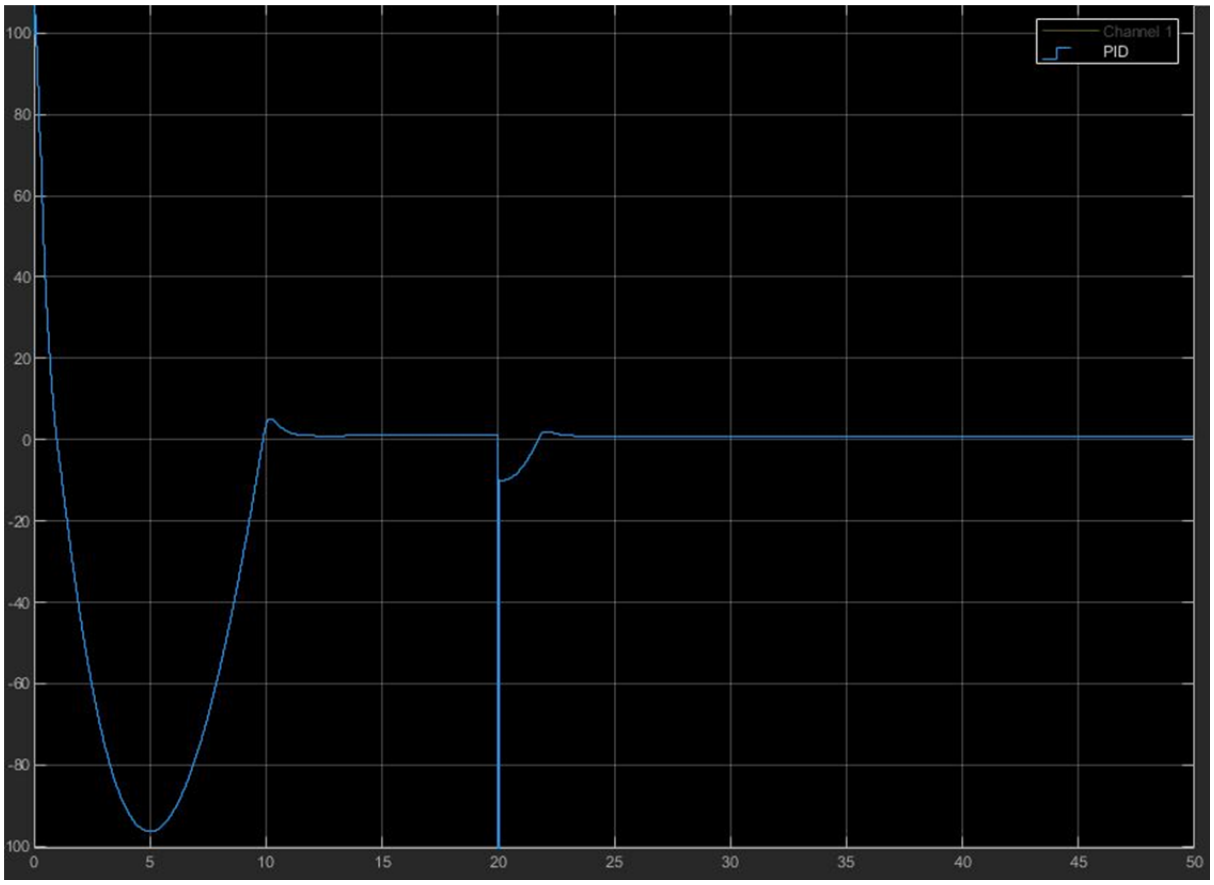

Figura 20. Representación de la salida del bloque PID respecto al tiempo

6.3. SATURACIÓN

En la figura 21 encontramos la salida del bloque de saturación, que como se ha comentado antes actúa para limitar los valores de salida del controlador PID para que no dañen el sistema.

Como la entrada del bloque es la salida del controlador PID, y los parámetros de limitación que hemos configurado son 35 por el límite superior y 0 por el límite inferior, la figura nos muestra el mismo resultado que el que tenemos en la figura 21, pero con los valores limitados.

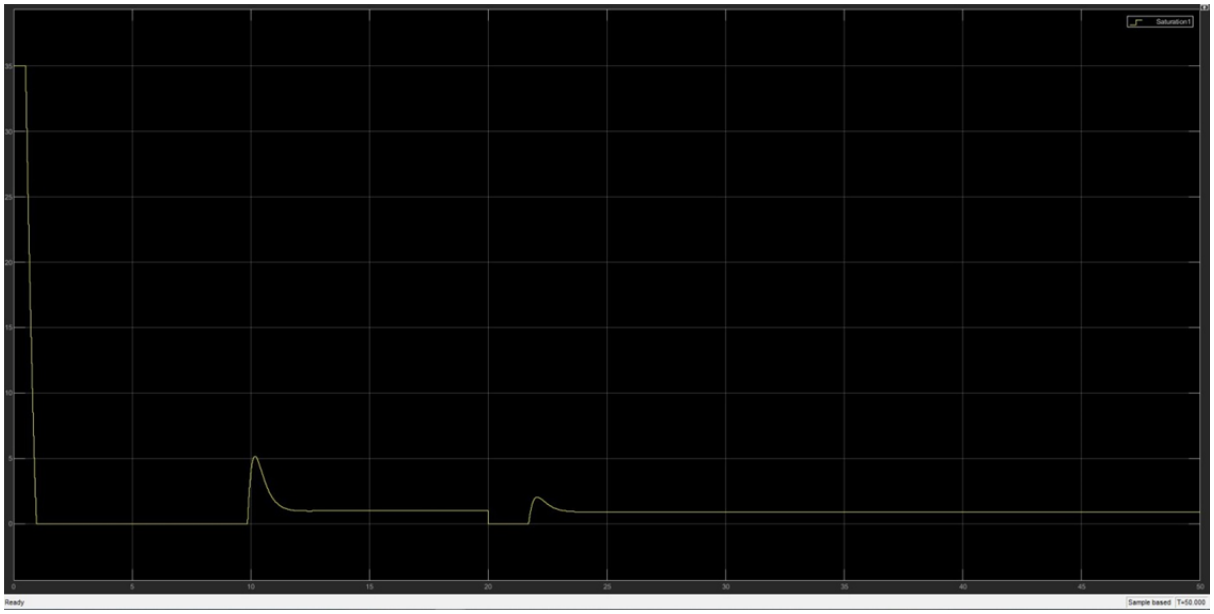

Figura 21. Representación de la salida del bloque de saturación respecto al tiempo

6.4. EXCITATRIZ

Como sabemos, la excitatriz en la encargada de suministrar la tensión y corriente continua para alimentar el rotor de un generador síncrono. En la figura 22, podemos ver como la salida de nuestra excitatriz, que sigue el mismo patrón que la figura 21, pero con los valores ya adaptados para alimentar el rotor de nuestra máquina síncrona, y también que tarda aproximadamente 5 segundo en estabilizarse después de la variación en la excitación de la máquina.

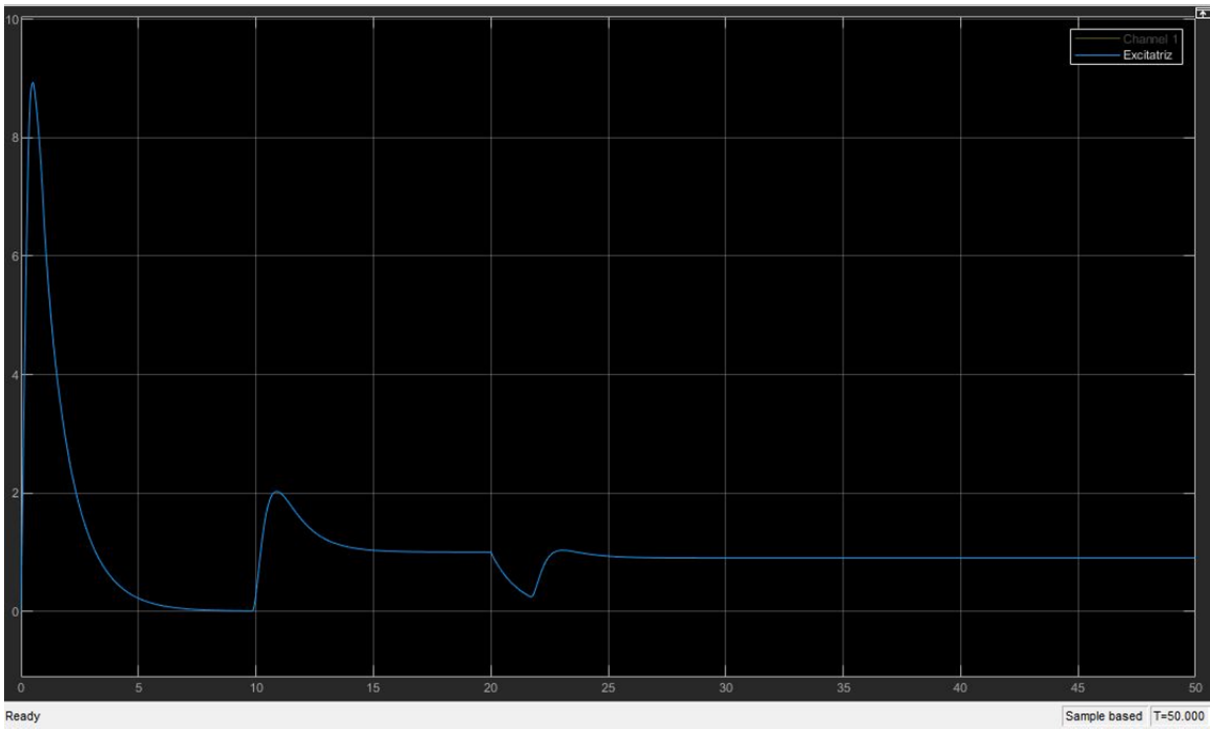

Figura 22. Representación de la salida del bloque de saturación respecto al tiempo

6.5. TRANSDUCTOR

En la figura 23 se muestra la comparativa de la salida del transductor con el *Step* de excitación.

Esta señal se utiliza para calcular el error del sistema, que va a alimentar nuestro controlador PID.

Cuando vemos que las dos señales mostradas coinciden, significa que el error es nulo, y que por consecuencia tenemos la respuesta del sistema que buscamos.

Sobre el transductor, es necesario comentar que con los parámetros usados, únicamente aporta dinamismo al sistema, retrasando la señal, ya que la ganancia de este es 1, y por lo tanto no modifica su valor.

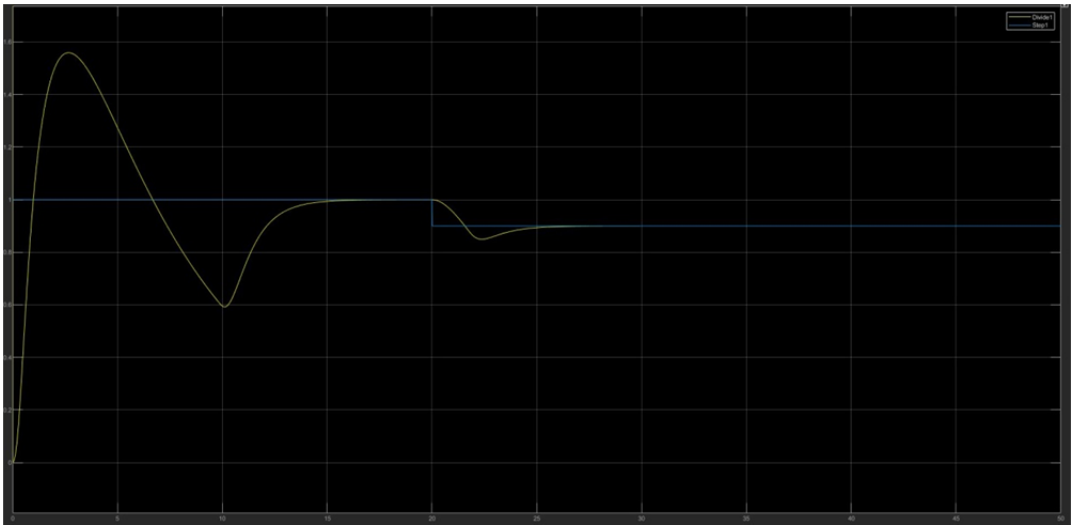

Figura 23. Representación comparativa de la salida del transductor y del Step de excitación.

6.6. SALIDAS MÁQUINA SÍNCRONA

Para terminar con este apartado, en la figura 24 se muestran las salidas directas de nuestra máquina síncrona, sobrepuestas, aunque como son de la misma amplitud, no se aprecian.

Pero si se puede apreciar perfectamente cómo una vez estabilizada la salida después de la puesta en marcha de la máquina síncrona, en el instante T=20s, hay una fluctuación de los valores, debido al cambio en la excitación del sistema.

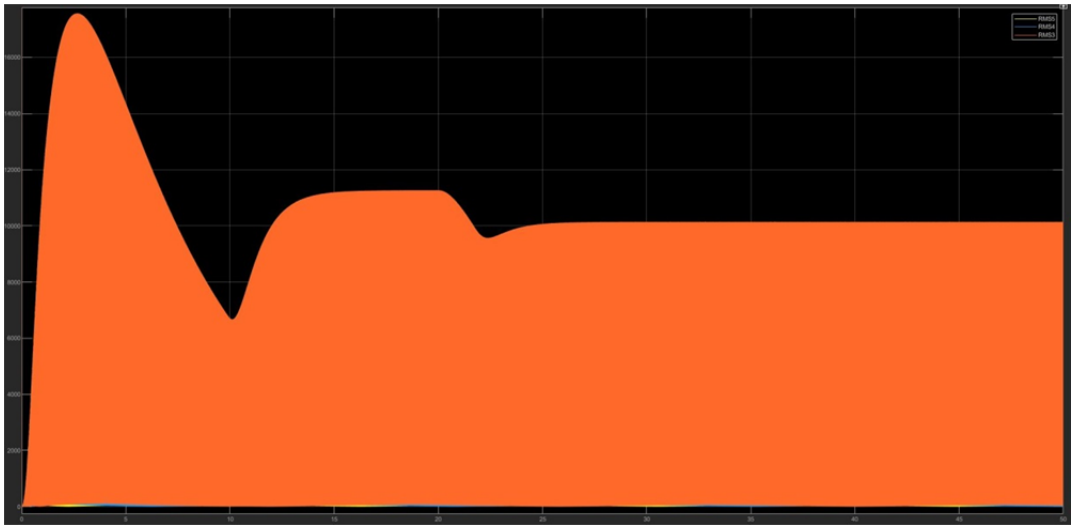

Figura 24. Representación de las salidas de la máquina síncrona

7. TRADUCCIÓN A CÓDIGO

Llegados a este punto, es hora de implementar el objetivo final del proyecto, introducir el sistema de control construido en los apartados anteriores dentro del código interno del bloque de la máquina síncrona.

Para ello se ha procedido de manera escalonada, componente por componente, llegando al final a obtener un sistema como el mostrado en la figura... en el que únicamente encontramos por la parte de los inputs, un bloque *Step*, que usamos a modo de excitación, y el bloque "*angular_vel*". Y como outputs las tres salidas de tensión de la propia máquina síncrona.

Para facilitar la comprensión de este apartado, se ha seguido un mismo orden a la hora de explicar el proceso:

- Mostrar la forma en el sistema de bloques
- Ecuaciones y cálculos necesarios para la traducción de bloques a código
- Comparativa de resultados entre la salida de los diferentes sistemas de bloques
- Escritura del código

En cuanto a la escritura del código, también se sigue un orden marcado:

- Declaración de variables y explicación si es necesario
- Implementación del código

También para facilitar a la comprensión del código, y ya que ha sido necesario añadir muchas variables. Se ha optado por Nombrar todas las variables de entrada a un conjunto como "U" seguido del nombre del conjunto al que entra, y las salidas como "Y", también seguido del nombre del conjunto del que salen. Por ejemplo, en la excitatriz la entrada será "*UExcitatriz*" y la salida "*YExcitatriz*"

Esto queda ya explicado en el apartado de Variables, pero vemos importante recodarlo en este punto para que no haya ningún tipo de confusión.

7.1. CONTROLADOR PID

En nuestro circuito, el controlador PID viene representado por el bloque *PID controler*, mostrado en la figura 25

Figura 25. Bloque PID controler

Para la traducción de este componente hemos decidido desglosar las diferentes ganancias KP, KI y KD, y hacer sus aproximaciones por separado para finalmente, unir el resultado para obtener la ecuación en diferencias.

La acción proporcional cumple la siguiente relación estática.

$$K_p \cdot e(k) \tag{35}$$

La acción integral se puede considerar como la suma del área de la aproximación del rectángulo de la función de error, que podemos escribir de la siguiente manera

$$K_i \cdot T_s \cdot \sum_{p=0}^{k} e(p) \tag{36}$$

Y la acción derivativa, finalmente, puede ser aproximada por:

$$K_d \cdot \left(\frac{e(k) - e(k-1)}{T_s} \right) \tag{37}$$

Por lo que podemos describir el controlador PID como la suma de estas ecuaciones, y nos encontramos con la ecuación..

$$mv(k) = K_p \cdot e(k) + K_i \cdot T_s \cdot \sum_{p=0}^{k} e(p) + K_d \cdot \frac{e(k) - e(k-1)}{T_s} \tag{38}$$

En la que *e*, es el error, o entrada al controlador, y *mv*, es la salida de este.

Como ya hemos conseguido la ecuación en diferencias, podemos traducirla a bloques para poder comparar y ver si el resultado coincide. Su construcción se puede observar en la figura 26.

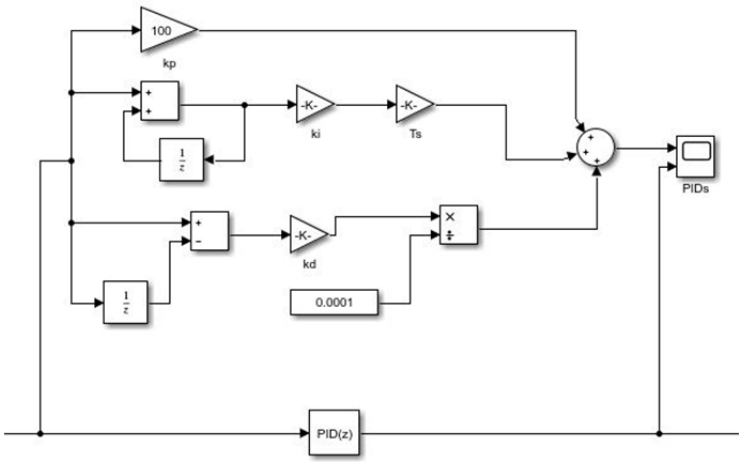

Figura 26. Bloque PID montado en paralelo con su propuesta equivalente

Y podemos observar los resultados del *Scope* en la gráfica de la figura 27, pudiendo dar el nuevo sistema por válido.

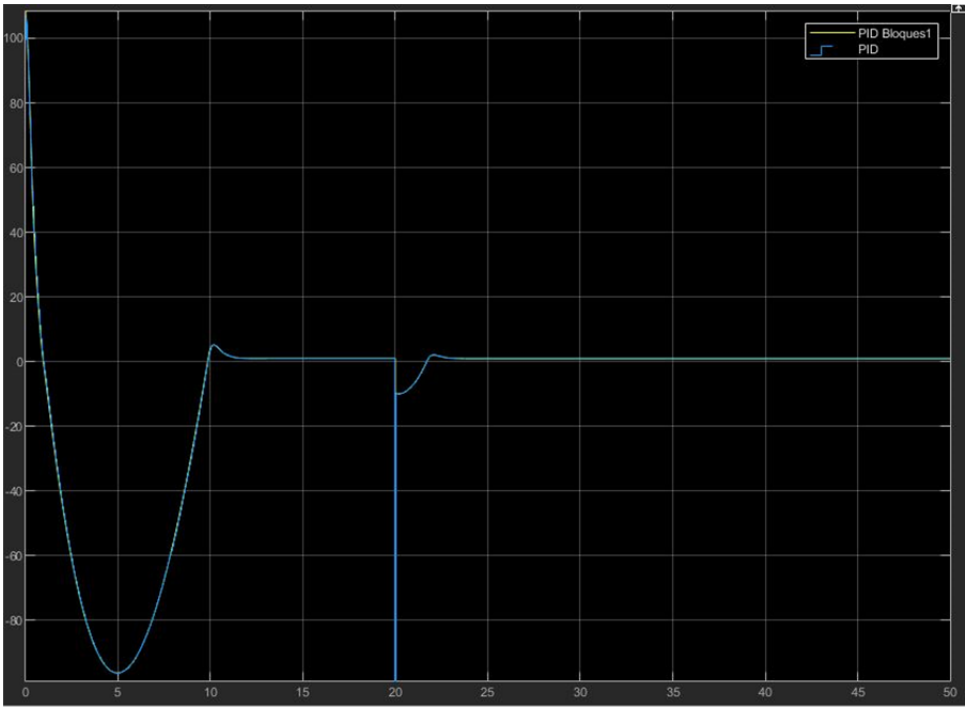

Figura 27. Comparativa de las salidas del bloque PID y de su circuito equivalente

Viendo que los resultados son positivos y que las gráficas coinciden, podemos proceder a pasar el sistema a código. Para ello se han declarado como parámetros los valores de las diferentes ganancias, junto con el valor de tau (periodo de muestreo), para que el usuario pueda modificarlos des del menú propio del bloque de la máquina síncrona.

```
tau = {0.0001,'s'};
Proporcional    = {107.366, '1'};
Integrador    = {64.7398, '1'};
Deribatiu    = {42.2741, '1'};
```

Los siguientes valores en cambio se han declarado en el campo de Variables, y no pueden ser modificados des del menú del bloque de nuestra máquina síncrona.

```
YPID =  { 0, 'V' };
ValorTau =  { 0, '1' } ;
SumatoriError = { 0, 'V' } ;
SumatoriErrorAnterior = { 0, 'V' } ;
Error = { 0, 'V' } ;
ErrorAnterior = { 0, 'V' }
```

A continuación, observamos el código que describe el comportamiento del controlador

```
ValorTau == value(tau,'s');
SumatoriErrorAnterior == delay(SumatoriError,tau);
ErrorAnterior == delay(Error,tau);
SumatoriError == SumatoriErrorAnterior + Error;

YPID == Proporcional * Error + Integrador * ValorTau * SumatoriError+ (Deribatiu*(Error-ErrorAnterior))/ValorTau;
```

En la primera línea del código observamos como guardamos en *ValorTau*, el valor numérico del parámetro tau, sin las unidades correspondientes. Esto es necesario para que el prográmanos deje ejecutar la operación, ya que sino mostraría un error de conflicto de unidades.

Las siguientes tres líneas son la traducción de un sumatorio, mediante las que vamos acumulando el valor del error en todos los instantes, sumando al valor de error acumulado, el valor del instante actual.

Y finalmente tenemos la ecuación que describe el comportamiento del controlador.

7.2. SATURACIÓN

En el sistema de bloques, la saturación es regulada por el bloque mostrado en la figura 28.

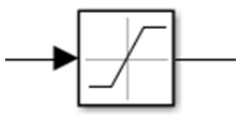

Figura 28. Bloque saturación

En este caso se utilizan dos valores del apartado de parámetros mostrados en la figura 29, por si el usuario quiere cambiar estos valores des del menú del bloque.

```
SaturacioLow = {0, 'V'};
SaturacioHigh = {35, 'V'};
```

Que nos permitirán ajustar el límite tanto inferior como superior de la saturación respectivamente.

También se hace uso de una nueva variable

```
YSaturacion= { 0, 'V' };
```

Que será la responsable de almacenar el valor de la salida de la saturación en cada momento.

La manera más fácil de llevas a cabo la implementación en código de este bloque es mediante la función IF.

```
if YPID < SaturacioLow
    YSaturacion== { 0, 'V' };
elseif YPID > SaturacioHigh
    YSaturacion== { 35, 'V' };
else
    YSaturacion== YPID;
end
```

Como se puede apreciar en la el código, comparamos el valor YPID correspondiente a la salida del controlador PID, con los límites de saturación, y en caso de que excedan, son modificados.

7.3. EXCITATRIZ

Nuestra excitatriz presenta la forma que se muestra en la figura 29.

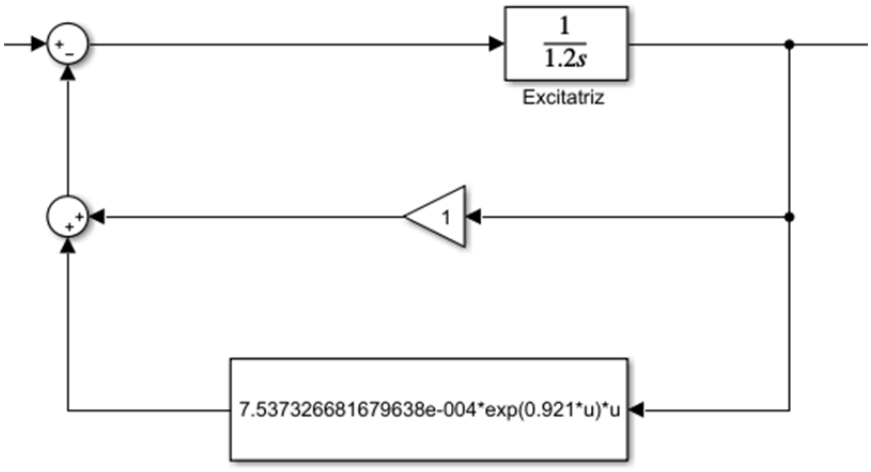

Figura 29. Excitatriz del modelo AC8B

Lo que nos impide traducir a código este apartado directamente es la función de transferencia "Excitatriz" con valor 1/1.2s, ya que el resto son meras ecuaciones.

Para solventarlo, Partimos de la ecuación

$$G(s) = \frac{1}{1.2s} \tag{39}$$

Vamos a substituir la s por su equivalente en el domino Z, que es el siguiente

$$s = \frac{1 - Z^{-1}}{TZ^{-1}} \tag{40}$$

Para facilitarnos el proceso multiplicamos por Z, y así simplificamos la ecuación, obteniendo

$$s = \frac{Z - 1}{T} \tag{41}$$

Seguidamente, procedemos a substituir la s en la ecuación G(s)

$$G(z) = \frac{1}{1.2\dfrac{(Z-1)}{T}} = \frac{T}{1.2(Z-1)} \tag{42}$$

Ahora nos interesa recuperar las Z^{-1} que hemos simplificado antes. Esto lo logramos multiplicando por Z^{-1}tanto en el numerador como en el denominador de la ecuacion.

También igualamos el resultado a $\frac{Y(z)}{U(z)}$ para obtener así nuestra ecuación en diferencias.

$$\mathrm{G(z)} = \frac{TZ^{-1}}{1.2(1-Z^{-1})} = \frac{Y(z)}{U(z)} \tag{43}$$

En este punto, como sabemos que la planta $\mathrm{G(z)}$ es igual a $\frac{Y(z)}{U(z)}$, donde $Y(z)$ es la salida y $U(z)$ es la entrada, podemos definir

$$\frac{Y(z)}{U(z)} = \frac{TZ^{-1}}{1.2(1-Z^{-1})} \tag{44}$$

Pasamos los denominadores al otro lado de la igualdad respectivamente obteniendo

$$TZ^{-1}U(z) = 1.2Y(z)(1-Z^{-1}) \tag{45}$$

Ahora, sabiendo que Z^{-1} es el equivalente a $[k-1]$, que hace referencia a la muestra anterior de la variable a la que acompaña, pasamos del dominio Z al domino k

$$TU_{(k-1)} = 1.2Y[k] - 1.2Y_{(k-1)} \tag{46}$$

Si aislamos la ecuación para quedarnos con la definición de $Y[k]$obtendremos que nuestro resultado es

$$Y[k] = \frac{T}{1.2}U_{(k-1)} + Y_{(k-1)} \tag{47}$$

Para saber si el resultado obtenido en la ecuación 46 es correcto, construimos la ecuación resultante con bloques, paralelamente al bloque Excitatriz, y comparamos los resultados mediante un *Scope*.

La transcripción de la ecuación a bloques y puesta en paralelo con el bloque original se puede apreciar en la Figura 30.

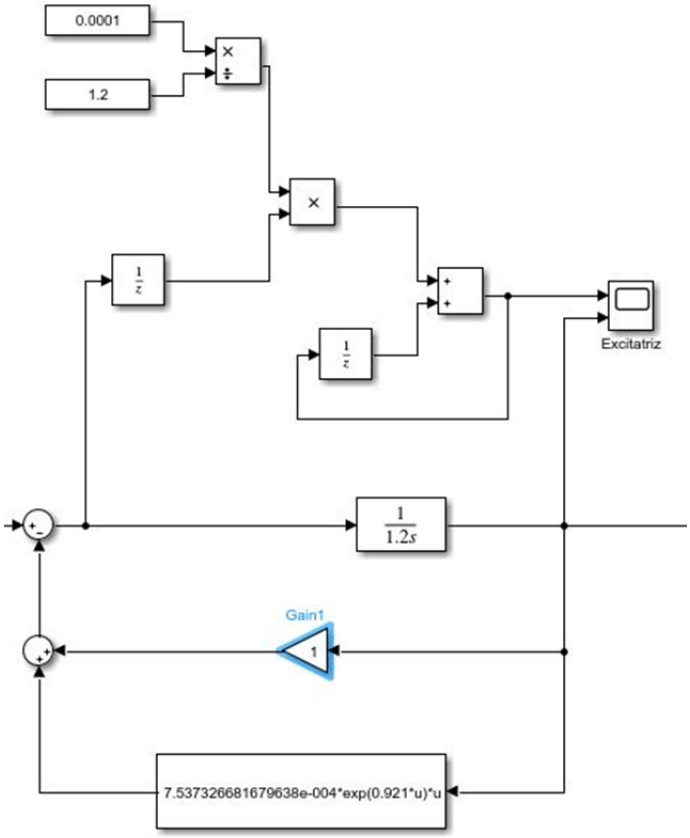

Figura 30. Excitatriz original montada en paralelo con su propuesta equivalente

Y si observamos el *Scope*, podemos apreciar que es las respuestas de ambas opciones coinciden perfectamente, como se muestra en la figura 31

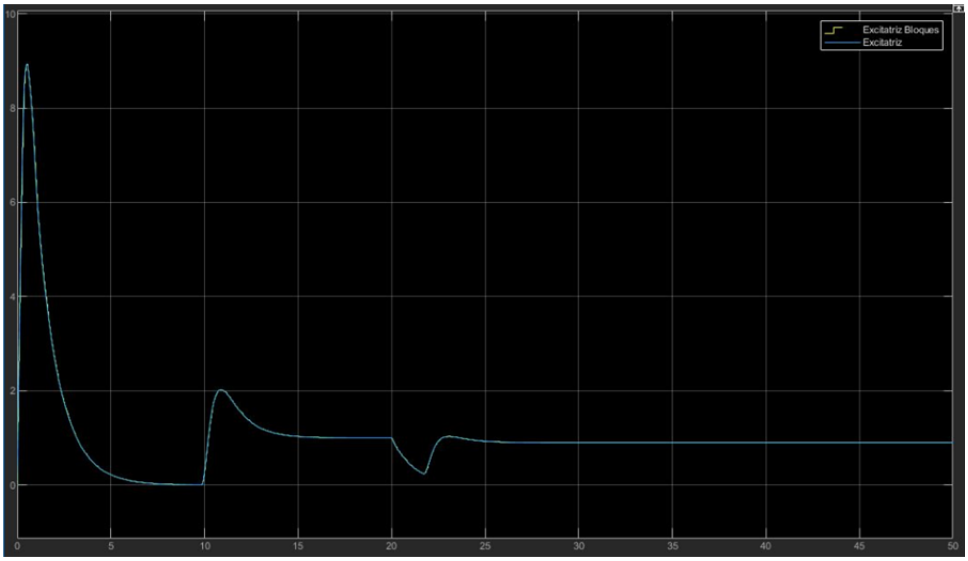

Figura 31. Comparativa de las salidas de la excitatriz y de su circuito equivalente

Al comprobar que las respuestas de ambos modelos coinciden, podemos, ahora sí, empezar la traducción a código.

Primero de todo es necesario declarar las variables que vamos a necesitar para la implementación, que son las siguientes:

```
Funcio = { 0, 'V' };
UExcitatriz = { 0, 'V' };
UExcitatrizAnterior = { 0, 'V' };
YExcitatriz = { 0, 'V' };
YExcitatrizAnterior = { 0, 'V' };
Yy = { 0, '1' };
```

La variable función es donde se guarda el resultado de la ecuación:

7.537326681679638e-004*exp(0.921*u)*u

La variable *Yy* como podemos observar, se diferencia de las demás a causa de que en el campo de unidades encontramos un "1" en lugar de "V". Eso es a causa de a la hora de calcular la función mencionada anteriormente, se hace uso de la función "exp()", y esta no acepta argumentos que tengan unidades físicas, así que se ha creado la variable *Yy* para guardar el valor de *YExcitatrizAnterior* pero sin unidades.

El código queda de la siguiente manera:

```
YExcitatrizAnterior == delay(YExcitatriz,tau);
UExcitatrizAnterior == delay(UExcitatriz,tau);
Yy == value(YExcitatrizAnterior,'V')
UExcitatriz == YSaturacion-(Funcio+YExcitatrizAnterior);
Funcio == 7.537326681679638e-004*exp(0.921*Yy)*YExcitatrizAnterior;
YExcitatriz == (0.0001/1.2)*UExcitatrizAnterior+YExcitatrizAnterior;
```

Las dos primeras líneas de código conseguimos las muestras anteriores de la salida y de la entrada de la propia excitatriz respectivamente, gracias a la función *delay*(argumento1, argumento2).

Seguidamente encontramos el caso mencionado anteriormente de quitar la unidad la variable *YExcitatrizAnterior* y esto lo conseguimos gracias a la función *value*(argumento, 'unidad').

En este punto tenemos todos los parámetros necesarios, así que únicamente traducimos el diagrama.

La variable *UExcitatriz* guarda la diferencia que hay entre la salida de la saturación y la realimentación de la propia Excitatriz, seguido del cálculo de Función, para llegar a *YExcitatriz*, que es la transcripción literal de la ecuación en diferencias a la que hemos llegado en la parte de los cálculos.

7.4. TENSIÓN NOMINAL DE EXCITACIÓN.

Como se ha comentado anteriormente, la señal saliente de la excitatriz, es necesario multiplicarla por el valor de la tensión nominal de excitación, para que así pueda alimentar debidamente el componente de nuestra máquina síncrona.

En el sistema de bloques tiene la forma que se muestra en la figura 32.

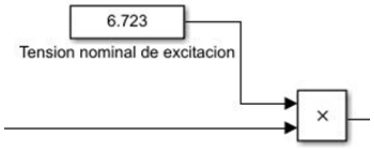

Figura 32. Representación en bloques del producto por tensión nominal de excitación

La trascripción a código es muy simple, y queda resuelta de la siguiente manera

```
VF == YExcitatriz*6.723;
```

7.5. CONVERSIÓN DE SALIDA TRIFÁSICA A POR UNIDAD.

Como se ha comentado anteriormente en este mismo trabajo, necesitamos un modo que nos permita, mediante las salidas de nuestra máquina síncrona, poder realimentar el circuito para poder obtener el error en cada instante, y dar esa información al controlador para poder actuar en consecuencia.

La representación mediante bloques es la ya mostrada anteriormente en la figura 18 y cómo podemos observar, se compone en su totalidad de ecuaciones matemáticas.

Gracias a ello la transcripción a código es muy sencilla como apreciamos a continuación:

```
vsarms = sqrt(sum((abs(vsa))^2)/length(vsa))
vsbrms = sqrt(sum((abs(vsb))^2)/length(vsb))
vscrms = sqrt(sum((abs(vsc))^2)/length(vsc))
USensor = sqrt(vsarms^2+vsbrms^2+vscrms^2)/13800/sqrt(3)*sqrt(3)
```

La única diferencia con los otros casos es que esta parte del código esta implementada en la sección "*Intermediates*", ya que fuera de esta sección mostraba errores en cuanto a las unidades de la operación.

Como variable encontramos únicamente *Usensor*, que será la que hace referencia a la entrada del transductor, pero como se encuentra definida en el apartado "*Intermediates*", no es necesario definirla junto con todas las variables.

7.6. TRANSDUCTOR

Finalmente encontramos el transductor, que está representado con un bloque *Transfer Función* tal y como se muestra en la figura 33.

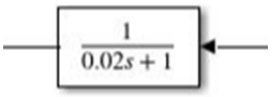

Figura 33. Bloque que representa nuestro transductor

Para poder llegar a escribir el código, vamos a necesitar discretizar la función y obtener su ecuación en diferencias.

Para ello, partimos de la ecuación

$$G(s) = \frac{1}{0.02s + 1} \tag{48}$$

De aquí podemos extraer que $\tau = 0.02$ y K=1

Para ello necesitamos la transformada en Z de la función, que para nuestro caso queda de la siguiente manera después de multiplicar numerador y denominador por Z^{-1}

$$G(z) = \frac{KbZ^{-1}}{1 - aZ^{-1}} \tag{49}$$

Donde $a = e^{-sT}$, siendo al mismo tiempo

$$s = \frac{-1}{\tau} = \frac{-1}{0.02} = -50$$

T, el periodo de muestreo de nuestro sistema, 0.0001s.

Por lo que

$$a = e^{-sT} = e^{0.0001*(-50)} = 0.995 \tag{50}$$

Y por consiguiente

$$b = 1 - a = 0.005 \tag{51}$$

Con sus valores calculados nuestra ecuación tiene esta forma

$$G(z) = \frac{1 * (0.005)Z^{-1}}{1 - 0.995Z^{-1}} \quad (52)$$

En este punto, como sabemos que la planta G(z) es igual a $\frac{Y(z)}{U(z)}$, donde $Y(z)$ es la salida y $U(z)$ es la entrada, podemos definir

$$\frac{Y(z)}{U(z)} = \frac{1 * (0.005)Z^{-1}}{1 - 0.995Z^{-1}} \quad (53)$$

Pasando los denominadores al otro lado de la igualdad respectivamente obtenemos

$$Y(z)(1 - 0.995Z^{-1}) = U(z)0.005Z^{-1} \quad (54)$$

Ahora, sabiendo que Z^{-1} es el equivalente a $[k-1]$, que hace referencia a la muestra anterior de la variable a la que acompaña, pasamos del dominio Z al domino k

$$Y(k) - 0.995Y_{(k-1)} = 0.005U_{(k-1)} \quad (55)$$

Si aislamos la ecuación para quedarnos con la definición de $Y[k]$obtendremos que nuestro resultado es

$$Y(k) = 0.005U_{(k-1)} + 0.995Y_{(k-1)} \quad (56)$$

Habiendo obtenido la ecuación 55, ya somos capaces de construir el sistema equivalente para ver si el comportamiento es igual en los dos casos. El sistema resultante en paralelo con el bloque del transductor original queda como se aprecia en la figura 34.

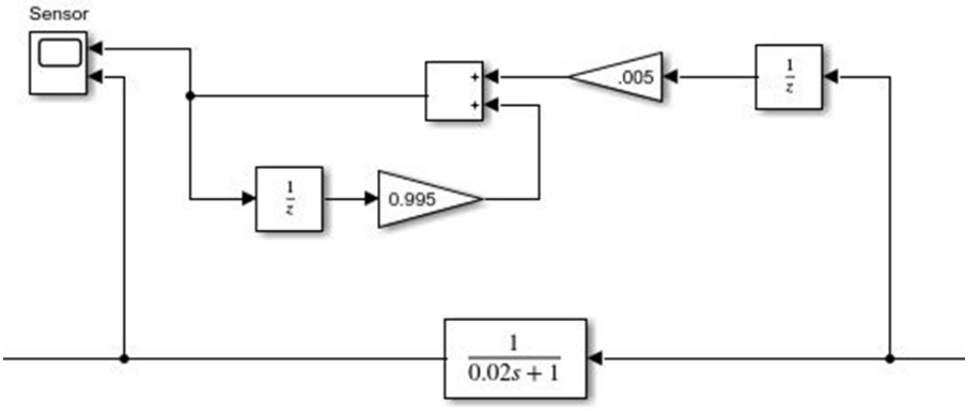

Figura 34. Bloque del original en paralelo con su propuesta equivalente

Y si observamos el resultado en el *Scope* podremos apreciar en la figura 35.

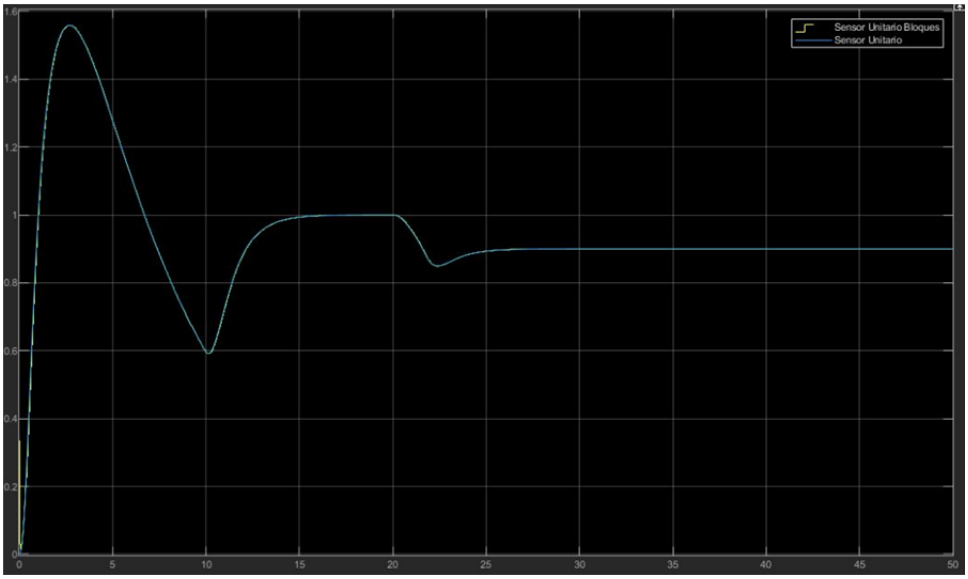

Figura 35. Comparativa de las salidas del transductor y de su circuito equivalente

Como se comportan de manera idéntica, procedemos a la implementación del código.

Las variables incluidas para esta parte son:

```
USensorAnterior = { 0, 'V' } ;
YSensorAnterior = { 0, 'V' };
YSensor = { 0, 'V' };
```

Y el código que describe su comportamiento es:

```
USensorAnterior == delay(USensor,tau);
YSensorAnterior == delay(YSensor,tau);
YSensor == 0.995 * YSensorAnterior+ 0.005 * USensorAnterior;
```

Las dos primeras líneas muestran la obtención mediante la función *Delay* de tanto la entrada como de la propia salida del transductor, necesarias para la implementación de la ecuación en diferencias que encontramos escrita en la tercera línea.

8. RESULTADOS

Una vez implementado el código, es necesario comprobar si se ha hecho una trascripción perfecta de los diagramas de bloques a código. Para ello se ha procedido a guardar en el *Workspace* de Matlab, datos de la simulación con el sistema de control construido con los bloques del propio *Simscape*, de este modo, llegados a este punto podrán ser comparados con los obtenidos mediante el sistema de control implementado en el código.

Lo que se muestra a continuación son graficas en la que se superponen las simulaciones del sistema en bloques y las simulaciones del sistema integrado en el código.

Es obligado recordar que lo que se estudia en este proyecto es el control de una máquina síncrona de polos salientes cuando esta percibe una variación en su excitación. Para ello, en el segundo 20 de la simulación, el *Step* que sirve a modo de excitación externa, varia su valor de 1 a 0.9.

8.1. COMPARATIVA BLOQUES-CÓDIGO: ERROR

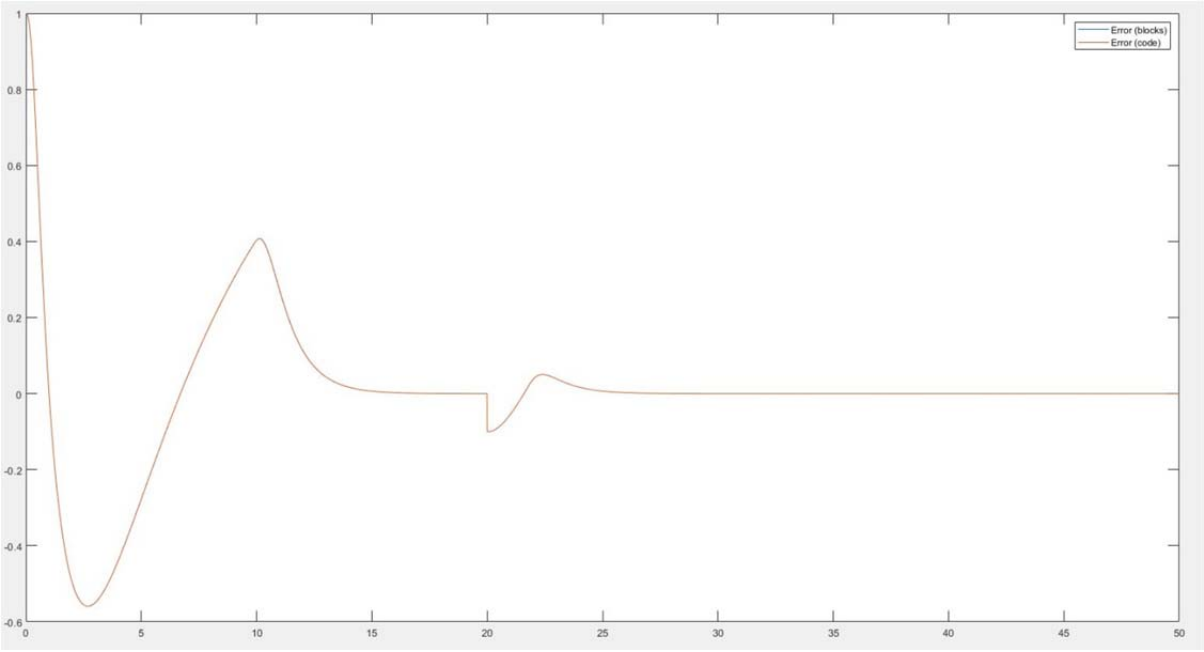

Figura 36. Comparativa diagrama de bloques- código del error

8.2. COMPARATIVA BLOQUES-CÓDIGO: PID

t

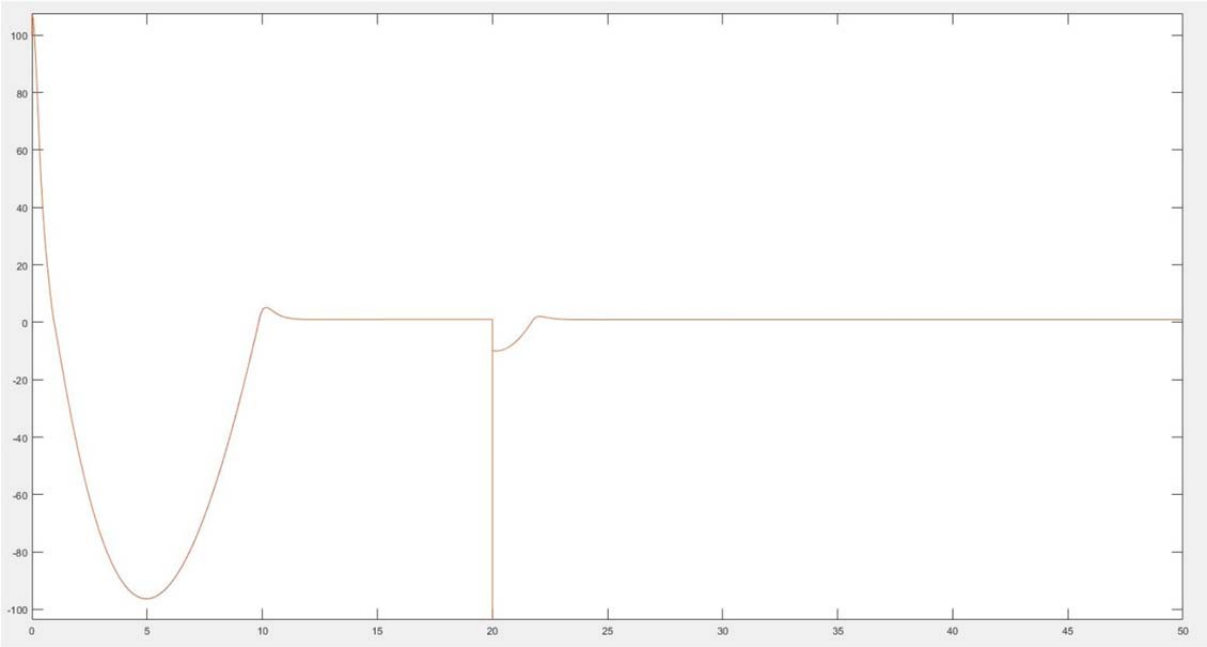

Figura 37. Comparativa diagrama de bloques - código del controlador PID

8.3. COMPARATIVA BLOQUES-CÓDIGO: SATURACIÓN

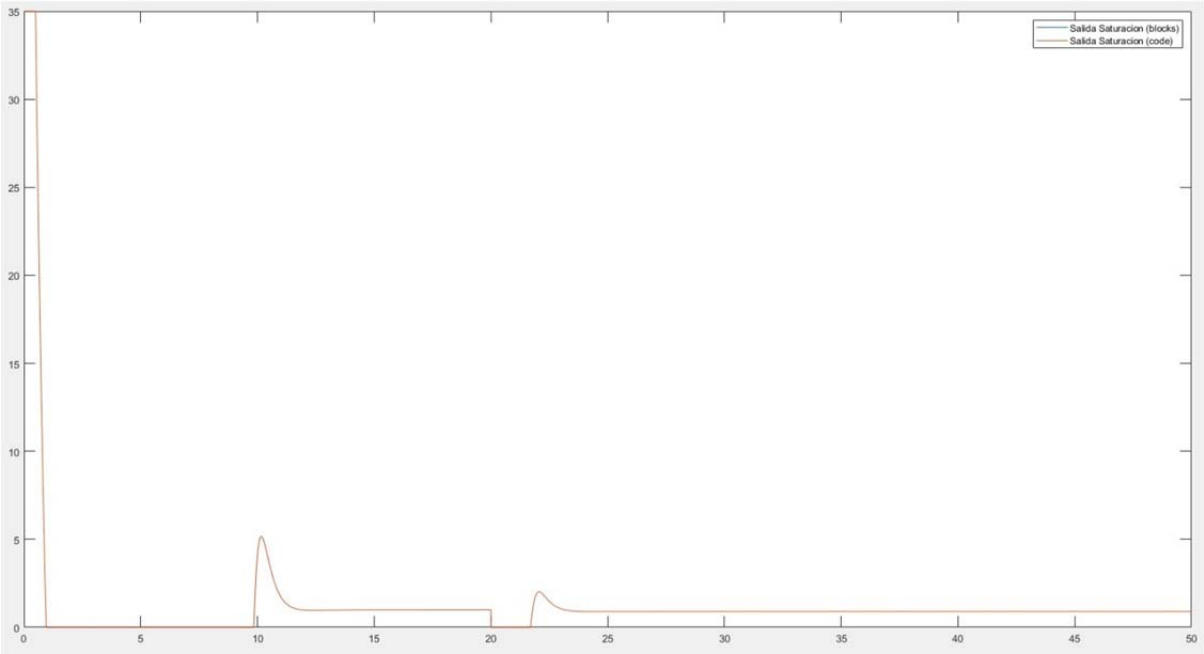

Figura 38. Comparativa diagrama de bloques- código de la saturación

8.4. COMPARATIVA BLOQUES-CÓDIGO: TRANSDUCTOR

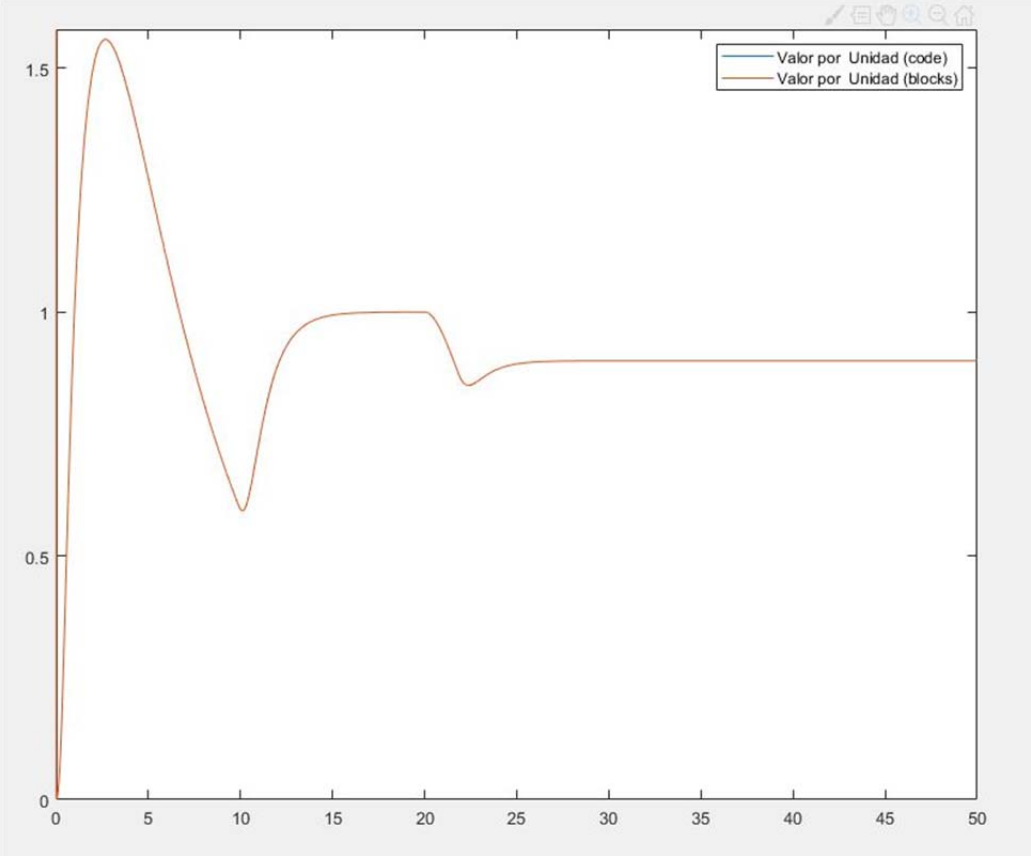

Figura 39. Comparativa diagrama de bloques- código del transductor

8.5. DIFERENCIA DE TIEMPO DE SIMULACIÓN.

Uno de los motivos por los que se ha decidido implementar el sistema de control de excitación de tensión dentro del código del propio componente, es para ahorrar tiempo durante el proceso de simulación.

Simular el sistema construido mediante los bloques externos a nuestro componente que nos proporciona *Simscape*, ha tardado 6 minutos y 6 segundos.

Por otra parte, la simulación del sistema con el sistema de control implementado en el código componente ha reducido ese tiempo a 2 minutos y 55 segundos.

Aunque es obligado decir que el tiempo de simulación varía dependiendo de las prestaciones del ordenador en el que se lleve a cabo, el haber simulado los dos casos en el mismo equipo, nos da la seguridad de poder afirmar que con el modelo con el sistema implementado en el código el tiempo de simulación se reduce a menos de la mitad.

9. CONCLUSIONES Y RECOMENDACIONES

9.1. CONCLUSIONES

En definitiva, en este trabajo se ha implementado un sistema de control para la excitación de tensión de una máquina síncrona de polos salientes, aplicando los parámetros marcados por la normativa, y usando estos mismos para calcular otros que se amoldaran mejor a nuestro sistema, como es el caso de las ganancias del controlador PID.

Se ha dado a conocer que es posible una traducción perfecta de los bloques que ofrecen las librerías de *Simscape* a pequeños subsistemas, fruto de la discretización de estos. Y podemos afirmar que se trata de una traducción eficaz debido a la coincidencia en la comparativa de los gráficos resultantes, como hemos podido ver.

También ha quedado demostrada la mayor eficacia que tiene simular los archivos transcritos a código en frente a los diagramas de bloque, ya que supone ahorro de tiempo de más del 50%.

9.2. RECOMENDACIONES PARA FUTURAS LINEAS DE TRABAJO

Con este proyecto se ha logrado un bloque de *Simscape* que define el comportamiento de una máquina síncrona de polos salientes con un sistema de control en la excitación eléctrica integrado. Una posibilidad para avanzar más con este modelo seria realizar también el modelador del sistema de excitación mecánica de la máquina.

También existe la posibilidad de cambiar el controlador para incluir un filtro derivativo, cosa que de seguro variara el resultado obtenido, pudiendo aportar mayor precisión en la respuesta. Como también resultaría un estudio variando en su totalidad los parámetros de nuestro controlador PID.

Con el sistema de control implementado, se podría conectar el componente obtenido a un sistema mayor, y estudiar el comportamiento de este variando la excitación de la máquina síncrona.

10. DISTRIBUCIÓN TEMPORAL DEL PROYECTO

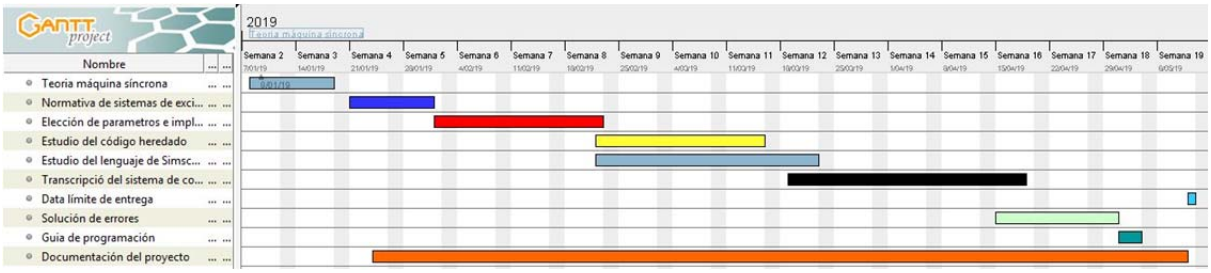

Figura 40. Distribución temporal del proyecto

Aunque la matriculación del proyecto fue para el cuatrimestre de otoño de 2018, no fue hasta inicios de 2019 que empecé a trabajar de manera seguida el proyecto. Es por ello que el *planning* de tiempo empieza entonces.

11. BIBLIOGRAFIA

[1] S. S. Paul Krause, OlegWasynczuk and S. Pekarek, Analysis Of Electric Machinery And Drive Systems. IEEE Press, 3rd edition ed., 2013.

[2] J. M. Aller, "Máquinas Eléctricas Rotativas: Introducción a la Teoría General". Editorial Equincio, Venezuela, 2007.

[3] IEEE Recommended Practice for Excitation System Models for Power System Stability Studies, IEEE Std 421.5, 2005.

[4] OGATA, Katsuhiko. Sistemas de control en tiempo discreto, Segunda edición, Prentice Hall, México,1996.

[5]IEEE Recommended Practice for Excitation System Models for Power System Stability Studies, IEEE Std 421.5, 2014

[6] P. Kundur, "Power System Stability and Control". McGraw-Hill, USA, 1994

[7] A. L. Sheldrake, "Handbook of Electrical Engineering for Practitioners in the Oil, Gas and Petrochemical Industry". John Willey ans Sons, England, 2003

[8] Signal Processing of power quality disturbances, Math H.J. Bollen AND Irene Y.H. Gu. Wiley-IEEE Press; 1st edition ed., 2006.

[9] Mathworks, Simscape Reference, 2018 ed., 2018.

[10] Mathworks, Simscape Language Guide, 2018b ed., 2018.

[11] Mathworks, Simscape Getting Started Guide, 2018b ed., 2018.

[12] www.mathworks.com/help/physmod/simscape/lang/creating-custom-components

12. IMPACTO AMBIENTAL

Este proyecto no tiene ningún impacto ambiental, más que el consumo de energía requerido por el ordenador durante las horas de trabajo.

El hecho de que no tenga ningún impacto ambiental serio se debe a que todo lo logrado y propuesto es en base a simulaciones realizadas en entornos virtuales.

13. PRESUPUESTO

Actividad	Coste(€/hora)	Horas	Coste total
Familiarización con la máquina síncrona	10	7	70
Normativas de sistemas de excitación	10	10	100
Parametrización modelo de excitación	10	20	200
Construcción modelo de excitación	10	10	100
Estudio del lenguaje de programación de Simscape	10	40	400
Implementación del código	10	55	550
Solución de errores	10	15	150
Comparación de resultados	10	7	70
Desarrollo de documentación	10	25	250
MATLAB license (Indiv. Annual)		-	800
TOTAL	-	**177**	**2690**

A. GUIA DE PROGRAMACIÓN

Este documento pretende ser una guía para todo aquel que pretenda iniciar una programación de un bloque *Custom Component* en *Simscape*, o desee entender de manera rápida el funcionamiento de un programa ya existente.

La información de este anexo procede de los documentos [9], [10], [11] y del domino [12]

A.1. INTRODUCCÓN

En el modelado físico, hay dos tipos de modelos:

- Conductual: un modelo que se implementa en función de su comportamiento físico, descrito por un sistema de ecuaciones matemáticas.
- Compuesto: un modelo que se construye a partir de otros bloques, conectados de cierta manera. Un ejemplo de una implementación de bloque compuesto o estructural es el bloque de válvulas, que se construiría en base a diversos bloques de orificio variable.

El lenguaje *Simscape* permite crear nuevos modelos conductuales y compuestos para cuando las bibliotecas de bloques estándar proporcionados con *Simscape* y sus productos complementarios no satisfacen los requisitos de un diseño.

Un requisito previo para crear componentes es tener los dominios adecuados para los nodos de tu componente, ya que este tipo de bloques trabajan únicamente con variables de dominios físicos, a diferencia de *Simulinik*, que trabaja con señales.

A.2. QUE ES SIMSCAPE

Simscape es un conjunto de bibliotecas de bloques y características especiales de la simulación para modelar sistemas físicos que existe en *Simulink*. Emplea el enfoque de red física, que difiere de la estándar *Simulink* (cuyo enfoque es de modelado), y es particularmente adecuado para la simulación de sistemas que consisten en componentes físicos reales.

Los bloques de *Simulink* representan las operaciones matemáticas básicas. Cuando se conectan los bloques de *Simulink*, el diagrama resultante es equivalente a la representación, del sistema diseño o modelo matemático. *Simscape* permite crear una representación de la red del sistema de diseño, basado en el enfoque de red física.

Según este enfoque, cada sistema se representa como compuesto de elementos funcionales que interactúan entre sí mediante el intercambio de energía a través de sus puertos.

Estos puertos de conexión son no direccionales. Imitan las conexiones físicas entre los elementos. Para entenderlo mejor, conectar los bloques de *Simscape* es lo mismo que realizar la conexión de componentes reales, tales como bombas, válvulas, etc. En otras palabras, los diagramas *Simscape* imitan el diseño del sistema físico. No se requieres que especificar las direcciones de flujo ni flujo de información al conectar bloques *Simscape*, al igual que no tienes que especificar esta información cuando conectas los componentes físicos reales.

El número de puertos de conexión para cada elemento se determina por el número de flujos de energía que intercambia cada bloque con otros elementos en el sistema y depende del nivel de idealización. Por ejemplo, una bomba hidráulica de desplazamiento fijo en su forma más simple puede ser representada como un elemento de dos puertos, con un flujo de energía asociado a la entrada (succión) y el otro con la salida.

Un flujo de energía se caracteriza por sus variables, y cada flujo de energía está asociado a dos variables, *Across* y *Through*.

A.3. TIPOS DE VARIABLES

El enfoque de red física admite dos tipos de variables:

- *Through*: variables que se miden con un indicador conectado en serie a un elemento.
- *Across*: variables que se miden con un medidor conectado en paralelo a un elemento.

En la figura 41 podemos observar las variables *Across* de cada tipo de dominio físico en el software *Simscape*:

Physical Domain	Across Variable
Electrical	Voltage
Hydraulic	Pressure
Magnetic	Magnetomotive force (mmf)
Mechanical rotational	Angular velocity
Mechanical translational	Translational velocity
Gas	Pressure and temperature
Moist Air	Pressure, temperature, specific humidity (water vapor mass fraction), and trace gas mass fraction
Thermal	Temperature
Thermal liquid	Pressure and temperature
Two-phase fluid	Pressure and specific internal energy

Figura 41. Comparativa diagrama de bloques- código del transductor

Y en la figura 42 podemos observar las variables *Through* de cada tipo de dominio físico en el software *Simscape*:

Physical Domain	Through Variable
Electrical	Current
Hydraulic	Flow rate
Magnetic	Flux
Mechanical rotational	Torque
Mechanical translational	Force
Gas	Mass flow rate and energy flow rate
Moist Air	Mixture mass flow rate, mixture energy flow rate, water vapor mass flow rate, and trace gas mass flow rate
Thermal	Heat flow
Thermal liquid	Mass flow rate and energy flow rate
Two-phase fluid	Mass flow rate and energy flow rate

Figura 42. Dominios físicos con sus correspondientes variables Through

A.4 CREACIÓN DE UN NUEVO DOMINIO:

Simscape Foundation ofrece una amplia variedad de dominios físicos que se pueden usar pero en caso de que no encontremos aquí las variables que deseamos usar, siempre podemos crear nuestro propio dominio, y, en consecuencia, nuestras propias variables.

Para crear un nuevo dominio procederemos de la siguiente manera:

Un archivo de dominio debe comenzar con la palabra clave del *domain*, seguido del nombre del dominio, y terminar con la palabra clave *end*. Los archivos de creación de dominio contienen únicamente la sección de declaración de variables.

Se requieren dos bloques de declaración:

- El bloque de declaración de variables *Across*: que comienza con la palabra clave *variables* y termina con la palabra clave *end*. Contiene declaraciones para todas las variables *Across* asociadas con el dominio.
- El bloque de declaración de variables *Through*: que comienza con la palabra clave *variables (Balancing = true)* y termina con la palabra clave *final*. Contiene declaraciones para todas las variables de paso asociadas con el dominio.

Un ejemplo sería el siguiente:

```
domain rotacional
% Define el dominio mecanico rotacional
% en cuanto a las variables across y through

  variables
    w = { 1 , 'rad/s' }; % velocidad angular
  end

  variables(Balancing = true)
    t = { 1 , 'N*m' }; % momento
  end

end
```

A.5 DESARROLLO DEL CÓDIGO

La creación de un código que define el comportamiento de un *Custom Component* consiste en la declaración de las diferentes secciones que sean precisadas.

A continuación se mencionan las secciones que puede incluir dicho código.

Para que el código funcione, no es necesaria la existencia de todas las secciones descritas en este documento.

A.5.1 COMPONENT

La palabra *component* comienza la definición de la clase de modelo de componente, seguido del nombre del componente, y termina con una palabra clave *end*. Sólo líneas en blanco y comentarios pueden preceder al componente.

La definición de clase del modelo de componente debe ser guardada en un archivo del mismo nombre con una extensión de nombre de archivo de .ssc.

Entre *component* y su correspondiente *end* irán descritas el resto de secciones que van a definir nuestro custom component. El orden en que se definan las secciones no es relevante, incluso podemos encontrar secciones repetidas.

A.5.2 NODES

La palabra *nodos* comienza un bloque de declaración, que termina con una palabra clave *end*. Este bloque contiene declaraciones de todos los nodos del componente, que corresponden a los puertos de conservación de un bloque *Simscape*. Es decir, en esta sección se definen las entradas y salidas físicas que va a tener el componente.

Cada nodo se define por asignación a un dominio existente. Consulte Declarar nodos componentes para obtener más información.

La sintaxis para definir un nodo necesita que conozcamos el dominio que va a tener ese nodo, y nos permite, aparte de dar un nombre al nodo creado, poner una etiqueta que se mostrara físicamente en el bloque y decidir en qué posición del bloque irá colocado.

El siguiente ejemplo crea un nodo llamado "Vm" de dominio *foundation electric*, que en el bloque se marcará con un V+ y que estará situado en el lado izquierdo.

```
Nodes

Vm = foundation.electric; % V+:left

end
```

A.5.3 INPUTS

La palabra *inputs* comienza un bloque de definición de entradas de componente, que termina con una palabra clave *end*. Este bloque contiene declaraciones para entradas de componentes. Las entradas aparecerán como puertos de entrada de señal física en el diagrama de bloques cuando el archivo del componente se incorpore a un modelo de *Simscape*. A diferencia de los nodos, que transmiten los valores que les llegan des de conexiones exteriores al componente, los inputs mantienen siempre el mismo valor.

Cada entrada se define como un valor con unidad, donde el valor puede ser un escalar, un vector o una matriz. Para un vector o una matriz, todas las señales tienen la misma unidad.

La especificación de un comentario opcional le permite controlar la etiqueta del puerto y la ubicación en el icono de bloque.

El siguiente ejemplo crea un input llamado "A" de valor 5 voltios, que en el bloque se marcará con un A+ y que estará situado en el lado derecho.

```
inputs
    A = { 5 , 'V' };  % A+:right
end
```

A.5.4 OUTPUTS

La palabra *outputs* inicia un bloque de definición de salidas de componente, que termina con una palabra clave *end*. Este bloque contiene declaraciones para salidas de componentes. Las salidas aparecerán como puertos de salida de señal física en el diagrama de bloques cuando el archivo del componente se incorpore a un modelo de *Simscape*.

Cada salida se define como un valor con unidad, donde el valor puede ser un escalar, un vector o una matriz. Para un vector o una matriz, todas las señales tienen la misma unidad. A diferencia de los nodos, que transmiten los valores que les llegan des del interior del componente, los outputs mantienen siempre el mismo valor.

La especificación de un comentario opcional le permite controlar la etiqueta del puerto y la ubicación en el icono de bloque.

El siguiente ejemplo crea un input llamado "B" de valor 5 voltios, que en el bloque se marcará con un B+ y que estará situado en el lado izquierdo.

```
outputs
    B = { 5 , 'V' };  % B+:left
end
```

A.5.5 PARAMETERTS

Permiten especificar parámetros ajustables para el bloque *Simscape* generado desde el archivo del componente. Los parámetros aparecerán en el cuadro de diálogo de bloque, como la que se muestra en la figura 43 y se pueden modificar al construir y simular un modelo.

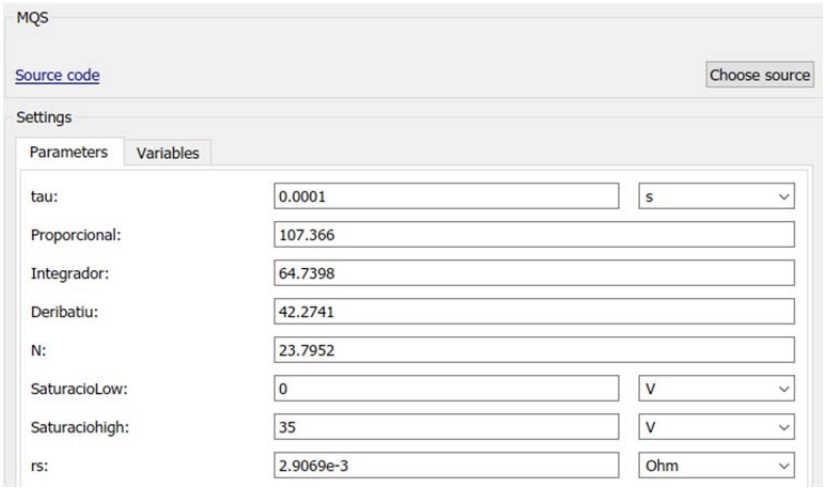

Figura 43. Cuadro de dialogo de un Custom Component

La palabra *parameters* comienza un bloque de definición de parámetros de componente, que termina con una palabra clave *end*. Este bloque contiene declaraciones de los parámetros del componente. Los parámetros aparecerán en el cuadro de diálogo de bloque cuando el archivo del componente se lleve a un modelo de *Simscape*. Cada parámetro se define como un valor con unidad. La especificación de un comentario opcional le permite controlar el nombre del parámetro en el cuadro de diálogo del bloque.

La siguiente sintaxis define un parámetro componente, *PerioroMuestreo*, como un valor con unidad. Donde el primer argumento es el valor inicial, y el segundo la unidad. En caso que querer un parámetro sin unidad, se sustituye la unidad en el segundo argumento por un '1'.

En este caso el valor es 0.01 y las unidades serán segundos.

```
parameters
    PeriodoMuestreo = { 0.01 , 's' };
end
```

A.5.6 VARIABLES

La palabra *variables* comienza un bloque de declaración de variables, que termina con una palabra clave *end*. En este bloque definimos las variables que usaremos a en nuestro programa, y les vamos a dar un valor inicial, junto con una unidad.

En un archivo de componente, este bloque contiene declaraciones para todas las variables asociadas con el componente. En un archivo de dominio, este bloque contiene declaraciones para todas las variables Across asociadas con el dominio. Además, los archivos de dominio deben tener un bloque de declaración de variables separado, con el atributo *balance* establecido en *true*, que contiene declaraciones para todas las variables de paso asociadas al dominio.

Para las variables del componente, es posible especificar adicionalmente la prioridad de inicialización, así como el valor nominal y la unidad, declarando la variable como una matriz de campo. La figura 44 muestra una tabla con las prioridades posibles.

Priority field in Simscape™ language	Resulting default priority in the block dialog box
priority = priority.high	High
priority = priority.low	Low
priority = priority.none (this is the default)	None

Figura 44. Tabla de prioridades en las variables

El ejemplo declara una variable llamada I, con valor inicial 0A y prioridad de inicialización alta.

```
variables
    I = { value = { 0, 'A' }, priority = priority.high };
end
```

El siguiente ejemplo define una variable Through que inicializa con un valor de 3 A.

```
variables(Balancing = true)
    through_var1 = { 3 , 'A' };
end
```

A.5.7 BRANCHES

La palabra *branches* da comienzo a la propia sección branches, que termina con una palabra clave *end*. Esta sección contiene una o más declaraciones de branches, que establecen la relación entre las variables de Through del componente y el dominio.

El nombre de la variable del componente no tiene que coincidir con el de la variable de dominio, pero las unidades deben ser proporcionales (por ejemplo, 'N', 'kg * m / s ^ 2', 'lbf')

Para establecer una conexión entre la variable de componente a y la variable de dominio a través (equilibrio) a, escriba una declaración de bifurcación, como:

```
branches
    a : node1.a -> node2.a;
end
```

En el siguiente ejemplo queda todo muy claro. En este caso isa, isb e isc son las variables declaradas anteriormente en el apartado de variables, y almacenan la corriente que hay entre sus respectivos nodos, y el nodo n.

```
branches %Through variables - Connections
         %ELECTRICAL: Current i
    isa : a.i -> n.i; % Current through from node a to node n
    isb : b.i -> n.i; % Current through from node b to node n
    isc : c.i -> n.i; % Current through from node c to node n
end %branches
```

Dependiendo del dominio de las variables, la letra que acompaña el nombre del nodo detrás del punto varía. En el caso anterior al ser corriente, es " .i"

Pero como vemos en el ejemplo siguiente, que se trata de dominio mecánico, hace referencia al torque, y ello se representa con " .t"

```
Tres : R.t -> C.t
```

A.5.8 EQUATIONS

La palabra *equations* comienza la sección de ecuaciones en un archivo componente; esta sección termina con una palabra clave *end*. Se ejecuta a lo largo de la simulación. El propósito de la sección de la ecuación es establecer las relaciones matemáticas entre las variables, parámetros, entradas, salidas, tiempo y las derivadas del tiempo de cada componente de un componente. Todos los miembros declarados en el componente están disponibles por su nombre en la sección de ecuación.

Algún ejemplo de ecuación sería el siguiente.

```
vs0 == sqrt(2/3)*( 1/sqrt(2)*vsa    +       1/sqrt(2)*vsb          +     1/sqrt(2)*vsc );
vsd == sqrt(2/3)*( cos(p*fitam)*vsa  +  cos(p*fitam - (2*pi/3))*vsb  +   cos(p*fitam + (2*pi/3))*vsc );
vsq == sqrt(2/3)*( -sin(p*fitam)*vsa -  sin(p*fitam - (2*pi/3))*vsb  -   sin(p*fitam + (2*pi/3))*vsc );
```

Es aquí también donde se implementan las relaciones de las variables *Across*, del mismo modo que se ha hecho con las variables *Through* en el apartado Branches.

El ejemplo que sigue seria el mismo caso que para calcular las intensidades de los nodos a, b y c respecto a n, pero en este caso de guarda el valor de la tensión en las variables vsa, vsb y vsc.

```
vsa == a.v - n.v;
vsb == b.v - n.v;
vsc == c.v - n.v;
```

A.5.9 INTERMEDIATES

La palabra *intermediates* comienza un bloque de declaración de intermedios, que termina con una palabra clave *end*. En un archivo componente, este bloque contiene declaraciones de términos intermedios nombrados para su uso en ecuaciones. Puede reutilizar estos términos intermedios en cualquier sección de ecuaciones del mismo componente.

Cuando se utiliza un término intermedio en una ecuación, se sustituye en última instancia con la expresión a la que se refiere. Se puede entender como un término intermedio, como en la definición de un alias para una expresión.

La declaración de términos intermedios ayuda con la reutilización y legibilidad del código.

Un ejemplo sería el siguiente.

```
intermediates
    int_term1 = expr1;
end
```

A.6 CREACIÓN DEL COMPONENTE E IMPORTACIÓN DEL FICHERO DE CODIGO

Una vez se ha escrito el código del componente, llega el momento de crear el bloque.

Para ello vamos a la biblioteca de *Simscape* y buscamos el bloque llamado *Simscape Component* y que podemos apreciar en la figura 45.

Figura 45. Bloque Simscape Component

Como podemos observar aparece un mensaje en el interior de bloque diciendo *".ssc Unspecified"*, eso es porque no se ha introducido el código aun.

Para ello, si accedemos al interior del bloque con doble clic, se nos abrirá el menú mostrado en la figura 46.

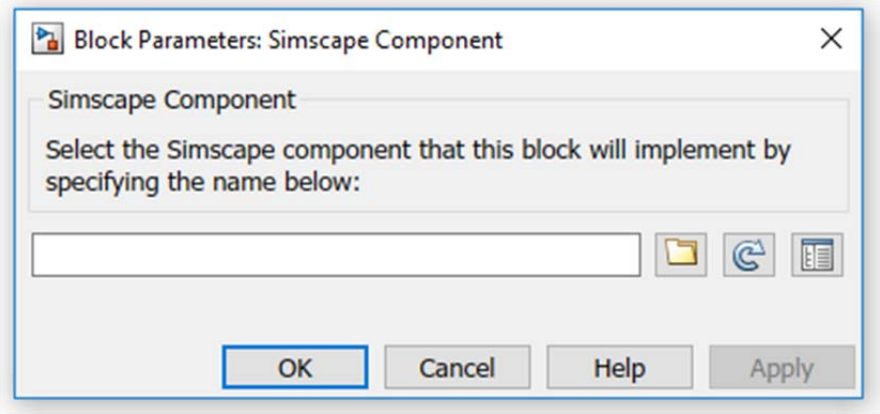

Figura 46. Cuadro de dialogo de Simscape Component sin archivo de código importado

Ahora tenemos que importar nuestro archivo al bloque, mediante la opción "*Select component file*", seleccionando el archivo de código que hemos creado. Cuando le demos al OK, el bloque cambiara, mostrando las entradas y salidas que hemos especificado mediante el código, y eso significará, que el bloque estará listo para ser conectado a un circuito.

B. SIMSCAPE RESULTS INSPECTOR

B.1 PROBLEMA

Cuando se simula un circuito que contiene un *Simscape Component*, los datos de las variables internas de este componente se nos muestran mediante una herramienta que recibe el nombre de *Simscape Results Explorer*.

Mediante esta herramienta podemos visualizar en forma de gráficos los valores de dichas variables respecto al tiempo. Algo muy útil si lo que se quiere es comparar dos variables internad del componente. O simplemente visualizar su comportamiento.

El problema aparece cuando necesitamos comparar estos gráficos, con otros externos al componente, como puede ser por ejemplo, un *Scope* que lea una de las entradas al componente. Aunque se encuentren conectados de manera externa a este, no hay manera directa de lograr superponerlos en el mismo gráfico.

Para lograrlo necesitamos de la ayuda de la herramienta llamada *Simulation Data Inspector.*

B.2 PROCEDIMINETO

Para ello necesitamos hacer una pequeña variación en nuestro sistema externo, y es que necesitamos sustituir el bloque *Scope*, por un bloque "to *Workspace*", que lo que va a hacer es crear una variable en el workspace de Matlab con el nombre que decidamos darle y enviar los valores medidos.

Habiendo hecho este cambio, simulamos, para que los datos que nos interesan sean guardados.

Finalizada la simulación se nos abrirá el *Simscape Results Explorer,* pero la minimizaremos, ya que no lo necesitamos.

Tenemos que ir, en nuestra ventana de *Simscape*, a Simulation → Output → Simulation Data Inspector, y se nos abrirá la ventana de la figura 47

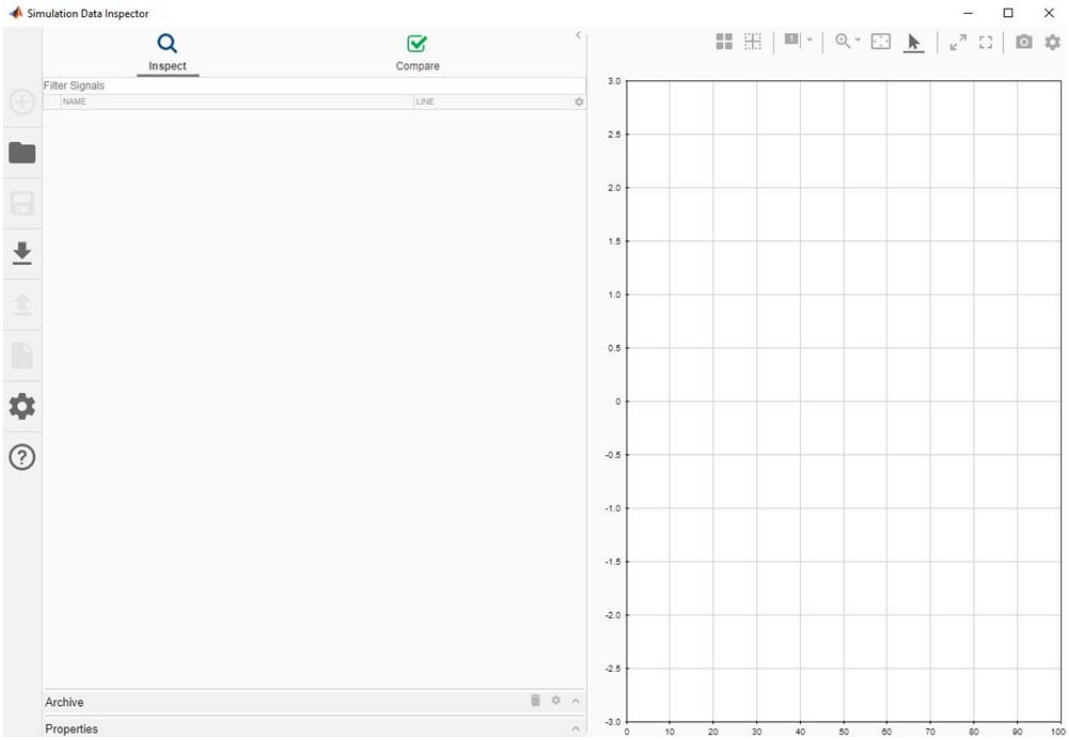

Figura 47. Vista inicial de Simulation Data Inspector

En la parte derecha, buscaremos la opción "Import", que nos va a permitir importar las variables internas de nuestro Simscape Component a esta herramienta.

En la sección Import tenemos que seleccionar "*Import from: Base Workspace*" e *"Import to: New Run*", y en la parte inferior asegurarnos de que el archivo con nuestro sistema esta seleccionado, como podemos apreciar en la figura 48.

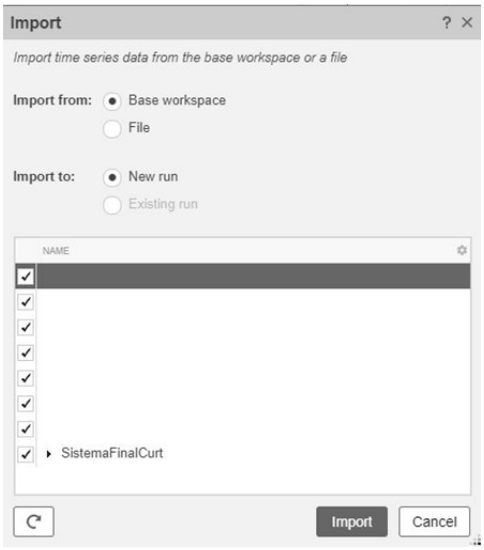

Figura 48. Ventana emergente de la opción "Import"

Cuando seleccionemos la opción "*Import*", que se encuentra en la parte inferior el programa cargará todas las variables de nuestro *Simscape Component*, y tendremos algo como lo mostrado en la figura 49, donde cada variable tendrá su color representativo a la derecha, y donde si seleccionamos algunas veremos cómo quedan representadas en el espacio grafico de la derecha de la pantalla. Pero esto no es lo que queremos.

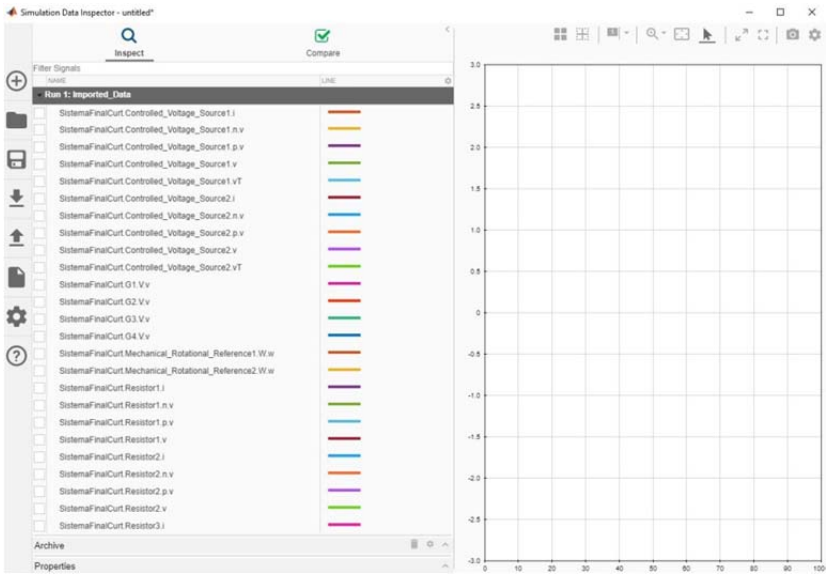

Figura 49. Simulation Data Inspectos con las variables de nuestro componente cargadas.

Lo que necesitamos de esta herramienta es que nos permita exportar la variable que queremos comparar al workspace de Matlab.

Para ello basta con hacer click derecho sobre la variable que queremos exportar, y seleccionar la opción "*Export*", y se nos abrirá una ventana emergente como la que se parecía en la figura 50.

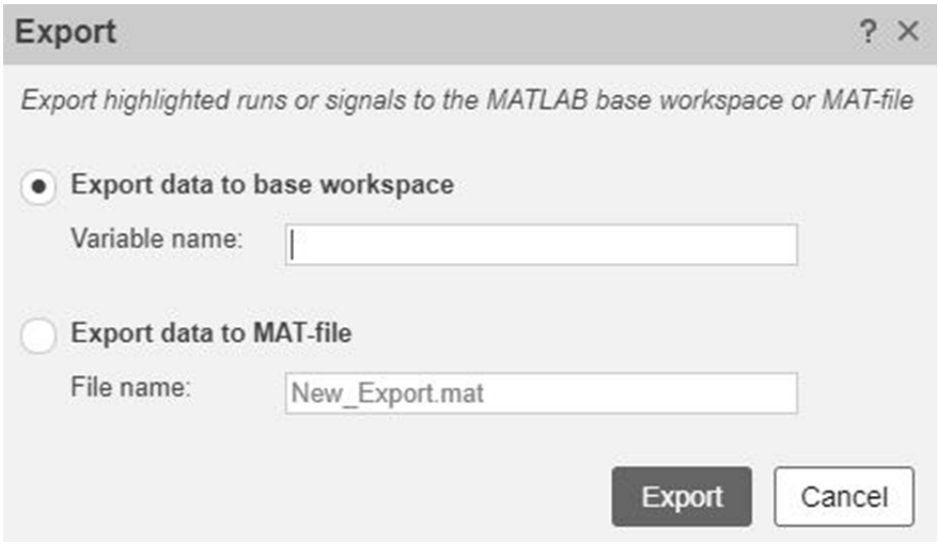

Figura 50. Proceso de exportación de una variable al workspace de Matlab

En este punto tenemos que seleccionar la opción *Export* data to base Workspace, y poner un nombre a nuestra variable.

Hecho esto, tendremos en el workspace de Matlab la variable externa, procedente del bloque "to Workspace" y la variable interna de nuestro *Simscape Component*.

Ahora, es tan fácil como hacer un simple plot de las dos variables para que nos aparezcan en la misma figura, y así podamos compararlas.

El comando para hacerlo es el siguiente, sustituyendo "variable1" y "variable2" por los nombres de nuestras variables en el workspace de Matlab.

Plot(variable1.time,variable1.data, variable2.time,variable2.data)

C. CODIGO RESULTANTE

```
component MQS

    nodes
        %Stator phase A electrical connection
        a = foundation.electrical.electrical; %a:right
        %Stator phase B electrical connection
        b = foundation.electrical.electrical; %b:right
        %Stator phase C electrical connection
        c = foundation.electrical.electrical; %c:right
        %Stator phase neutral electrical connection
        n = foundation.electrical.electrical; %n:right
        %Field winding positive connection
        fp = foundation.electrical.electrical; %+:left
        %Field winding negative connection
        fm = foundation.electrical.electrical; %-:left
        %Rotor shaft
        R = foundation.mechanical.rotational.rotational; %R:left
        %Rotor shaft angle
        C = foundation.mechanical.rotational.rotational; %C:left

    end% nodes

    parameters

        tau = {0.0001,'s'};
        Proporcional   = {107.366, '1'};
        Integrador   = {64.7398, '1'};
        Deribatiu   = {42.2741, '1'};


        SaturacioLow = {0, 'V'};
        SaturacioHigh = {35, 'V'};

        rs  = {2.9069e-3, 'Ohm'};
        rF  = {5.9013e-4, 'Ohm'};
        r1D = {1.1900e-2, 'Ohm'};
        r1Q = {2.0081e-2, 'Ohm'};

        L0  = {3.0892e-4, 'H'};
        Ld  = {3.52532e-3, 'H'};
        Lq  = {1.28045e-3, 'H'};
        L1D = {3.70716e-3, 'H'};
        L1Q = {2.00803e-3, 'H'};
```

```
    LF  = {3.52352e-3, 'H'};

    Md  = {3.2164e-3, 'H'};
    Mq  = {9.7153e-4, 'H'};
    MFD = {3.2164e-3, 'H'};

    B   = {0, '(N*m)/(rad/s)'};    %Friction coefficient
    J   = {0.2498, 'kg*m^2'};  %Rotor Inertia
   p    = {20, '1'};           %Pole pairs

end%parameters

variables

    ExcitacioUnitaria = { 0, 'V' } ;

    YPID =  { 0, 'V' };
    ValorTau =  { 0, '1' } ;
    SumatoriError = { 0, 'V' } ;
    SumatoriErrorAnterior = { 0, 'V' } ;
    Error = { 0, 'V' } ;
    ErrorAnterior = { 0, 'V' }

    USensorAnterior = { 0, 'V' } ;
    YSensorAnterior = { 0, 'V' };
    YSensor = { 0, 'V' };

    isa = { 0, 'A' }; %A terminal current
    isb = { 0, 'A' }; %B terminal current
    isc = { 0, 'A' }; %C terminal current
    vsa = { 0, 'V' }; %A terminal voltage
    vsb = { 0, 'V' }; %B terminal voltage
    vsc = { 0, 'V' }; %C terminal voltage
    VF =  { 0, 'V' }; %Excitation voltage


    YSaturacion= { 0, 'V' };

    Funcio = { 0, 'V' };
    UExcitatriz = { 0, 'V' };
    UExcitatrizAnterior = { 0, 'V' };
    YExcitatriz = { 0, 'V' };
```

```
        YExcitatrizAnterior = { 0, 'V' };
        Yy = { 0, '1' };



        is0 = { value = { 0, 'A' }, priority = priority.high };
        isd = { value = { -6.0149, 'A' }, priority = priority.high };
        isq = { value = { -7.0069, 'A' }, priority = priority.high };
        iF  = { value = { 16.5143, 'A' }, priority = priority.high };
        i1D = { value = { 0, 'A' }, priority = priority.high };
        i1Q = { value = { 0, 'A' }, priority = priority.high };

        vs0 = { 0, 'V' };
        vsd = { 0, 'V' };
        vsq = { 0, 'V' };

        wm  = { value = { 18.8496*1,'rad/s'}, priority = priority.high };
        fitam  = { value = { -37.5051 ,'deg'}, priority = priority.high };

        Te   = {0, 'N*m'};
        Tres = {0, 'N*m'};

    end%variables

    branches

        isa : a.i -> n.i;
        isb : b.i -> n.i;
        isc : c.i -> n.i;
        iF : fp.i -> fm.i;
        Tres : R.t -> C.t

   end %branches

   equations
        vsa == a.v - n.v;
        vsb == b.v - n.v;
        vsc == c.v - n.v;
        ExcitacioUnitaria  == fp.v-fm.v;
        wm   == R.w - C.w;

%EntradaPid
Error == ExcitacioUnitaria - YSensor;
```

```
%Controlador PID
ValorTau == value(tau,'s');
SumatoriErrorAnterior == delay(SumatoriError,tau);
ErrorAnterior == delay(Error,tau);
SumatoriError == SumatoriErrorAnterior + Error;

YPID == Proporcional * Error + Integrador * ValorTau * SumatoriError+ (Deribatiu*(Error-ErrorAnterior))/ValorTau;


%Exitador

if YPID < SaturacioLow
    YSaturacion== { 0, 'V' };
elseif YPID > SaturacioHigh
    YSaturacion== { 35, 'V' };
else
    YSaturacion== YPID;
end


YExcitatrizAnterior == delay(YExcitatriz,tau);
UExcitatrizAnterior == delay(UExcitatriz,tau);
Yy == value(YExcitatrizAnterior,'V')
UExcitatriz == YSaturacion-(Funcio+YExcitatrizAnterior);
Funcio == 7.537326681679638e-004*exp(0.921*Yy)*YExcitatrizAnterior;
YExcitatriz == (0.0001/1.2)*UExcitatrizAnterior+YExcitatrizAnterior;



VF == YExcitatriz*6.723;
vs0 == sqrt(2/3)*( 1/sqrt(2)*vsa    +       1/sqrt(2)*vsb          +     1/sqrt(2)*vsc );
vsd == sqrt(2/3)*( cos(p*fitam)*vsa  +  cos(p*fitam - (2*pi/3))*vsb  +   cos(p*fitam + (2*pi/3))*vsc );
vsq == sqrt(2/3)*( -sin(p*fitam)*vsa -  sin(p*fitam - (2*pi/3))*vsb  -   sin(p*fitam + (2*pi/3))*vsc );


%transductor
USensorAnterior == delay(USensor,tau);
YSensorAnterior == delay(YSensor,tau);
YSensor == 0.995 * YSensorAnterior+ 0.005 * USensorAnterior;
```

```
%Eqs Matlab

is0.der == (vs0 - is0*rs)/L0;

isd.der == ((L1D*Md - MFD*Md)*(VF - iF*rF))/...
           (L1D*Md^2 + LF*Md^2 + Ld*MFD^2 - 2*MFD*Md^2 - L1D*LF*Ld) +...
           ((MFD^2 - L1D*LF)*(vsd - isd*rs + Lq*isq*p*wm + Mq*i1Q*p*wm))/...
           (L1D*Md^2 + LF*Md^2 + Ld*MFD^2 - 2*MFD*Md^2 - L1D*LF*Ld) - ...
           (i1D*r1D*(LF*Md - MFD*Md))/(L1D*Md^2 + LF*Md^2 + Ld*MFD^2 - 2*MFD*Md^2 - L1D*LF*Ld);

isq.der == (L1Q*(isq*rs - vsq + Ld*isd*p*wm + Md*i1D*p*wm + Md*iF*p*wm))/...
           (Mq^2 - L1Q*Lq) - (Mq*i1Q*r1Q)/(Mq^2 - L1Q*Lq);

iF.der == ((L1D*Md - MFD*Md)*(vsd - isd*rs + Lq*isq*p*wm + Mq*i1Q*p*wm))/...
          (L1D*Md^2 + LF*Md^2 + Ld*MFD^2 - 2*MFD*Md^2 - L1D*LF*Ld) +...
          ((Md^2 - L1D*Ld)*(VF - iF*rF))/(L1D*Md^2 + LF*Md^2 + Ld*MFD^2 -...
          2*MFD*Md^2 - L1D*LF*Ld) + (i1D*r1D*(Md^2 - Ld*MFD))/...
          (L1D*Md^2 + LF*Md^2 + Ld*MFD^2 - 2*MFD*Md^2 - L1D*LF*Ld);

i1D.der == ((LF*Md - MFD*Md)*(vsd - isd*rs + Lq*isq*p*wm + Mq*i1Q*p*wm))/...
           (L1D*Md^2 + LF*Md^2 + Ld*MFD^2 - 2*MFD*Md^2 - L1D*LF*Ld) - ...
           ((Md^2 - Ld*MFD)*(VF - iF*rF))/(L1D*Md^2 + LF*Md^2 + Ld*MFD^2 - ...
           2*MFD*Md^2 - L1D*LF*Ld) - (i1D*r1D*(Md^2 - LF*Ld))/ ...
           (L1D*Md^2 + LF*Md^2 + Ld*MFD^2 - 2*MFD*Md^2 - L1D*LF*Ld);

i1Q.der == (Lq*i1Q*r1Q)/(Mq^2 - L1Q*Lq) - ...
           (Mq*(isq*rs - vsq + Ld*isd*p*wm + Md*i1D*p*wm + Md*iF*p*wm))/(Mq^2 - L1Q*Lq);

Te    == p*( isq*( Md*iF + Md*i1D ) - isd*Mq*i1Q + ( Ld - Lq )*isd*isq );
wm.der == (1/J)*( Te + Tres - B*wm );
fitam.der == wm;                                    %dxmdt dxdt(8)

%We undo Park's transformation

isa == sqrt(2/3)*(1/sqrt(2)*is0 + cos(p*fitam)*isd - sin(p*fitam)*isq);
isb == sqrt(2/3)*(1/sqrt(2)*is0 + cos(p*fitam - (2*pi/3))*isd - sin(p*fitam - (2*pi/3))*isq);
isc == sqrt(2/3)*(1/sqrt(2)*is0 + cos(p*fitam + (2*pi/3))*isd - sin(p*fitam + (2*pi/3))*isq);
```

```
    end %equations
  intermediates
  % f = 0.316 / Re_d^0.25; % Darcy friction factor
  %conversion trifasica
  vsarms = sqrt(sum((abs(vsa))^2)/length(vsa))
  vsbrms = sqrt(sum((abs(vsb))^2)/length(vsb))
  vscrms = sqrt(sum((abs(vsc))^2)/length(vsc))
  USensor = sqrt(vsarms^2+vsbrms^2+vscrms^2)/13800/sqrt(3)*sqrt(3)




  end
  end
```