Trabajo de Fin de Grado

Aplicación de técnicas de control fraccionario y comparación de resultados con respecto al control clásico

Autores:

Sara Estévez Pérez

Javier León Gil

Alfonso Rodrigo Matesanz García

Tutores:

Leopoldo Acosta Sánchez

Antonio Luis Morell González

Julio, 2015

AGRADECIMIENTOS

En esta página queremos hacer mención a todas aquellas personas que de una manera u otra han hecho posible la realización de este trabajo.

A nuestros padres, hermanos y familiares, por su incondicional apoyo durante todos estos años de esfuerzos y lucha continua.

A todo el personal del laboratorio de Ingeniería de Sistemas de la Universidad de La Laguna, por su ayuda en la solución de todos los inconvenientes que nos han surgido. Mención especial a D. Manuel Fernández Vera y D. Jonay T. Toledo Carrillo por las horas dedicadas a nuestro favor y su implicación personal.

A nuestro profesor D. Santiago Torres Álvarez, por su inestimable ayuda en los momentos que no conseguíamos avanzar por problemas ajenos al desempeño del trabajo de fin de grado.

Finalmente, a nuestros tutores, D. Leopoldo Acosta Sánchez y D. Antonio Luis Morell González, por depositar la confianza en nosotros para desarrollar este trabajo y por su inestimable ayuda en todo momento, aún en la distancia.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS:

ÍNDICE DE TABLAS

Universidad de La Laguna
Escuela Superior de Ingeniería y Tecnología
Sección de Ingeniería Industrial

1. Introducción.

1.1 Objetivo del Trabajo de Fin de Grado.

1.1.1 Resumen.

El objetivo del presente Trabajo de Fin de Grado es la aplicación e implementación de técnicas de control clásico y fraccionario sobre el sistema siguiente:

- Doble rotor: compuesto por un rotor principal (vertical) el cual se encarga de la sustentación aerodinámica y un rotor secundario (horizontal) encargado de estabilizarlo.

Por un lado, se diseñará un programa en el software matemático MATLAB con el fin de obtener los valores teóricos de las variables necesarias para implementar cada sistema de control, además de realizar la simulación y comparación entre los dos métodos aplicados.

Por otro lado, se llevará a cabo la implementación de los sistemas de control en las plantas reales existentes en el laboratorio, de manera que se obtengan los valores experimentales de dichas variables; pudiendo comprobar con ello la exactitud del programa diseñado y los métodos aplicados.

1.1.2 Abstract

The objective of the EOG project presented is the application and implementation of classical techniques of control and fractional control on the following system:

- Twin rotor: it is formed by a main rotor (vertical), in charge of the aerodynamic lift, and a secondary rotor (horizontal), responsible for stabilize it.

On one side, we designed a program using the mathematical software MATLAB to obtain the theoretical values of the variables needed to implement each control system. Furthermore, we have simulated and compared the results between the two methods applied.

Moreover, we are going to implement these control systems on the platforms available in the university laboratory. So that, we could obtain the experimental values of those variables, being able to verify the accuracy of the program designed and the methodology applied.

2. Control PID clásico.

El controlador PID (Proporcional, Integral y Derivativo) se trata de un controlador realimentado cuyo fin es hacer que el error en el estacionario entre el valor de referencia (la consigna) y la salida de la planta se haga cero de forma asintótica en el tiempo.

Es un tipo de control muy importante y útil dado que no es necesario conocer el modelo de la planta a controlar, aunque también siguen siendo igual de útiles para sistemas cuyo modelo de planta si es conocido. Aunque a veces no nos proporciona el control más óptimo de la planta.

$$C(s) = K_p \cdot \left(1 + \frac{1}{T_i \cdot s} + T_d \cdot s\right)$$

En el control PID clásico realizamos tres acciones de control expuestas a continuación.

2.1 Acción proporcional

La acción proporcional $K_c \cdot e(t)$: Con la acción proporcional sola, no llegaremos nunca al valor de consigna, únicamente nos acercaremos y tendremos un cierto valor de error.

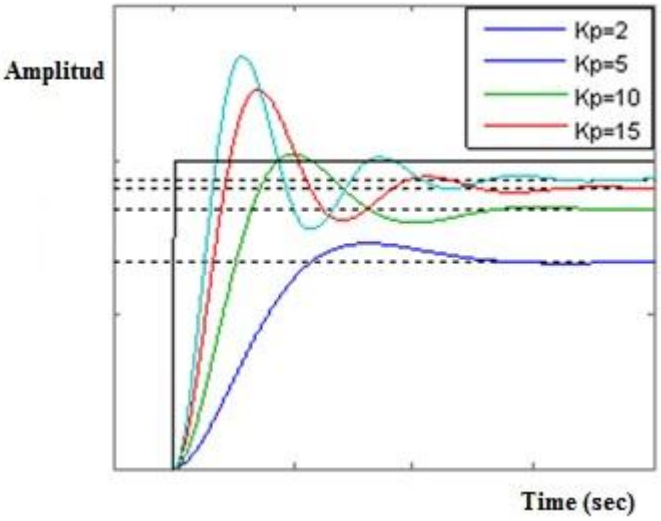

Figura 1. Efecto acción proporcional

2.2 Acción integral

La acción integral $\frac{K_c}{T_i} \cdot \int_0^t e(t)dt$: Elimina el error en el estacionario para poder lograr el valor de consigna. El error es integrado, por lo que tiene la función de promediarlo o sumarlo en el tiempo. Entonces se multiplica por una constante $K_i = \frac{1}{T_i}$, (K_i es la ganancia integral, T_i el tiempo integral).

La acción integral afecta negativamente al comportamiento dinámico del lazo, haciendo la respuesta más oscilatoria al provocar un acercamiento a las condiciones de inestabilidad.

La acción integral no debe aplicarse sola, debe ir acompañada como mínimo de la acción proporcional:

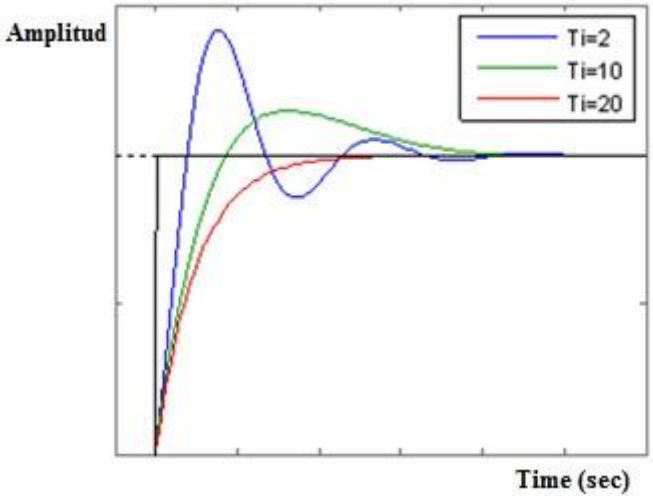

Figura 2. Efecto de la acción integral

2.3 Acción derivativa

La acción derivativa $K_c \cdot T_d \cdot \frac{de(t)}{dt}$: Obtiene como el producto de una ganancia denominada tiempo derivativo y la derivada del error en el tiempo. Actúa cuando hay un cambio en el valor absoluto del error.

Tiene un efecto positivo estabilizador sobre la dinámica del lazo de realimentación, es capaz de aportar una acción correctora con errores pequeños, *anticipándose* a la respuesta del sistema.

Cuando el sistema tiene perturbaciones o ruido (señales de baja amplitud y alta frecuencia), la acción derivativa no debe usarse debido a que el efecto que hará sobre estas señales será de amplificación, pudiendo llegar a provocar la inestabilidad del sistema.

La acción derivativa tampoco puede aplicarse sola, debe ir acompañada como mínimo de la acción proporcional:

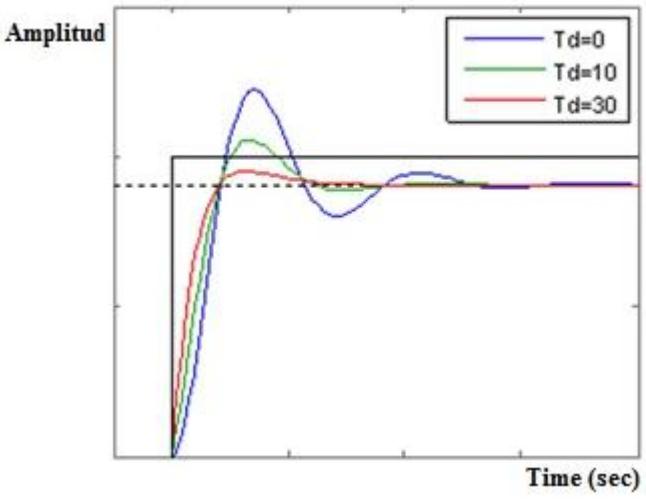

Figura 3. Efecto de la acción derivativa

Existen varias posibilidades con estas tres acciones de control:

- PI

$$u(t) = Kc \cdot \left[e(t) + \frac{1}{T_i} \cdot \int_0^t e(t)dt \right]$$

- PD

$$u(t) = Kc \cdot \left[e(t) + T_d \cdot \frac{de(t)}{dt} \right]$$

- PID

$$u(t) = Kc \cdot \left[e(t) + T_d \cdot \frac{de(t)}{dt} + \frac{1}{T_i} \cdot \int_0^t e(t)dt \right]$$

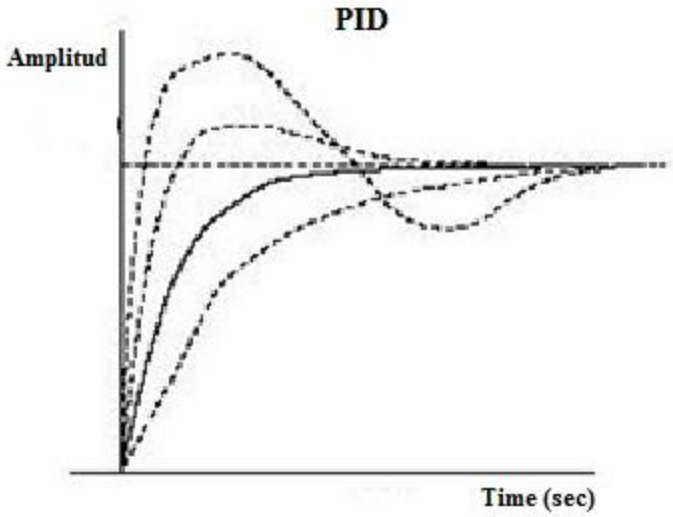

Figura 4. Efecto de un PID

2.4 Relación de parámetros del controlador con los sistemas a controlar

En la tabla 1 mostramos la combinación de todas las ventajas que podemos encontrar de las acciones del PID

Parámetro	Kp	Ti	Td
Tiempo de Crecimiento	Disminuye	Aumenta	Poco afectada
Sobrepaso	Aumenta	Disminuye	Disminuye
Estabilidad	Reduce	Disminuye	Aumenta
Error en estacionario	No eliminado	Eliminado	No eliminado

Tabla 1. Relación de parámetros y características de las acciones de control

3. Control fraccionario

El control fraccionario propone el uso de operadores y sistemas fraccionarios con el objetivo de introducir más grados de libertad con los que poder trabajar y así ajustar mejor el comportamiento de un sistema. Además de aportarnos más flexibilidad, el añadir un parámetro de fraccionario, me permite jugar con las pendientes del diagrama de BODE y que su respuesta sea más suave.

3.1 Estudio temporal:

$$\int_0^t f(x)dx$$

$$\int_0^t \int_0^x f(y)\,dy\,dx = \int_0^t (t-y)\,f(y)\,dy$$

$$\int_0^t \int_0^x \int_0^z f(y)dy\,dz\,dx = \frac{1}{2}\int_0^t (t-y)^2\ f(y)\,dy$$

De forma genérica:

Integral de orden n:

$$I^n f(t) = \int_0^t \frac{(t-y)^{n-1}\,f(y)}{(n-1)!}\,dy = \frac{1}{(n-1)!}\int_0^t (t-y)^{n-1}\,f(y)\,dy$$

Nota: en la formula anterior el tener factoriales (números enteros) supone un problema a la hora de trabajar con integrales fraccionarias. Por ello hacemos lo siguiente:

Función Gamma: $\Gamma(n+1) = n!$

3.1.1 Riemann-Liouville (convolución)

$$I^n f(t) = \frac{1}{\Gamma(n)} \int_0^t (t-y)^{n-1} f(y) dy$$

$$\Gamma(\alpha+1) = \alpha\Gamma(\alpha) \rightarrow \Gamma(\alpha) = \frac{\Gamma(\alpha+1)}{\alpha}$$

$$\Gamma(\alpha) = \frac{\Gamma(\alpha+2)}{(\alpha+1)\alpha}$$

Como tienes la definición de integral fraccionaria también tenemos una derivada fraccionaria.

- Caso ordinario: $D^\alpha I^\alpha f(t) = f(t) \rightarrow$ Este caso siempre funciona.
- Caso fraccionario: $D^\alpha = I^{-\alpha} \rightarrow I^{-\alpha} I^\alpha f(t) = f(t)$

  $D \cdot I \cdot I^{-\alpha} I^\alpha = D I^{(1-\alpha)} I^\alpha f(t) = D^\alpha I^\alpha f(t) = f(t)$

Para las derivadas fraccionarias tenemos dos definiciones:

- 1ª Definición de *Grünwald-Letnikov*

$$D^\alpha f(t) = D \cdot I^{(1-\alpha)} f(t) = \frac{d}{dt}\left[\frac{1}{\Gamma(1-\alpha)} \int_0^t (t-y)^{1-\alpha-1} f(y)\, dy\right]$$

- 2ª Definición de *Caputo*

$$D^\alpha f(t) = I^{(1-\alpha)} D = \frac{1}{\Gamma(1-\alpha)} \int_0^t f(y)\, (t-y)^{1-\alpha-1} dy$$

La mejor definición es la de *Caputo*, porque se basa en la resolución integral y ésta siempre mejora una función.

3.2 Acciones de control.

Estudiando las acciones de control del tipo Ks^{μ} $\mu \in [-1, 1]$, observaremos que dos efectos dan la acción integral y la derivativa a un sistema de orden fraccionario.

Antes de mirar cada acción de control, debemos recordar que:

$$\mu = 0 \rightarrow Ks^0 = K \rightarrow Acción\ proporcional$$

$$\mu = 1 \rightarrow Ks^1 = Ks \rightarrow Acción\ derivativa$$

$$\mu = -1 \rightarrow Ks^{-1} = \frac{K}{s} \rightarrow Acción\ integral$$

La fórmula de un controlador fraccionario se compone de la siguiente manera:

$$C(s) = K_p \cdot \left(1 + \frac{1}{T_i \cdot s^{\lambda}} + T_d \cdot s^{\mu}\right)$$

3.2.1 Acción integral:

Como ya sabemos la acción integral disminuye el error en el estacionario, permitiéndonos llegar al valor de consigna. Además, hace que el sistema vaya más lento y que disminuya la estabilidad relativa de éste. A continuación mostraremos una gráfica para los distintos valores del orden fraccionario μ.

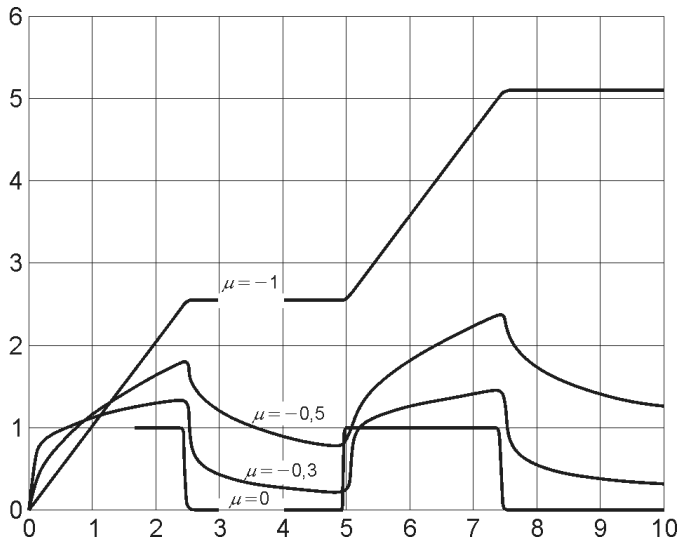

Figura 5. Efectos de la acción integral sobre una onda cuadrada según el orden de integración

Para el **dominio temporal**, se observa que introduciendo parámetros fraccionarios, la acción de control es creciente, por lo que se elimina el error en el estacionario. Y también, se observa un decrecimiento cuando el error tiene a cero, debido a esto, se produce una menor inestabilidad del sistema.

En el **dominio de la frecuencia**, el uso de coeficientes fraccionarios introduce una variación constante de las pendientes de la curva de la magnitud, pudiendo así hacer un ajuste aún más preciso de los parámetros del sistema. Lo mismo ocurre con la fase.

3.2.2 Acción derivativa:

La acción derivativa tiene la capacidad de evitar los sobrepasamientos, suavizando así la respuesta del sistema. Además tiende a enfatizar los efectos de los ruidos y perturbaciones de alta frecuencia. A continuación mostraremos una gráfica para los distintos valores del orden fraccionario μ.

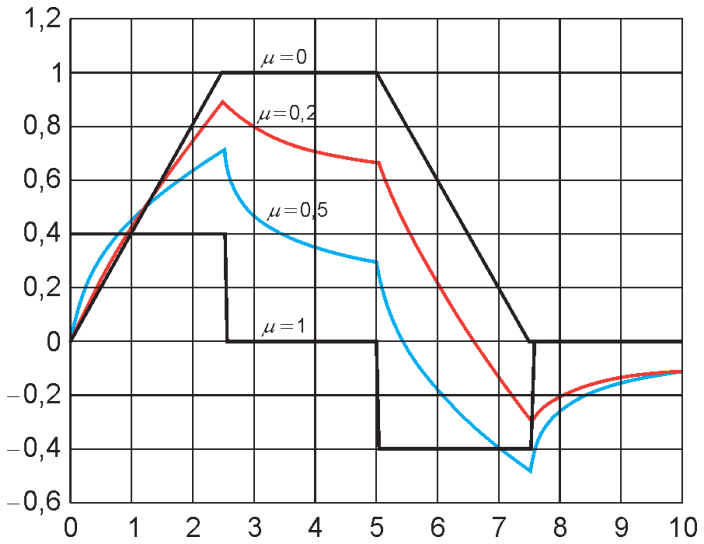

Figura 6. Efectos de la acción derivativa sobre una función trapezoidal según el orden de derivación.

Observando dicha gráfica, se llega a una conclusión paralela al caso de la acción integral tanto para el dominio temporal como el de las frecuencias.

Principalmente, dada su practicidad y claridad, nos centraremos en el dominio de las frecuencias para calcular todos los parámetros y sintonización del PID fraccionario que necesitaremos en nuestros sistemas.

4. Cálculo teórico para la sintonización del PID clásico

El objetivo es obtener los parámetros de sintonización del PID (constante proporcional: K_p, parámetro integral: T_i y el parámetro derivativo: T_d) a partir de una serie de especificaciones de diseño: margen de fase ($\emptyset_m$), frecuencia de cruce (ω_c) y el margen de ganancia (M_g).

La función de transferencia del sistema expresada de forma general será:

$$(4.1)\quad G(s) = \frac{N(S)}{D(s)} = \frac{a \cdot s + b}{c \cdot s^2 + d \cdot s + e} \cdot e^{-s\tau}$$

Donde τ es nuestro retardo, y lo especificaremos nosotros y $s = j\omega$.

Nosotros lo expresaremos de la siguiente manera:

$$(4.2)\quad G(s) = G(s)' \cdot e^{-s\tau}$$

Donde G(S)' será nuestra función de transferencia sin el retardo.

Mientras que la expresión general de nuestro controlador para el PID clásico es:

$$(4.3)\quad R(s) = K_p \cdot (1 + K_i \cdot S^{-1} + T_d \cdot S)$$

Y sabemos que está compuesta por una parte real y otra imaginaria:

$$(4.4)\quad R(j\omega) = R_R(\omega) + jR_I(\omega)$$

Sustituyendo $s = j\omega$ en la expresión (4.4) tenemos:

$$(4.4.1)\quad R(j\omega) = K_p \cdot \left(1 + \frac{K_i}{j\omega} + T_d \cdot j\omega\right) = K_p \cdot \left(1 - \frac{K_i\, j}{\omega} + T_d \cdot j\omega\right)$$

$$= K_p + j \cdot \left(T_d \cdot \omega \cdot K_p - \frac{K_i}{\omega} \cdot K_p\right) \quad ;$$

$$R_R(\omega) = K_p \quad ; \quad R_I(\omega) = K_p \cdot \left(T_p\omega - \frac{K_i}{\omega}\right)$$

Para la sintonización de este controlador se deben seguir las siguientes especificaciones de diseño:

1) $|G'(j\omega_c)| \cdot |R(j\omega_c)| = 1$
2) $fase: \arg\left(G'(j\omega_c)\right) - \tau\omega_c + \arg\left(R(j\omega_c)\right) + \pi = \emptyset_m$
3) $\left|G'(j\omega_g)\right| \cdot \left|R(j\omega_g)\right| = \frac{1}{M_g}$
4) $fase: \arg\left(G'(j\omega_g)\right) - \tau\omega_g + \arg\left(R(j\omega_g)\right) + \pi = 0$

Para conseguir el sistema de ecuaciones del que obtendremos los parámetros del controlador, hay que operar con esas especificaciones de diseño:

- De la especificación 1) sacamos:

$$(4.5)\ |G'(j\omega_c)| \cdot |R(j\omega_c| = 1 \rightarrow |R(j\omega_c| = \frac{1}{|G'(j\omega_c)|}$$

- De la 2)

$$(4.6)\ \arg\left(G'(j\omega_c)\right) + arg\left(R(j\omega_c)\right) - \tau\omega_c + \pi = \emptyset_m \rightarrow$$
$$\arg\left(R(j\omega_c)\right) = \emptyset_m - \pi + \tau\omega_c - arg(G'(j\omega_c))$$

Descomponiendo la parte real e imaginaria:

$$(4.7)\ R_R(\omega_c) = |R(j\omega_c)| \cdot \cos\left(arg\left(\ R(j\omega_c)\right)\right) \cdot \frac{1}{|G'(j\omega_c)|}$$
$$\cdot \cos\left(\emptyset_m + \tau\omega_c - \pi - arg\left(G'(j\omega_c)\right)\right)$$

$$(4.8)\ R_I(\omega_c) = |R(j\omega_c)| \cdot sen\left(arg\left(R(j\omega_c)\right)\right) \cdot \frac{1}{|G'(j\omega_c)|}$$
$$\cdot sen\left(\emptyset_m + \tau\omega_c - \pi - arg\left(G'(j\omega_c)\right)\right)$$

- De la especificación 3) sacamos:

$$(4.9)\ \left|G'(j\omega_g)\right| \cdot \left|R(j\omega_g)\right| = \frac{1}{M_g} \rightarrow \left|R(j\omega_g)\right| = \frac{1}{M_g \cdot |G'(j\omega_g)|}$$

- Y de la 4)

$$(4.10)\ \arg G'\left(j\omega_g\right) + argR(j\omega_g) - \tau\omega_g + \pi = 0 \rightarrow$$
$$\arg R(j\omega_g) = \tau\omega_g - \pi - arg\left(G'(j\omega_g)\right)$$

Descomponiendo la parte real e imaginaria:

$$(4.11)\ R_R(\omega_g) = |R(j\omega_g)| \cdot \cos(\arg R(j\omega_g)) = \frac{1}{M_g \cdot |G'(j\omega_g)|} \cdot \cos\left(\tau\omega_g - \pi - argG'(j\omega_g)\right)$$

$$(4.12)\ R_I(\omega_g) = |R(j\omega_g)| \cdot sen\left(argR(j\omega_g)\right) = \frac{1}{M_g|G'(j\omega_g)|} \cdot sen(\tau\omega_g - \pi - argG'(j\omega_g))$$

Considerando la ecuación general de un controlador PID y el sistema sobre el que trabajamos, obtendremos una ecuación más compacta para poder calcular los parámetros:

Nuestra matriz A se compone:

$$(4.13)\quad A = \begin{pmatrix} R_R(\omega_c) \\ R_I(\omega_c) \\ R_R(\omega_g) \\ R_I(\omega_g) \end{pmatrix} = \begin{pmatrix} K_p \\ K_p \cdot T_d \cdot \omega_c - \dfrac{K_p \cdot K_i}{\omega_c} \\ K_p \\ K_p \cdot T_d \cdot \omega_g - \dfrac{K_p \cdot K_i}{\omega_g} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & \omega_c & -\dfrac{1}{\omega_c} \\ 1 & 0 & 0 \\ 0 & \omega_g & -\dfrac{1}{\omega_g} \end{pmatrix} \cdot \begin{pmatrix} K_p \\ K_p \cdot T_d \\ K_p \cdot K_i \end{pmatrix}$$

Mientras que la matriz B, sacada a partir de las especificaciones de diseño nos queda:

$$(4.14)\quad B = -\begin{pmatrix} \cos(\emptyset_m + \tau\omega_c - < G'(j\omega_c))/|G'(j\omega_c)| \\ \sin(\emptyset_m + \tau\omega_c - < G'(j\omega_c))/|G'(j\omega_c)| \\ \cos(\tau\omega_g - < G(j\omega_g))/(Mg \cdot |G'(j\omega_g)|) \\ \sin(\tau\omega_g - < G'(j\omega_g))/(Mg \cdot |G'(j\omega_g)|) \end{pmatrix}$$

Por lo que el sistema final sobre el que vamos a trabajar la sintonización queda de la forma: A·X=B

$$(4.15)\quad \begin{pmatrix} 1 & 0 & 1 \\ 1 & \omega_c & -\frac{1}{\omega_c} \\ 1 & 0 & 0 \\ 0 & \omega_g & -\frac{1}{\omega_g} \end{pmatrix} \cdot \begin{pmatrix} K_p \\ K_p \cdot T_d \\ K_p \cdot K_i \end{pmatrix}$$

$$= -\begin{pmatrix} \cos(\emptyset_m + \tau\omega_c - < G'(j\omega_c))/|G'(j\omega_c)| \\ \sin(\emptyset_m + \tau\omega_c - < G'(j\omega_c))/|G'(j\omega_c)| \\ \cos(\tau\omega_g - < G(j\omega_g))/(Mg \cdot |G'(j\omega_g)|) \\ \sin(\tau\omega_g - < G'(j\omega_g))/(Mg \cdot |G'(j\omega_g)|) \end{pmatrix}$$

Siendo X la matriz donde están los parámetros que buscamos.

Para que esta ecuación se pueda resolver, el rango de A y B debe ser igual a 3:

$$rango((A\ B)) = 3$$

Tenemos como incógnita, a parte de los parámetros del controlador que queremos, la frecuencia de ganancia ω_g , por lo que para encontrarla debemos buscar su valor realizando los cálculos con las matrices A y B, conociendo lo siguiente:

Dado que el rango de A con B tiene que ser igual a 3 para que el sistema tenga solución, y que $\omega_c \neq \omega_g$, la única forma de que la condición de rango se cumple es que la primera y tercera fila de la ecuación (4.14) sean iguales, obteniendo así la siguiente ecuación:

$$\cos(\emptyset_m + \tau\omega_c - < G'(j\omega_c))/|G'(j\omega_c)| = \cos(\tau\omega_g - < G(j\omega_g))/(Mg \cdot |G'(j\omega_g)|)$$

También sabemos que $\omega_c < \omega_g$, y esto queda demostrado siguiendo el criterio de estabilidad de BODE.

Por lo que simplificadamente tendremos:

$$(4.16)\quad g(\omega_c) = h(\omega_g)$$

Para encontrar el valor de ω_g hay muchas opciones matemáticas, pero se ha tomado la decisión de realizarlo de forma gráfica, y a partir de ahí elegir la ω_g que mejor se adecue a la situación. Aquí se muestra una imagen de cómo se selecciona dicho valor:

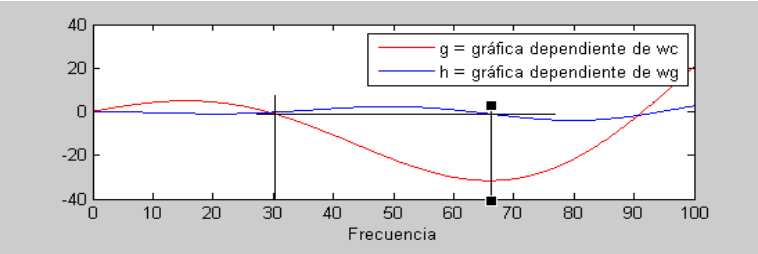

Figura 7. Elección de la frecuencia deseada

Se traza una línea vertical sobre la frecuencia ω_c que se desea (en esta gráfica es de 30), en el corte con la gráfica dependiente de ω_g, se traza una línea horizontal hasta que vuelva a cortar con otro punto de la gráfica, la frecuencia en la que se da en ese nuevo corte es la frecuencia ω_g.

Una vez hallada la frecuencia correspondiente y calculadas las matrices A y B, se procede al cálculo de los parámetros del controlador, quedando la ecuación de la siguiente manera:

$$(4.17) \quad X = -(A^T \cdot A)^{-1} \cdot A^T \cdot B$$

Todos estos cálculos y pruebas, vienen recogidos en un programa que se encuentra explicado en el apartado 11.1.2.

5. Cálculo teórico para la sintonización del PID fraccionario

El objetivo es obtener los parámetros de sintonización del PID (constante proporcional: K_p, parámetro integral: T_i y el parámetro derivativo: T_d) a partir de una serie de especificaciones de diseño: margen de fase ($\emptyset_m$), frecuencia de cruce (ω_c) y el margen de ganancia (M_g).

La función de transferencia del sistema expresada de forma general será:

$$G(s) = \frac{N(S)}{D(s)} = \frac{a \cdot s + b}{c \cdot s^2 + d \cdot s + e} \cdot e^{-s\tau}$$

Donde τ es nuestro retardo, y lo especificaremos nosotros y $s = j\omega$.

Nosotros lo expresaremos de la siguiente manera:

$$G(s) = G(s)' \cdot e^{-s\tau}$$

Donde G(S)' será nuestra función de transferencia sin el retardo.

Mientras que la expresión general de nuestro controlador para el PID fraccionario es:

$$R(s) = K_p \cdot \left(1 + K_i \cdot s^{-\lambda} + T_d \cdot s^{\mu}\right)$$

Como $s = j\omega$:

$$R(j\omega) = R_R(\omega) + jR_I(\omega)$$

$$R(j\omega) = K_p \cdot \left(1 + K_i \cdot (j\omega)^{-\lambda} + T_d \cdot (j\omega)^{\mu}\right)$$

En forma polar:

$$j\omega = \omega e^{j\cdot(\pi/2)} = \left[e^{j\cdot(\pi/2)} = \cos\left(\frac{\pi}{2}\right) + j \cdot \sin\left(\frac{\pi}{2}\right)\right] = \omega \cdot \cos\left(\frac{\pi}{2}\right) + j\omega \cdot \sin\left(\frac{\pi}{2}\right) \rightarrow$$

$$R(j\omega) = K_p \cdot \left(1 + K_i \cdot \left(\omega e^{j\cdot(\pi/2)}\right)^{-\lambda} + T_d \cdot (\omega e^{j\cdot(\pi/2)})^{\mu}\right) =$$

$$= K_p \cdot \left(1 + K_i \cdot \omega^{-\lambda} e^{-j\cdot(\pi/2)\cdot\lambda} + T_d \cdot \omega^{\mu} e^{j\cdot(\pi/2)\cdot\mu}\right) =$$

$$= K_p \cdot \left(1 + K_i \cdot \omega^{-\lambda} \cdot \left(\cos\left(-\frac{\pi}{2}\lambda\right) + j \cdot \sin\left(-\frac{\pi}{2}\lambda\right)\right) + T_d \cdot \omega^{\mu} \left(\cos\left(\frac{\pi}{2}\mu\right) + j \cdot \sin\left(\frac{\pi}{2}\mu\right)\right)\right) \rightarrow$$

$$R(j\omega) = K_p \cdot \left(1 + K_i \cdot \omega^{-\lambda} \cdot \left(\cos\left(\frac{\pi}{2}\lambda\right) - j \cdot \sin\left(\frac{\pi}{2}\lambda\right)\right) + T_d \cdot \omega^{\mu} \left(\cos\left(\frac{\pi}{2}\mu\right) + j \cdot \sin\left(\frac{\pi}{2}\mu\right)\right)\right)$$

De ahí podemos obtener la parte real y la parte imaginaria:

$$R_R(\omega) \equiv \left(K_p + K_p \cdot K_i \cdot \omega^{-\lambda} \cdot \cos\left(\frac{\pi}{2}\lambda\right) + K_p \cdot T_d \cdot \omega^{\mu} \cdot \cos\left(\frac{\pi}{2}\mu\right)\right)$$

$$R_I(\omega) \equiv \left(-K_p \cdot K_i \cdot \omega^{-\lambda} \cdot \sin\left(\frac{\pi}{2}\lambda\right) + K_p \cdot T_d \cdot \omega^{\mu} \cdot \sin\left(\frac{\pi}{2}\mu\right)\right)$$

Para la sintonización de este controlador se deben seguir las siguientes especificaciones de diseño:

5) $|G'(j\omega_c)| \cdot |R(j\omega_c)| = 1$

6) $fase: \arg\left(G'(j\omega_c)\right) - \tau\omega_c + \arg\left(R(j\omega_c)\right) + \pi = \emptyset_m$

7) $\left|G'(j\omega_g)\right| \cdot \left|R(j\omega_g)\right| = \frac{1}{M_g}$

8) $fase: \arg\left(G'(j\omega_g)\right) - \tau\omega_g + \arg\left(R(j\omega_g)\right) + \pi = 0$

Para conseguir el sistema de ecuaciones del que obtendremos los parámetros del controlador, hay que operar con esas especificaciones de diseño:

- De la especificación 1) sacamos:

$$|G'(j\omega_c)| \cdot |R(j\omega_c| = 1 \rightarrow |R(j\omega_c| = \frac{1}{|G'(j\omega_c)|}$$

- De la 2)

$$\arg\left(G'(j\omega_c)\right) + arg\left(R(j\omega_c)\right) - \tau\omega_c + \pi = \emptyset_m \rightarrow$$

$$\arg\left(R(j\omega_c)\right) = \emptyset_m - \pi + \tau\omega_c - arg(G'(j\omega_c))$$

Descomponiendo la parte real e imaginaria:

$$R_R(\omega_c) = |R(j\omega_c)| \cdot \cos\left(arg\left(R(j\omega_c)\right)\right) \cdot \frac{1}{|G'(j\omega_c)|} \cdot \cos\left(\emptyset_m + \tau\omega_c - \pi - arg\left(G'(j\omega_c)\right)\right)$$

$$R_I(\omega_c) = |R(j\omega_c)| \cdot sen\left(arg\left(R(j\omega_c)\right)\right) \cdot \frac{1}{|G'(j\omega_c)|} \cdot sen\left(\emptyset_m + \tau\omega_c - \pi - arg\left(G'(j\omega_c)\right)\right)$$

- De la especificación 3) sacamos:

$$\left|G'(j\omega_g)\right| \cdot \left|R(j\omega_g)\right| = \frac{1}{M_g} \rightarrow \left|R(j\omega_g)\right| = \frac{1}{M_g \cdot |G'(j\omega_g)|}$$

- Y de la 4)

$$\arg G'(j\omega_g) + argR(j\omega_g) - \tau\omega_g + \pi = 0 \rightarrow$$

$$\arg R(j\omega_g) = \tau\omega_g - \pi - arg\left(G'(j\omega_g)\right)$$

Descomponiendo la parte real e imaginaria:

$$R_R(\omega_g) = \left|R(j\omega_g)\right| \cdot \cos(\arg R(j\omega_g)) = \frac{1}{M_g \cdot \left|G'(j\omega_g)\right|} \cdot \cos\left(\tau\omega_g - \pi - argG'(j\omega_g)\right)$$

$$R_I(\omega_g) = \left|R(j\omega_g)\right| \cdot sen\left(argR(j\omega_g)\right) = \frac{1}{M_g\left|G'(j\omega_g)\right|} \cdot sen\left(\tau\omega_g - \pi - argG'(j\omega_g)\right)$$

Considerando la ecuación general de un controlador PID fraccionario y el sistema sobre el que trabajamos, obtendremos una ecuación más compacta para poder calcular los parámetros:

Nuestra matriz A se compone:

$$A = \begin{pmatrix} R_R(\omega_c) \\ R_I(\omega_c) \\ R_R(\omega_g) \\ R_I(\omega_g) \end{pmatrix}$$

Las matrices X y B siguen siendo las mismas que para el caso del PID clásico.

$$A = \begin{pmatrix} K_p + K_p \cdot K_i \cdot {\omega_c}^{-\lambda} \cdot \cos\left(\frac{\pi}{2}\lambda\right) + K_p \cdot T_d \cdot {\omega_c}^{\mu} \cdot \cos\left(\frac{\pi}{2}\mu\right) \\ -K_p \cdot K_i \cdot {\omega_c}^{-\lambda} \cdot \sin\left(\frac{\pi}{2}\lambda\right) + K_p \cdot T_d \cdot {\omega_c}^{\mu} \cdot \sin\left(\frac{\pi}{2}\mu\right) \\ K_p + K_p \cdot K_i \cdot {\omega_g}^{-\lambda} \cdot \cos\left(\frac{\pi}{2}\lambda\right) + K_p \cdot T_d \cdot {\omega_g}^{\mu} \cdot \cos\left(\frac{\pi}{2}\mu\right) \\ -K_p \cdot K_i \cdot {\omega_g}^{-\lambda} \cdot \sin\left(\frac{\pi}{2}\lambda\right) + K_p \cdot T_d \cdot {\omega_g}^{\mu} \cdot \sin\left(\frac{\pi}{2}\mu\right) \end{pmatrix}$$

$$A \cdot X = B \rightarrow$$

$$\begin{pmatrix} 1 & {\omega_c}^{-\lambda} \cdot \cos\left(\frac{\pi}{2}\lambda\right) & {\omega_c}^{\mu} \cdot \cos\left(\frac{\pi}{2}\mu\right) \\ 0 & -{\omega_c}^{-\lambda} \cdot \sin\left(\frac{\pi}{2}\lambda\right) & {\omega_c}^{\mu} \cdot \sin\left(\frac{\pi}{2}\mu\right) \\ 1 & {\omega_g}^{-\lambda} \cdot \cos\left(\frac{\pi}{2}\lambda\right) & {\omega_g}^{\mu} \cdot \cos\left(\frac{\pi}{2}\mu\right) \\ 0 & -{\omega_g}^{-\lambda} \cdot \sin\left(\frac{\pi}{2}\lambda\right) & {\omega_g}^{\mu} \cdot \sin\left(\frac{\pi}{2}\mu\right) \end{pmatrix} \cdot \begin{pmatrix} K_p \\ K_p \cdot K_i \\ K_p \cdot T_d \end{pmatrix} = - \begin{pmatrix} \cos(\emptyset_m + \tau\omega_c - < G'(j\omega_c)) / |G'(j\omega_c)| \\ \sin(\emptyset_m + \tau\omega_c - < G'(j\omega_c)) / |G'(j\omega_c)| \\ \cos(\tau\omega_g - < G(j\omega_g)) / (Mg \cdot |G'(j\omega_g)|) \\ \sin(\tau\omega_g - < G'(j\omega_g)) / (Mg \cdot |G'(j\omega_g)|) \end{pmatrix}$$

Para este resultado final procederemos a calcular los parámetros del controlador PID de dos formas distintas:

5.1 Cálculo sin la acción integral y los coeficientes fraccionarios iguales.

Suponiendo que Ki desaparece (Ki=0), que ω_g es un parámetro ya especificado al principio (es decir, es conocido) y que $\mu = \lambda$

Con esta suposición la matriz A nos queda de la siguiente manera:

$$(5.1.1)\quad A = \begin{pmatrix} 1 & {\omega_c}^{\mu} \cdot \cos\left(\frac{\pi}{2}\mu\right) \\ 0 & {\omega_c}^{\mu} \cdot \sin\left(\frac{\pi}{2}\mu\right) \\ 1 & {\omega_g}^{\mu} \cdot \cos\left(\frac{\pi}{2}\mu\right) \\ 0 & {\omega_g}^{\mu} \cdot \sin\left(\frac{\pi}{2}\mu\right) \end{pmatrix}$$

Para que la ecuación se pueda resolver, se tiene que cumplir que el rango

((A B))=2

$$(5.1.1.1)\quad \begin{vmatrix} 1 & {\omega_c}^{\mu} \cdot \cos\left(\frac{\pi}{2}\mu\right) \\ 0 & {\omega_c}^{\mu} \cdot \sin\left(\frac{\pi}{2}\mu\right) \end{vmatrix} = {\omega_c}^{\mu} \cdot \sin\left(\frac{\pi}{2}\mu\right)$$

$$(5.1.1.2)\quad \begin{vmatrix} 0 & {\omega_c}^{\mu} \cdot \sin\left(\frac{\pi}{2}\mu\right) \\ 1 & {\omega_g}^{\mu} \cdot \cos\left(\frac{\pi}{2}\mu\right) \end{vmatrix} = -{\omega_c}^{\mu} \cdot \sin\left(\frac{\pi}{2}\mu\right)$$

$$(5.1.1.3)\quad \begin{vmatrix} 1 & {\omega_g}^{\mu} \cdot \cos\left(\frac{\pi}{2}\mu\right) \\ 0 & {\omega_g}^{\mu} \cdot \sin\left(\frac{\pi}{2}\mu\right) \end{vmatrix} = {\omega_g}^{\mu} \cdot \sin\left(\frac{\pi}{2}\mu\right)$$

Observando dichos resultados vemos que no hay ningún determinante nulo, por lo que el rango de la matriz es 2, es decir, tiene solución.

$$(5.1.2)\quad \begin{pmatrix} 1 & {\omega_c}^{\mu} \cdot \cos\left(\frac{\pi}{2}\mu\right) \\ 0 & {\omega_c}^{\mu} \cdot \sin\left(\frac{\pi}{2}\mu\right) \\ 1 & {\omega_g}^{\mu} \cdot \cos\left(\frac{\pi}{2}\mu\right) \\ 0 & {\omega_g}^{\mu} \cdot \sin\left(\frac{\pi}{2}\mu\right) \end{pmatrix} \cdot \begin{pmatrix} K_p \\ K_p \cdot T_d \end{pmatrix}$$

$$= -\begin{pmatrix} \cos(\emptyset_m + \tau\omega_c - < G'(j\omega_c))/|G'(j\omega_c)| \\ \sin(\emptyset_m + \tau\omega_c - < G'(j\omega_c))/|G'(j\omega_c)| \\ \cos(\tau\omega_g - < G(j\omega_g))/(Mg \cdot |G'(j\omega_g)|) \\ \sin(\tau\omega_g - < G'(j\omega_g))/(Mg \cdot |G'(j\omega_g)|) \end{pmatrix}$$

Operando las matrices del sistema de ecuación del apartado (5.1.2), obtenemos el siguiente sistema de ecuaciones:

a) $K_p + K_p \cdot T_d \cdot {\omega_c}^{\mu} \cdot \cos\left(\frac{\pi}{2}\mu\right) = -\cos(\emptyset_m + \tau\omega_c - < G'(j\omega_c))/|G'(j\omega_c)|$

b) $K_p \cdot T_d \cdot {\omega_c}^{\mu} \cdot \sin\left(\frac{\pi}{2}\mu\right) = -\sin(\emptyset_m + \tau\omega_c - < G'(j\omega_c))/|G'(j\omega_c)|$

c) $K_p + K_p \cdot T_d \cdot {\omega_g}^{\mu} \cdot \cos\left(\frac{\pi}{2}\mu\right) = -\cos(\tau\omega_g - < G(j\omega_g))/(Mg \cdot |G'(j\omega_g)|$

d) $K_p \cdot T_d \cdot {\omega_g}^{\mu} \cdot \sin\left(\frac{\pi}{2}\mu\right) = -\sin(\tau\omega_g - < G'(j\omega_g))/(Mg \cdot |G'(j\omega_g)|)$

Despejando primero el apartado b), tenemos:

$$K_p \cdot T_d = \frac{-\sin(\emptyset_m + \tau\omega_c - < G'(j\omega_c))/|G'(j\omega_c)|}{\sin\left(\frac{\pi}{2}\mu\right) \cdot {\omega_c}^{\mu}}$$

Ahora sustituimos el resultado en el apartado a), nos queda:

$$K_p + {\omega_c}^{\mu} \cdot \frac{-\sin(\emptyset_m + \tau\omega_c - < G'(j\omega_c))/|G'(j\omega_c)|}{\sin\left(\frac{\pi}{2}\mu\right) \cdot {\omega_c}^{\mu}}$$

$$\cdot \cos\left(\frac{\pi}{2}\mu\right) = -\cos(\emptyset_m + \tau\omega_c - < G'(j\omega_c))/|G'(j\omega_c)|$$

$$\mathbf{(5.1.3)}\quad \boldsymbol{K_p} - \mathbf{sin}(\emptyset_{\boldsymbol{m}} + \boldsymbol{\tau\omega_c} - < \boldsymbol{G'(j\omega_c)})/|\boldsymbol{G'(j\omega_c)}| \cdot \boldsymbol{cot}\left(\frac{\boldsymbol{\pi}}{\mathbf{2}}\boldsymbol{\mu}\right)$$
$$= -\mathbf{cos}(\emptyset_{\boldsymbol{m}} + \boldsymbol{\tau\omega_c} - < \boldsymbol{G'(j\omega_c)})/|\boldsymbol{G'(j\omega_c)}|$$

Repetimos el mismo paso para c y d. Primero despejando en d):

$$K_p \cdot T_d = \frac{-\sin(\tau\omega_g - < G'(j\omega_g))/(Mg \cdot |G'(j\omega_g)|)}{\sin\left(\frac{\pi}{2}\mu\right) \cdot {\omega_g}^{\mu}}$$

Y sustituimos en c):

$$K_p + {\omega_g}^{\mu} \cdot \frac{-\sin(\tau\omega_g - < G'(j\omega_g))/(Mg \cdot |G'(j\omega_g)|)}{\sin\left(\frac{\pi}{2}\mu\right) \cdot {\omega_g}^{\mu}} \cdot$$

$$\cdot \cos\left(\frac{\pi}{2}\mu\right) = -\cos(\tau\omega_g - < G(j\omega_g))/(Mg \cdot |G'(j\omega_g)|$$

$$\mathbf{(5.1.4)}\ \boldsymbol{K_p - \sin(\tau\omega_g - < G'(j\omega_g))/(Mg \cdot |G'(j\omega_g)|) \cdot \cot\left(\frac{\pi}{2}\mu\right)}$$
$$\boldsymbol{= -\cos(\tau\omega_g - < G(j\omega_g))/(Mg \cdot |G'(j\omega_g)|}$$

Dado que Kp=Kp, igualamos las ecuaciones (5.1.3) y (5.1.4):

$$-\frac{\cos(\emptyset_m + \tau\omega_c - < G'(j\omega_c))}{|G'(j\omega_c)|} + \frac{\sin(\emptyset_m + \tau\omega_c - < G'(j\omega_c))}{|G'(j\omega_c)|} \cdot cot\left(\frac{\pi}{2}\mu\right)$$
$$= -\frac{\cos\left(\tau\omega_g - < G(j\omega_g)\right)}{Mg \cdot |G'(j\omega_g)|} + \frac{\sin(\tau\omega_g - < G'(j\omega_g))}{(Mg \cdot |G'(j\omega_g)|} \cdot cot\left(\frac{\pi}{2}\mu\right)$$

$$\frac{\sin(\emptyset_m + \tau\omega_c - < G'(j\omega_c))}{|G'(j\omega_c)|} \cdot cot\left(\frac{\pi}{2}\mu\right) - \frac{\sin(\tau\omega_g - < G'^{(j\omega_g)})}{(Mg \cdot |G'(j\omega_g)|} \cdot cot\left(\frac{\pi}{2}\mu\right)$$
$$= -\frac{\cos\left(\tau\omega_g - < G(j\omega_g)\right)}{Mg \cdot |G'(j\omega_g)|} + \frac{\cos(\emptyset_m + \tau\omega_c - < G'(j\omega_c))}{|G'(j\omega_c)|}$$

$$cot\left(\frac{\pi}{2}\mu\right) = \frac{-\frac{\cos\left(\tau\omega_g - < G(j\omega_g)\right)}{Mg \cdot |G'(j\omega_g)|} + \frac{\cos(\emptyset_m + \tau\omega_c - < G'(j\omega_c))}{|G'(j\omega_c)|}}{\frac{\sin(\emptyset_m + \tau\omega_c - < G'(j\omega_c))}{|G'(j\omega_c)|} - \frac{\sin(\tau\omega_g - < G'^{(j\omega_g)})}{(Mg \cdot |G'(j\omega_g)|}}$$

$$\frac{-\frac{\cos\left(\tau\omega_g - < G(j\omega_g)\right)}{Mg \cdot |G'(j\omega_g)|} + \frac{\cos(\emptyset_m + \tau\omega_c - < G'(j\omega_c))}{|G'(j\omega_c)|}}{\frac{\sin(\emptyset_m + \tau\omega_c - < G'(j\omega_c))}{|G'(j\omega_c)|} - \frac{\sin(\tau\omega_g - < G'^{(j\omega_g)})}{(Mg \cdot |G'(j\omega_g)|}} = H$$

$$cot\left(\frac{\pi}{2}\mu\right) = H \rightarrow \quad \frac{\pi}{2}\cdot\mu = cot^{-1}(H)$$

$$(\mathbf{5.1.5})\ \boldsymbol{\mu = \frac{2}{\pi}\cdot cot^{-1}(H)}$$

Para ponerlo en práctica, realizamos un código de Matlab (referenciado en el apartado 11.1.3). Una vez introducidos las especificaciones del diseño y los parámetros correspondientes, se observa que no es un método válido.

5.2 Cálculo con todos los parámetros del controlador y los coeficientes fraccionarios iguales

Teniendo en cuenta los tres parámetros del controlador, que ω_g es un parámetro ya especificado al principio (es decir, es conocido) y que $\lambda = \mu$

$$(5.2.1)\ A = \begin{pmatrix} 1 & {\omega_c}^{-\lambda}\cdot\cos\left(\frac{\pi}{2}\lambda\right) & {\omega_c}^{\lambda}\cdot\cos\left(\frac{\pi}{2}\lambda\right) \\ 0 & -{\omega_c}^{-\lambda}\cdot\sin\left(\frac{\pi}{2}\lambda\right) & {\omega_c}^{\lambda}\cdot\sin\left(\frac{\pi}{2}\lambda\right) \\ 1 & {\omega_g}^{-\lambda}\cdot\cos\left(\frac{\pi}{2}\lambda\right) & {\omega_g}^{\lambda}\cdot\cos\left(\frac{\pi}{2}\lambda\right) \\ 0 & -{\omega_g}^{-\lambda}\cdot\sin\left(\frac{\pi}{2}\lambda\right) & {\omega_g}^{\lambda}\cdot\sin\left(\frac{\pi}{2}\lambda\right) \end{pmatrix}$$

Por lo que el sistema a resolver queda de la siguiente manera:

$$\begin{pmatrix} 1 & {\omega_c}^{-\lambda}\cdot\cos\left(\frac{\pi}{2}\lambda\right) & {\omega_c}^{\lambda}\cdot\cos\left(\frac{\pi}{2}\lambda\right) \\ 0 & -{\omega_c}^{-\lambda}\cdot\sin\left(\frac{\pi}{2}\lambda\right) & {\omega_c}^{\lambda}\cdot\sin\left(\frac{\pi}{2}\lambda\right) \\ 1 & {\omega_g}^{-\lambda}\cdot\cos\left(\frac{\pi}{2}\lambda\right) & {\omega_g}^{\lambda}\cdot\cos\left(\frac{\pi}{2}\lambda\right) \\ 0 & -{\omega_g}^{-\lambda}\cdot\sin\left(\frac{\pi}{2}\lambda\right) & {\omega_g}^{\lambda}\cdot\sin\left(\frac{\pi}{2}\lambda\right) \end{pmatrix} \cdot \begin{pmatrix} K_p \\ K_p\cdot K_i \\ K_p\cdot T_d \end{pmatrix}$$

$$= -\begin{pmatrix} \cos(\emptyset_m + \tau\omega_c - < G'(j\omega_c))/|G'(j\omega_c)| \\ \sin(\emptyset_m + \tau\omega_c - < G'(j\omega_c))/|G'(j\omega_c)| \\ \cos(\tau\omega_g - < G(j\omega_g))/(Mg\cdot|G'(j\omega_g)|) \\ \sin(\tau\omega_g - < G'(j\omega_g))/(Mg\cdot|G'(j\omega_g)|) \end{pmatrix}$$

Para la resolución de este sistema, el rango de la matriz A ampliada a B debe ser igual a 3:

$$rango\big((A\ B)\big) = 3$$

Por lo que el determinante de esta matriz mencionada debe ser igual a cero:

$$det\big((A\ B)\big) = 0$$

Esto sólo ocurrirá para ciertos valores de λ, por lo que será necesario calcularlos para la obtención de las especificaciones del controlador.

Obtenido este valor y comprobado su adecuación, es posible quedarnos con tan solo tres filas de las matrices A y B y proceder a resolver el sistema, calculando la matriz X por el método mencionado en (4.17).

La resolución de este problema se hace a partir de un programa de MATLAB adjunto en el apartado 11.1.4.

5.3 Cálculo con todos los parámetros del controlador y los coeficientes fraccionarios distintos

Teniendo en cuenta los tres parámetros del controlador, que ω_g es un parámetro ya especificado al principio (es decir, es conocido) y que $\mu \neq \lambda$

$$(5.3.1)\quad A = \begin{pmatrix} 1 & {\omega_c}^{-\lambda} \cdot \cos\left(\frac{\pi}{2}\lambda\right) & {\omega_c}^{\mu} \cdot \cos\left(\frac{\pi}{2}\mu\right) \\ 0 & -{\omega_c}^{-\lambda} \cdot \sin\left(\frac{\pi}{2}\lambda\right) & {\omega_c}^{\mu} \cdot \sin\left(\frac{\pi}{2}\mu\right) \\ 1 & {\omega_g}^{-\lambda} \cdot \cos\left(\frac{\pi}{2}\lambda\right) & {\omega_g}^{\mu} \cdot \cos\left(\frac{\pi}{2}\mu\right) \\ 0 & -{\omega_g}^{-\lambda} \cdot \sin\left(\frac{\pi}{2}\lambda\right) & {\omega_g}^{\mu} \cdot \sin\left(\frac{\pi}{2}\mu\right) \end{pmatrix}$$

La resolución por este método es básicamente similar al anterior, salvo con la excepción que μ ≠ λ, por lo que tendremos una incógnita más.

Así, se sigue el mismo procedimiento, buscando que el rango de la matriz A ampliada con B sea 3 (por lo que su determinante deberá ser igual a 0).

Para resolver esta cuestión, se decidió ir fijando los valores de λ y luego calcular el valor de μ que hará el determinante cero. De esta manera, se tendrán diferentes pares de valores λ-μ que son válidos para el sistema, pudiendo posteriormente pasar a obtener los parámetros del PID con la resolución de la matriz X (4.17) mencionada con anterioridad.

La resolución matemática de este problema se realiza a partir de un programa de MATLAB generado, adjunto en el apartado 11.1.5.

6. Simulaciones con la función de transferencia de un sistema

En este apartado se realizan simulaciones, cambiando parámetros del sistema: ganancia, polos y retardo. Y así ver las posibles respuestas y su correcto funcionamiento.

6.1 Sistema de primer orden

6.1.1 Cambiando ganancias:

La gráfica de arriba pertenece al control fraccionario, mientras que la de abajo al control clásico. Los datos empleados y que se mantendrán fijos en este apartado son:

- Para el fraccionario

$$\lambda = 0.95$$
$$\mu = 0.97$$
$$\text{Kp} = 0.0427$$
$$\text{Ki} = 0.0745$$
$$\text{Kd} = 0.1479$$

- Para el clásico

$$\text{Kp}=0.3117$$

$$\text{Ki}=0.0178$$

$$\text{Kd}=0.0076$$

Y aquí mostramos los resultados:

- Ganancia=2

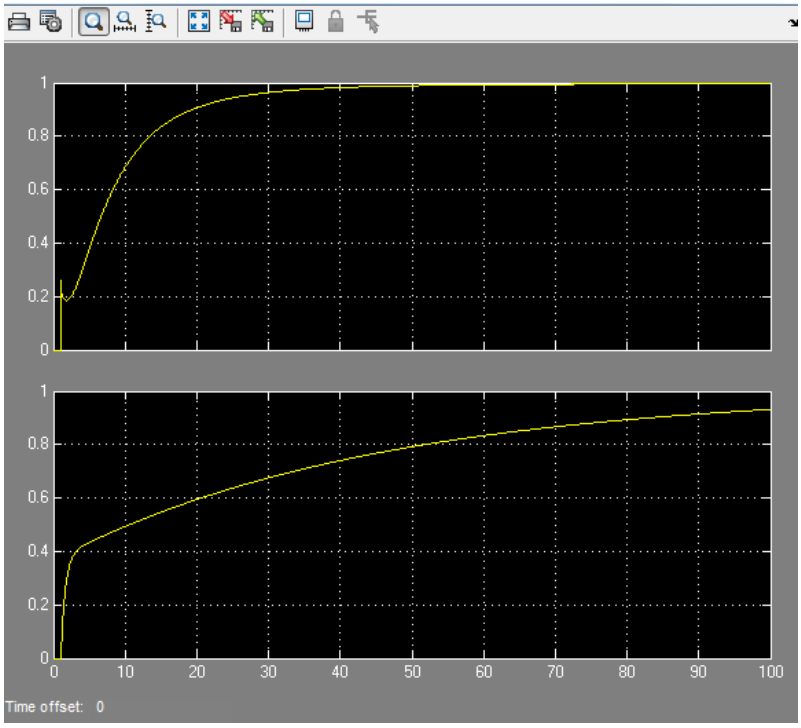

Figura 8. Efecto cambio ganancia (G=2)

- Ganancia=3

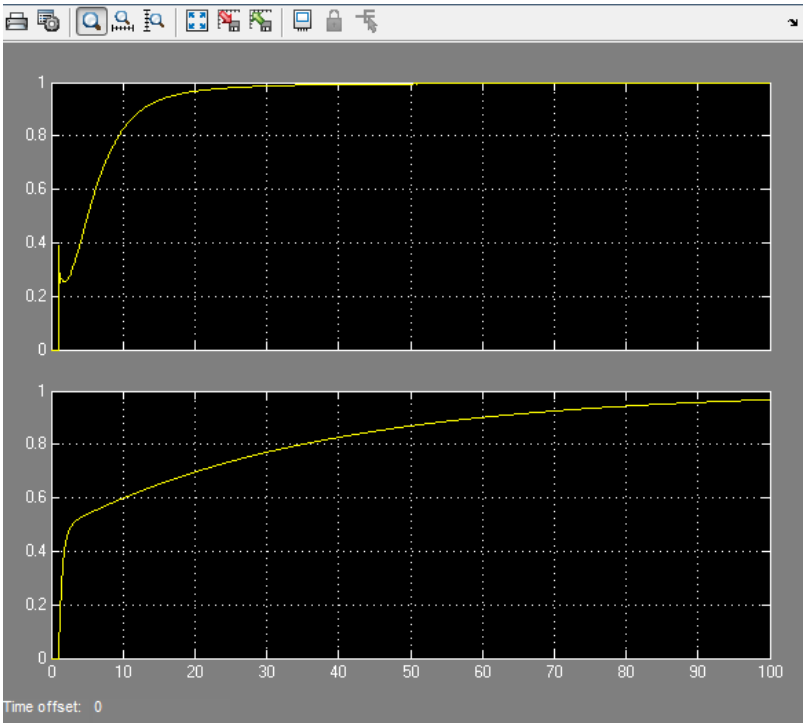

Figura 9. Efecto cambio ganancia (G=3)

- Ganancia=4

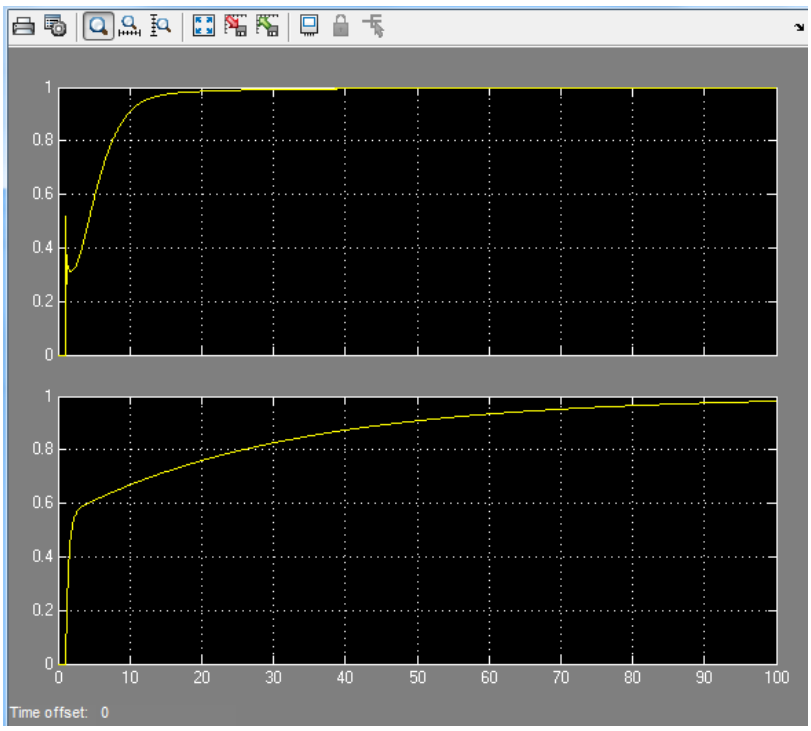

Figura 10. Efecto cambio ganancia (G=4)

- Ganancia=5

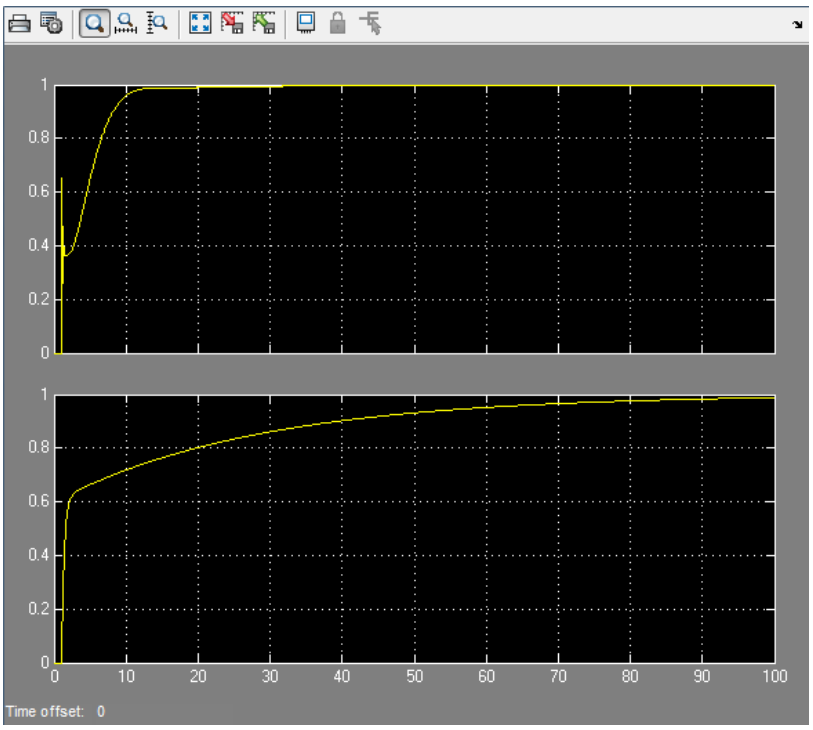

Figura 11. Efecto cambio ganancia (G=5)

- Ganancia=6

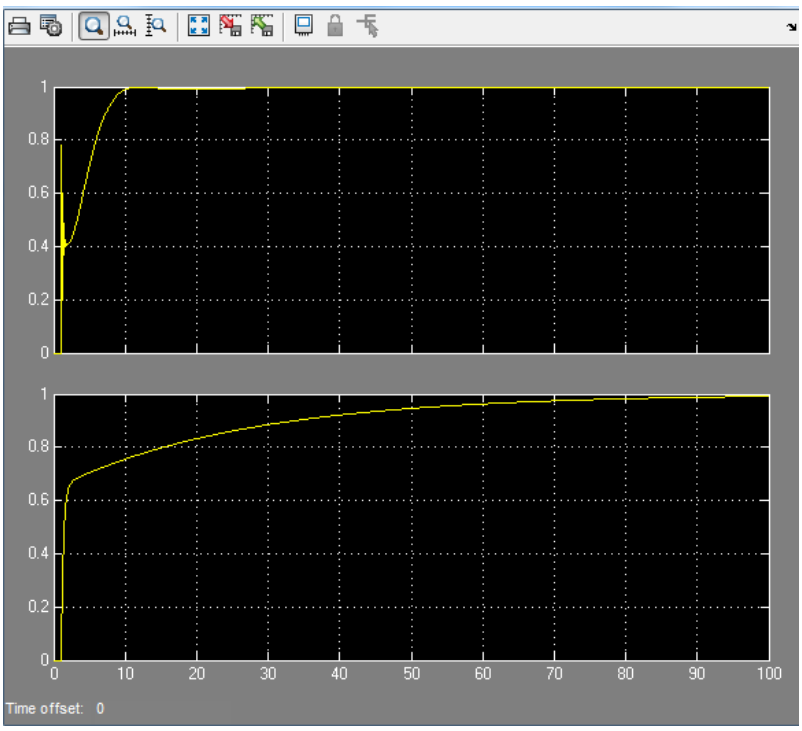

Figura 12. Efecto cambio ganancia (G=6)

- Ganancia=7

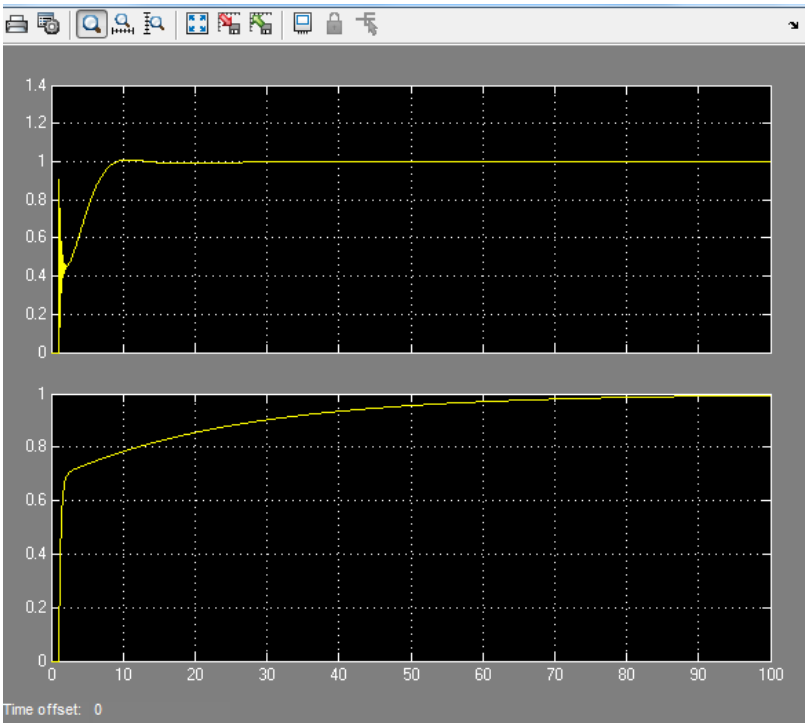

Figura 13. Efecto cambio ganancia (G=7)

- Ganancia=8

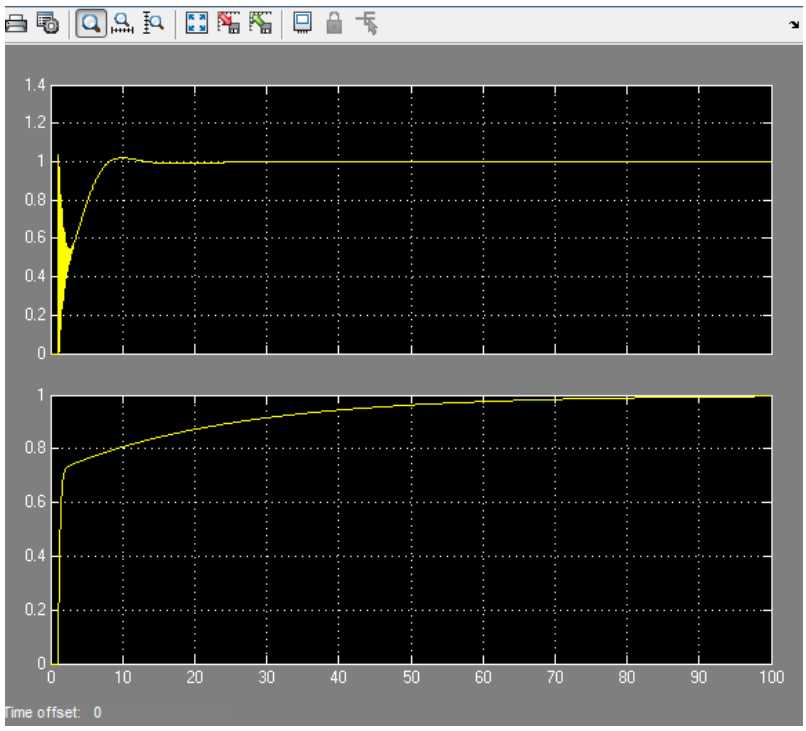

Figura 14. Efecto cambio ganancia (G=8)

- Ganancia=9

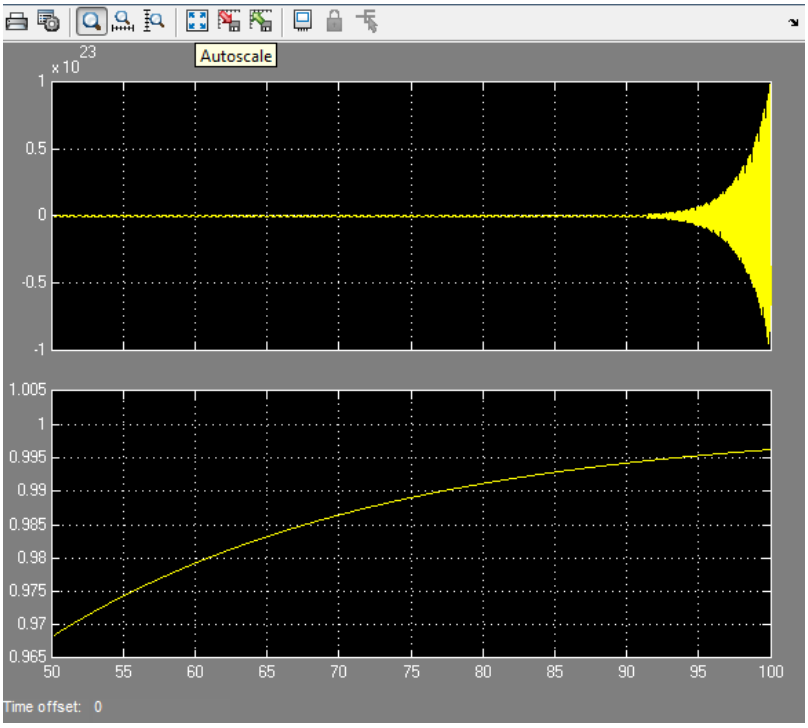

Figura 15. Efecto cambio ganancia (G=9)

Por lo que podemos observar, el control fraccionario aparecen oscilaciones y se vuelve inestable antes que el control clásico. El control clásico, con las ganancias mostradas no llega a volverse inestable, mientras que el fraccionario comienza a ser inestable a partir de la ganancia G=9.

6.1.2 Cambiando retardo:

La gráfica de arriba pertenece al control fraccionario, mientras que la de abajo al control clásico. Los datos empleados y que se mantendrán fijos en este apartado son:

- Para el fraccionario

$$\lambda = 0.95\ y\ \mu = 0.97 \rightarrow \mathrm{Kp} = 0.0427, \mathrm{Ki} = 0.0745, \mathrm{Kd} = 0.1479$$

- Para el clásico

Kp=0.3117, Ki=0.0178, Kd=0.0076

Y aquí mostramos los resultados:

- Retardo=0

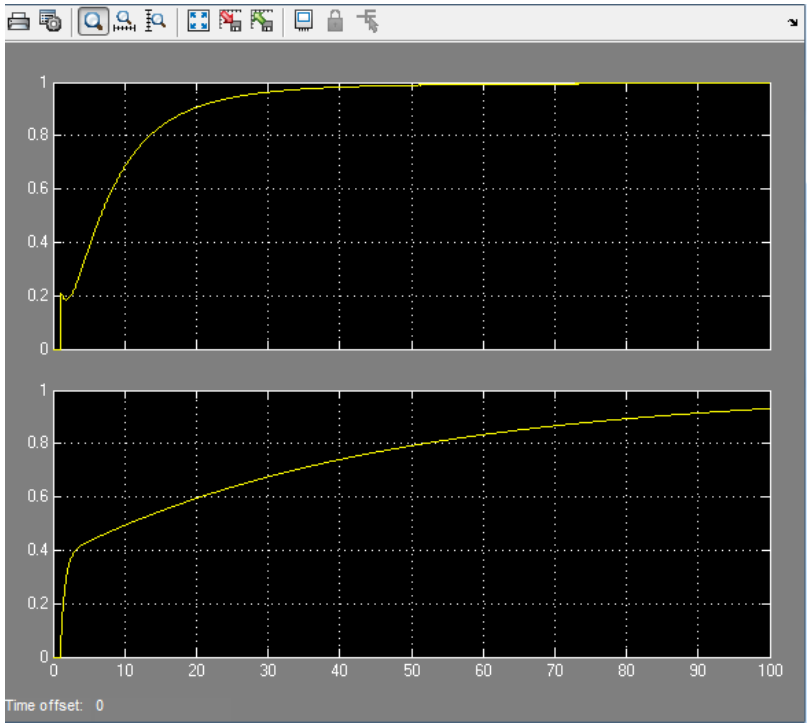

Figura 16. Efecto cambio de retardo (R=0)

- Retardo=0.05

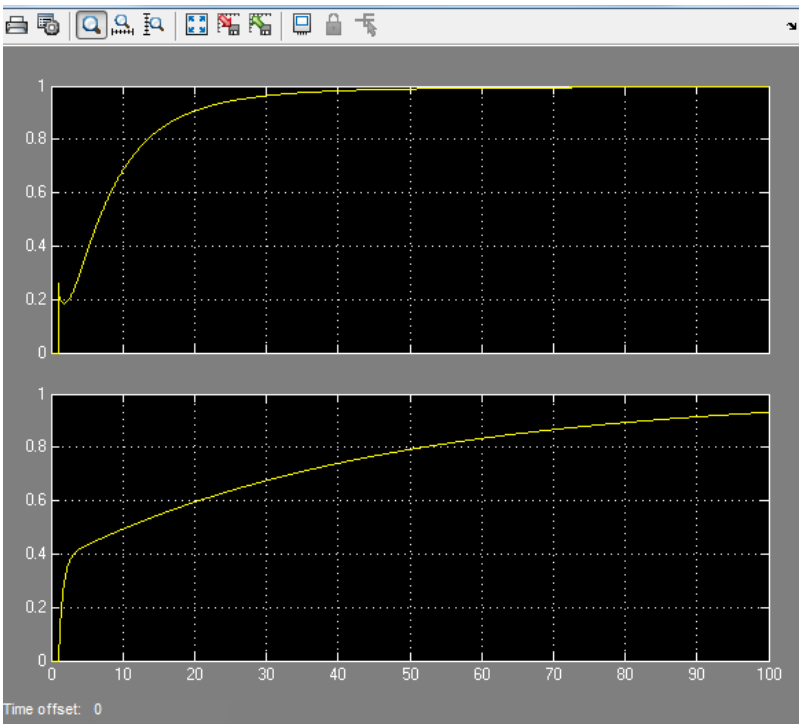

Figura 17. Efecto cambio de retardo (R=0.05)

- Retardo=0.10

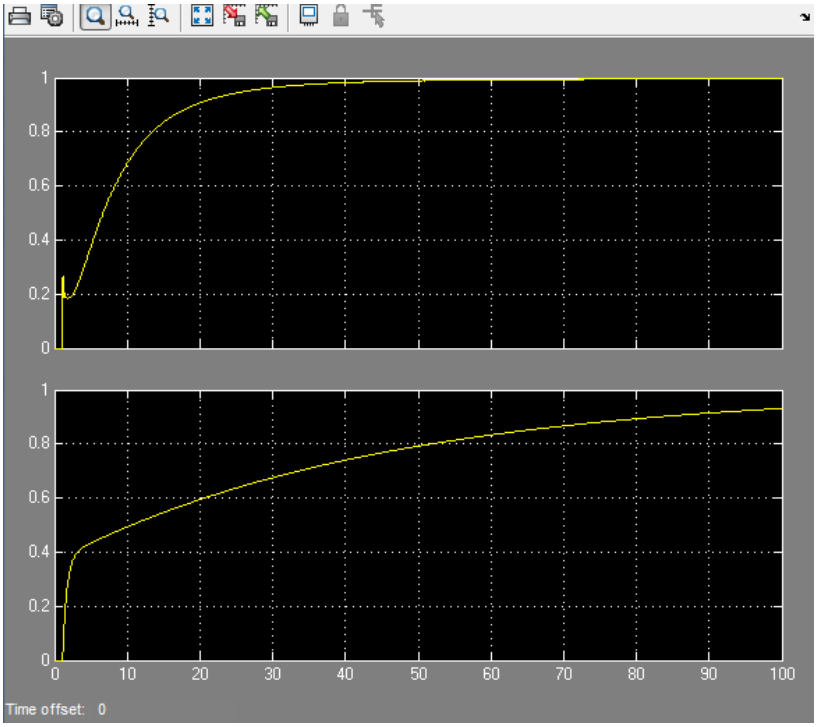

Figura 18. Efecto cambio de retardo (R=0.10)

- Retardo=0.15

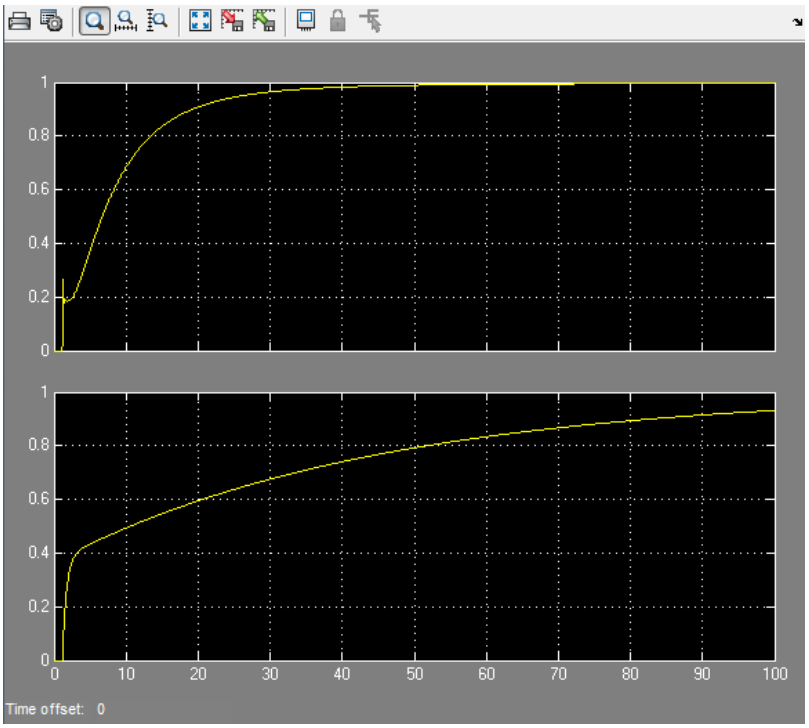

Figura 19. Efecto cambio de retardo (R=0.15)

- Retardo=0.20

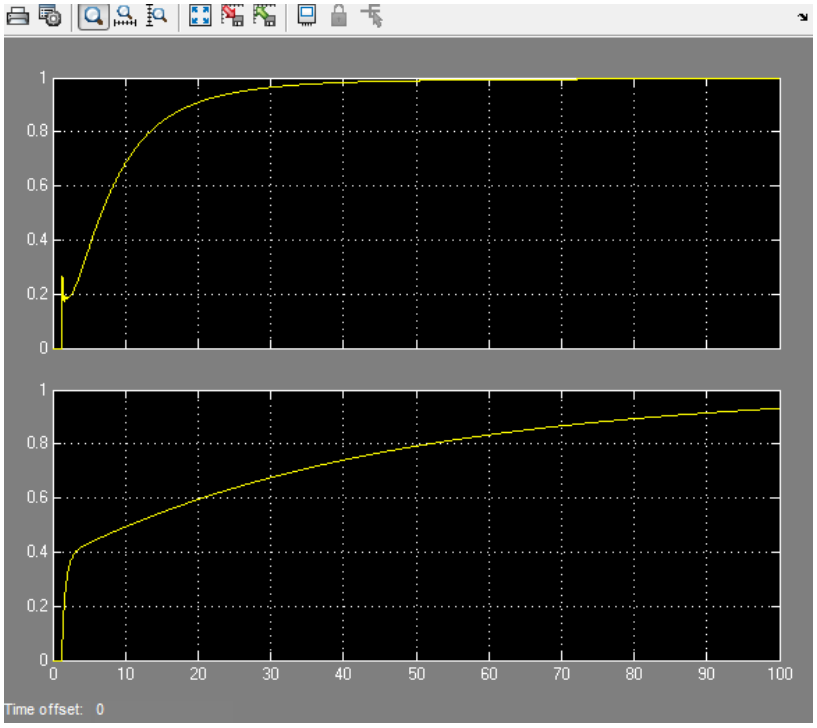

Figura 20. Efecto cambio de retardo (R=0.20)

- Retardo=0.25

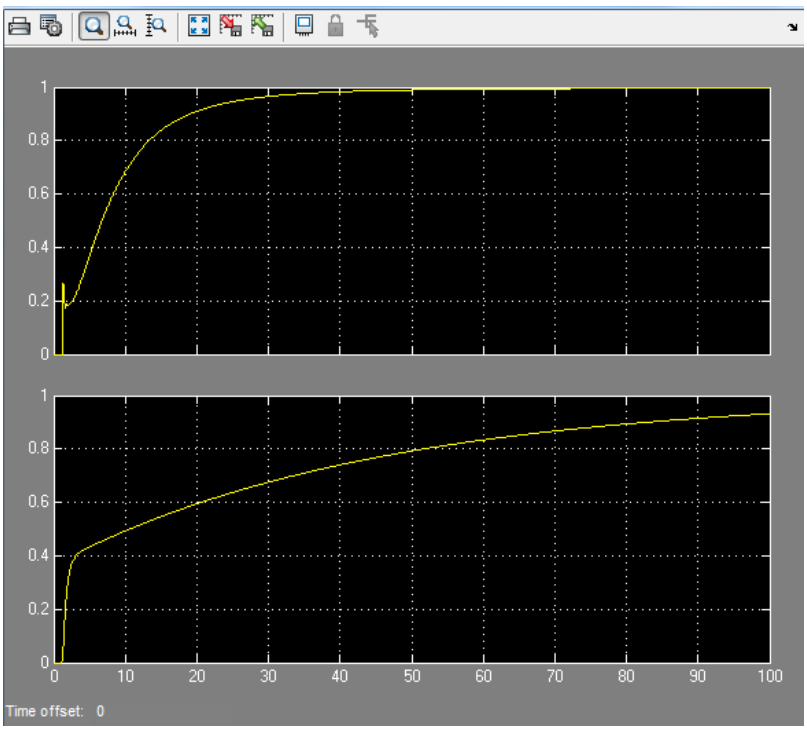

Figura 21. Efecto cambio de retardo (R=0.25)

- Retardo=0.30

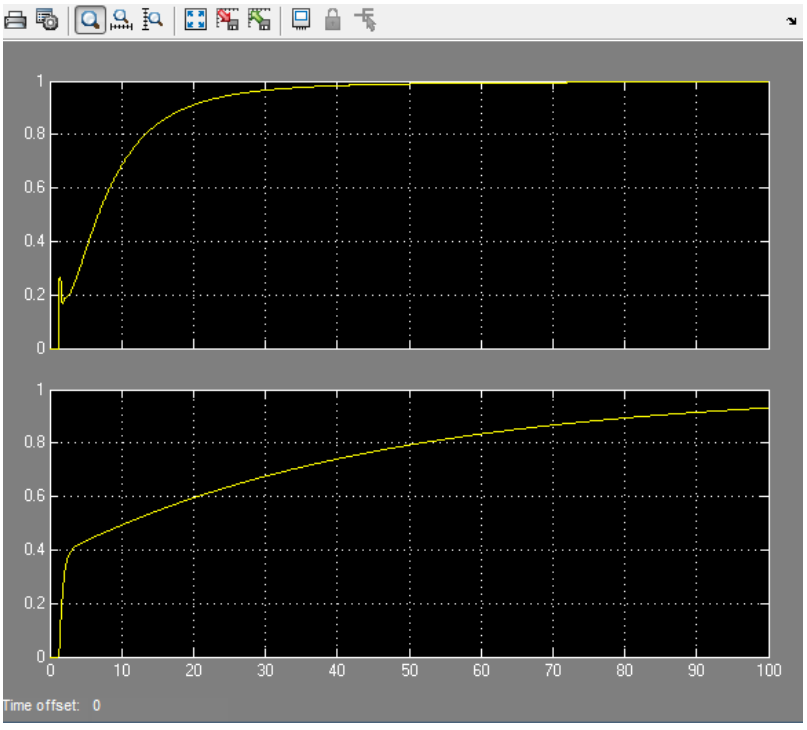

Figura 22. Efecto cambio de retardo (R=0.30)

- Retardo=0.35

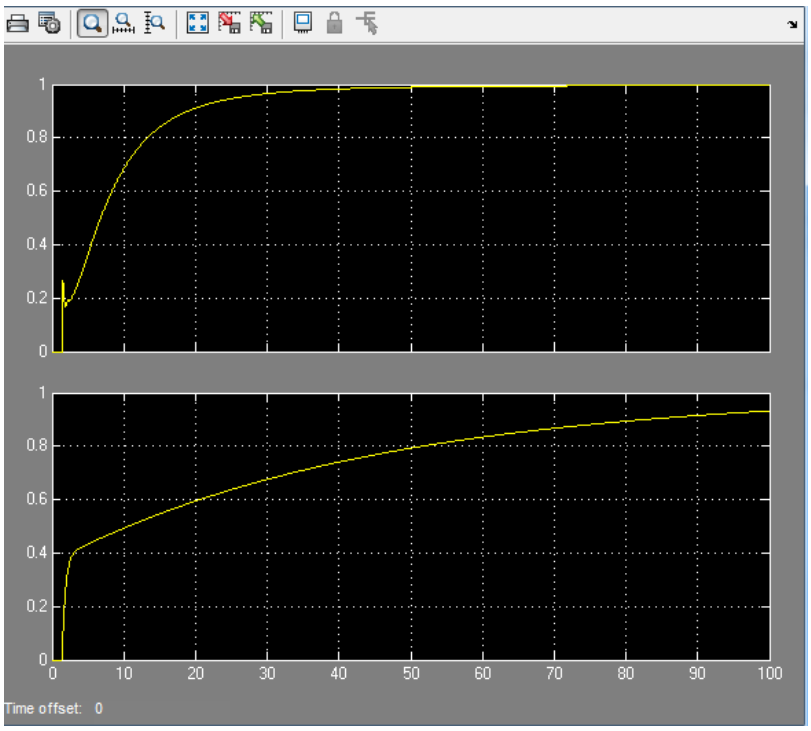

Figura 23. Efecto cambio de retardo (R=0.35)

- Retardo=0.40

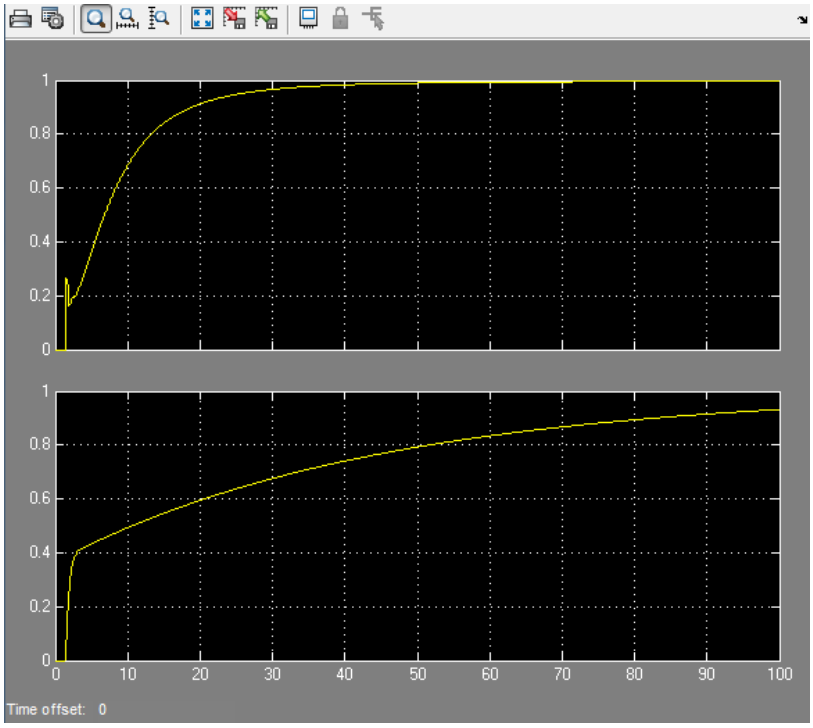

Figura 24. Efecto cambio de retardo (R=0.40)

- Retardo=0.45

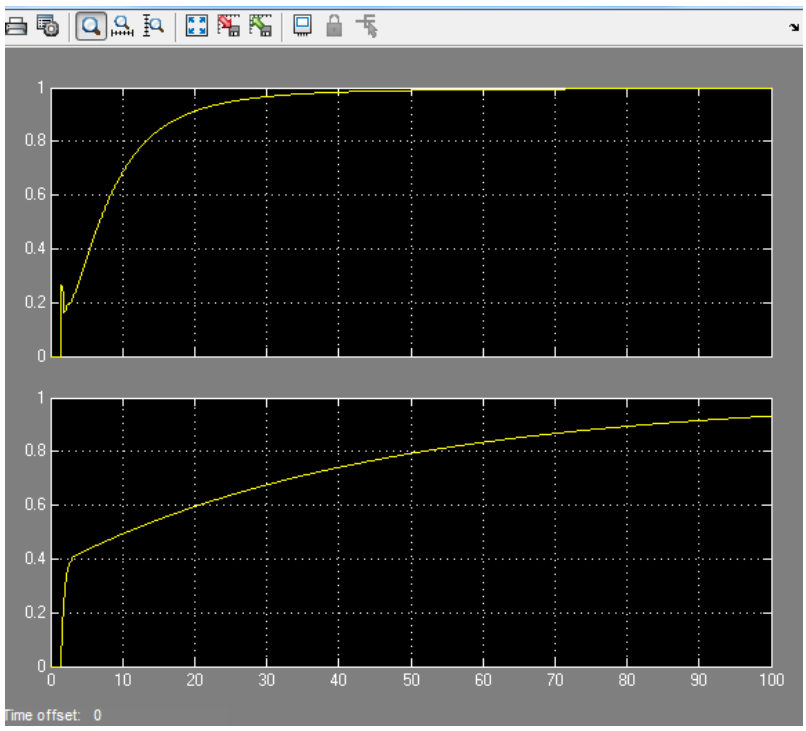

Figura 25. Efecto cambio de retardo (R=0.45)

- Retardo=1

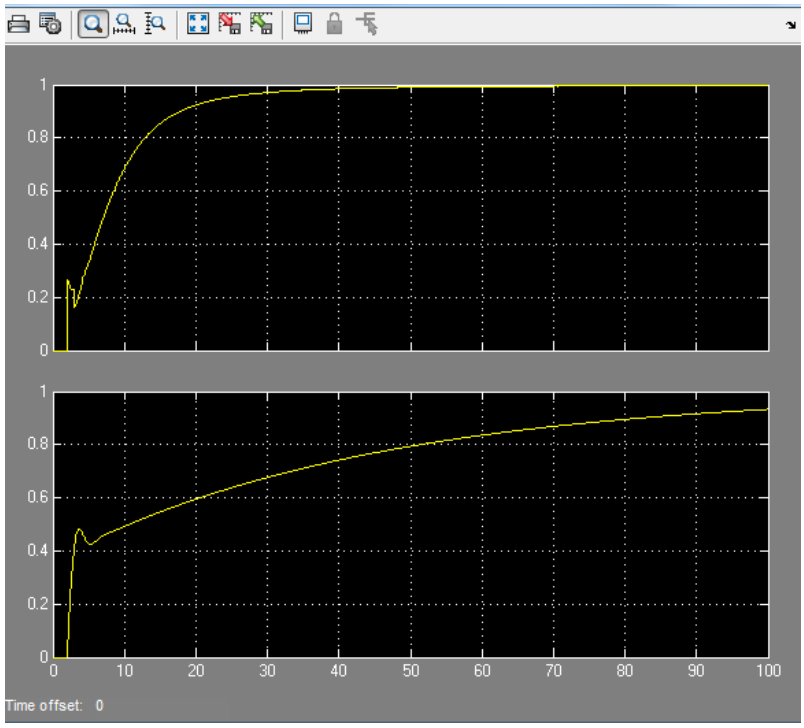

Figura 26. Efecto cambio de retardo (R=1)

- Retardo=2

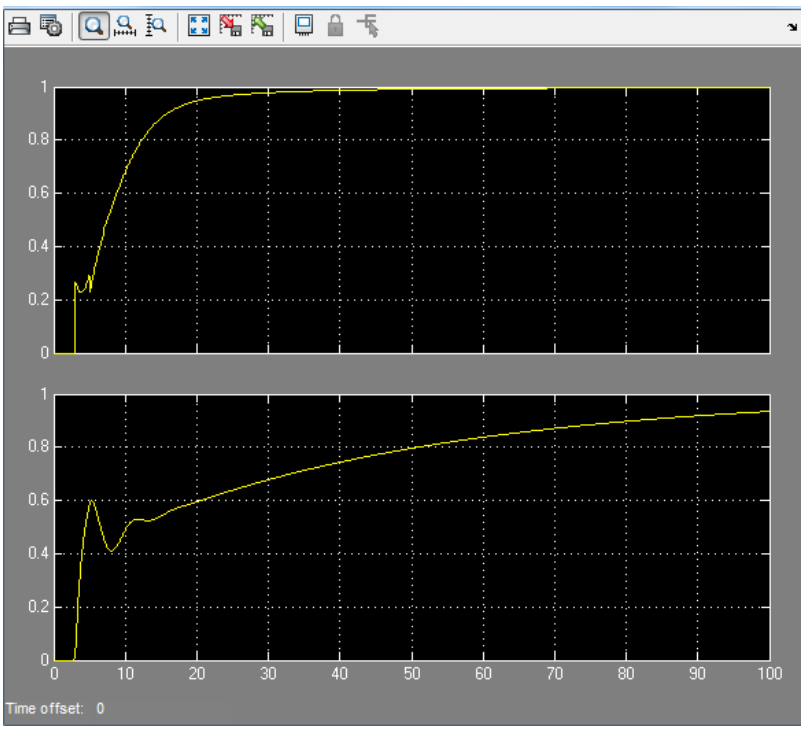

Figura 27. Efecto cambio de retardo (R=2)

- Retardo=3

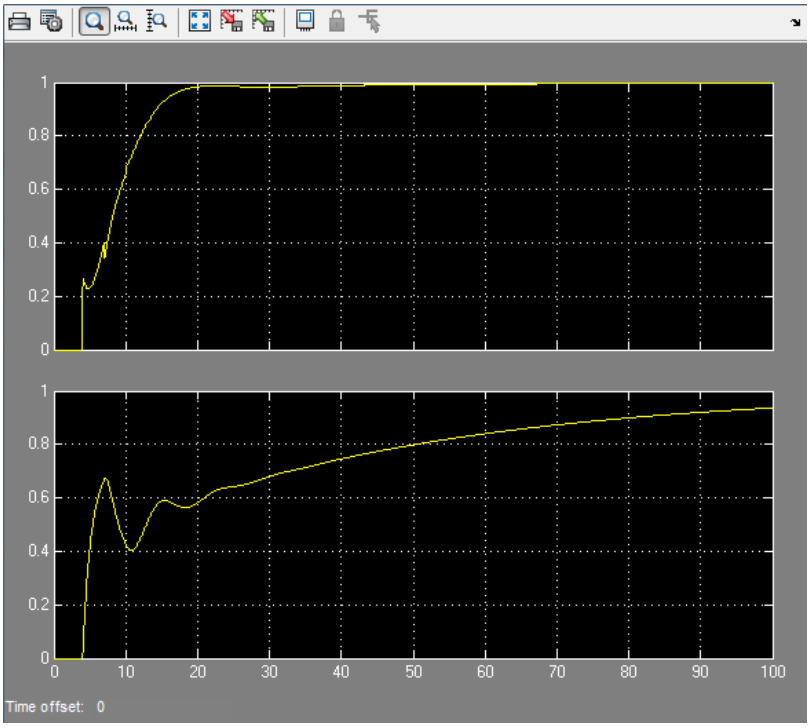

Figura 28. Efecto cambio de retardo (R=3)

- Retardo=4

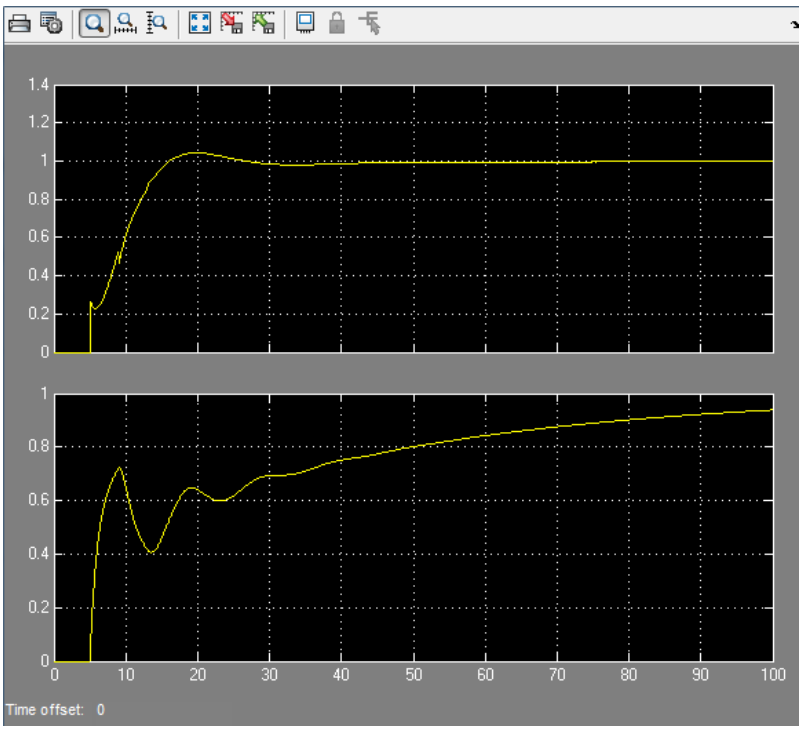

Figura 29. Efecto cambio de retardo (R=4)

- Retardo=5

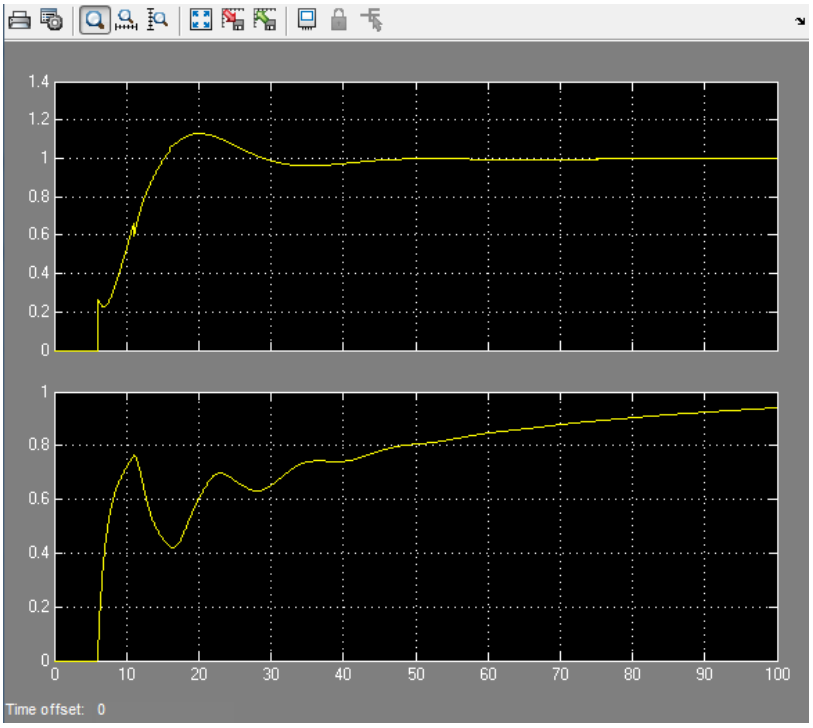

Figura 30. Efecto cambio de retardo (R=5)

- Retardo=6

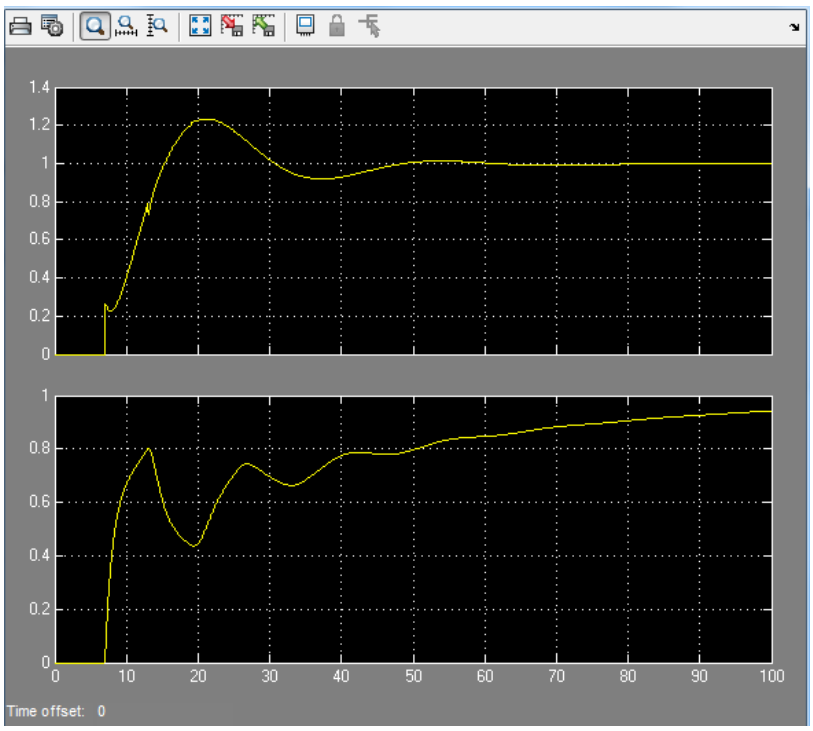

Figura 31. Efecto cambio de retardo (R=6)

- Retardo=10

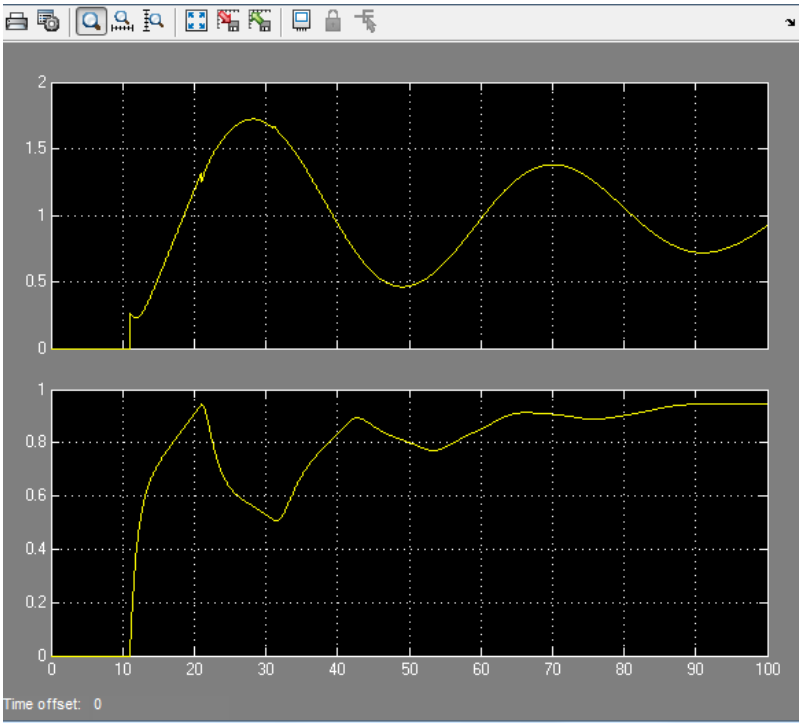

Figura 32. Efecto cambio de retardo (R=10)

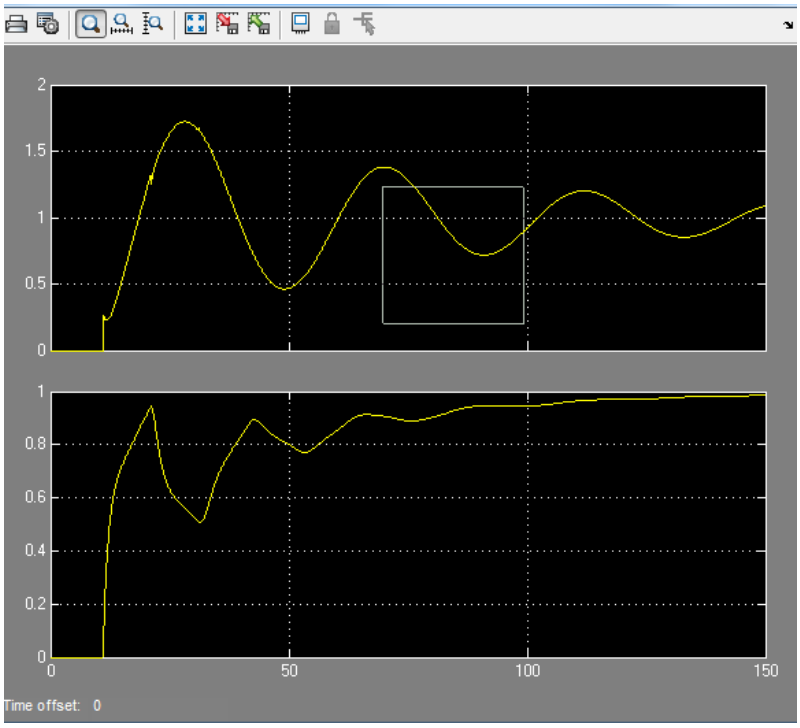

Figura 33. Ampliación escala del efecto cambio de retardo (R=10)

Para los retardos más pequeños sigue habiendo una buena respuesta. Sin embargo, con los retardos mayores aparecen muchas más oscilaciones, pero observamos que ningún sistema llega a volverse inestable.

6.1.3 Cambiando polos del sistema:

La gráfica de arriba pertenece al control fraccionario, mientras que la de abajo al control clásico. Los datos empleados y que se mantendrán fijos en este apartado son:

- Para el fraccionario

$$\lambda = 0.95 \; y \; \mu = 0.97 \rightarrow \text{Kp} = 0.0427, \text{Ki} = 0.0745, \text{Kd} = 0.1479$$

- Para el clásico

Kp=0.3117, Ki=0.0178, Kd=0.0076

Y aquí mostramos los resultados:

$$G(s) = \frac{2}{(s+A)}$$

- Polo = 1

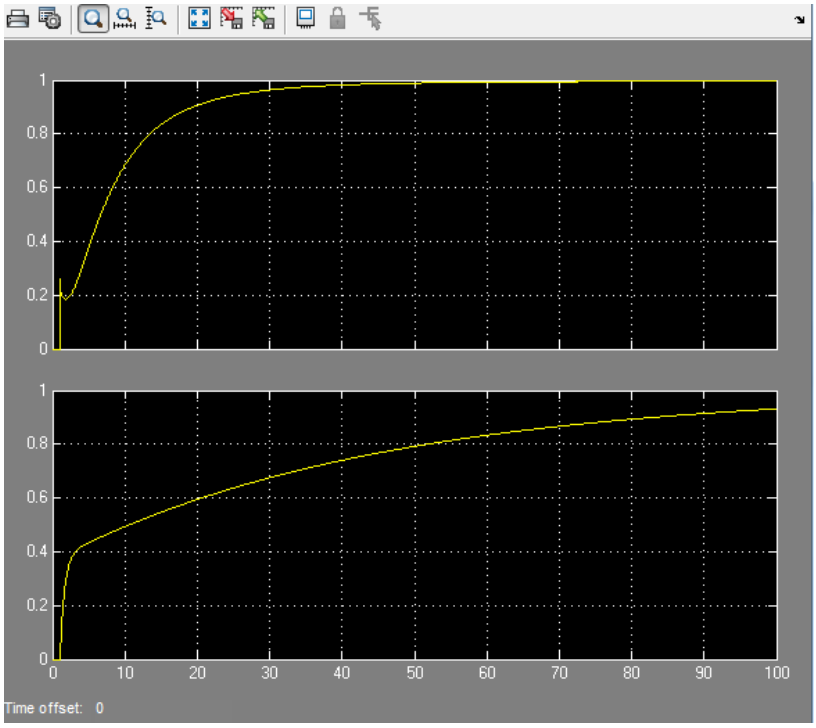

Figura 34. Efecto cambio de polo (A=1)

- Polo = 2

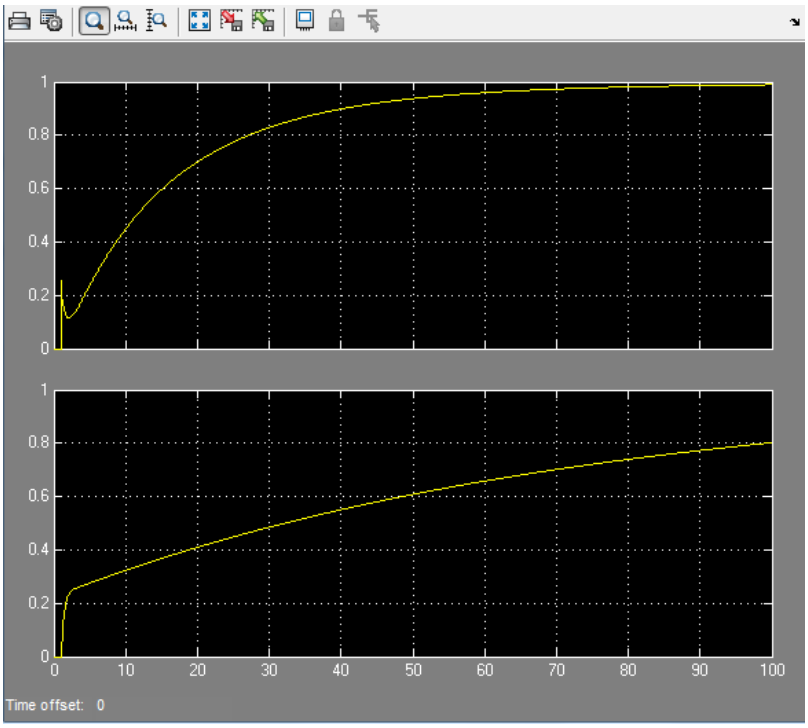

Figura 35. Efecto cambio de polo (A=2)

- Polo = 3

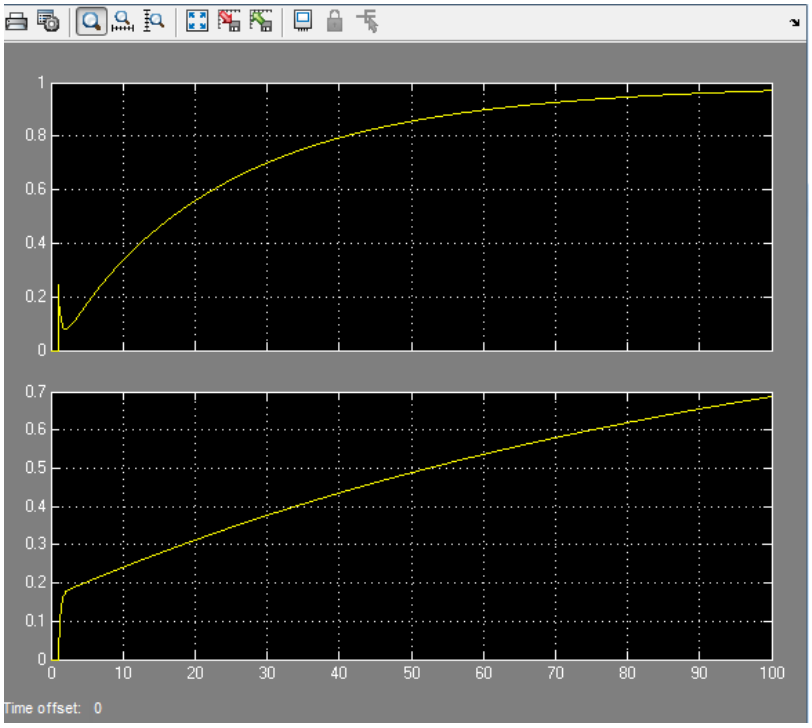

Figura 36. Efecto cambio de polo (A=3)

- Polo = 4

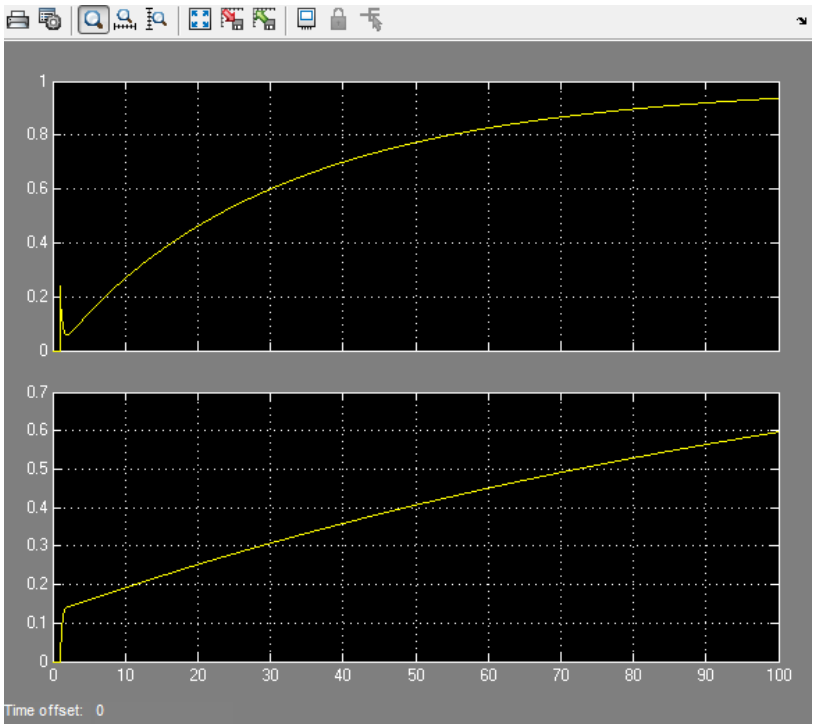

Figura 37. Efecto cambio de polo (A=4)

- Polo = 5

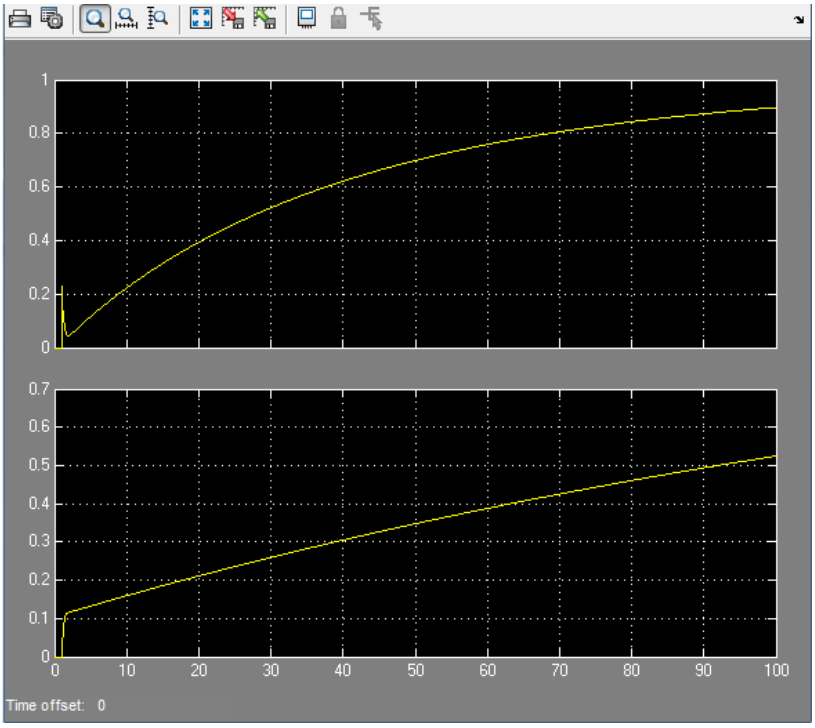

Figura 38. Efecto cambio de polo (A=5)

- Polo = 6

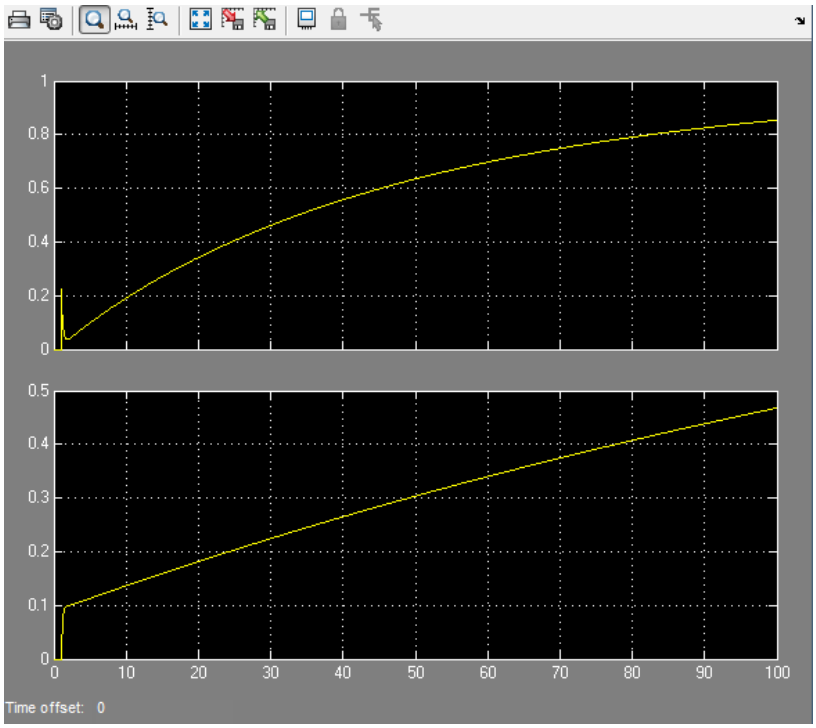

Figura 39. Efecto cambio de polo (A=6)

- Polo = 7

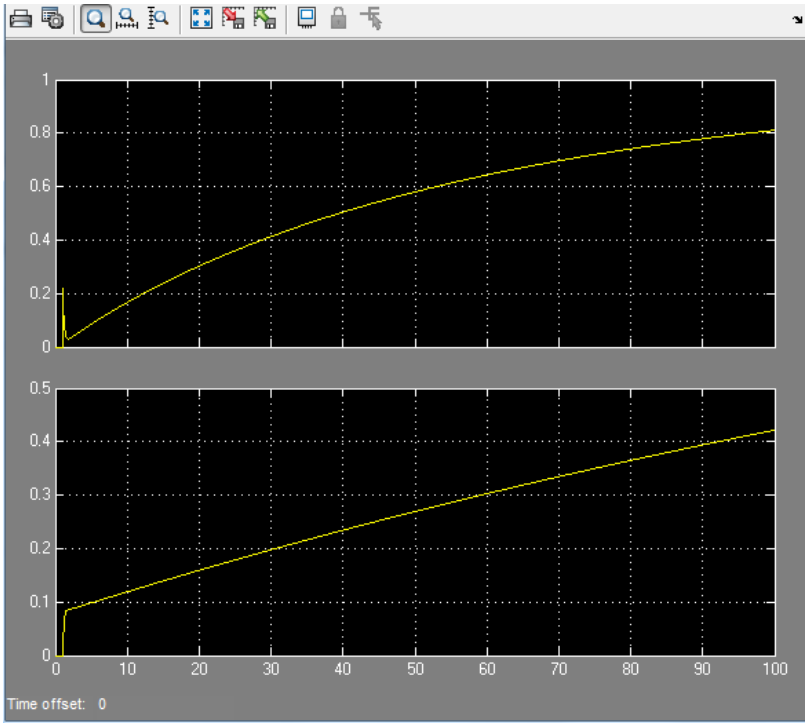

Figura 40. Efecto cambio de polo (A=7)

- Polo = 8

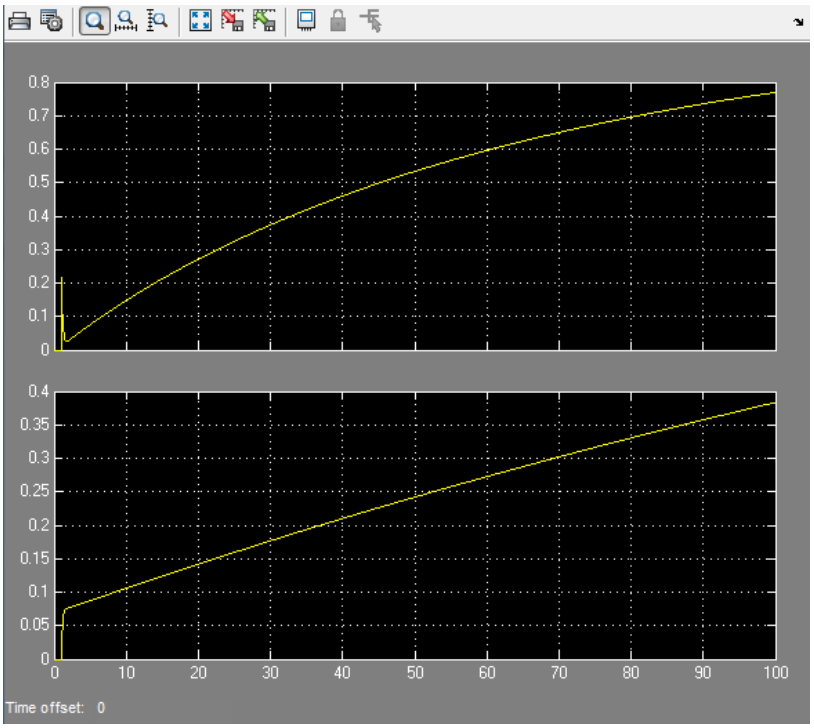

Figura 41. Efecto cambio de polo (A=8)

- Polo = 9

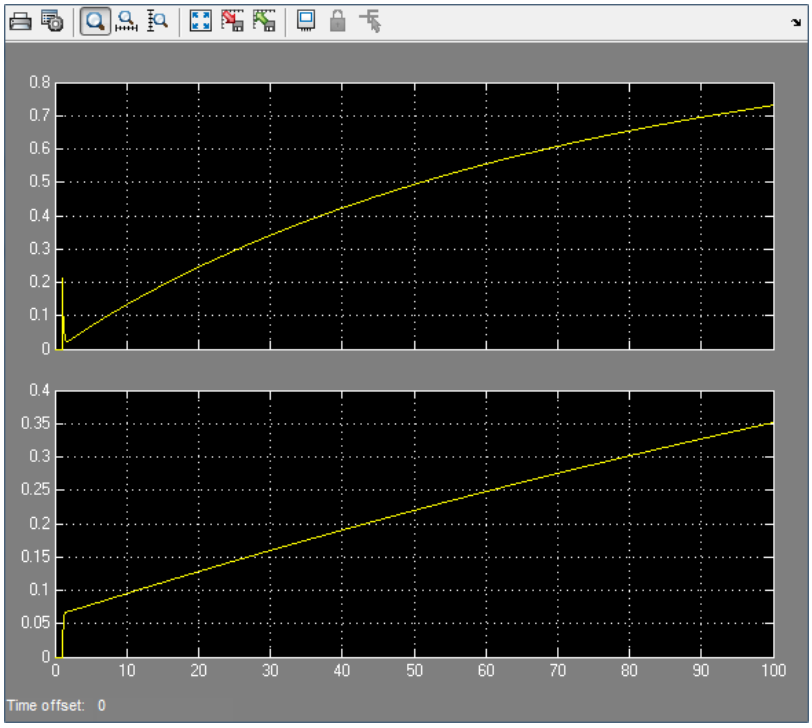

Figura 42. Efecto cambio de polo (A=9)

- Polo = 10

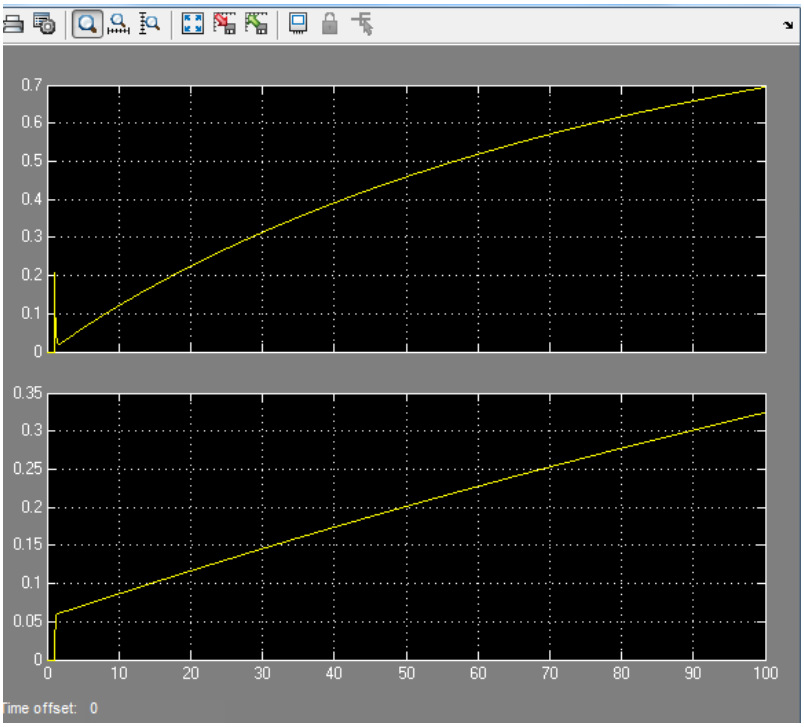

Figura 43. Efecto cambio de polo (A=10)

Respecto a las variaciones en los polos del sistema, vemos que a medida que los aumentamos, el sistema deja de llegar al valor de consigna y, que este hecho ocurre antes en el PID clásico.

6.2 Sistema de segundo orden

Pese que los sistemas reales estudiados en el laboratorio son de primer orden, se han realizado simulaciones para sistemas de segundo de orden con el fin de comprobar la utilidad del código implementado para otro tipo de sistemas reales.

La función de transferencia empleada para la simulación es la siguiente:

$$G(s) = \frac{1}{s^2 + 2s + 1}$$

La gráfica de arriba pertenece al control fraccionario, mientras que la de abajo al control clásico. Los parámetros empleados son los del PID clásico, que se mantendrán fijos en este apartado (Kp=0.0362, Td= 0.1178, Ti = 0.00095), y veremos los distintos valores para el caso fraccionario:

- $\lambda = 0.91\ y\ \mu = 0.7367 \rightarrow \mathrm{Kp} = 0.0194, \mathrm{Ki} = 0.023, \mathrm{Kd} = 0.80$

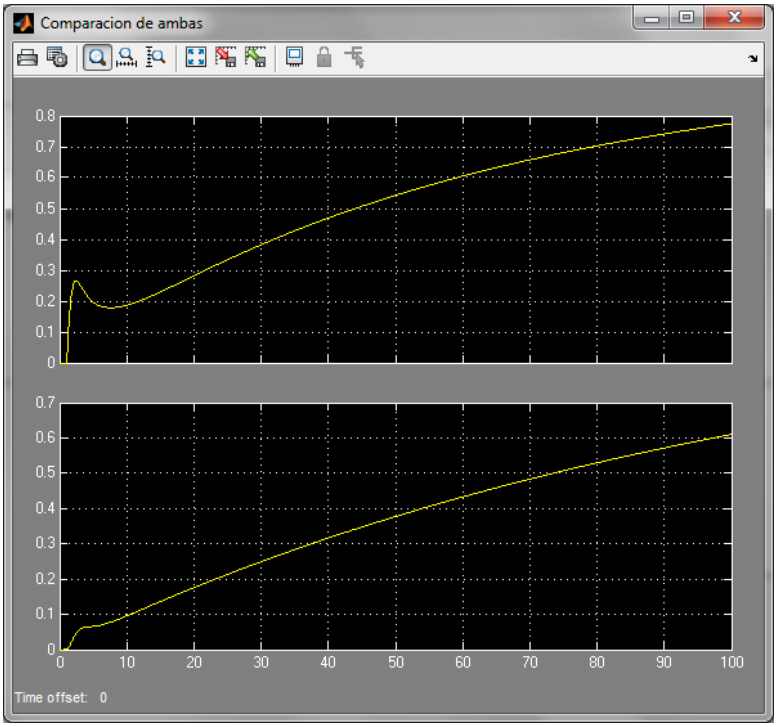

Figura 44. Simulación segundo orden

- $\lambda = 0.92\ y\ \mu = 0.8057 \rightarrow \mathrm{Kp} = 0.0407, \mathrm{Ki} = 0.0445, \mathrm{Kd} = 0.3604$

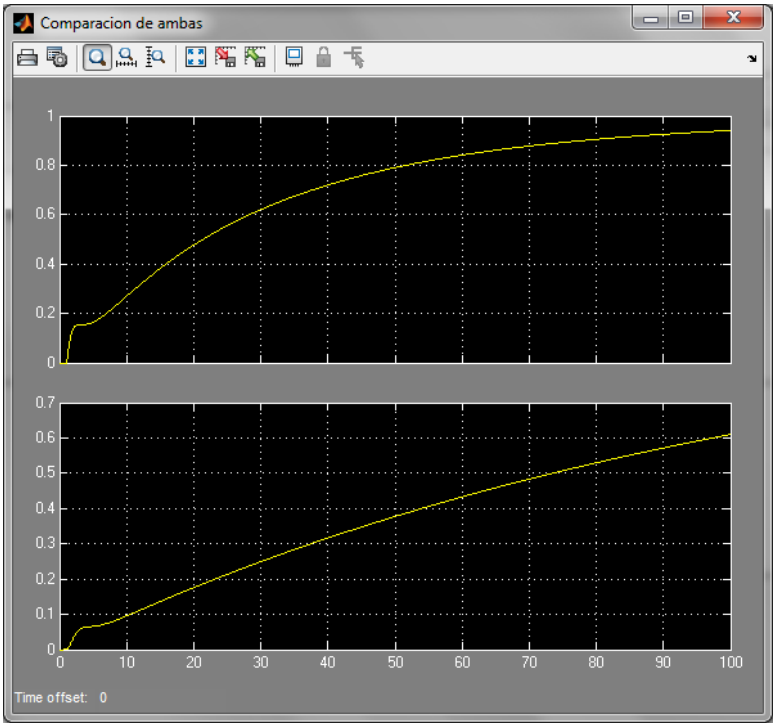

Figura 45. Simulación segundo orden

- $\lambda = 0.93\ y\ \mu = 0.877 \rightarrow \mathrm{Kp} = 0.0609, \mathrm{Ki} = 0.0607, \mathrm{Kd} = 0.23$

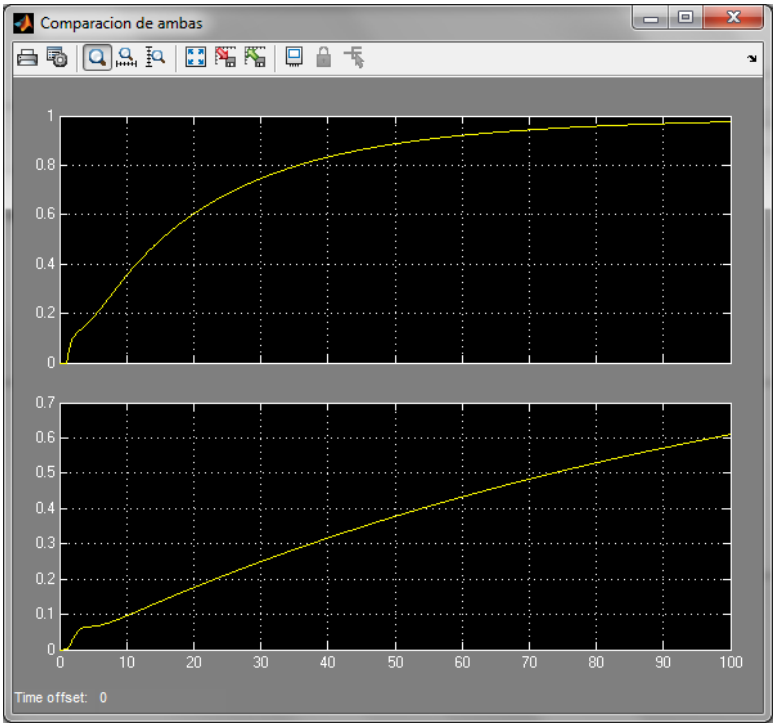

Figura 46. Simulación segundo orden

- $\lambda = 0.94\ y\ \mu = 0.95 \rightarrow \mathrm{Kp} = 0.0792, \mathrm{Ki} = 0.0787, \mathrm{Kd} = 0.1667$

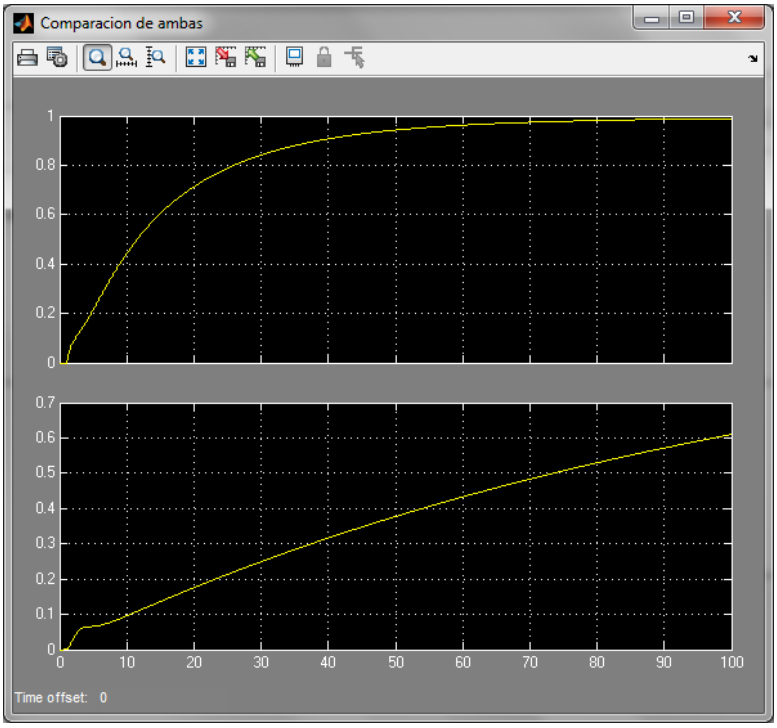

Figura 47. Simulación segundo orden

7. Simulación para sistemas reales

7.1 Doble rotor (TRMS)

7.1.1 Descripción del doble rotor (TRMS)

El doble rotor, como su propio nombre indica, está formado por dos motores: un motor principal y otro en la cola. Cuando el motor principal gira, se genera un par, éste causa que la parte central del sistema gire en dirección opuesta. Por lo que para contrarrestar dicho giro, se pone un rotor más pequeño perpendicular al principal en la cola. El objetivo del rotor de la cola será oponerse a todo par que genere el giro del rotor principal.

El TRMS está compuesto por dos rotores perpendiculares (cada uno de ellos) a la columna central y tienen libertad de movimiento en las direcciones vertical y

horizontal. Tiene dos grados de libertad debido a que el rotor puede girar sobre los ejes vertical y horizontal.
Cuando no se usa o se quiere cambiar los parámetros de control, uno o los dos ejes se pueden boquear con unos tornillos de fijación que se ven en la figura 49.

Cuando el tornillo del eje horizontal está suelto, el peso del eje vertical en su posición de descanso forma con el brazo del rotor un ángulo de 28º, mostrada en la figura 48. Hay que tener en cuanto dicho ángulo para aplicar la potencia al motor vertical para levantar el brazo.

El doble rotor es un sistema multivariable, no lineal y fuertemente acoplado. Y su comportamiento real se asemeja al comportamiento real de un helicóptero pero dos diferencias importantes:

- El TRMS se controla a través de la velocidad de los motores controlando la aplicación de voltaje, mientras que un helicóptero real va más o menos a una velocidad constante y la fuerza propulsiva se controla con el cambio de posición de las palas del rotor.
- Un helicóptero real tiene movimiento libre, mientas que el TRMS está en una plataforma fija.

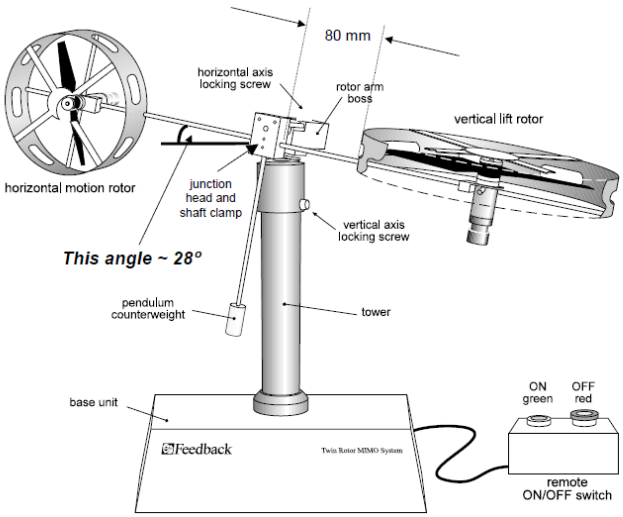

Figura 48. Montaje del TRMS de feedback

El doble rotor tiene tres modos de operación:

- Una sola dirección usando la cola del rotor: movimiento en el plano horizontal.
- Una sola dirección usando el rotor principal: movimiento en el plano vertical.
- Dos direcciones usando ambos planos.

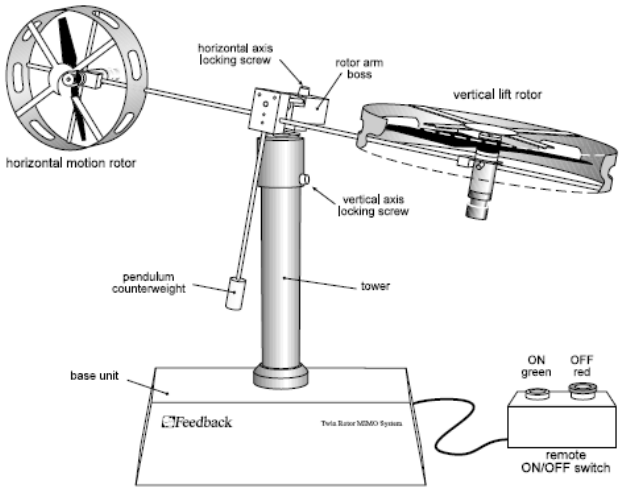

Figura 49. Esquema TRMS mostrando los tornillos de bloqueo

7.1.2 Modelo del sistema de doble rotor

Para realizar el estudio de los dos tipos de controladores con los códigos realizados en MATLAB (Apartado 13: ANEXOS), lo primero que vamos a necesitar es el modelo del sistema a controlar. A continuación. En la figura 50, les mostramos el diagrama de bloque del TRMS a partir del cual vamos a trabajar.

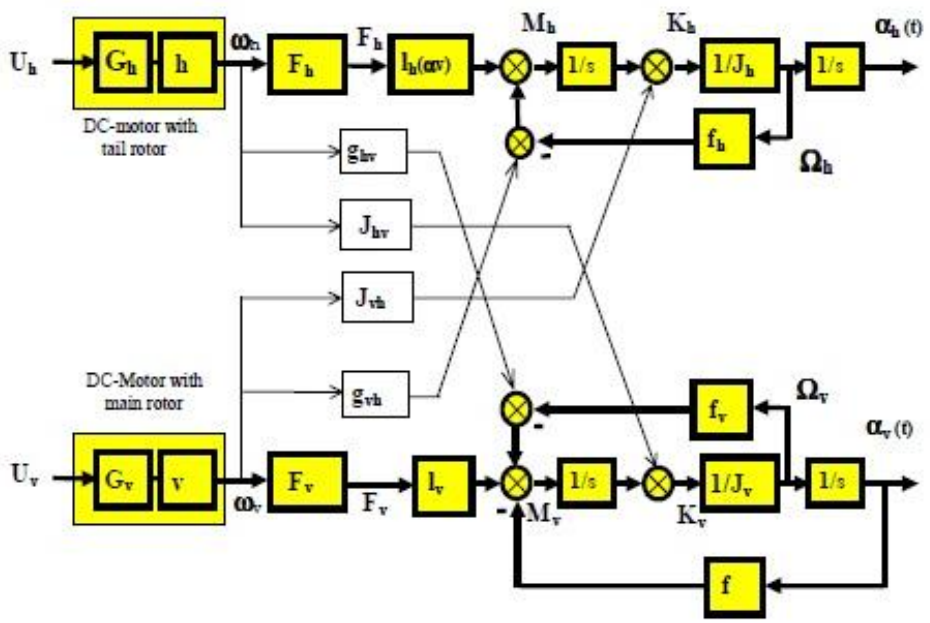

Figura 50. Diagrama de bloques del TRMS

Como ya contamos al principio, se trata de un sistema acoplado y no lineal. En las siguientes tablas, le mostramos qué es cada uno de los parámetros del diagrama de bloques y cuáles son las no linealidades que se encuentran presentes en él.

En la primera tabla (tabla 2), nos referiremos a los parámetros del rotor de cola, en la siguiente (tabla 3) haremos referencia a los del rotor principal y en la última (tabla 4) a todos los parámetros de acoplamiento de ambos rotores.

α_h	Posición horizontal del TRMS (posición de guiñada)
Ω_h	Velocidad angular del TRMS (velocidad de guiñada)
U_h	Voltaje aplicado al rotor de cola
G_h	Función de transferencia linear del rotor de cola
h	Parte no lineal del rotor de cola
ω_h	Velocidad angular del rotor de cola
F_h	Función no lineal, cuadrática: $F \approx K \cdot \omega^2$
I_h	Brazo eficaz de la fuerza aerodinámica del rotor de cola
J_h	No linealidad del momento de inercia respecto al eje vertical
M_h	Par de giro horizontal
K_h	Momento angular horizontal
f_h	Momento de la fuerza de fricción del eje vertical

Tabla 2. Parámetros del rotor de cola

α_v	Posición vertical del TRMS (posición de cabeceo)
Ω_v	Velocidad angular del TRMS (velocidad de cabeceo)
U_v	Voltaje aplicado al rotor principal
G_v	Función de transferencia linear del rotor principal
v	Parte no lineal del rotor principal
Ω_v	Velocidad angular del rotor principal
F_v	Función no lineal, cuadrática: $F \approx K \cdot \omega^2$
I_v	Brazo de la fuerza aerodinámica del rotor principal
J_v	No linealidad del momento de inercia respecto al eje horizontal
M_v	Par de giro vertical
K_v	Momento angular vertical
f_v	Momento de la fuerza de fricción del eje horizontal

Tabla 3. Parámetros del rotor principal

$\mathbf{J_{hv}}$	Momento angular vertical del rotor de cola
$\mathbf{J_{vh}}$	Momento angular horizontal del rotor principal
$\mathbf{g_{vh}}$	Función no lineal provocada por el momento de giro respecto de ω_v
$\mathbf{g_{hv}}$	Función no lineal provocada por el momento de giro respecto de ω_h

Tabla 4. Parámetros de acoplamiento

Para nuestra comprobación a realizar en el sistema real obviaremos los acoplamientos entre ambos rotores. Y vamos a simplificar el modelo para buscar una única función de transferencia a partir de cual podamos sacar los parámetros del PID clásico y fraccionario con los códigos realizados que se encuentran en los ANEXOS.

Nuestro diagrama de bloque lo simplificaremos uniendo todas las ganancias que posee el sistema y dejándolo en función de la no linealidad y la salida, quedando de la siguiente manera:

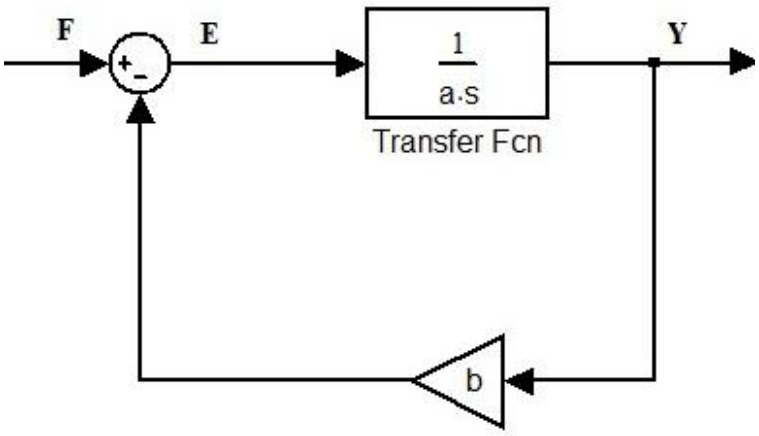

Figura 51. Simplificación del diagrama de bloques

Sabemos que:

$$\begin{cases} Y = \dfrac{1}{a \cdot s} \cdot E \\ E = F - bY \end{cases}$$

Por tanto:

$$Y = \frac{1}{a \cdot s} \cdot (F - bY) = \frac{1}{a \cdot s} F - \frac{b}{a \cdot s} Y \rightarrow$$

$$\left(1 + \frac{1}{a \cdot s}\right) Y = \frac{1}{a \cdot s} F \rightarrow$$

$$\boldsymbol{Y = \frac{1}{a \cdot s + b} F}$$

Con esto podemos ver que se trata de un sistema de primer orden, lo que es lógico debido a que todos los motores de corriente continua son siempre sistema de primer orden.

El mismo bloque sirve tanto para el rotor principal como para el de cola y debemos buscar una función de transferencia tal que así:

$$G(s) = \frac{K}{\tau \cdot s + 1}$$

Donde K son todas las ganancias del sistema recogidas en una misma como dijimos al principio y τ es la constante de tiempo del sistema.

7.1.3 Esquema Simulink del fabricante FEEDBACK

Para la realización de las simulaciones de los sistemas reales, mostramos el esquema de Simulink que es proporcionado por el fabricante con el modelo PID clásico y posteriormente, la modificación que tuvimos que realizar para introducir el PID fraccionario.

7.1.3.1 Esquema de Control Clásico

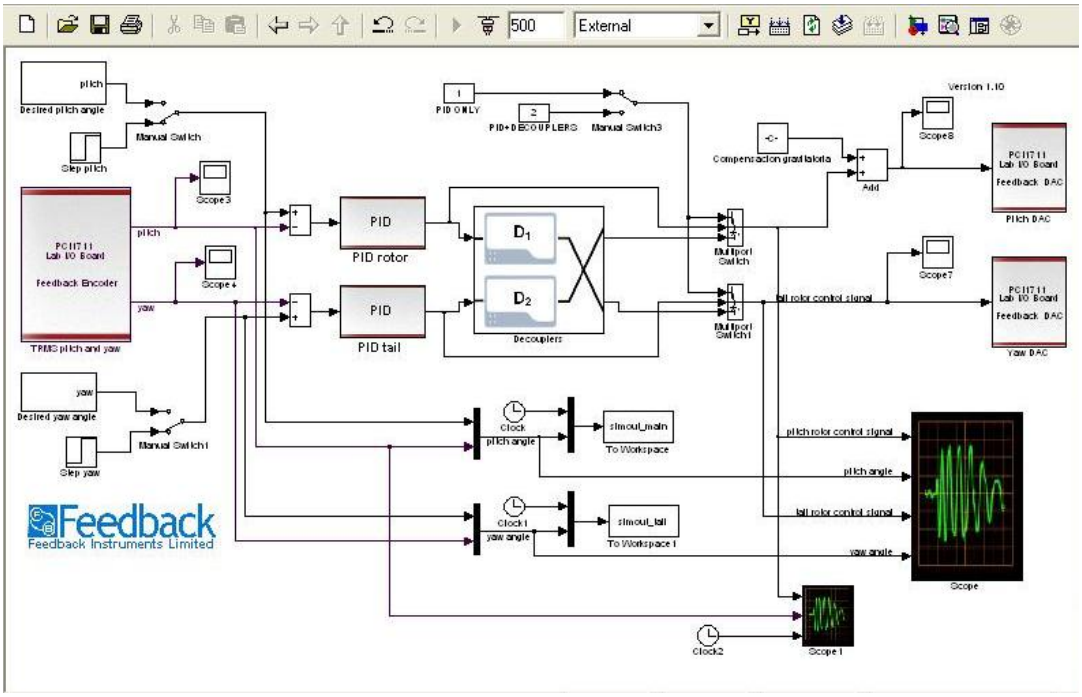

Figura 52. Esquema Simulink Control Clásico

7.1.3.2 Esquema de Control Fraccionario

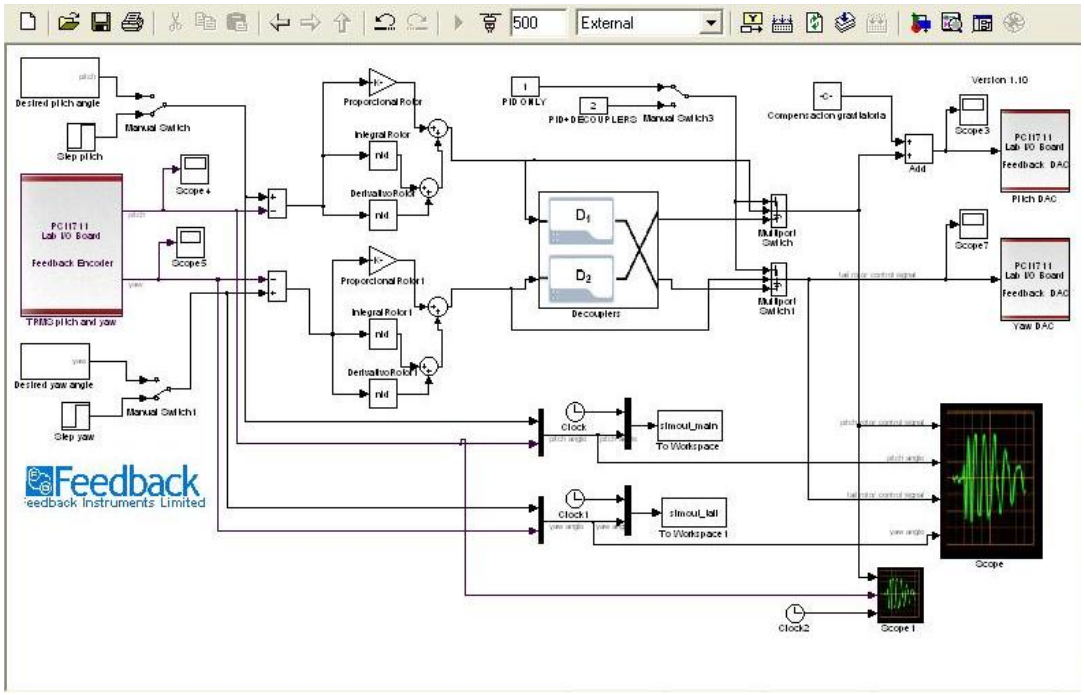

Figura 53. Esquema Simulink Control Clásico

En ambos esquemas de simulación (figuras 52 y 53), tuvimos que añadir la compensación gravitatoria para contrarrestar el peso del rotor principal. Además de añadirle varios Scope para verificar que todo funciona correctamente.

7.1.4 Inconvenientes durante la implementación

A la hora de realizar las simulaciones reales con el TRMS (Twin Rotor MIMO System) se tuvo que lidiar con varios inconvenientes que iban surgiendo según se solucionaba los anteriores.

En un primer momento, tuvimos un problema con el sensor de temperatura que mide el sobrecalentamiento del motor principal. Desde que se encendía el doble rotor, sin haber sido utilizado previamente, se encendía el LED de aviso por calentamiento. Por lo que hubo que desmontar todo el TRMS para comprobar de dónde venía el fallo.

Después, se disponía de una versión de MATLAB 6.5, ya que los drivers que se disponían del sistema eran para esta versión. A la hora de implementar la simulación con el controlador PID clásico no hubo ningún problema, pero al implementar el controlador PID fraccionario nos encontramos con que el bloque *ninteger*, necesario para la acción fraccionaria, no era soportado por versiones inferiores a MATLAB 7.1.

Por ello, se decidió instalar una versión más reciente del programa, para lo que se tuvo que solicitar al suministrador del equipo (*Feedback*) que nos proporcionara los drivers para la nueva versión, lo que llevó un tiempo en conseguir. Por otra parte, debido a la antigüedad del ordenador disponible en el laboratorio y a su poca capacidad de disco, resultaba inviable la instalación de una nueva versión, ya que requería demasiado espacio para el que se disponía, por lo que se procedió a la sustitución del equipo por otro con mayor capacidad y velocidad.

En este nuevo equipo había instalada una versión válida de MATLAB, pero a la hora de conectar el PC con el TRMS, no se detectaba el driver (pese a que estaba correctamente instalado y desde el modelo de *Simulink* sí se detectaba). Buscando cual podía ser el error, se encontró que podía ser que al compilar se mostrara un mensaje de que la compilación era satisfactoria pero realmente no estaba siendo así. Por ello, buscamos qué compiladores se disponían y dimos que sólo estaba el compilador por defecto de MATLAB, lo cual podía ser el error que teníamos.

Se procedió a instalar un compilador diferente, pero el programa seguía sin detectarlo (debía hacerlo automáticamente), por lo que se decidió a reinstalar de nuevo MATLAB, consiguiendo finalmente comunicarnos con el sistema.

Durante las pruebas de sintonización, se detectó que los valores devueltos por el TRMS eran siempre cero (por lo que no estaba existiendo realimentación), de manera que se procedió a "abrir" el sistema y comprobar que todas las conexiones estuvieran correctas y llegasen los valores de tensión adecuados, que los encoder de medición estuviesen debidamente conectados, etc. Durante este proceso, se detectó un fallo que fue corregido, permitiendo poder realizar las simulaciones reales con el sistema.

8. Comparación controlador PID clásico y PID fraccionario.

Uno de los principales objetivos de usar el control fraccionario frente al tradicional, es la idea de introducir dos coeficientes más en nuestra fórmula del controlador para tener más grados de libertad en el sistema. Y así poder ajustar mejor las pendientes en el Diagrama de Bode y tener una mayor flexibilidad en la elección de los parámetros del controlador.

El uso del PID fraccionario nos permite elegir, además de lo que ya podíamos hacer con el PID clásico, las pendientes de la curva de magnitud, y las aportaciones de fase a bajas y altas frecuencias. En la siguiente gráfica, podemos ver los distintos puntos que nos permite alcanzar un controlador frente al otro. Viendo de forma ilustrativa, el PID clásico solo nos dejaría trabajar sobre cuatro puntos (equivalentes a los cuatro vértices de un cuadrado), mientras que el PID fraccionario puede ocupar toda el área de dicho cuadrado definido por los vértices de los puntos posibles en el PID clásico.

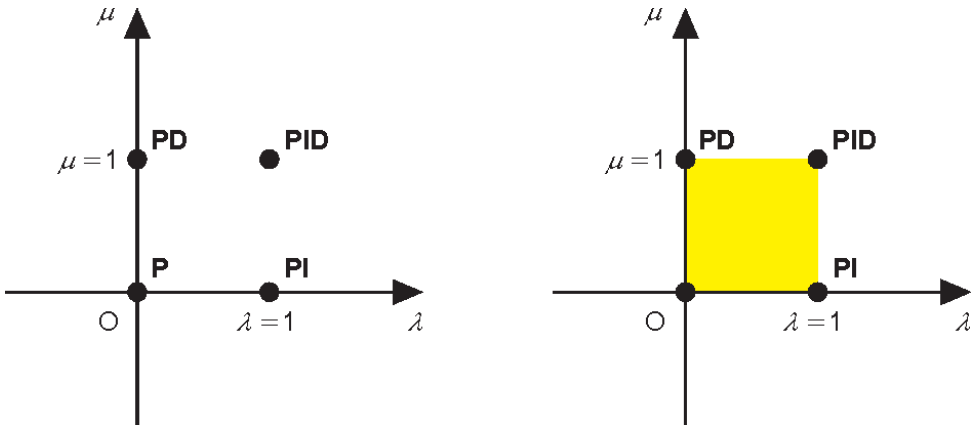

Figura 54. Ilustración gráfica de las aportaciones de un PID tradicional y uno fraccionario

Tras haber realizado y ejecutado los códigos correspondientes para el cálculo de los parámetros de los controladores, tanto el fraccionario como el clásico, realizamos las correspondientes simulaciones. Los resultados se mostrarán en el siguiente apartado.

8.1 Simulaciones de los sistemas para el control fraccionario y control clásico por ordenador:

El sistema sobre el que vamos a trabajar será:

$$G(s) = \frac{2}{s+1}$$

Y el bloque de control creado en Simulink para el fraccionario es el siguiente:

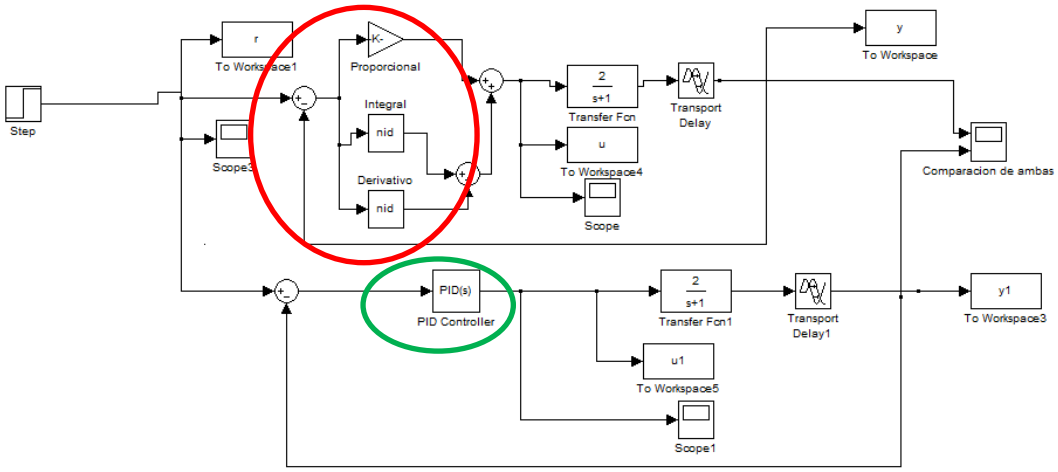

Figura 55. Esquemas prueba de Simulink

El color rojo indica el bloque del PID fraccionario, mientras que el verde el PID clásico.

Donde el bloque *nid*, incorpora los valores que deben tener nuestros índices fraccionarios y las ganancias correspondientes. Para el caso derivativo, nuestro índice se debe poner positivo, mientras que para el integral, el índice se debe poner con un signo negativo.

Las gráficas de la parte alta corresponden a la simulación del control fraccionario mientras que las de la parta baja son de control clásico. Para éste último, siempre utilizaremos los mismos parámetros que son obtenidos a partir del código MATLAB que se adjunta en el apartado 10.

Para el caso PID clásico sólo se realizó un código para el cálculo de los parámetros. Para el fraccionario, se realizaron dos códigos para los dos casos distintos expuestos a continuación.

8.1.1 Cuando $\mu \neq \lambda$:

Tras las gráficas obtenidas en el programa de MATLAB adjunto en el ANEXO V, los valores que sacamos que nos proporcionan unos parámetros adecuados son los siguientes:

- $\lambda = 0.92\ y\ \mu = 0.7317$

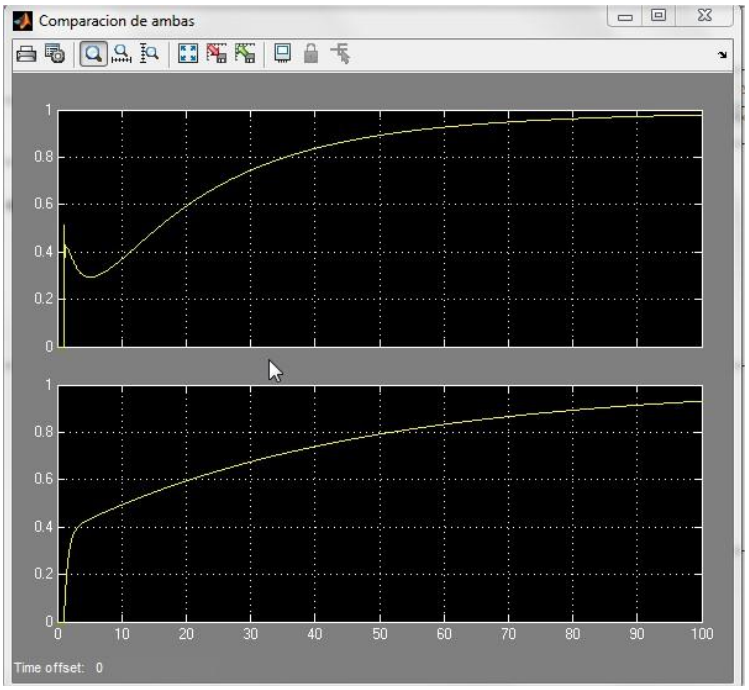

Figura 56. Comparación de simulaciones fraccionario/clásico

- $\lambda = 0.93\ y\ \mu = 0.8079$

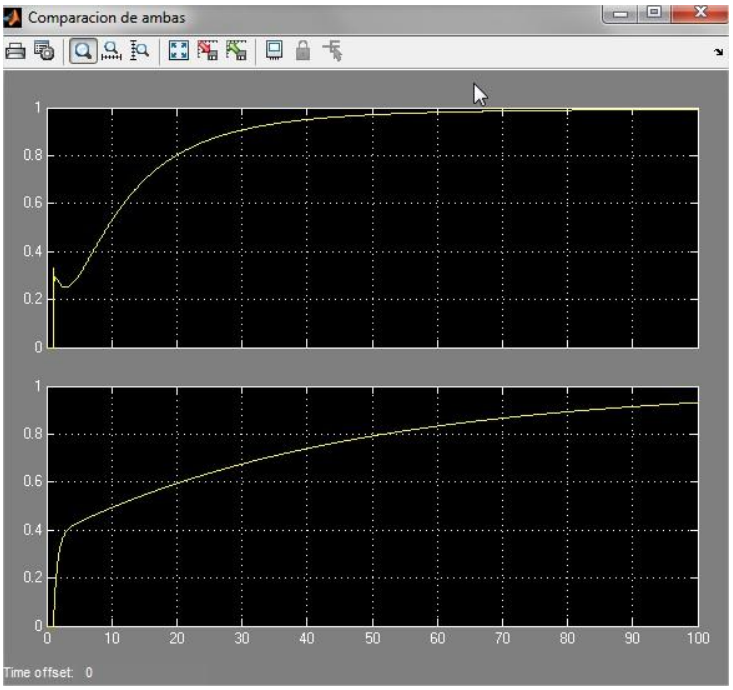

Figura 57. Comparación de simulaciones fraccionario/clásico

- $\lambda = 0.94\ y\ \mu = 0.8869$

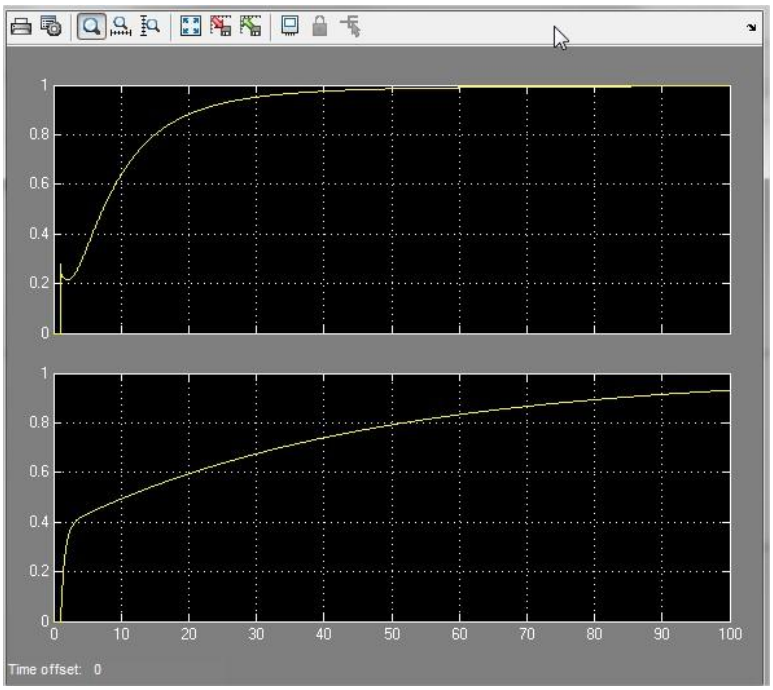

Figura 58. Comparación de simulaciones fraccionario/clásico

- $\lambda = 0.95\ y\ \mu = 0.972$

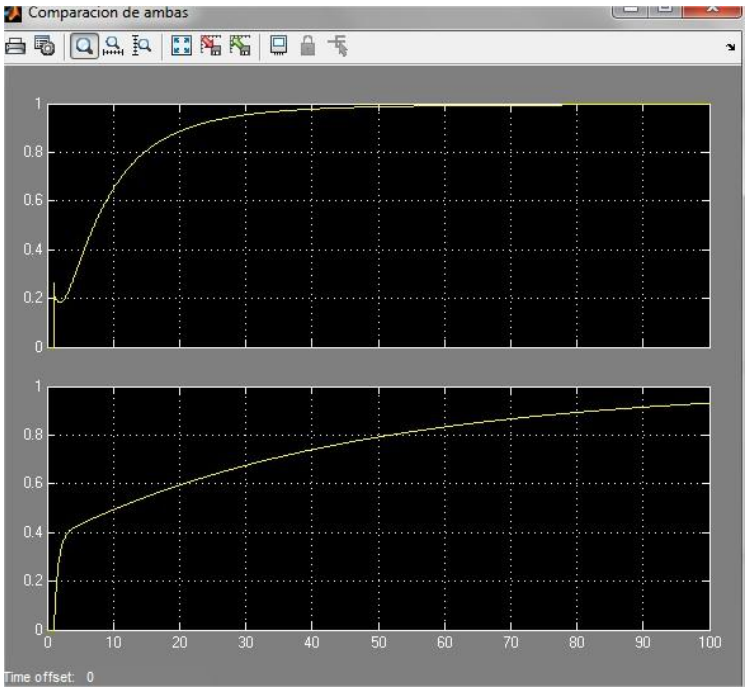

Figura 59. Comparación de simulaciones fraccionario/clásico

Como observamos en las gráficas, con el uso del PID fraccionario se producen pequeños sobrepasamientos en el régimen transitorio, que van disminuyendo a medida que $\lambda\ y\ \mu$ van aumentando, mientras que el PID clásico alcanza el valor de consigna de forma suave y no se producen sobrepasamientos. Sin embargo, el PID fraccionario responde más rápido, llegando mucho más rápido al valor de referencia que el PID clásico.

8.1.2 Cuando $\mu = \lambda$:

Tras las gráficas obtenidas en el programa de MATLAB adjunto en el ANEXO IV, el valor que sacamos y que nos da unos parámetros adecuados es el siguiente:

- $\lambda = \mu = 0.947$

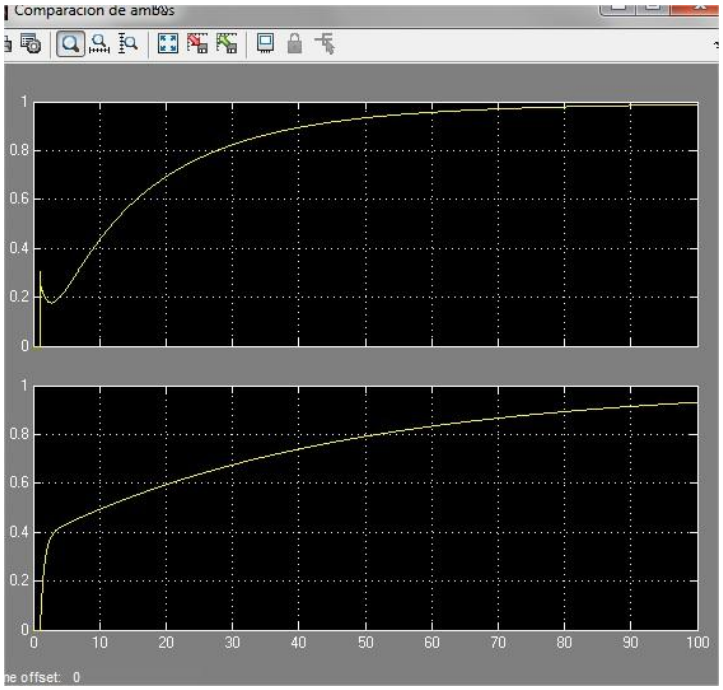

Figura 60. Comparación de simulaciones fraccionario/clásico

En este código solo se obtiene una única solución posible. Observando la gráfica, vemos que el comportamiento se asemeja a los resultados del apartado anterior. En el fraccionario se produce pequeños sobrepasamientos, mientras que el clásico alcanza el valor de consigna de forma suave y sin sobrepasamientos. De la misma manera, el PID fraccionario responde más rápido, llegando en un menor tiempo al valor de referencia que el PID clásico.

8.2 Simulaciones para el control fraccionario y clásico en sistemas reales.

8.2.1 Doble rotor (TRMS)

Sabemos que tanto el sistema del rotor principal como el de la cola se componen por un sistema de primer orden tal que así:

$$G(s) = \frac{K}{\tau \cdot s + 1}$$

Por lo que antes de realizar las simulaciones, sacamos todos los parámetros proporcionados por los códigos para distintos sistemas de primer orden a fin de ajustar mejor el modelo del TRMS.

- $G(s) = \frac{1}{s+1}$

Clásico				
ωc	ωg	Kp	Kd	Ki
1	31,12	x	x	x
3,55	42,22	x	x	x
5,33	99,9	x	x	x
25,26	49,21	x	x	x
27,5	37,65	x	x	x
29,5	64,14	x	x	x
33,63	78,99	x	x	x
90,75	91,21	0,3256	0,278	0,0004
91,95	97,15	0,3161	0,0143	0,0086

Tabla 5. Tabla de datos para Control Clásico

Fraccionario				
λ	μ	Kp	Kd	Ki
0,9567	0,9567	x	x	x
0,98	Sin Corte			
0,97	Sin Corte			
0,96	Sin Corte			
0,95	0,929	0,1594	0,1583	0,0912
0,94	0,881	0,067	0,1979	0,0743
0,93	0,8028	0,0473	0,295	0,0575
0,92	0,7275	0,026	0,5663	0,0345
0,91	0,6557	0,0025	6,3331	0,0035
0,9	0,545	x	x	x
0,85	0,306	x	x	x
0,8	0,1011	x	x	x
0,7	Sin Corte			
0,6	Sin Corte			
0,5	Sin Corte			
0,4	Sin Corte			

Tabla 6. Tabla de datos para Control Fraccionario

- $G(s) = \frac{1}{s+2}$

Clásico				
ωc	ωg	Kp	Kd	Ki
4	45,24	x	x	x
27	42,68	x	x	x
33,8	75,72	x	x	x
92,71	99,95	0,3133	0,005	0,0332

Tabla 7. Tabla de datos para Control Clásico

Fraccionario				
λ	μ	Kp	Kd	Ki
0,9612	0,9612	x	x	x
0,98	Sin Corte			
0,97	Sin Corte			
0,96	Sin Corte			
0,95	0,929	0,1594	0,1583	0,0912
0,94	0,846	0,1216	0,2186	0,0734
0,93	0,7645	0,0809	0,3462	0,0541
0,92	0,6865	0,0361	0,8172	0,0264
0,91	0,6136	x	x	x
0,9	0,545	x	x	x
0,85	0,264	x	x	x
0,8	0,062	x	x	x
0,7	Sin Corte			
0,6	Sin Corte			
0,5	Sin Corte			
0,4	Sin Corte			

Tabla 8. Tabla de datos para Control Fraccionario

- $G(s) = \frac{1}{s+3}$

Clásico				
ωc	ωg	Kp	Kd	Ki
2,55	41,28	x	x	x
27,26	45,47	x	x	x
30,58	30,85	x	x	x
32,91	70,32	x	x	x
92,91	99,99	0,3151	0,0037	0,0589

Tabla 9. Tabla de datos para Control Clásico

Fraccionario				
λ	μ	Kp	Kd	Ki
0,9625	0,9625	x	x	x
0,98	Sin Corte			
0,97	Sin Corte			
0,96	Sin Corte			
0,95	0,9239	0,0648	0,7136	0,003
0,94	0,8408	x	x	x
0,93	0,7578	x	x	x
0,92	0,6817	x	x	x
0,91	0,6076	x	x	x
0,9	0,5345	x	x	x
0,85	0,2503	x	x	x
0,8	Sin Corte			
0,7	Sin Corte			
0,6	Sin Corte			
0,5	Sin Corte			
0,4	Sin Corte			

Tabla 10. Tabla de datos para Control Fraccionario

- $G(s) = \frac{2}{s+1}$

Clásico				
ωc	ωg	Kp	Kd	Ki
3	40,56	x	x	x
5,4	99,98	x	x	x
29,45	29,47	x	x	x
33,28	75,55	x	x	x
92,4	99,36	0,3128	0,009	0,0144

Tabla 11. Tabla de datos para Control Clásico

Fraccionario				
λ	μ	Kp	Kd	Ki
0,942	0,942	0,035	0,1891	0,0378
0,98	Sin Corte			
0,97	Sin Corte			
0,96	Sin Corte			
0,95	0,972	0,0426	0,1482	0,0682
0,94	0,8869	0,0335	0,1981	0,0697
0,93	0,8079	0,0237	0,2948	0,0555
0,92	0,7317	0,0131	0,564	0,0339
0,91	0,6557	0,0012	6,3331	0,0035
0,9	0,5916	x	x	x
0,85	0,3103	x	x	x
0,8	0,1031	x	x	x
0,7	Sin Corte			
0,6	Sin Corte			
0,5	Sin Corte			
0,4	Sin Corte			

Tabla 12. Tabla de datos para Control Fraccionario

- $G(s) = \frac{2}{s+2}$

Clásico				
ωc	ωg	Kp	Kd	Ki
0,5	34,43	x	x	x
1,47	37,22	x	x	x
2,96	41,52	x	x	x
4,13	46,13	x	x	x
25,93	99,78	x	x	x
32,98	72,15	x	x	x
92,38	98,28	0,3167	0,0108	0,0118
92,71	99,94	0,3138	0,0065	0,0221

Tabla 13. Tabla de datos para Control Fraccionario

Fraccionario				
λ	μ	Kp	Kd	Ki
0,9612	0,9612	x	x	x
0,98	Sin Corte			
0,97	Sin Corte			
0,96	Sin Corte			
0,95	0,9349	0,0482	0,2935	0,0108
0,94	0,845	x	x	x
0,93	0,7718	x	x	x
0,92	0,6927	x	x	x
0,91	0,6196	x	x	x
0,9	0,5506	x	x	x
0,85	0,2646	x	x	x
0,8	0,06607	x	x	x
0,7	Sin Corte			
0,6	Sin Corte			
0,5	Sin Corte			
0,4	Sin Corte			

Tabla 14. Tabla de datos para Control Fraccionario

- $G(s) = \frac{2}{s+3}$

Clásico				
ωc	ωg	Kp	Kd	Ki
0,5	35,17	x	x	x
3,53	44,98	x	x	x
26,67	99,98	x	x	x
27,17	46,08	x	x	x
30	33,7	x	x	x
30,65	64,19	x	x	x
34,64	79,05	x	x	x
91,4	92,54	0,3286	0,0949	0,0013
92	95,5	0,3236	0,0233	0,0052
92,91	99,98	0,3157	0,0052	0,0309

Tabla 15. Tabla de datos para Control Clásico

Fraccionario				
λ	μ	Kp	Kd	Ki
0,9626	0,9626	x	x	x
0,98	Sin Corte			
0,97	Sin Corte			
0,96	Sin Corte			
0,95	0,9179	0,1166	0,1625	0,0899
0,94	0,8328	0,088	0,2267	0,0735
0,93	0,751	0,057	0,3682	0,0526
0,92	0,6727	0,0229	0,9667	0,023
0,91	0,5993	x	x	x
0,9	0,5305	x	x	x
0,85	0,2492	x	x	x
0,8	0,485	x	x	x
0,7	Sin Corte			
0,6	Sin Corte			
0,5	Sin Corte			
0,4	Sin Corte			

Tabla 16. Tabla de datos para Control Fraccionario

Se ajustó las funciones de transferencia para cada motor, para el rotor principal la función de transferencia es $G(s) = \frac{2}{s+2}$, mientras que para el de cola es $G(s) = \frac{2}{s+1}$

Las observaciones destacadas de la implementación real son:

El control clásico es más sensible que el fraccionario frente a perturbaciones. En el ángulo *pitch*, el control fraccionario sufre menos oscilaciones que el clásico; mientras que en el ángulo *yaw*, el fraccionario presenta más oscilaciones que el clásico. Aparentemente el controlador PID fraccionario se estabiliza antes que el clásico.

Las gráficas resultantes de las simulaciones son las siguientes:

- Para el control PID clásico:

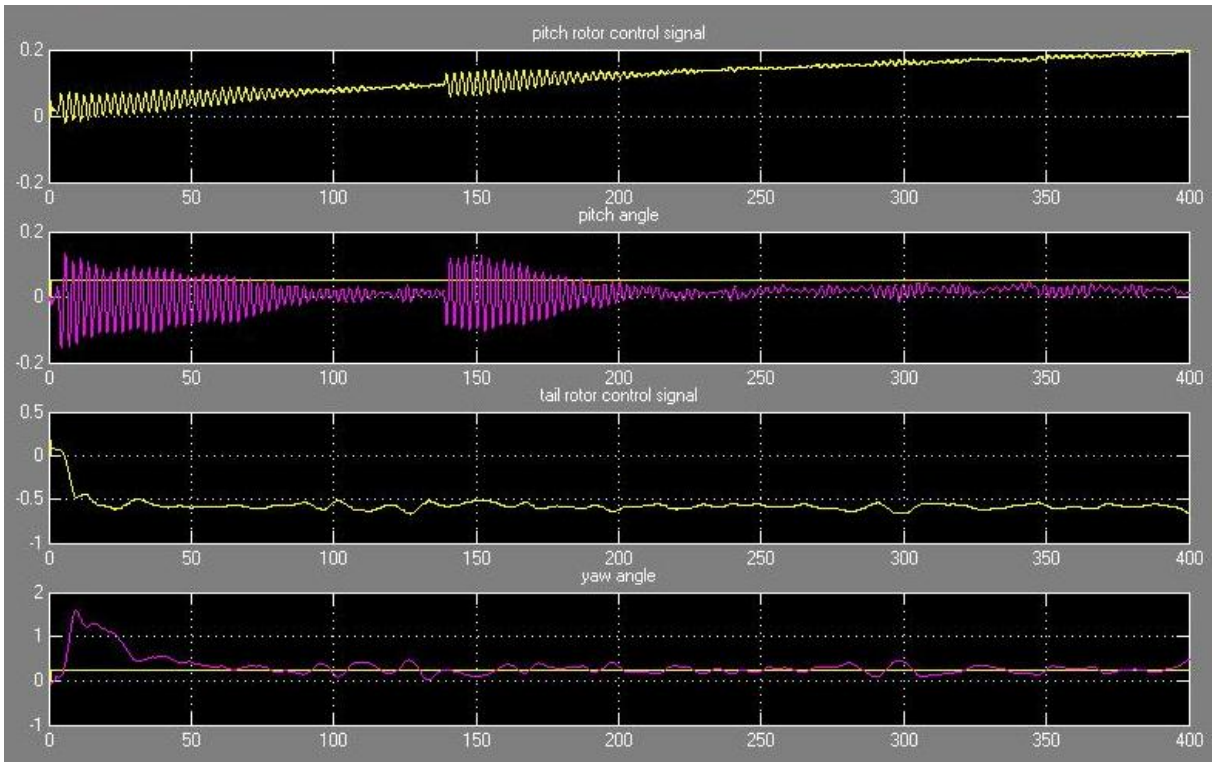

Figura 61. Gráfica Simulación Real Control Clásico con perturbación

- Para el control PID fraccionario:

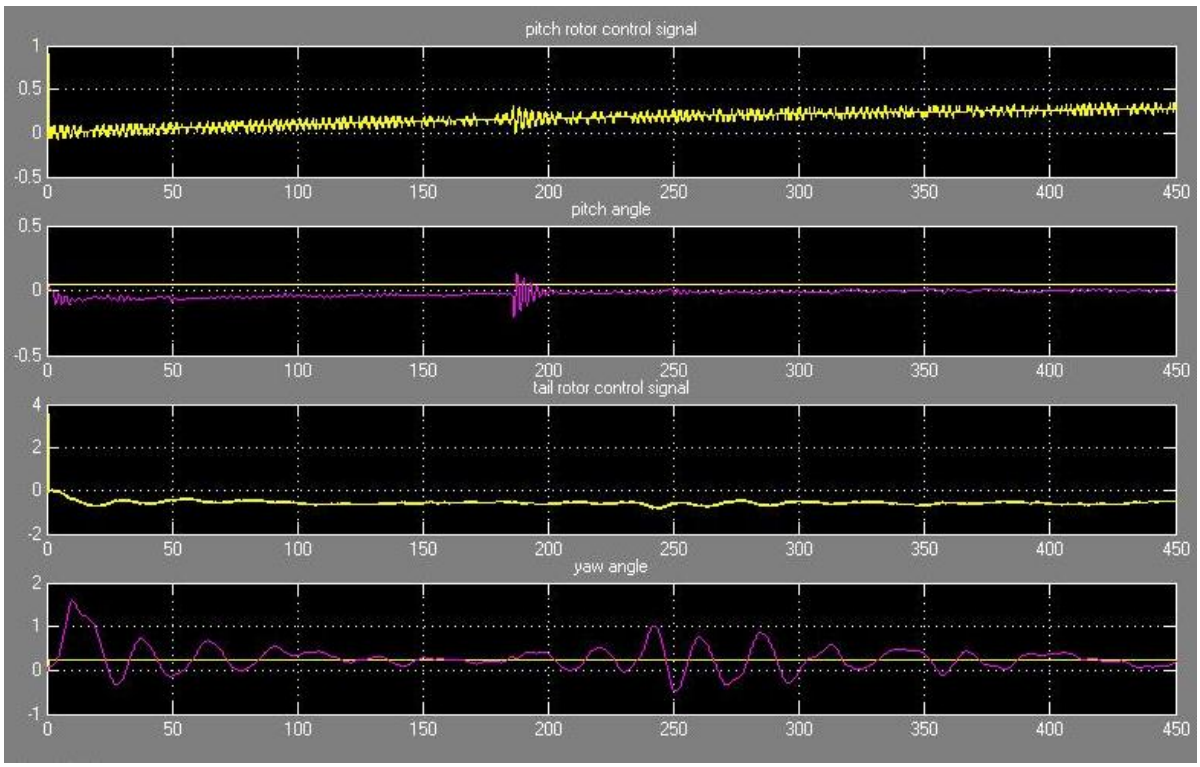

Figura 62. Gráfica Simulación Real Control Fraccionario con perturbación

9. Calculo teórico de pseudoestados fraccionarios.

Como propuesta alternativa en la sintonización de un controlador se ha buscado la implementación de un PI a partir de subvariables de estado fraccionarias.

El método de variables de estado permite mayor flexibilidad a la hora de aplicarlo a los sistemas, pues al tratarse de ecuaciones diferenciales de primer orden se puede aplicar tanto a sistemas lineales como no lineales, además de que no deben suponerse las condiciones iniciales nulas (cosa que sí debe hacerse con el método de la función de transferencia).

Las variables de estado de un sistema podrían definirse como aquel conjunto de variables $x_1(t)$, $x_2(t)$,…, $x_n(t)$ tal que su conocimiento en cualquier instante de tiempo y la información de la entrada posterior son suficientes para determinar las variables de estado para cualquier instante de tiempo posterior. Así mismo, las variables de estado deben cumplir que para cualquier instante de tiempo definan el estado inicial del sistema, pues de otra manera sería imposible proceder a su control.

Si suponemos un sistema LTI (Linear Time Invariant), sistema lineal invariante en el tiempo, representado por:

$$y^n(t) + a_1 y^{n-1}(t) + \cdots + a_{n-1}\dot{y}(t) + a_n y(t) = u(t) \tag{9.1}$$

Si tomamos como variables de estado las salidas y sus derivadas:

$$\begin{aligned} x_1 &= y \\ x_2 &= \dot{y} \\ &\vdots \\ x_n &= y^{n-1} \end{aligned} \tag{9.2}$$

De manera que sus derivadas quedan:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ &\vdots \\ \dot{x}_n &= -a_n x_1 - a_{n-1} x_2 - \cdots - a_2 x_{n-1} - a_1 x_n + u(t) \end{aligned} \tag{9.3}$$

Aplicando el teorema de Lagrange queda de la siguiente manera:

$$(9.4) \quad \begin{aligned} sX_1 &= X_2 \\ sX_2 &= X_3 \\ &\vdots \end{aligned}$$

Al establecer integradores fraccionarios $\alpha_1, \alpha_2, \ldots, \alpha_n$ tales que:

$$(9.5) \quad \propto_1 + \propto_2 + \cdots + \propto_n = 1$$

Se pueden expresar diferentes subestados fraccionarios, dependiendo del número '*n*' que se elija. De esta manera, si establecemos como entrada del sistema la variable '*u*' y como salida la variable '*y*', con $y_1, y_2, \ldots y_n$ como salidas intermedias, se obtiene la siguiente secuencia:

$$(9.6) \quad \begin{aligned} y_1 &= s^{-\propto 1} u \\ y_2 &= s^{-\propto 2}\, y_1 \\ &\vdots \\ y_{n-1} &= s^{-\propto n-1}\, y_{n-2} \\ y_n &= s^{-\propto n}\, y_{n-1} \end{aligned}$$

El diagrama de bloques para el controlador implementado quedaría de la siguiente manera:

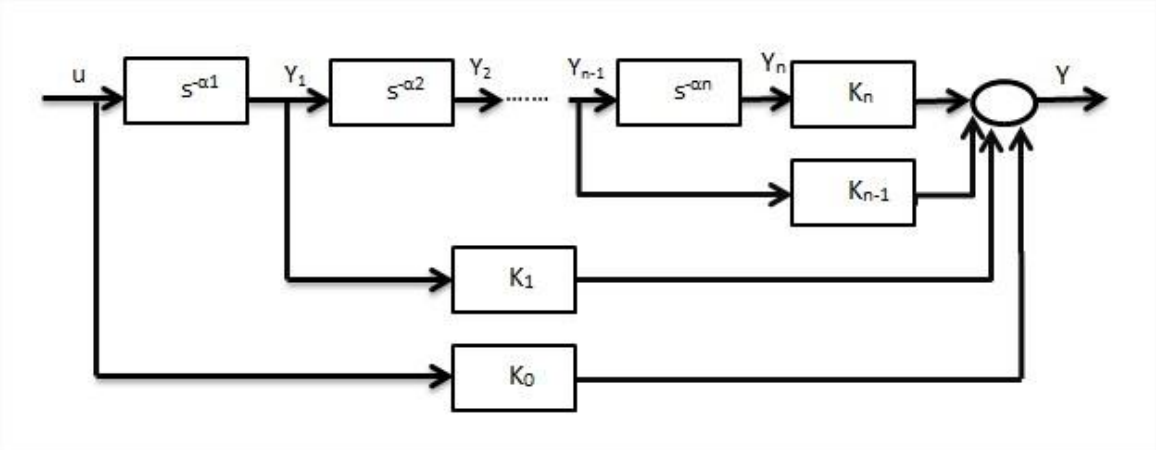

Figura 63. Diagrama de bloque generalizado para pseudoestados fraccionarios

Donde K_0, K_1,…, K_n son las ganancias proporcionales para cada uno de los subestados.

Como ejemplo de resolución, se han elegido tres subestados (n=3), por lo que el diagrama del controlador queda:

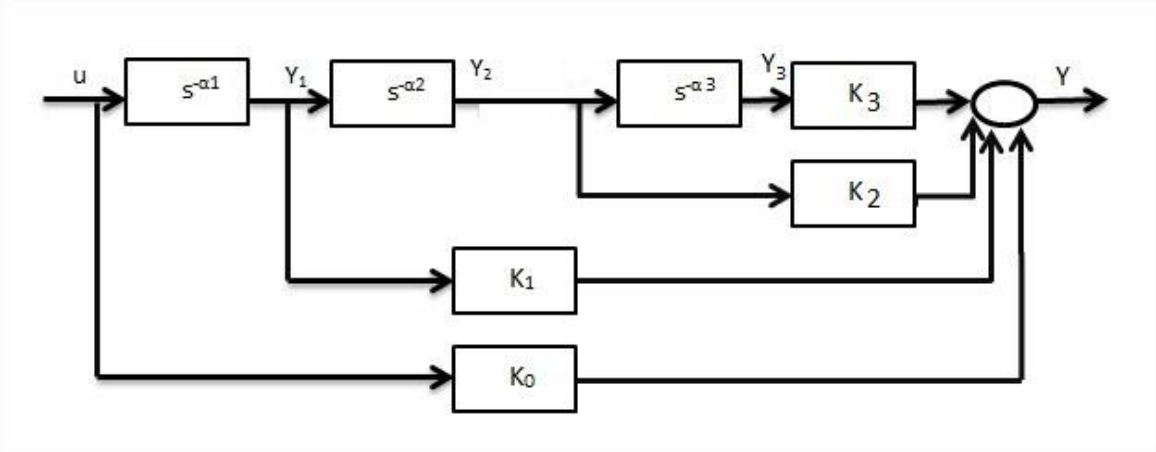

Figura 64. Diagrama de bloque para n=3 de los pseudoestados fraccionarios

Al resolver el diagrama se obtiene:

$$(9.7) \quad Y = K_3Y_3 + K_2Y_2 + K_1Y_1 + K_0u$$

Que puede desarrollarse tal que:

$$(9.8) \quad \begin{aligned} y_1 &= s^{-\propto 1}u \\ y_2 &= s^{-\propto 2}\, y_1 = s^{-\propto 2}s^{-\propto 1}u \\ y_3 &= s^{-\propto 3}\, y_3 = s^{-\propto 3}s^{-\propto 2}s^{-\propto 1}u \end{aligned}$$

Lo que lleva a la siguiente expresión:

$$(9.9) \quad Y = K_3s^{-\propto 3}s^{-\propto 2}s^{-\propto 1}u + K_2s^{-\propto 2}s^{-\propto 1}u + K_1s^{-\propto 1}u + K_0u$$

Que haciendo la relación Y/u dará la expresión del controlador:

$$(9.10) \quad \frac{Y}{u} = R(S) = K_0 + \frac{K_1}{S^{\propto 1}} + \frac{K_2}{S^{(\propto 1+\propto 2)}} + \frac{K_3}{S^{(\propto 1+\propto 2+\propto 3)}}$$

Sabemos que S=jω, por lo que sustituyendo en R(S) tenemos:

$$(9.11) \quad R(\mathrm{j}\omega) = K_0 + \frac{K_1}{(\mathrm{j}\omega)^{\propto 1}} + \frac{K_2}{(\mathrm{j}\omega)^{(\propto 1+\propto 2)}} + \frac{K_3}{(\mathrm{j}\omega)^{(\propto 1+\propto 2+\propto 3)}}$$

Además $j\omega = \omega e^{j\frac{\pi}{2}}$, sustituyendo en la ecuación (9.11) y situando los parámetros del denominador en el numerador para operar de forma más sencilla nos queda:

$$(9.12) \quad R(j\omega) = K_3\left(\omega e^{j\frac{\pi}{2}}\right)^{-\propto 3}\left(\omega e^{j\frac{\pi}{2}}\right)^{-\propto 2}\left(\omega e^{j\frac{\pi}{2}}\right)^{-\propto 1} + K_2\left(\omega e^{j\frac{\pi}{2}}\right)^{-\propto 2}\left(\omega e^{j\frac{\pi}{2}}\right)^{-\propto 1} + K_1\left(\omega e^{j\frac{\pi}{2}}\right)^{-\propto 1} + K_0$$

A continuación vamos operando los términos de la ecuación para poder pasarlo a forma polar:

$$(9.13)\ R(j\omega) = K_3\left(\omega^{-\propto 3}e^{-j\frac{\pi}{2}\propto 3}\right)\left(\omega^{-\propto 2}e^{-j\frac{\pi}{2}\propto 2}\right)\left(\omega^{-\propto 1}e^{-j\frac{\pi}{2}\propto 1}\right) + K_2\left(\omega^{-\propto 2}e^{-j\frac{\pi}{2}\propto 2}\right)\left(\omega^{-\propto 1}e^{-j\frac{\pi}{2}\propto 1}\right) + K_1\left(\omega^{-\propto 1}e^{-j\frac{\pi}{2}\propto 1}\right) + K_0$$

$$(9.14)\ R(j\omega) = K_3(\omega^{-\propto 3-\propto 2-\propto 1})e^{-j\frac{\pi}{2}(\propto 3+\propto 2+\propto 1)} + K_2(\omega^{-\propto 2-\propto 1})e^{-j\frac{\pi}{2}(\propto 2+\propto 1)} + K_1(\omega^{-\propto 1})e^{-j\frac{\pi}{2}\propto 1} + K_0$$

Ahora pasaremos a forma polar, $e^{j\omega} = \cos(\omega) + jsin(\omega)$, por lo que finalmente la fórmula del controlador será:

(9.15)

$$R(j\omega) = \frac{K_3}{\omega^{\propto 3+\propto 2+\propto 1}} \cdot \left[\cos\left(-\frac{\pi}{2}(\propto_3+\propto_2+\propto_1)\right) + j\sin\left(-\frac{\pi}{2}(\propto_3+\propto_2+\propto_1)\right)\right] + \frac{K_2}{\omega^{\propto 2+\propto 1}} \cdot \left[\cos\left(-\frac{\pi}{2}(\propto_2+\propto_1)\right) + j\sin\left(-\frac{\pi}{2}(\propto_2+\propto_1)\right)\right] + \frac{K_1}{\omega^{\propto 1}} \cdot \left[\cos\left(-\frac{\pi}{2}\propto_1\right) + j\sin\left(-\frac{\pi}{2}\propto_1\right)\right] + K_0$$

Separando la parte real de la imaginaria y con equivalencias trigonométricas de los signos de senos y cosenos nos quedamos con que:

(9.16)

$$R_R(j\omega) = \frac{K_3}{\omega^{\propto 3+\propto 2+\propto 1}} \cdot \cos\left(\frac{\pi}{2}(\propto_3+\propto_2+\propto_1)\right) + \frac{K_2}{\omega^{\propto 2+\propto 1}} \cdot \cos\left(\frac{\pi}{2}(\propto_2+\propto_1)\right) + \frac{K_1}{\omega^{\propto 1}} \cdot \cos\left(\frac{\pi}{2}\propto_1\right) + K_0$$

(9.17)

$$R_I(j\omega) = -\frac{K_3}{\omega^{\propto 3+\propto 2+\propto 1}} \cdot \sin\left(\frac{\pi}{2}(\propto_3+\propto_2+\propto_1)\right) - \frac{K_2}{\omega^{\propto 2+\propto 1}} \cdot \sin\left(\frac{\pi}{2}(\propto_2+\propto_1)\right) - \frac{K_1}{\omega^{\propto 1}} \cdot \sin\left(\frac{\pi}{2}\propto_1\right)$$

Como propuesta de método de resolución para la sintonización de este controlador, emplearemos las mismas especificaciones de diseño que los apartados 4 y 5 de la memoria, donde todos los parámetros, salvo K_3, K_2, K_1 y K_0, son conocidos. A continuación se muestra la matriz A preparada para la sintonización:

(9.18) $$A = \begin{pmatrix} R_R(\omega_c) \\ R_I(\omega_c) \\ R_R(\omega_g) \\ R_I(\omega_g) \end{pmatrix} =$$

$$= \begin{pmatrix} \frac{K_3}{\omega_c{}^{\propto 3+\propto 2+\propto 1}} \cdot \cos\left(\frac{\pi}{2}(\propto_3+\propto_2+\propto_1)\right) + \frac{K_2}{\omega_c{}^{\propto 2+\propto 1}} \cdot \cos\left(\frac{\pi}{2}(\propto_2+\propto_1)\right) + \frac{K_1}{\omega_c{}^{\propto 1}} \cdot \cos\left(\frac{\pi}{2}\propto_1\right) + K_0 \\ -\frac{K_3}{\omega_c{}^{\propto 3+\propto 2+\propto 1}} \cdot \sin\left(\frac{\pi}{2}(\propto_3+\propto_2+\propto_1)\right) - \frac{K_2}{\omega_c{}^{\propto 2+\propto 1}} \cdot \sin\left(\frac{\pi}{2}(\propto_2+\propto_1)\right) - \frac{K_1}{\omega_c{}^{\propto 1}} \cdot \sin\left(\frac{\pi}{2}\propto_1\right) \\ \frac{K_3}{\omega_g{}^{\propto 3+\propto 2+\propto 1}} \cdot \cos\left(\frac{\pi}{2}(\propto_3+\propto_2+\propto_1)\right) + \frac{K_2}{\omega_g{}^{\propto 2+\propto 1}} \cdot \cos\left(\frac{\pi}{2}(\propto_2+\propto_1)\right) + \frac{K_1}{\omega_g{}^{\propto 1}} \cdot \cos\left(\frac{\pi}{2}\propto_1\right) + K_0 \\ -\frac{K_3}{\omega_g{}^{\propto 3+\propto 2+\propto 1}} \cdot \sin\left(\frac{\pi}{2}(\propto_3+\propto_2+\propto_1)\right) - \frac{K_2}{\omega_g{}^{\propto 2+\propto 1}} \cdot \sin\left(\frac{\pi}{2}(\propto_2+\propto_1)\right) - \frac{K_1}{\omega_g{}^{\propto 1}} \cdot \sin\left(\frac{\pi}{2}\propto_1\right) \end{pmatrix}$$

Finalmente se plantea de esta manera:

$$\begin{pmatrix} \frac{\cos\left(\frac{\pi}{2}(\propto_3+\propto_2+\propto_1)\right)}{\omega_c{}^{\propto 3+\propto 2+\propto 1}} & \frac{\cos\left(\frac{\pi}{2}(\propto_2+\propto_1)\right)}{\omega_c{}^{\propto 2+\propto 1}} & \frac{\cos\left(\frac{\pi}{2}\propto_1\right)}{\omega_c{}^{\propto 1}} & 1 \\ \frac{-\sin\left(\frac{\pi}{2}(\propto_3+\propto_2+\propto_1)\right)}{\omega_c{}^{\propto 3+\propto 2+\propto 1}} & \frac{-\sin\left(\frac{\pi}{2}(\propto_2+\propto_1)\right)}{\omega_c{}^{\propto 2+\propto 1}} & \frac{-\sin\left(\frac{\pi}{2}\propto_1\right)}{\omega_c{}^{\propto 1}} & 0 \\ \frac{\cos\left(\frac{\pi}{2}(\propto_3+\propto_2+\propto_1)\right)}{\omega_g{}^{\propto 3+\propto 2+\propto 1}} & \frac{\cos\left(\frac{\pi}{2}(\propto_2+\propto_1)\right)}{\omega_g{}^{\propto 2+\propto 1}} & \frac{\cos\left(\frac{\pi}{2}\propto_1\right)}{\omega_g{}^{\propto 1}} & 1 \\ \frac{-\sin\left(\frac{\pi}{2}(\propto_3+\propto_2+\propto_1)\right)}{\omega_g{}^{\propto 3+\propto 2+\propto 1}} & \frac{-\sin\left(\frac{\pi}{2}(\propto_2+\propto_1)\right)}{\omega_g{}^{\propto 2+\propto 1}} & \frac{-\sin\left(\frac{\pi}{2}\propto_1\right)}{\omega_g{}^{\propto 1}} & 0 \end{pmatrix} \begin{pmatrix} K_3 \\ K_2 \\ K_1 \\ K_0 \end{pmatrix}$$

10. Explicación de los códigos

10.1 Códigos Matlab

Pese a que se ha intentado generar un código lo más explicativo posible, se considera necesaria una explicación más detallada del mismo.

En este apartado no se pretende exponer por duplicado todos los cálculos teóricos realizados hasta llegar a la creación de este código, por lo que se harán con asiduidad referencias a los distintos puntos de la memoria del proyecto, donde se encuentra explicado detalladamente el procedimiento seguido para las deducciones y cálculos llevados a cabo.

10.1.1 Función de transferencia sin retardo

F_transferencia_SinRetardo

El objeto de esta función es generar una función de transferencia [G(s)] a partir de los valores de sus términos ordinales (a, b, c, d, e) para una frecuencia (ω_c) determinada. Así mismo, también devuelve como parámetros de salida el módulo y la fase de la función de transferencia generada.

Pese a que en el programa MATLAB existe un comando que nos permite obtener una función de transferencia, se ha determinado que era aconsejable generar esta función para poder hacer un uso más práctico de ella, ya que con la función implementada en el programa se genera una función de transferencia simbólica, la cual limita bastante su utilización para cálculos posteriores.

Esta implementación no conlleva mucha complejidad, por lo que con los comentarios establecidos en el código es más que suficiente para su entendimiento.

```
function [G, Gwcmod, Gwcp] = F_transferencia_SinRetardo(a,b,c,d,e, wc)

% se "genera" el término de Lagrange (s = jw) con la frecuencia deseada
    s = wc*1i;
% Definimos el numerador de la función de transferencia:
%  As + B
    num = a*s + b;
% Definimos el denominador de la función de transferencia:
%  Cs^2 + Ds + E
    den = c*(s).^2 + d*s + e;

% Generamos la función de transferencia deseada:

%             As + B
% G(s) = -----------------
%          Cs^2 + Ds + E

    G = num/den


 % Calculamos el modulo de la funcion de transferencia del sistema
    Gwcmod=abs(G)
 % y su fase
    Gwcp=angle(G)

end
```

10.1.2 PID Clásico

clasicoGrafica2

El objetivo de este programa es obtener los parámetros de control para un PID clásico (Kp, Td y Ti) a partir de las especificaciones de diseño de su frecuencia de cruce (wc),que en nuestro caso se entiende por una frecuencia de cruce provisional, pues a partir de los resultados gráficos obtenidos se solicitará al usuario una nueva frecuencia más acorde al sistema en cuestión; su margen de fase (Øm, nombrado en el programa como Fim) y su margen de ganancia (Mg), así como el retardo del sistema (t).

En un principio, se le pide al usuario que introduzca los valores deseados para la función de transferencia especificada, calculando esta, su módulo y su fase con una llamada a la función F_transferencia_SinRetardo explicada anteriormente.

```
function [Kp, Td, Ti]= clasicoGrafica2 (wc1, Mg, Fim, t)

% Introducimos los valores de la funcion de transferencia del sistema
        disp('Sabiendo que la funcion de transferencia tiene el formato siguiente');
        disp('       As + B       ');
        disp('--------------------');
        disp('   Cs^2 + Ds + E     ');
      a = input('Ingrese el valor de la variable a: ');
      b = input('Ingrese el valor de la variable b: ');
      c = input('Ingrese el valor de la variable c: ');
      d = input('Ingrese el valor de la variable d: ');
      e = input('Ingrese el valor de la variable e: ');

 % Llamamos a la función generada para que nos cree la función de
 % transferencia deseada y nos proporcione su fase (Gwcp) y su módulo (Gwcmod)
    [G, Gwc1mod, Gwc1p] = F_transferencia_SinRetardo(a,b,c,d,e, wc1);
```

A continuación, se realiza el bucle necesario para la creación de las gráficas de g (w_c) y h (w_g) *(4.16)*. Primero, comentar que el análisis se realiza hasta una frecuencia de 100 radianes, con una resolución de 0.1 a la hora de realizar la representación gráfica, por lo que el bucle se finaliza en 99.9 para obtener 1000 puntos de representación.

En el interior del bucle se genera de nuevo la función de transferencia, esta vez de una manera desglosada y no realizando la llamada a la función correspondiente, debido a que su valor irá variando para las diferentes frecuencias muestreadas. Así mismo, se calcula el módulo y la fase de ella, pues son necesarios para la representación de las funciones g (w_c) y h (w_g), establecidas en el código como F1B y F3B respectivamente (en alusión a la fila de la matriz B (4.14) que hacen referencia). Estas variables se definen como vectores, de manera que al ir aumentando de iteración se van añadiendo nuevos valores al vector, obteniendo a la finalización del bucle los puntos necesarios para la representación gráfica deseada.

```
% Bucle para creación de gráfica relativa a wc(F1B) y wg(F3B)

 aa=1; % inicialización de los vectores donde se calcularán las ecuaciones

  for k = 0:0.1:(99.9) % se pone hasta 99.9 para que los vectores generados
                       %   sean de la misma longitud

      s = k*1i;

  numk = a*s + b;
  denk = c*(s)^2 + d*s + e;

  Gwk = (numk/denk);
  % Nos quedamos con el modulo de la funcion de transferencia del
  % sistema y su fase
  Gwkmod=abs(Gwk);
  Gwkp=angle(Gwk);

  % Calculo del valor de F1B(grafica dependiente de wc)
  F1B(aa) = [(cos(Fim + t*k - Gwkp)/Gwkmod)];
  % Calculo del valor de F3B(grafica dependiente de wg)
  F3B(aa) = [cos(t*k - Gwkp + t*k)/(Mg*Gwkmod)];

  % aumentamos el indice del vector
  aa=aa+1;
  end  % finalizacion del bucle for
```

A la hora de la representación gráfica, se ha decidido realizar dos niveles de resolución para una mejor visualización, debido a que la representación gráfica de h (w_g) tiene valores inferiores a la de g (w_c).

En la parte superior [*subplot (2, 1, 1)*] se representa una visión más amplia de ambas funciones, mientras que en la inferior [*subplot (2, 1, 2)*] se establece el límite de los ejes de la gráfica a tres veces el mínimo y máximo valor de la función h (w_g), F3B en el código.

El resto de código en este subapartado consiste en el tratamiento de la forma en la que se muestra la gráfica, como puede ser la etiqueta de los ejes (mediante el comando *xlabel*) o la inclusión de una leyenda para identificar cada función.

```
% REPRESENTACION GRAFICA

% Se divide en dos gráficas
% una primera donde se comparan las dos ecuaciones y se marca la wc
% elegida
  subplot(2,1,1)

  w = linspace (0,100,1000); % generamos 1.000 ptos entre 0 y 100
% se calcula mínimo y maximo para trazar la recta en la wc deseada
   minimo = min(F1B);
   maximo = max(F1B);
 y = linspace (minimo,maximo,1000); %generación de vector vertical para
                                    %marcar frecuencia wc

 plot(w,F1B,'r',w,F3B,wc1,y,'k')
 xlabel('Frecuencia');
 legend('g = gráfica dependiente de wc','h = gráfica dependiente de wg');

 % segunda gráfica donde se "amplia" la representación de la ecuacion
 % dependiente de wg (generalmente de valores más pequeños)para una mejor
 % visualización
  subplot(2,1,2)

 plot(w,F1B,'r',w,F3B)
 % se calcula mínimo y maximo de F3B para establecer límite de la gráfica
   mini = min(F3B);
   maxi = max(F3B);
 axis([ 0 100 (3*mini) (3*maxi)]);
```

Como una ayuda extra para el usuario a la hora de decidir qué frecuencia de cruce (w_c) elegir para el sistema en cuestión, se ha buscado obtener las posibles frecuencias válidas para el mismo. Es decir, frecuencias cuyo valor en $g(w_c)$ tenga un igual en $h(w_g)$ a frecuencias mayores, correspondiendo la frecuencia a la que ocurre con el valor de w_g que se necesita para cumplir las especificaciones de diseño.

Para ello, se implementa la siguiente parte del código, donde se genera un bucle que en este caso tendrá una resolución de 0.01, de manera que pueda hacer un mayor barrido de las frecuencias y obtenerse mejores resultados. Se crea de nuevo la función de transferencia al ir modificándose en cada iteración y se calcula el valor de $g(w_c)$[F1B en el código generado], redondeando su valor a dos decimales para optimizar el tiempo de procesamiento del código.

Obtenido este valor, se entra en un nuevo bucle a partir de la frecuencia en cuestión a la que nos encontremos y se realiza el mismo procedimiento pero con la función de $h(w_g)$[en el código F3B]. Una vez calculado su valor, se realiza una

comparación con el de F1B, de manera que si son iguales se guardará en una matriz (M) el valor de la w_c (que equivale a la 'q' del código) y de w_g (cuyo equivalente será 'n' del código). Si no fuesen iguales los valores, se continuará a la siguiente iteración para proseguir con las comparaciones, de manera que al finalizar el bucle tendremos en la matriz (M) las parejas de w_c y w_g que son válidas para el sistema que se desea.

```
% BUCLE PARA OBTENCION DE WCs POSIBLES Y WG QUE SE TENDRÍAN
%
%
 v = 1; % inicializacion para la matriz donde se guardarán las wc
        % y su correspondiente wg

for q = 0:0.01:(99.99)

    s = q*1i;
numq = a*s + b;
denq = c*(s)^2 + d*s + e;

Gwq = (numq/denq);

% Nos quedamos con el modulo de la funcion de transferencia del
% sistema y su fase
Gwqmod=abs(Gwq);
Gwqp=angle(Gwq);

% Calculo del valor de F1B(grafica dependiente de wc)
F1B = [(cos(Fim + t*q - Gwqp)/Gwqmod)];
% Redondeamos a 2 numeros decimales
f1b = round(F1B*10^2)/10^2
```

```
for n = q:0.01:(99.99)   % bucle que recorrerá desde el valor de la frecuencia
                         % de wc hacia valores más altos
    s = n*1i;
numn = a*s + b;
denn = c*(s)^2 + d*s + e;

Gwn = (numn/denn);
% Nos quedamos con el modulo de la funcion de transferencia del
% sistema y su fase
Gwnmod=abs(Gwn);
Gwnp=angle(Gwn);
% Calculo del valor de F3B(grafica dependiente de wg)
F3B = [cos(t*n - Gwnp + t*n)/(Mg*Gwnmod)];
f3b = round(F3B*10^2)/10^2;

if ( f1b == f3b )
    M(v,1) = q;
    M(v,2) = n;
     v = v+1;
     break;
else
    n = n + 0.01;
end
end  % finalizacion bucle for(n)
 q = q + 0.01;
end  % finalizacion bucle for(q)
% mostramos la matriz generada, donde aparecera en la primera columna
% los valores de Wc y en la segunda su Wg correspondiente
M
size(M)
```

Una vez concluido este bucle, saldrán por pantalla dos mensajes en los que se solicita al usuario que introduzca, primero un valor de w_c que sea válido para el sistema y luego la w_g correspondiente para la frecuencia de cruce elegida.

Con estos valores, se calculan las matrices A *(4.13)* y B *(4.14)*, cuya obtención se explica detalladamente en el *Apartado 4* de la memoria; y con estas matrices generadas se calcula la matriz X *(4.17)*, de la cual pueden extraerse los valores de Kp, Td y Ti necesarios para la sintonización del PID clásico.

```
    % Obtenidas las posibles WCs y WGs, se solicita al usuario que elija
    % una Wc válida y su WG correspondiente:
    wc = input('Ingrese un valor de WC que sea válido para el sistema: ');
    wg = input('Ingrese el valor de WG correspondiente a la Wc elegida: ');

    [Gwc, Gwcp, Gwcmod] = F_transferencia_SinRetardo(a,b,c,d,e, wc);
    [Gwg, Gwgp, Gwgmod] = F_transferencia_SinRetardo(a,b,c,d,e, wg);

% Construimos las matrices para sacar las variables y las calculamos con
% la wg hallada
  A=[1 0 0;
   0 wc -1/wc;
   1 0 0;
   0 wg -1/wg];

  B= -[ cos(Fim + t*wc - Gwcp)/Gwcmod;
    sin(Fim + t*wc - Gwcp)/Gwcmod;
    cos(t*wg - Gwgp)/(Mg*Gwgmod);
    sin(t*wg - Gwgp)/(Mg*Gwgmod)];

% Generamos la matriz (A B) para comprobar que el rango es el correcto
  AB = [A B];
  rango = rank(AB);
% Teniendo A y B podemos obtener el valor de X
  X = -(inv(A'*A))*A'*B;
% Extraemos de la matriz X los valores deseados
  Kp = X(1,1);
  Td = X(2,1)/Kp;
  Ti = (1/X(3,1))*Kp;
end
```

10.1.3 PID fraccionario cuando Ki=0

FraccionarioPruebaSINKI

En esta función se realiza la implementación del método propuesto en el Apartado 5.1, en el cual se intenta conseguir los valores de los parámetros necesarios para un PD fraccionario (suponemos acción integral nula) con los coeficientes fraccionarios iguales ($\lambda = \mu$). Las especificaciones de diseño son la frecuencia de cruce (w_c), el margen de fase ($Ø_m$, nombrado en el programa como Fim) y el margen de ganancia (Mg), así como el retardo del sistema (t) y el valor de w_g.

Para empezar, se le pide a usuario que introduzca las características del sistema con el que se va a trabajar y se genera la función de transferencia mediante la llamada a la función F_transferencia_SinRetardo comentada con anterioridad.

```
function [Kp, Kd, mu]= FraccionarioPruebaSINKI (wc, Mg, Fim, t, wg)

% Introducimos los valores de la funcion de transferencia del sistema
        disp('Sabiendo que la funcion de transferencia tiene el formato siguiente');
        disp('       As + B       ');
        disp('--------------------');
        disp('   Cs^2 + Ds + E     ');
      a = input('Ingrese el valor de la variable a: ');
      b = input('Ingrese el valor de la variable b: ');
      c = input('Ingrese el valor de la variable c: ');
      d = input('Ingrese el valor de la variable d: ');
      e = input('Ingrese el valor de la variable e: ');


 % Llamamos a la función generada para que nos cree la función de
 % transferencia deseada y nos proporcione su fase (Gwcp) y su módulo (Gwcmod)
    [G, Gwcp, Gwcmod] = F_transferencia_SinRetardo(a,b,c,d,e, wc);
    [G, Gwgp, Gwgmod] = F_transferencia_SinRetardo(a,b,c,d,e, wg);
```

Al conocer todos los parámetros de la matriz B (4.14), se calculan los valores de sus filas, que luego se utilizarán en la resolución del método especificado.

```
% MATRIZ B =
%   -[ cos(Fim + t*wc - Gwcp)/Gwcmod;
%     sin(Fim + t*wc - Gwcp)/Gwcmod;
%     cos(t*wg - Gwgp- t*w)/(Mg*Gwgmod);
%     sin(t*wg - Gwgp)/(Mg*Gwgmod)];

F1B= - cos(Fim + t*wc - Gwcp)/Gwcmod;
F2B= -sin(Fim + t*wc - Gwcp)/Gwcmod;
F3B= - cos(t*wg - Gwgp)/(Mg*Gwgmod);
F4B= -sin(t*wg - Gwgp)/(Mg*Gwgmod);
```

En la ecuación (5.1.5) se especifica cómo obtener el valor del orden fraccionario, donde la variable H, compuesta por diferentes operandos de valores, puede establecerse como:

$$H = \frac{F3B - F1B}{-F2B + F4B}$$

De esta manera, se define el valor del orden fraccionario y se calcula las matrices A (4.13) y B (4.14), a partir de las cuales se puede obtener la matriz X, de la que se extraen los parámetros para el controlador PD fraccionario.

```
mu = (2/pi)*acot((F3B-F1B)/(-F2B+F4B));

A=[1      ((wc^mu)*cos((pi/2)*mu))  ;
   0      ((wc^mu)*sin((pi/2)*mu))  ;
   1      ((wg^mu)*cos((pi/2)*mu))  ;
   0      ((wg^mu)*sin((pi/2)*mu)) ];

B = [F1B;
    F2B;
    F3B;
    F4B];
% Teniendo A y B podemos obtener el valor de X
    X = -(inv(A'*A))*A'*B;
 % Extraemos de la matriz X los valores deseados
 Kp = X(1,1);
 Kd = X(2,1)/Kp;

end
```

**la resolución por este método lleva a valores que no son válidos para un controlador, por lo que se descarta su utilización para fines prácticos, tanto simulados como reales.*

10.1.4 PID fraccionario cuando λ=μ

FraccionarioPrueba2

Este código se implementa para la resolución del método planteado en el Apartado 5.2 de la memoria, es decir, para aquel caso en el que tenemos como especificaciones la frecuencia de cruce (w_c), el margen de fase ($Ø_m$, nombrado en el programa como Fim) y el margen de ganancia (Mg), así como el retardo del sistema (t). En este caso particular, se incluye también como especificación w_g y se establece que $\lambda = \mu$.

Al igual que en el anterior código se empieza pidiendo al usuario que introduzca los valores deseados para la función de transferencia especificada, calculando esta, su módulo y su fase con una llamada a la función

F_transferencia_SinRetardo explicada anteriormente, con la diferencia que en este caso se realizan dos llamadas, una para w_c y otra para w_g.

```
function [Kp, Kd, Ki, landaF]= FraccionarioPrueba2 (wc, Mg, Fim, t, wg)

% Introducimos los valores de la funcion de transferencia del sistema
        disp('Sabiendo que la funcion de transferencia tiene el formato siguiente');
        disp('       As + B       ');
        disp('--------------------');
        disp('   Cs^2 + Ds + E     ');
      a = input('Ingrese el valor de la variable a: ');
      b = input('Ingrese el valor de la variable b: ');
      c = input('Ingrese el valor de la variable c: ');
      d = input('Ingrese el valor de la variable d: ');
      e = input('Ingrese el valor de la variable e: ');

 % Llamamos a la función generada para que nos cree la función de
 % transferencia deseada y nos proporcione su fase (Gwcp) y su módulo (Gwcmod)
    [G, Gwcmod, Gwcp] = F_transferencia_SinRetardo(a,b,c,d,e, wc);
    [G, Gwgmod, Gwgp] = F_transferencia_SinRetardo(a,b,c,d,e, wg);
```

A continuación, como se disponen de todos los valores necesarios para el cálculo de la matriz B, se obtienen los valores de cada una de las filas de esta matriz, que luego serán utilizados en la siguiente parte del código.

```
% MATRIZ B =
%    -[ cos(Fim + t*wc - Gwcp)/Gwcmod;
%       sin(Fim + t*wc - Gwcp)/Gwcmod;
%       cos(t*wg - Gwgp- t*w)/(Mg*Gwgmod);
%       sin(t*wg - Gwgp)/(Mg*Gwgmod)];

F1B= - cos(Fim + t*wc - Gwcp)/Gwcmod;
F2B= -sin(Fim + t*wc - Gwcp)/Gwcmod;
F3B= - cos(t*wg - Gwgp)/(Mg*Gwgmod);
F4B= -sin(t*wg - Gwgp)/(Mg*Gwgmod);
```

Como se explica en el apartado referente a este código (Apartado 5.2), se pretende buscar un valor para λ que haga el determinante de la matriz A ampliada con B igual a cero. Para ello, se crea un bucle donde se va variando el valor de landa entre 0 y 1 y se va calculando el valor del determinante de la matriz M (denominada así en el código, es la matriz de A ampliada con B). Este valor se guarda en un vector (*valores*) que será el utilizado para la representación gráfica.

```
% GENERACION DE VALORES PARA LA GRAFICA
aa=1; %inicializacion del vector

for landa=0:0.01:0.99

   % Matriz compuesta A y B
   M=[1    ((wc)^(-landa)*cos((pi/2)*landa))  ((wc^landa)*cos((pi/2)*landa))   F1B;
      0     -((wc)^(-landa)*sin((pi/2)*landa))  ((wc^landa)*sin((pi/2)*landa))  F2B;
      1     ((wg)^(-landa)*cos((pi/2)*landa))  ((wg^landa)*cos((pi/2)*landa))  F3B;
      0     -((wg)^(-landa)*sin((pi/2)*landa))  ((wg^landa)*sin((pi/2)*landa)) F4B];

   valores(aa) = det(M);

   aa=aa+1;

end
size(valores);
```

En la representación gráfica se mostrará los valores que va tomando del determinante de la matriz, por lo que se opta por generar una línea horizontal en el valor cero para facilitar la visualización de este corte. Es por ello por lo que se crea un vector (w2) de 2500 puntos entre 0 y 1.

Al igual que en representaciones anteriores, se divide en dos gráficas, la superior será una imagen general y la inferior una imagen ampliada. En este caso se amplía según la zona de localización del corte con cero, por lo que dependerá del sistema en cuestión y deberá modificarse esta línea de código dependiendo del sistema con el que se trabaje.

```
% REPRESENTACION GRAFICA
w = linspace (0,1,100);
w2 = linspace(0,1,2500);

subplot(2,1,1)
plot(w,valores, w2, 0)
xlabel('Valor de landa');
legend('valores del det de la matriz M');

subplot(2,1,2)
plot(w,valores, w2, 0)
axis([ 0.92 0.99 -0.2 0.2]); % limites establecidos despues de observar
                             % el comportamiento de la grafica
xlabel('Valor de landa');
legend('valores del det de la matriz M');
```

A partir de la gráfica obtenida, se podrá extraer el valor de λ que cumple las condiciones de diseño, pidiéndose al usuario que inserte este mediante un mensaje.

Con este valor de λ, se calcularán las matrices A y B reducidas, válidas para obtener los valores de implementación del PID mediante el cálculo de la matriz X.

```
% Peticion de valor de landa
  landaF = input('Ingrese el valor de landa de la gráfica que cumple det(M)=0: ');

% Con este valor, calculamos las matrices A y B reducidas
  A1=[1     ((wc)^(-landaF)*cos((pi/2)*landaF))    ((wc^landaF)*cos((pi/2)*landaF));
      0     -((wc)^(-landaF)*sin((pi/2)*landaF))   ((wc^landaF)*sin((pi/2)*landaF));
      1     ((wg)^(-landaF)*cos((pi/2)*landaF))    ((wg^landaF)*cos((pi/2)*landaF))];

  B1= -[ cos(Fim + t*wc - Gwcp)/Gwcmod;
    sin(Fim + t*wc - Gwcp)/Gwcmod;
    cos(t*wg - Gwgp)/(Mg*Gwgmod)];

% Teniendo A y B podemos obtener el valor de X
  X = -(inv(A1'*A1))*A1'*B1;

% Extraemos de la matriz X los valores deseados
  Kp = X(1,1);
  Kd = X(2,1)/Kp;
  Ki = (1/X(3,1))*Kp;
end
```

10.1.5 PID fraccionario cuando λ≠μ

FraccionarioPrueba3

La implementación de este código se realiza para la resolución del método planteado en el Apartado 5.3 de la memoria, es decir, para aquel caso en el que tenemos como especificaciones la frecuencia de cruce (w_c), el margen de fase ($Ø_m$, nombrado en el programa como Fim) y el margen de ganancia (Mg) y el retardo del sistema (t), así como el valor de w_g. En este caso particular, se incluye también como especificación que $\lambda \neq \mu$.

Al igual que en el anterior código se empieza pidiendo al usuario que introduzca los valores deseados para la función de transferencia especificada, calculando esta, su módulo y su fase con una llamada a la función F_transferencia_SinRetardo explicada anteriormente, realizándose dos llamadas a esta función, una para w_c y otra para w_g.

```
function [Kp, Kd, Ki, landaF, muF]= FraccionarioPrueba3 (wc, Mg, Fim, t, wg)

% Introducimos los valores de la funcion de transferencia del sistema
        disp('Sabiendo que la funcion de transferencia tiene el formato siguiente');
        disp('       As + B       ');
        disp('--------------------');
        disp('   Cs^2 + Ds + E     ');
      a = input('Ingrese el valor de la variable a: ');
      b = input('Ingrese el valor de la variable b: ');
      c = input('Ingrese el valor de la variable c: ');
      d = input('Ingrese el valor de la variable d: ');
      e = input('Ingrese el valor de la variable e: ');


 % Llamamos a la función generada para que nos cree la función de
 % transferencia deseada y nos proporcione su fase (Gwcp) y su módulo (Gwcmod)
    [G, Gwcmod, Gwcp] = F_transferencia_SinRetardo(a,b,c,d,e, wc);
    [G, Gwgmod, Gwgp] = F_transferencia_SinRetardo(a,b,c,d,e, wg);
```

Al igual que en el caso anterior, disponemos de los valores necesarios para calcular la matriz B por lo que se obtienen los valores de cada una de las filas de esta matriz, que luego serán utilizados en la siguiente parte del código.

```
% MATRIZ B =
%   -[ cos(Fim + t*wc - Gwcp)/Gwcmod;
%     sin(Fim + t*wc - Gwcp)/Gwcmod;
%     cos(t*wg - Gwgp- t*w)/(Mg*Gwgmod);
%     sin(t*wg - Gwgp)/(Mg*Gwgmod)];

F1B= - cos(Fim + t*wc - Gwcp)/Gwcmod;
F2B= -sin(Fim + t*wc - Gwcp)/Gwcmod;
F3B= - cos(t*wg - Gwgp)/(Mg*Gwgmod);
F4B= -sin(t*wg - Gwgp)/(Mg*Gwgmod);
```

La siguiente parte del código consiste en la generación de una matriz de los valores (denominada *valores* en el código) que va tomando el determinante de la matriz A ampliada con B para las posibles parejas de λ-μ. Este bucle se realiza con una resolución de 0.01 entre 0 y 1, por lo que se obtiene una matriz de 100x100.

A la vez que se van generando los valores del determinante de la matriz, se generan dos vectores (vlanda y vu en el código) que representan los 100 valores (debido a la resolución escogida en el bucle) que pueden tener λ y μ.

```
% GENERACION DE GRAFICA EN 3D

 w = linspace (0,1,100);
 aa=1;
for landa=0:0.01:0.99
vlanda(aa)=landa;
vu(aa)=landa;
bb=1;
 for u = 0:0.01:0.99

%Matriz compuesta A y B
M=[1    ((wc)^(-landa)*cos((pi/2)*landa))  ((wc^u)*cos((pi/2)*u))   F1B;
   0     -((wc)^(-landa)*sin((pi/2)*landa))  ((wc^u)*sin((pi/2)*u))  F2B;
   1     ((wg)^(-landa)*cos((pi/2)*landa))  ((wg^u)*cos((pi/2)*u))  F3B;
   0     -((wg)^(-landa)*sin((pi/2)*landa))  ((wg^u)*sin((pi/2)*u)) F4B];

valores(aa,bb) = det(M);

bb=bb+1;

 end
 aa=aa+1;
 end
 size(valores);
```

Finalizado el cálculo de todos los valores, se representarán en una gráfica 3-D, donde se verá en el eje X los valores de λ, en el eje Y los valores de μ y en el eje Z los valores del determinante de la matriz A ampliada con B para cada par λ-μ.

```
% DIBUJO DE GRAFICA 3D
plot3(vlanda,vu,valores)
xlabel('landa')
ylabel('mu')
```

Como la visualización de los valores necesarios se antoja un tanto dificultosa en la gráfica, se ha decidido generar gráficas en 2-D fijando diferentes valores de λ y graficando el valor del determinante en relación a la variable μ.

Para ello, se vuelve a crear el bucle anteriormente mencionado, pero con una resolución mayor (de 0,001) de manera que los puntos a representar pasen de 100 a 1000, lo que dará mayor exactitud en los resultados.

```
% GENERACION DE GRAFICAS 2D
     cc=1;
for land=0:0.001:0.999
   dd=1;
    for mu = 0:0.001:0.999

%Matriz compuesta A y B
M=[1    ((wc)^(-land)*cos((pi/2)*land))  ((wc^mu)*cos((pi/2)*mu))   F1B;
   0     -((wc)^(-land)*sin((pi/2)*land))  ((wc^mu)*sin((pi/2)*mu))  F2B;
   1     ((wg)^(-land)*cos((pi/2)*land))  ((wg^mu)*cos((pi/2)*mu))  F3B;
   0     -((wg)^(-land)*sin((pi/2)*land))  ((wg^mu)*sin((pi/2)*mu)) F4B];

valoresD(cc, dd) = det(M);
dd=dd+1;
 end
 cc=cc+1;
 end
 size(valoresD)
```

A continuación, se elige una muestra representativa de los valores de λ, con el fin de poder comprobar el correcto funcionamiento del método planteado. También se generan los vectores 'v' y 'z', los cuales se utilizarán en la representación gráfica como los valores de μ y para trazar una línea horizontal en cero respectivamente. Pese a que ambos vectores son del mismo tamaño, por lo que podría haberse generado uno sólo para ambas cosas, se decide realizarlo de esta manera por si se quisiera modificar la resolución en el bucle, de manera que sólo se tenga que cambiar el vector 'v' (ya que si se reduce el utilizado en la horizontal en cero no realizará la función de ayudar a la detección visual del corte de los valores del determinante con cero).

```
% Valores elegidos para representar
valores1=valoresD(400,:);
valores2=valoresD(500,:);
valores3=valoresD(600,:);
valores4=valoresD(700,:);
valores5=valoresD(800,:);
valores6=valoresD(850,:);
valores7=valoresD(900,:);
valores8=valoresD(950,:);

v=[0:0.001:0.999];
tamv=size(v);
tamval=size(valores1);
z=linspace(0,1,1000);
```

En la representación gráfica, se fijan las gráficas mediante el comando *figure* para poder visualizar todas las gráficas generadas. Como añadido, se etiquetarán las

gráficas de manera que se facilita la relación entre la λ fijada en cada una y el valor de μ que le corresponderá.

```
% REPRESENTACION EN 2D
figure,plot(v,valores1, z, 0)
xlabel('mu (landa fijo=0.4)');
set(gcf,'Name','Valores mu con landa=0.4') ;
legend('valores del det de la matriz AB');

figure,plot(v,valores2, z, 0)
 xlabel('mu (landa fjo=0.5)');
 set(gcf,'Name','Valores u con landa=0.5') ;
 legend('valores del det de la matriz AB');

figure,plot(v,valores3, z, 0)
  xlabel('mu (landa fjo=0.6)');
  set(gcf,'Name','Valores u con landa=0.6') ;
  legend('valores del det de la matriz AB');

figure,plot(v,valores4, z, 0)
  xlabel('mu (landa fjo=0.7)');
  set(gcf,'Name','Valores u con landa=0.7') ;
  legend('valores del det de la matriz AB');

figure,plot(v,valores5, z, 0)
  xlabel('mu (landa fjo=0.8)');
  set(gcf,'Name','Valores u con landa=0.8') ;
  legend('valores del det de la matriz AB');
```

```
figure,plot(v,valores6, z, 0)
  xlabel('mu (landa fjo=0.85)');
  set(gcf,'Name','Valores u con landa=0.85') ;
  legend('valores del det de la matriz AB');

figure,plot(v,valores7, z, 0)
  xlabel('mu (landa fjo=0.9)');
  set(gcf,'Name','Valores u con landa=0.9') ;
  legend('valores del det de la matriz AB');

figure,plot(v,valores8, z, 0)
  xlabel('mu (landa fjo=0.95)');
  set(gcf,'Name','Valores u con landa=0.95') ;
  legend('valores del det de la matriz AB');
```

Generadas todas las gráficas, se le solicita al usuario que introduzca los valores de μ y λ que, a partir de la visualización de las gráficas, cumplan las condiciones especificadas.

Introducidos estos, se calcula las mencionadas matrices A y B, con las que podrá realizarse el cálculo de la matriz X y obtener los parámetros de diseño del PID fraccionario.

```
  % Peticion de valor de mu y landa definitivos elegidos
muF = input('Ingrese el valor de mu de la gráfica que cumpla det(M)=0: ');
landaF = input('Ingrese el valor de landa de la cual cogemos el mu deseado: ');

 % Estos valores, obtenemos las matrices A y B
B1= -[ cos(Fim + t*wc - Gwcp)/Gwcmod;
   sin(Fim + t*wc - Gwcp)/Gwcmod;
   cos(t*wg - Gwgp)/(Mg*Gwgmod)];

A1=[1     ((wc)^(-landaF)*cos((pi/2)*landaF))    ((wc^muF)*cos((pi/2)*muF));
    0      -((wc)^(-landaF)*sin((pi/2)*landaF))   ((wc^muF)*sin((pi/2)*muF));
    1      ((wg)^(-landaF)*cos((pi/2)*landaF))    ((wg^muF)*cos((pi/2)*muF))];

% Teniendo A y B podemos obtener el valor de X
   X = -(inv(A1'*A1))*A1'*B1;
% Extraemos de la matriz X los valores deseados
Kp = X(1,1);
Kd = X(2,1)/Kp;
Ki = (1/X(3,1))*Kp;

end
```

11. Conclusión del Trabajo de Fin de Grado

11.1 Conclusión

El desarrollo de este trabajo de fin de grado nos ha servido para aprender a profundizar la ingeniería de control, en cuanto que se ha implementado técnicas de control alternativas a las más clásicas vistas en el grado; investigando dichas técnicas y creando métodos de sintonización para probarlo en plantas reales y comparar sus resultados.

A lo largo de la realización del trabajo, hemos desempeñado tareas y desplegado habilidades de muy diversa índole, entre las que podemos destacar las siguientes:

- A partir de artículos de investigación, crear métodos matemáticos alternativos a los propuestos.
- Implementación en MATLAB de los métodos matemáticos propuestos.
- Extracción de información relevante de los manuales.
- Realización de simulaciones reales, sintonizando controladores PID para la obtención de las condiciones óptimas.
- Investigación y propuesta matemática para un tipo de control en desarrollo actual, como son los pseudoestados fraccionarios.
- Solventar los distintos inconvenientes que surgen al trabajar con sistemas reales.

En definitiva, hemos sido capaces de desempeñar la mayoría de las competencias contempladas en la guía docente del trabajo de fin de grado, entre las que podemos destacar:

- Conocimiento en materias básicas y tecnológicas, que les capacite para el aprendizaje de nuevos métodos y teorías, y les dote de versatilidad para adaptarse a nuevas situaciones.
- Capacidad de resolver problemas con iniciativa, toma de decisiones, creatividad, razonamiento crítico y de comunicar y transmitir conocimientos, habilidades y destrezas en el campo de la Ingeniería Electrónica Industrial
- Capacidad de trabajar en un entorno multilingüe y multidisciplinar.

- Capacidad para trabajar en equipo de forma eficaz.
- Capacidad para aprender y trabajar de forma autónoma.

Estamos muy satisfechos con la realización nuestro trabajo de fin de grado, que nos ha enriquecido en muchos aspectos tanto académicos como personales y, hemos conseguido unos resultados tangibles y satisfactorios.

11.2 Conclusion

Our EOG work has helped us to learn to deepen control engineering, as it has been implemented alternative control techniques to more traditional views on the degree. Investigating these techniques and creating tuning methods to test in real plants and compare their results.

Throughout the EOG work, we have done tasks and displayed skills of various kinds, among which we can highlight the following:

- From research articles, create alternative mathematical methods.
- Implementation of proposed mathematical methods with Matlab.
- Extraction of relevant information from manuals.
- Realization of realistic simulations, tuning PID controllers to obtain the optimal conditions.
- Research and mathematical proposal for a control type in development nowadays, such as fractional pseudo-states.
- Solve the various problems that arise while working with real systems.

We have been able to perform most of the competences laid down in the teaching guide the work of end of degree, among which we highlight:

- Knowledge in basic materials and technology that will enable them to learn new methods and theories, and equip them with versatility to adapt to new situations.
- Ability to solve problems with initiative, decision making, creativity, critical thinking and to communicate and transmit knowledge, skills and abilities in the field of Industrial Electronics Engineering.
- Ability to work in a multilingual and multidisciplinary environment.
- Ability to teamwork effectively.
- Ability to learn and work independently.

We are extremely satisfied with our EOG work, which has enriched us in many ways both academic and personal, and we have achieved some tangible and satisfactory results.

12. Referencias

[1] Teoría de control y cálculo fraccionario: Rev.R.Acad.Cienc.Exact.Fís.Nat. (Esp), Vol., Nº. , pp , 200; Monográfico: Cálculo fraccionario

[2] Control PID: C.A. Monje et al /Control Engineering Practice 16 (2008).

[3] Control PID: V. Feliu-Batlle et al / Control Engineering Practice 15 (2007).

[4] Doble rotor:

- http://www.aeromodelismo.com.es/helicopteros/helicopteros_rc_electrico/helicopterorc_coaxial.htm
- Manuales Feedback para TRMS

[5] Imágenes:

- http://www.lra.unileon.es/es/book/export/html/268
- Manual FEEDBACK
- Rev.R.Acad.Cienc.Exact.Fís.Nat. (Esp), Vol., Nº. , pp , 200; Monográfico: Cálculo fraccionario

[6] Códigos de MATLAB y Esquemas de Simulink: Empresa: MathWorks.

- Sitio web: http://www.mathworks.es/
- Centro de documentación: http://www.mathworks.es/es/help/index.html
- Descarga del programa: http://www.mathworks.es/products/matlab/

[7] Apuntes de la asignatura de Ingeniería de Control

ANEXO I:
Código de
Función de transferencia
sin retardo

```
% Obtención de la función de transferencia (sin inclusión de retardo) a
% partir de sus valores ordinales y para una frecuencia concreta. Devolverá
% así mismo, el módulo y la fase de dicha función de transferencia.

% Autores:
    % Sara Estévez Pérez
    % Javier León Gil
    % Alfonso Rodrigo Matesanz García


function [G, Gwcmod, Gwcp] = F_transferencia_SinRetardo(a,b,c,d,e, wc)

% se "genera" el término de Lagrange (s = jw) con la frecuencia deseada
    s = wc*1i;
% Definimos el numerador de la función de transferencia:
%  As + B
    num = a*s + b;
% Definimos el denominador de la función de transferencia:
%  Cs^2 + Ds + E
    den = c*(s).^2 + d*s + e;

% Generamos la función de transferencia deseada:

%             As + B
% G(s) = -----------------
%          Cs^2 + Ds + E

    G = num/den


 % Calculamos el modulo de la funcion de transferencia del sistema
    Gwcmod=abs(G)
 % y su fase
    Gwcp=angle(G)

end
```

Anexo II:
Código
PID Clásico

```
% Obtención de parámetros de control para un PID clásico a partir de las
% especificaciones de diseño siguientes:
    % wc = frecuencia de cruce
    % Mg = margen de ganancia
    % Fim = margen de fase
    % t = retardo del sistema
% Autores:
    % Sara Estévez Pérez
    % Javier León Gil
    % Alfonso Rodrigo Matesanz García

function [Kp, Td, Ti]= clasicoGrafica2 (wc1, Mg, Fim, t)

% Introducimos los valores de la funcion de transferencia del sistema
        disp('Sabiendo que la funcion de transferencia tiene el formato
siguiente');
        disp('       As + B       ');
        disp('--------------------');
        disp('   Cs^2 + Ds + E     ');
      a = input('Ingrese el valor de la variable a: ');
      b = input('Ingrese el valor de la variable b: ');
      c = input('Ingrese el valor de la variable c: ');
      d = input('Ingrese el valor de la variable d: ');
      e = input('Ingrese el valor de la variable e: ');

 % Llamamos a la función generada para que nos cree la función de
 % transferencia deseada y nos proporcione su fase (Gwcp) y su módulo
(Gwcmod)
    [G, Gwc1mod, Gwc1p] = F_transferencia_SinRetardo(a,b,c,d,e, wc1);

 % Debemos encontrar un valor de wg que haga cumplir
 % rango (AB) = 3, para ello la primera fila de la matriz B debe ser igual
 % a la tercera fila de dicha matriz.
 % MATRIZ B =
 %   -[ cos(Fim + t*wc - Gwcp)/Gwcmod;
 %     sin(Fim + t*wc - Gwcp)/Gwcmod;
 %     cos(t*wg - Gwgp- t*w)/(Mg*Gwgmod);
 %     sin(t*wg - Gwgp)/(Mg*Gwgmod)];

 % Para hacer ésto, se hará mediante un bucle, y primero pedimos el valor
 % de las características del sistema para su funcion de transferencia


  % Bucle para creación de gráfica relativa a wc(F1B) y wg(F3B)

   aa=1; % inicialización de los vectores donde se calcularán las ecuaciones

    for k = 0:0.1:(99.9) % se pone hasta 99.9 para que los vectores generados
                         %   sean de la misma longitud
```

```
    s = k*1i;

numk = a*s + b;
denk = c*(s)^2 + d*s + e;

Gwk = (numk/denk);
% Nos quedamos con el modulo de la funcion de transferencia del
% sistema y su fase
Gwkmod=abs(Gwk);
Gwkp=angle(Gwk);

% Calculo del valor de F1B(grafica dependiente de wc)
F1B(aa) = [(cos(Fim + t*k - Gwkp)/Gwkmod)];
% Calculo del valor de F3B(grafica dependiente de wg)
F3B(aa) = [cos(t*k - Gwkp + t*k)/(Mg*Gwkmod)];

% aumentamos el indice del vector
aa=aa+1;
end  % finalizacion del bucle for


% REPRESENTACION GRAFICA

% Se divide en dos gráficas
% una primera donde se comparan las dos ecuaciones y se marca la wc
% elegida
  subplot(2,1,1)

  w = linspace (0,100,1000); % generamos 1.000 ptos entre 0 y 100
% se calcula mínimo y maximo para trazar la recta en la wc deseada
   minimo = min(F1B);
   maximo = max(F1B);
 y = linspace (minimo,maximo,1000); %generación de vector vertical para
                                    %marcar frecuencia wc

 plot(w,F1B,'r',w,F3B,wc1,y,'k')
 xlabel('Frecuencia');
 legend('g = gráfica dependiente de wc','h = gráfica dependiente de wg');

 % segunda gráfica donde se "amplia" la representación de la ecuacion
 % dependiente de wg (generalmente de valores más pequeños)para una mejor
 % visualización
  subplot(2,1,2)

 plot(w,F1B,'r',w,F3B,wc1,y,'k')
  xlabel('Frecuencia');
 legend('g = gráfica dependiente de wc','h = gráfica dependiente de wg');
 % se calcula mínimo y maximo de F3B para establecer límite de la gráfica
```

```
       mini = min(F3B);
       maxi = max(F3B);
     axis([ 0 100 (3*mini) (3*maxi)]);

    %
    %
    %
    % BUCLE PARA OBTENCION DE WCs POSIBLES Y WG QUE SE TENDRÍAN
    %
    %
     v = 1; % inicializacion para la matriz donde se guardarán las wc
            % y su correspondiente wg

  for q = 0:0.01:(99.99)

        s = q*1i;
    numq = a*s + b;
    denq = c*(s)^2 + d*s + e;

    Gwq = (numq/denq);

    % Nos quedamos con el modulo de la funcion de transferencia del
    % sistema y su fase
    Gwqmod=abs(Gwq);
    Gwqp=angle(Gwq);

    % Calculo del valor de F1B(grafica dependiente de wc)
    F1B = [(cos(Fim + t*q - Gwqp)/Gwqmod)];
    % Redondeamos a 2 numeros decimales
    f1b = round(F1B*10^2)/10^2

    for n = q:0.01:(99.99)   % bucle que recorrerá desde el valor de la
frecuencia
                             % de wc hacia valores más altos
        s = n*1i;
    numn = a*s + b;
    denn = c*(s)^2 + d*s + e;

    Gwn = (numn/denn);
    % Nos quedamos con el modulo de la funcion de transferencia del
    % sistema y su fase
    Gwnmod=abs(Gwn);
    Gwnp=angle(Gwn);
    % Calculo del valor de F3B(grafica dependiente de wg)
    F3B = [cos(t*n - Gwnp + t*n)/(Mg*Gwnmod)];
    f3b = round(F3B*10^2)/10^2;

    if ( f1b == f3b )
        M(v,1) = q;
```

```
        M(v,2) = n;
         v = v+1;
         break;
    else
        n = n + 0.01;
    end
    end  % finalizacion bucle for(n)
     q = q + 0.01;
    end  % finalizacion bucle for(q)
    % mostramos la matriz generada, donde aparecera en la primera columna
    % los valores de Wc y en la segunda su Wg correspondiente
    M
    size(M)

    % Obtenidas las posibles WCs y WGs, se solicita al usuario que elija
    % una Wc válida y su WG correspondiente:
    wc = input('Ingrese un valor de WC que sea válido para el sistema: ');
    wg = input('Ingrese el valor de WG correspondiente a la Wc elegida: ');

    [Gwc, Gwcp, Gwcmod] = F_transferencia_SinRetardo(a,b,c,d,e, wc);
    [Gwg, Gwgp, Gwgmod] = F_transferencia_SinRetardo(a,b,c,d,e, wg);

% Construimos las matrices para sacar las variables y las calculamos con
% la wg hallada
  A=[1 0 0;
   0 wc -1/wc;
   1 0 0;
   0 wg -1/wg];

  B= -[ cos(Fim + t*wc - Gwcp)/Gwcmod;
    sin(Fim + t*wc - Gwcp)/Gwcmod;
    cos(t*wg - Gwgp)/(Mg*Gwgmod);
    sin(t*wg - Gwgp)/(Mg*Gwgmod)];

% Generamos la matriz (A B) para comprobar que el rango es el correcto
  AB = [A B];
  rango = rank(AB);
% Teniendo A y B podemos obtener el valor de X
  X = -(inv(A'*A))*A'*B;
% Extraemos de la matriz X los valores deseados
  Kp = X(1,1);
  Td = X(2,1)/Kp;
  Ti = (1/X(3,1))*Kp;
end
```

Anexo III:
Código
PD fraccionario

```
% Obtención de parámetros de control para un PID fraccionario a partir de las
% especificaciones de diseño siguientes:
    % wc = frecuencia de cruce
    % Mg = margen de ganancia
    % Fim = margen de fase
    % t = retardo del sistema
% También se establece el valor de wg
% Se supone que no existe acción integral (Ki = 0)
% Y que landa = mu

% Autores:
    % Sara Estévez Pérez
    % Javier León Gil
    % Alfonso Rodrigo Matesanz García

function [Kp, Kd, mu]= FraccionarioPruebaSINKI (wc, Mg, Fim, t, wg)

% Introducimos los valores de la funcion de transferencia del sistema
        disp('Sabiendo que la funcion de transferencia tiene el formato
siguiente');
        disp('       As + B       ');
        disp('--------------------');
        disp('   Cs^2 + Ds + E     ');
      a = input('Ingrese el valor de la variable a: ');
      b = input('Ingrese el valor de la variable b: ');
      c = input('Ingrese el valor de la variable c: ');
      d = input('Ingrese el valor de la variable d: ');
      e = input('Ingrese el valor de la variable e: ');


 % Llamamos a la función generada para que nos cree la función de
 % transferencia deseada y nos proporcione su fase (Gwcp) y su módulo (Gwcmod)
    [G, Gwcp, Gwcmod] = F_transferencia_SinRetardo(a,b,c,d,e, wc);
    [G, Gwgp, Gwgmod] = F_transferencia_SinRetardo(a,b,c,d,e, wg);


 % MATRIZ B =
 %   -[ cos(Fim + t*wc - Gwcp)/Gwcmod;
 %     sin(Fim + t*wc - Gwcp)/Gwcmod;
 %     cos(t*wg - Gwgp- t*w)/(Mg*Gwgmod);
 %     sin(t*wg - Gwgp)/(Mg*Gwgmod)];

 F1B= - cos(Fim + t*wc - Gwcp)/Gwcmod;
 F2B= -sin(Fim + t*wc - Gwcp)/Gwcmod;
 F3B= - cos(t*wg - Gwgp)/(Mg*Gwgmod);
 F4B= -sin(t*wg - Gwgp)/(Mg*Gwgmod);
```

```
mu = (2/pi)*acot((F3B-F1B)/(-F2B+F4B));

A=[1      ((wc^mu)*cos((pi/2)*mu))  ;
   0      ((wc^mu)*sin((pi/2)*mu))  ;
   1      ((wg^mu)*cos((pi/2)*mu))  ;
   0      ((wg^mu)*sin((pi/2)*mu)) ];

B = [F1B;
    F2B;
    F3B;
    F4B];
% Teniendo A y B podemos obtener el valor de X
    X = -(inv(A'*A))*A'*B;
 % Extraemos de la matriz X los valores deseados
 Kp = X(1,1);
 Kd = X(2,1)/Kp;

end
```

Anexo IV:
Código
PID fraccionario cuando λ=μ

```
% Obtención de parámetros de control para un PID fraccionario a partir de
las
% especificaciones de diseño siguientes:
    % wc = frecuencia de cruce
    % Mg = margen de ganancia
    % Fim = margen de fase
    % t = retardo del sistema
% También se establece el valor de wg y landa=mu

% Autores:
    % Sara Estévez Pérez
    % Javier León Gil
    % Alfonso Rodrigo Matesanz García

function [Kp, Kd, Ki, landaF]= FraccionarioPrueba2 (wc, Mg, Fim, t, wg)

% Introducimos los valores de la funcion de transferencia del sistema
        disp('Sabiendo que la funcion de transferencia tiene el formato
siguiente');
        disp('       As + B       ');
        disp('--------------------');
        disp('   Cs^2 + Ds + E     ');
      a = input('Ingrese el valor de la variable a: ');
      b = input('Ingrese el valor de la variable b: ');
      c = input('Ingrese el valor de la variable c: ');
      d = input('Ingrese el valor de la variable d: ');
      e = input('Ingrese el valor de la variable e: ');

 % Llamamos a la función generada para que nos cree la función de
 % transferencia deseada y nos proporcione su fase (Gwcp) y su módulo
(Gwcmod)
    [G, Gwcmod, Gwcp] = F_transferencia_SinRetardo(a,b,c,d,e, wc);
    [G, Gwgmod, Gwgp] = F_transferencia_SinRetardo(a,b,c,d,e, wg);

 % MATRIZ B =
 %   -[ cos(Fim + t*wc - Gwcp)/Gwcmod;
 %     sin(Fim + t*wc - Gwcp)/Gwcmod;
 %     cos(t*wg - Gwgp- t*w)/(Mg*Gwgmod);
 %     sin(t*wg - Gwgp)/(Mg*Gwgmod)];

 F1B= - cos(Fim + t*wc - Gwcp)/Gwcmod;
 F2B= -sin(Fim + t*wc - Gwcp)/Gwcmod;
 F3B= - cos(t*wg - Gwgp)/(Mg*Gwgmod);
 F4B= -sin(t*wg - Gwgp)/(Mg*Gwgmod);

 % GENERACION DE VALORES PARA LA GRAFICA
 aa=1; %inicializacion del vector

 for landa=0:0.01:0.99
```

```
    % Matriz compuesta A y B
    M=[1    ((wc)^(-landa)*cos((pi/2)*landa))
((wc^landa)*cos((pi/2)*landa))   F1B;
       0     -((wc)^(-landa)*sin((pi/2)*landa))
((wc^landa)*sin((pi/2)*landa))  F2B;
       1     ((wg)^(-landa)*cos((pi/2)*landa))
((wg^landa)*cos((pi/2)*landa))  F3B;
       0     -((wg)^(-landa)*sin((pi/2)*landa))
((wg^landa)*sin((pi/2)*landa)) F4B];

    valores(aa) = det(M);

    aa=aa+1;

 end
 size(valores);

  % REPRESENTACION GRAFICA
  w = linspace (0,1,100);
  w2 = linspace(0,1,2500);

 subplot(2,1,1)
 plot(w,valores, w2, 0)
 xlabel('Valor de landa');
 legend('valores del det de la matriz M');

 subplot(2,1,2)
 plot(w,valores, w2, 0)
 axis([ 0.92 0.99 -0.2 0.2]); % limites establecidos despues de observar
                              % el comportamiento de la grafica
 xlabel('Valor de landa');
 legend('valores del det de la matriz M');

 % Peticion de valor de landa
   landaF = input('Ingrese el valor de landa de la gráfica que cumple
det(M)=0: ');

 % Con este valor, calculamos las matrices A y B reducidas
   A1=[1     ((wc)^(-landaF)*cos((pi/2)*landaF))
((wc^landaF)*cos((pi/2)*landaF));
       0     -((wc)^(-landaF)*sin((pi/2)*landaF))
((wc^landaF)*sin((pi/2)*landaF));
       1     ((wg)^(-landaF)*cos((pi/2)*landaF))
((wg^landaF)*cos((pi/2)*landaF))];

   B1= -[ cos(Fim + t*wc - Gwcp)/Gwcmod;
     sin(Fim + t*wc - Gwcp)/Gwcmod;
     cos(t*wg - Gwgp)/(Mg*Gwgmod)];
```

```
 % Teniendo A y B podemos obtener el valor de X
   X = -(inv(A1'*A1))*A1'*B1;

 % Extraemos de la matriz X los valores deseados
   Kp = X(1,1);
   Kd = X(2,1)/Kp;
   Ki = (1/X(3,1))*Kp;
end
```

Anexo V:
Código
PID fraccionario
cuando λ≠μ

```
% Obtención de parámetros de control para un PID fraccionario a partir de
las
% especificaciones de diseño siguientes:
    % wc = frecuencia de cruce
    % Mg = margen de ganancia
    % Fim = margen de fase
    % t = retardo del sistema
% También se establece el valor de wg y landa distinto de mu

% Autores:
    % Sara Estévez Pérez
    % Javier León Gil
    % Alfonso Rodrigo Matesanz García

function [Kp, Kd, Ki, landaF, muF]= FraccionarioPrueba3 (wc, Mg, Fim, t,
wg)

% Introducimos los valores de la funcion de transferencia del sistema
        disp('Sabiendo que la funcion de transferencia tiene el formato
siguiente');
        disp('       As + B       ');
        disp('--------------------');
        disp('   Cs^2 + Ds + E     ');
      a = input('Ingrese el valor de la variable a: ');
      b = input('Ingrese el valor de la variable b: ');
      c = input('Ingrese el valor de la variable c: ');
      d = input('Ingrese el valor de la variable d: ');
      e = input('Ingrese el valor de la variable e: ');


 % Llamamos a la función generada para que nos cree la función de
 % transferencia deseada y nos proporcione su fase (Gwcp) y su módulo
(Gwcmod)
    [G, Gwcmod, Gwcp] = F_transferencia_SinRetardo(a,b,c,d,e, wc);
    [G, Gwgmod, Gwgp] = F_transferencia_SinRetardo(a,b,c,d,e, wg);


 % MATRIZ B =
 %   -[ cos(Fim + t*wc - Gwcp)/Gwcmod;
 %     sin(Fim + t*wc - Gwcp)/Gwcmod;
 %     cos(t*wg - Gwgp- t*w)/(Mg*Gwgmod);
 %     sin(t*wg - Gwgp)/(Mg*Gwgmod)];

 F1B= - cos(Fim + t*wc - Gwcp)/Gwcmod;
 F2B= -sin(Fim + t*wc - Gwcp)/Gwcmod;
 F3B= - cos(t*wg - Gwgp)/(Mg*Gwgmod);
 F4B= -sin(t*wg - Gwgp)/(Mg*Gwgmod);
```

```
 % GENERACION DE GRAFICA EN 3D

  w = linspace (0,1,100);
  aa=1;
 for landa=0:0.01:0.99
vlanda(aa)=landa;
vu(aa)=landa;
bb=1;
 for u = 0:0.01:0.99

%Matriz compuesta A y B
M=[1    ((wc)^(-landa)*cos((pi/2)*landa))  ((wc^u)*cos((pi/2)*u))   F1B;
   0     -((wc)^(-landa)*sin((pi/2)*landa))  ((wc^u)*sin((pi/2)*u))  F2B;
   1     ((wg)^(-landa)*cos((pi/2)*landa))  ((wg^u)*cos((pi/2)*u))  F3B;
   0     -((wg)^(-landa)*sin((pi/2)*landa))  ((wg^u)*sin((pi/2)*u)) F4B];

valores(aa,bb) = det(M);

bb=bb+1;

 end
 aa=aa+1;
 end
 size(valores);

 % DIBUJO DE GRAFICA 3D
 plot3(vlanda,vu,valores)
 xlabel('landa')
 ylabel('mu')


 % GENERACION DE GRAFICAS 2D
      cc=1;
 for land=0:0.001:0.999
    dd=1;
     for mu = 0:0.001:0.999

%Matriz compuesta A y B
M=[1    ((wc)^(-land)*cos((pi/2)*land))  ((wc^mu)*cos((pi/2)*mu))   F1B;
   0     -((wc)^(-land)*sin((pi/2)*land))  ((wc^mu)*sin((pi/2)*mu))  F2B;
   1     ((wg)^(-land)*cos((pi/2)*land))  ((wg^mu)*cos((pi/2)*mu))  F3B;
   0     -((wg)^(-land)*sin((pi/2)*land))  ((wg^mu)*sin((pi/2)*mu)) F4B];

valoresD(cc, dd) = det(M);
dd=dd+1;
 end
 cc=cc+1;
 end
 size(valoresD)
```

```
% Valores elegidos para representar
valores1=valoresD(400,:);
valores2=valoresD(500,:);
valores3=valoresD(600,:);
valores4=valoresD(700,:);
valores5=valoresD(800,:);
valores6=valoresD(850,:);
valores7=valoresD(900,:);
valores8=valoresD(950,:);

v=[0:0.001:0.999];
tamv=size(v);
tamval=size(valores1);
z=linspace(0,1,1000);

% REPRESENTACION EN 2D
figure,plot(v,valores1, z, 0)
xlabel('mu (landa fijo=0.4)');
set(gcf,'Name','Valores mu con landa=0.4') ;
legend('valores del det de la matriz AB');

figure,plot(v,valores2, z, 0)
 xlabel('mu (landa fjo=0.5)');
 set(gcf,'Name','Valores u con landa=0.5') ;
 legend('valores del det de la matriz AB');

figure,plot(v,valores3, z, 0)
  xlabel('mu (landa fjo=0.6)');
  set(gcf,'Name','Valores u con landa=0.6') ;
  legend('valores del det de la matriz AB');

figure,plot(v,valores4, z, 0)
  xlabel('mu (landa fjo=0.7)');
  set(gcf,'Name','Valores u con landa=0.7') ;
  legend('valores del det de la matriz AB');

figure,plot(v,valores5, z, 0)
  xlabel('mu (landa fjo=0.8)');
  set(gcf,'Name','Valores u con landa=0.8') ;
  legend('valores del det de la matriz AB');

figure,plot(v,valores6, z, 0)
  xlabel('mu (landa fjo=0.85)');
  set(gcf,'Name','Valores u con landa=0.85') ;
  legend('valores del det de la matriz AB');

figure,plot(v,valores7, z, 0)
  xlabel('mu (landa fjo=0.9)');
```

```
   set(gcf,'Name','Valores u con landa=0.9') ;
   legend('valores del det de la matriz AB');

 figure,plot(v,valores8, z, 0)
   xlabel('mu (landa fjo=0.95)');
   set(gcf,'Name','Valores u con landa=0.95') ;
   legend('valores del det de la matriz AB');

    % Peticion de valor de mu y landa definitivos elegidos
  muF = input('Ingrese el valor de mu de la gráfica que cumpla det(M)=0:
');
  landaF = input('Ingrese el valor de landa de la cual cogemos el mu
deseado: ');

   % Estos valores, obtenemos las matrices A y B
  B1= -[ cos(Fim + t*wc - Gwcp)/Gwcmod;
     sin(Fim + t*wc - Gwcp)/Gwcmod;
     cos(t*wg - Gwgp)/(Mg*Gwgmod)];

  A1=[1     ((wc)^(-landaF)*cos((pi/2)*landaF))
((wc^muF)*cos((pi/2)*muF));
       0     -((wc)^(-landaF)*sin((pi/2)*landaF))
((wc^muF)*sin((pi/2)*muF));
       1     ((wg)^(-landaF)*cos((pi/2)*landaF))
((wg^muF)*cos((pi/2)*muF))];

 % Teniendo A y B podemos obtener el valor de X
    X = -(inv(A1'*A1))*A1'*B1;
 % Extraemos de la matriz X los valores deseados
 Kp = X(1,1);
 Kd = X(2,1)/Kp;
 Ki = (1/X(3,1))*Kp;

end
```