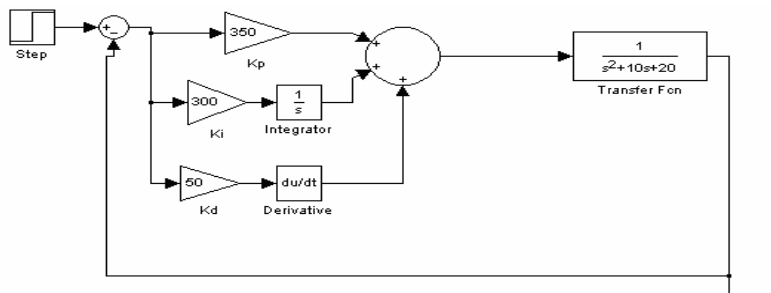




UNIVERSIDAD MAYOR DE SAN SIMON  
FACULTAD DE CIENCIAS Y TECNOLOGIA  
PROGRAMA ELEKTRO  
SOCIEDAD CIENTÍFICA DE INGENIERÍA ELÉCTRICA -  
ELECTRÓNICA

## ANALISIS DE SISTEMAS DE CONTROL DINAMICO CON MATLAB



Autor : Sergio Ureña Delgado

# ANALISIS DE FUNCION DE TRANSFERENCIA

## Introducir Funciones de Transferencia en Matlab:

Una de las maneras de crear un sistema LTI (Linear Time Invariant Sytem), es mediante la función de transferencia. En este apartado veremos como implementar Fdts para sistemas SISO (Single Input, Single Output, una entrada, una salida):

Existen tres maneras básicas de representar la función de transferencia en Matlab:

### A. Usando el comando tf:

Para introducir una FdT del tipo  $W(s) = \frac{n(s)}{d(s)}$ , usando el comando tf, escribir

```
>>W=tf(num,den)
```

donde num y den son vectores representando los coeficientes de los polinomios  $n(s)$  y  $d(s)$ , respectivamente. El resultado de la variable W es un objeto del tipo TF, conteniendo el numerador y el denominador.

#### Ejemplo 1:

Función de transferencia con el comando tf(num,den)

Tenemos tres sistemas (sys1, sys2, sys3) con las siguientes Fdts, vamos a introducirlas usando el comando tf(num,den)

$$H1(s) = \frac{s+1}{s+2} \quad H2(s) = \frac{s^2+2s+3}{s^3+4s^2+5s+6} \quad H3(s) = \frac{3s^2}{2s^2+6s+3}$$

---

```
%-----  
% Ejemplo 1  
% Sistemas de control dinámico con MATLAB  
% En este ejemplo introduciremos las Fdts de los sistemas sys1, sys2, sys 3  
% mediante el comando tf(num,den)  
%-----  
%Definición del sistema sys1:  
  
num=[1 1]; % Aquí se define la variable vector del numerador  
den=[1 2]; % y del denominador.  
sys1=tf(num,den); % Creación del sistema sys1  
  
%Definición del sistema sys2, mediante una sola línea:  
  
sys2=tf([1 2 3],[1 4 5 6]);
```

```

%Definición del sistema sys3:

num=[3 0 0]; % Atención con los ceros
den=[2 6 3];
sys3=tf(num,den);

% Visualización de las Fdts
sys1
sys2
sys3

```

---

Transfer function:

$$\frac{s + 1}{s + 2}$$

Transfer function:

$$\frac{s^2 + 2s + 3}{s^3 + 4s^2 + 5s + 6}$$

Transfer function:

$$\frac{3s^2}{2s^2 + 6s + 3}$$


---

## B. Como expresión racional usando la variable s de Laplace

Primero se define la variable s como un objeto TF:

```
>> s=tf('s');
```

y luego introducimos la función de transferencia como una expresión racional con la variable s.

Nota: Solo hay que definir la variable s como TF una sola vez. Todas las expresiones que vengan a continuación serán objetos TF

Ejemplo 2:

Introducir las Fdts H(s) y G(s) mediante la definición de s como objeto TF.

$$H(s) = \frac{3s^2}{2s^2 + 6s + 3} \quad G(s) = \frac{1}{(s+5)(s+2)^2}$$


---

```

%-----
% Ejemplo 2
% Sistemas de control dinámico con MATLAB
% En este ejemplo introduciremos las FDTs H(s) y G(s)
%-----

% Primero definimos la variable s como un objeto TF

s=tf('s');

% Ahora podemos introducir las funciones directamente:

h=(3*s^2)/(2*s^2+6*s+3);
g=1/((s+5)*(s+2)^2);

% Finalmente las visualizamos:

disp('H(s)');
h
disp('G(s)');
g

```

---

H(s)

Transfer function:  

$$\frac{3 s^2}{2 s^2 + 6 s + 3}$$

-----  

$$2 s^2 + 6 s + 3$$

G(s)

Transfer function:  

$$\frac{1}{s^3 + 9 s^2 + 24 s + 20}$$

-----  

$$s^3 + 9 s^2 + 24 s + 20$$

---

### C. Mediante modelos ZPK (Zero-Pole-Gain)

Una forma alternativa de representar la Fdt, es mediante la factorización del numerador y del denominador:

$$h(s) = \frac{(s - z_1) \dots (s - z_m)}{(s - p_1) \dots (s - p_n)}$$

La ventaja de esta forma es que los ceros (z) y los polos (k) de la FdT son fáciles de visualizar.

La sintaxis para especificar este tipo de modelos es:

```
>>H=zpk(z,p,k)
```

donde,  
z: Son los ceros de la FdT en forma de vector.  
p: Son los polos de la FdT en forma de vector.  
k: Es la ganancia del sistema.

### Ejemplo 3:

Introduce la siguientes FdTs definidas como modelos ZPK:

$$G(s) = \frac{-2s}{(s-2)(s^2-2s+2)} \quad W(s) = \frac{(s+1)(s+3)}{s(s+2)(s+4)}$$

---

```
% %-----
% Ejemplo 3
% Sistemas de control dinámico con MATLAB
% En este ejemplo introduciremos las FDTs G(s) y W(s)
% como modelos zpk (zero-pole-gain)
%-----
% G(s)
% Primero definimos los ceros:
z=[0];
% Después los polos:
p=[1-i 1+i 2];
% y la ganancia
k=[-2]
% Finalmente el comando
G=zpk(z,p,k);
% W(s)
% Aquí lo haremos mediante la definición de s como objeto ZPK
s=zpk('s');
W=((s+1)*(s+3))/(s*(s+2)*(s+4));
% Visualización
disp('G(s)=')
G
disp('W(s)=')
W
```

---

k =        -2

G(s)=

Zero/pole/gain:

      -2 s

-----  
(s-2) (s^2 - 2s + 2)

W(s)=

Zero/pole/gain:

(s+1) (s+3)

-----  
s (s+2) (s+4)

---

## Expansión de fracciones parciales en matlab

MATLAB tiene un comando para obtener la expansión en fracciones parciales  $B(s)/A(s)$ .

Considere la siguiente función:

$$\frac{B(s)}{A(s)} = \frac{num}{den} = \frac{b_0 s^n + b_1 s^{n-1} + \dots + b_n}{s^n + a_1 s^{n-1} + \dots + a_n}$$

en donde algunos  $a_i$  y  $b_j$  pueden ser 0. En MATLAB, los vectores renglón `num` y `den` especifican los coeficientes del numerador y el denominador en la función.

El comando,

```
[r,p,k]=residue(num,den)
```

encuentra los residuos, los polos y los términos directamente de una expansión de fracciones parciales del cociente de polinomios  $B(s)$  y  $A(s)$ .

$$\frac{B(s)}{A(s)} = \frac{r(1)}{s-p(1)} + \frac{r(2)}{s-p(2)} + \dots + \frac{r(n)}{s-p(n)} + k(s)$$

observar  $p(1) = -p_1$ ,  $p(2) = -p_2$ ,  $p(n) = -p_n$  ;  $r(1) = a_1$ ,  $r(2) = a_2$ ,  $r(n) = a_n$  ;  $k(s)$  es un término directo

### A. Ejemplo de descomposición en funciones parciales

Dada la siguiente función, descomponer en fracciones parciales.

$$F(s) = \frac{s+3}{s^2+3s+2} \quad \text{utilizando el comando}$$

```
[r,p,k]=residue(num,den)
```

Obtendremos los residuos, los polos y el término directo.

Ejemplo 4: Comando `residue (num,den)`

Dada la siguiente función, descomponer en fracciones parciales.

$$F(s) = \frac{s+3}{s^2+3s+2}$$

```

%-----
% Ejemplo 4
% Sistemas de control dinámico con MATLAB
% ejemplo: Comando residue
%-----
% Descomponer en fracciones parciales
% Primero introduciremos los valores del numerador y del denominador
num=[ 1 3]
den=[1 3 2]
% Utilizando el comando residue, obtendremos los polos y los residuos
[r,p,k]=residue(num,den)

```

---

```

num =
     0     1     3

```

```

den =
     1     3     2

```

```

r =
    -1
     2

```

```

p =
    -2
    -1

```

```

k =
     []

```

---

### **Método de expansión mediante fracciones parciales:**

Existen tres tipos de fracciones parciales. La forma de las fracciones parciales para cada uno de los tipos es la siguiente:

#### 1. Factores lineales en el denominador

Expresión

Fracciones parciales

$$\frac{f(s)}{(s+a)(s+b)(s+c)}$$

$$\frac{A}{s+a} + \frac{B}{s+b} + \frac{C}{s+c}$$

#### 2. Factores lineales repetidos en el denominador

Expresión

Fracciones parciales

$$\frac{f(s)}{(s+a)^n}$$

$$\frac{A}{s+a} + \frac{B}{(s+a)^2} + \frac{C}{(s+a)^3} + \dots + \frac{N}{(s+a)^n}$$

3. Factores cuadráticos en el denominador, cuando estos se factorizan sólo con términos imaginarios.

Expresión

$$\frac{f(s)}{as^2 + bs + c}$$

Fracciones parciales

$$\frac{As + B}{as^2 + bs + c}$$

o si también hay un factor lineal en el denominador

Expresión

$$\frac{f(s)}{(as^2 + bs + c)(s + d)}$$

Fracciones parciales

$$\frac{As + B}{as^2 + bs + c} + \frac{C}{s + d}$$

**Método de expansión en fracciones parciales sirve para encontrar las transformadas inversas de LaPlace.**



## Transformada de laplace mediante MATLAB

Matlab permite obtener transformadas y antitransformadas de Laplace mediante su módulo de matemática simbólica.

### **Procedimiento:**

- Declarar una variable simbólica con la instrucción `syms`.
- Obtener la transformada para una expresión definida utilizando la variable simbólica anterior

### **Instrucciones de Matlab correspondientes a cada una de las transformadas:**

- `laplace` transformada de Laplace
- `ilaplace` transformada inversa de Laplace

Para calcular transformadas de Laplace, MATLAB utiliza el comando `laplace(f(t))` que se encuentra en el toolbox simbólico. Por defecto la variable de integración es 't' y sólo es necesario definir las variables y constantes simbólicas que se utilicen por medio del comando `sym`.

Para calcular la transformada inversa de Laplace, MATLAB utiliza el comando `ilaplace(f(s))`, siendo `s` la variable independiente por defecto. Este comando devuelve el resultado en función de 't', también aquí es necesario definir las variables y constantes simbólicas que se utilicen por medio del comando `sym`

Para obtener la transformada y la antitransformada de Laplace de una función, se puede introducir de diferentes formas:

#### A. Primer caso:

Para poder calcular la transformada de Laplace introducimos el siguiente comando:

```
>> syms t;  
f = f(t);  
ans = laplace(f)
```

con el comando `syms` definimos la variable `t` y aplicando `laplace` nos calcula la transformada.

#### Ejemplo 5: Primer caso

Calcular la transformada de Laplace de la siguiente función:

$$f(t) = 4 - 4e^{-0.5t}$$

---

```
%-----  
% Ejemplo 5  
% Sistemas de control dinámico con MATLAB
```

```
% En este ejemplo calcularemos la transformada de Laplace de una función
%-----

% Primero definimos la variable t
syms t;
% Luego introducimos la función que queremos transformar
f=(4-4*exp(-0.5*t))
%introducimos el comando laplace
ans=laplace(f)
% para arreglar el resultado
pretty(ans)
```

---

f =

4-4\*exp(-1/2\*t)

ans =

4/s/(2\*s+1)

$$\frac{4}{s(2s+1)}$$


---

Para calcular la anti transformada de Laplace introduciremos el siguiente comando

```
» syms s;
f = f(s);
ans = ilaplace(f)
```

con el comando syms definimos la variable s y aplicando ilaplace nos calcula la anti transformada.

### B. Segundo caso:

Supongamos que queremos encontrar la transformada de Laplace de  $\sin(w*t)$ , donde la variable de integración es 't' y 'w' es una constante, en la línea de comando escribimos:

```
» w=sym('w');
» t=sym('t');
» laplace(f(t,w))
```

Primero se define la constante w y luego la variable t.

### Ejemplo 6: Antitransformada de Laplace

Calcular la antitransformada de Laplace de la siguiente función:

$$f(s) = \frac{s+5}{s^2+3s+2}$$

```

%-----
% Ejemplo 6
% Sistemas de control dinámico con MATLAB
% En este ejemplo veremos como se calcula la antitransformada de una función
%-----

% Definimos la variable s
syms s
%introducimos la función que queremos antitransformar
f=(s+5)/(s^2+3*s+2);
%Aplicamos el comando ilaplace
ans=ilaplace(f)

```

---

```

ans =

-3*exp(-2*t)+4*exp(-t)

```

---

Para el cálculo de la antitransformada aplicamos:

```

» w=sym('w');
» t=sym('s');
» ilaplace(f(s,w))

```

Primero se define la constante w y luego la variable s

Ejemplo 7: Caso2. Antitransformada de Laplace

Calcular la antitransformada de Laplace de la siguiente función:

$$f(s) = \frac{w}{s^2 + w^2}$$


---

```

%-----
% Ejemplo 7
% Sistemas de control dinámico con MATLAB
% En este ejemplo veremos otra forma de calcular la antitransformada de
Laplace
%-----
% Definimos la constante 'w'
w=sym('w');
%definimos la variable s
s=sym('s');

%utilizamos el comando ilaplace
ilaplace(w/(s^2+w^2))

```

---

```
ans =  
  
sin(w*t)
```

---

### C. Tercer caso:

Otra forma más simple para encontrar la transformada es utilizando:

```
» syms a s t w x  
laplace(f(t))  
laplace(f(a,s))  
laplace(f(w,x))
```

Definiendo primero las variables y las constantes, sin comas ni comillas.

Ejemplo 8: Caso3. Transformada de Laplace

Calcular la transformada de Laplace de las siguientes funciones:

$$f(t) = t^5$$

$$f(s) = e^{as}$$

$$f(x) = \cos(wx)$$

---

```
%-----  
% Ejemplo 8  
% Sistemas de control dinámico con MATLAB  
% En este ejemplo veremos otra forma de calcular la transformada de Laplace  
%-----  
  
% definimos las variables y las constantes  
syms a s t w x  
  
% introducimos el comando laplace para calcular la transformada de cada  
función  
  
laplace(t^5)  
pretty(ans)  
laplace(exp(a*s))  
pretty(ans)  
laplace(cos(w*x),t)  
pretty(ans)
```

---

```
ans =  
  
120/s^6
```

120  
---

$$\frac{6}{s}$$

ans =  
1/(t-a)

$$\frac{1}{t-a}$$

ans =  
t/(t^2+w^2)

$$\frac{t}{t^2 + w^2}$$

Para el cálculo de la antitransformada

```
» syms a s t w x
ilaplace(f(s))
ilaplace(f(t))
ilaplace(f(w,y))
```

Definiendo primero las variables y las constantes, sin comas ni comillas.

#### Ejemplo 9: Caso 3. Antitransformada de Laplace

Calcular la antitransformada de Laplace de las siguientes funciones:

$$f(s) = \frac{1}{s-1}$$

$$f(t) = \frac{1}{t^2+1}$$

$$f(y) = \frac{y}{y^2+w^2}$$

```
%-----
% Sistemas de control dinámico con MATLAB
% En este ejemplo veremos otra forma de calcular la antitransformada de
% Laplace
%-----

% definimos las variables y las constantes de las funciones
syms s t w x y

% introducimos el comando ilaplace
```

```

ilaplace(1/(s-1))
pretty(ans)
ilaplace(1/(t^2+1))
pretty(ans)
ilaplace(y/(y^2 + w^2),y,x)
pretty(ans)

```

---

```

ans =
    exp(t)
                                exp(t)

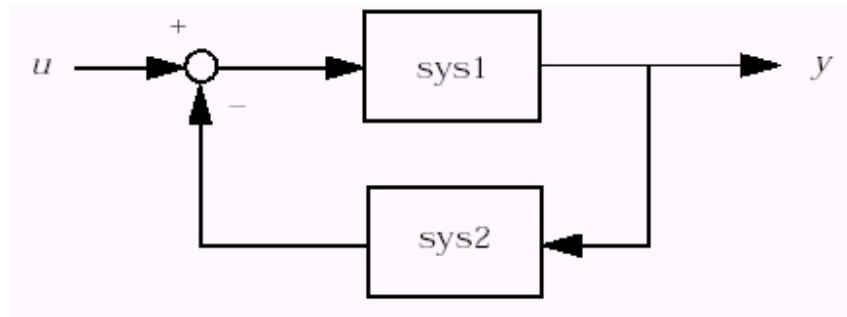
ans =
    sin(x)
                                sin(x)

ans =
    cos(w*x)
                                cos(w x)

```

---

## Sistemas realimentados en MATLAB:



Matlab dispone de un comando para crear sistemas realimentados a partir de dos sistemas LTI (sys1 y sys2), la sintaxis básica es la siguiente:

```
>>w=feedback(sys1,sys2)
```

El sistema resultante w tiene como entrada u y como salida y. Los sistemas sys1 y sys2 pueden ser de diferente tipo (por ejemplo sys1 TF y sys2 ZPK), pero ambos han de ser continuos o discretos. El tipo de sistema resultante LTI dependerá de las reglas de prioridad de sistemas. (Ver introducción a Control Toolbox).

Por defecto Matlab asume una realimentación negativa, para aplicar una realimentación positiva la sintaxis es:

```
>>w=feedback(sys1,sys2,+1)
```

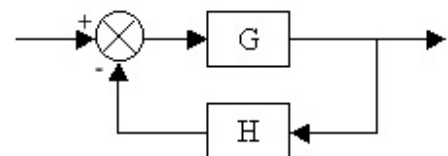
Para más detalles sobre este comando escribir en Matlab: **help feedback**

Ejemplo 10: Sistemas Realimentados en Matlab con feedback.

Calcula el sistema realimentado de la figura para los siguientes casos:

$$A: G(s) = \frac{10}{s^2 + 5s + 3}; H(s) = \frac{5(s+4)}{(s+1)(s+2)}$$

$$B: G(s) = \frac{1}{(s+2)(s+5)}; H(s) = 1$$



```
%-----
% Ejemplo 10
% Sistemas de control dinámico con MATLAB
% En este ejemplo veremos el uso del comando feedback para definir dos
% sistemas
%realimentados.
%-----
```

```
% A:
% Primero definimos G y H
G=tf([10],[1 5 3]);
H=zpk(-4,[-1 -2],5);
% Y luego aplicamos el comando
A=feedback(G,H);
```

```
% B:
% Definición de G:
G=zpk([],[-2 -5],1); % Al no tener cero indicamos [] y ganancia 1
% Al ser realimentación unitaria hacemos:
B=feedback(G,1);
```

```
% Visualización
A,B
```

---

```
Zero/pole/gain:
      10 (s+1) (s+2)
-----
(s+3.849) (s+5.298) (s^2 - 1.147s + 10.1)
```

```
Zero/pole/gain:
      1
-----
(s+2.382) (s+4.618)
```

El resultado es un objeto ZPK, esto es debido a las reglas de prioridad de sistemas en Matlab

---

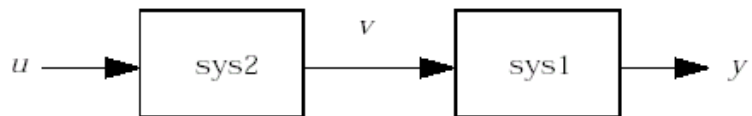


## Simplificación de bloques con MATLAB

La control System Toolbox dispone de varias herramientas para la interconexión de sistemas. Básicamente podemos distinguir:

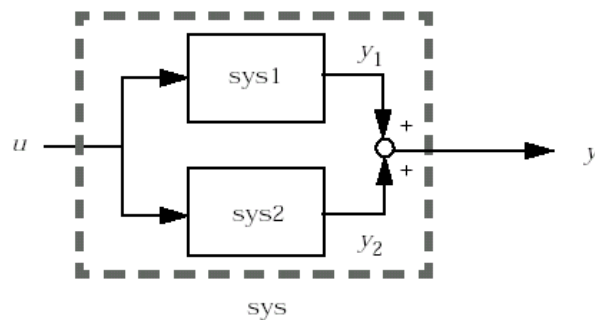
### A. Mediante operaciones aritméticas

Matlab permite realizar la mayoría de operaciones aritméticas con los sistemas LTI. Con la ayuda de estas operaciones podemos llegar a simplificar sistemas, como por ejemplo:



```
>>sys=sys1*sys2
```

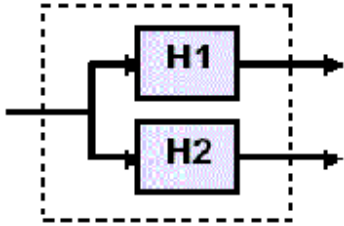
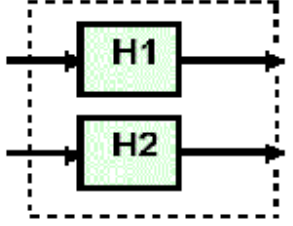
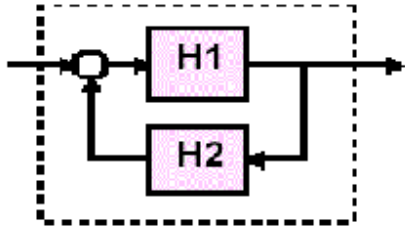
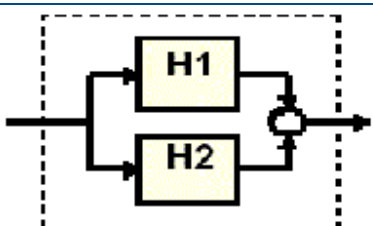
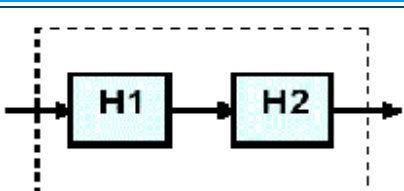
```
>>sys=sys1+sys2
```



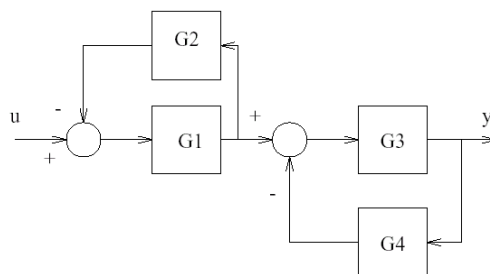
### B. Mediante funciones propias

Aparte de las operaciones aritméticas la Control System ToolBox tiene una serie de funciones para la interconexión de sistemas. Entre ellas podemos destacar:

Función	Descripción	Sintaxis basica	Diagrama
[ , ]	Sumar sistema entrantes	$H=[H1;H2]$	<pre>graph LR; u1 --&gt; H1[H1]; u2 --&gt; H2[H2]; H1 --&gt; S((+)); H2 --&gt; S; S -- y --&gt; y;</pre>

[;]	Distribuir Entradas	$H=[H1,H2]$	
append	Crear modelos independientes	$H=\text{append}(H1,H2)$	
feedback	<a href="#">Sistema Realimentado</a>	$H=\text{feedback}(H1,H2)$	
parallel	Es el paralelo de dos sistemas, es equivalente a sumar sistemas.	$H=\text{parallel}(H1,H2)$	
series	Sistemas en serie, es equivalente a multiplicar sistemas.	$H=\text{series}(H1,H2)$	

Ejemplo 12: Consideramos el siguiente diagrama de bloques:



En este diagrama de bloques las funciones son del tipo:

$$G_1(s) = \frac{a_1(s)}{b_1(s)} \quad G_2(s) = \frac{a_2(s)}{b_2(s)} \quad G_3(s) = \frac{a_3(s)}{b_3(s)} \quad G_4(s) = \frac{a_4(s)}{b_4(s)}$$

Escribiremos un fichero m, el cual cojera los vectores  $a_1, b_1, a_2, b_2, a_3, b_3, a_4, b_4$  correspondientes a las funciones de transferencia y calcularemos la FdT equivalente al sistema.

En el workspace de MATLAB definiremos estos polinomios que representan nuestras funciones de transferencia

```
>> a1=[1 2 3];  
>> b1=[4 5 6 6];  
>> a2=[7 ];  
>> b2=[7 8 6];  
>> a3=[1 0 4];  
>> b3=[7 8 9 4];  
>> a4=[1 2];  
>> b4=[1 2 3 4 5];
```

---

```
%-----  
% Ejemplo 12  
% Sistemas de control dinámico con MATLAB  
% Ejemplo de simplificación usando funciones propias  
% En este caso se supone que los vectores a1,b1,a2,b2,a3,b3,a4,b4 ya  
% están definidos en el entorno de trabajo de matlab  
%-----  
  
% Creación de los bloques  
  
G1=tf(a1,b1);  
G2=tf(a2,b2);  
G3=tf(a3,b3);  
G4=tf(a4,b4);  
  
% Simplificación:  
  
G34=feedback(G3,G4);  
G12=feedback(G1,G2);  
G=series(G12,G34)
```

---

Transfer function:

```
7 s^10 + 36 s^9 + 136 s^8 + 360 s^7 + 774 s^6 + 1290 s^5 + 1781 s^4 + 1956 s^3  
+ 1742 s^2 + 1008 s + 360  
-----  
196 s^12 + 1085 s^11 + 3504 s^10 + 8375 s^9 + 16198 s^8 + 25662 s^7 + 33928  
s^6 + 37634 s^5 + 34508 s^4
```

$$15056 s^2 + 6449 s + 1596$$

$$+ 25914 s^3 +$$

## **ANÁLISIS DE LA RESPUESTA TRANSITORIA**

### **Análisis impulsional con MATLAB**

En la práctica, la función impulso es físicamente imposible. Sin embargo el análisis impulsional es una herramienta usada habitualmente, ya que gracias a ella es posible comprobar la estabilidad y la velocidad de un sistema.

Para ver la respuesta de un sistema ante una entrada impulsional, en MATLAB usamos el comando `impulse`:

*Sintaxis:*

- **`>>Impulse(sys)`**

Esta función dibujara la respuesta impulsional para un sistema LTI. El modelo puede ser continuo o discreto, SISO o MIMO. La duración de la simulación será determinada automáticamente para mostrar correctamente la respuesta transitoria del sistema.

- **`>>Impulse(sys,t)`**

De esta forma determinamos el tiempo de simulación explícitamente. Podemos especificar `t` como el tiempo final o también mediante un vector del tipo: `t = 0:dt:Tfinal` donde `dt` es el tiempo de muestreo.

- **`>>impulse(sys1,sys2,...,sysN)`  
**`>>impulse(sys1,sys2,...,sysN,t)`****

Esta es la sintaxis para dibujar la respuesta impulsional de varios sistemas LTI en una misma figura.

- **`>>impulse(sys1,'y:',sys2,'g--')`**

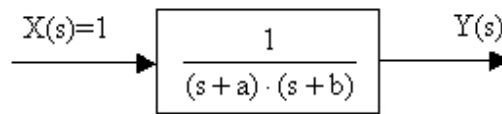
Como cualquier gráfico en MATLAB, podemos especificar colores y estilos de línea, como por ejemplo:

- **`>>[y,t] = impulse(sys)`**

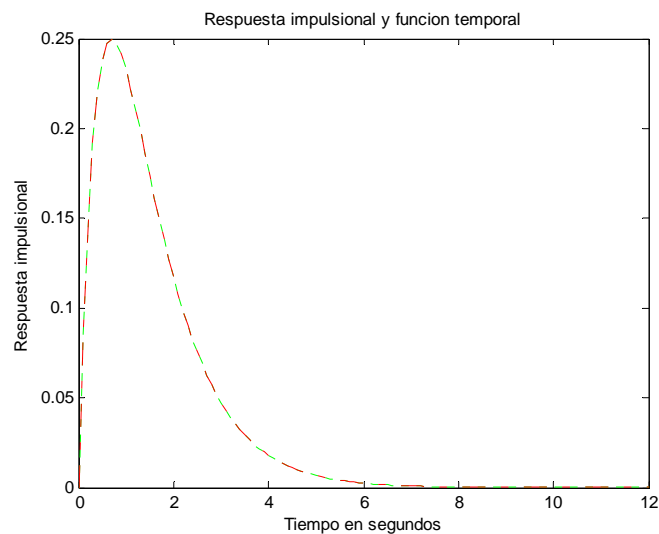
De esta forma obtenemos la respuesta impulsional `y`, y el vector de tiempo `t`, utilizado en la simulación. No se dibuja ninguna gráfica.

Ejemplo 13:

Dado el siguiente sistema crea un fichero `m` que realice:



- Representa la respuesta impulsional del sistema.
- En otro gráfico representa la respuesta impulsional hasta  $t=12$  s.
- Realiza una comparación entre la respuesta encontrada con el comando `impulse`, y la respuesta antitransformando  $Y(s)$




---

```

%-----
% Ejemplo 13
% Sistemas de control dinámico con MATLAB
% En este ejemplo veremos el uso del comando impulse
%-----
% Primero definimos el sistema
clf %Borrado de posibles gráficos
a=1;
b=2;
s=zpk('s');
sys=1/((s+a)*(s+b));
figure(1)
impulse(sys) %Sintaxis básica para respuesta impulsional
% Ahora variaremos el tiempo hasta 12 seg.
t=0:0.1:12;
figure(2) % lo representamos en otro grafico
impulse(sys,t) %Sintaxis de la función impulso definiendo el tiempo
% Ahora realizaremos una comparación entre la función impulse
% y la respuesta temporal
y=impulse(sys,t);%Guardamos la salida en el vector y
figure(3)
plot(t,y,'g:');%Representamos la respuesta con el comando plot
f=(exp(-a*t)-exp(-b*t))/(b-a);%Respuesta temporal
  
```

```
hold on
plot(t,f,'r-.')
xlabel('Tiempo en segundos');
ylabel('Respuesta impulsional');
title('Respuesta impulsional y función temporal');
```

## Análisis impulsional con SIMULINK

Primero nos definimos una función de transferencia en el workspace

```
>> num=[1 5];
>> den=[1 3 2];
>> sis=tf(num,den)
```

Transfer function:  

$$\frac{s + 5}{s^2 + 3s + 2}$$

-----  

$$s^2 + 3s + 2$$

```
>> impulse(sis) % Dibujamos la función impulse desde Matlab
>> grid
>> [x y] = ginput(1)
```

Siendo el punto más alto el siguiente

```
x = 0.4064
y = 1.3283
```

Simulink no tiene ninguna fuente de entrada de impulsos, pero es posible la aproximación de un impulso de entrada con dos entradas step para crear un único pulso de gran magnitud y muy corta duración. El área de este es de un solo pulso de la magnitud del impulso. Para crear este pulso, el primer step de entrada comienza en el momento cero y amplitud positiva. El segundo step de entrada comienza en el momento T y tiene consecuencias negativas de amplitud. La magnitud de cada uno de los pasos es igual a la magnitud del impulso deseado dividido por el tiempo de retardo T.

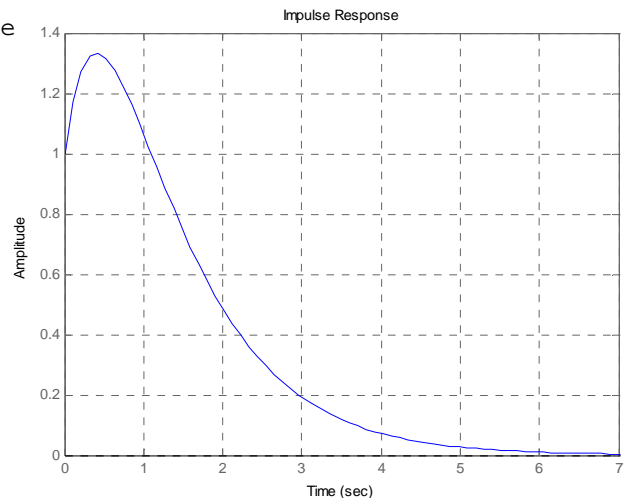
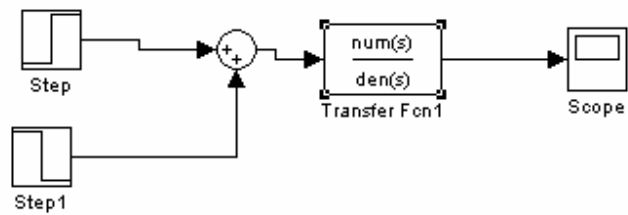
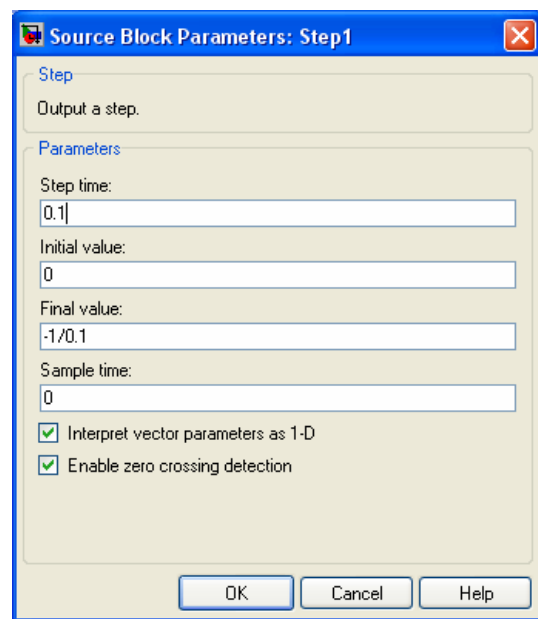
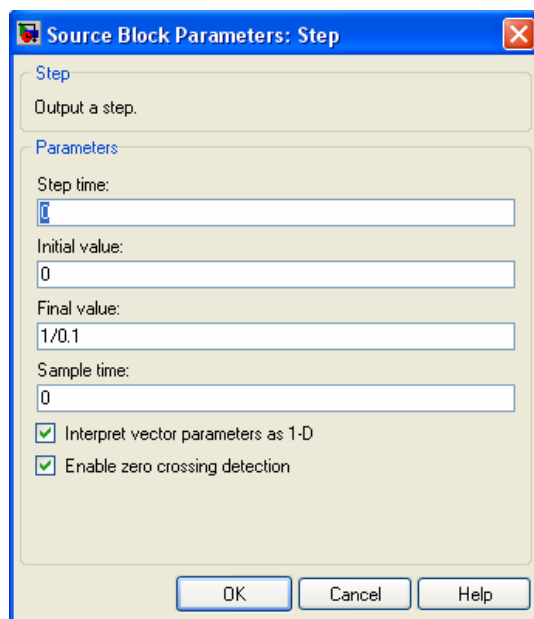


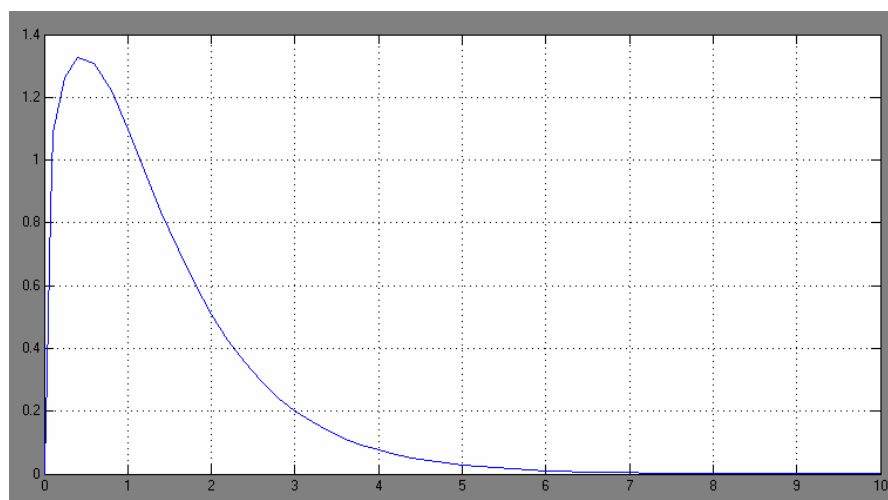
Diagrama de bloques en simulink



### Configuración de los bloques Step



### Grafica de salida en SCOPE



Se verifico que la grafica es idéntica a la anterior grafica obtenida desde Matlab.

## **Respuesta escalón unitario con MATLAB**

El análisis de un sistema frente a un escalón unitario es el más usado en la ingeniería de control. El comando utilizado para realizar este análisis con MATLAB es `step`.

La sintaxis del comando es prácticamente igual que la del comando [`impulse`](#):

```
>> step(sys)
```

Esta función dibujara la respuesta de un sistema LTI, ante una entrada escalón unitario. El modelo puede ser continuo o discreto, SISO o MIMO. La respuesta para un sistema con múltiples entradas será una colección de respuestas escalón por cada entrada. En este caso la duración de la simulación es determinada automáticamente por MATLAB.

```
>>step(sys,t)
```

De esta forma determinamos el tiempo de simulación explícitamente. Podemos especificar `t` como el tiempo final o también mediante un vector del tipo: `t = 0:dt:Tfinal` donde `dt` es el tiempo de muestreo.

```
>>step(sys1,sys2,...,sysN)
>>step(sys1,sys2,...,sysN,t)
```

Esta es la sintaxis para dibujar la respuesta escalón de varios sistemas LTI en una misma figura.

```
>>step(sys1,'y:',sys2,'g--')
```

Como cualquier gráfico en MATLAB, podemos especificar colores y estilos de línea.

```
>>[y,t] = step(sys)
```

De esta forma obtenemos la respuesta escalón `y`, y el vector de tiempo `t`, utilizado en la simulación. No se dibuja ninguna gráfica

---

### Ejemplo 14:

Tenemos dos sistemas con las siguientes funciones de transferencia:



$$sys1 = \frac{1}{s^2 + 0.5s + 1} \quad y \quad sys2 = \frac{1}{s^2 + 0.5s + 4}$$

Realizar con MATLAB una gráfica donde veamos la respuesta de los dos sistemas ante un escalón unitario con un tiempo de simulación de 30s. También representar en la misma gráfica, la función escalón unitario.

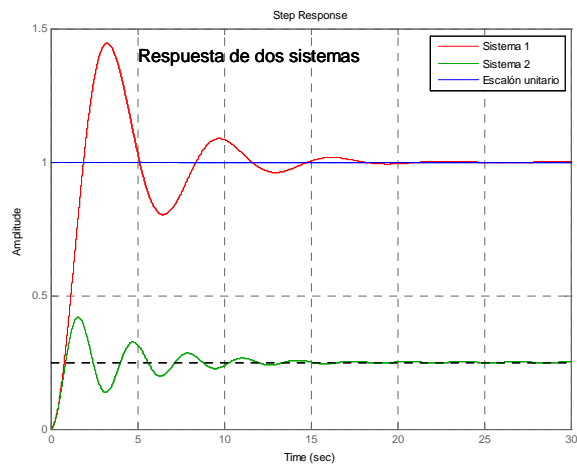
```
%-----
% Ejemplo 14
% Sistemas de control dinámico con MATLAB
% En este ejemplo veremos el uso del comando step
%-----

% Definición de los sistemas:
sys1=tf([1],[1 0.5 1]);
sys2=tf([1],[1 0.5 4]);

% Representación de la respuesta:
t=0:0.01:30; % Respuesta hasta los 30 s.
step(sys1,'r', sys2,'g',t); % Representación en la misma grafica

% Aplicaremos rejilla y pondremos un titulo con text:
grid
text(5, 1.4, 'Respuesta de dos sistemas', 'FontSize',13);

% Representación de la entrada
t0 = -2.0:0.01:-0.01; % definición u(t)=0, -2<=t<=-.01
u0 = zeros(size(t0));
t1 = 0:0.01:30; % definición u(t)=1, 0<=t<=25
u1 = ones(size(t1));
t = [t0 t1]; % creamos t and u(t)
u = [u0 u1];
hold on
plot(t,u);
legend('Sistema 1','Sistema 2','Escalón unitario');
```

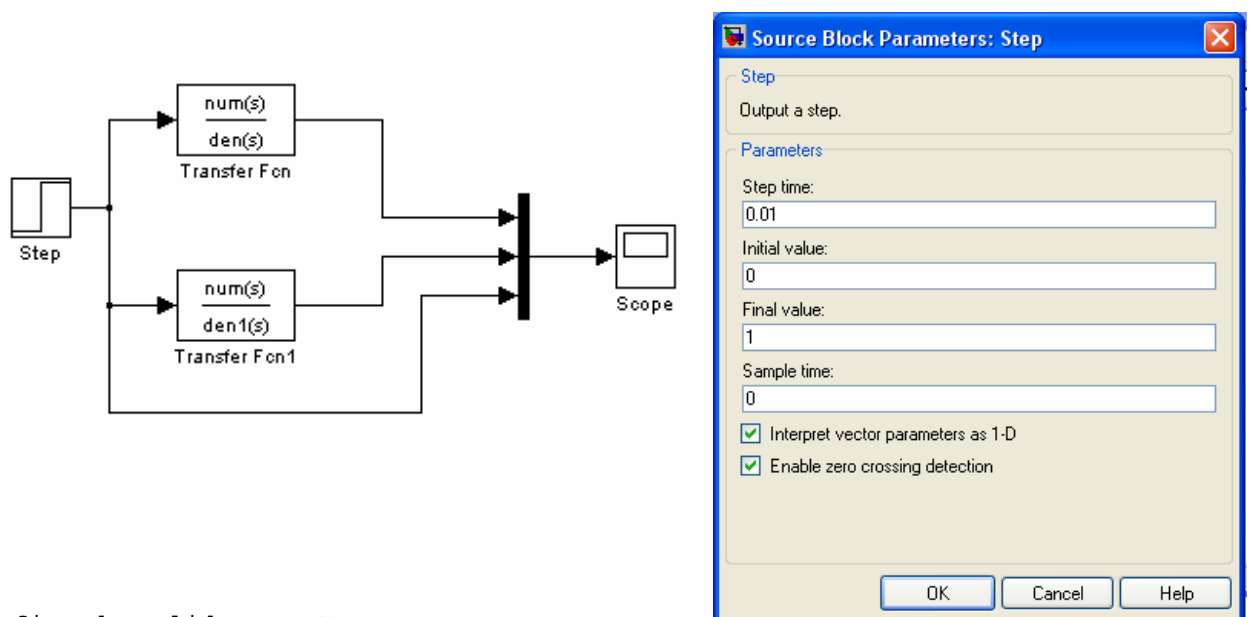


## Respuesta escalón unitario con Simulink

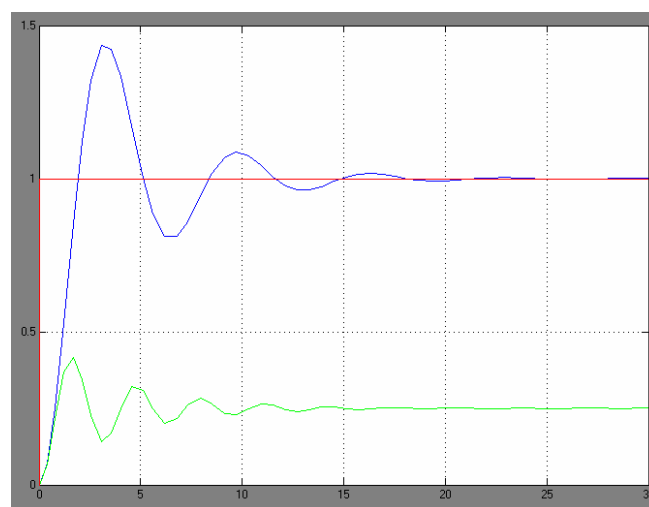
Tenemos que determinar las funciones de transferencia en el workspace para poder usar en el simulink. Usaremos las mismas funciones de transferencia del ejemplo anterior para hacer una comparación de las graficas resultantes.

```
>> num=[1];  
>> den=[1 0.5 1];  
>> den1=[1 0.5 4];
```

Diagrama de bloques en simulink Configuración del bloque Step



Grafica de salida en SCOPE



## Respuesta ante una rampa con MATLAB

En la Control System Toolbox no existe ningún comando que determine la respuesta de un sistema ante una rampa unitaria. A pesar de este inconveniente, podemos encontrar la respuesta mediante el comando [step](#):

Para obtener la respuesta a una entrada en rampa de la función de transferencia del sistema  $G(s)$ , hay que dividir  $G(s)$  por  $s$  y utilizar la orden de respuesta a un salto.

Por ejemplo sea el sistema:

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

Para una entrada en rampa unitaria se tiene que  $R(s) = 1/s^2$ . Por tanto:

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

Entonces aplicamos el comando `step` a  $G(s)/s$

Ejemplo 15:

Dado el siguiente sistema con una FdT  $G(s) = \frac{C(s)}{R(s)} = \frac{1}{s^2 + s + 1}$ , representar la respuesta ante una rampa unitaria para un tiempo de respuesta de 7s. Incluir también en la misma gráfica la función rampa aplicada en la entrada, para así poder ver el error del sistema.

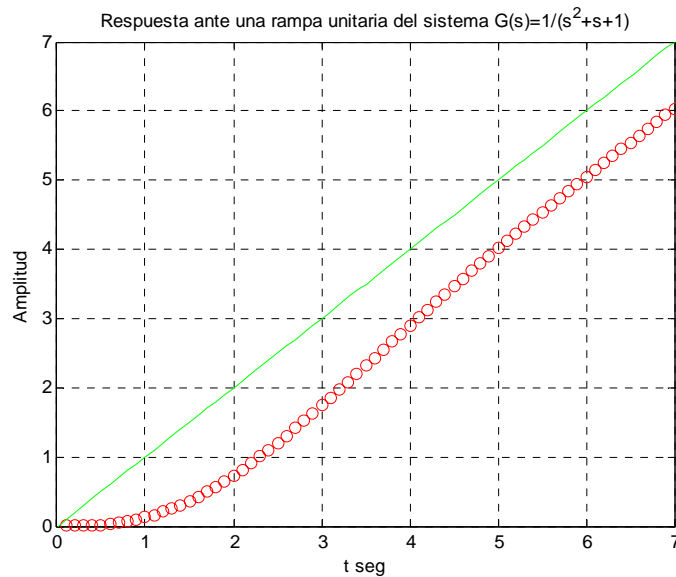
---

```
%-----
% Ejemplo 15
% Sistemas de control dinámico con MATLAB
% Respuesta ante una entrada unitaria en rampa
%-----
%***La respuesta a una entrada unitaria en rampa se obtiene como la respuesta
%***a un salto unitario de G(s)/s

%Primero introducimos el numerador y denominador de G(s)/s
num=[1];
den=[1 1 1 0];
%Creamos el sistema
sys=tf(num,den);
%Especificamos el tiempo de calculo en 7 s.
t=0:0.1:7;
%Guardamos la respuesta en un vector c
c=step(sys,t);
```

```
%Al representar la respuesta a una rampa, añadimos la entrada de referencia
%La entrada de referencia es t.El comando plot es:
plot(t,c,'ro',t,t,'g-');
%Introducimos la rejilla y las etiquetas:
grid
title('Respuesta ante una rampa unitaria del sistema G(s)=1/(s^2+s+1)');
xlabel('t seg');ylabel('Amplitud');
```

---

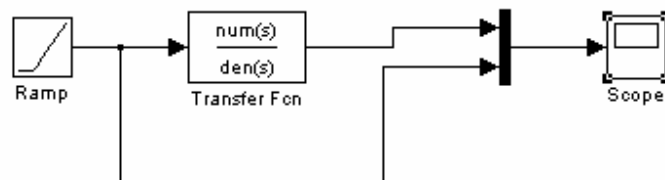


## Respuesta ante una rampa con Simulink

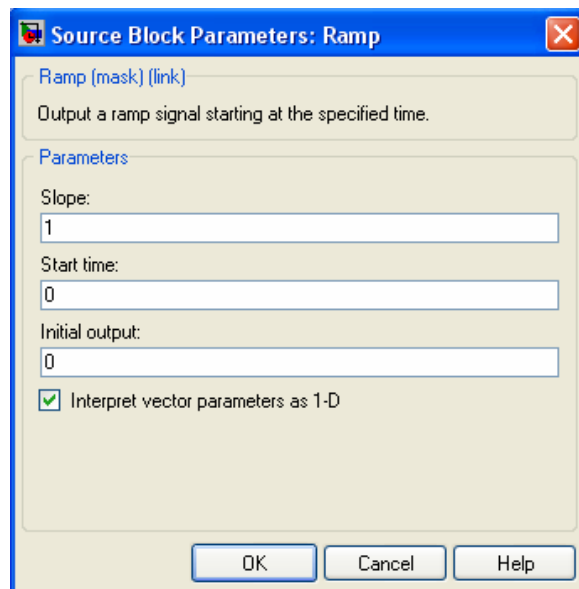
Tenemos que determinar la función de transferencia en el workspace para poder usar en el simulink. Usaremos la misma función de transferencia del ejemplo anterior para hacer una comparación de las graficas resultantes.

```
>> den=[1 1 1];
>> num=[1];
```

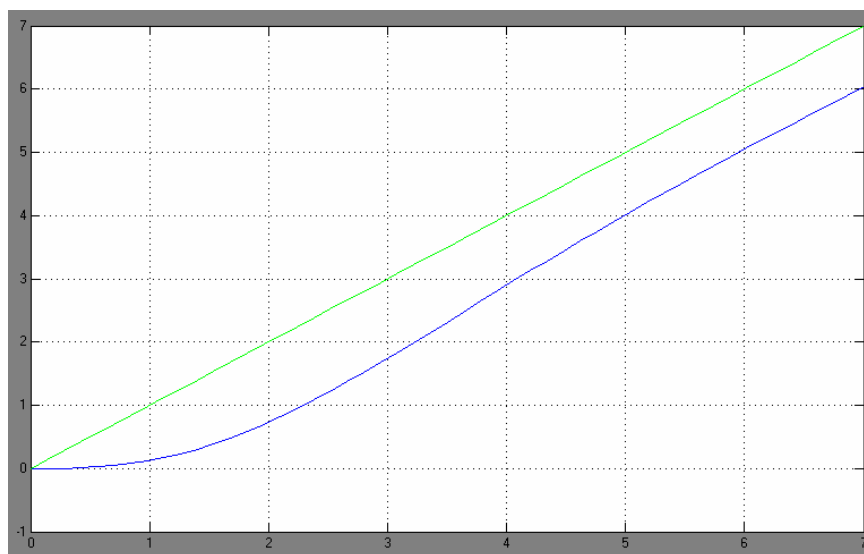
Diagrama de bloques en simulink



Configuración del bloque Ramp



Grafica de salida en SCOPE



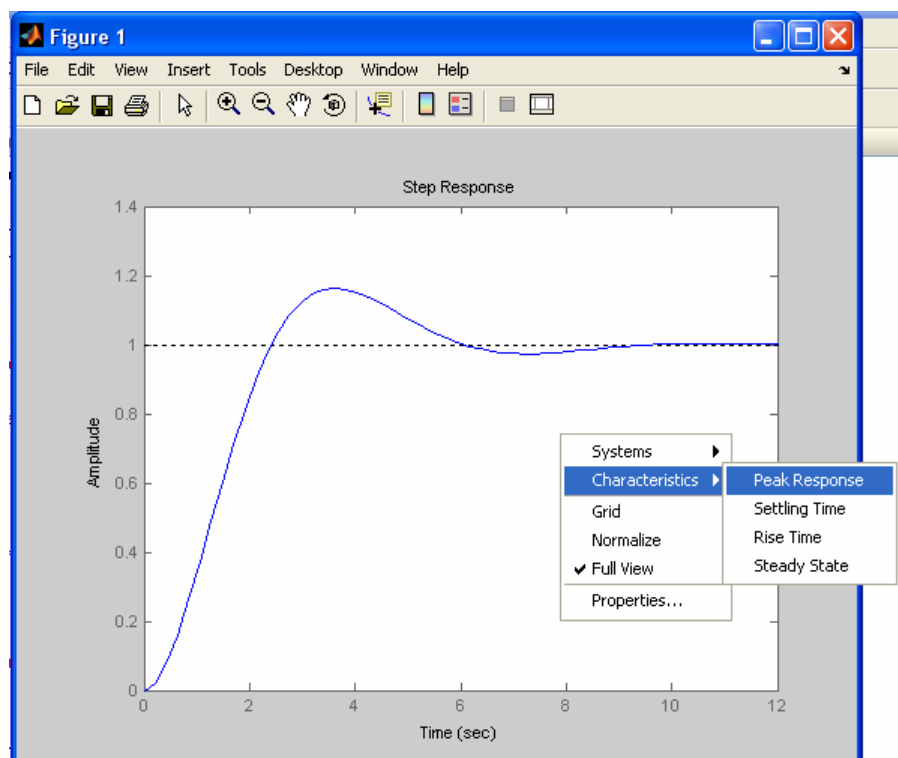
## Sistemas de primer y segundo orden

Como vimos en el tema 2 los sistemas lineales están definidos por una ecuación diferencial. Estas ecuaciones se pueden clasificar como de primer orden, segundo orden, tercer orden..., de acuerdo con la derivada de mayor orden en la ecuación. En esta sección estudiaremos el comportamiento temporal de los sistemas de primer orden y segundo orden.

### Análisis mediante ficheros m:

Para realizar el análisis temporal de cualquier sistema, realizaremos los siguientes pasos:

- 1) Modelizar el modelo según los comandos vistos: [tf](#), [zpk](#).
- 2) Aplicar los análisis temporales que creamos convenientes, según la excitación step, impulse, rampa... Para una excitación cualquiera aplicar los comandos [lsim](#), [gensig](#).
- 3) Visualizar los resultados obtenidos. A continuación exponemos como obtener las características de tiempo de respuesta, tiempo de subida, sobreimpulso... de un sistema:



Una vez hemos obtenido la respuesta del sistema hacemos un doble clic sobre la figura o bien haciendo un clic derecho, seleccionamos Properties...

De esta forma accederemos a un cuadro de dialogo de propiedades del gráfico.

En este cuadro se puede cambiar:

-Etiquetas de dibujo: El título, etiquetas en x y en y.

-Límites: Podemos poner autoescalado o introducir una nueva escala para x e y.

-Estilo: Aquí podemos activar la rejilla (show grid), cambiar las propiedades de la fuente y los colores de la gráfica.

Pero la propiedad más interesante son las características de la respuesta. Esta opción sólo está disponible cuando tratamos con [step](#) o [impulse](#). Las opciones son:

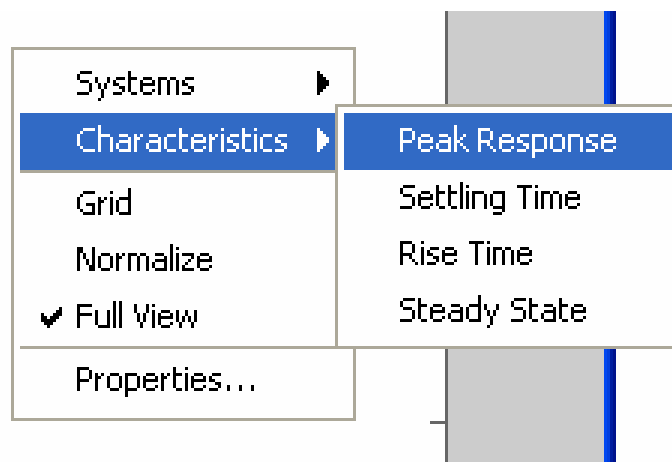
- Peak Response: Muestra el tiempo de pico del sistema.

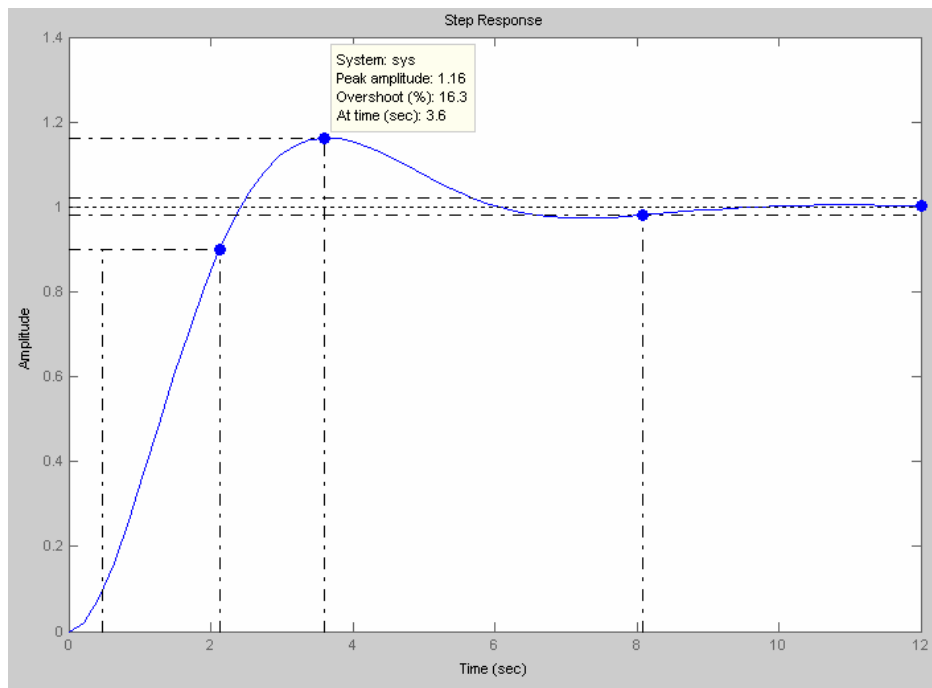
- Settling time: Muestra el tiempo de respuesta. Por defecto da el tiempo de respuesta al 2 %, es decir cuando la respuesta llega al 98 % del valor final. Aquí podemos especificar otros porcentajes.

- Rise Time: Muestra el tiempo de subida. Por defecto está establecido entre el 10 % y el 90 %, pudiendo cambiar este valor.

- Steady state: Muestra el estado estable de la respuesta. (el valor final).

Para la respuesta impulsional solo disponemos de peak response y settling time.



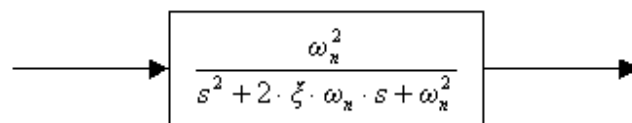


Una vez hemos marcado las características que nos interesa destacar, veremos como aparecen unos puntos en la respuesta. Al acercarnos con el cursor veremos una etiqueta con las propiedades de ese punto. Para mantener estas etiquetas hacer un clic en ellas.

Por ejemplo para el caso de la figura, la etiqueta del tiempo de pico nos muestra, el instante en que se produce (3.6), la amplitud de pico (1.16) y el sobreimpulso (16.3 %).

#### Ejemplo 17:

Dado un sistema de segundo orden, realizar unas gráficas en 3D y 2D donde se vea claramente la diferencia entre un sistema sin amortiguamiento, subamortiguado, críticamente amortiguado y sobreamortiguado



```
%-----
% Ejemplo 17
%Sistemas de control dinámico con MATLAB
%En este ejemplo variaremos el parámetro zeta en un sist.de 2ºorden
%-----
clear all;

%Declaración de los parámetros:
```



```

wn=1;%la frecuencia será fija
zeta=[0 0.5 1 2];% Variaremos zeta
t=0:15/200:(15-15/200);

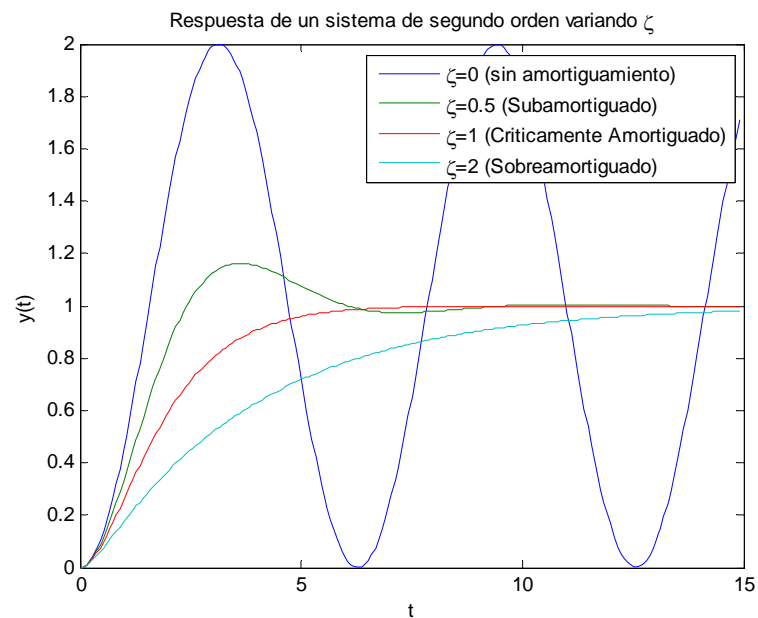
%Por cada parámetro zeta almacenaremos la respuesta
%del sistema en una matriz y:
for n=1:length(zeta)
    num=wn^2;
    den=[1 2*zeta(n)*wn wn^2];
    W=tf(num,den);
    y(:,n)=step(W,t);
end;

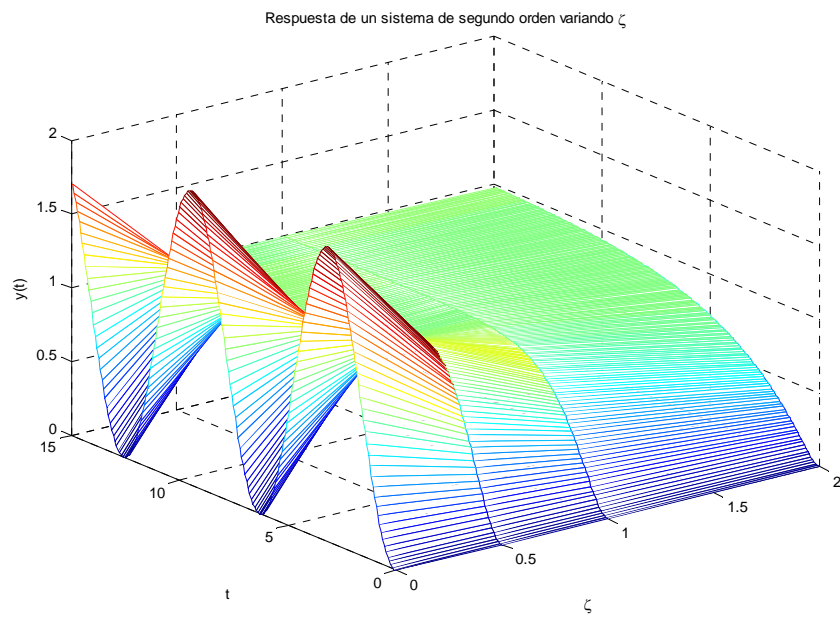
%Representación en 3D de la respuesta
figure(1);
mesh(zeta,t,y);
xlabel('\zeta');
ylabel('t');
zlabel('y(t)');
title('Respuesta de un sistema de segundo orden variando \zeta');

%Representación en 2D de la respuesta
figure(2);
plot(t,y);xlabel('t');ylabel('y(t)');
title('Respuesta de un sistema de segundo orden variando \zeta');
legend('\zeta=0 (sin amortiguamiento)',...
'\zeta=0.5 (Subamortiguado)',...
'\zeta=1 (Críticamente Amortiguado)',...
'\zeta=2 (Sobreamortiguado)');

```

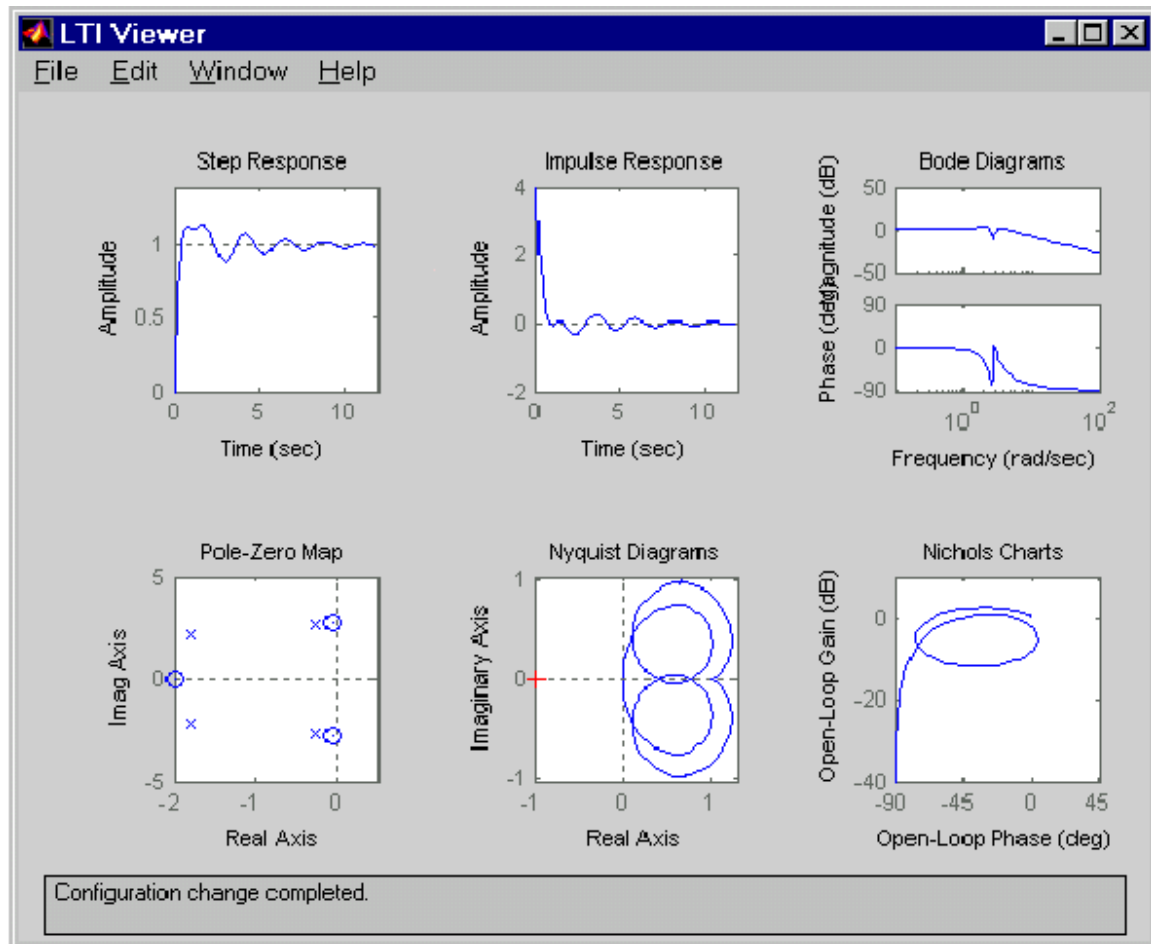
---





## Análisis con LTI Viewer

El LTI Viewer es una interfaz gráfica de usuario (GUI) para analizar la respuesta de sistemas, ya sea de forma temporal o frecuencial. Incluye respuestas ante escalón unitario ([step](#)), impulso ([impulse](#)), [Bode](#), [Nyquist](#), Nichols, [mapa de polos](#), [lsim](#) y gráficos de condiciones iniciales (initial plots).



Para inicializar LTI viewer para analizar la respuesta de un sistema LTI, usamos el comando `ltiview`. La sintaxis es la siguiente:

```
>>ltiview
```

Cuando invocamos la orden sin argumentos inicializamos directamente LTI Viewer para el análisis de la respuesta en sistemas LTI

```
>>ltiview(sys1,sys2,...,sysn)
```

De esta forma se abre LTI Viewer conteniendo la respuesta escalón unitario (step) de los modelos `sys1`, `sys2`, ..., `sysn`. También es posible especificar un color y estilo de línea para cada sistema.

```
>>ltiview('tipo de gráfica',sys)
```

De esta forma podemos especificar que tipo de respuesta queremos ver. El tipo de gráfica solo puede ser '[step](#)', '[impulse](#)', 'initial', '[lsim](#)', '[pzmap](#)', '[bode](#)', '[nyquist](#)', 'nichols' y 'sigma'. También podemos escribir un vector de strings para indicar varios tipos a la vez.  
Como por ejemplo: `ltiview({'step';'impulse'},sys)`

```
>>ltiview('tipo de gráfica',sys,extras)
```

De esta forma podemos introducir datos adicionales de entrada soportados por las funciones de respuesta. Por ejemplo: `ltiview('lsim',sys1,sys2,u,t,x0)`

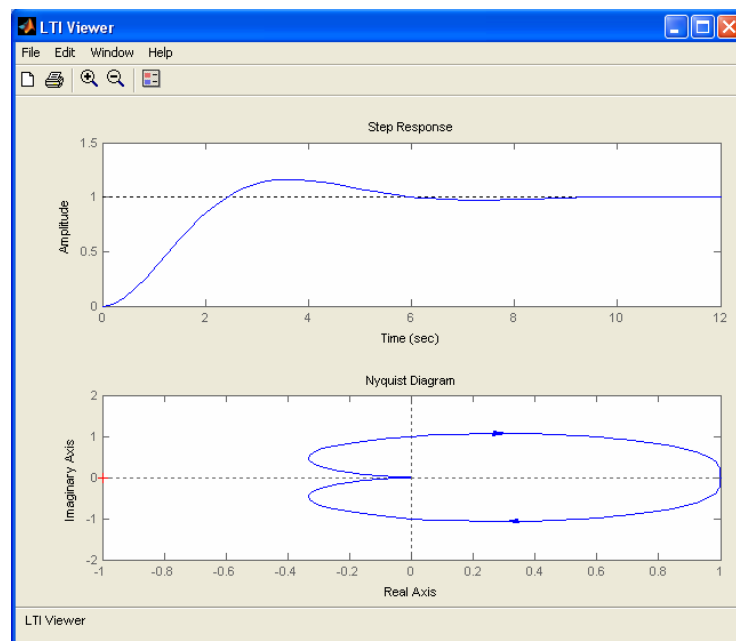
En el siguiente viewlet veremos las principales herramientas que nos aporta el LTI Viewer para el análisis temporal.

A continuación algunos ejemplos de la forma de usar LTIVIEW

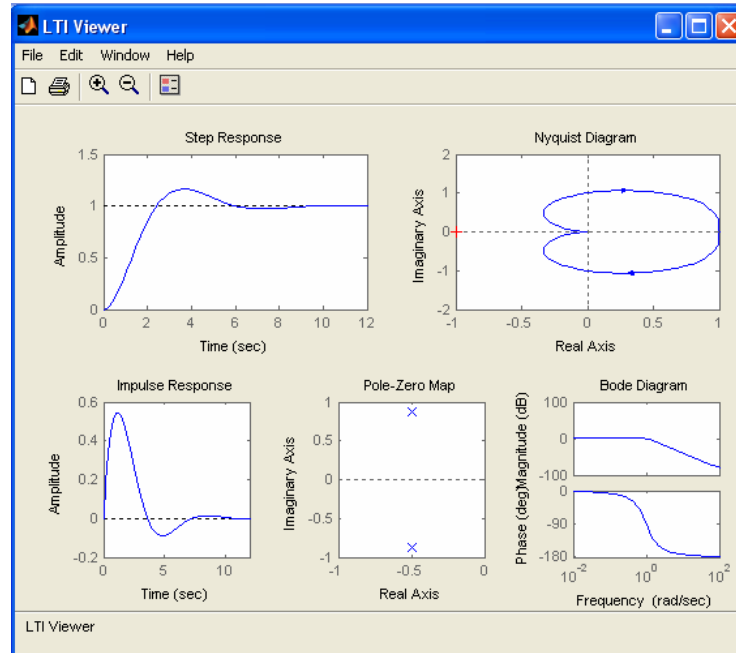
Primero definimos 3 funciones de transferencia cualquiera

---

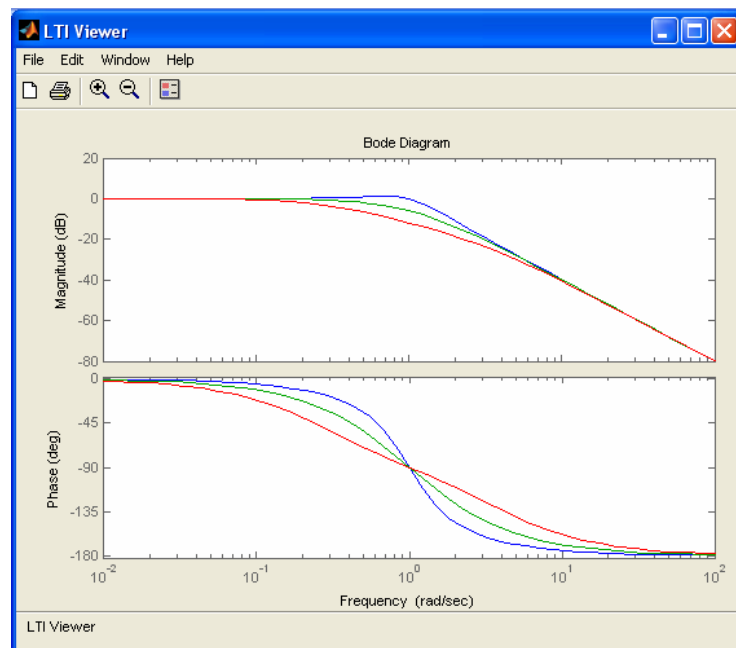
```
>> ltiview({'step';'nyquist'},sys)
```



```
>> ltiview({'step';'nyquist';'impulse';'pzmap';'bode';},sys)
```



```
>> ltiview('bode',sys,sys1,sys2)
```



# ANALISIS DEL LUGAR GEOMETRICO DE LAS RAICES

## Influencia en las modificaciones en la estructura de polos y ceros:

Para sistemas lineales el requerimiento de estabilidad se puede definir en términos de los polos de la función de transferencia en lazo cerrado. Los polos son las raíces del polinomio del denominador y los ceros las raíces del numerador de la función de transferencia. En esta sección veremos la influencia de los polos y ceros sobre el sistema, así como su representación.

## Influencia de los polos y ceros en un sistema genérico

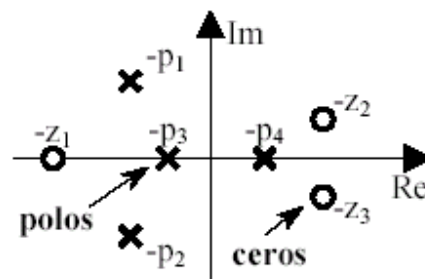
Conclusiones:

- El numerador no influye en el tipo de respuesta, esto viene determinado por el denominador.
- Ceros ( $B_i$ ,  $C_i$  y  $D_i$ ): Influyen en la amplitud de la respuesta, no en la forma.
- Polos ( $P_i$ ): Influye en el tiempo exponencial de caída de la función
- Los polos complejos influyen en la amortiguación de la respuesta

## Influencia de polos y ceros: Estabilidad

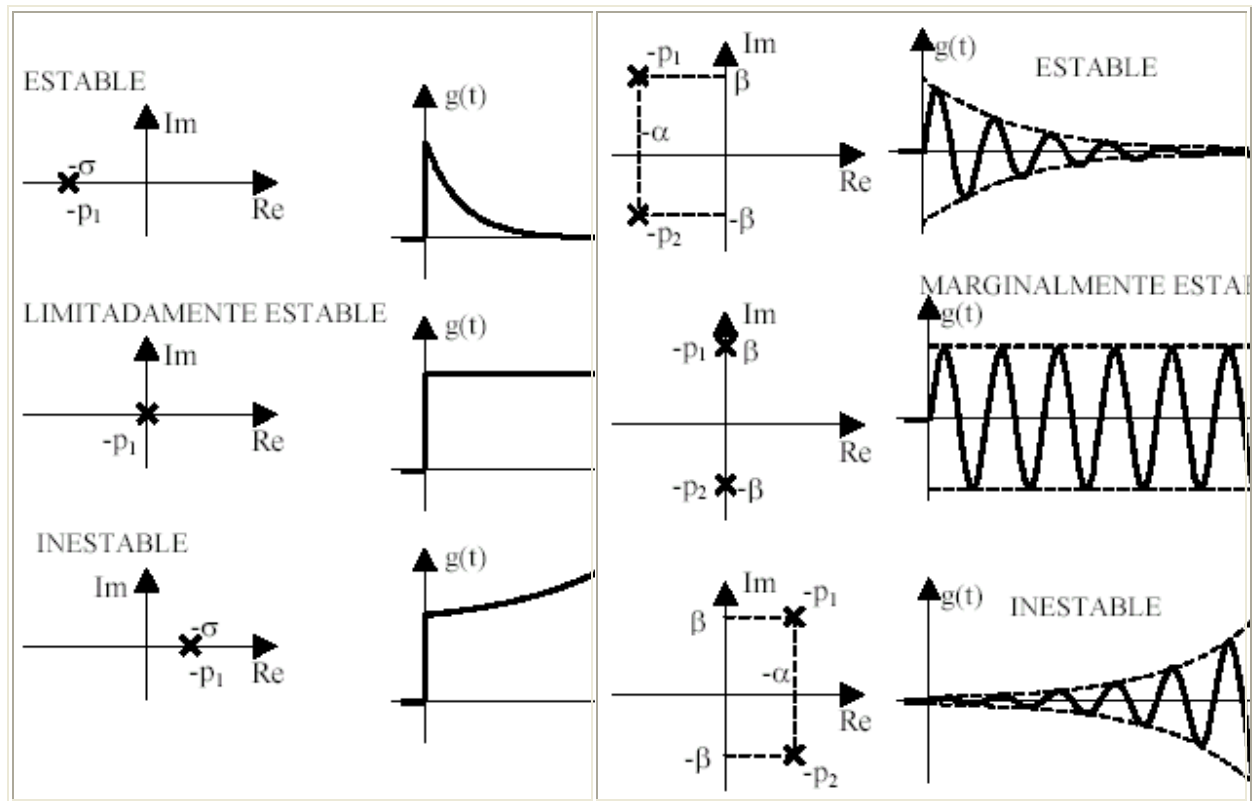
$$W(s) = \frac{\prod_{j=1}^m (s + z_j)}{\prod_{i=1}^n (s + p_i)} ;$$

$n \geq m$   $\left\{ \begin{array}{l} -z_j \text{ son las raíces del numerador (ceros)} \\ -p_i \text{ son las raíces del denominador (polos)} \end{array} \right.$   
 $z_j, p_i \in \mathbb{C}$



Según la posición de los polos podemos determinar si un sistema es estable o no:

- Un sistema es **estable** si todos sus polos están situados en el semiplano complejo negativo.
- Un sistema es **inestable** si algún polo está situado en el semiplano complejo positivo o si existen polos múltiples en el eje imaginario o en el origen.
- Un sistema es **limitadamente estable** si existe un solo polo en el origen, estando los demás situados en el semiplano negativo.
- Un sistema es **marginalmente estable** si existe una pareja simple (no múltiples) de polos complejos conjugados sobre el eje imaginario, estando los restantes polos situados en el semiplano negativo.



### Dibujar los polos y ceros de un modelo LTI

Para representar gráficamente los polos y ceros de un sistema o modelo LTI usaremos el comando pzmap. La sintaxis de este comando es:

```
>> pzmap(sys)
```

Dibuja un mapa de los polos y ceros del sistema sys. El sistema LTI puede ser discreto o continuo. Los polos son representados por x y los ceros por o.

```
>> pzmap(sys1,sys2,...,sysN)
```

Dibuja el mapa de polos y ceros para varios sistemas LTI en la misma figura. Los modelos LTI pueden tener diferentes entradas y salidas, ser continuos o discretos.

```
>> [p,z]=pzmap(sys)
```

Al invocar los argumentos de esta forma, se almacena los polos y ceros del sistema en los vectores p y z. En este caso no se realiza la representación gráfica.

Existen lo siguientes comandos relacionados:

```
>>p=pole(sys)
```

De esta forma se calcula los polos del sistema sys, pudiendo ser SISO o MIMO, y se almacenan en la variable p

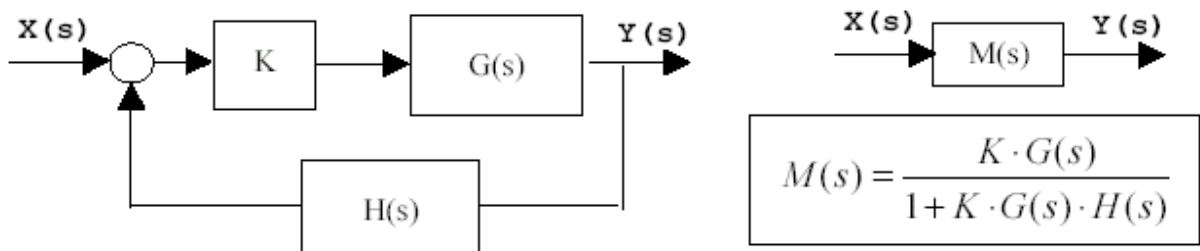
```
>>z=zero(sys)
>>[z, ganancia]=zero(sys)
```

Mediante este comando calcularemos los ceros de un sistema sys y los almacenaremos en la variable z. También podemos obtener la ganancia, añadiendo otro parámetro.

## Cálculo del lugar de la raíces con MATLAB

El toolbox de control de MATLAB también dispone de comandos para el cálculo del lugar de las raíces. Las instrucciones que utilizaremos serán rlocus y rlocfind.

Normalmente partimos de un sistema como el siguiente:



El lugar de las raíces nos muestra la evolución de los polos del sistema realimentado o en cadena cerrada  $M(s)$  cuando el parámetro  $K$  varía desde cero hasta infinito.

Las instrucción rlocus permite obtener este trazado, y utiliza como argumento el sistema en cadena abierta definido por la función de transferencia  $G(s)H(s)$

### Sintaxis:

```
>>rlocus(sys);
>>rlocus(sys,k);
>>rlocus(sys1,'r',sys2,'y:',sys3,'gx',...)
```

Calcula y dibuja el lugar de las raíces de un sistema SISO en cadena abierta cuando el parámetro  $k$  varia desde cero hasta infinito. Mediante  $k$  podemos indicar al sistema los valores para los cuales serán calculados los polos. Con la tercera instrucción podemos dibujar el lugar de las raíces en una misma gráfica. Como en todos los gráficos podemos especificar un color, estilo de línea, y una marca para cada modelo.

```
>>[r,k] = rlocus(sys)
>>r = rlocus(sys,k)
```



Cuando utilizamos estos argumentos, `r` será una matriz que contiene las localizaciones de las raíces complejas. Cada fila de la matriz corresponde a la ganancia del vector `K`.

#### Ejemplo 20:

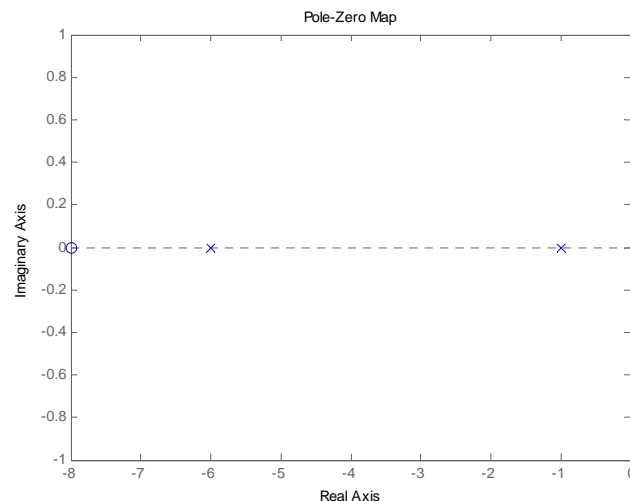
Un sistema tiene la siguiente función de transferencia en lazo abierto,

$$H(s) = \frac{s+8}{s^2+7s+6}$$
 Representar los polos y ceros de  $H(s)$  y almacenarlos en variables.

---

```
%-----  
% Sistemas de control dinámico con MATLAB  
% En este ejemplo veremos como tratar los polos y ceros de un sistema  
%-----  
  
% Definición del sistema  
  
H=tf([1, 8],[1, 7, 6]);  
  
%Representación de los polos y ceros  
pzmap(H)  
  
% Tambien seria posible mediante ltiviewer  
% ltiview({'pzmap'},H)  
  
% Almacenar la respuesta  
  
p=pole(H); % determina los polos de H(s)  
z=zero(H); % determina los ceros de H(s)
```

---



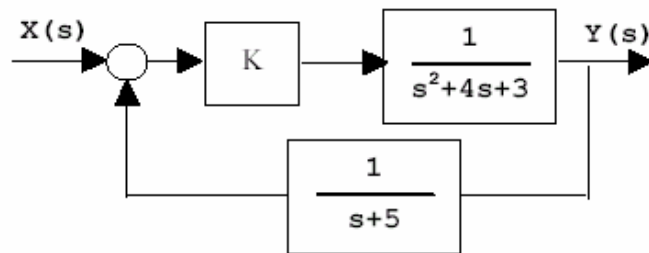
### Ejemplo 21:

Según el diagrama de la figura,

a) Obtener el lugar de las raíces

b) Determinar el valor de K, a partir del cual:

- el sistema es inestable
- el sistema presenta sobreoscilación para una entrada escalón unitario



a)

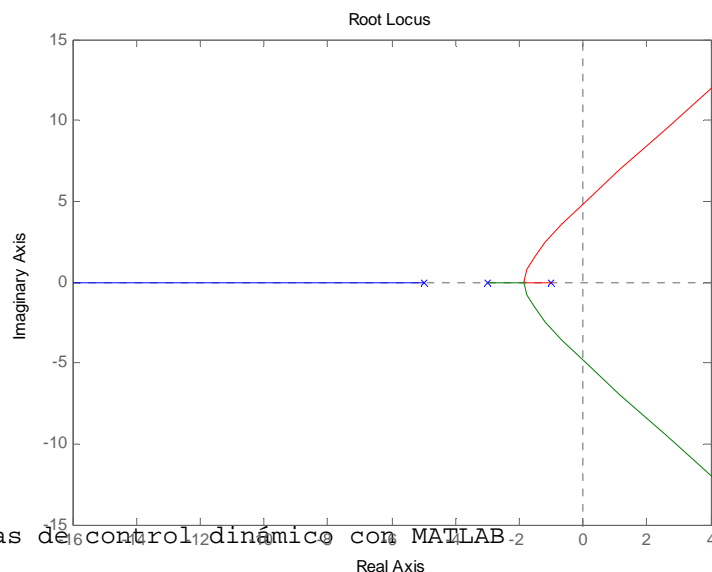
El primer paso es definir las funciones de transferencia  $G(s)$ ,  $H(s)$  y  $G(s) \cdot H(s)$ . Llamaremos a estas funciones `sis_g`, `sis_h` y `sis_gh`, respectivamente. A continuación utilizaremos el sistema `sis_gh` recién creado como argumento para la `inst`

---

```
%-----
% Sistemas de control dinámico con MATLAB
% En este ejemplo veremos lugar geometrico de la raiz
%-----

sis_g=tf(1,[1 4 3]);
sis_h=tf(1,[1 5]);
sis_gh=series(sis_g,sis_h);
rlocus(sis_gh)
```

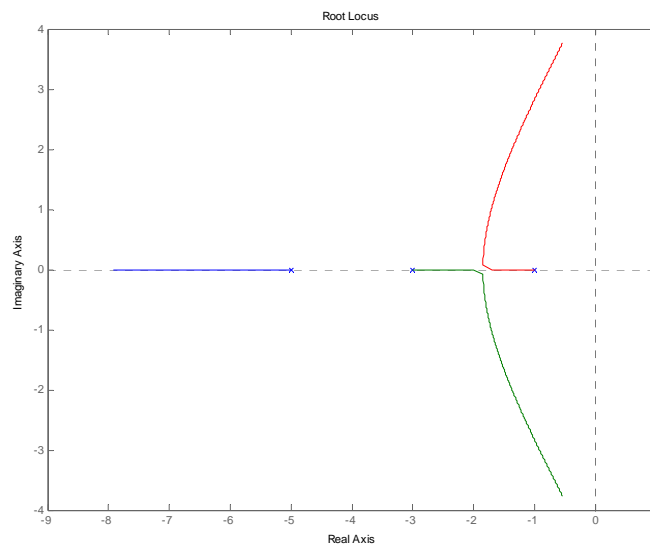
---



Interpretar el gráfico resultante es sencillo: muestra la situación en el plano complejo de los polos del sistema realimentado o en cadena cerrada  $W(s)$ . Cada rama representa la situación de uno de los polos; en este caso aparecen tres ramas dibujadas con tres colores distintos para mayor claridad. Los puntos de comienzo ( $K=0$ ) de cada rama coinciden con los polos en cadena abierta (cruces sobre el gráfico) y puntos de finalización ( $K=\infty$ ) de cada rama tienden a infinito en este caso.

Si no se añade ningún parámetro extra, MATLAB elegirá automáticamente los valores de  $K$  entre 0 e infinito para los cuales calculará el lugar de las raíces. En determinadas ocasiones interesa elegir manualmente el rango de valores deseado para  $K$ . Para ello basta con introducir un nuevo parámetro en `rlocus`:

```
» rlocus(sis_gh, [0:.1:100]) %K de 0 a 100 a intervalos de 0.1
```



Si no se desea un resultado gráfico, sino que se desea conocer los valores numéricos de los polos en cadena cerrada para cada valor de  $K$  la instrucción a teclear será:

```
[r, k] = rlocus(sis_gh);
```

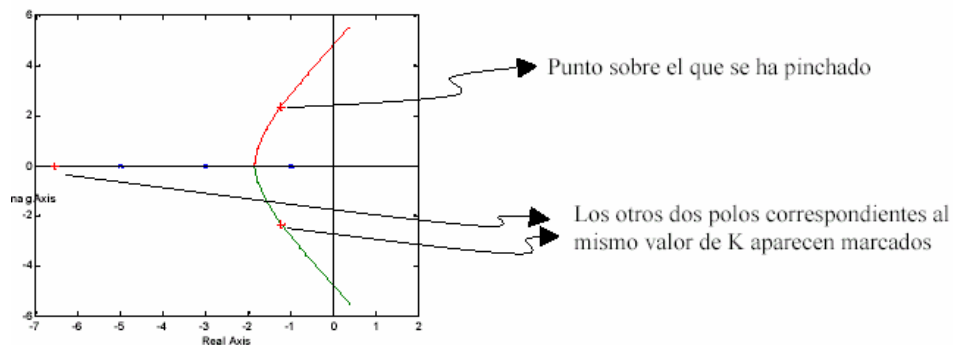
La variable `k` contendrá los valores del parámetro  $K$  utilizados para el cálculo del lugar de las raíces; la variable `r` contendrá los polos del sistema para cada valor de  $K$ .

También es posible comprobar sobre el propio gráfico los valores del parámetro  $K$  correspondientes a cada punto del lugar de las raíces. Para ello se emplea la instrucción `rlocfind`. Esta instrucción, ejecutada a continuación de `rlocus`, permite pinchar con el ratón sobre un punto cualquiera del lugar de las raíces y obtener el valor del polo más cercano al punto donde se ha pinchado, el

valor de K correspondiente a ese polo y la situación del resto de polos para ese valor de K (aparecen marcados en rojo sobre el diagrama):

```
» rlocus(sis_gh)
» rlocfind(sis_gh)
Select a point in the graphics window
```

A continuación se debe pinchar con el ratón sobre un punto cualquiera del lugar de las raíces:



La respuesta que aparece en la ventana de comandos indica el valor de s en el punto del lugar de las raíces donde se ha pinchado (selected point) y el valor de K correspondiente (ans):

```
Select a point in the graphics window
```

```
selected_point =

    -1.4405 + 1.7845i

ans =

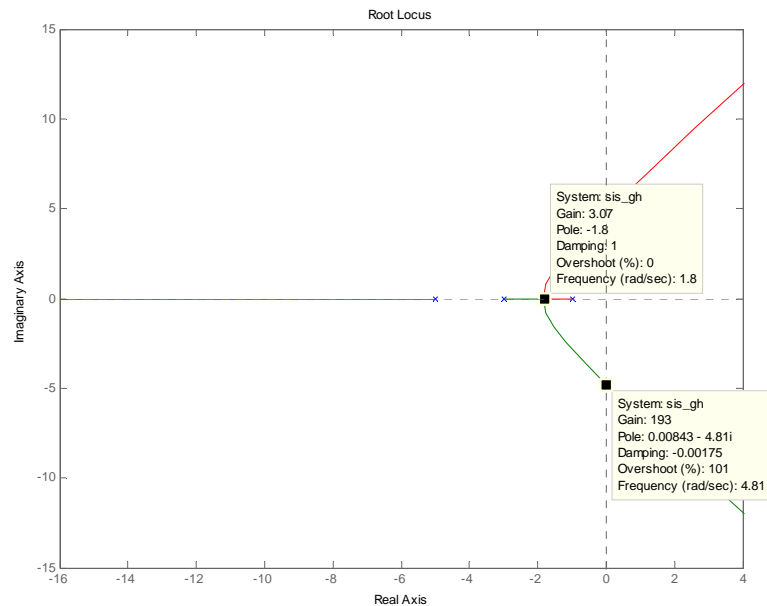
    17.3446
```

Tal y como indica MATLAB, en este caso el punto donde se ha pinchado es  $s = -1.4405 + 1.7845i$  y el valor de K para el cual el sistema presenta ese polo es  $K = 17.3446$ .

b)

Según lo visto en la teoría, el sistema será inestable cuando sus polos estén situados en el semiplano complejo positivo. También cuando empiecen aparecer los polos complejos, el sistema presentara una sobreoscilación. Para ello clicamos sobre la gráfica para encontrar el valor de K. Como vemos en la figura a parte del valor de la ganancia y el polo, también nos indica otros datos como la frecuencia del sistema y la sobreoscilación. Del estudio de los puntos encontramos:

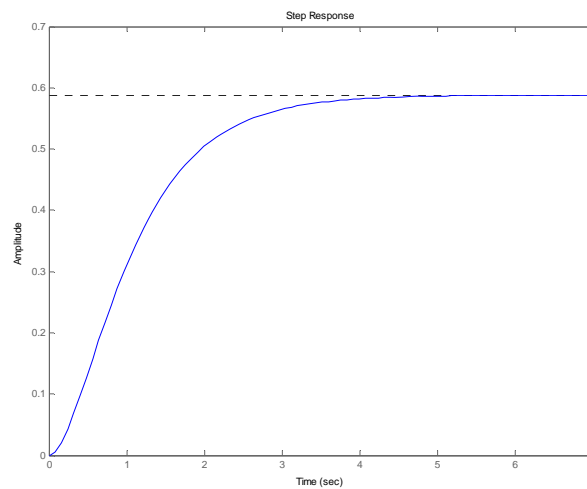
- Punto a partir del cual aparecen los polos complejos:  $K=3.08$
- Punto a partir del cual aparecen los polos inestables:  $K=192$



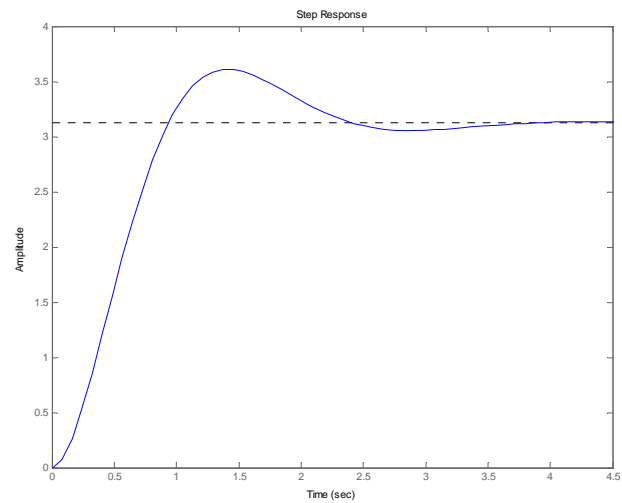
La forma de comprobar este comportamiento es asignar valores a  $K$  y comprobar mediante la instrucción `step` la respuesta a escalón del sistema realimentado  $W(s)$ . La forma de calcular el sistema realimentado  $W(s)$  será mediante la instrucción `feedback` vista anteriormente.

Probaremos en primer lugar un valor de  $K=2$ , de modo que los polos sean reales. En el gráfico de la respuesta a escalón obtenido se debe comprobar cómo no existe sobreoscilación.

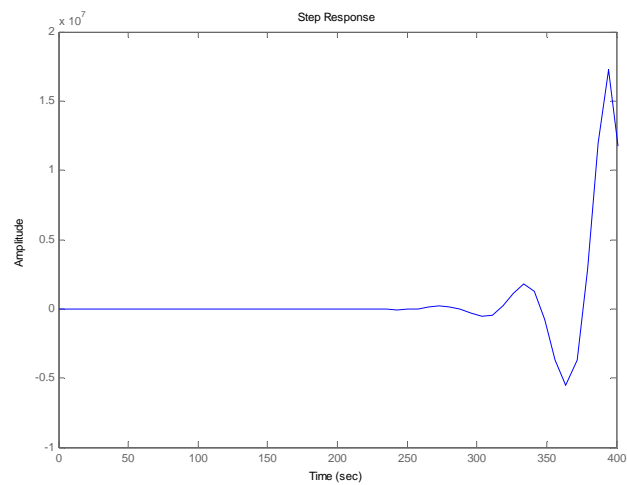
```
>> k=2;
>> sis_w=feedback(k*sis_g,sis_h);
>> step(sis_w)
```



```
>> k=25;
>> sis_w=feedback(k*sis_g,sis_h);
>> step(sis_w)
```



```
>> k=200;
>> sis_w=feedback(k*sis_g,sis_h);
>> step(sis_w)
```



## Respuesta en frecuencia

Cuando aplicamos a un sistema una entrada senoidal, la salida también será senoidal y de la misma frecuencia. La salida puede diferir de la entrada en amplitud y fase. En esta sección se verán las transformaciones necesarias en la FdT para adaptarla a entradas en frecuencia.

## Respuesta frecuencial con MATLAB

Ejemplo 26:

Dada la siguiente función de transferencia  $W(s) = \frac{100}{s + 40}$ ,

a) Encontrar con MATLAB la ecuación de la respuesta ante una entrada

$$x(t) = 5 * \sqrt{2} * \cos(30)$$

b) Representar la entrada y la salida en una gráfica.

---

a)

Lo primero que debemos hacer es sustituir  $s$  por  $j\omega$ , en la función de transferencia:

$$W(j\omega) = \frac{100}{j\omega + 40} = G \angle \varphi$$

La entrada del sistema es del tipo  $x(t) = A \cos(\omega t)$ . En nuestro caso .  
 $\omega = 30 \text{ rad/s}$

sustituimos en la FdT isócrona y mediante MATLAB obtenemos la ganancia y la fase Introducimos la FdT como un número complejo  $fdt = (100 / (j * 30 + 40))$ . Con la instrucción `abs` calculamos la ganancia y con `angle` el desfase:

```
%-----  
%Sistemas de control dinámico con MATLAB  
% En este ejemplo veremos como obtener una respuesta en frecuencia y su  
%representación  
%-----  
  
%Lo primero es substituir en la FdT s por jw
```

```

fdt=(100/(j*30+40)) %FdT isócrona para w=30

%Obtención de la ganancia y el desfase
ganancia=abs(fdt)
desfase=angle(fdt)

%Representacion de la respuesta

sys=tf([100],[1 40]); %Definición del sistema

t=0:0.001:2; %Vector de tiempo
u = 5*sqrt(2)*cos(30*t); %Entrada del sistema

[y,x] = lsim(sys,u,t); %Calculo de la respuesta

plot(t,y,t,u) %Representación entrada-salida
legend('Entrada','Salida');
%Comprobación de la ganancia del sistema:
g=max(y)/max(u);

```

---

```
fdt = 1.6000 - 1.2000i
```

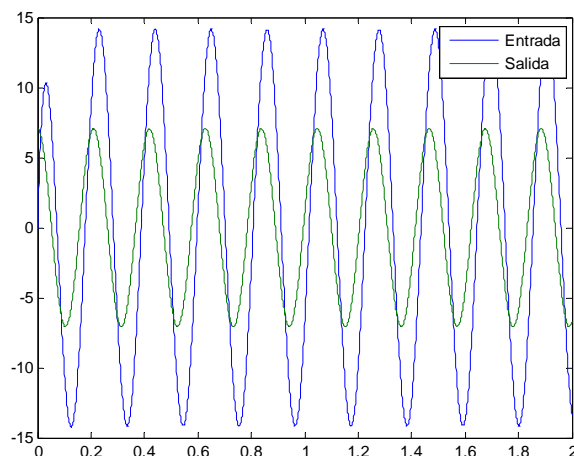
```
ganancia = 2
```

```
desfase = -0.6435
```

Según lo obtenido la ecuación de la salida es:  $y(t) = 2*5\sqrt{2} * \cos(30t - 0.643)$

b)

Para la representación de la respuesta usaremos el comando lsim. Primero creamos el modelo tf de manera habitual. Definimos la ecuación de la entrada y el vector de tiempo. Finalmente representamos la entrada y la salida con plot.





## DIAGRAMA DE BODE CON MATLAB

Ejemplo 27:

$$\text{sys1} = \frac{10(s+1)}{(s+2)(s+5)} \quad \text{sys2} = \frac{1}{(s^2+1)}$$

Según estas FdTs hacer:

a) Diagrama de Bode de sys1

b) Diagrama de Bode de sys2 con un rango de frecuencias entre  $[10^{-1}, 10^1]$  y con 101 puntos de frecuencia. ¿Qué sucede? ¿Cómo podemos solucionarlo?

c) Diagrama de Bode de los dos sistemas, comparándolos.

---

```
%-----  
%Sistemas de control dinámico con MATLAB  
% Ejemplo: Uso del comando bode  
%-----  
  
% Creación de los modelos LTI  
s=tf('s');  
sys1=(10*(s+1))/((s+2)*(s+5));  
sys2=1/(s^2+1)  
  
% Diagrama de Bode del primer sistema  
bode(sys1);  
  
% Diagrama de Bode del segundo sistema  
figure  
w=logspace(-1,1,101);%intervalo de frecuencias  
bode(sys2,w);  
  
figure  
bode(sys2)  
  
% Comparación de los sistemas  
figure  
bode(sys1, 'o', sys2, 'r--');
```

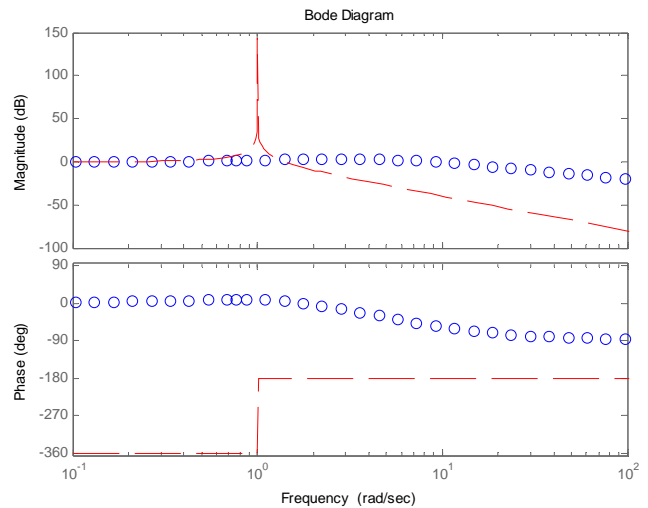
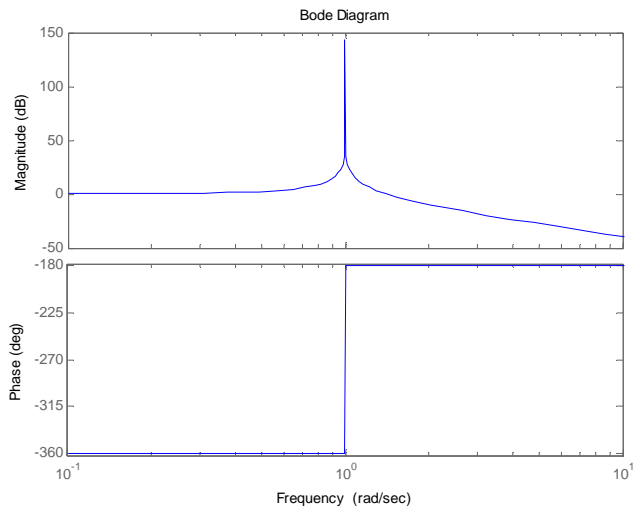
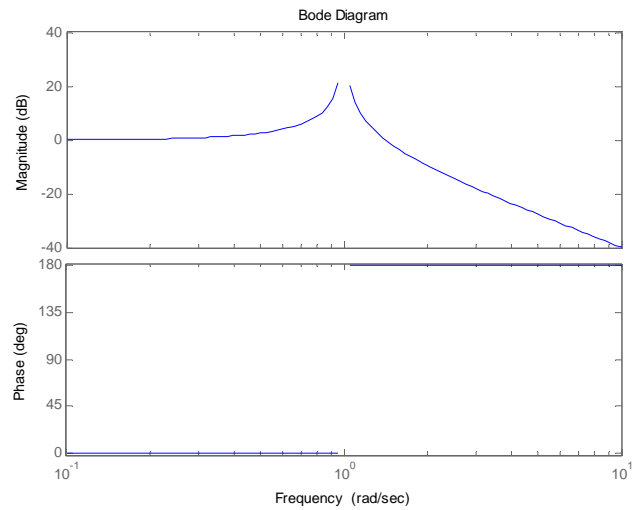
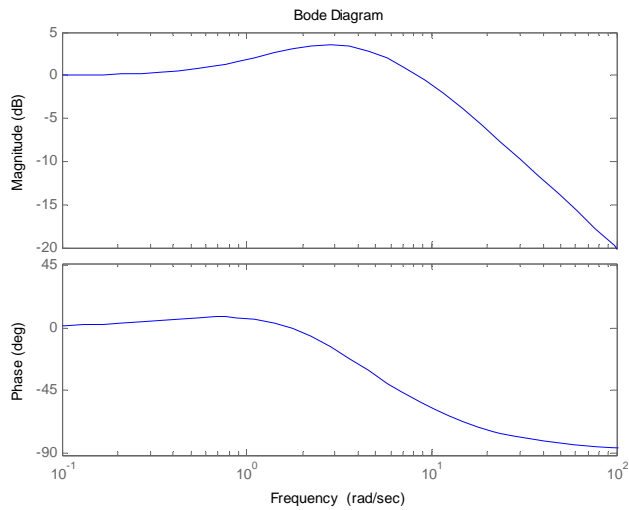
---

Transfer function:

$$\frac{1}{s^2 + 1}$$

a)

Como siempre creamos el modelo de la manera habitual y ejecutamos el comando `bode`. También podríamos ejecutar el comando `ltiview('bode',sys1)` para representar el diagrama con la herramienta LTIVIEWER.



## DIAGRAMA DE NYQUIST CON MATLAB

Representa el diagrama de Nyquist para la siguiente función de transferencia:

$$H(s) = \frac{2s^2 + 5s + 1}{s^2 + 2s + 3}$$

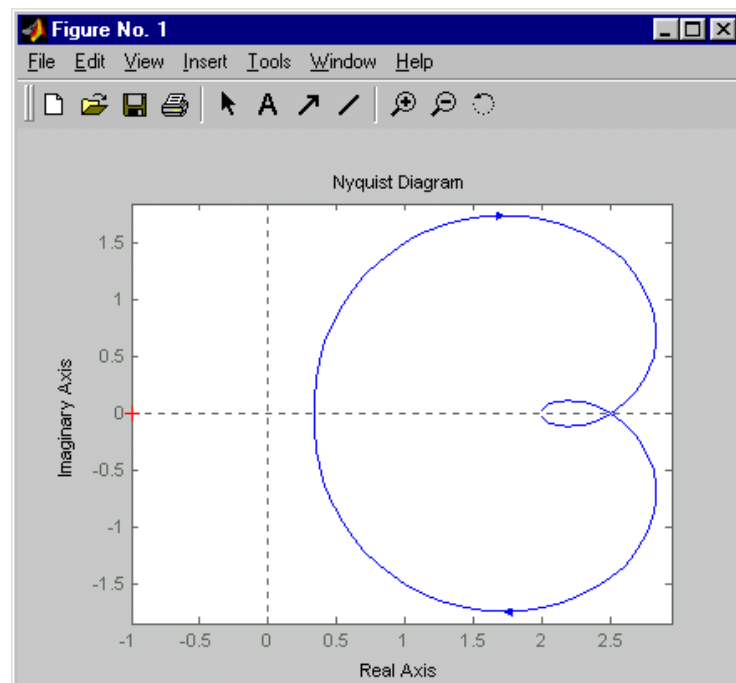
Primero se define el sistema mediante el comando `tf` y luego aplicamos el comando `nyquist`:

```
num=[2 5 1]  
den=[1 2 3]
```

```
H = tf(num,den)
```

```
nyquist(H)
```

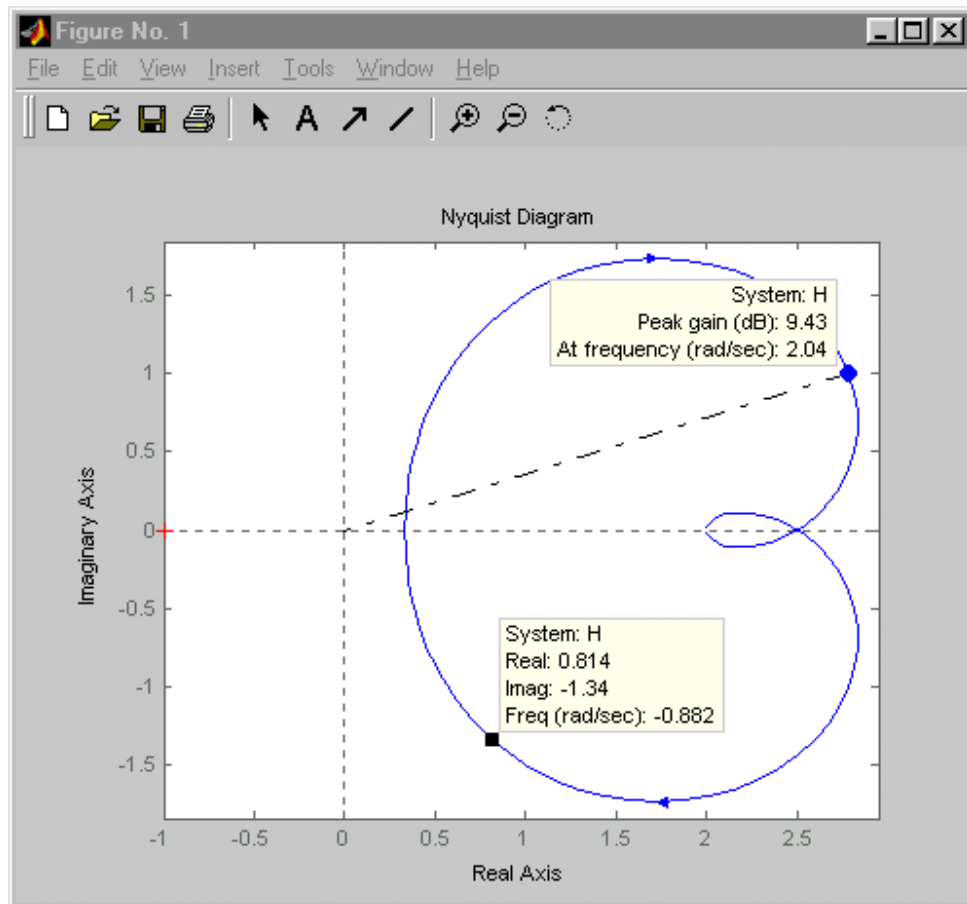
Resultando el diagrama de nyquist de la siguiente figura:



En la opción Zoom del menú contextual del gráfico existen dos opciones de zoom específicas para los diagramas de Nyquist:

- Tight: Estrecha la gráfica para poder ver todas las ramas que forman el Nyquist
- On (-1,0): Hace un zoom alrededor del punto critico (-1,0)

Como características del menú contextual tenemos la respuesta de pico, donde nos mostrará la ganancia y la frecuencia. También si clicamos en cualquier lugar de la curva veremos la frecuencia en ese punto y los valores real y imaginario:



## Estudio de estabilidad, márgenes de fase y ganancia

### Ejemplo 28:

Dado un sistema realimentado con la siguiente función de transferencia en lazo

abierto:  $T(s) = \frac{K(s+1)(s+5)}{s(s^2-12s+400)}$ . Realizar:

- a) Para K=20 representar el diagrama de Bode y Nyquist
- b) Estudiar la estabilidad del sistema. Margen de ganancia y fase.
- c) En el caso crítico de auto oscilación, encontrar el valor de K y la frecuencia de oscilación crítica.

---

a)

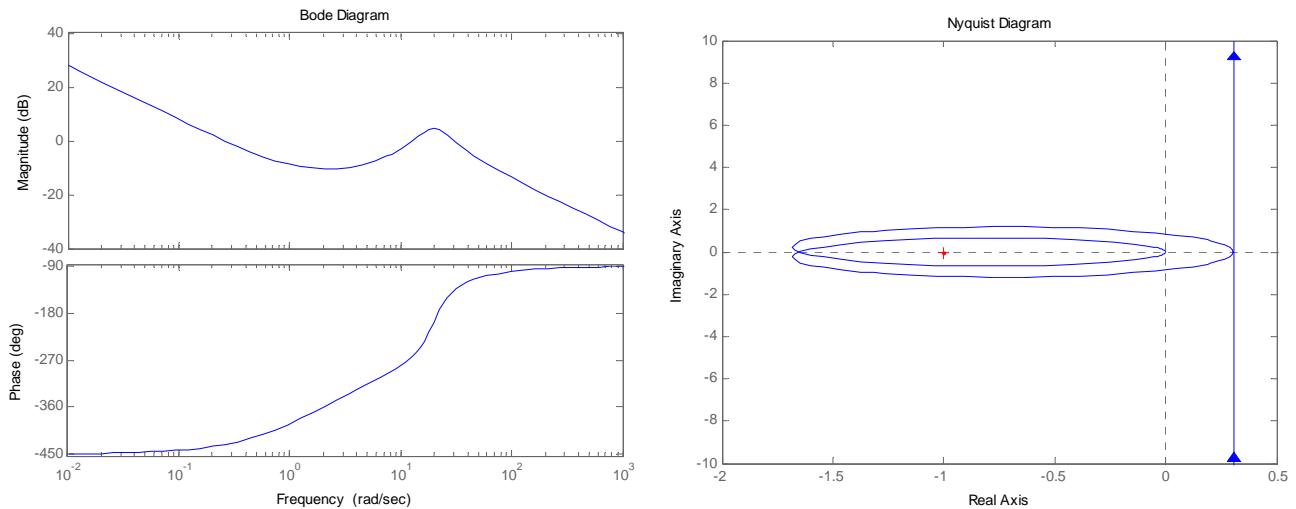
Lo primero que tenemos que hacer es definir la estructura básica del sistema, con k=1. Lo llamaremos T1. Para este caso, K=20, multiplicamos T1 por 20. Para realizar los diagramas utilizaremos los comandos bode y nyquist:

```
%-----  
% Sistemas de control dinámico con MATLAB  
% Ejemplo: En este ejemplo realizaremos un estudio de estabilidad basándonos  
% en el criterio de Nyquist  
%-----  
%Determinación del sistema como objeto LTI  
s=tf('s');  
T1=((s+1)*(s+5))/(s*(s^2-12*s+400));  
  
%A) Curva de Bode y polar para k=20  
  
T=20*T1  
figure  
bode(T);  
figure  
nyquist(T);  
  
%B) Estabilidad y márgenes de fase/ganancia  
  
figure  
pzmap(T);  
S = ALLMARGIN(T);
```

```
%C) Encontrar K

figure
rlocus(T1);
K=abs(rlocfind(T1));
%msgbox(K,'VALOR DE K');
```

---



Para ver el Nyquist correctamente hay que hacer un zoom alrededor del punto  $(-1,0)$ , esta es una opción que se encuentra dentro del menú contextual del gráfico (clic derecho)

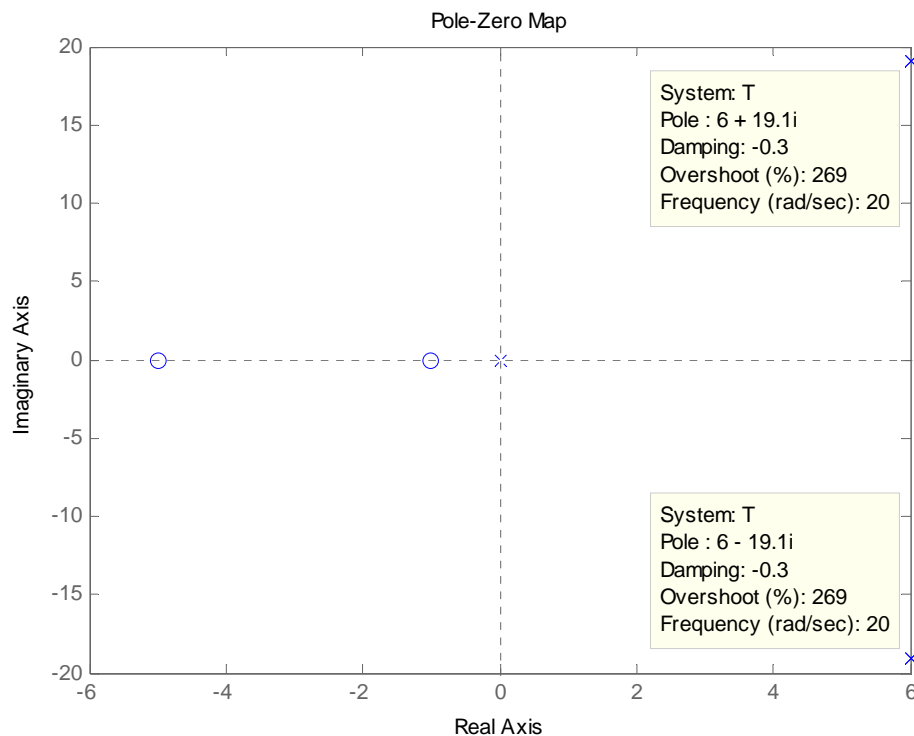
b)

Para realizar el estudio de estabilidad del sistema en lazo cerrado nos basaremos en el criterio de estabilidad de Nyquist. Este criterio dice:

$$MV(T[j\omega], (-1,0j)) = p$$

Las medias vueltas que realiza, se pueden observar directamente del diagrama de Nyquist trazado, en nuestro caso  $M.V.=2$  (Se analiza la rama donde las frecuencias van desde cero hasta infinito)

Ahora tenemos que definir  $p$ , es decir el número de polos de  $T$  situados en el semiplano derecho. Para ver la situación de los polos usaremos el comando **pzmap**, resultando:



Tal como se ve en el mapa de polos y cero, este sistema tiene dos polos en el semiplano derecho, por tanto  $p=2$

En definitiva tenemos que el sistema da dos medias vueltas y tiene 2 polos con raíces reales positivas, por tanto cumple el criterio de Nyquist y el sistema es estable para  $K=20$

Ahora pasaremos a calcular los márgenes de ganancia y fase. Para calcular esto aplicaremos la sentencia: **S = ALLMARGIN(T)**. De esta forma S contendrá los datos más importantes del sistema

```
>> S
```

```
S =
```

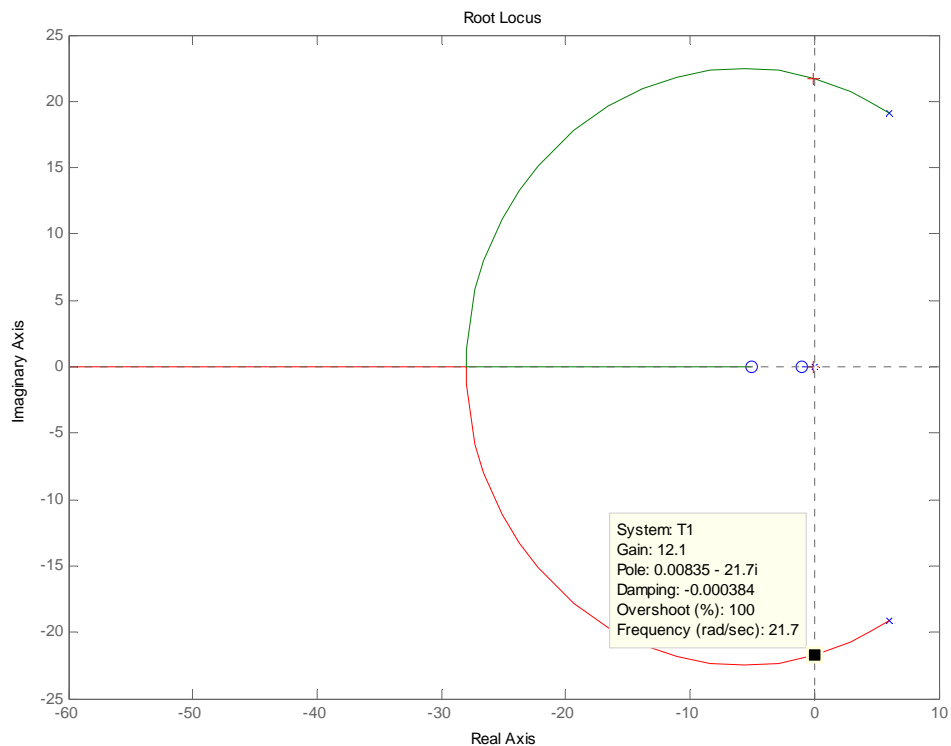
```
GainMargin: 0.6064
GMFrequency: 21.7438
PhaseMargin: [107.9047 -81.5583 42.3142]
PMFrequency: [0.2586 12.9775 29.7972]
DelayMargin: [7.2825 0.3745 0.0248]
DMFrequency: [0.2586 12.9775 29.7972]
Stable: 1
```

Tal como se ve en la figura, el margen de ganancia del sistema es de 0.6064, y el margen de fase 107.9047. (Salen más valores porque MATLAB realiza el estudio para el rango de frecuencia entre menos infinito y infinito)

Además de los márgenes, nos da más datos como la frecuencia de oscilación  $w = 21.7438 \text{ rad/s}$   
También nos confirma lo que nosotros ya sabíamos, que el sistema es estable en lazo cerrado. ( 1 es estable, 0 es inestable)

c)

En el apartado anterior obtuvimos la frecuencia de oscilación del sistema ( $w = 21.74 \text{ rad/s}$ ). Para encontrar el valor de K ejecutaremos la instrucción rlocus sobre T1. Haciendo resulta la figura siguiente



Para encontrar el valor de K solo tenemos que clicar sobre la gráfica justo en el punto donde los polos empiezan a estar en el semiplano derecho. Haciendo esto resulta que  $K = 12.2$  (Gain en MATLAB).

$K = 12.2330$



## PID

### Segundo método Ziegler-Nichols:

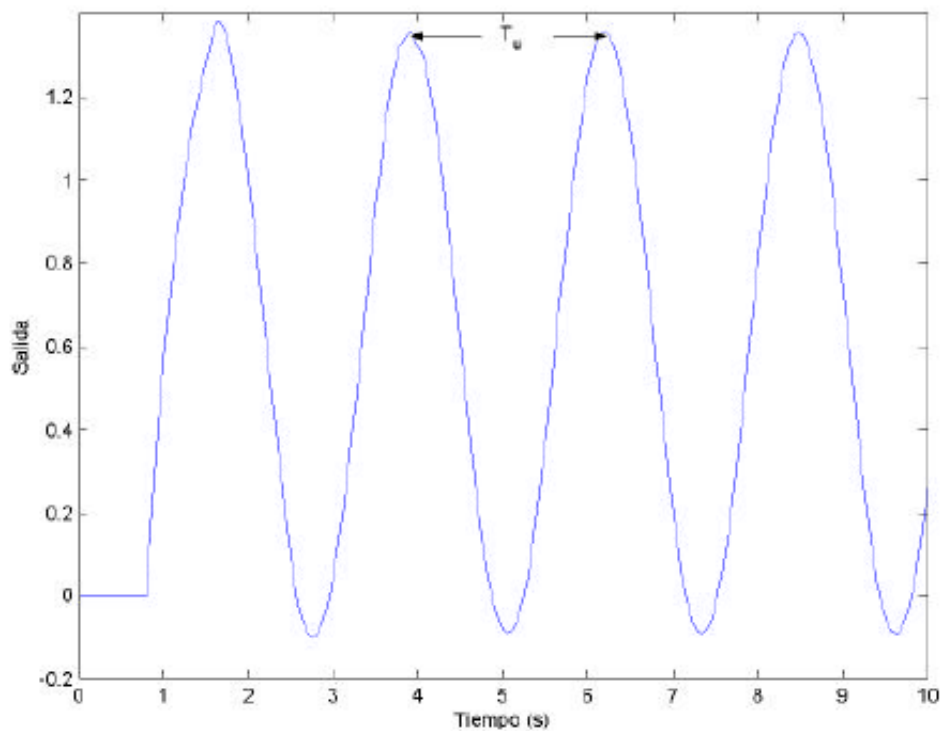
El segundo método de Ziegler-Nichols, o método de respuesta en frecuencia es un método alternativo de sintonización de PID que puede describirse como sigue:

En primer lugar es necesario ajustar las ganancias integral y derivativa a cero, esto es  $K_i = 0$  y  $K_d = 0$ .

A continuación, partiendo de un valor bajo de la ganancia proporcional,  $K_p$ , vamos aumentando ésta gradualmente hasta conseguir un comportamiento oscilatorio mantenido en la respuesta del sistema tal como muestra la gráfica. A esta ganancia la llamaremos  $K_U$ .

El otro parámetro que nos hace falta es el periodo de oscilación del sistema para esta ganancia, que llamaremos  $T_U$ , y que se calcula como muestra la gráfica.

Con los valores de  $K_U$  y  $T_U$  entramos en la tabla 2 de Ziegler-Nichols y calculamos los parámetros correspondientes.



A partir de esos valores, Ziegler-Nichols sugirieron que se establecieran los valores de los sistemas P, PI y PID en función de la tabla.

Controlador	$K_p$	$K_i$	$K_d$
P	$0.5K_U$	0	0
PI	$0.45K_U$	$\frac{1.2}{T_U}$	0
PID	$0.6K_U$	$\frac{2}{T_U}$	$0.125T_U$

Tabla 2: Parámetros del PID según el método de respuesta en frecuencia de Ziegler-Nichols

O también

Controlador	Kp	Ti	Td
P	0.5Kcr	0	0
PI	0.45Kcr	1/1.2 Pcr	0
PID	0.6Kcr	0.5Pcr	0.125Pcr

Calcular las respuesta en el dominio temporal y caracterizar la respuesta según la ganancia estática a lazo cerrado ( $K_0$ ), sobreoscilación (SO), tiempo de subida, ( $t_s$ ), tiempo de establecimiento ( $t_e$ ) y ratio de decaimiento (rd).

Observamos que si es conocido el método matemático, a partir del lugar de raíces es posible la obtención de esa  $K_{cr}$  y  $P_{cr}$  en función del corte de las ramas con el eje  $j\omega$ .

Tanto si experimentalmente el sistema no presenta oscilaciones, como si analíticamente el lugar de raíces no corta el eje  $j\omega$ , este método no puede ser utilizado.

### Resumen

**PD:** Este controlador introduce un cero en el sistema

**PI:** Este controlador introduce un cero y un polo en el sistema

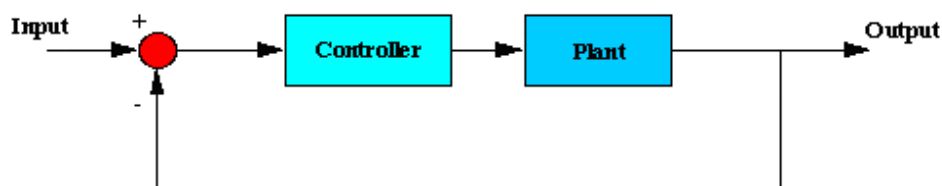
$$G_c(s) = K_p + \frac{K_i}{s} + s * K_d$$

## EJEMPLO PD con Matlab y simulink

### Acción de control proporcional.

Dada la función de transferencia de un sistema cerrado. Deseamos obtener gráficamente la señal de salida de la planta.

Esquema de la planta a estudiar:



Función de transferencia de la planta con el controlador:  $K_p = 300$

$$\frac{Y(s)}{U(s)} = \frac{k_p}{s^2 + ms + (b + k_p)}$$

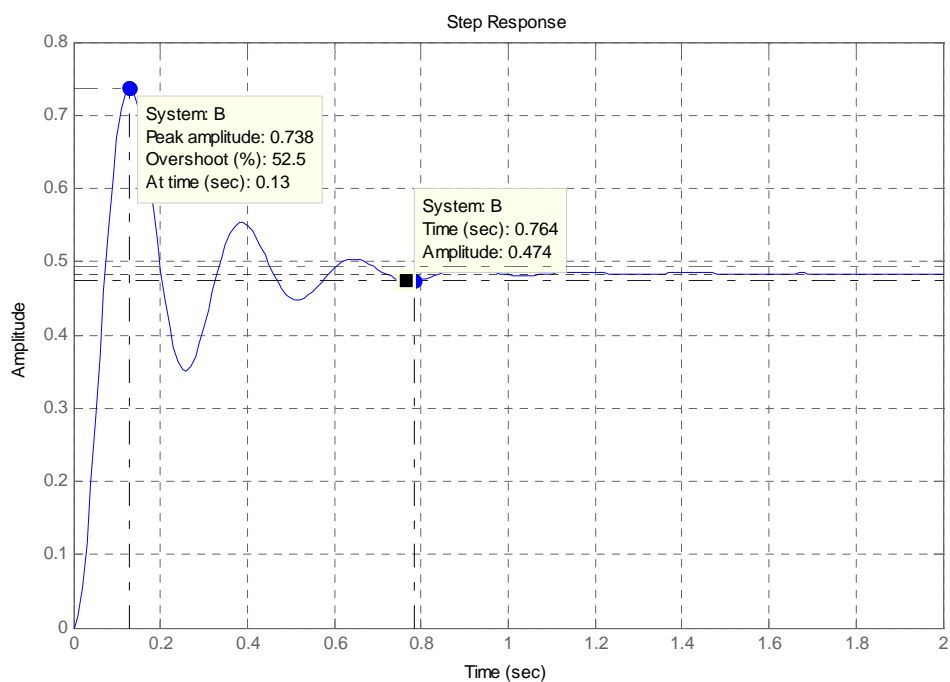
m=10   b=20

---

```
%-----
% Sistemas de control dinámico con MATLAB
% Acción de control proporcional
%-----
% primero definimos el sistema

Kp=300;% introducimos el valor de kp
num=[Kp];
den=[1 10 20+Kp];
Sis=tf(num,den);
t=0:0.01:2;
step(Sis,t)% utilizando este comando observamos gráficamente el resultado
grid;
```

---



Observando la respuesta vemos que la amplitud pico = 0.738 el sobre impulso=52.5% y el tiempo de establecimiento =0.764 seg  
En este ejemplo cambiando el valor de Kp se puede ver claramente los efectos del controlador proporcional en un sistema.

Otra forma de ver los efectos de K es :

```
%-----
% Sistemas de control dinámico con MATLAB
```

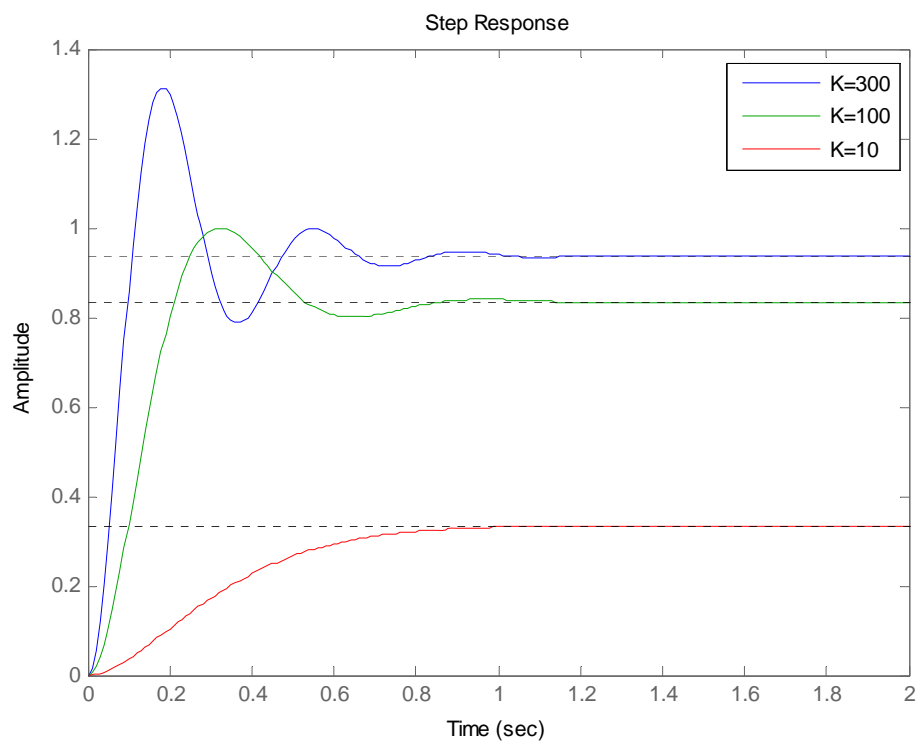
```

% Acción de control proporcional
%-----
% primero definimos el sistema

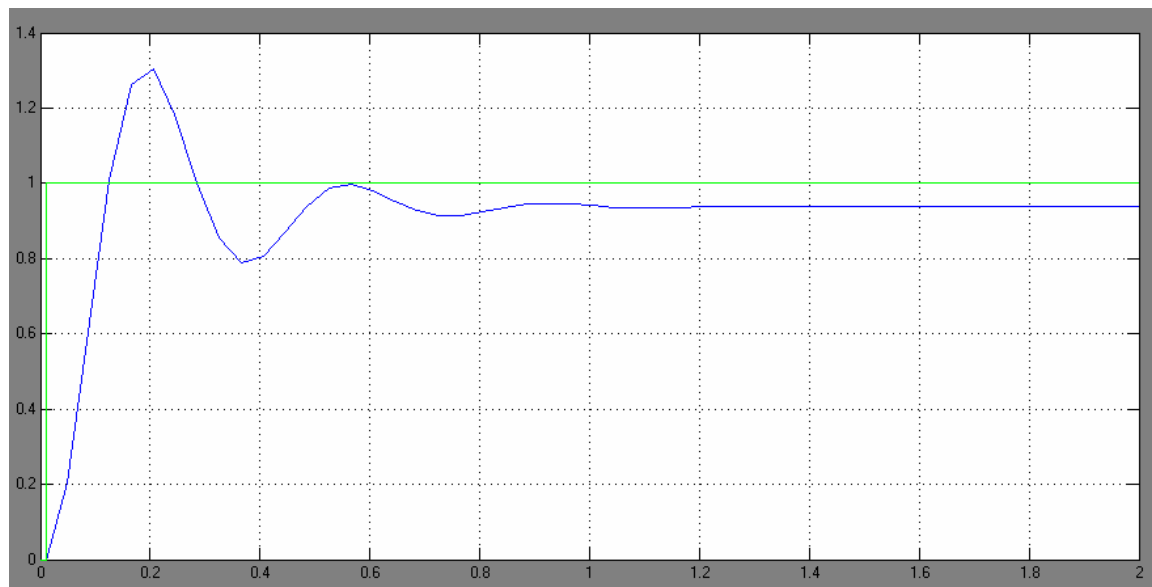
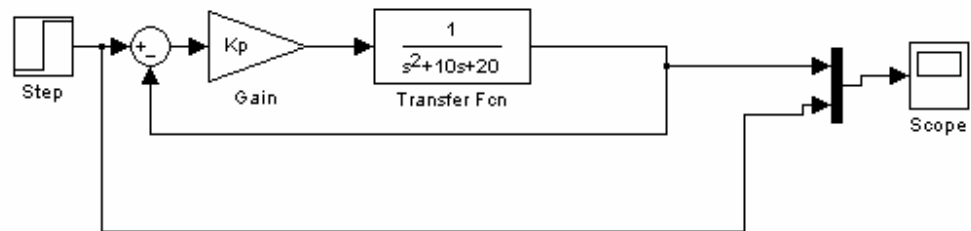
Kp=300;% introducimos el valor de kp
Kp1=100;
Kp2=10;
num=[Kp];
num1=[Kp1];
num2=[Kp2];
den=[1 10 20+Kp];
den1=[1 10 20+Kp1];
den2=[1 10 20+Kp2];
Sis=tf(num,den);
Sis1=tf(num1,den1);
Sis2=tf(num2,den2);
t=0:0.01:2;
step(Sis,Sis1,Sis2,t)% utilizando este comando observamos gráficamente el
resultado
legend('K=300','K=100','K=10');

```

---



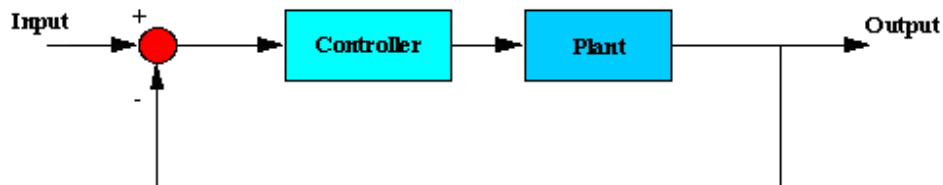
Ahora realizaremos el mismo análisis con simulink



## Control Proporcional Derivativo

Acción de control proporcional- derivativa.

Dada la función de transferencia de un sistema cerrado. Deseamos obtener gráficamente la señal de salida de la planta.



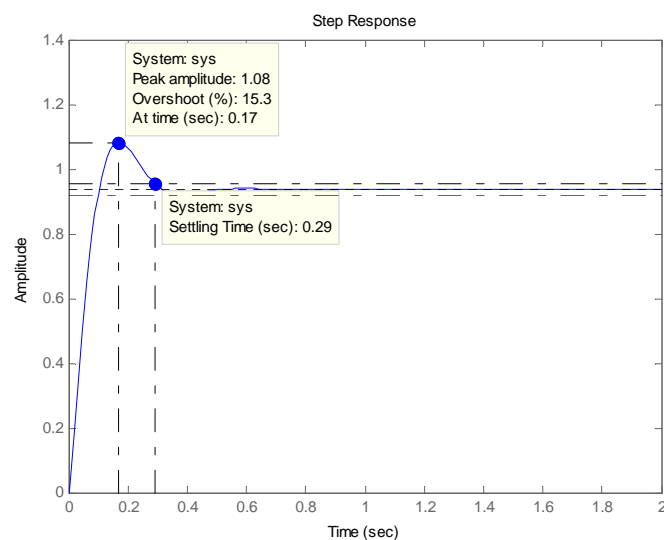
El valor de  $K_p = 300$  y  $K_d = 10$

$$\frac{Y(s)}{U(s)} = \frac{K_D s + K_p}{s^2 + (m + K_D)s + (b + K_p)}$$

```
%-----
% Sistemas de control dinámico con MATLAB
% Acción de control Proporcional-derivativa
%-----
% primero definimos el sistema

Kp=300; % introducimos los valores de kp i kd
Kd=10;
num=[Kd Kp];
den=[1 10+Kd 20+Kp];

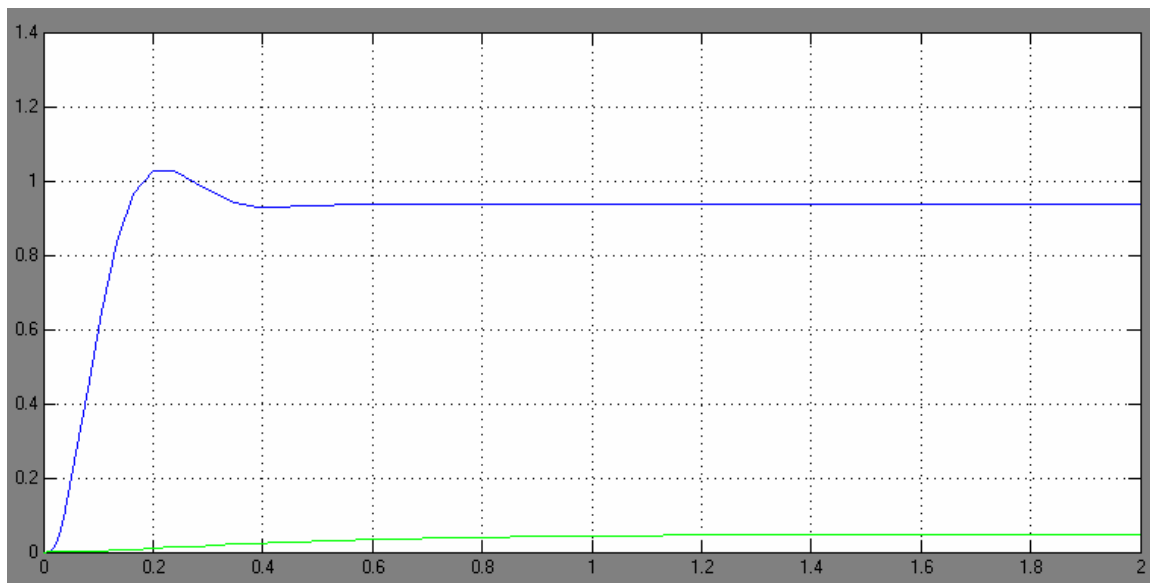
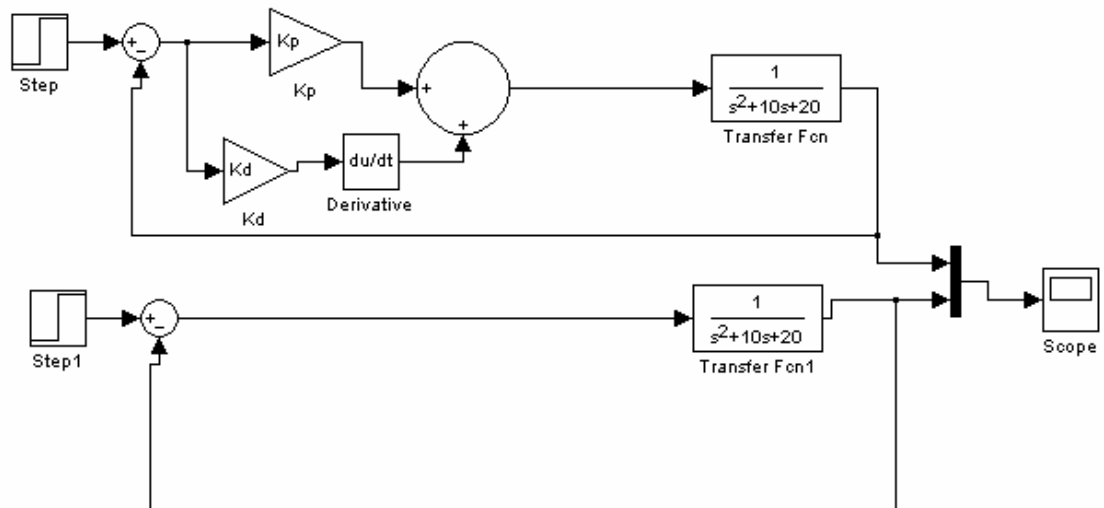
t=0:0.01:2;
step(num,den,t)% obtenemos el grafico
```



Observando la respuesta vemos que la amplitud pico = 1.08 el sobre impulso=15.3% y el tiempo de establecimiento =0.29 seg

sobre

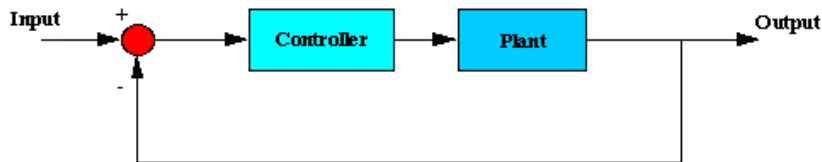
Ahora realizaremos el mismo análisis con simulink



## Control Proporcional Integral

Acción de control proporcional-integral

Dada la función de transferencia de un sistema cerrado. Deseamos obtener gráficamente la señal de salida de la planta.



La función de transferencia de la planta con el controlador es la siguiente:

El valor de  $K_i = 70$  y  $K_p = 30$

$$\frac{Y(s)}{U(s)} = \frac{K_p s + K_i}{s^3 + ms^2 + (b + K_p) + K_i}$$

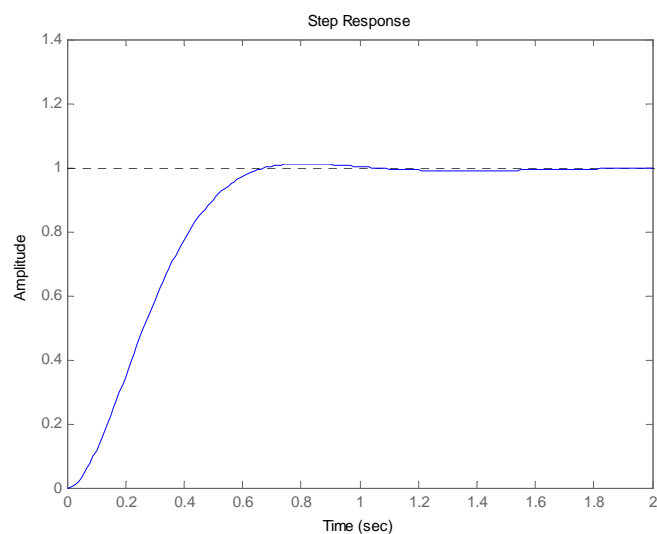
---

```
%-----
% Sistemas de control dinámico con MATLAB
% Acción de control Proporcional-integral
%-----
% primero definimos el sistema
```

```
Kp=30;% introducimos los valores de ki y kp
Ki=70;
num=[Kp Ki];
den=[1 10 20+Kp Ki];
```

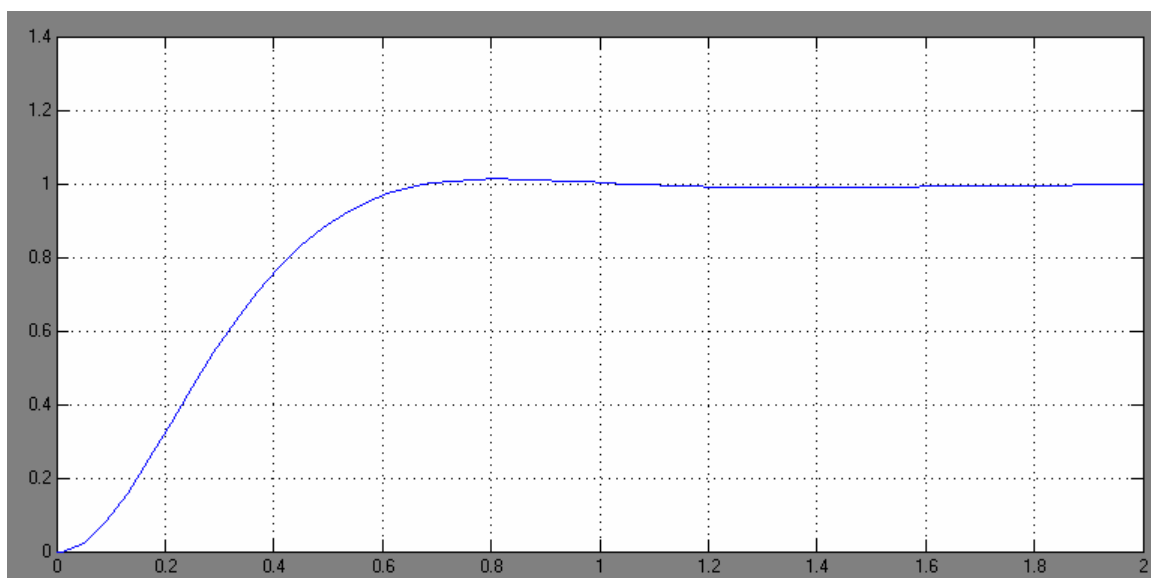
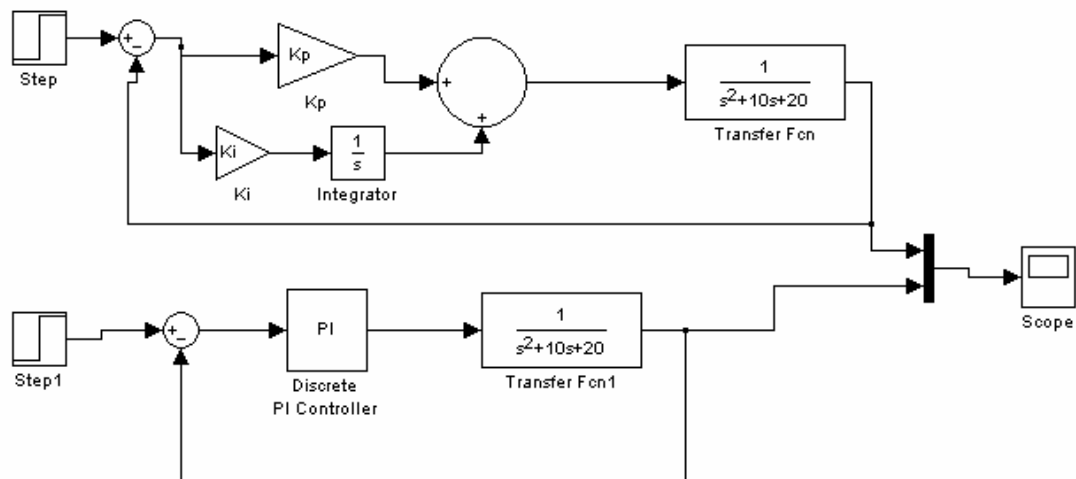
```
t=0:0.01:2;
step(num,den,t)% obtenemos el resultado gráficamente
```

---





Ahora realizaremos el mismo análisis con simulink

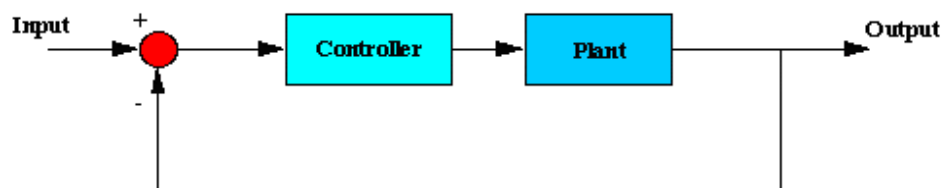


Solo se distingue una curva ya que las 2 curvas son exactamente iguales.

## Control Proporcional Integral Derivativo

Acción de control proporcional-integral-derivativa

Dada la función de transferencia de un sistema cerrado. Deseamos obtener gráficamente la señal de salida de la planta.



La función de transferencia obtenida con el controlado:

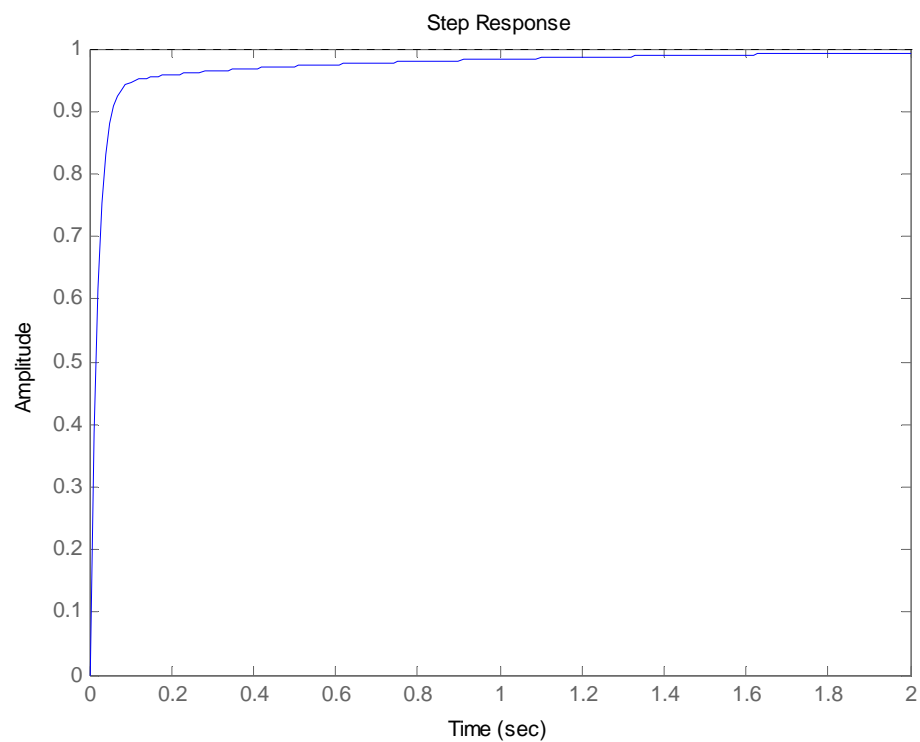
$$\frac{Y(s)}{U(s)} = \frac{K_D s^2 + K_p s + K_I}{s^3 + (m + K_D) s^2 + (b + K_p) s + K_I}$$

El valor de  $K_i = 300$ ,  $K_d = 50$  y  $K_p = 350$

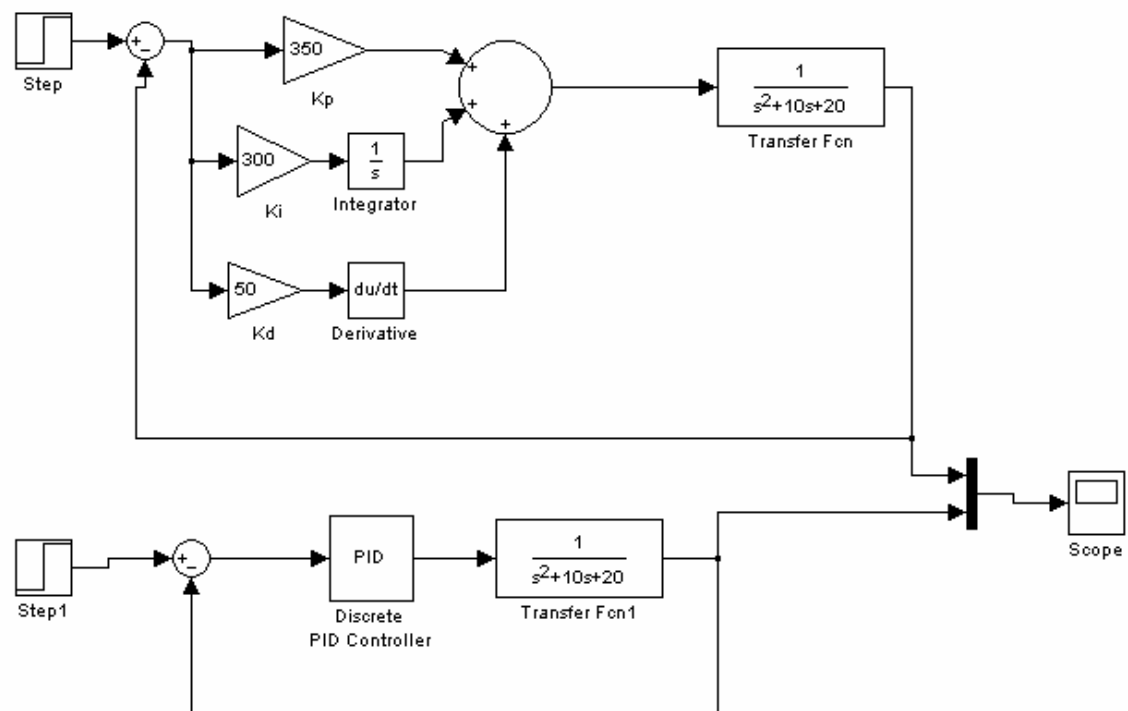
---

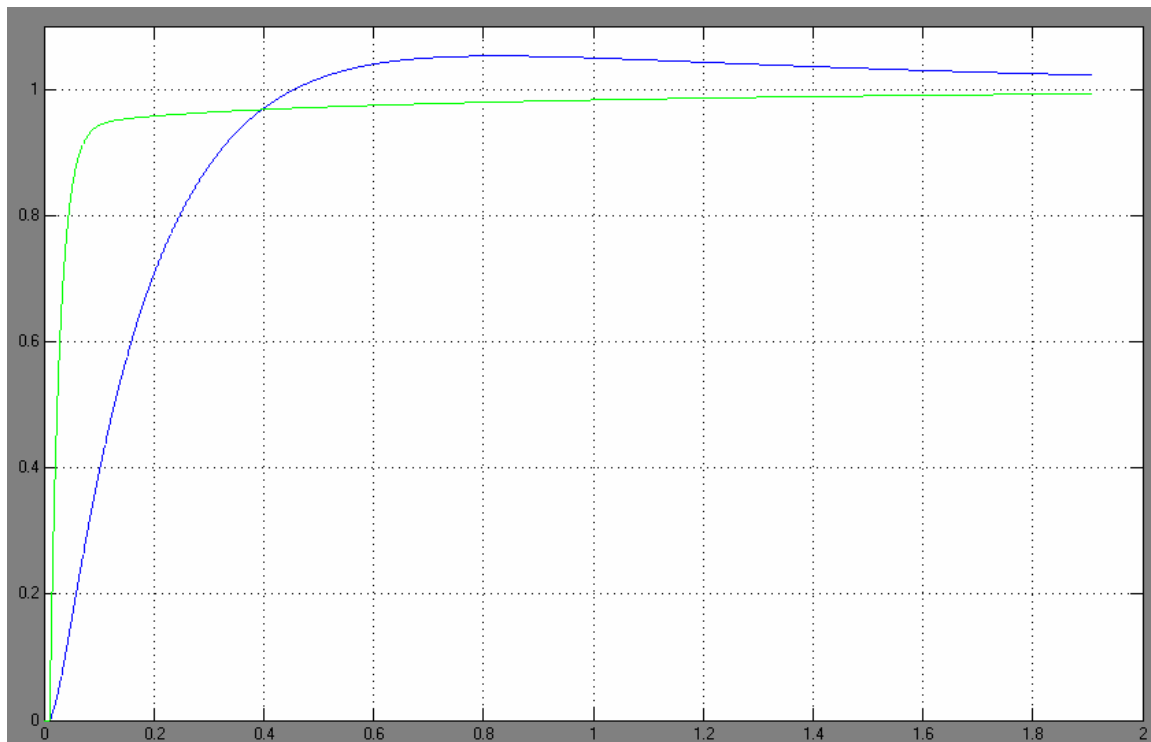
```
%-----  
  
% Sistemas de control dinámico con MATLAB  
% Acción de control Proporcional-integral-derivativa  
%-----  
% primero definimos el sistema  
  
Kp=350;% introducimos los valores  
Ki=300;  
Kd=50;  
  
num=[Kd Kp Ki];  
den=[1 10+Kd 20+Kp Ki];  
  
t=0:0.01:2;  
step(num,den,t)% obtenemos el resultado gráficamente
```

---



Ahora realizaremos el mismo análisis con simulink





Existe diferencia entre curvas \*\*\*\*

## PROBLEMAS

P1.

Sea un proceso, cuya función de transferencia es:

$$G(s) = \frac{10}{(s+2)(s+1)^3}$$

- determine las ganancias del controlador PI resultante de aplicar el primer método de Ziegler y Nichols. Repita el ejercicio usando el segundo método de Ziegler y Nichols.
- Sobre el plano  $s$  indique la ubicación de los polos y ceros (en lazo cerrado) resultantes de aplicar cada controlador.

Sol. 1er método:  $R = 1.267$ ,  $L = 1.2$ ;  
2do método:  $K_u = 0.86$ ,  $P_u = 5.31$

La función es

$$G(s) = \frac{10}{s^4 + 5s^3 + 9s^2 + 7s + 2}$$

```
%Problema 1 PID
% Tenemos que encontrar el valor de KU para cual la primera curva
% oscila
% El valor de TU lo obtenemos pulsando en cada punta de la curva que
% acabamos de encontrar
clc;
close all;
KU=0.86;
W=tf([10],[1 5 9 7 2]);
t=0:0.1:40;
F=feedback(KU*W,1);
step(F,t)
[x y]=ginput(6);
X=x(2)-x(1);
X1=x(4)-x(3);
X2=x(6)-x(5);
TU=(X+X1+X2)/3;
fprintf('El valor de TU es %5.3f\n',TU);
%TU=5.340;
disp('Controlador P');
Kp_p=0.5*KU;
fprintf('El valor de Kp para el controlador proporcional es %5.1f\n',Kp_p)

disp(' ')

disp('Controlador PI');
Kp_pi=0.45*KU;
Ki_pi=1.2/TU;
fprintf('El valor de Kp para el controlador PI es %5.3f\n',Kp_pi);
fprintf('El valor de Ki para el controlador PI es %5.4f\n',Ki_pi);
```

```

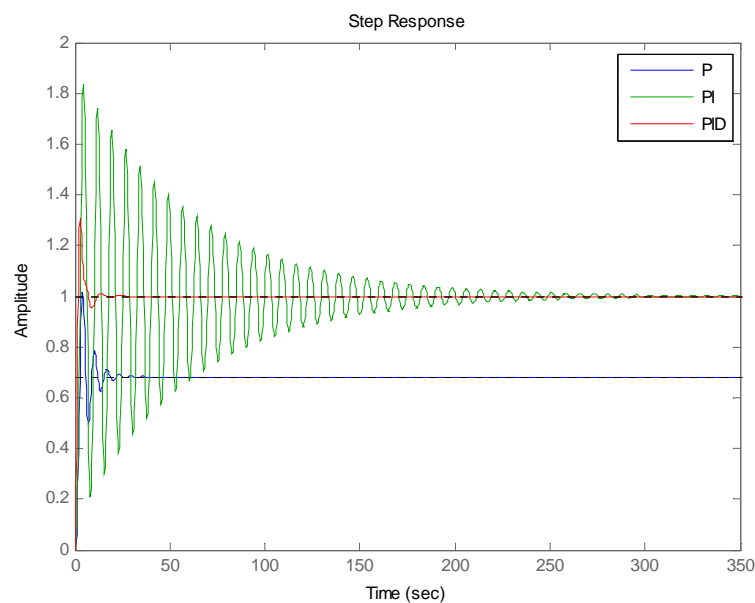
disp(' ')

disp('Controlador PID');
Kp_pid=0.6*KU;
Ki_pid=2/TU;
Kd_pid=0.125*TU;
fprintf('El valor de Kp para el controlador PID es %5.3f\n',Kp_pid);
fprintf('El valor de Ki para el controlador PID es %5.4f\n',Ki_pid);
fprintf('El valor de Kd para el controlador PID es %5.4f\n',Kd_pid);

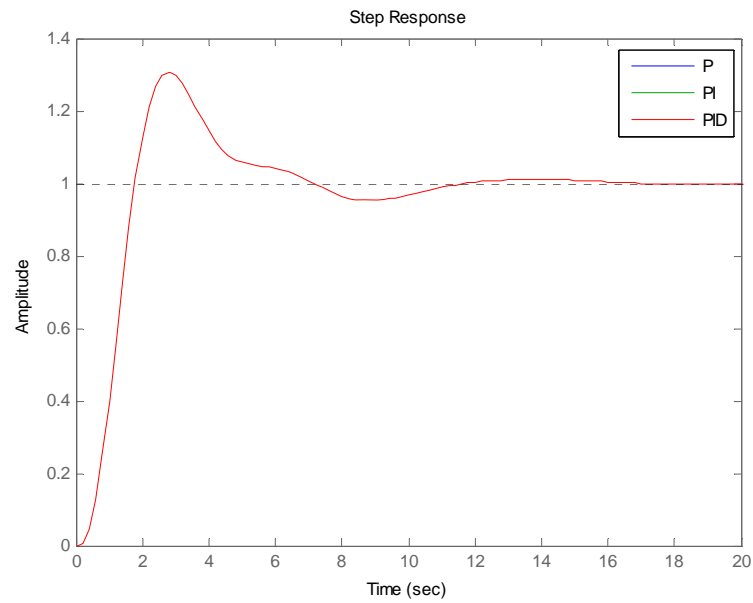
figure
t=0:0.01:30;
%G=feedback(W,1)      ;           % Funcion sin controlador alguno
P=Kp_p;               % Funcion con controlador P
Pi=tf([Kp_pi Ki_pi],[1 0]);    % Funcion con controlador PI
Pid=tf([Kd_pid Kp_pid Ki_pid],[1 0]);
H=feedback(P*W,1);
J=feedback(Pi*W,1);
F=feedback(Pid*W,1);
step(H,J,F);
legend('P','PI','PID')

```

**Grafica de P, PI y PID al mismo tiempo**



**Grafica de PID**



## Simulink

