UNIVERSIDAD NACIONAL DE CORDOBA

Fecha

04/12/2015

Integrantes

Brignone, Matías Nicolás. 38503382  
Rodriguez, Lucia Fernanda. 38203622

Monografia de sistemas de control I

Control de posición de una pelota sobre una vara

Tabla de contenido

[**OBJETIVOS** 2](#_Toc437453225)

[**MODELO MATEMÁTICO** 3](#_Toc437453226)

[**RESPUESTA TRANSITORIA** 7](#_Toc437453227)

[**ANÁLISIS DE ESTABILIDAD** 9](#_Toc437453228)

[**Criterio de Routh-Hurwitz** 9](#_Toc437453229)

[**Método del Lugar de Raíces** 9](#_Toc437453230)

[**COMPENSACIÓN** 11](#_Toc437453231)

[**Compensación por Lugar de Raíces** 12](#_Toc437453232)

[**Compensación por Variables de Estado** 14](#_Toc437453233)

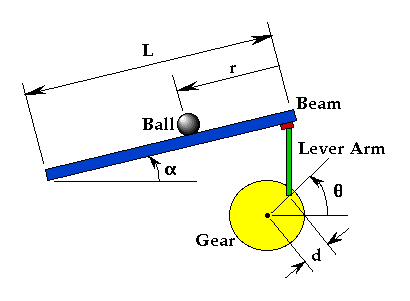
[**CONCLUSIÓN** 20](#_Toc437453234)

[**ANEXOS** 21](#_Toc437453235)

# **OBJETIVOS**

El objetivo de la presente monografía es aplicar los conocimientos aprendidos en la materia *Sistemas de Control I* a un problema práctico concreto.

En nuestro caso, lo que se hará es controlar la posición de una pelota sobre una palanca que se encuentra fija en un extremo y en el otro extremo tiene una leva unida a un servomotor, el cual en función del sentido y del ángulo que gire, inclinará la palanca hacia arriba o hacia abajo, provocando un movimiento de la pelota por acción de la gravedad. A continuación se presenta un diagrama del sistema propuesto:



Figura

En concordancia con el diagrama, lo que se hará es controlar la variable “r”en función de “θ”.

# **MODELO MATEMÁTICO**

En la *Figura 1*, las variables en juego son las siguientes:

α = ángulo de la palanca.

θ = ángulo del servomotor.

r = coordenada de la pelota a lo largo de la palanca.

L = longitud de la palanca.

d = distancia leva-eje servomotor.

m = masa de la pelota.

R = radio de la pelota.

J = momento de inercia de la pelota

Para nuestro caso elegimos los siguientes valores:

m = 0.09 Kg

R = 0.01 m

d = 0.04 m

g = 9.8 m/s^2

L = 0.1 m

A continuación se presenta el diagrama de cuerpo libre de nuestro sistema, siendo la coordenada *x* equivalente a la distancia *r*.

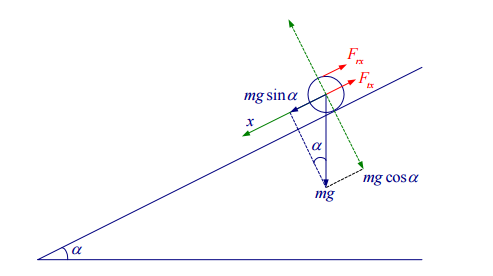


Figura 2

Si despreciamos las fuerzas de rozamiento, las únicas 2 fuerzas que quedan en juego son la fuerza debida al movimiento traslacional de la pelota ( ) y al movimiento rotacional () de la misma.

De acuerdo a las leyes de Newton, se tiene que la fuerza debido al movimiento traslacional es:

Siendo w la velocidad angular de la pelota, el torque debido al movimiento rotacional está dado por:

Siendo el momento de inercia de una esfera , se obtiene finalmente:

Realizando la sumatoria de fuerzas a lo largo del eje x:

Despejando nuestra variable de interés obtenemos:

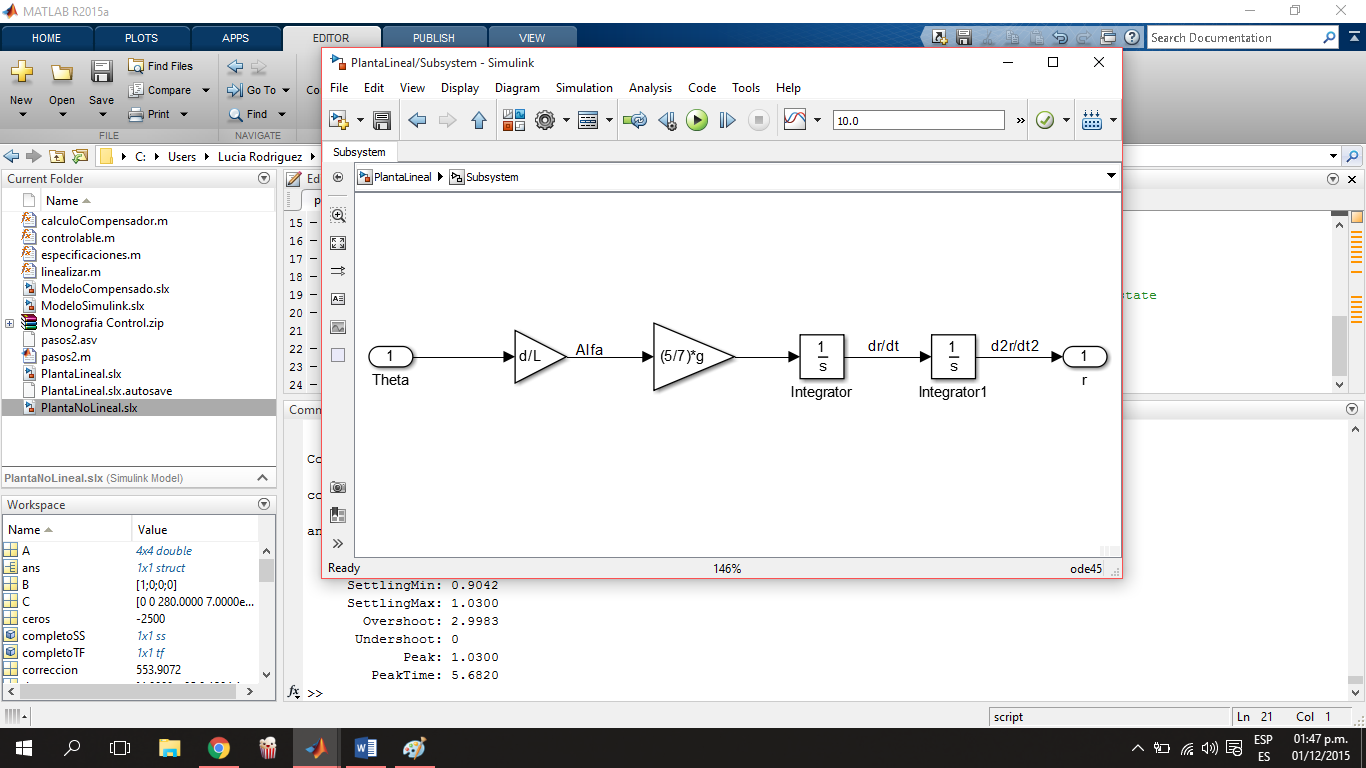
Debido a la presencia del seno del ángulo, la ecuación es no lineal. Pero si se considera que el mencionado ángulo no variará demasiado en torno a nuestro punto de trabajo, podemos considerar, trabajando en radianes, que sin = , resultando la siguiente ecuación:

Pero dado que lo que se controlará es el ángulo θ del motor, es necesario relacionarlo con mediante la siguiente ecuación:

Reemplazando dicho valor obtenemos:

Para determinar la función de transferencia del sistema, es necesario realizar la transformada de Laplace de la ecuación anterior y luego despejar el cociente salida/entrada:

Dicho modelo fue implementado en SIMULINK para luego incorporarlo a los demás bloques que completan el sistema.



El ángulo será controlado a través de un servomotor que determina la posición mediante modulación por ancho de pulso, enviando señales que se traducirán en un valor medio de continua máximo de 5V, teniendo una inclinación máxima de 60° y una constante de tiempo de 0.12s, su función de transferencia será:

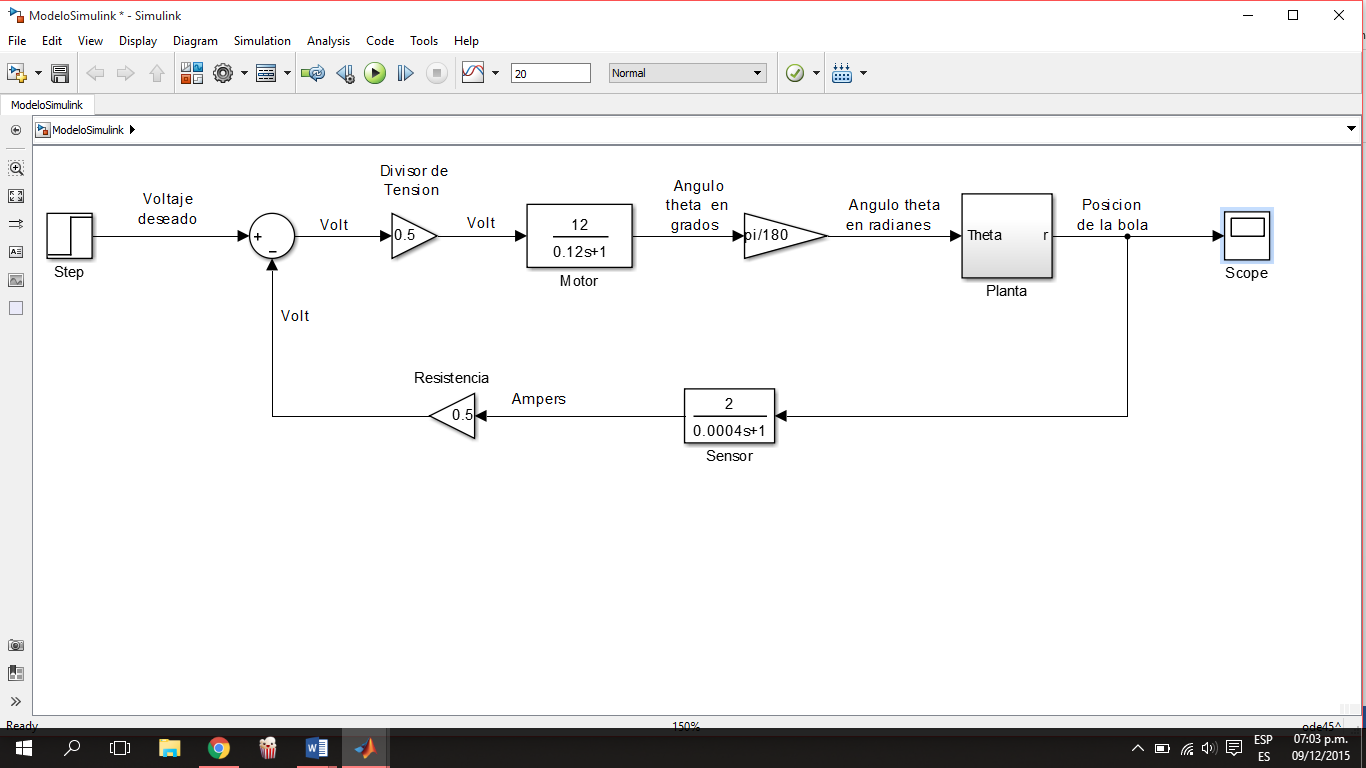
A su vez, se necesitará colocar un sensor de distancia en uno de los extremos para conocer la ubicación de la pelota, el cual tendrá una distancia en su entrada y entregará a su salida un valor de corriente. La distancia máxima que mide son 10cm y entrega una corriente máxima de 20mA, con una constante de tiempo de 0.4ms (se necesitaba un sensor lo suficientemente veloz para poder realimentar y actualizar la posición lo más rápido posible). Así, la función de transferencia resultante para el sensor es:

Este valor de corriente debe ser transformado en un valor de voltaje. En la práctica se colocaría una resistencia y de acuerdo a la ley de Ohm se tendría V = IR, por lo que simplemente se debe colocar una ganancia para acondicionar los valores de voltaje que se compararán. Dicha ganancia tendrá un valor de 0.5, para que la máxima tensión entregada sean 10V (20mA \* 0.5kΩ), que coincide con la entrada correspondiente a la máxima distancia deseada.

Cabe destacar que si se desea que la pelota esté ubicada en la posición r = 1cm, esto se traduce en una tensión de 1V, mientras que r = 10cm se traduce en una tensión de 10V, por ello no es necesario colocar un bloque que ajuste los valores de distancia en cm a tensión en voltios, ya que sería una ganancia unitaria.

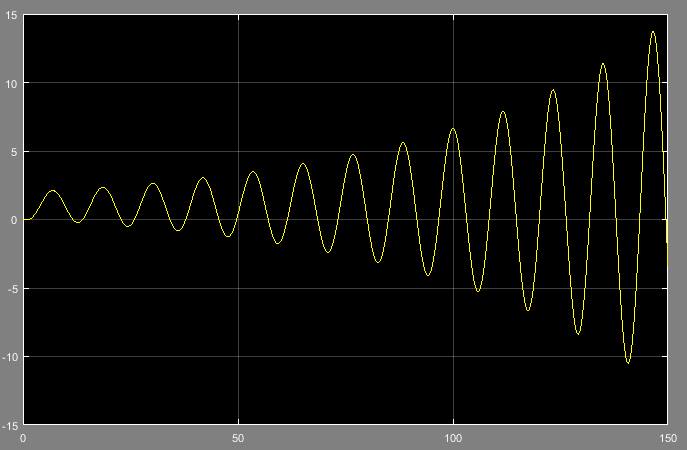
A su vez, se debe tener en cuenta que la salida del bloque motor se encuentra en grados, mientras que el modelo matemático de la planta fue linealizado considerando una entrada en radianes, por lo que es necesario agregar una ganancia de que realice la conversión de grados a radianes. Además, el máximo voltaje que puede soportar el servomotor es de 5V, y le pueden llegar hasta 10V por lo que se deberá colocar previamente una ganancia de 0.5 que ajuste los valores.

Finalmente, al implementar el sistema completo en SIMULINK, se obtiene el siguiente esquema:



# **RESPUESTA TRANSITORIA**

Se analizará la respuesta del sistema ante una entrada de tipo escalón, ya que ésta será la entrada típica del sistema. El valor final del escalón representa la posición en la que se desea que quede la pelota.



Como se puede apreciar, el sistema es inestable, ya que comienza a oscilar tomando valores cada vez mayores, por lo que será necesario el uso de algún tipo de compensador.

Mediante MATLAB calculamos la función de transferencia global de nuestro sistema, mediante el siguiente comando:

*>> completoTF = feedback (0.5\*sistema\*motor\*pi/180, sensor\*0.5)*

Obteniendo lo siguiente:

**Errores y Tipo de Sistema**

A continuación se procede a determinar los errores en régimen permanente de nuestro sistema para diferentes entradas (escalón, rampa y parábola), a partir de lo cual se podrá determinar de qué tipo es el sistema.

Para una entrada R(s), siendo *q* un factor que agrupa todas las ganancias, el error en régimen permanente de nuestro sistema estará dado por:

Entrada Escalón ()

Entrada Rampa ()

Entrada Parábola ()

Esto deja en evidencia que el sistema en cuestión es de ***tipo 2***.

# **ANÁLISIS DE ESTABILIDAD**

Se determinará si el sistema es estable mediante dos métodos diferentes que deberán arrojar los mismos resultados: el *criterio de Routh-Hurwitz* y el *método del lugar de raíces*.

## **Criterio de Routh-Hurwitz**

Siendo la función de transferencia a lazo cerrado la siguiente:

La ecuación característica resultante es:

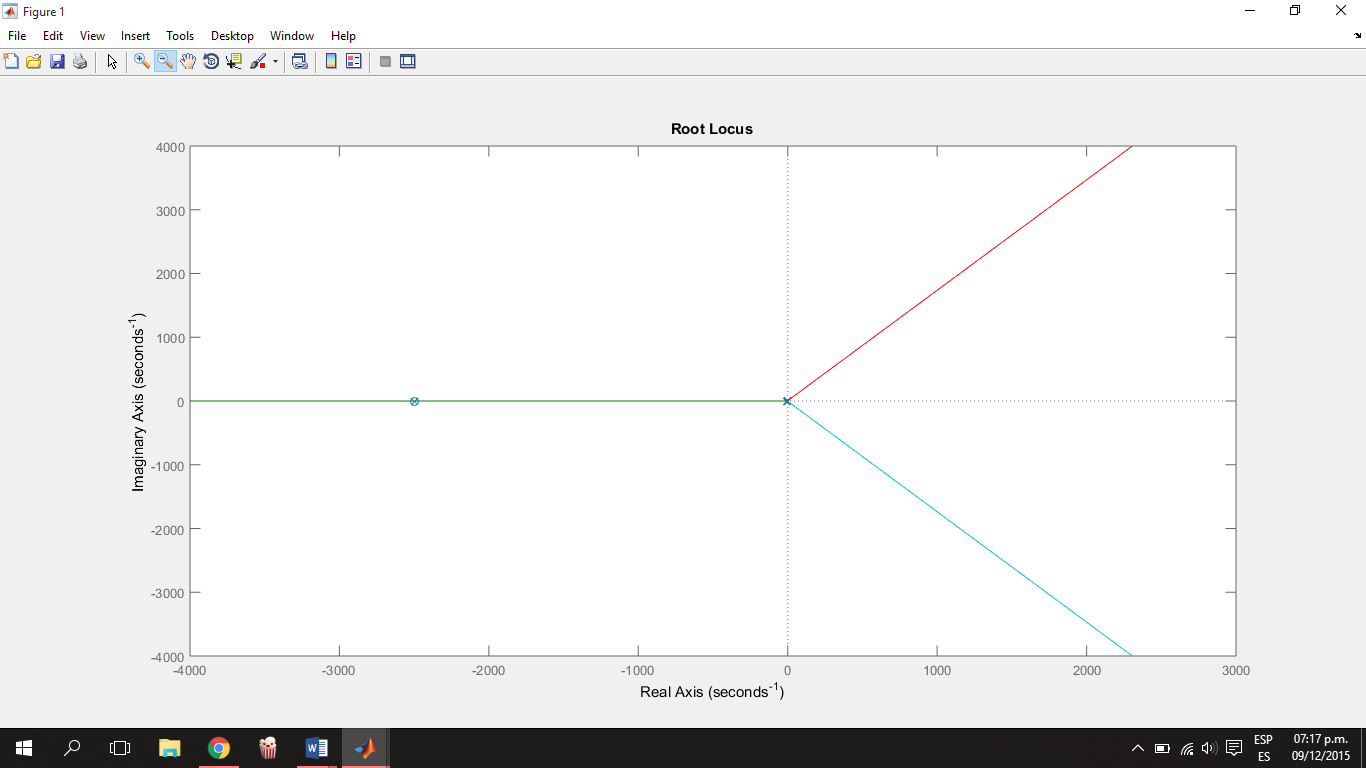
Con ella se trabajará en el criterio de Routh-Hurwitz, el cual nos permite determinar si el sistema es estable, es decir, si todas las raíces de la ecuación característica tienen parte real negativa, sin necesidad de resolver el polinomio.

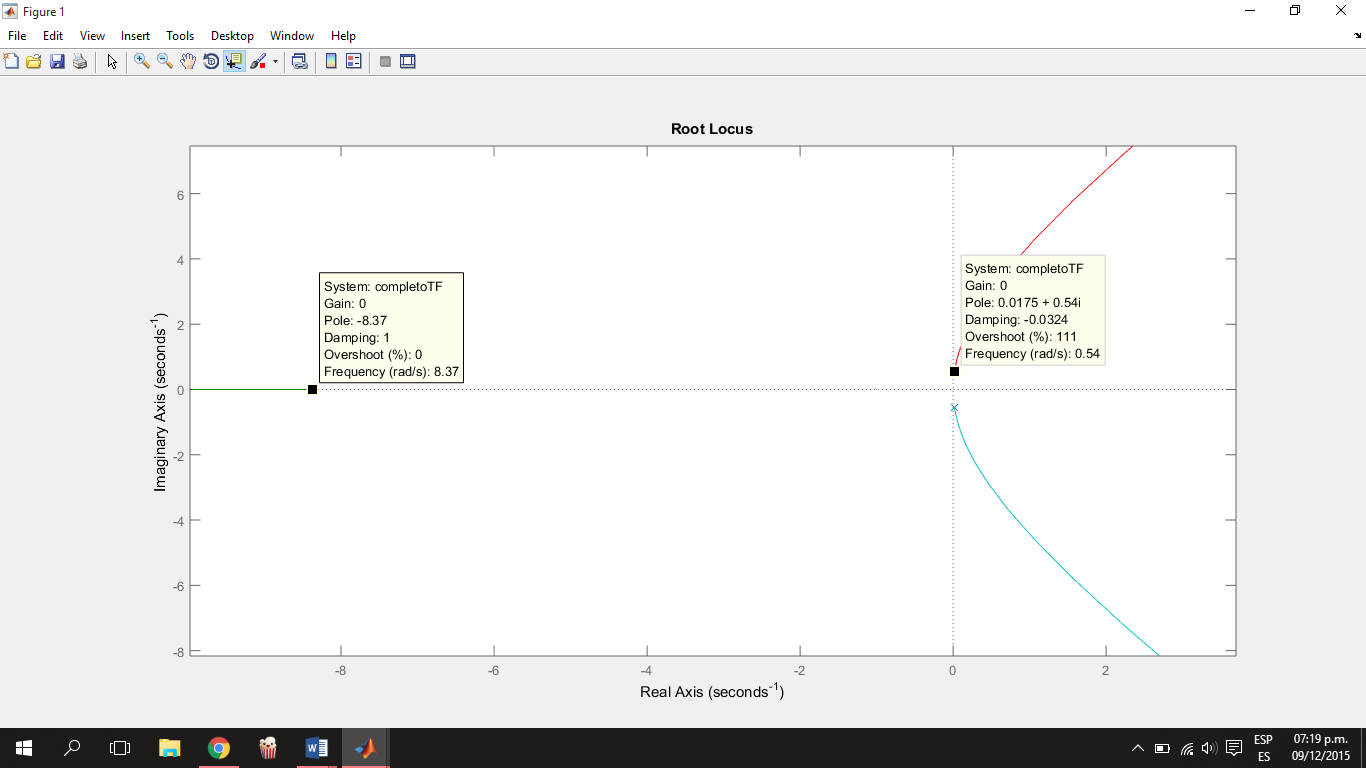
|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **s4** | 0.00864 | 180 | 52.78 |
| **s3** | 21.67 | 0 | 0 |
| **s2** | 180 | 52.78 | 0 |
| **s1** | -6.35 | 0 | 0 |
| **s0** | 52.78 | 0 | 0 |

Como se puede apreciar, ha habido un doble cambio de signo, lo cual indica que el sistema posee 2 polos con parte real positiva y por lo tanto el sistema es siempre inestable. Esto mismo podrá verificarse con el método del lugar de raíces que se detalla a continuación.

## **Método del Lugar de Raíces**

Este método consiste en graficar en el plano *s* los polos y ceros del sistema para diferentes valores de ganancia *k*. Para hacerlo, nos valdremos del comando *rlocus* de MATLAB, el cual justamente grafica el lugar de raíces del sistema que se le pase como argumento. Así, se obtuvo la siguiente representación:





Como se puede ver, para una ganancia *0* el sistema tiene un polo y un cero en -2500 (primera figura) que se cancelan mutuamente, un polo en -8.37 que se corre hacia - a medida que aumenta la ganancia, y 2 polos complejos conjugados en 0.02 ± 0.54i que se ubican cada vez más a la derecha a medida que aumenta la ganancia, por lo que el sistema será siempre inestable.

# **COMPENSACIÓN**

La necesidad de un compensador queda en evidencia al observar que el sistema es fuertemente inestable, por lo que ese será el principal objetivo del compensador. Además, se buscará satisfacer los siguientes criterios de diseño:

* Tiempo de Asentamiento (al ±5% del valor final) = 4 segundos
* Sobrepaso = 3%

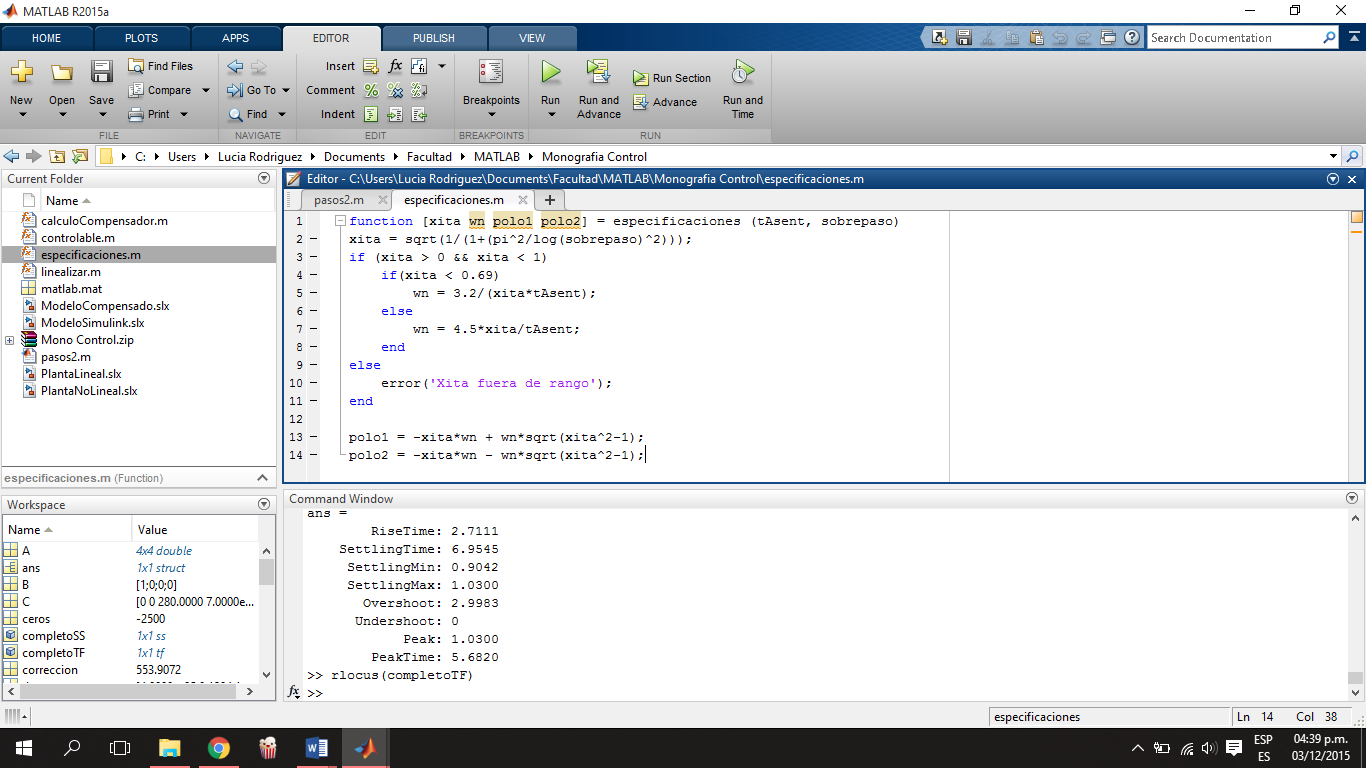
El tiempo que tarda la pelota en llegar a la posición deseada no es un factor crítico en nuestro sistema, por lo que un tiempo de asentamiento de 4 segundos resulta un valor sumamente razonable.

Con respecto al sobrepaso, este no deberá ser demasiado alto para evitar oscilaciones excesivas en la palanca, pudiendo provocar que la pelota se desprenda de ella o que se vaya más allá de los límites de la misma, por lo que se eligió un valor relativamente bajo de 3%.

A partir de estas especificaciones es posible determinar el valor de ξ (xita) y :

A partir de estos valores, se determina la ubicación de los polos en:

Todo este proceso se automatizó en MATLAB creando la función *‘especificaciones’* que se muestra a continuación:

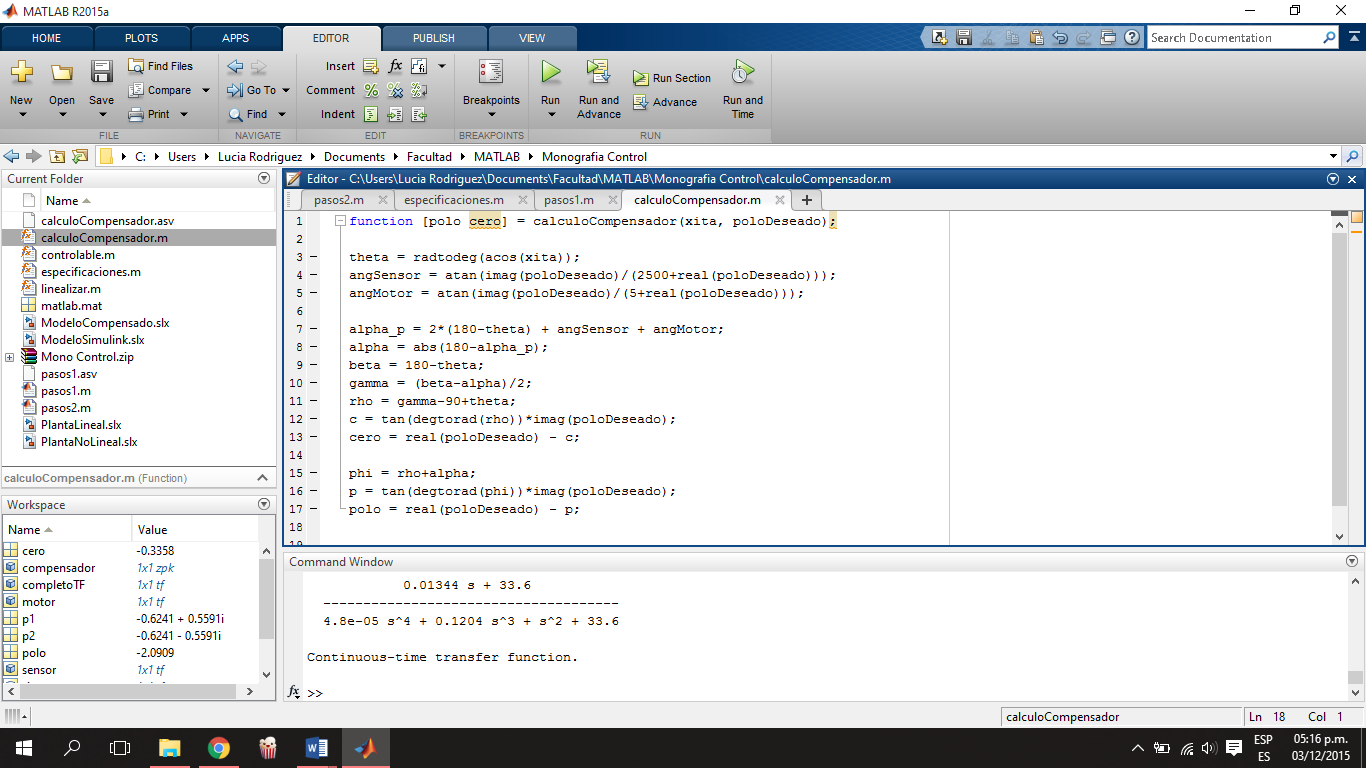


Una vez determinado el polo deseado, se intentó realizar una compensación por lugar de raíces.

## **Compensación por Lugar de Raíces**

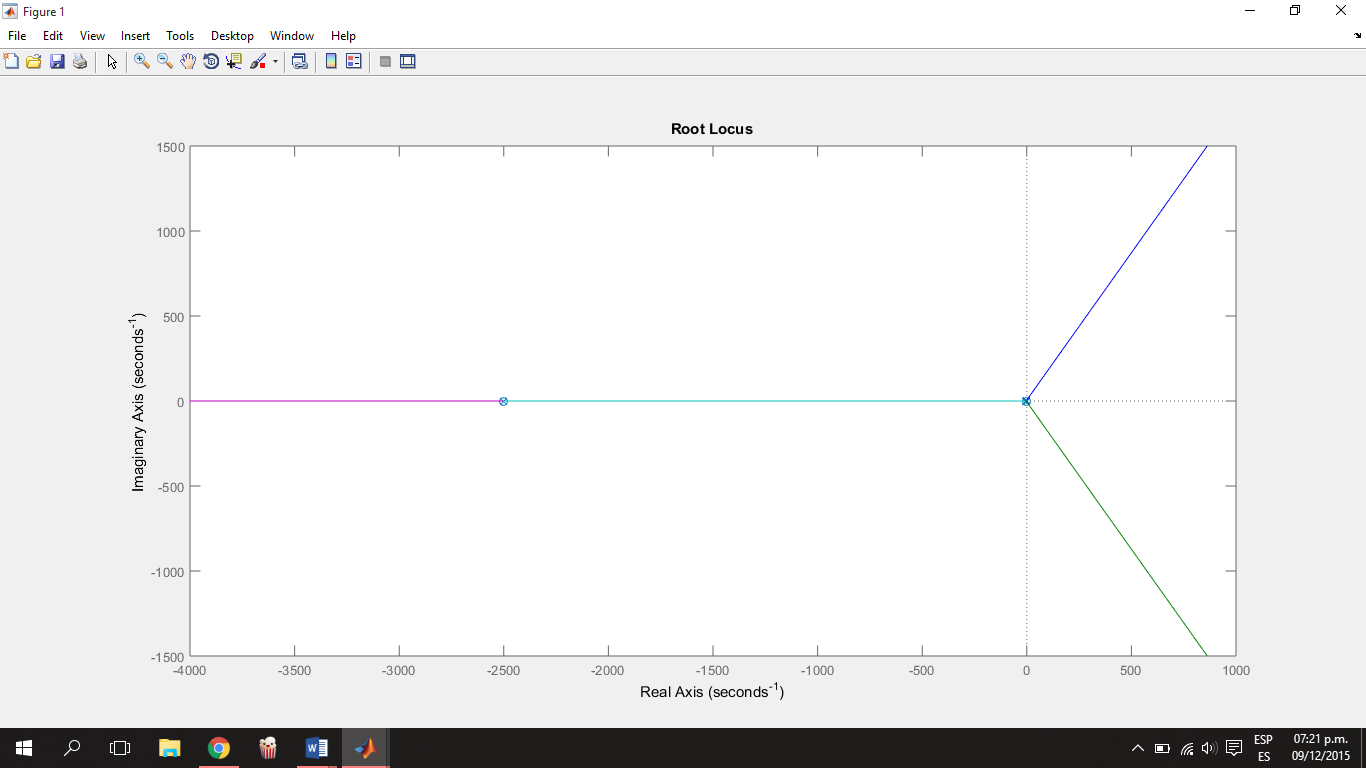
Este método consiste en agregar un compensador C de la forma:

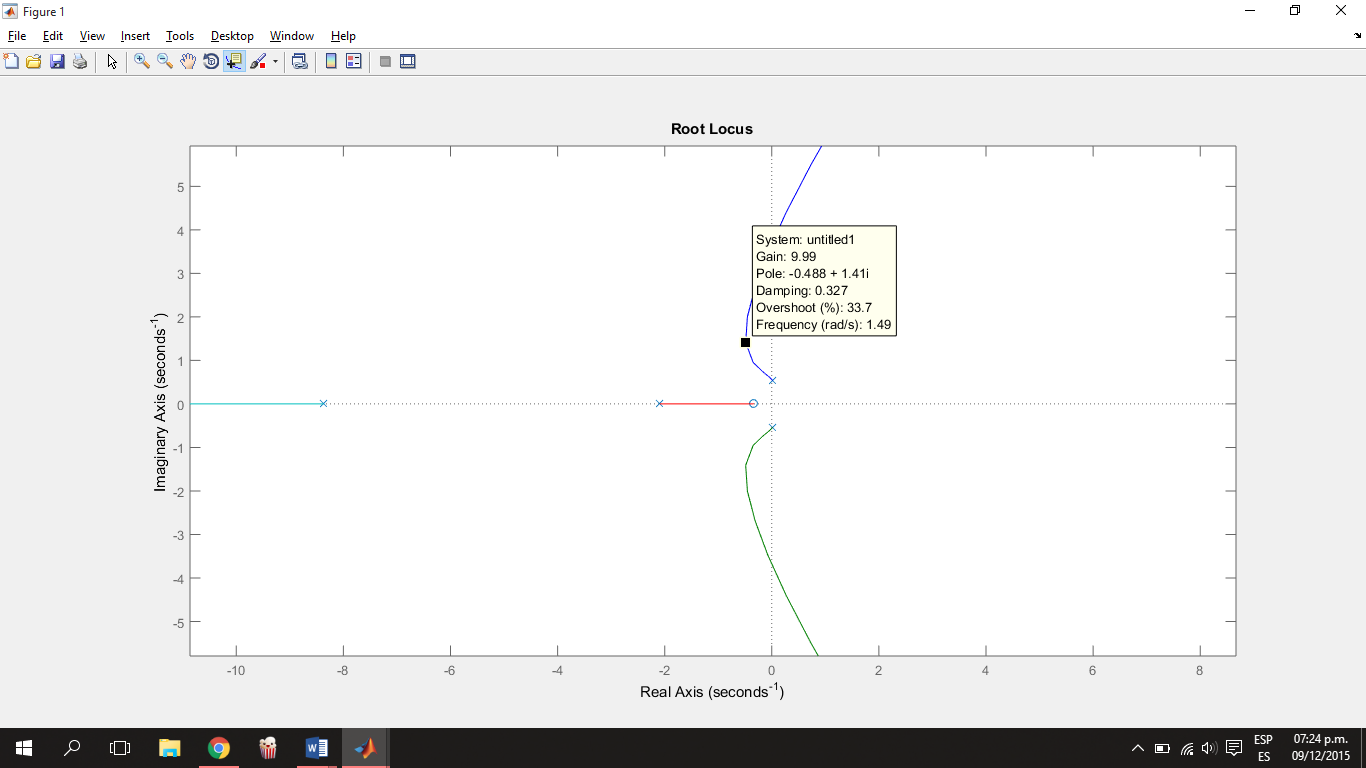
Aplicando el método de la bisectriz visto en clase, se calcula el ángulo que debe contribuir el compensador para obtener los polos deseados, y a partir del mismo se determina la ubicación del polo y el cero del compensador. Dicho proceso de automatizó en la función “cálculoCompensador” que se muestra a continuación:



Al ejecutarla con los polos deseados calculados anteriormente, se obtuvieron los siguientes polos y ceros para el compensador:

Para determinar la ganancia, el camino más sencillo es graficar nuevamente el lugar de raíces del sistema con el compensador y determinar el valor de *k* para el cual se tienen los polos deseados. Desafortunadamente, al graficar el nuevo lugar de raíces, se obtuvo lo siguiente:



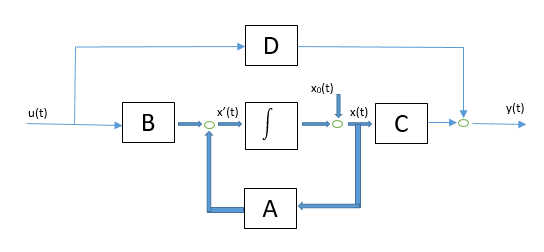


Como se puede ver, si bien se logra estabilizar el sistema para algunos valores de , no existe ningún valor que logre que el lugar de raíces pase por los polos deseados, por lo que el método elegido no funciona para la compensación que se desea realizar. Por esto es que se decidió compensar trabajando con variables de estado.

Para realizar esta compensación por lugar de raíces, se detallaron los pasos en un script con el nombre *“pasos1”*, que puede encontrarse en los anexos.

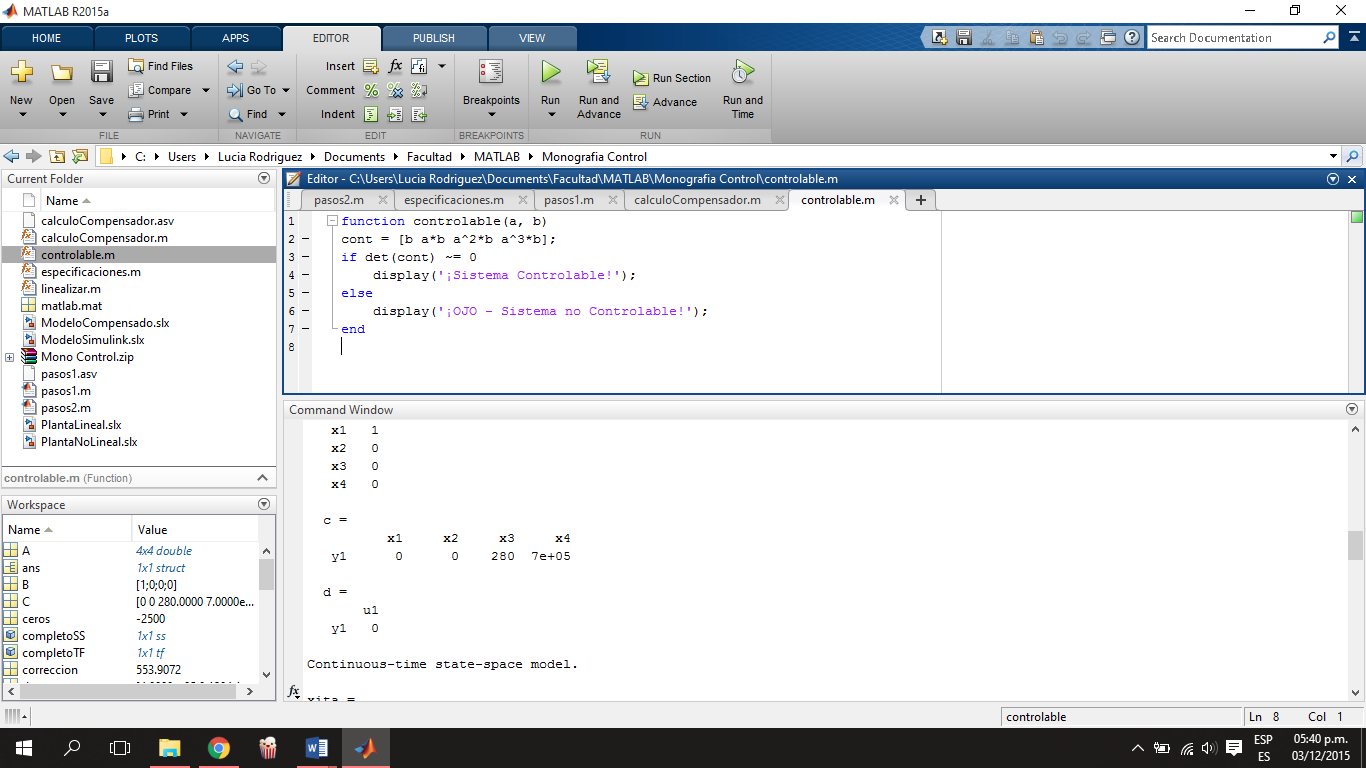
## **Compensación por Variables de Estado**

Debido a todo esto elegimos compensar el sistema utilizando el método por variables de estado, para lo cual fue necesario llevar nuestra función de transferencia al espacio de estados tal como se muestra en el siguiente esquema:



Para obtener las matrices *A, B, C* y *D* nos valdremos del comando *“tf2ss”* de Matlab llegando a los siguientes valores:

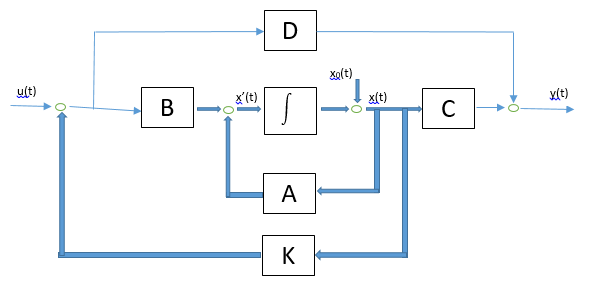
Luego verificamos que el sistema sea controlable, para garantizar que los polos puedan ser ubicados en cualquier lugar que se desee, mediante la siguiente función *“controlable”* que implementa el método de *Kalman* para nuestro sistema:



De no cumplirse la condición de controlabilidad, se intentaría de todas formas ubicar los polos en el lugar deseado pero sin garantizar que sea posible.

Dado que nuestras especificaciones determinan la ubicación de solo 2 polos y se necesitan 4 para implementar el método, fijamos los 2 polos restantes en valores negativos lo suficientemente alejados del origen para que no influyan en la respuesta transitoria y se cumplan nuestras condiciones aun con su presencia. Se eligieron los siguientes polos: -30 y -60.

Luego mediante el comando *“place”*, se determinó el vector K, que se realimentara de acuerdo al siguiente esquema:

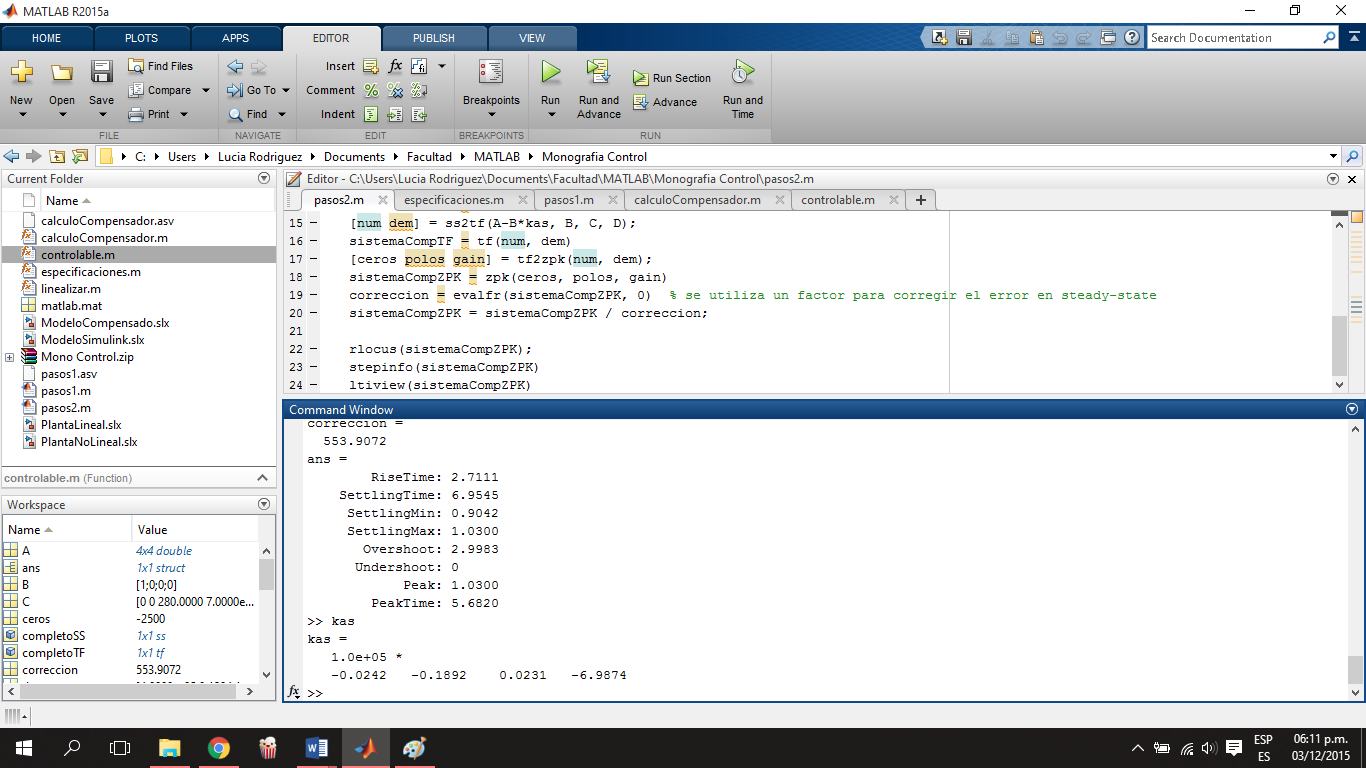


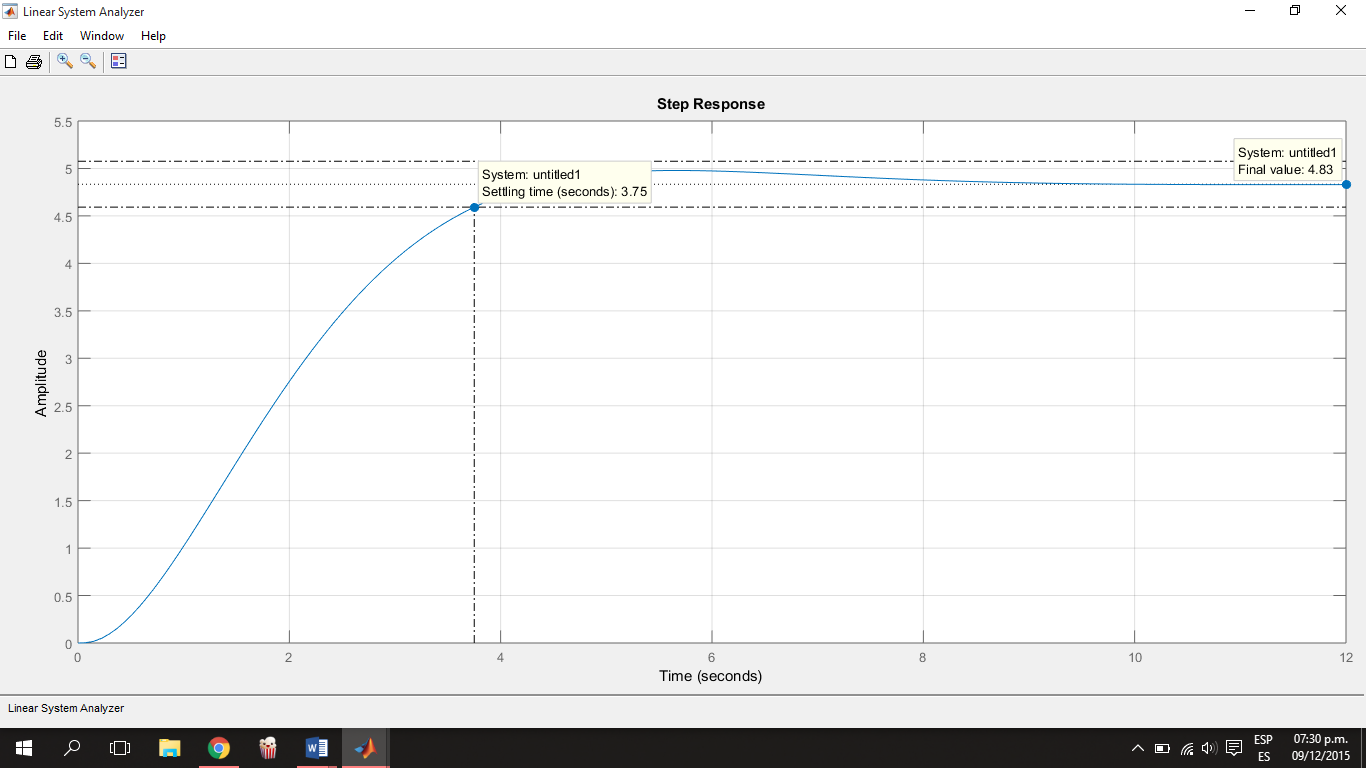
Los valores del vector K son los siguientes:

Para obtener la función final ya compensada, es necesario modificar la matriz *A* utilizando el vector anterior de la siguiente manera:

Ahora utilizamos esta nueva matriz para generar el sistema compensado a través de la función *“ss2tf”* obteniendo lo siguiente:

Se puede ver que los polos están ubicados en la posición deseada. Esto cumple con el tiempo de asentamiento y el sobrepaso deseado como se puede observar en la información de la respuesta al escalón unitario:

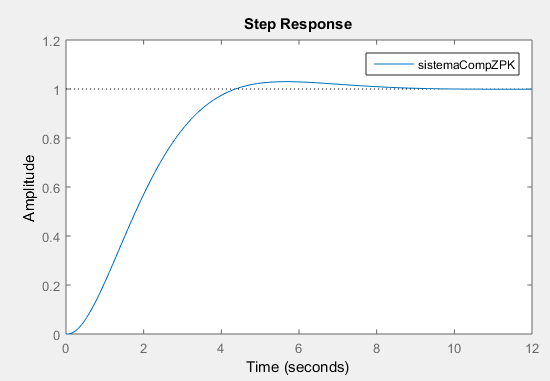


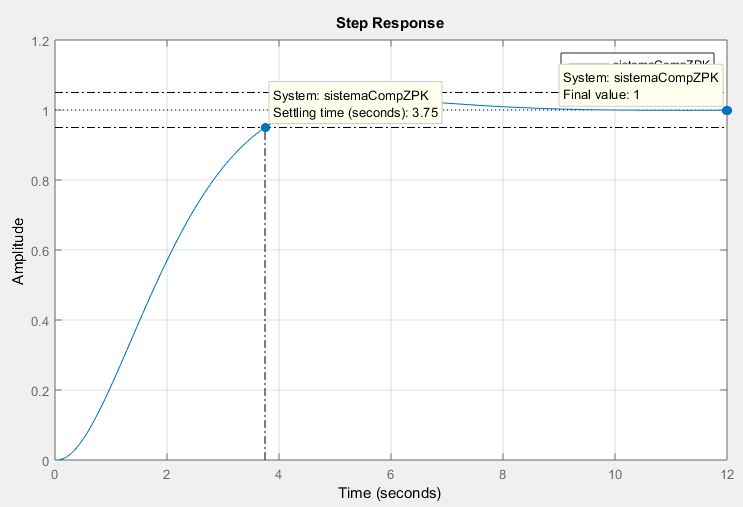
Téngase en cuenta que el tiempo de asentamiento especificado por Matlab se encuentra a un ±2% del valor final. Mediante un simple ajuste se puede comprobar en MATLAB que considerándolo al ±5% se obtiene el valor deseado. Pero con esta función se tiene un indeseado error en estado estable. Por lo tanto es necesario realizar una corrección dividiendo por un determinado factor dado por la función de transferencia valuada en cero. Esto se logró utilizando el siguiente comando:

*>> correccion = evalfr(sistemaCompZPK, 0)*

En este caso particular, esta corrección tiene un valor de 4.83 aproximadamente. Realizando la división mencionada se obtiene la función final:

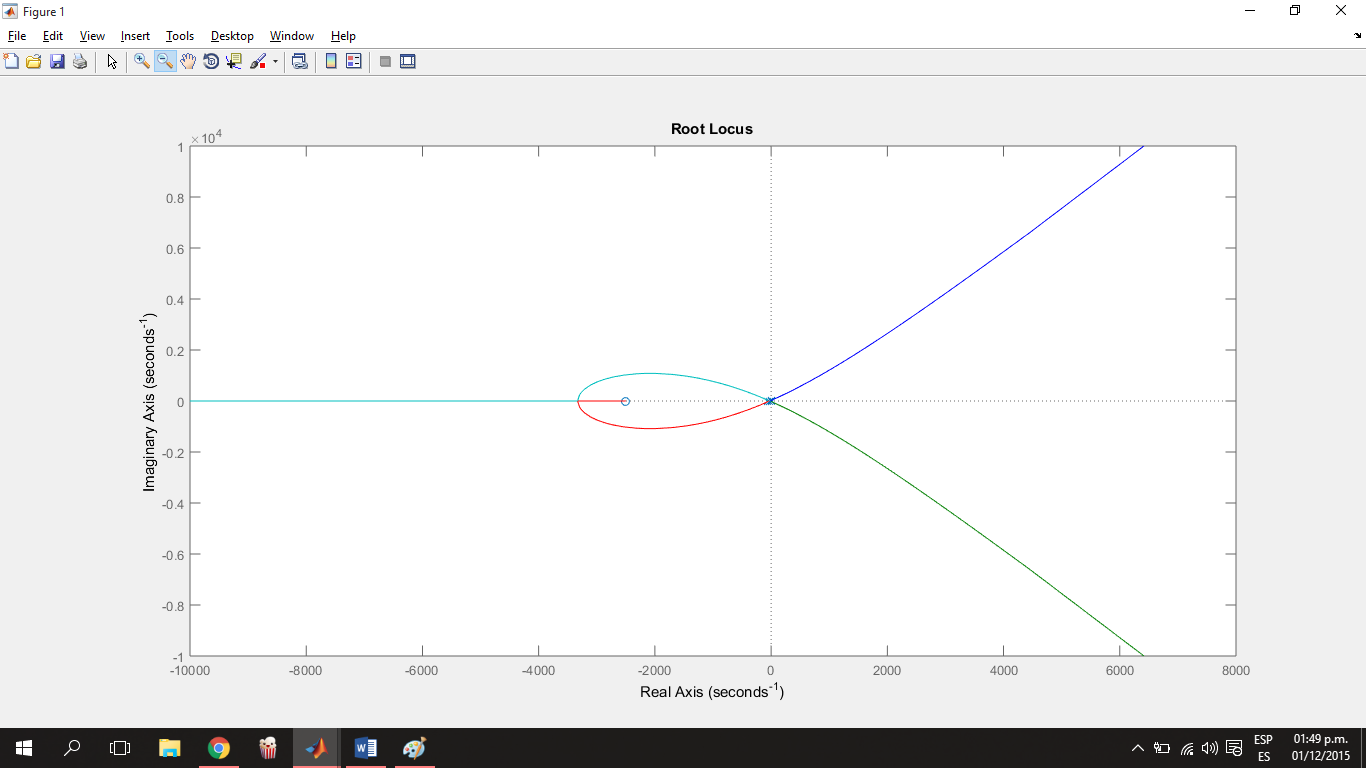
Ahora procedemos a mostrar la respuesta al escalón unitario de la función:

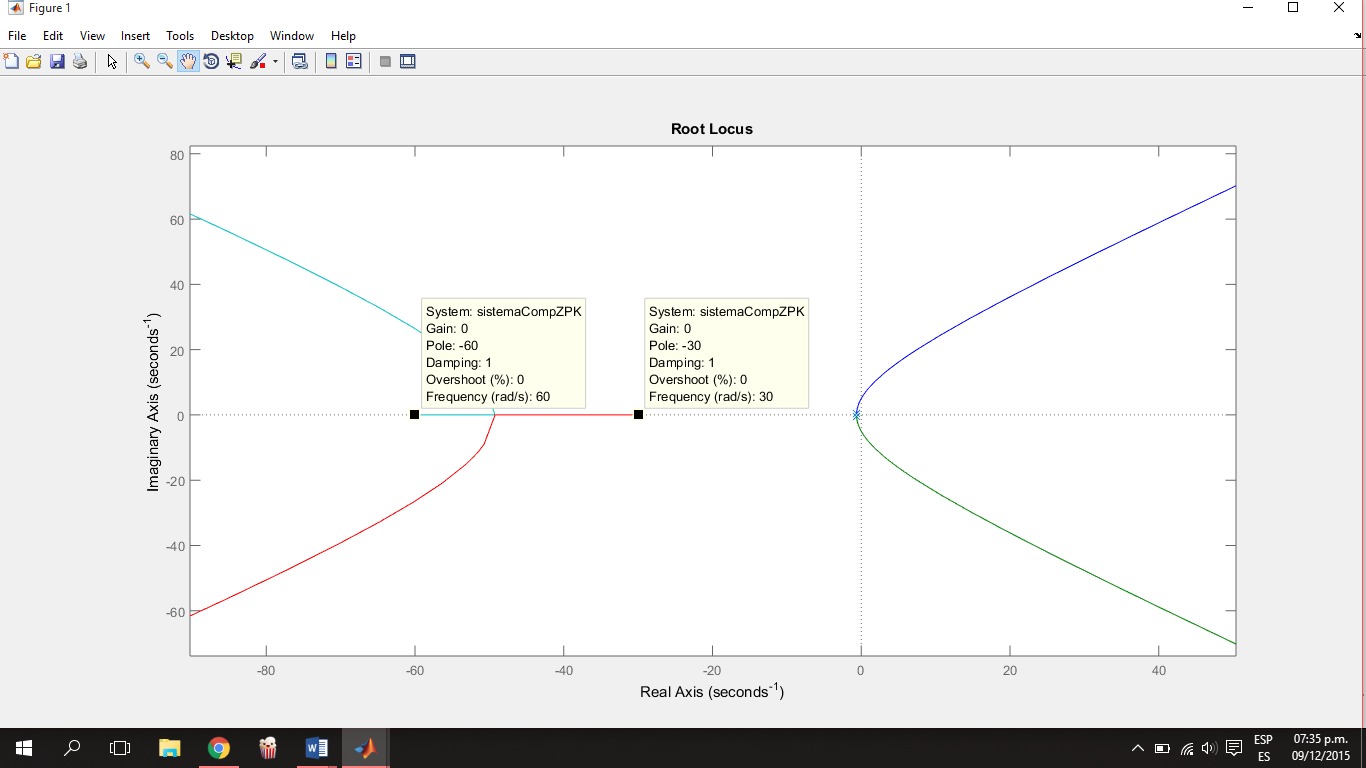


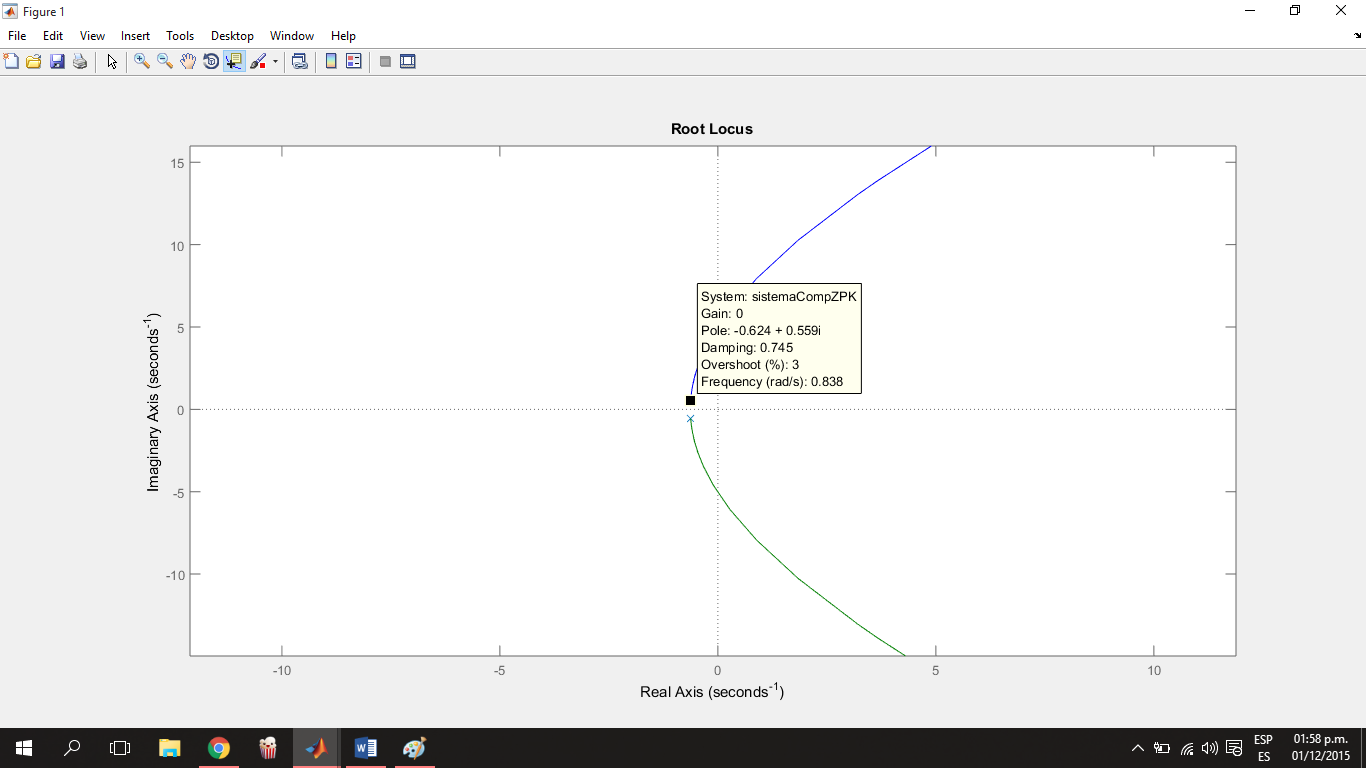


Como podemos ver el tiempo de asentamiento (al ±5%) es, como habíamos propuesto, aproximadamente igual a los 4 segundos. Además se alcanza el valor final deseado que es el 1 que se envió como entrada. De esta forma, como ya se había explicado antes, siendo equivalentes la tensión en volts a la posición deseada en centímetros, a la salida se obtiene la ubicación solicitada.

A continuación se muestra el lugar de raíces de la función final, ya compensada y estable:







Así se puede ver que el lugar de raíces es el esperado, con los polos en sus correspondientes ubicaciones. Además, en este caso si existen valores de kc tal que el sistema sea estable, manteniendo todos sus polos del lado izquierdo del origen.

Todos los pasos llevados a cabo para esta compensación se encuentran especificados en un script llamado *“pasos2”* el cual se encuentra detallado en los anexos.

# **CONCLUSIÓN**

De esta forma, aplicando los conocimientos adquiridos a lo largo de toda la materia *Sistemas de Control I*, hemos sido capaces de resolver un problema práctico real, el cual podría ser aprovechado para solucionar situaciones más complejas partiendo de esta base.

Se pudo comprobar como el método de compensación por lugar de raíces, no siempre es efectivo aun cuando los cálculos aparentemente digan lo contrario. Sin embargo, aplicando el método de compensación por variables de estado si fue posible realizar una resolución adecuada. En la práctica este es el método más utilizado ya que permite un mayor control en sistemas más complejos como en este caso.

También se pudo verificar que los métodos teóricos aprendidos, como el de Routh-Hurwitz para determinar la estabilidad, y el método de Kalman para definir la controlabilidad de un sistema son coherentes con los resultados obtenidos.

Finalmente, queremos destacar que el trabajo sirvió no solo para integrar lo visto a lo largo del semestre sino también para profundizar los contenidos y llevarlo a un campo menos abstracto y más realista.

# **ANEXOS**

**Resolución del sistema con lugar de raíces (pasos1):**

sistema = tf(2.8, [1 0 0])

motor = tf(12, [0.12 1])

sensor = tf(2, [0.0004 1])

[xita wn p1 p2] = especificaciones(4, 0.03)

[polo cero] = calculoCompensador(xita, p1)

compensador = zpk(cero, polo, 1)

completoTF = feedback(0.5\*sistema\*motor\*pi/180, sensor\*0.5)

rlocus(completoTF\*compensador);

**Resolución del sistema con variables de estado (pasos2):**

sistema = tf(2.8, [1 0 0])

motor = tf(12, [0.12 1])

sensor = tf(2, [0.0004 1])

completoTF = feedback(0.5\*sistema\*motor\*pi/180, sensor\*0.5) % armo el diagrama de 'ModeloSimulink'

[n d]= tfdata(completoTF, 'v');

[A B C D]= tf2ss(n, d); % los valores corresponden al numerador y denominador de la variable completoTFpas

completoSS = ss(A, B, C, D)

[xita wn p1 p2] = especificaciones(4, 0.03) % determino polos de acuerdo al t de asentamiento (5%) y al sobrepaso

poloFijo1 = -30; poloFijo2 = -60; % fijamos polos lejos para que no afecten

controlable(A, B); % verificamos que sea controlable

kas = place(A, B, [p1 p2 poloFijo1 poloFijo2]);% determinamos el valor del vector K

sistemaCompSS = ss(A-B\*kas, B, C, D) % creamos el sistema compensado

[num dem] = ss2tf(A-B\*kas, B, C, D);

sistemaCompTF = tf(num, dem)

[ceros polos gain] = tf2zpk(num, dem);

sistemaCompZPK = zpk(ceros, polos, gain)

correccion = evalfr(sistemaCompZPK, 0) % se utiliza un factor para corregir el error en steady-state

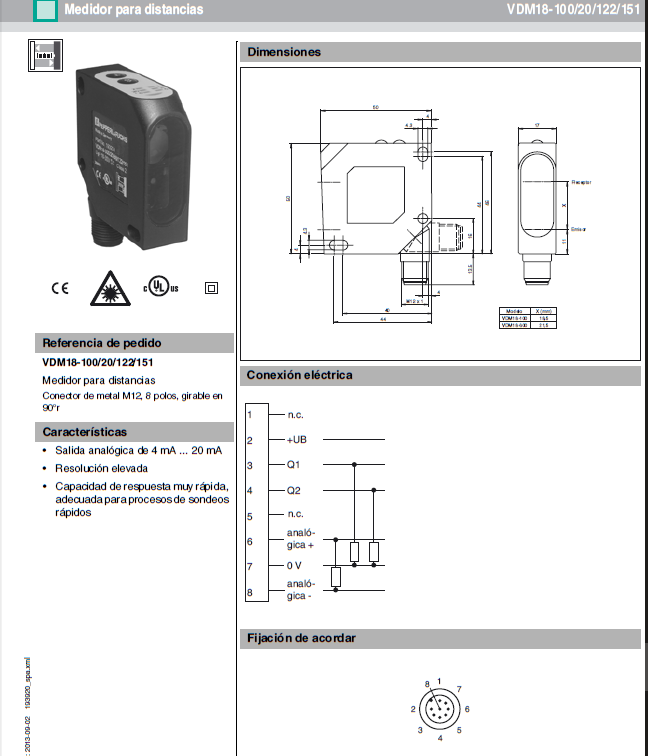
sistemaCompZPK = sistemaCompZPK / correccion

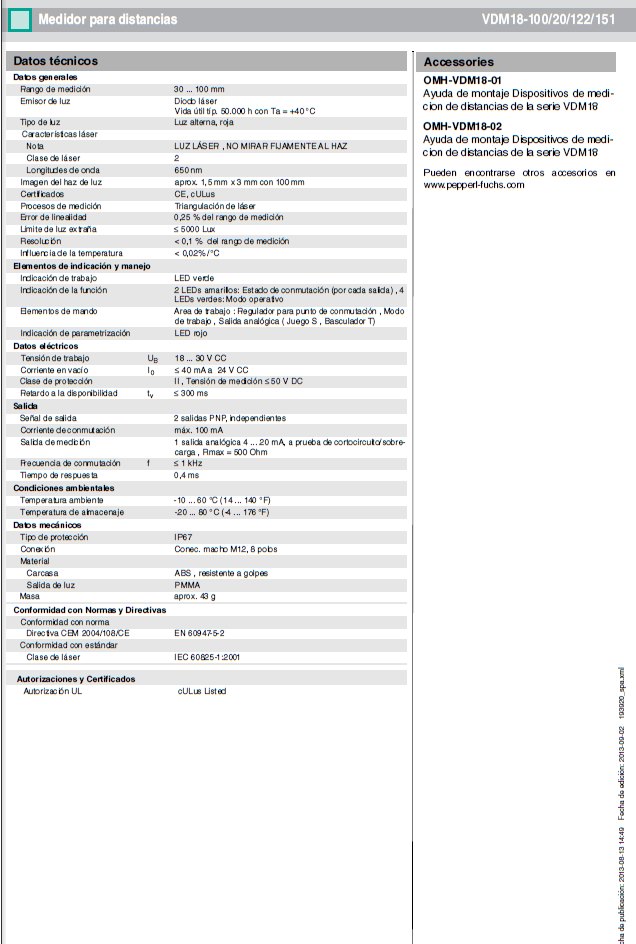
rlocus(sistemaCompZPK);

stepinfo(sistemaCompZPK)

ltiview(sistemaCompZPK)

**Datasheet Sensor de Distancia:**



****

**Datasheet Servomotor:**

**MG996R High Torque**

**Metal Gear Dual Ball Bearing Servo**

This High-Torque MG996R Digital Servo features metal gearing resulting in extra high 10kg

stalling torque in a tiny package. The MG996R is essentially an upgraded version of the

famous MG995 servo, and features upgraded shock-proofing and a redesigned PCB and IC

control system that make it much more accurate than its predecessor. The gearing and motor

have also been upgraded to improve dead bandwith and centering. The unit comes complete

with 30cm wire and 3 pin 'S' type female header connector that fits most receivers, including

Futaba, JR, GWS, Cirrus, Blue Bird, Blue Arrow, Corona, Berg, Spektrum and Hitec.

This high-torque standard servo can rotate approximately 120 degrees (60 in each direction).

You can use any servo code, hardware or library to control these servos, so it's great for

beginners who want to make stuff move without building a motor controller with feedback &

gear box, especially since it will fit in small places. The MG996R Metal Gear Servo also

comes with a selection of arms and hardware to get you set up nice and fast!

**Specifications**

Weight: 55 g

Dimension: 40.7 x 19.7 x 42.9 mm approx.

Stall torque: 9.4 kgf·cm (5 V ), 11 kgf·cm (6 V)

Operating speed: 0.12 s/60º (5 V), 0.1 s/60º (6 V)

Operating voltage: 4.8 V a 7.2 V

Running Current 500 mA –

Stall Current 2.5 A (6V)

Dead band width: 5 μs

Stable and shock proof double ball bearing design

Temperature range: 0 ºC –

900 mA (6V)

55 ºC