**                                          **

**LABORATORIO DE INGENIERÍA DE CONTROL**

**CURSO 2018/19**

**PRÁCTICA Nº: 2**

**Título: CONTROLADORES PID- ANÁLISIS Y DISEÑO.**

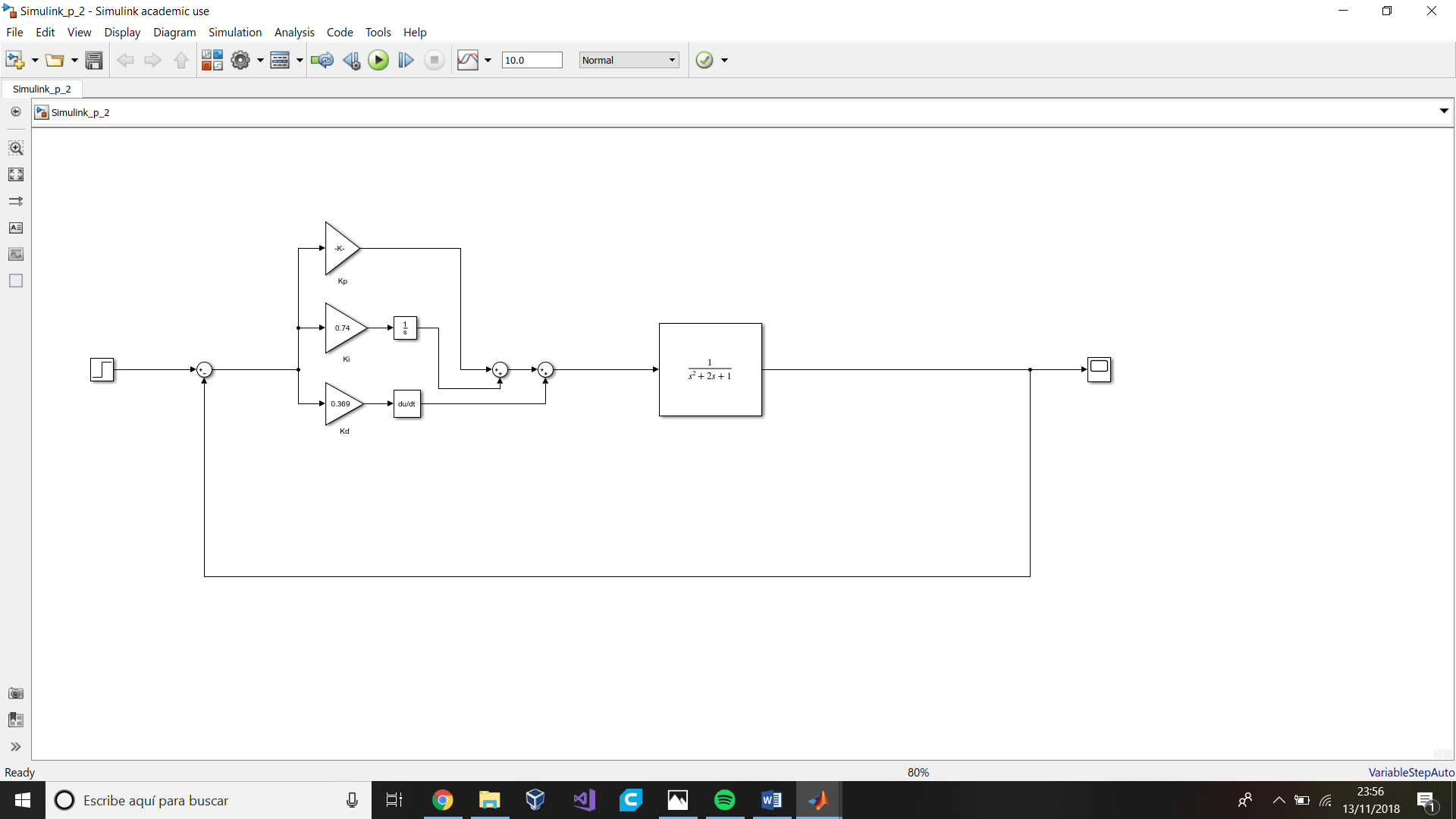
**APELLIDOS, NOMBRE: Martínez Trapiello, Alberto**

**NÚMERO DE MATRICULA: 52731**

**INTRODUCCIÓN**

Tras aprender en la práctica sobre el modelado de funciones en Matlab, proseguimos avanzando con el diseño de reguladores PID mediante los distintos métodos para poder comparar entre ellos y ver sus ventajas y desventajas.

Para ello comenzamos con ayuda de la herramienta Simulimk con un diseño de un controlador para el motor, planteado como una función de trasferencia, y observamos la salida con el regulador. A partir de este primer resultado nos disponemos a replicar el control realizado, pero partiendo de una FDT distinta:



Dando esta simulación la siguiente gráfica de salida en el “scope” como resultado de la aplicación del regulador PID diseñado sobre el sistema elegido:



Siendo la gráfica en azul la repsuesta sin PID y la amarilla con PID seleccionada para obtener una respuesta justo en el punto en el que los polos se encuentran con igual parte real que imaginaria ().

De acuerdo con lo pedido por la profesora durante la realización de la práctica se mostrarán las gráficas correspondientes a los apartados exigidos, y no de aquellos que (aún habiendo sido cumplimentados previamente al día de la práctica) no se consideran de interés.

**EFECTO DE LA PATADA EN LA CONSIGNA**

Para este apartado se siguen las recomendaciones de la teoría para desarrollar un controlador que disminuya la patada en la consigna mediante la adición de un filtro paso bajo a la acción derivativa, ya que eliminará el efecto nocivo de las altas frecuencias, dejando a bajas frecuencias el efecto derivativo. Este filtro se modifica en función de un valor N que se va modificando para ver las variaciones:

Td=1; Gc=Kp\*(1+1/Ti/s+Td\*s); step(feedback(G\*Gc,1)), hold on

for N=[100,1000,10000,1:10]

Gc=Kp\*(1+1/Ti/s+Td\*s/(1+Td\*s/N)); G\_c=feedback(G\*Gc,1); step(G\_c)

end

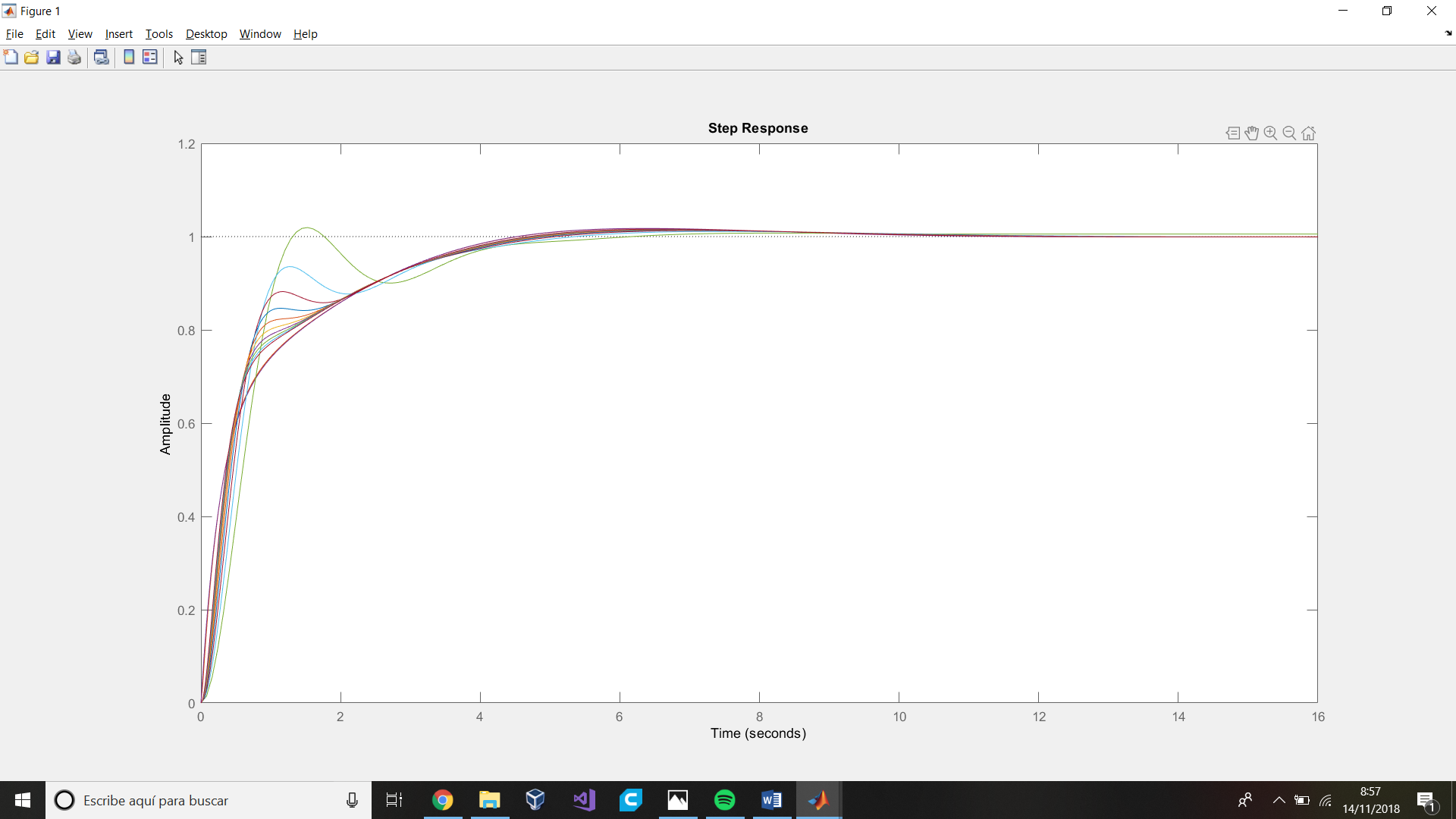
figure; [y,t]=step(G\_c); err=1-y; plot(t,err)

%Dentro de las distintas podibilidades de N se comprueba que para valores

%de N superiores a 25 no se nota el efecto sobre la respuesta en el tiempo

%aunque valores de N mayores sólo harían que la respuesta en altas

%frecuencias sea tanto mayor, y por eso la mejor será la de menor N.



**DISEÑO DE LA FDT**

Para comenzar con este apartado se determina, a partir de la FDT detallada anteriormente, un regulador PID mediante el método de asignación directa de polos debido a que al tratarse de un diseño que no se va implementar en físico, disfrutará de los beneficios de esta técnica sin tener que enfrentarnos a las dificultades de su implementación física.

Por ello se procede a adjuntar el código implementado en el script de Matlab (con sus correspondientes comentarios descriptivos), seguidos de las gráficas asociadas:

Gp = tf ( 1, [1 2 1]);

%Para determinar el controlador PID se aplica el método de asignación

%directa de polos buscando una respuesta con un coeficiente de

%amortiguamiento de 0.707 y una frecuencia natural no amortiguada de 1.41 y

%alfa igual a 1: Finalmente quedan los siguientes valores para el control

%PID:

K = 3.829458496; Ti = 1.35; Td = 0.369;

Gc = tf(K\*[Ti\*Td Ti 1], [Ti 0])

%Por último comprobamos ambas gráficas para ver que el control con PID se

%aproxima a la respuesta deseada

%y = A/(s+alfa\*wn)(s^2+2\*sita\*wn+wn^2);

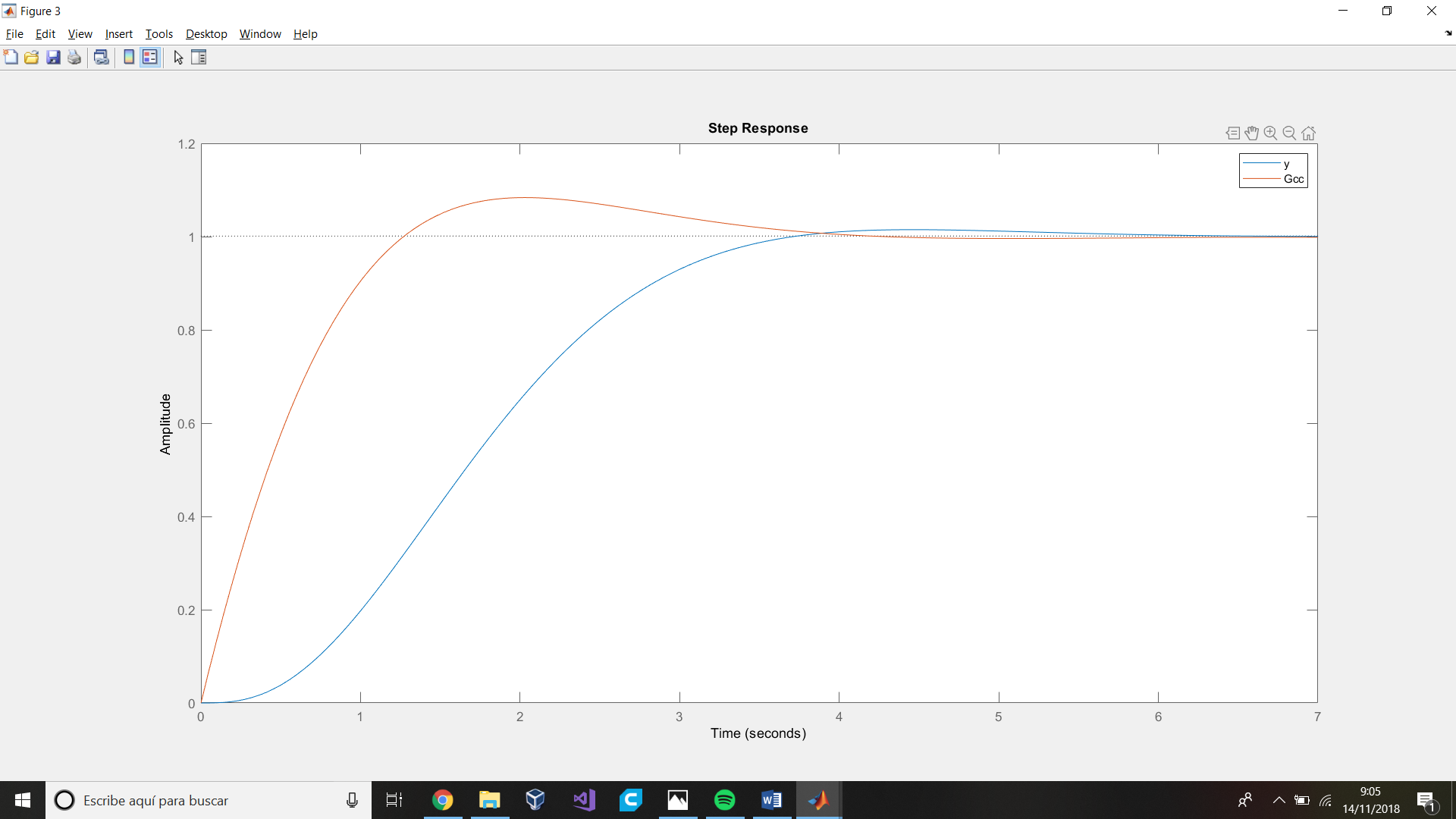
y = tf (2.83286, [1 3.41 4.829458496 2.8297]);

Gcc = Gc\*Gp/(1 + Gc\*Gp);

step(y); hold on; step(Gcc);

figure;

Quedando la siguiente gráfica:



En ella podemos ver que el sistema con la el PID se vuelve más rápido, pero no demasiado oscilatorio. Este era el objetivo buscado mediante la asignación directa de polos.

**PI-D**

A continuación, vemos la otra solución al problema de la patada en la consigna, que consiste en tomar la señal que entra en el efecto derivativo desde la salida en lugar de desde el error.

%PI-D:

figure;

G=tf(1,[1 2 1]); Kp = 3.829458496; Ti = 1.35; Td = 0.369; N=10; s=tf('s');

Gc=Kp\*(1+1/Ti/s+Td\*s/(1+Td\*s/N));

G\_c=feedback(G\*Gc,1); Gc1=Kp\*(1+1/s/Ti);

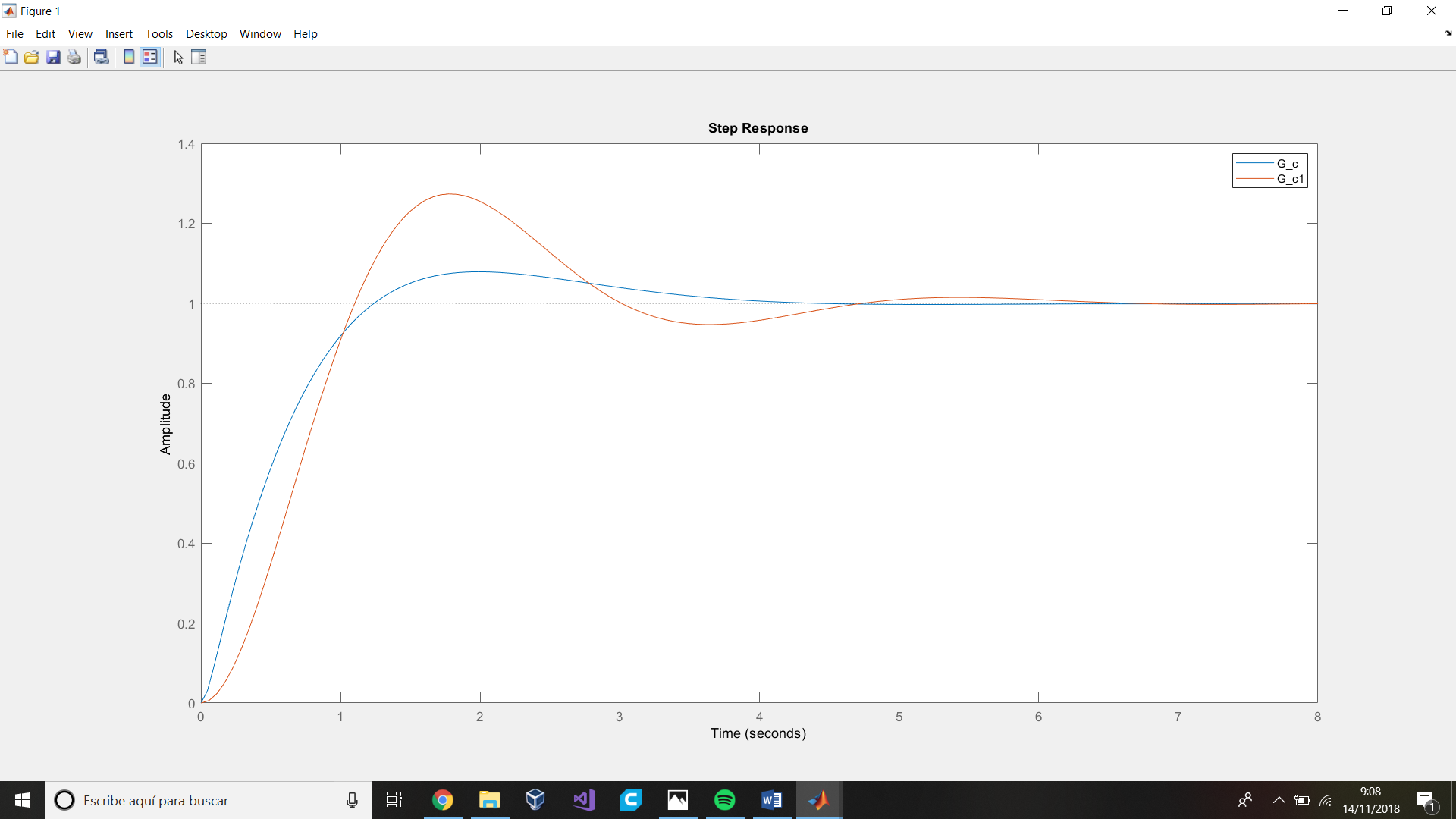
H=((1+Kp/N)\*Ti\*Td\*s^2+Kp\*(Ti+Td/N)\*s+Kp)/(Kp\*(Ti\*s+1)\*(Td/N\*s+1));

G\_c1=feedback(G\*Gc1,H); step(G\_c,G\_c1);

%Para este resultado podemos observar que la aproximación mediante el PI-D

%nos da una respuesta más rápida y con más sobre oscilación, sin embargo

%que no se produce un salto tan pronunciado al inicio



En esta grafica se puede ver que claramente el sistema realimentado es más sobreoscilatorio que una vez aplicado el controlador PI-D, como era de esperar ya que se cancela el efecto de la patada en la consigna.

**ZN**

Tras esto se comienza a realizar el diseño de PID mediante el método empírico de Ziegler-Nichols:

%PID con ZN

G=tf(1, [1 2 1 0]);

nyquist(G); axis([-0.2,0.6,-0.4,0.4])

[Kc,pp,wg,wp]=margin(G); [Kc,wg], Tc=2\*pi/wg;

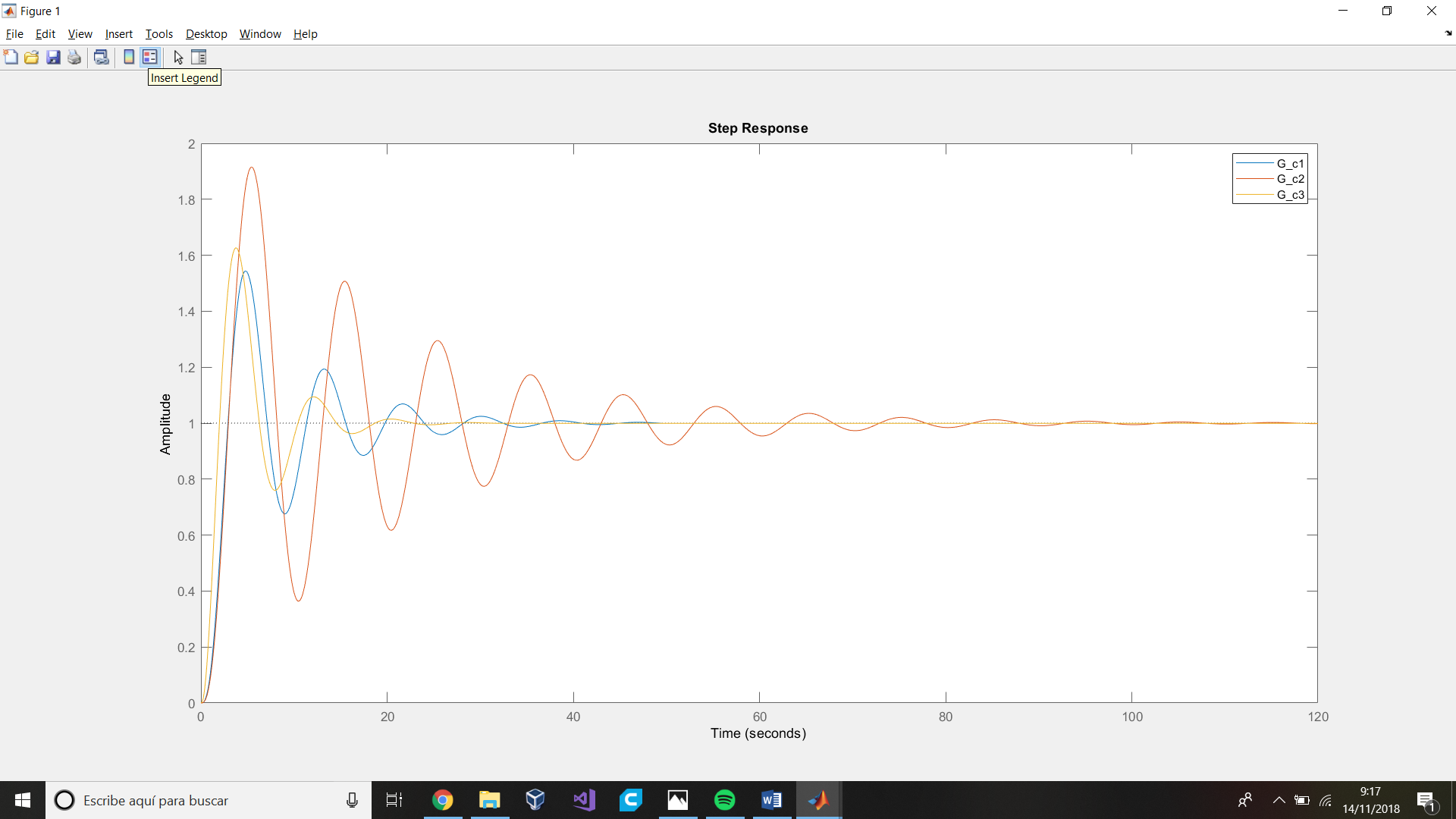
[Gc1,Kp1]=zn(1,[Kc,Tc,10]); Kp1

[Gc2,Kp2,Ti2]=zn(2,[Kc,Tc,10]); [Kp2,Ti2]

[Gc3,Kp3,Ti3,Td3]=zn(3,[Kc,Tc,10]); [Kp3,Ti3,Td3]

G\_c1=feedback(G\*Gc1,1); G\_c2=feedback(G\*Gc2,1);

G\_c3=feedback(G\*Gc3,1); step(G\_c1,G\_c2,G\_c3);



Para este método se usa la función de las prácticas “zn”, y vemos los tres reguladores representados. Se puede observar que en este caso el controlador P (G\_c1) es el menos oscilatorio y conforme se añade la acción integral (G\_c2) se vuelve más oscilatorio y finalmente con las tres acciones (G\_c3) se consigue reducir las oscilaciones.

**CHIEN-HRONES-RESWICK**

Pasamos al siguiente método empírico y vemos sus mejoras frente a ZN:

%Una vez guardada la función para el cálculo de Chreswick se procede a

%calcular los controladores para nuestro sistema.

figure;

s=tf('s'); G = tf(1,[1 2 1]);

[Gc1,Kp,Ti,Td]=zn(3,[k,L,T,N])

[Gc2,Kp,Ti,Td]= chreswickpid (3,1,[k,L,T,N,0])

[Gc3,Kp,Ti,Td]= chreswickpid (3,1,[k,L,T,N,20])

[Gc4,Kp,Ti,Td]= chreswickpid (3,2,[k,L,T,N,0])

G\_zn = feedback(G\*Gc1,1);

G\_chr\_2 = feedback(G\*Gc2,1);

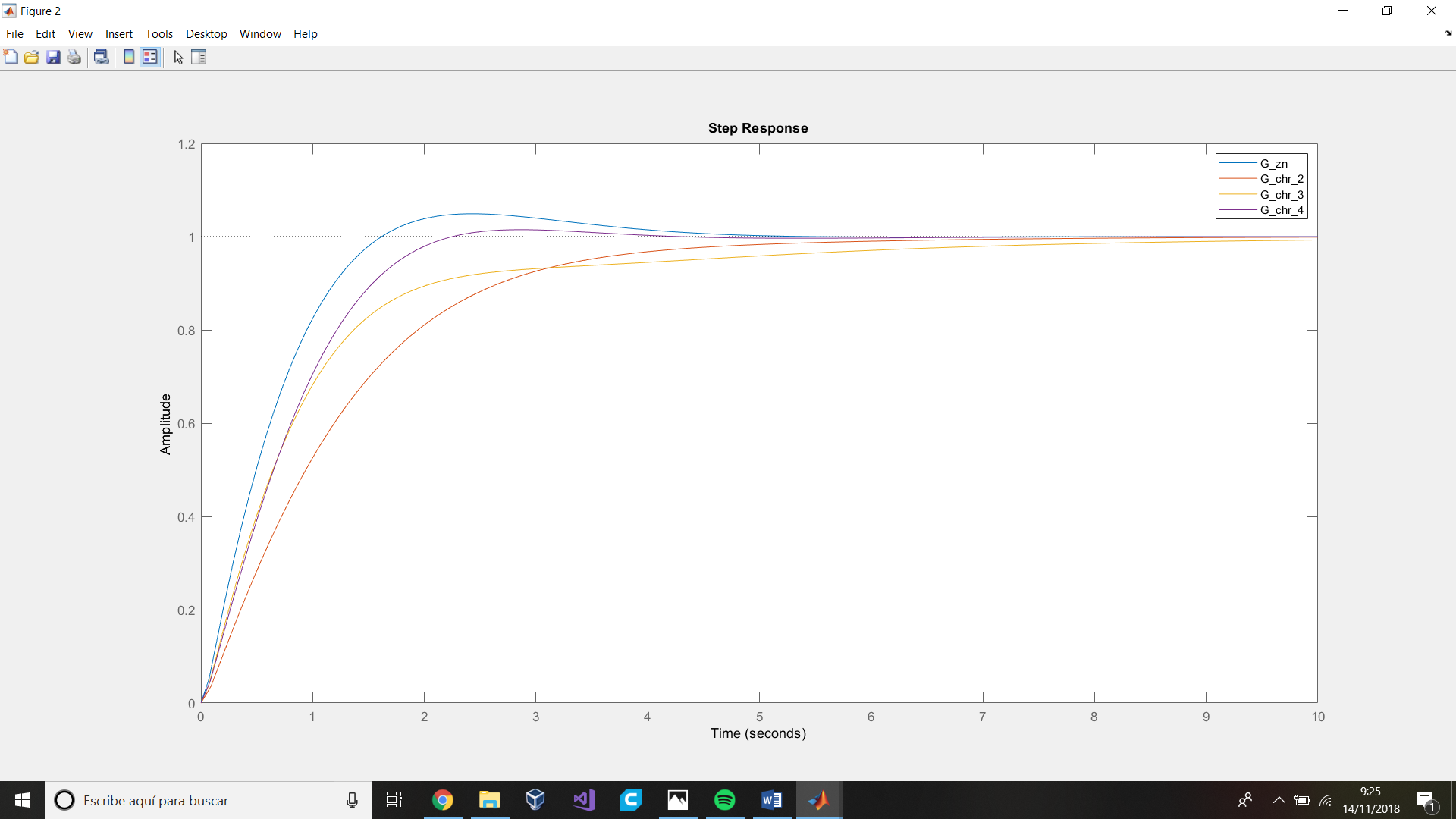
G\_chr\_3 = feedback(G\*Gc3,1);

G\_chr\_4 = feedback(G\*Gc4,1);

step(G\_zn, G\_chr\_2, G\_chr\_3, G\_chr\_4, 10);

%Con chreswick se puede observar que las respuestas sse vuelven menos rápidas

%y menos oscilatorias que con el método de ZN



En esta gráfica se puede observar cómo se vuelve el sistema más lento y menos oscilatorio, y conforme se añaden acciones de control se mejora la repuesta (siendo G\_chr\_2, G\_chr\_3, G\_chr\_4 los controladores P, PD, PID respectivamente).

**COHEN-COON**

Finalmente se concluye la práctica con el diseño de un regulador Cohen-Coon que, aunque se suele utilizar con sistemas con retardo, lo aplicaremos a nuestra sencilla FDT para ver su efecto.

%Ejercicio 7:

figure;

G=tf(1,[1 2 1]);

[Gc1,Kp1]=cohenpid(1,[k,L,T,10])

[Gc2,Kp2,Ti2]=cohenpid(2,[k,L,T,10])

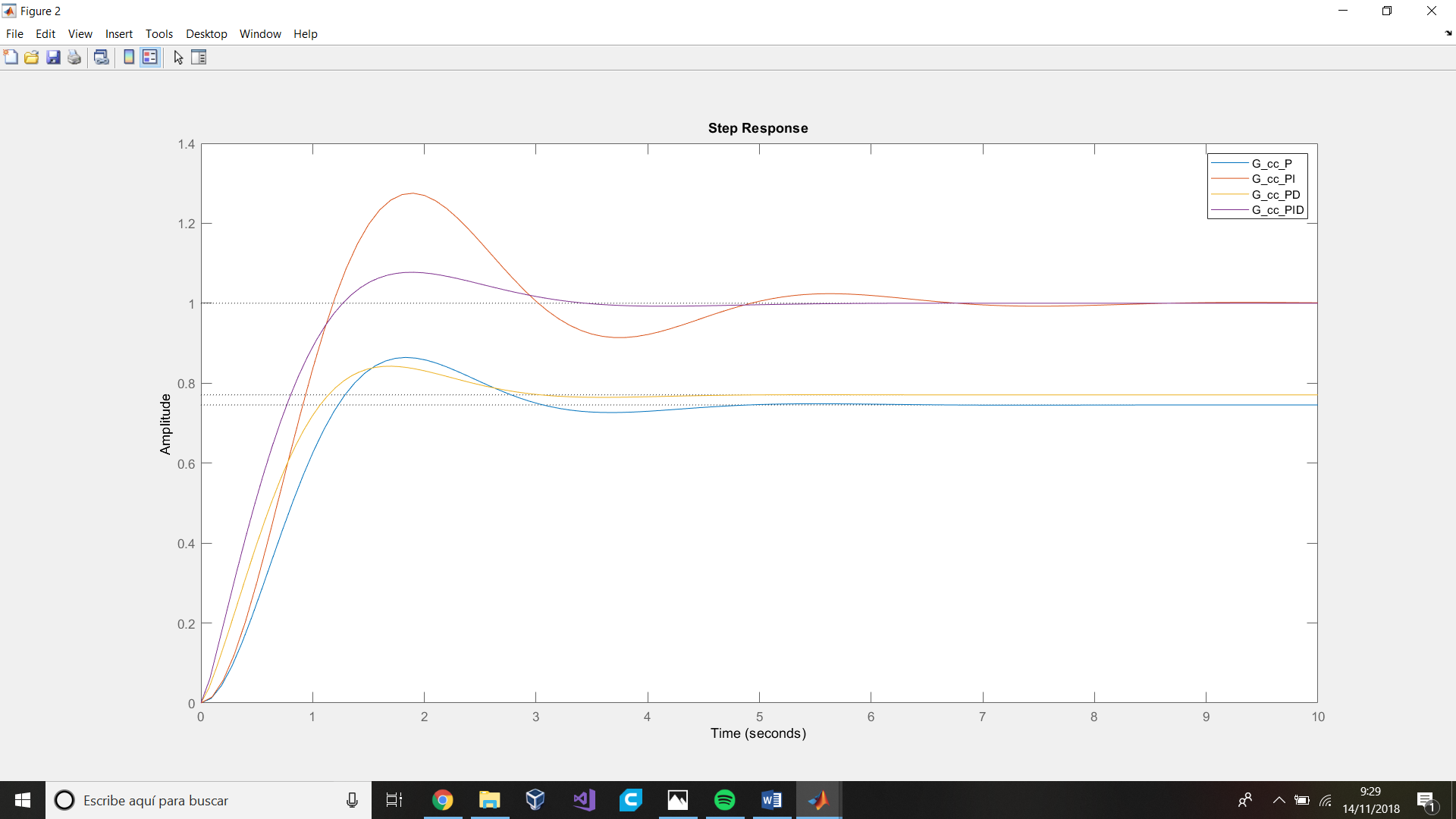
[Gc3,Kp3,Ti3,Td3]=cohenpid(5,[k,L,T,10])

[Gc4,Kp4,Ti4,Td4]=cohenpid(3,[k,L,T,10])

G\_cc\_P=feedback(G\*Gc1,1); G\_cc\_PI=feedback(G\*Gc2,1);

G\_cc\_PD=feedback(G\*Gc3,1); G\_cc\_PID=feedback(G\*Gc4,1);

step(G\_cc\_P,G\_cc\_PI,G\_cc\_PD,G\_cc\_PID,10)



En este caso se puede ver un ejemplo claro del efecto de las distintas acciones de control, ya que el P tiene un gran error en régimen permanente y es lento, pero al añadir distintas acciones de control se va mejorando la respuesta. Se ve cómo la acción integral reduce el error en régimen permanente mientras que vuelve al sistema muy oscilatorio y finalmente se llega a la respuesta más rápida y menos oscilatoria con el PID, consiguiendo a su vez error nulo en régimen permanente.