Diseño teórico-práctico de un controlador para motor eléctrico Minimotors 012C

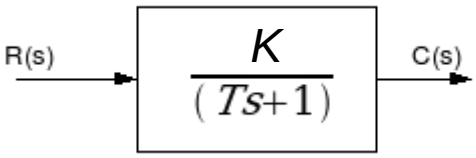
Autores:
Aguilera Díez, Javier
Blanco Cano, Jorge
Cano Molina, José Manuel
Mrani Alaoui, Idriss
Yebra Granados, Mª del Mar

Índice

1. Análisis y modelado del motor simplificado como sistema de primer orden.

En este apartado consideraremos un modelo simplificado del motor, analizándo su respuesta en velocidad como sistema de primer orden. El esquema sería el siguiente.

Simplificando:



Cuya función de transferencia es : $\frac{C(s)}{R(s)} = \frac{K}{Ts+1}$

Observamos que se trata de un esquema en lazo abierto que nos permitirá estudiar su respuesta en velocidad.

Hacemos que la señal r(t) a la entrada sea una señal escalón de amplitud 1 V, cuya transformada de Laplace será R(s) = $\frac{1}{s}$

Descomponiendo en fracciones simples obtenemos :

$$C(s) = K \cdot \left(\frac{1}{s} - \frac{1}{s+(1/T)} \right)$$

Volviendo al dominio del tiempo y considerando condiciones iniciales nulas obtenemos que la respuesta de nuestro sistema es :

$$c(t) = K \cdot (1 - e^{-t/T})$$

1.1 Estudio experimental

Debemos determinar experimentalmente cuál será el valor concreto de T, ya que sabemos que el polo de este sistema de primer orden estará en s = $\frac{-1}{T}$. Para ello recurrimos a la simulación gráfica obtenida a partir de montar un circuito equivalente en la calculadora analógica y estudiar la respuesta real del motor:

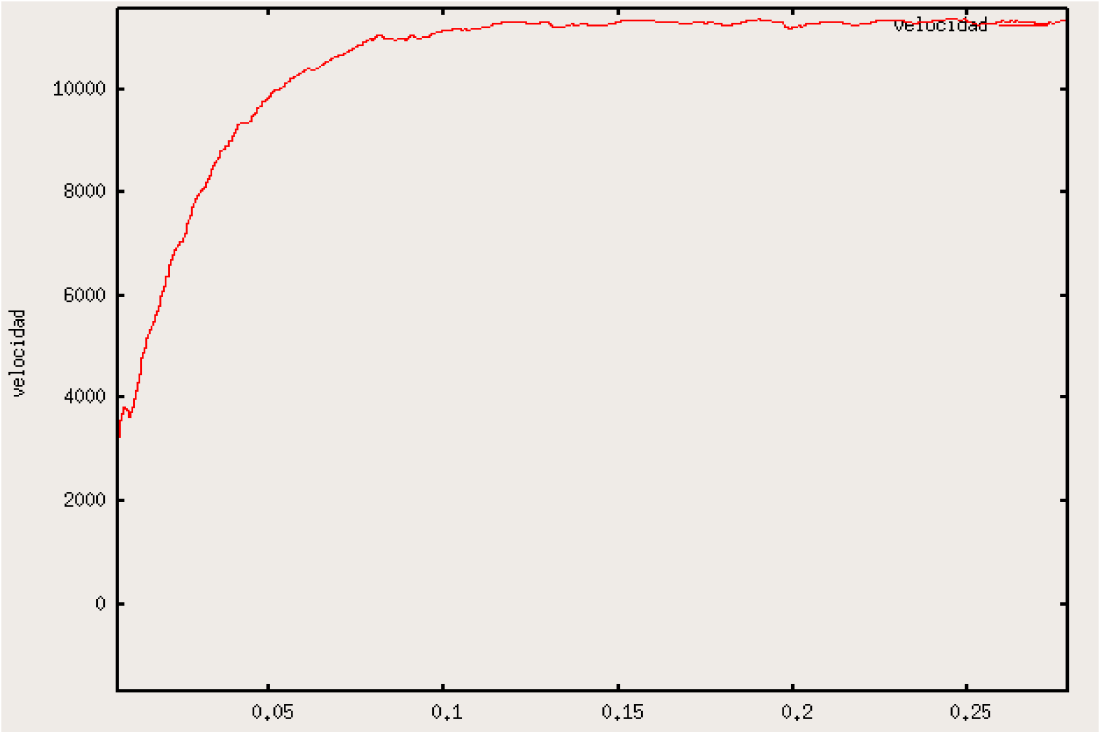


Figura 1.1 - Respuesta en velocidad del motor simplificado

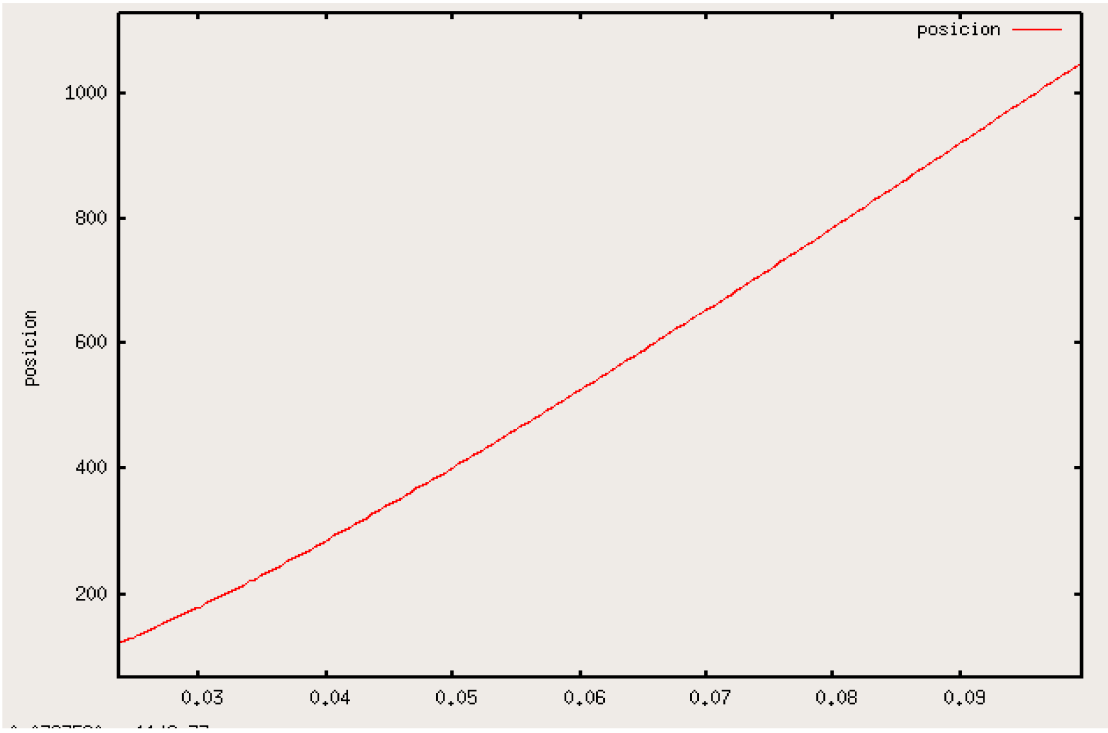


Figura 1.2 - Respuesta en posición del motor simplificado

Aplicando el criterio de la constante de tiempo, sabemos que el instante en que la respuesta en velocidad alcance el 63.2 % de su valor medio en régimen permamente, será el instante T.

$$0.632 \cdot 11200 = 7078.4$$

Por tanto buscamos en que instante la gráfica vale 7078.4 . Para ello recurrimos al archivo de datos que genera el simulador y disponer de mayor precisión.

Columna 1: tiempo (s).	Columna 2: posicion.	Columna 3: velocidad.
0.0237450003623962	121.50000000000000000	6911.50000000000000000
0.0240090005099773	124.50000000000000000	6976.50000000000000000
0.0242730006575584	126.50000000000000000	6976.50000000000000000
0.0245379991829395	128.50000000000000000	6976.50000000000000000
0.0248019993305206	130.50000000000000000	7046.50000000000000000
0.0250659994781017	133.50000000000000000	7046.50000000000000000
0.0253299996256828	135.50000000000000000	7046.50000000000000000
0.0255950000137091	137.50000000000000000	7046.50000000000000000
0.0258590001612902	140.50000000000000000	7215.50000000000000000
0.0261230003088713	142.50000000000000000	7215.50000000000000000
0.0263880006968975	145.50000000000000000	7215.50000000000000000
0.0267069991677999	148.50000000000000000	7395.50000000000000000

Figura 1.3 - Extracto de los datos almacenados

Obteniendo T = 0.0256 (s)

Por tanto el polo estará aproximadamente en la posición s = $\frac{-1}{0{,}0256} \simeq -40$

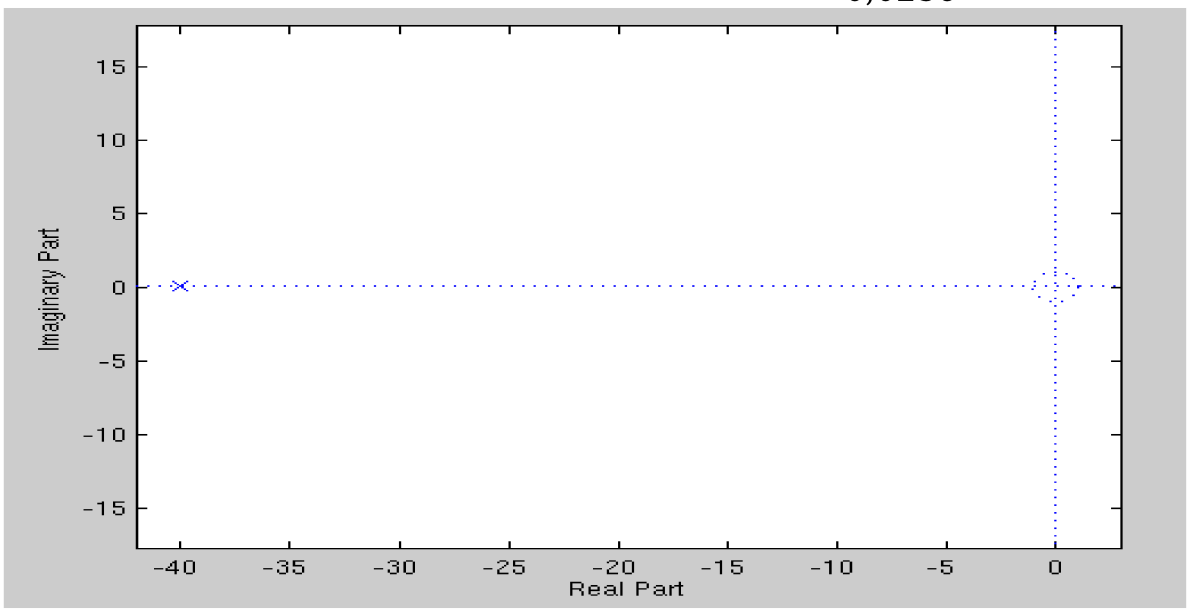


Figura 1.4 – Diagrama de polos y ceros del sistema.

Observamos que el polo se encuentra en el semiplano negativo, como era de esperar dada la estabilidad del sistema.

La ganancia K será estudiada mas adelante, en el punto 2. Ya que precisa de una introducción acerca del software de simulación.

2. Estudio analítico del modelo del motor. Identificación de parámetros.

Una vez hemos modelado el motor que disponemos en el laboratorio como sistema de primer orden en un intento de simplificar el modelo, en este apartado vamos a modelar el motor de una manera más realista aunque haciendo las simplificaciones pertinentes.

Nuestra planta es un motor de corriente continua, un dispositivo electromecánico que transforma la energía eléctrica en energía mecánica, empleada en hacer girar el eje. Por tanto un motor tiene dos subsistemas: uno eléctrico formado por una bobina y una resistencia, donde la energía eléctrica se transforma en fuerza contraelectromotriz, y otro mecánico formado por un rotor, donde el par mecánico se materializa en forma de velocidad de giro del eje.

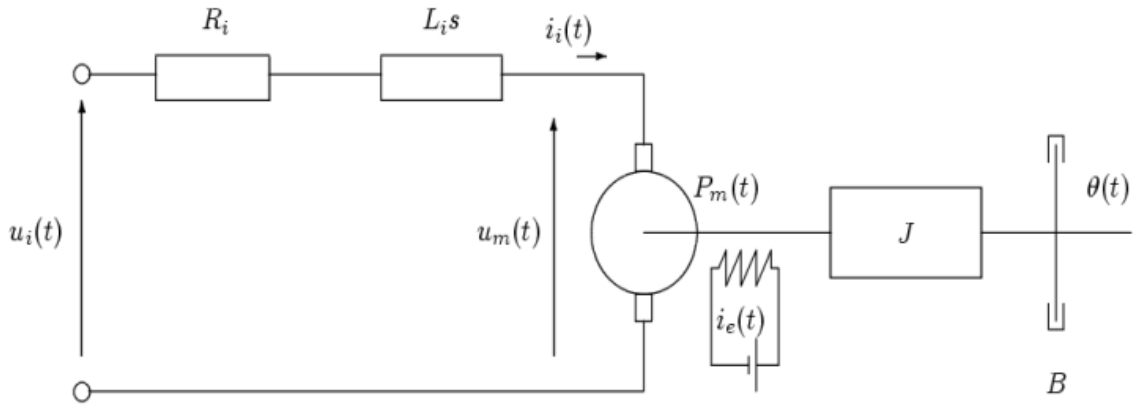


Figura 2.1 - Esquema eléctrico.

Nuestros parámetros principales son:

R ≡ Resistencia Terminal
L ≡ Inductancia
J ≡ Momento de inercia
B ≡ fricción mecánica
K ≡ Constante de velocidad
K´ ≡ Constante de corriente

Las ecuaciones correspondientes en el dominio del tiempo son las siguientes:

$$u(t) = R \cdot i(t) + L\frac{\mathrm{di(t)}}{dt} + K \cdot \dot{\theta}$$
$$K' \cdot i(t) = J \cdot \ddot{\theta} + B \cdot \dot{\theta}$$

Aplicando la transformada de Laplace quedan a las ecuaciones anteriores obtenemos:

$$U(s) = (R + L \cdot s) \cdot I(s) + K \cdot \theta(s)$$
$$K' \cdot I(s) = J \cdot \theta(s) + B \cdot \theta(s)$$

Operando convenientemente obtenemos la función de transferencia en el dominio de la frecuencia tomando como salida la variable posición que hemos llamado θ:

$$G'(s)=\frac{K'}{L\,J\,s^3+R\,J\,s^2+K\,K'\,s}$$

Para obtener la función de transferencia cogiendo como salida la variable velocidad basta con derivar en el dominio del tiempo la posición que se corresponde con multiplicar por s en el dominio de la frecuencia.

$$G(s) = s\,G'(s)$$

Con dichas funciones de transferencia podremos simular la respuesta de nuestro motor en lazo abierto. Para ello recurrimos a la hoja de características del motor utilizado (Minimotors 012C) e identificamos los parámetros necesarios para construirlas:

Parámetro en la ecuación	Nombre en la hoja de características	Valor en la hoja de características	Valor en el Sistema Internacional
K	K	$2.3\ \frac{mV}{rpm}$	$0.022\ \frac{Vs}{rad}$
K´	K	$22\ \frac{mNm}{A}$	$0.022\ \frac{Nm}{A}$
L	L	580 µH	$580\cdot10^{-6}H$
R	R	5.3Ω	5.3 Ω
J	J	$14\,g\,cm^2$	$14\cdot10^{-7}Kg\,m^2$
B	M	1.1mNm	1.1mNm

Nota: podemos considerar en nuestros cálculos la fricción mecánica B despreciable.

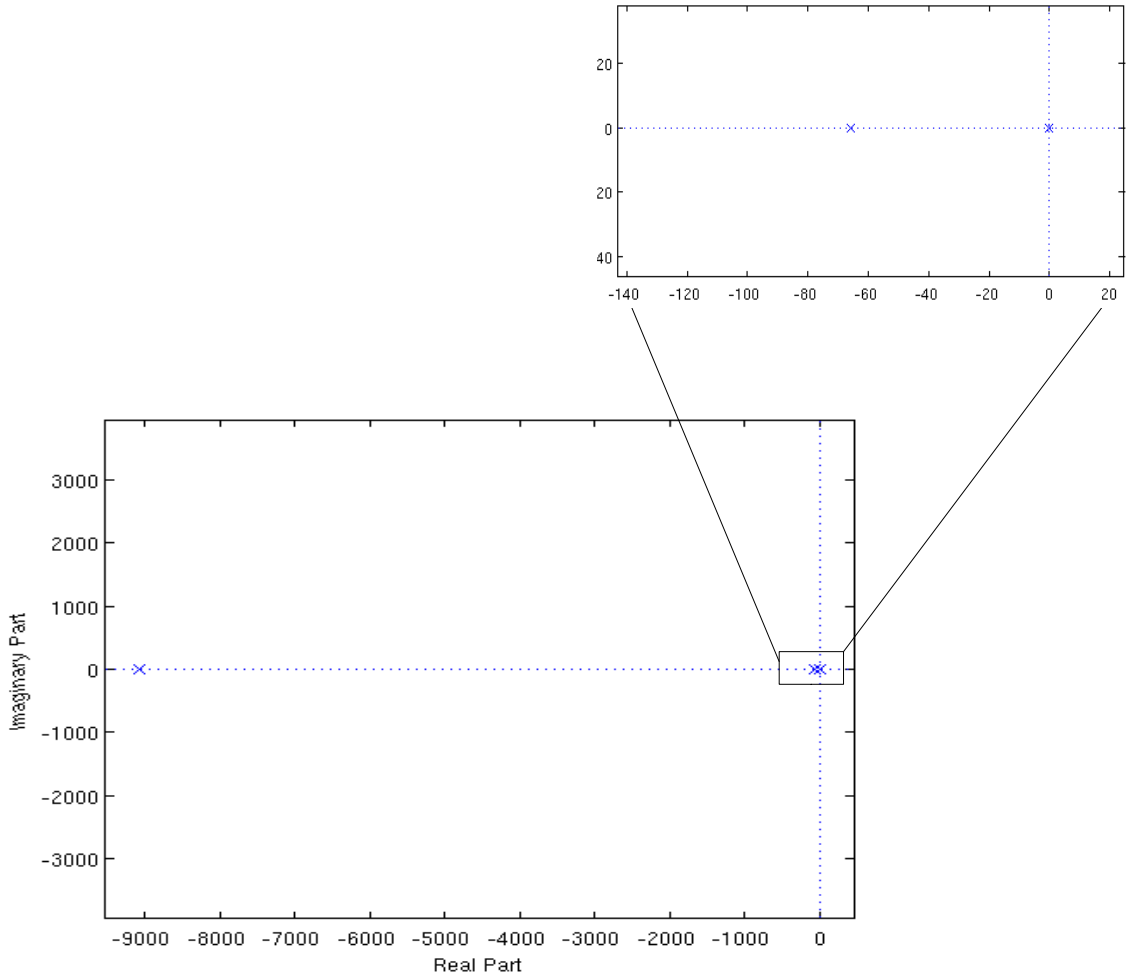
Con estos valores definimos en Matlab la función de transferencia como un stript llamado modeloMotor para simular la respuesta al escalón en posición y velocidad y poder comprobar que se corresponde con la respuesta del motor real:

```
num = [22e-3];
den = [8120e-13 7.42e-6 4.84e-4 0];
F = tf(num,den)
```

Después de la ejecución del script, una vez obtenida la función F invocamos a la siguiente función que programamos para obtener el diagrama de polos y ceros a partir de una función de transferencia dada:

```
function [] = diagramazp(F)
ZP = zpk(F);
zplane(ZP.z{:},ZP.p{:});
end
```

A continuación mostramos el diagrama obtenido:



Observamos que efectivamente es un sistema de tercer orden con un polo en el origen, y otros dos polos de los cuales se puede ver claramente la existencia de un polo dominante frente a otro. Esto es debido a que la respuesta dinámica parte eléctrica es mucho más rápida que la mecánica. Debido a la distancia entre ambos polos y al claro dominio en la respuesta dinámica de la parte mecánica del sistema se puede considerar válida la simplificación de la planta como sistema de segundo orden en posición.

Una vez visto el diagrama de polos y ceros ejecutamos el comando de Matlab step(F)
Para obtener la respuesta temporal al impulso:

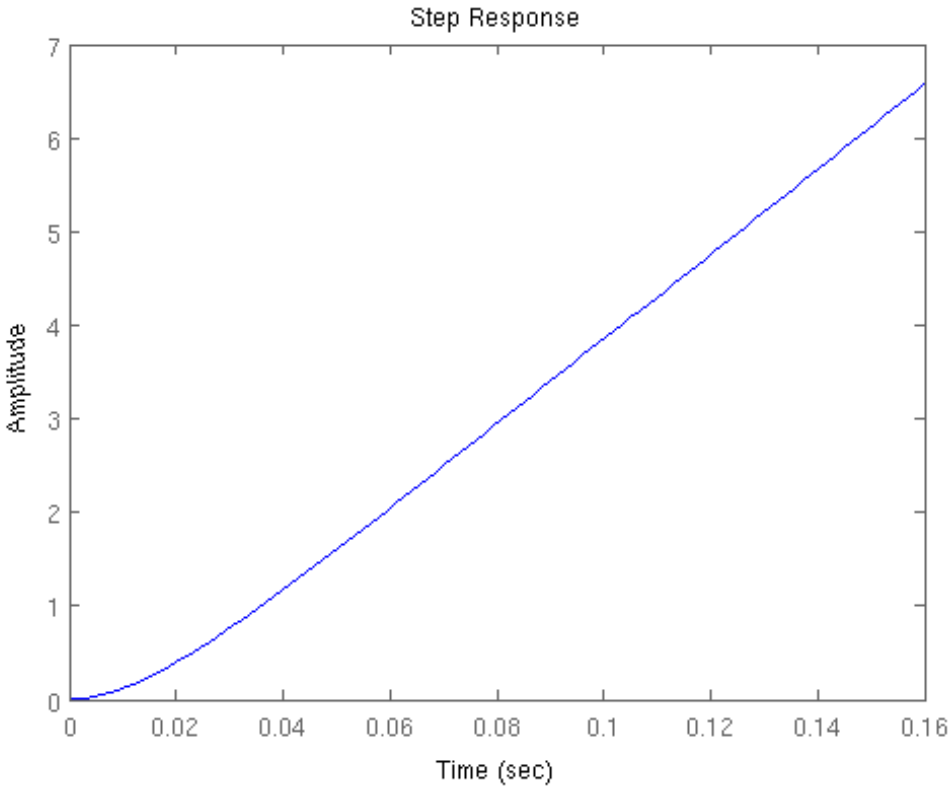


Figura 2.3 - Respuesta al escalón.

A continuación obtenemos la función de transferencia tomando como salida la velocidad, para ello basta con multiplicar por s en el dominio de la frecuencia.

Simulamos la respuesta al impulso y obtenemos lo siguiente:

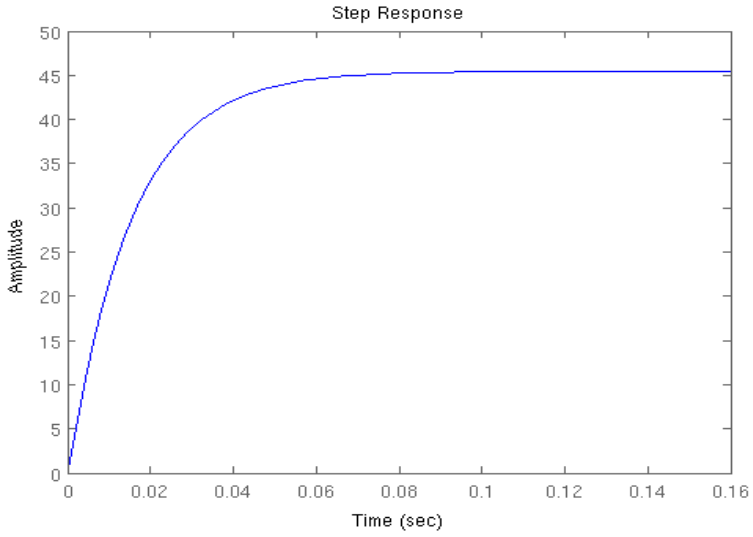


Figura 2.4 - Respuesta al escalón.

Finalmente utilizamos el simulador con el motor del laboratorio y comparamos con lo simulado en Matlab:

Respuesta al impulso en posición:

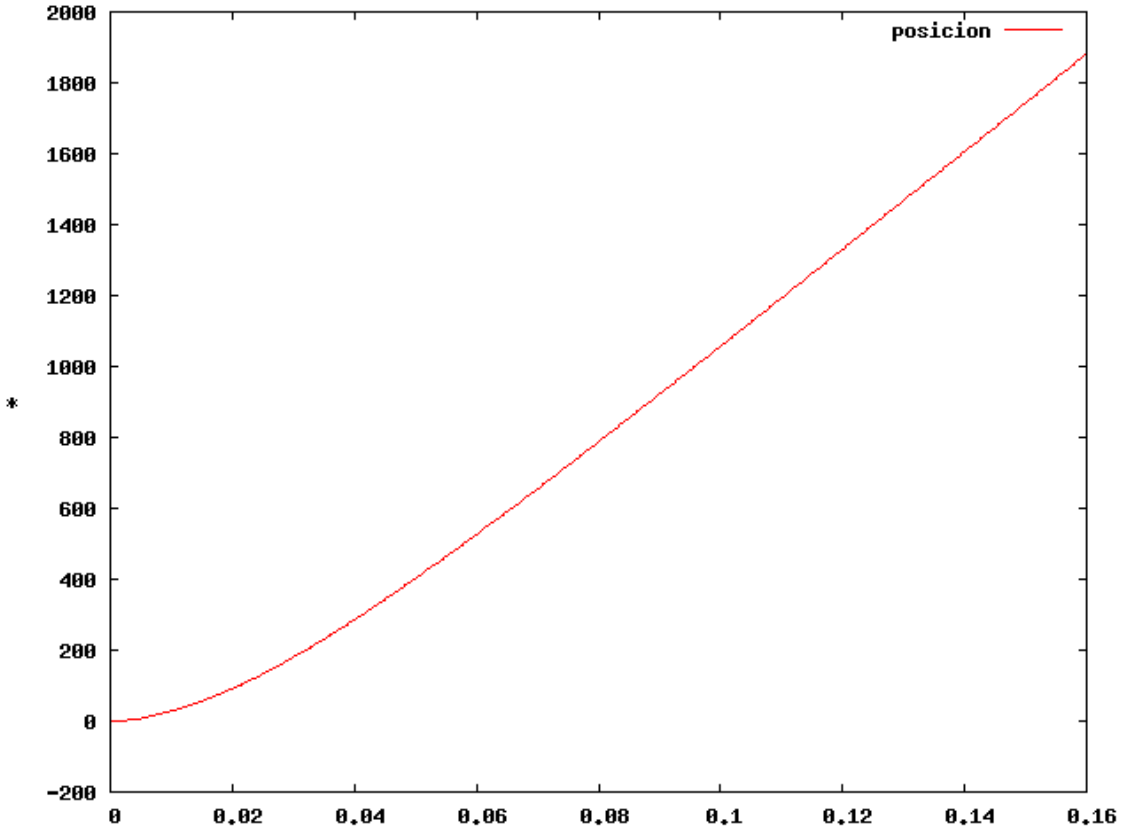


Figura 2.5 - Respuesta en posición.

Respuesta al impulso es velocidad:

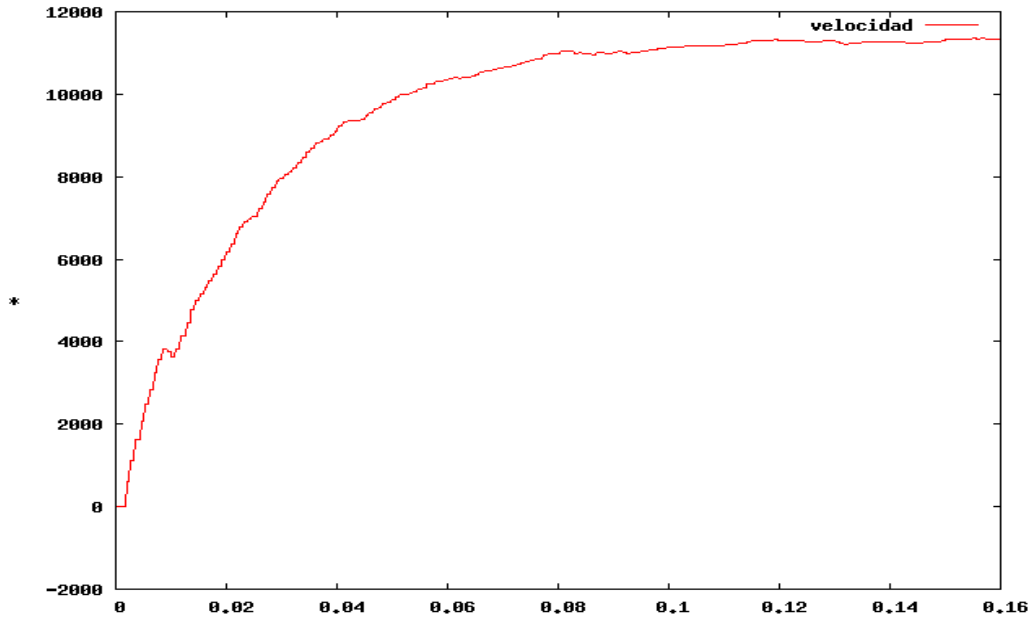


Figura 2.6 - Respuesta al impulso es velocidad.

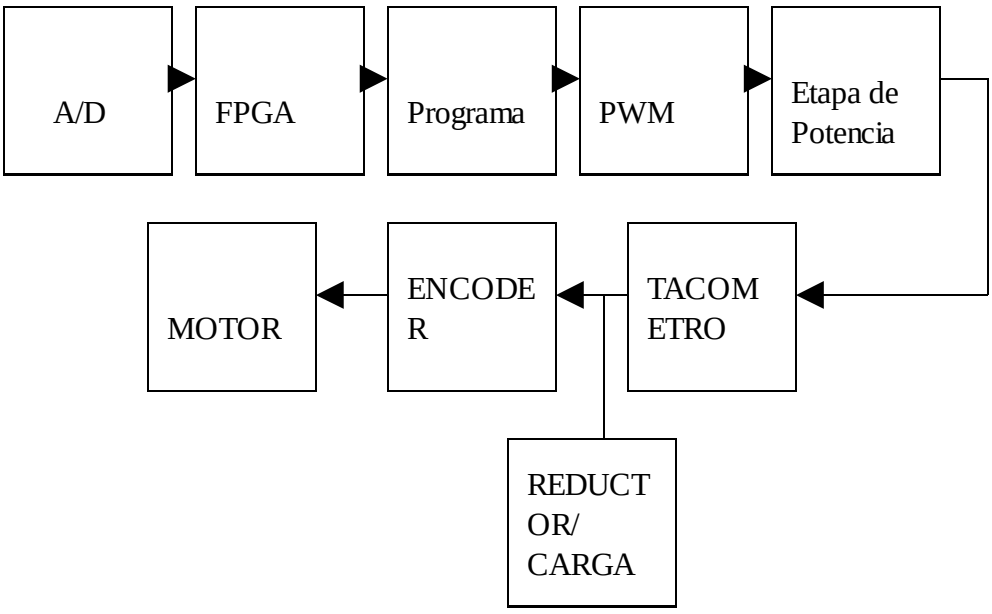
A la vista de los resultados comprobamos que el modelado realizado es bastante realista.

Observamos que en el eje temporal de la respuesta en velocidad no existen a penas diferencias entre la respuesta simulada teóricamente con Matlab y la generada por el motor en el laboratorio.
Sin embargo encontramos diferencias en cuanto a amplitudes se refiere. Observamos que en la simulación realizada en el laboratorio las amplitudes de la respuesta de multiplican por un factor considerable, esto puede deberse a diversos factores.

Si realizamos un diagrama de bloques del sistema que en realidad manejamos en el laboratorio, podríamos pensar que si cada módulo introduce alguna ganancia esta podría ser una causa razonable.

Analizando el problema más detenidamente nos damos cuenta que el módulo que hemos nombrado "programa" introduce una ganancia de 65.53 para normalizar que coincide razonablemente con la ganancia que a partir de las gráficas hemos deducido.

Teniendo en cuenta que en la simulación en Matlab al ejecutar el comando step, estamos calculando la respuesta al escalón unidad, es decir amplitud 1V; y que en el laboratorio introducíamos en el programa una amplitud de 5V que se corresponden con 4V reales aproximadamente (ya que 15V son 11.9V). Nosotros calculamos una ganancia total introducida de 62.7.



3. Modelado del motor en el espacio de estados.

Para realizar el modelado del motor, en primer lugar es necesario conocer las ecuaciones diferenciales que describen su comportamiento. El modelado en el espacio de estados nos permite expresar una ecuación diferencial de n-ésimo orden mediante una ecuación diferencial matricial de primer orden. Pues bien, esta ecuación matricial es lo que denominaremos ecuación de estado. Conociendo todos los valores iniciales de las variables de estado así como las funciones de entrada al sistema (en nuestro caso solo va a haber una: u(t)), seremos capaces de predecir el comportamiento del motor en todo instante de tiempo.

A partir de las ecuaciones en el dominio del tiempo, y considerando condiciones iniciales nulas :

$$u(t) = R\cdot i(t) + L\cdot \frac{di(t)}{dt} + k\cdot \frac{d\Theta(t)}{dt} \quad => \quad \frac{di(t)}{dt} = \frac{u(t)-R\cdot i(t)-k\cdot \theta'(t)}{L}$$

$$\tau = J\cdot \frac{d\Theta^2(t)}{dt^2} + \beta\cdot \frac{d\Theta(t)}{dt} = k'\cdot i(t) \quad => \quad \frac{d\Theta^2(t)}{dt^2} = \frac{K'\cdot i(t)-\beta\cdot \theta'(t)}{J}$$

Tomando:

$X_1 = \theta(t)$

$X_2 = \theta'(t)$ $\quad => \quad \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} \theta(t) \\ \theta'(t) \\ i(t) \end{pmatrix}$

$X_3 = i(t)$

Cogiendo como salida $\theta(t)$, para obtener la posición:

$$\theta(t) = X_1 = y = C^t \cdot X = C^t \cdot \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix}, \text{ luego } \quad C^t = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$$

$$\mathbf{X' = A\cdot x + B\cdot u}$$

$$X'_1 = X_2 ;$$

$$X'_2 = \frac{K'}{J} \cdot X_3 - \frac{\beta}{J} \cdot X_2 ;$$

$$X'_3 = \frac{-K}{L} \cdot X_2 - \frac{R}{L} \cdot X_3 + \frac{1}{L} \cdot u ;$$

Como:

$$\begin{pmatrix} X'_1 \\ X'_2 \\ X'_3 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & \frac{-\beta}{J} & \frac{K'}{J} \\ 0 & \frac{-K}{L} & \frac{-R}{L} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \frac{1}{L} \end{pmatrix} \cdot u ;$$

El sistema va a ser del orden de la matriz A, en este caso orden 3.

Obtenemos la función de transferencia:

$S \cdot X(s) = S \cdot \begin{pmatrix} X_1(s) \\ X_2(s) \\ X_3(s) \end{pmatrix} = A \cdot X(s) + b \cdot U(s) \quad => \quad X(s) \cdot \left[s \cdot I - A \right] = b \cdot U(s)$

Despejando $X(s) = \left[\left[s \cdot I - A \right]^{-1} \cdot b \right] \cdot U(s)$

$Y(s) = C^t \; X(s)$ siendo $C^t = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$ => $Y(s) = X_1(s)$;

Luego $G_\theta = C^t \cdot \left[\left[s \cdot I - A \right]^{-1} \cdot b \right] \cdot U(s)$;

con : $C^t = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$

$$I = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & \frac{-\beta}{J} & \frac{K}{J} \\ 0 & \frac{-K}{L} & \frac{-R}{L} \end{pmatrix}$$

$$U(s) = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{L} \end{pmatrix}$$

Si queremos coger como salida la velocidad bastaría con cambiar C^t , $C^t = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}$

Finalmente $\left[s \cdot I - A\right] = \begin{pmatrix} 0 & 1 & 0 \\ 0 & \frac{-\beta}{J} & \frac{K}{J} \\ 0 & \frac{-K}{L} & \frac{-R}{L} \end{pmatrix}$ $|s \cdot I - A| = 0$ => autovalores ≡ polos

A partir de los datos que nos da el fabricante obtenemos la siguiente matriz de modelado de nuestro motor:

K = 0.022 $\frac{\text{Vs}}{\text{rad}}$

K' = 0.022 $\frac{\text{Nm}}{\text{A}}$

L = $580 \cdot 10^{-6}$ H

R = 5.3 Ω

J = $14 \cdot 10^{-7} Kg\,m^2$

β = 1.1· 10^{-3} mNm

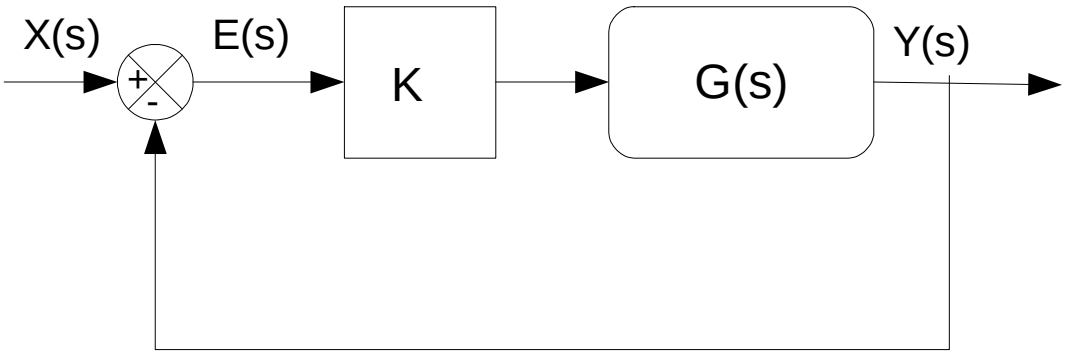
$$\begin{pmatrix} X'_1 \\ X'_2 \\ X'_3 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & -785.71 & 15714{,}28 \\ 0 & -37.93 & -9137.93 \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \frac{1}{L} \end{pmatrix} \cdot u$$

4. Análisis de un controlador proporcional mediante el lugar de raíces.

Al diseñar un sistema de control lineal, se comprueba que el método del lugar de las raíces resulta muy útil, ya que indica la forma en la que se modificarán los polos y ceros en lazo cerrado a a partir de los datos en lazo abierto. Gracias a este método se obtienen aproximaciones excelentes y de forma muy rápida.

Representaremos las curvas del lugar de raíces en el plano complejo S = σ + jw.

Tomaremos como ejemplo la función de transferencia de nuestro motor, y utilizaremos matlab para representar el lugar de raíces de dicha función:



Por tanto $H(s)=\frac{Y(s)}{X(s)}=\frac{K\cdot G(s)}{1+K\cdot G(s)}$

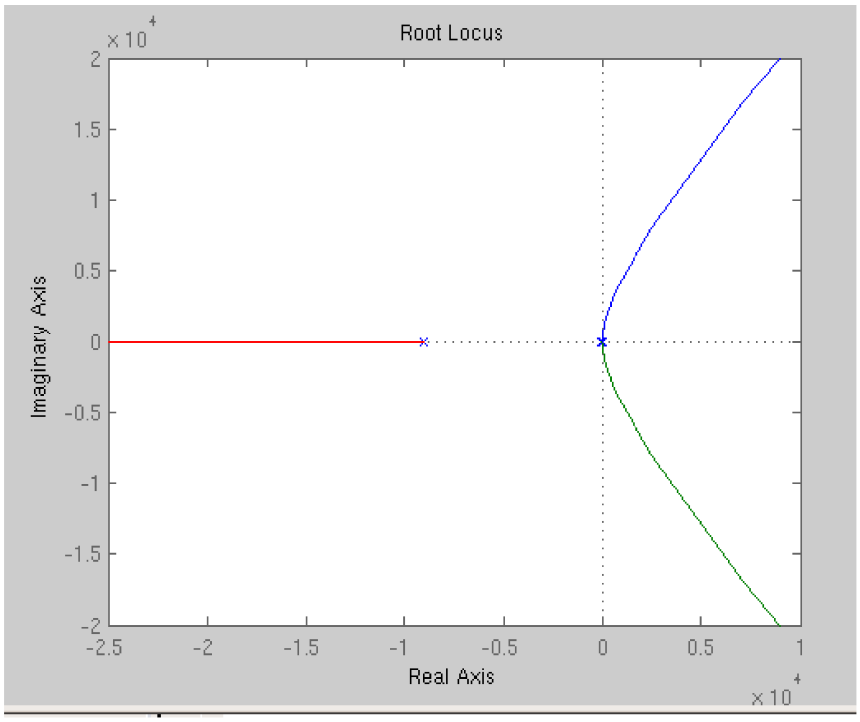


Figura 4.1. - Lugar de raíces del motor.

Observamos que para distintos valores de K logramos desplazar el polo hacia donde creamos conveniente. Lo que obtenemos por tanto es un controlador proporcional que nos permite desplazar el polo hacia la derecha o hacia la izquierda para buscar que la respuesta se estabilice más lenta o más rápidamente. En nuestro caso particular del motor, esto se traduce en que alcance la posición de destino antes o después.

Para un valor de k = 1, obtenemos la siguiente gráfica

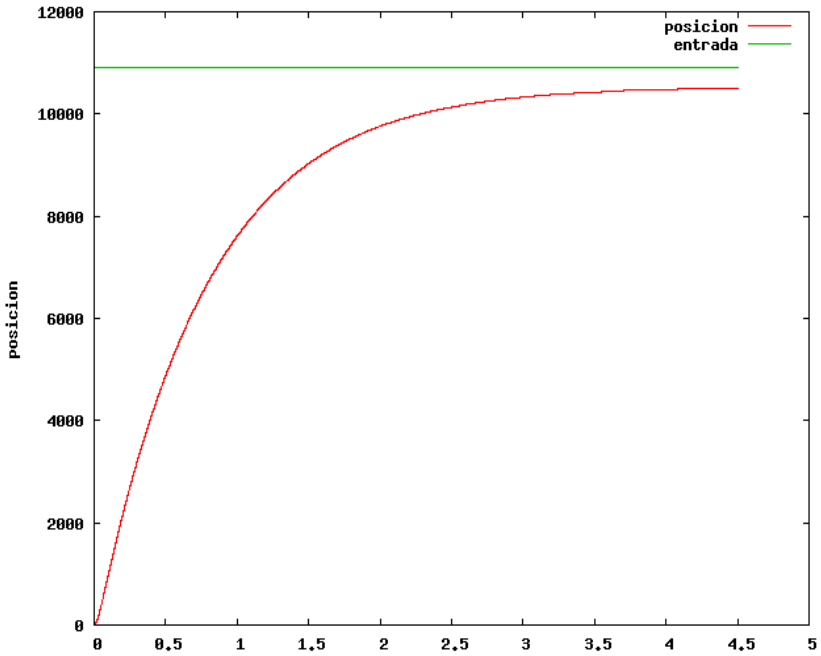


Figura 4.2 - Respuesta del motor en posición.

Mientras que para k =20 obtenemos una respuesta mucho más rápida.

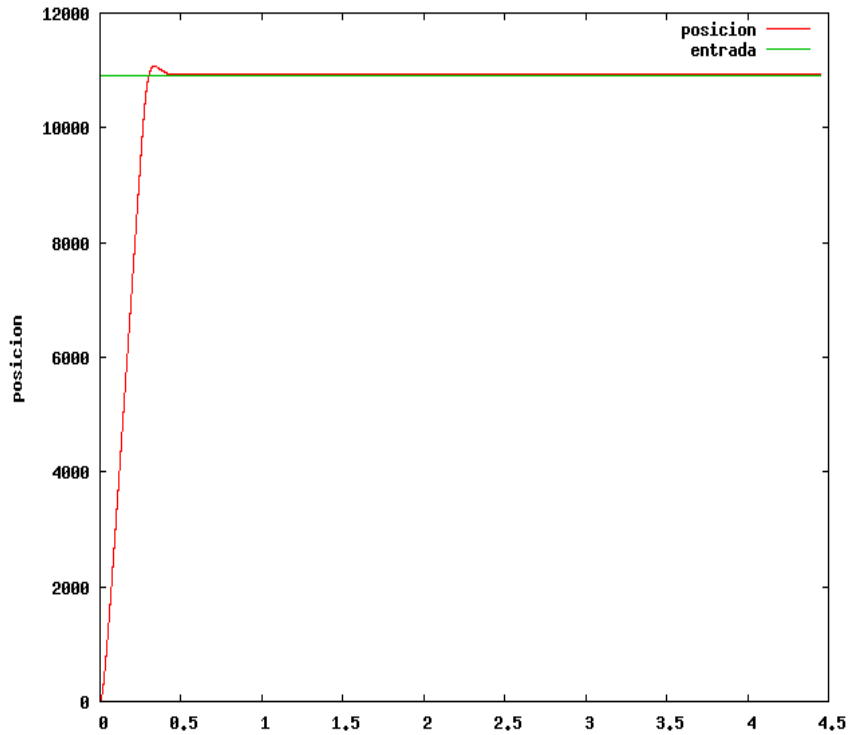


Figura 4.3 - Respuesta del motor en posición.

5. Adaptación del motor mediante un filtro

Para diseñar un PID siguiendo el criterio de Ziegler-Nichols, debemos calcular la K crítica de nuestro sistema mediante el lugar de raices y experimentalmente. Sin embargo el motor es un sistema muy estable que, como vemos en su lugar de raices no se desestabilizaría con una K inferior a 7600.

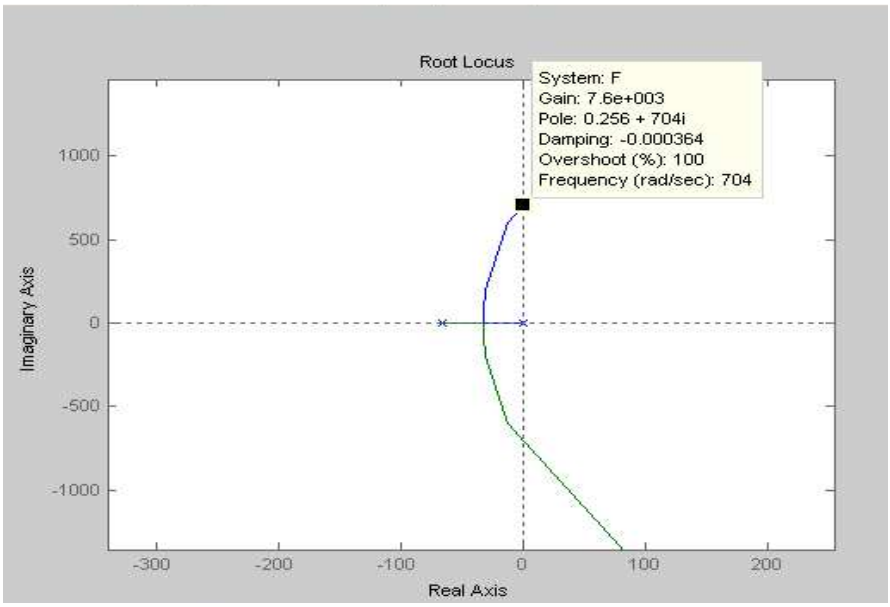
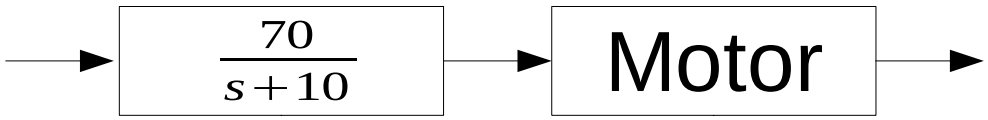


Figura 5.1 - Lugar de raíces del motor.

Por ello decidimos anteponer un filtro al motor que añada un polo y permita desestabilizar el motor con una K inferior.



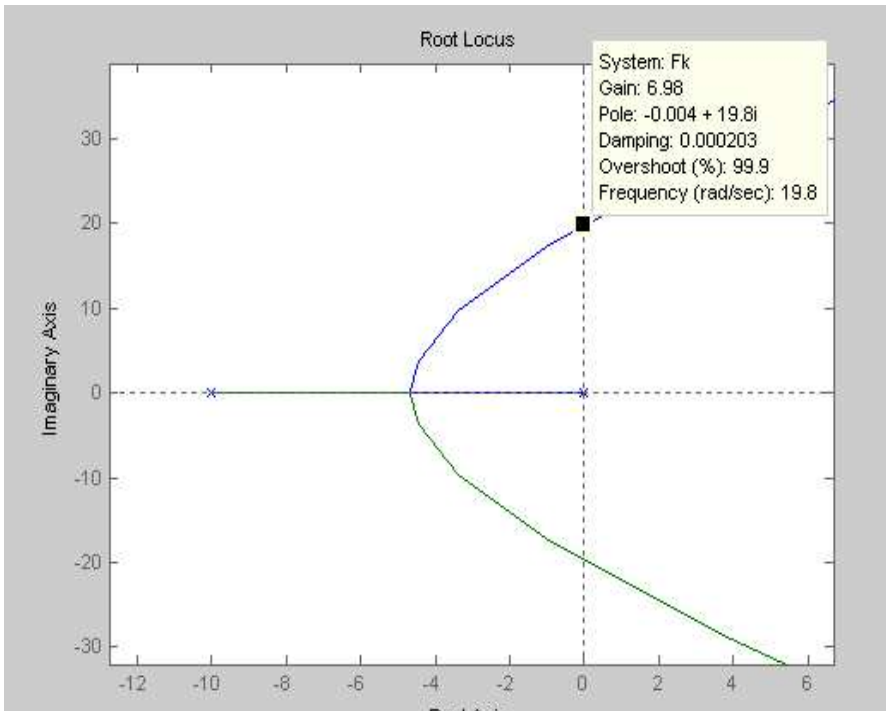
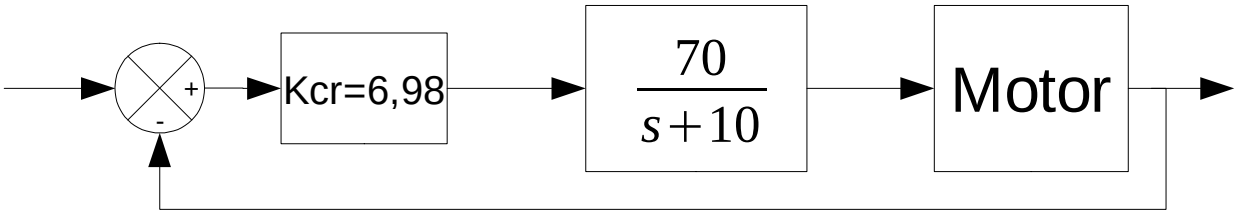


Figura 5.1: Lugar de raices de conjunto motor + filtro

En este caso 6,98 ya es un valor que podemos manejar. Ahora tendremos que calcular el periodo de oscilación del sistema realimentado con esta K. Para ello montamos una realimentación con un controlador proporcional con el valor de K que hemos calculado.



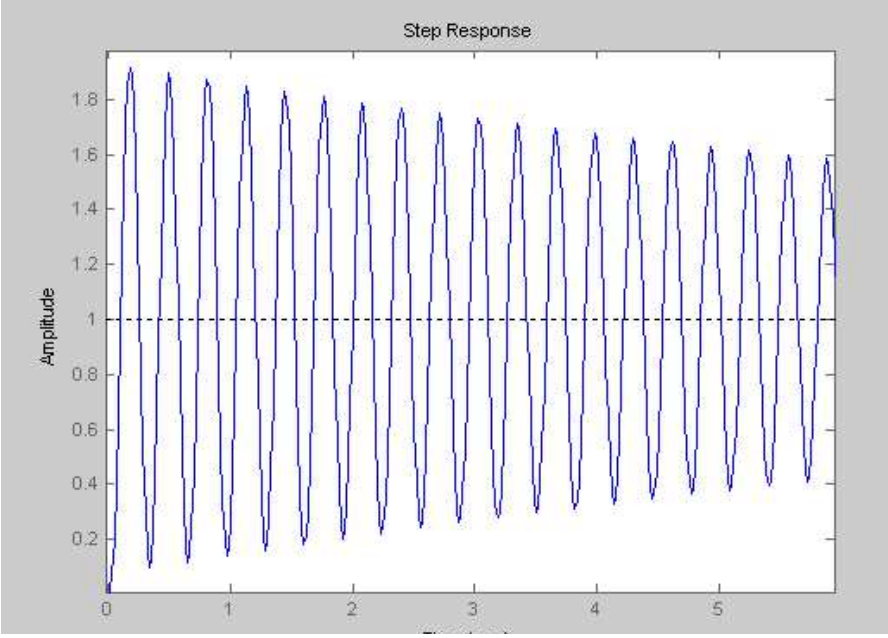


Figura 5.3: Respuesta al escalón de sistema motor + filtro realimentado con controlador proporcional de K = 6.98

De esta manera ya podemos utilizar el método de Ziegler-Nichols para calcular los parámetros del PID.

6. Diseño de controlador PID mediante Ziegler Nichols

6.1 Diseño teórico

Aplicando el criterio de Ziegler-Nichols a los valores de Kcr y Tcr calculados en el punto anterior obtenemos:

K = 4.1880

Ti = 0.1595

Td = 0.0399

$$G_c = 4.188 \cdot [1 + 0.0399 \cdot s + \frac{1}{0.1595 \cdot s}]$$

Obteniendo la siguiente respuesta al escalón al realimentar el motor.

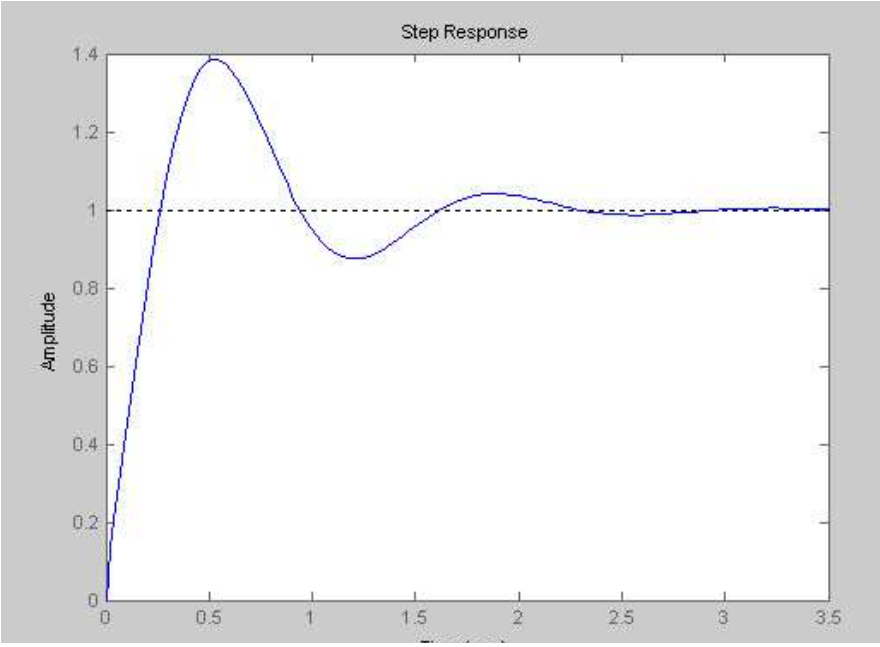


Figura 6.1: Respuesta al escalón de motor realimentado con PID

6.2 Optimización del diseño

A partir de estos valores hemos desarrollado un programa en Matlab que busca mejorar las prestaciones del controlador. Para ello se van variando los parámetros del PID desde un 70% de su valor original hasta un 130 %, registrando los mejores valores. (El código del script que hemos realizado (optimizacionpid.m) está disponible en el anexo.

Obenemos lo siguiente:

K = 5.4444

Ti = 0.2074

Td = 0.0519

$$G_c = 5.4444 \cdot [1 + 0.0519 \cdot s + \frac{1}{0.2074 \cdot s}]$$

Obteniendo una respuesta al escalón:

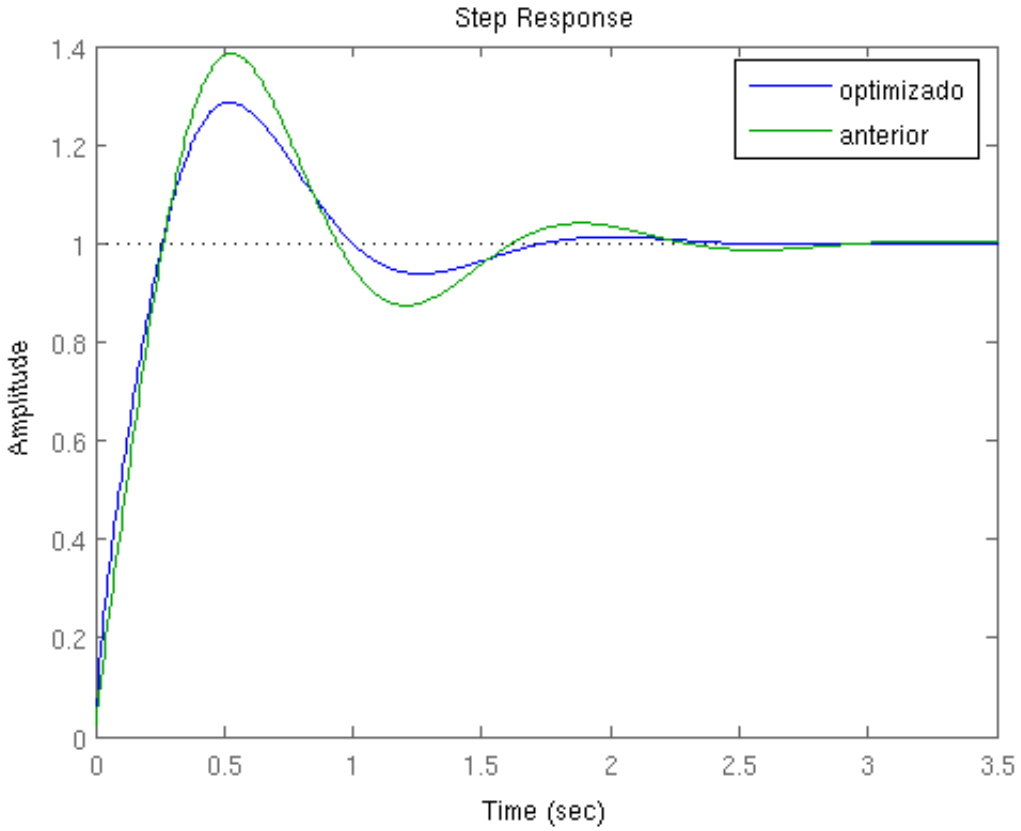


Figura 6.2: Comparación respuestas al escalón de PID original y optimizado.

Con los nuevos parámetros hemos disminuido considerablemente la elongación y el tiempo de respuesta.

6.3 Desarrollo Práctico

Hasta ahora el desarrollo del PID ha sido meramente teórico y simulado en Matlab, aunque desde un principio hemos orientado los resultados para que el controlador fuera realizable en el laboratorio. Por ejemplo a la hora de diseñar el filtro, y poder obtener una K lo suficientemente grande para poder ajustarla en potenciómetros y lo suficientemente pequeña para no provocar saturaciones indeseadas.

A partir de ahora comienzaremos a tener en cuenta problemas de realizabilidad que no se habían tenido en cuenta hasta este momento:

6.3.1 Causalidad del derivador

Como sabemos, en la práctica nunca podremos construir un derivador ideal, ya que se trata de un sistema no causal, para poder realizarlo habrá que añadir un polo que no afecte demasiado al funcionamiento, dando lugar en nuestras ecuaciones a un nuevo coeficiente N, que determinará la ubicación del polo y que tendrá que ser mucho mayor que Td (en principio).

$$Td \cdot s \rightarrow \frac{Td \cdot s}{(Td/N) \cdot s + 1}$$

En Matlab comprobamos que para valores muy diferentes de N no se produce excesivo error:

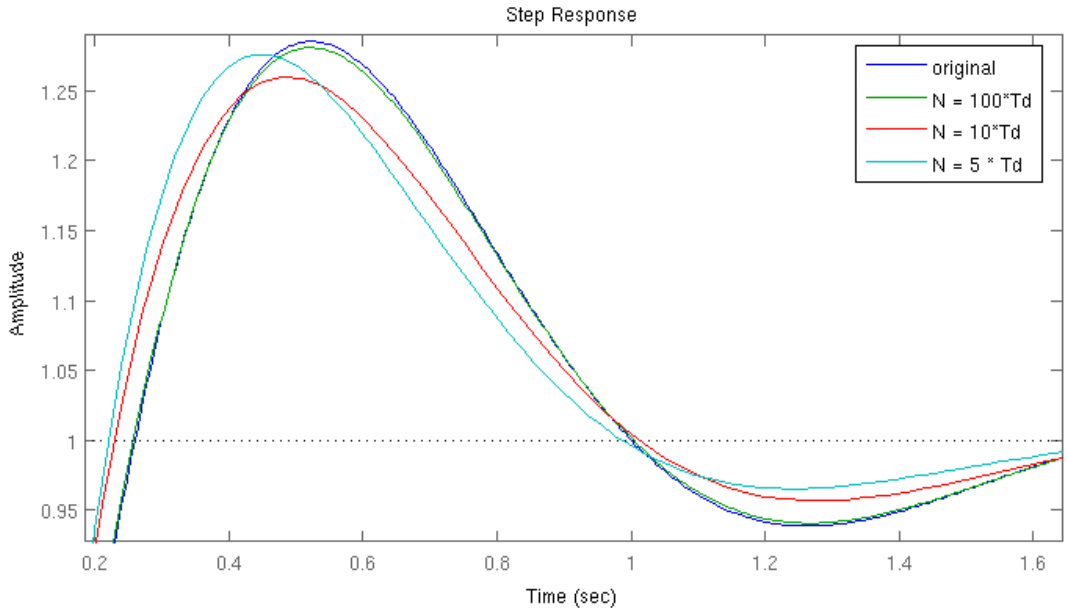


Figura 6.3: Detalle la comparación de respuesta al escalón para distintos valores de N

6.3.2 Limitaciones electrónicas

En el laboratorio contamos con una calculadora analógica que dispone de dos sumadores, dos integradores dos inversores y cuatro potenciómetros. Los sumadores e integradores están realizados con amplificadores operacionales y además permiten atenuar o amplificar la señal por un factor de 10.

A la hora de montar nuestros diseños en la calculadora, contamos con limitaciones en las ganancias que podemos utilizar, además de problemas con la saturación de los amplificadores operacionales especialmente cuando utilizamos la amplificación x10 de algún operador.

Para implementar el controlador PID utilizamos el siguiente esquema:

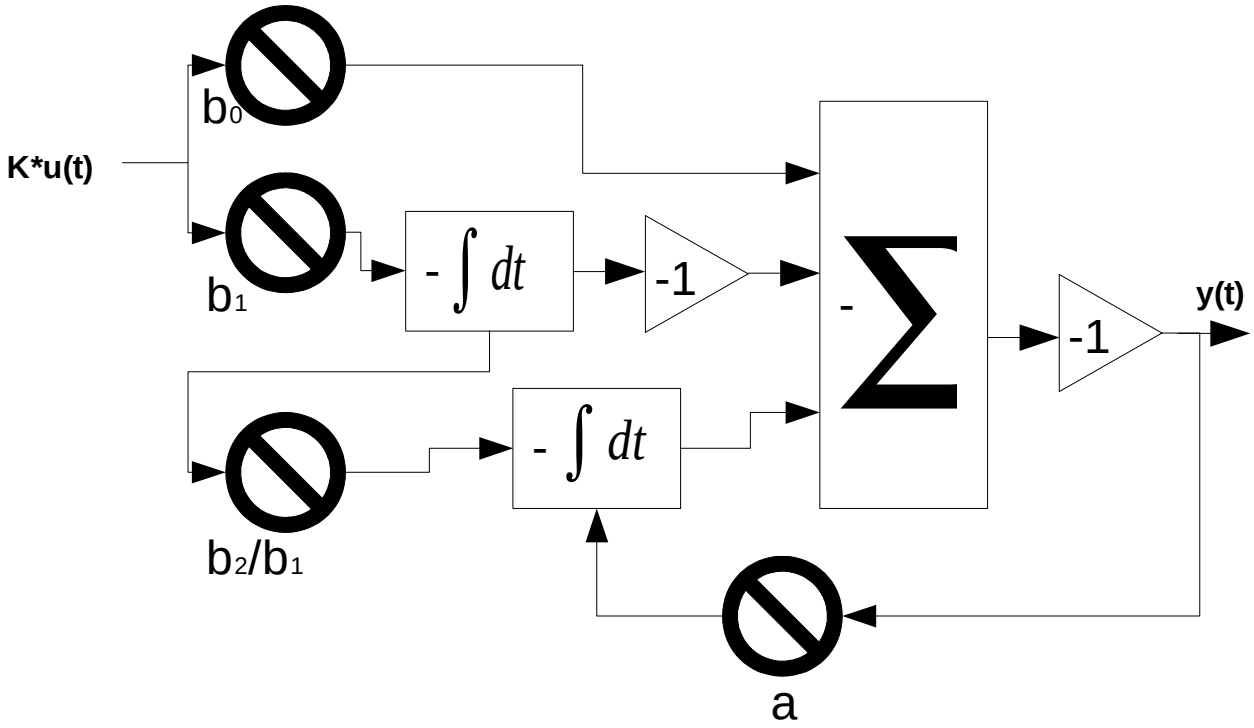


Figura 6.4: Esquema del montaje de PID en calculadora analógica

El montaje en el dominio del tiempo quedará:

$$y(t)=b_0\cdot u(t)+b_1\cdot\int u(t)\,dt+b_2\cdot\int\int u(t)\,dt\,dt-a\cdot\int y(t)\,dt$$

Por lo que la función de transferencia queda:

$$\frac{Y}{U}=\frac{b_0\cdot s^2+b_1\cdot s+b_2}{a\cdot s+1}$$

La K se introducirá digitalmente en la realimentación que realiza el programa de ordenador, quedando fuera del montaje. Por lo tanto el montaje debería ser equivalente a:

$$1+\frac{\tau_d\cdot s}{(\tau_d/N)\cdot s+1}+\frac{1}{\tau_i\cdot s}$$

Identificando obtenemos:

$$b_0 = N + 1$$

$$b_1 = \frac{\tau_d + N \cdot \tau_i}{\tau_d \cdot \tau_i}$$

$$b_2 = \frac{N}{\tau_d \cdot \tau_i}$$

$$a = \frac{N}{\tau_d}$$

Aplicandolo a nuestros parámetros de PID (para un N diez veces Td) obtenemos:

b0 = 1.5190

b1 = 14.8216

b2 = 48.2160

a = 10

Que es un resultado no realizable, ya que entre otras cosas, el potenciómetro b2/b1 tendrá un valor mayor que la unidad, que es algo que no podemos montar. Por ello debemos variar la N para consguir unos valores implementables. Llegando a un valor de N de 1.3 veces Td, obteniendo estos valores, ya definitivos:

b0 = 1.0675

b1 = 6.1216

b2 = 6.2681

a = 1.3000

Para implementarlos, pondremos el sumador en x10, y los potenciómetros b0, b1 y a en un décimo de su valor (0.107, 0.612 y 0.13 respectivamente).

A continuación representamos la simulación de la respuesta al escalón en el PID diseñado originalmente conjuntamente con los cambio que hemos ido realizando hasta llegar al controlador realizable (gráfica roja).

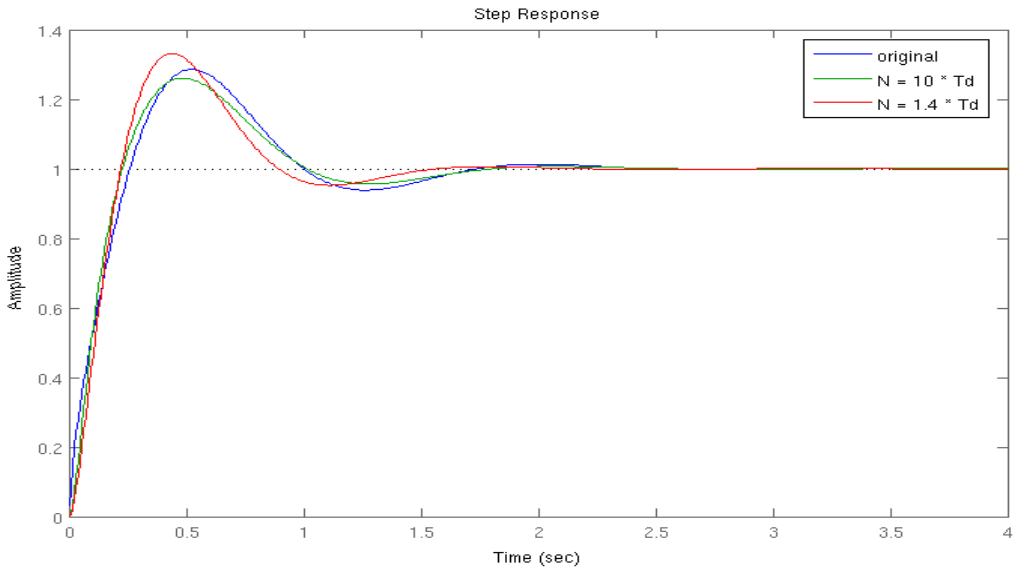


Figura 6.5: Comparación de respuesta del controlador realizable (rojo) con otros de distina N

En la siguiente gráfica se representa la respuesta medida experimentalmente en el laboratorio que concuerda completamente con las simulaciones realizadas.

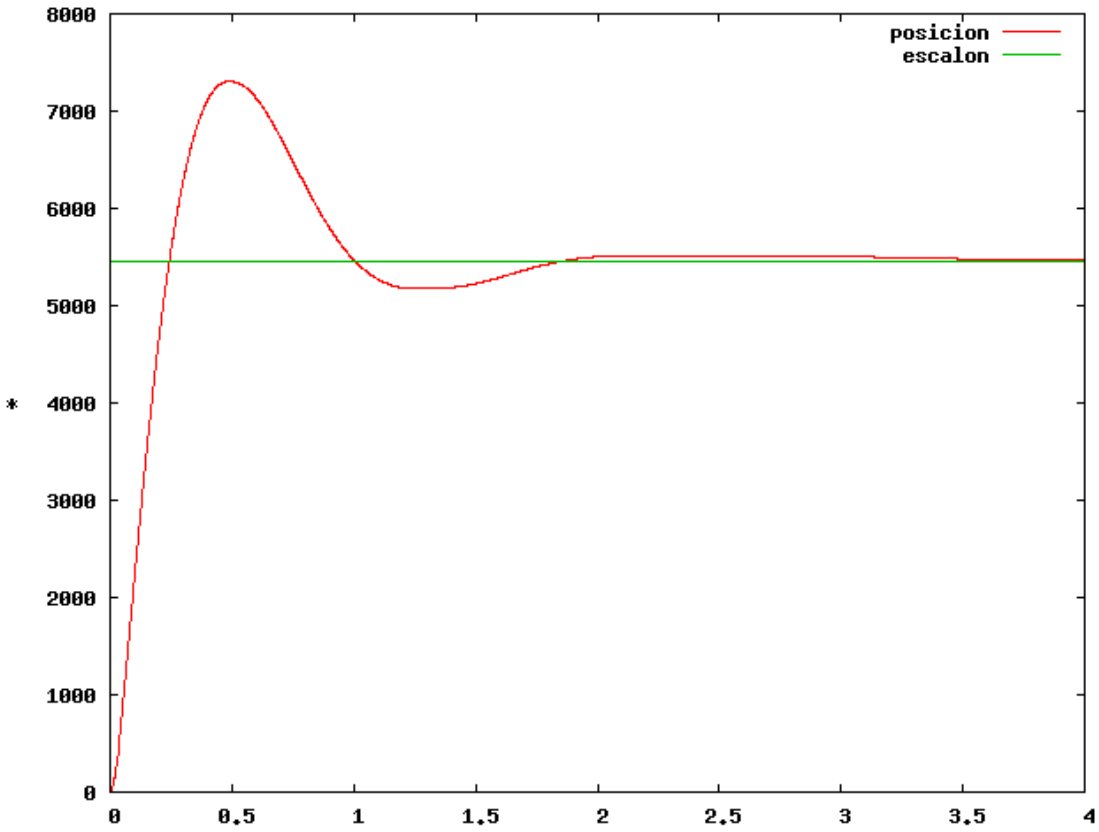


Figura 6.6: Respuesta experimental obtenida en laboratorio

7. Ziegler-Nichols ampliado (utilizando el análisis en el dominio de la frecuencia)

En primer lugar, dibujamos con Matlab el diagrama de Nyquist de la función de transferencia de nuestro motor:

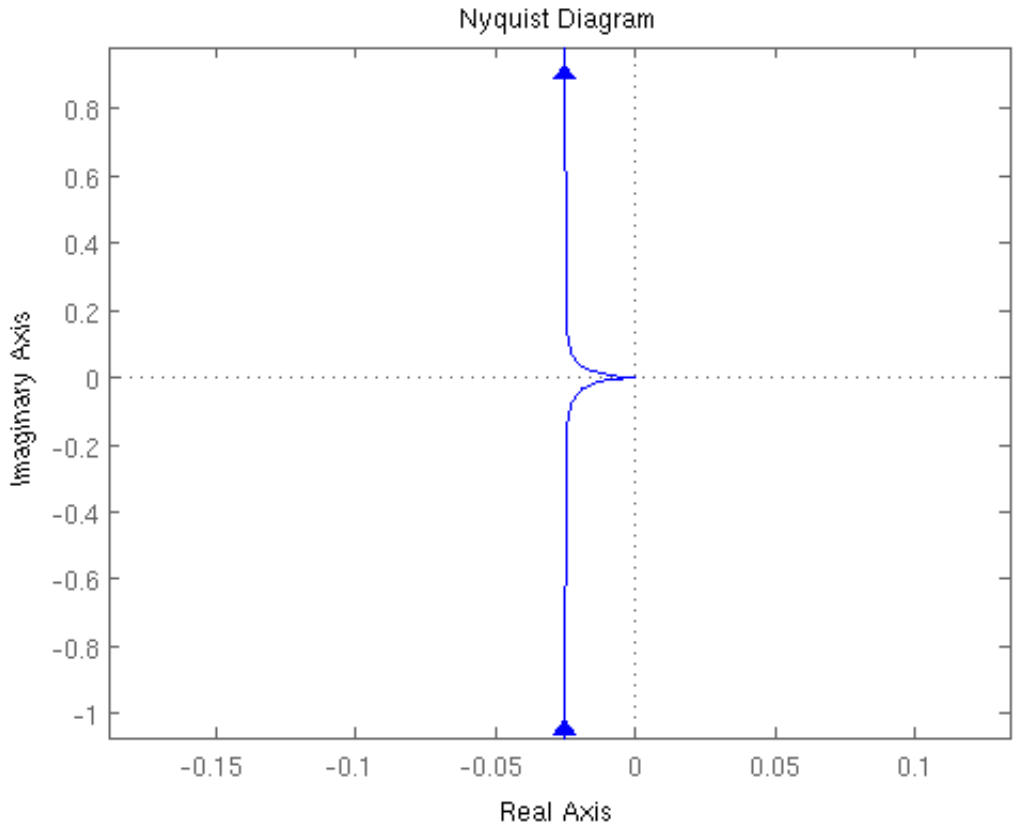


Figura 7.1 - Diagrama de Nyquist.

El procedimiento a seguir es:

-Vamos a escoger un punto A de la gráfica en función de la frecuencia de trabajo. Las frecuencias óptimas para trabajar con el motor se encuentran comprendidas entre unos 0,2 Hz y 4,5 Hz; así que elegimos f_0 = 2,5 Hz,

$$G_p(jw_0) = A = r_a e^{j(\pi + \varnothing_a)}$$

Elegimos a ojo otro punto, que llamaremos B, de la gráfica al que suponemos que se debería desplazar A para cumplir los requisitos exigidos

$$G_p(jw_0) = B = r_b e^{j(\pi + \varnothing_b)}$$

Ahora calculamos el vector complejo que desplazaría dicho punto y que sería la respuesta del controlador:

$$G_p(jw_0) = C = r_c e^{j\varnothing_c}$$

Este punto C cumple que:

$$r_b e^{j(\pi+\varnothing_n)} = r_a \cdot r_c e^{j(\pi+\varnothing_b+\varnothing_c)}$$

Si identificamos cada término, obtenemos el módulo y la fase de la respuesta del controlador en función de los puntos inicio y destino:

$$r_c = \frac{r_b}{r_a}$$

$$\varnothing_c = \varnothing_b - \varnothing_a$$

En función del módulo y fase del punto C calculamos los parámetros del controlador PID:

$$K_p = r_c \cos(\varnothing_c)$$

$$W_0 \cdot T_d - \frac{1}{(w_0 \cdot T_i)} = \tan(\varnothing_c)$$

Para hallar el valor de Td y Ti sólo tenemos una ecuación. Uno de los métodos que más se usan para hallarlos es usar la relación Td=αTi. Sustituyendo en la ecuación anterior y dejando como única incógnita Ti, nos queda una ecuación cuadrática, cuya solución positiva es:

$$T_i = \frac{(\tan\frac{(\varnothing_c)}{\alpha} + \sqrt{\left(\frac{(tan\varnothing_c)}{\alpha}\right)^2 + \frac{4}{\alpha}})}{(2 \cdot w)}$$

Una vez hallados estos parámetros, ya puedo escribir la función de transferencia del controlador PID:

$$G_c(j \cdot w) = K_p \cdot (1 + s \cdot T_d + \frac{1}{(s \cdot T_i)})$$

A continuación vamos a realizar una descripción de los pasos seguidos en el laboratorio para obtener los parámetros de nuestros sistema:

1. En primer lugar elegimos un punto A en base a nuestro diagrama de Nyquist obtenido a partir de la función de transferencia de nuestro motor. Optamos por A = -0.0219-j0.0579 para la cual tenenmos una W_0 = 15.5 rad/s y por lo tanto una frecuencia de unos f_0 = 2,5 Hz que se ajusta dentro del margen que perseguíamos.

   El módulo de A es 0.0619 y su fase es 4.35 rad.

   Ahora tenemos que buscar otro punto B cuya fase sea mayor que la de A para que la diferencia sea positiva. Iterando con diferentes valores que permitan que nuestra K_p se ajuste a la obtenida en los experimentos con la calculadora analógica (esta K_p era la que permitía la realizabilidad del montaje en el laboratorio evitando la saturación de los componentes) .

El punto B obtenido finalmente es B = -0.0249-j0.352 cuyo módulo es 0.3528 y su fase 4.64 rad. Obtenemos con estos valores una K_p = 5.465 muy próxima a 5.44 que es la obtenida experimentalmente. Una vez obtenidos estos valores y cogiendo α = 0.25 obtenemos

$$T_i = 0.1728$$
$$T_d = 0.0432$$

En matlab ejecutamos la siguiente secuencia de código para ver la respuesta final del motor y obtenemos la siguiente gráfica (tomando como entrada una señal escalón, step):

```
>> H=tf(1)

Transfer function:
1

>> J=tf([0.0432, 0],[1])

Transfer function:
0.0432 s

>> K=tf([5.787],[1,0])

Transfer function:
5.787
-----
  s

>> F=5.465*(H+J+K)

Transfer function:
0.2361 s^2 + 5.465 s + 31.63
----------------------------
           s

>> G=tf([40],[1,40,0])

Transfer function:
    40
----------
s^2 + 40 s

>> D=realimenta(F,G)

Transfer function:
   9.444 s^2 + 218.6 s + 1265
---------------------------------
s^3 + 49.44 s^2 + 218.6 s + 1265

>> step(D)
```

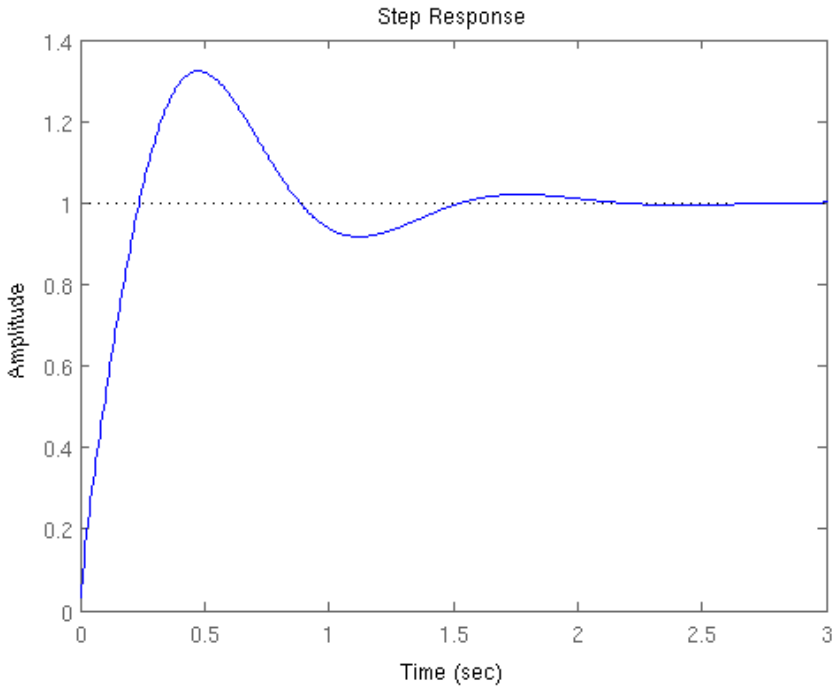


Figura 7.2 – Respuesta del motor.

La fórmula final obtenida con estos valores es :

$$G_c(j\cdot w)=\frac{(0.2361\cdot s^2+5.465\cdot s+31.63)}{s}$$

El problema de la fórmula $G_c(j\cdot w)=K_p\cdot(1+s\cdot T_d+\frac{1}{(s\cdot T_i)})$ es la inestabilidad introducida por el término $s\cdot T_d$. Para solucionar esto, optamos por la siguiente fórmula modificada, cuya aproximación es bastante buena con N relativamente grande:

$$G_c(j\cdot w)=K_p\cdot(1+s\cdot\frac{T_d}{(\frac{T_d\cdot s}{N}+1)}+\frac{1}{(s\cdot T_i)})$$

Uno de los potenciómetros de la calculadora analógica se utiliza para controlar el parámetro $\frac{b_2}{b_1}$. Por tanto debemos buscar una N tal que $\frac{b_2}{b_1}$ < 1.

Iterando con algunos valores obtenemos finalmente N = 1.2 · T_d . Para realizar estas iteraciones hemos utilizado un archivo de matlab implementado por nosotros mismos, llamado “calculoab.m”.

La función de transferencia obtenida es:

$$G_c(j\cdot w)=\frac{(4.554\cdot s^2+32.05\cdot s+31.63)}{(0.8333\cdot s^2+s)}$$

8. Diseño de un sistema de control con dos grados de libertad.

Sea el sistema de control con dos grados de libertad que se muestra en la figura, se pretende el diseño de los controladores G_{c1} y G_{c2} de forma que la suma de ambos constituya un PID para controlar nuestra planta G_p que será el motor modelado en apartados anteriores.

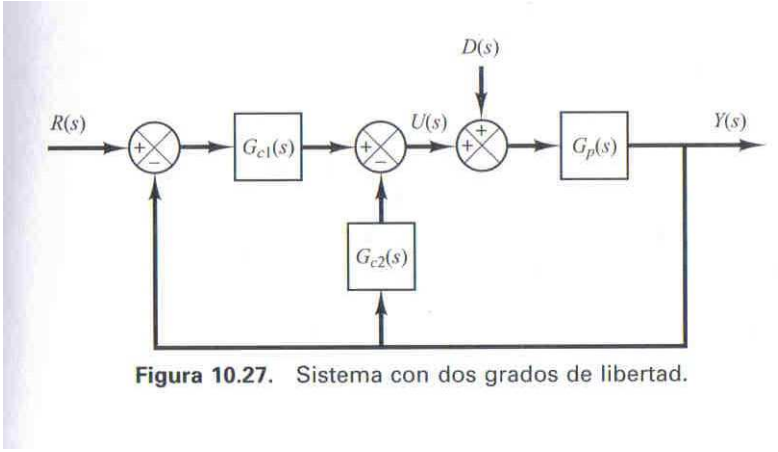


Figura 10.27. Sistema con dos grados de libertad.

Utilizaremos el modelo del motor simplificado obtenido en el laboratorio para la simplificación de los cálculos:

$$G_p = \frac{40}{s(s+40)}$$

Nuestras especificaciones expuestas a continuación nos impondrán a priori las condiciones a imponer a nuestro diseño:

Sobreelongación: $5\% \leq M_p \leq 10\%$

Tiempo de asentimiento: $t_s \leq 1$sg (en base al criterio del 5%)

Tiempo de subida: $t_r \leq 0.5$sg

Error en estado estacionario para señales de referencia rampa y parábola nulos

Que la respuesta a una entrada escalón unitario tienda a cero rápidamente

Para empezar el análisis de nuestro sistema es conveniente identificar la función de transferencia de nuestro sistema teniendo en cuenta que posee dos entradas diferentes: la señal de referencia y una entrada de perturbación. Aplicando el principio de superposición obtenemos los siguientes resultados:

$$\frac{Y(s)}{D(s)} = \frac{G_p}{1+G_p(G_{c1}+G_{c2})}$$

$$\frac{Y(s)}{R(s)} = \frac{G_p G_{c1}}{1+G_p(G_{c1}+G_{c2})}$$

A partir de ahora llamaremos:

$$G_c = G_{c1} + G_{c2}$$

La ecuación característica para ambas funciones de transferencia es la misma por tanto tendrán los mismos polos.

A continuación se define:

$$G_c = \frac{k(s+\alpha)(s+\beta)}{s} = \frac{k[s^2+(\alpha+\beta)s+\alpha\beta]}{s}$$

De esta manera se asume que G_c es un controlador PID y se diseña así para satisfacer los requisitos sobre la respuesta a la entrada de perturbación en escalón D(s).

En este caso la función de transferencia en lazo cerrado con respecto a la entrada de perturbación sustituyendo la función de trasferencia de la planta y el controlador queda de la siguiente manera:

$$\frac{Y(s)}{D(s)} = \frac{40}{s(s+40)+\dfrac{40k(s+\alpha)(s+\beta)}{s}} = \frac{40s}{s^2(s+40)+40k(s+\alpha)(s+\beta)}$$

Se observa que la presencia de s en el numerador de Y(s)/D(s) asegura que la respuesta en estado estacionario a la entrada de perturbación en escalón es cero si aplicamos el teorema del valor final.

Suponemos que los polos en lazo cerrado dominante queremos que sean complejos conjugados y vendrán dados por:

$$s = -a \pm jb$$

y el polo en lazo cerrado lo suponemos real localizado en :

$$s = -c$$

Llegada esta parte del problema debemos tener en cuenta como requisitos impuestos que la respuesta a la entrada de perturbación se amortigüe rápidamente, la sobreelongación máxima en la respuesta a la entrada de referencia escalón unitario, el tiempo de asentimiento y que los errores en estado estacionario en las respuestas a entradas de referencia en rampa y aceleración sean cero.
Para buscar los valores adecuados a “a”, “b” y “c” comenzamos probando con valores como los que siguen:

$$2 \leq a \leq 5 \qquad 1 \leq b \leq 3 \qquad 10 \leq c \leq 14$$

Los valores que se escojan para a y b serán determinantes para que la respuesta de la entrada en escalón para la perturbación se amortigüe rápidamente ya que se trata del valor de los polos en lazo cerrado dominantes.

El denominador de Y(s)/D(s) será determinante también en la respuesta para la entrada de referencia ya que coincide con el denominador de Y(s)/R(s).

Este denominador queda de la siguiente forma:

$$s^2(s+40)+40k(s+\beta)(s+\alpha)=s^3+40(1+k)s^2+40k(\alpha+\beta)s+40k\alpha\beta=$$

$$(s+a+jb)(s+a-jb)(s+c)=s^3+(2a+c)s^2+(a^2+b^2+2ac)s+(a^2+b^2)c$$

De esta manera, identificando podemos obtener los siguientes valores que nos determinan G_c :

$$k=\frac{2a+c-40}{40}$$
$$(\alpha+\beta)=\frac{a^2+b^2+2ac}{40k}$$
$$\alpha\beta=\frac{(a^2+b^2)c}{40k}$$

Para la satisfacción del requisito de que los errores en estado estacionario en las respuestas a entradas de referencia en rampa y aceleración sean cero desarrollamos la función de transferencia en lazo cerrado Y(s)/R(s) como sigue:

$$\frac{Y(s)}{R(s)}=\frac{(2a+c)s^2+(a^2+b^2+2ac)s+(a^2+b^2)c}{s^3+(2a+c)s^2+(a^2+b^2+2ac)s+(a^2+b^2)c}$$

Para buscar los valores utilizaremos un método computacional programando en el script de Matlab pid2grados.m disponible en el anexo.

Los resultados de la ejecución son los siguientes:
Resultado pid2grados:

n	a	b	c	elong	ts	tsub	k	alfamasbeta	alfaporbeta
1.0000	2.2000	1.0000	14.0000	1.1384	0.5700	0.1100	-0.5400	-3.1222	-3.7852
2.0000	2.2000	1.0000	13.8000	1.1394	0.5700	0.1100	-0.5450	-3.0532	-3.6969
3.0000	2.0000	1.4000	14.0000	1.1380	0.6000	0.1100	-0.5500	-2.8164	-3.7927
4.0000	2.0000	1.4000	13.8000	1.1391	0.6000	0.1100	-0.5550	-2.7550	-3.7049
5.0000	2.0000	1.2000	14.0000	1.1352	0.6000	0.1100	-0.5500	-2.7927	-3.4618
6.0000	2.0000	1.2000	13.8000	1.1363	0.6000	0.1100	-0.5550	-2.7315	-3.3816
7.0000	2.0000	1.2000	13.6000	1.1373	0.6000	0.1100	-0.5600	-2.6714	-3.3029
8.0000	2.0000	1.2000	13.4000	1.1383	0.6100	0.1100	-0.5650	-2.6124	-3.2255
9.0000	2.0000	1.2000	13.2000	1.1394	0.6100	0.1100	-0.5700	-2.5544	-3.1495
10.0000	2.0000	1.0000	14.0000	1.1329	0.5900	0.1100	-0.5500	-2.7727	-3.1818
11.0000	2.0000	1.0000	13.8000	1.1339	0.6000	0.1100	-0.5550	-2.7117	-3.1081
12.0000	2.0000	1.0000	13.6000	1.1349	0.6000	0.1100	-0.5600	-2.6518	-3.0357
13.0000	2.0000	1.0000	13.4000	1.1359	0.6100	0.1100	-0.5650	-2.5929	-2.9646
14.0000	2.0000	1.0000	13.2000	1.1369	0.6100	0.1100	-0.5700	-2.5351	-2.8947
15.0000	2.0000	1.0000	13.0000	1.1380	0.6200	0.1100	-0.5750	-2.4783	-2.8261
16.0000	2.0000	1.0000	12.8000	1.1391	0.6200	0.1100	-0.5800	-2.4224	-2.7586

Escogemos la fila número 6.

Con los valores escogidos obtenemos sustituyendo:

$$G_c = \frac{-0.555s^2 + 1.516s + 1.877}{s}$$

Y la función de transferencia Y(s)/D(s) queda como sigue:

$$\frac{Y(s)}{D(s)} = \frac{40s}{s^3 + 17.8 + 60.64s + 75.08}$$

A continuación determinaremos G_{c1} por el método de asignación de ceros. Como sabemos podemos expresar Y(s)/R(s) como sigue:

$$\frac{Y(s)}{R(s)} = \frac{G_p G_{c1}}{1 + G_p G_c} = \frac{\frac{40}{s(s+40)} G_{c1}}{1 + \frac{40}{s(s+40)} \frac{-0.555s^2 + 1.516s + 1.877}{s}} = = \frac{40s\, G_{c1}}{s^3 + 17.8s^2 + 60.64s + 75.08}$$

Ahora necesitamos satisfacer los requisitos para las respuestas a las entradas en escalón rampa y aceleración.

Como el numerador contiene s G_{c1} debe incluir un integrador para cancelarla. Esta s en el numerador era necesaria en la función de transferencia Y(s)/D(s) en lazo cerrado para la obtención de error nulo en estado estacionario para la entrada de perturbación escalón, pero en Y(s)/R(s) no precisa tenerla.

Finalmente aplicamos la técnica de asignación de ceros igualando:

$$40s\, G_{c1} = 17.8s^2 + 60.64s + 75.08$$

Por tanto:

$$G_{c1} = 0.455s + 1{,}561 + \frac{1.877}{s}$$

Al final nos queda G_{c1} como un controlador PID.

Finalmente obtenemos G_{c2} despejando de $G_c = G_{c1} + G_{c2}$ y obtenemos:

$$G_{c2} = -s$$

Por último Y(s)/R(s) la podriamos obtener o bien a partir de los valores obtenidos de a, b y c o

sustituyendo la G_{c1} obtenida en la expresión de arriba.

Una vez tenemos las funciones de transferencia podemos simular con Matlab las respuestas al escalón correspondientes.
Respuesta al escalón de la función de transferencia Y(s)/R(s) simulada con Matlab:

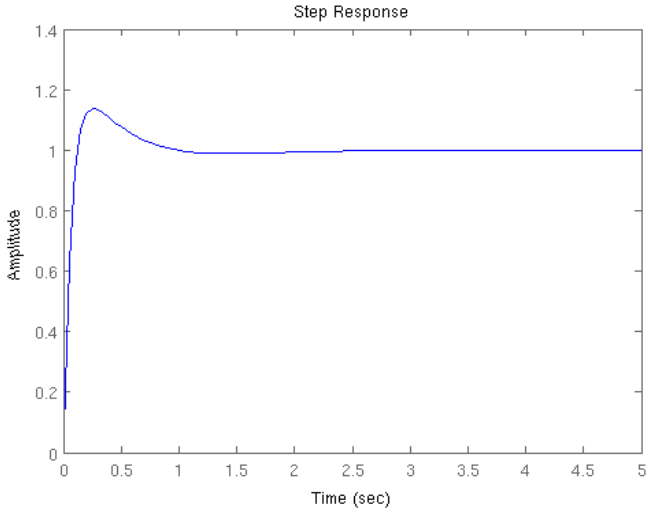


Figura 8.1- Respuesta al escalón de la función de transferencia Y(s)/R(s) simulada con Matlab.

Respuesta al escalón tomando la entrada de perturbación:

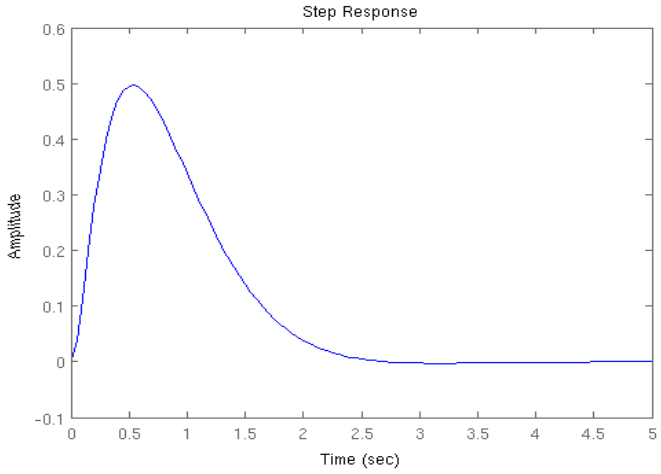


Figura 8.2- Respuesta al escalón tomando la entrada de perturbación.

Respuesta total de la suma aplicando superposición:

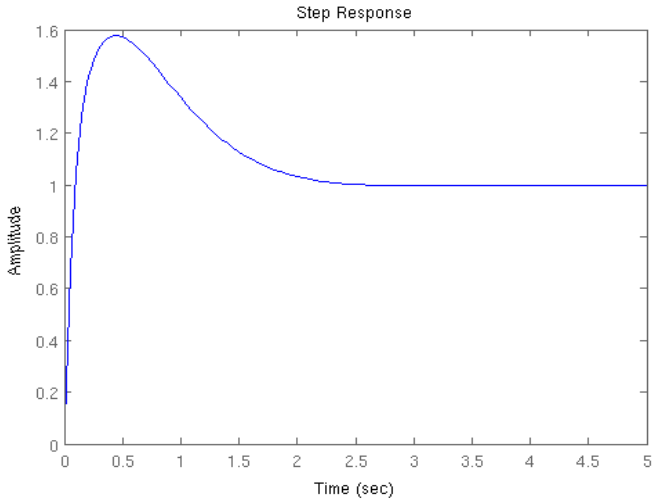


Figura 8.3 - Respuesta total de la suma aplicando superposición.

Respuesta a la entrada de referencia rampa:

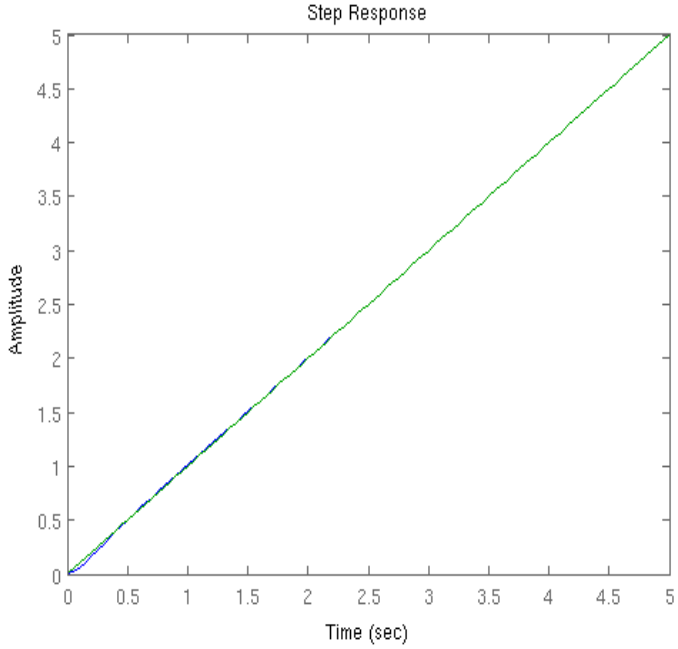


Figura 8.4 - Respuesta a la entrada de referencia rampa.

Respuesta a la entrada de referencia parábola:

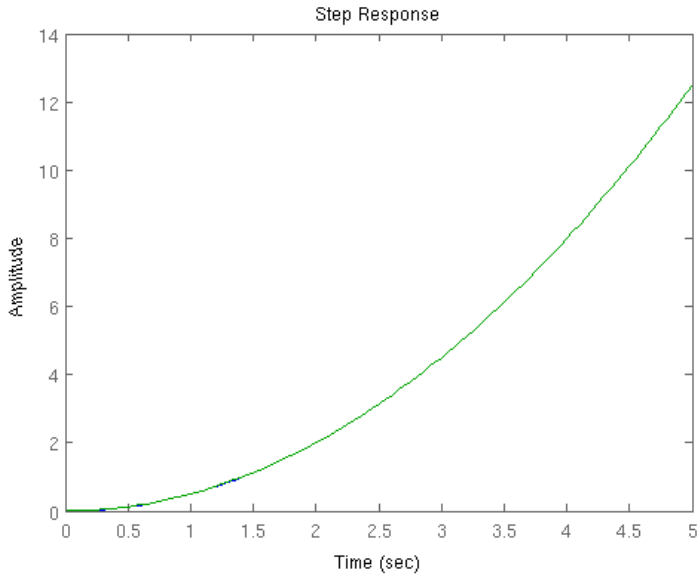


Figura 8.5 - Respuesta a la entrada de referencia parábola.

Simulación en el laboratorio sin perturbación:

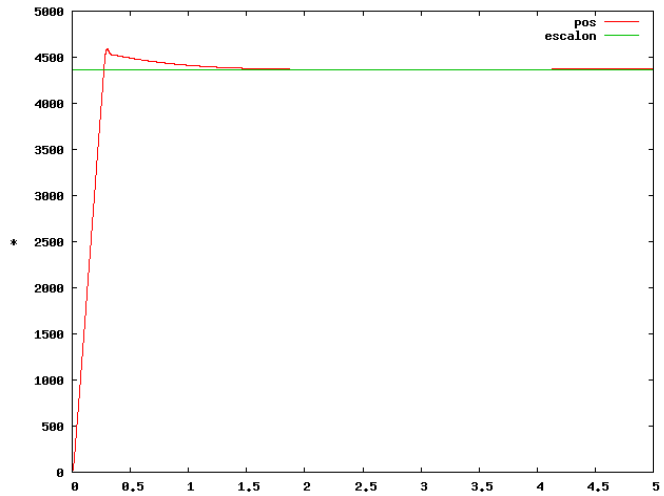


Figura 8.6: Simulación en el laboratorio sin perturbación

Simulación en el laboratorio con perturbación:

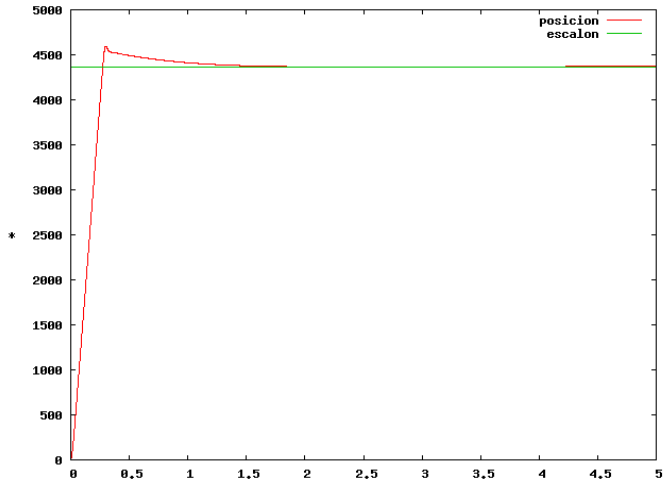


Figura 8.7: Simulación en el laboratorio con perturbación

Nota: en la simulación en el laboratorio al utilizar el modelo del motor simplificado ha sido necesario corregir la ganancia introducida por el programa multiplicando la K_p correspondiente al G_{c1} por un factor de 65 que pide el modo 12, para que encajen las respuestas simuladas teóricamente y en el laboratorio con el motor real.

La implementación consiste en montar en la calculadora analógica el controlador PID y simularlo con el modo 12 al que le tenemos que meter la k del PID mencionado anteriormente y la función del controlador G_{c2} que en nuestro caso de trata de un derivador.

Para introducir a dicho programa la k del controlador G_{c1} debemos expresar dicha función de la siguiente manera:

$$G_{c1} = 0.455s + 1{,}561 + \frac{1.877}{s} = k\left(1 + T_D s + \frac{1}{T_I s}\right)$$

Donde identificando obtenemos:

$$k = 1.516; \quad T_D = 0.2935; \quad T_I = 0.3514$$

Para poder implementarlo en la calculadora analógica de la que disponemos en el laboratorio calculamos los valores necesarios explicados en apartados anteriores correspondientes a la implementación de un PID. Para el cálculo ejecutamos un programita que creamos anteriormente para ir un poco más rápido:

b0= N+1
b1=(Td+N*Ti)/(Td*Ti)
b2=N/(Td*Ti)
a = -N/Td

Tomando como N= 1.5 T_D

Anexo: Código fuente MATLAB

Durante el desarrollo, MATLAB ha sido una herramienta fundamental para realizar los cálculos. Para adaptarlo a nuestras necesidades hemos desarrollado diversos scripts y funciones que nos han facilitado el trabajo, a continuación explicamos cada una de ellas.

modelomotor.m, modelomotorsimplificado.m

Entrada: void

Salida: Función de transferencia del motor en posición, ya sea el real o el simplificado:

Código: Para el motor simplificado: (El código para modelomotor.m es igual cambiando los valores de num y den)

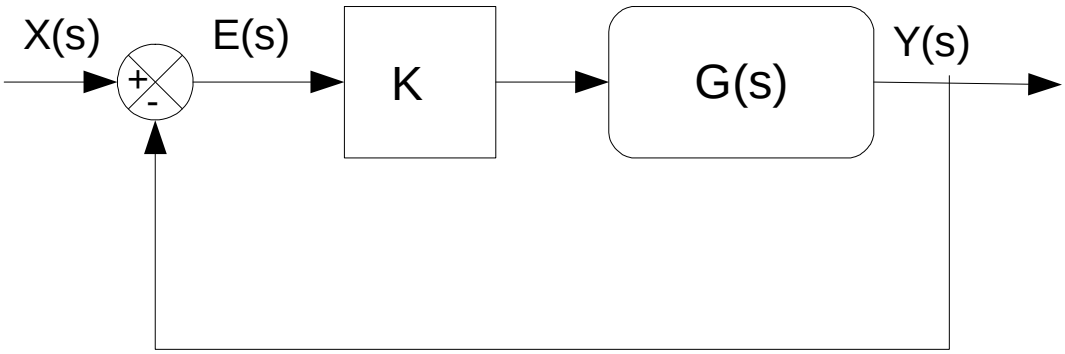
```
function [F] = modelomotorplicado()
    num = [40];
    den = [1 40 0];
    F = tf(num,den)

end
```

realimenta.m

Entrada: Función de transfrencia de un sistema G y de un controlador K (el término K no tiene nada que ver con que el controlador sea proporcional, nos sirve cualquier sistema lineal).

Salida: Función de transferencia del sistema completo realimentado siguiendo el siguiente esquema:



Código:

```
function [F] = realimenta(G,K)
    G1 = G*K;
    F = feedback(G1,1);
end
```

estable.m

Entrada: Función de transferencia de un sistema.

Salida: 1 si el sistema es estable, 0 en caso contrario. Para calcularlo se aplica el criterio de polos en el semiplano negativo.

Código:

```
function[estabilidad] = estable(F)
    [num,den] = tfdata(F,'v');
    polos = roots(den);
    estabilidad = 1;
    for i = 1:size(polos,1);
        estabilidad = estabilidad & (real(polos(i))<0);
    end
end
```

calculoab.m

Función que nos calcula los valores de los parámetros a, b0, b1 y b2 para el montaje en la calculadora analógica del PID.

Entrada: Valor de los parámetros Ti, Td y N del PID.

Salida: Valor de los parámetros a, b0, b1 y b2.

Código

```
function[a, b0, b1, b2] = calculoab(Ti,Td,N)
      b0= N+1;
      b1=(Td+NTi)/(TdTi);
      b2=N/(TdTi);
      a = N/Td;
end
```

respuestarampa.m

Función que calcula la respuesta a la rampa de un sistema.

Entrada: Función de transferencia del sistema

Salida: Gráfica de la respuesta a la rampa.

```
function[] = respuestarampa(F)
      polorigen = tf([1],[1 0]);
      step(F*polorigen);
end
```

respuestaparabola.m

Función que calcula la respuesta a la parábola de un sistema.

Entrada: Función de transferencia del sistema

Salida: Gráfica de la respuesta a la parábola

```
function[] = respuestarampa(F)
      polodobleorigen = tf([1],[1 0 0]);
      step(F*polodobleorigen);
end
```

optimizacionpid.m

Script que a a partir de unos valores de los parámetros del PID, los varía desde un 60% a un 130% para mejorar resultados. Guarda los valores en una variable llamada *table*.

```
clear;
t = 0:0.001:1;
n = 1;
F = modelomotorsimplificado;
Kzn = 4,1880
Tizn = 0.1595;
Tdzn = 0.0399;

for i = 1:7;
     K(i) = Kzn*(0.6+i*0.1);
     for j=1:7;
         Ti(j) = Tizn*(0.6+j*0.3);
         for h = 1:7;
             Td(h) = Tdzn*(0.6+h*0.3);
             N(h)  = 10*Td(h);
             P = tf([1],[1]);
             D = tf([Td(h) 0],[1]);
             I = tf([1],[Ti(j) 0]);

             Gc = K(i)*(P + I + D);
             Fr = realimenta(F, Gc);

             y = step(Fr,t);
             m = max(y);
             s = 1001;
             while y(s) > 0.95 & y(s) < 1.05;
                 s = s-1;
             end
             ts = (s-1)*0.001;
             if estable(Fr)
                  if m<1.30 & m>1.05 & ts<1.80;
                        table(n,:) = [K(i) Ti(j) Td(h) N(h) m ts];
                        n = n+1;
                  end
            end
         end
     end
end
```

pid2grados.m

Script que a partir de unos rangos de la ubicación de los polos, la planta del motor y unas especificaciones, calcula el parámetro Gc del PID de dos grados de libertad por el método de asignación de polos. En relaidad se trata de una revisión del código que aparece en el libro “Ingenieria moderna de control” de Katsuhiko Ogata.

```
clear;

t = 0:0.01:4;
n = 1;
for i = 1:16;
     a(i) = 5.2-i*0.2;
     for j=1:11;
         b(j) = 3.2-j*0.2;
         for h = 1:21;
             c(h) = 14.2-h*0.2;
             k = (2*a(i)+c(h)-40)/40;
             alfamasbeta = (a(i)^2+b(j)^2+2*a(i)*c(h))/(40*k);
             alfaporbeta = ((a(i)^2+b(j)^2)*c(h))/(40*k);
             num=[0 2*a(i)+c(h) a(i)^2+b(j)^2+2*a(i)*c(h)
                  (a(i)^2+b(j)^2)*c(h)];
             den=[1 2*a(i)+c(h) a(i)^2+b(j)^2+2*a(i)*c(h)
                  (a(i)^2+b(j)^2)*c(h)];
             y = step(num,den,t);
             m = max(y);
             s = 401;
             while y(s) > 0.95 & y(s) < 1.05;
                 s = s-1;
             end;
             ts = (s-1)*0.01;
             s = 1;
             while y(s) < 0.95;
                 s = s+1;
             end;
             tsub = (s-1)*0.01;

             if m<1.14 & m>1.02 & ts<1.0 & tsub < 0.5;

                 Gc(n) = k*tf([1 alfamasbeta alfaporbeta],[1 0]);
                 table(n,:) = [n a(i) b(j) c(h) m ts tsub k alfamasbeta
                               alfaporbeta];
                 n = n+1;
             end
         end
     end
end
```