TEORÍA DE SISTEMAS

PRÁCTICA 7 SISTEMAS. SISTEMAS DISCRETOS Y MUESTREADOS

OBJETIVOS DE LA PRÁCTICA

· Estudiar las funciones disponibles en Matlab y Simulink para el modelado y simulación de sistemas discretos y muestreados.

· Función de Transferencia en z
· Muestreo y reconstrucción de señales en Simulink.
· Discretización de sistemas en Matlab.

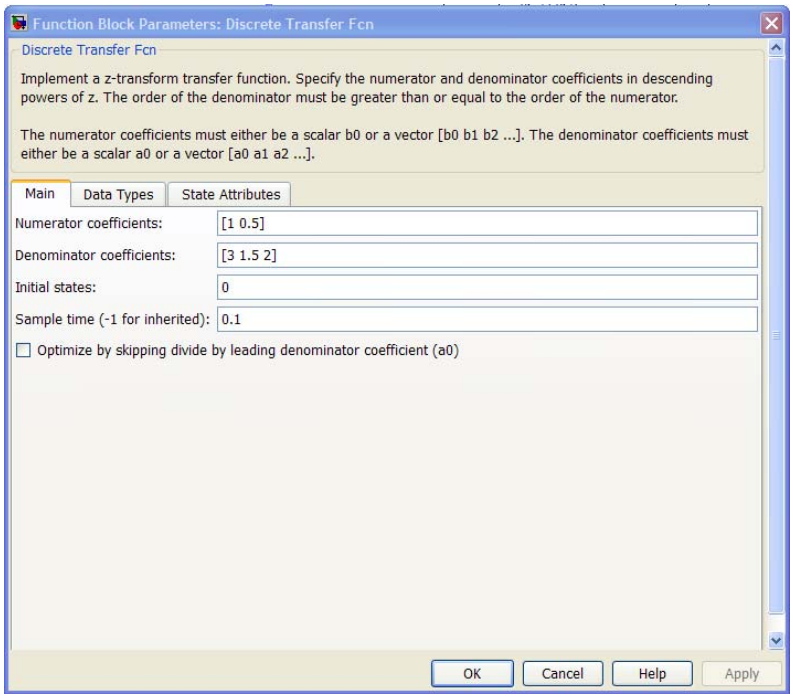
1. INTRODUCCIÓN DE SISTEMAS DISCRETOS EN SIMULINK

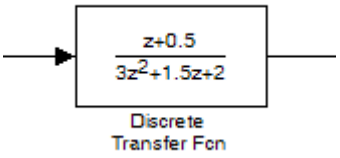
Al igual que los sistemas continuos, los sistemas discretos se pueden representar en Simulink mediante su función de transferencia. Para ello, existen dos posibilidades:

1. Utilizar el bloque **'Discrete Transfer Function'** de la categoría **'Discrete'**. Los coeficientes de numerador y denominador se introducen como vectores, al igual que en el caso de los sistemas continuos; y hay que especificar un parámetro adicional: el tiempo de muestreo (*sample time*). Por ejemplo, si queremos introducir la siguiente función de transferencia:

$$G(z) = \frac{(z + 0.5)}{(3z^2 + 1.5z + 2)}$$

… y suponiendo que el periodo de muestreo deseado fuese 0.1 segundos, los parámetros que tendríamos que introducir y el resultado obtenido serían:

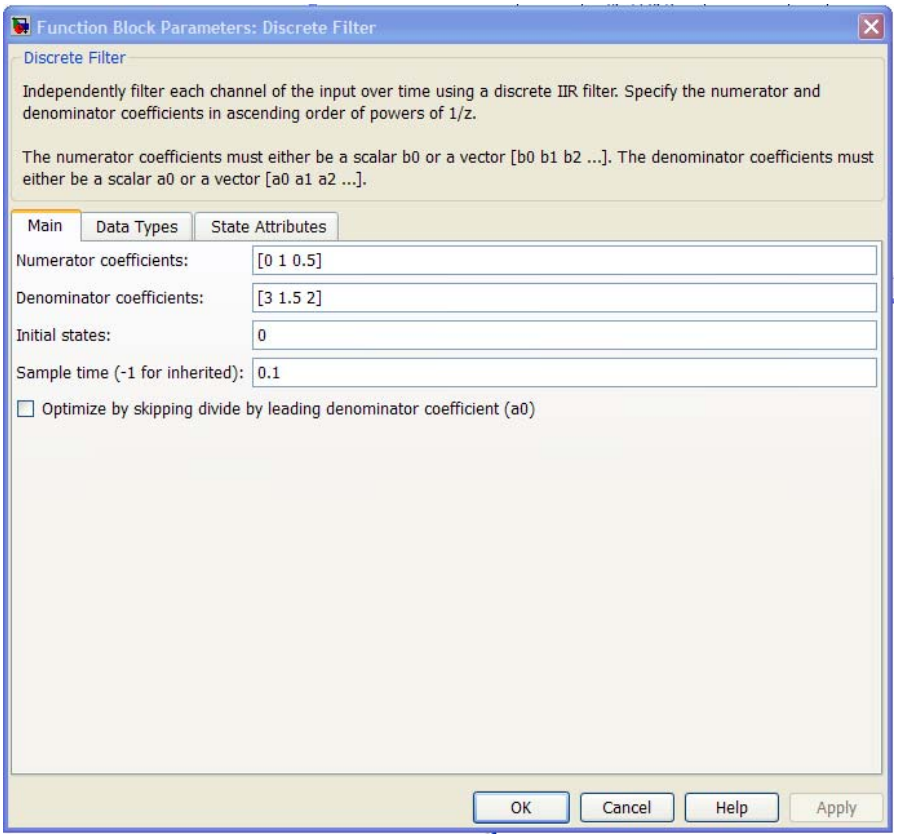




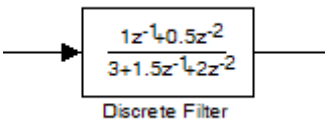
2. Utilizar el bloque **'Discrete Filter'**, también de la categoría **'Discrete'**. Los coeficientes de numerador y denominador se introducen como vectores, al igual que en el caso anterior, pero con la particularidad de que se trabaja en potencias negativas de z. Por ejemplo, la función de transferencia anterior, expresada en potencias negativas de z quedaría:

$$G(z) = \frac{(z + 0.5)}{(3z^2 + 1.5z + 2)} \cdot \frac{z^{-2}}{z^{-2}} = \frac{(z^{-1} + 0.5z^{-2})}{(3 + 1.5z^{-1} + 2z^{-2})}$$

… y suponiendo el mismo periodo de muestreo de 0.1 segundos, la forma de introducir el bloque sería:

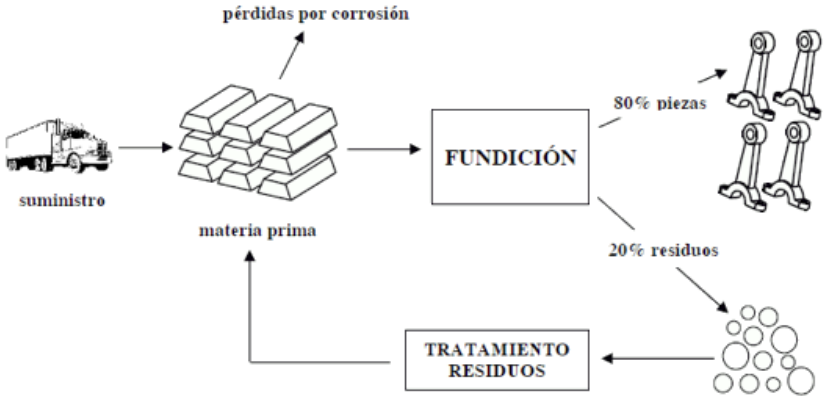


… y el resultado obtenido:



EJERCICIO 1: SIMULACIÓN DE UN SISTEMA DISCRETO EN SIMULINK Y PASO DE LOS RESULTADOS A MATLAB

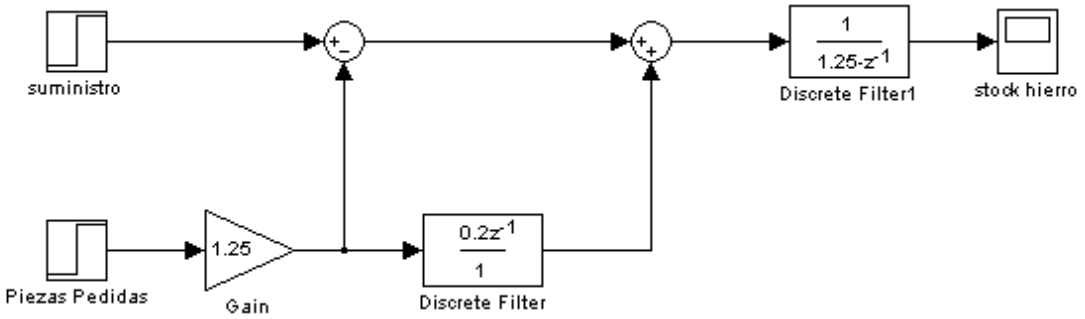
Simularemos el comportamiento de una fundición, cuyo esquema se muestra en la figura siguiente:



Básicamente, a la fundición llega diariamente un suministro de lingotes de hierro, que se procesa para obtener un 80% de piezas y un 20% de residuos. Estos residuos son tratados, con un tratamiento que dura un día completo, y se convierten de nuevo en lingotes listos para ser procesados. Al mismo tiempo, del stock de materia prima se pierde un cierto porcentaje por corrosión.

Se puede modelar el sistema con dos entradas: el suministro diario de materia prima y las piezas pedidas por los clientes (suponemos que sólo se fabrican las piezas que se han pedido); y una única salida: la cantidad de materia prima en stock. De este modo, buscamos ver cómo evoluciona el stock en función del suministro y de la cantidad de piezas pedidas.

El esquema resultante de Simulink sería el siguiente:



Se pide:

1. Introducir el esquema de Simulink mostrado (atención: el tiempo de muestreo debe ser igual para todos los bloques; fijaremos este valor a uno, indicando un periodo de muestreo de un día).

2. Comprobar el efecto de aplicar un escalón de 20 unidades (un aumento de 20 toneladas diarias) a la entrada **'suministro'** mientras la entrada **'piezas pedidas'** se mantiene constante. Para ello haremos los siguientes ajustes:
   - el valor del escalón para **'suministro'** se fijará adecuadamente
   - el valor del escalón para **'piezas pedidas'** se fijará en cero.

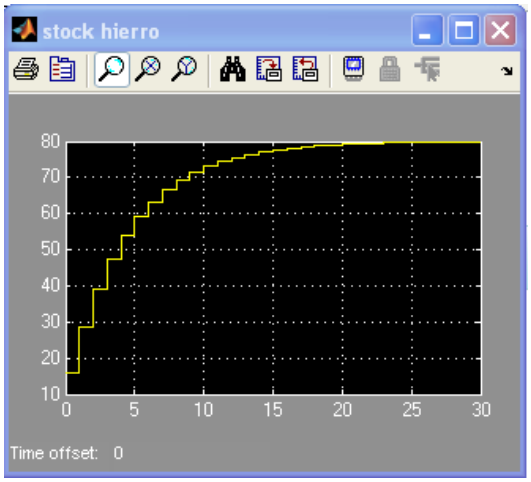
3. Comprobar el efecto de aplicar un escalón de 50 unidades (50 toneladas diarias) a la entrada 'piezas pedidas'. Para ello, haremos los ajustes contrarios:
   - el valor del escalón para **'piezas pedidas'** se fijará en el valor adecuado.
   - el valor del escalón para **'suministro'** se fijará en cero.

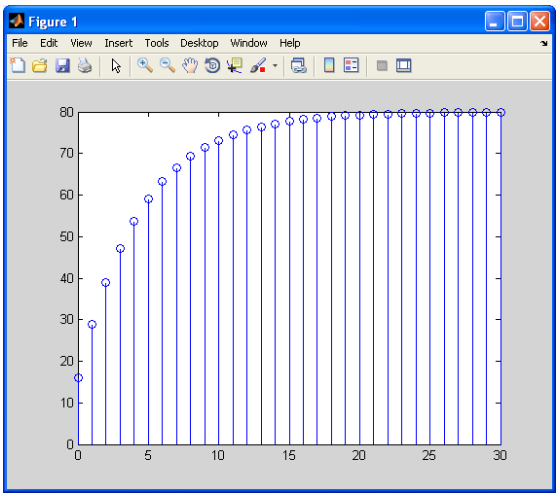
4. En ambos casos, se deberá mostrar el resultado (variación del stock de hierro) en Simulink y luego enviar los resultados a Matlab para representarlos con las funciones **`stem`** o **`stairs`** (similar a plot pero para variables discretas). Recomendación: consultar la ayuda de las funciones **`stem`** y **`stairs`**.

Ejemplo:

```
>> stem(stock(:,1), stock(:,2))
>> stairs(stock(:,1), stock(:,2))
```

Como ejemplo, se muestran los resultados que se deberían obtener (sobre Simulink y sobre Matlab) en el apartado 2. Tras simular el sistema durante suficiente tiempo se comprueba como el valor final para el incremento de stock es de aproximadamente 80 toneladas:



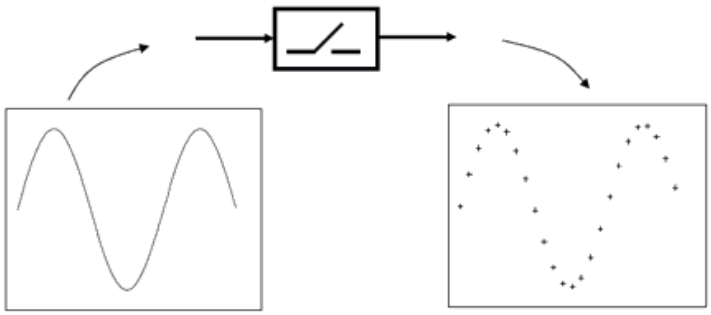


2. MUESTREO Y RECONSTRUCCIÓN EN SIMULINK

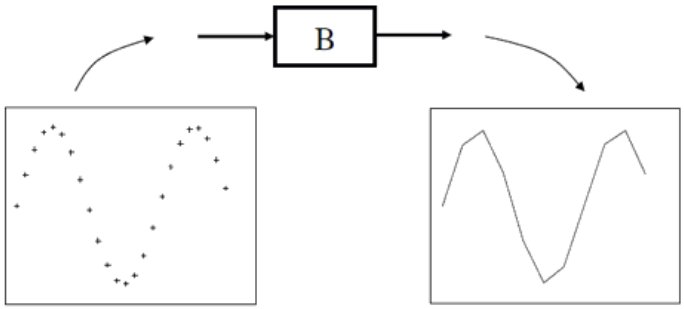
En sistemas compuestos por elementos **continuos** junto a elementos **discretos** siempre están presentes dos elementos fundamentales:

- muestreador
- bloqueador

El **muestreador** transforma una señal **continua** en una secuencia de valores **discretos**



El **bloqueador** realiza la operación contraria: transforma una secuencia **discreta** en una señal **continua**

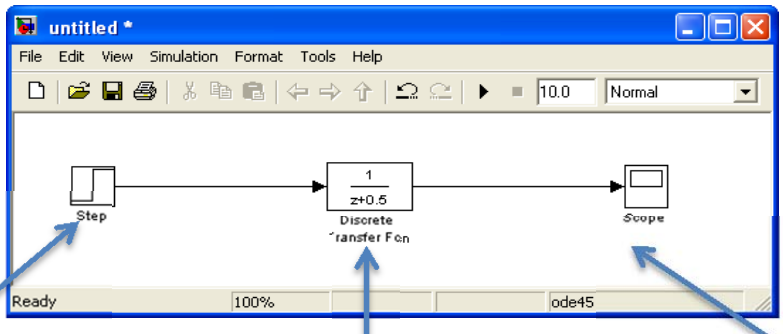


Se puede apreciar cómo la señal reconstruida no coincide completamente con la original

Introducción automática de muestreadores y bloqueadores en Simulink

Cuando en un esquema de Simulink se introducen señales continuas como entrada de bloques discretos, el programa considera que existe un elemento **muestreador** intercalado. Análogamente, si se introducen señales discretas como entrada de bloques continuos, el programa considera que existe un **bloqueador** (de orden cero) intercalado.

Ejemplo: el siguiente esquema de Simulink:

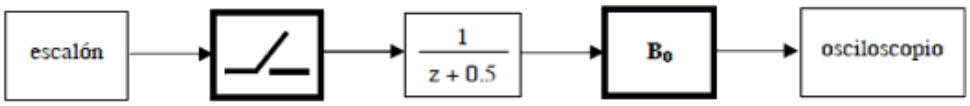


Escalón:
- Genera una señal continua

Bloque discreto:
- Requiere entrada discreta
- Genera salida discreta

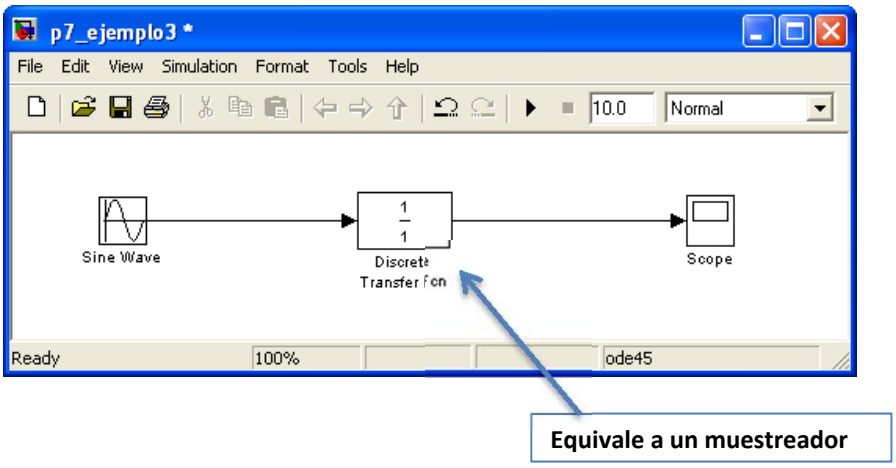
Osciloscopio:
- Requiere entrada continua

...representaría realmente esta otra situación:



Forma de introducir manualmente un muestreador

Si deseamos introducir un bloque que se comporte como un muestreador, basta con crear un bloque discreto que no realice ninguna operación (función de transferencia unidad):



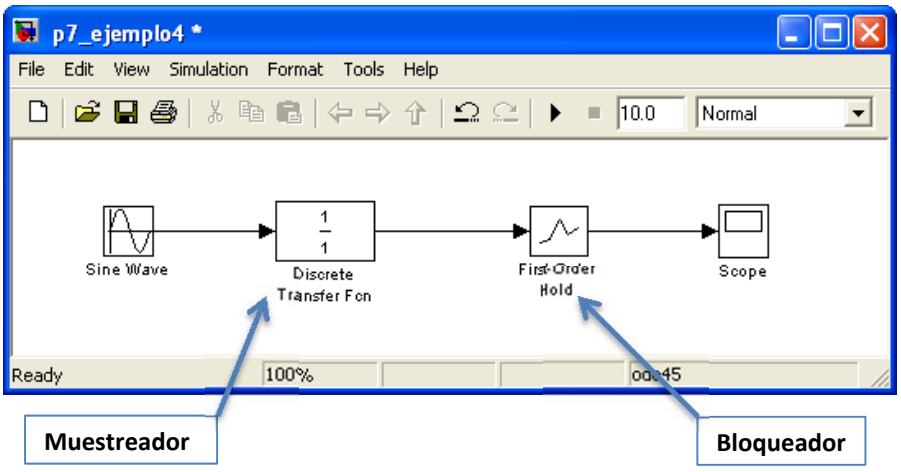
El periodo de muestreo se ajusta con el campo **'Sample Time'** del bloque discreto introducido, que consideraremos que viene indicado en segundos.

Forma de introducir manualmente un bloqueador

En Simulink existen dos tipos de bloqueador: de orden cero y de orden uno. Estos elementos los encontramos dentro de la categoría **'Discrete'**:

- **'Zero-Order hold'** es el bloqueador de orden cero
- **'First-Order hold'** es el bloqueador de orden uno

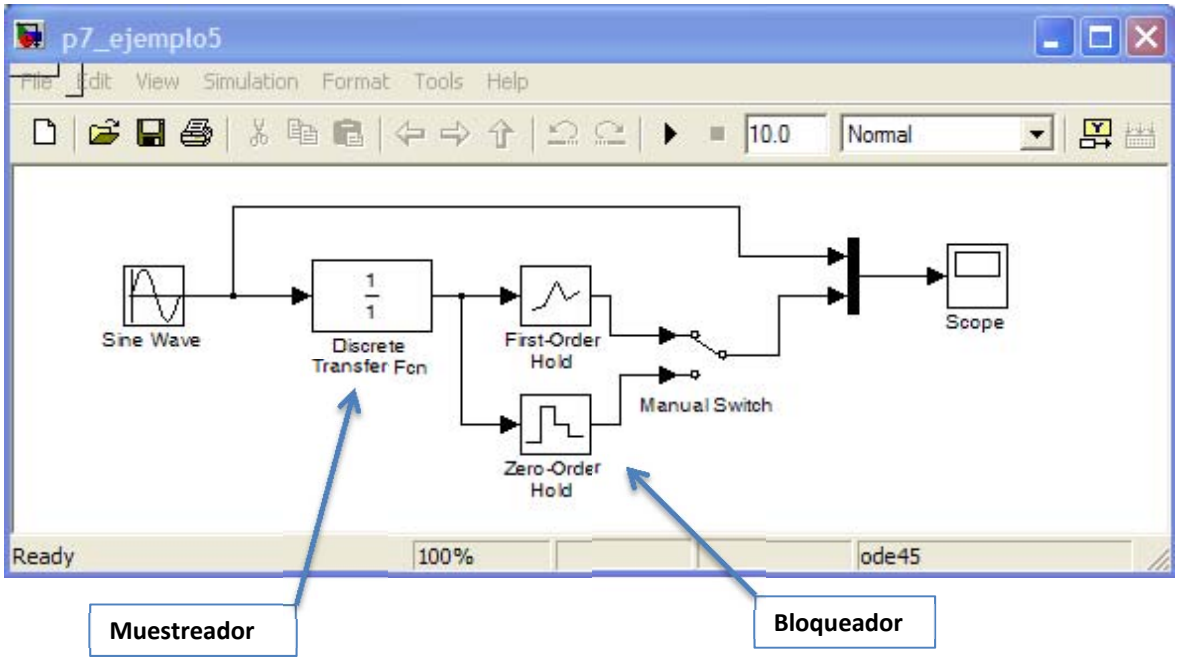
El siguiente ejemplo representa un muestreo seguido de una reconstrucción con un bloqueador de orden uno para una señal senoidal:



3. CALIDAD DE LA RECONSTRUCCIÓN EN FUNCIÓN DEL TIEMPO DE MUESTREO Y DEL TIPO DE BLOQUEADOR

El objetivo será comprobar la fidelidad con que es posible reconstruir una señal después de haber sido muestreada.

Utilizaremos para ello la señal senoidal y construiremos un esquema similar al del ejemplo anterior, donde se representan en el osciloscopio la señal original y la señal reconstruida, de modo que se puede apreciar la diferencia entre ambas.



Comprobaremos los resultados obtenidos para los siguientes casos:

- **Con el bloqueador de orden uno:**
  - periodo de muestreo de 1 segundo
  - periodo de muestreo de 0.5 segundos
  - periodo de muestreo de 0.25 segundos
- **Con el bloqueador de orden cero:**
  - periodo de muestreo de 1 segundo
  - periodo de muestreo de 0.5 segundos
  - periodo de muestreo de 0.25 segundos

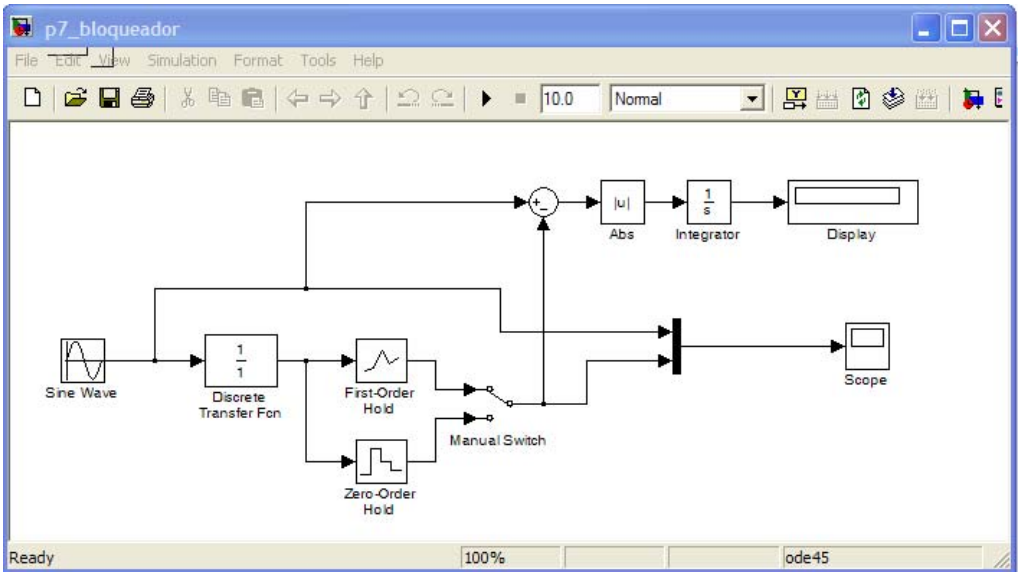
NOTA IMPORTANTE: para obtener resultados correctos es necesario ajustar el tiempo de muestreo al mismo valor en el **muestreador** y en el **bloqueador**

El efecto que debe apreciarse es que al reducir el periodo de muestreo la señal reconstruida se parece más a la original.

EJERCICIO 2: CUANTIFICACIÓN DE LOS ERRORES COMETIDOS AL RECONSTRUIR

Cuantificación de los errores cometidos con cada muestreador:

Los experimentos realizados anteriormente permiten observar a simple vista qué bloqueador reconstruye mejor la señal de partida. Para obtener una medida cuantitativa de los errores de reconstrucción es posible crear un esquema similar al de la figura:



Como añadidos sobre el esquema anterior, se han incluido una serie de bloques que hacen las siguientes operaciones: en primer lugar, restan la señal original de la señal reconstruida, de modo que la diferencia es el error cometido en cada instante. Este error se pone en forma de valor absoluto con el bloque **'Abs'** de la categoría **'Math'**; se integra para obtener el error total; y se muestra en un indicador numérico (bloque **'Display'** de la categoría **'Sinks'**)

De esta forma, el resultado que aparece en el display al final de la simulación es la suma (integral) de los errores producidos en cada instante. Con este montaje repetiremos las pruebas realizadas en el ejercicio anterior: bloqueadores de orden cero y uno a cada una de las distintas frecuencias y anotaremos los resultados obtenidos.

NOTA IMPORTANTE: Dado que la integral suma los errores producidos durante todo el tiempo de simulación, si queremos que los resultados sean comparables debemos hacer ese tiempo de simulación idéntico para todos los casos. Elegiremos un tiempo de simulación de 10 segundos. Del mismo modo, y para hacer los resultados uniformes, la señal senoidal siempre tendrá frecuencia 1 rad/s y amplitud unitaria (tomará valores entre -1 y +1).

A obtener con Matlab:

Tabla en la que se incluyan los errores obtenidos para cada tipo de bloqueador y cada periodo de muestreo. La tabla debe tener un aspecto como el siguiente:

Periodo de muestreo	Error con bloqueador de orden cero	Error con bloqueador de orden uno
1 seg		
0.5 seg		
0.25 seg		

- Los errores deben disminuir al reducirse el periodo de muestreo.
- Los errores deben ser menores con un bloqueador de orden uno.

4. REPRESENTACIÓN EN MATLAB DE SISTEMAS DISCRETOS

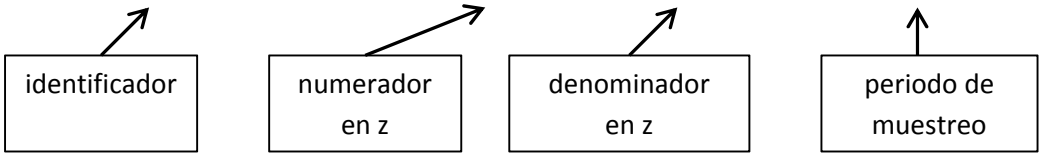
Un sistema discreto se representa en Matlab mediante su función de transferencia en **z**. Caben dos posibilidades:

- Instrucción **tf**: permite escribir la función de transferencia en potencias positivas de z
- Instrucción **filt**: permite escribir la función de transferencia en potencias negativas de z

El formato con el que se deben especificar los datos es el siguiente:

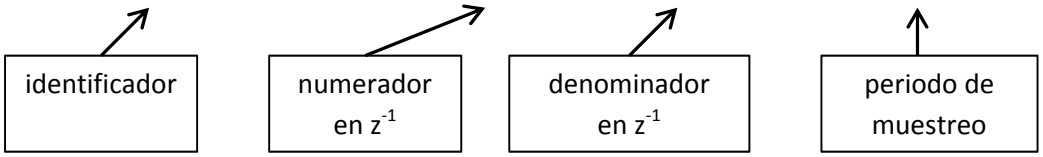
- **Instrucción tf:** se utiliza el mismo formato que para sistemas continuos pero se añade un parámetro más: el periodo de muestreo:

```
nombre_sistema = tf (numerador, denominador, periodo)
```



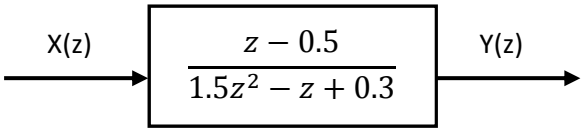
- **Instrucción filt:** también utiliza el mismo formato pero los coeficientes de numerador y denominador están expresados en potencias negativas

```
nombre_sistema = filt (numerador, denominador, periodo)
```



Ejemplo:

Deseamos representar el siguiente sistema discreto:



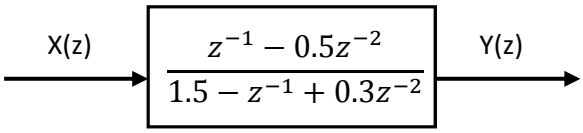
En el que supondremos que el periodo de muestreo es de 0,1 segundos.

- Empleando la instrucción **tf**, deberíamos escribir el siguiente código Matlab:

```
» sis2 = tf([1 -.5], [1.5 -1 .3], .1)

Transfer function:
z - 0.5
----------------
1.5 z^2 - z + 0.3

Sampling time: 0.1
```

- Empleando la instrucción **filt**, deberíamos en primer lugar expresar la función de transferencia en potencias negativas de z multimplicando numerador y denominador por $\mathbf{z^{-2}}$:

$$\frac{z-0.5}{1.5z^2-z+0.3}=\frac{z^{-1}-0.5z^{-2}}{1.5-z^{-1}+0.3z^{-2}}$$



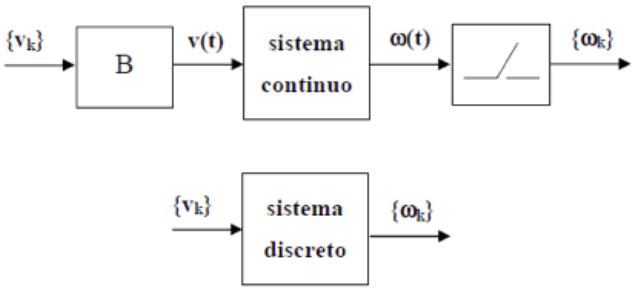
Una vez hecho esto, la orden Matlab a teclear es la siguiente:

```
» sis3 = filt([0 1 -.5], [1.5 -1 .3], .1)

Transfer function:
z^-1 - 0.5 z^-2
---------------------
1.5 - z^-1 + 0.3 z^-2

Sampling time: 0.1
```

Las funciones **sis2** y **sis3** recién creadas se comportarán, por tanto, exactamente igual.

5. DISCRETIZACIÓN DE SISTEMAS CONTINUOS

Matlab también dispone de herramientas para la obtención del sistema discreto equivalente a un sistema continuo dado:

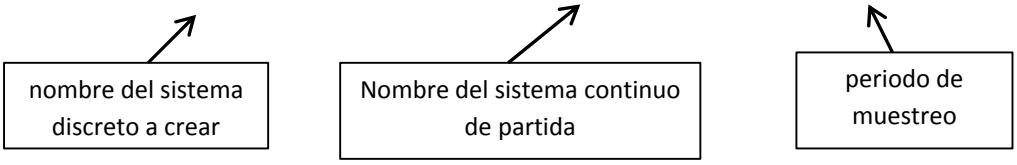


Este proceso lo sabemos resolver manualmente mediante la fórmula de los residuos, y los resultados dependen del tipo de bloqueador empleado: de orden cero, de orden uno, etc.

La instrucción de Matlab que calcula el equivalente discreto para un sistema continuo dado es **c2d**.

El formato más simple para la instrucción **c2d** es el siguiente:

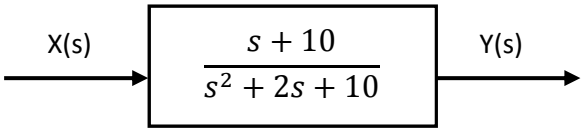
```
sistema_discreto = c2d (sistema_continuo,  periodo)
```



En este formato, la instrucción c2d asume que el bloqueador empleado es un bloqueador de orden cero.

Ejemplo:

Se desea obtener el equivalente discreto para el sistema continuo sis1:



La instrucción de Matlab para introducir el sistemas continuo **sis1**:

```
» sis1 = tf([1 10], [1 2 10])
Transfer function:
      s + 10
---------------
s^2 + 2 s + 10
```

Para obtener el equivalente discreto considerando un bloqueador de orden cero y un periodo de muestreo de **0.2** segundos, la instrucción Matlab a teclear sería:

```
» sis1d = c2d(sis1, .2)

Transfer function:
0.3243 z - 0.005408
----------------------
z^2 - 1.351 z + 0.6703

Sampling time: 0.2
```

Podemos ver como Matlab nos devuelve la función de transferencia correspondiente al sistema discreto equivalente, que en este caso hemos llamado **sis1d**.

Utilización de un bloqueador de orden uno:

En el caso de utilizar un bloqueador de orden uno, el formato de la instrucción **c2d** cambia y es necesario añadir un parámetro más que especifica el tipo de bloqueador a utilizar. En el caso de nuestro sistema **sis1**, la expresión Matlab sería:

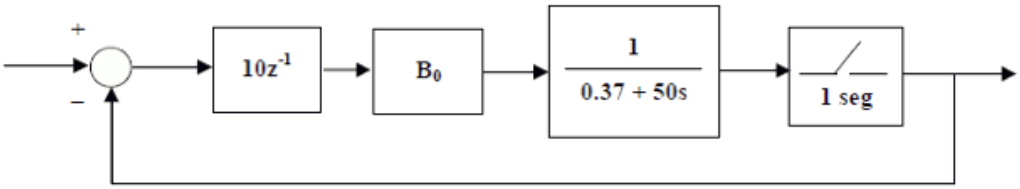
```
» sis1d = c2d(sis1, .2, 'foh')

Transfer function:
0.1444 z^2 + 0.2003 z - 0.02582
------------------------------
z^2 - 1.351 z + 0.6703

Sampling time: 0.2
```

Donde **'foh'** significa *first order hold* o, lo que es lo mismo, bloqueador de orden uno.

Podemos ver como la función de transferencia discreta obtenida es bastante distinta.

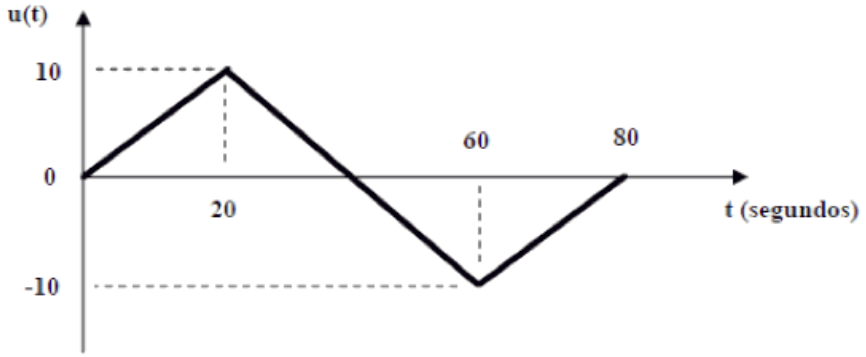
EJERCICIO 3: RESOLUCIÓN DE UN PROBLEMA COMPLETO CON MATLAB

Dado el siguiente sistema:



Se pide:

- Obtener el equivalente discreto para la parte continua
- Reducir el diagrama hasta obtener la función de transferencia total que relaciona entrada y salida
- Representar gráficamente la respuesta del sistema a un escalón de 10 unidades
- Representar la respuesta del sistema ante la siguiente señal de entrada:



***NOTA*:** en este caso las variables **t** y **u** se deben definir con un intervalo de 1 segundo, por ser ese el periodo de muestreo del sistema discreto resultante.
Las funciones para obtener la respuesta ante una entrada son las mismas vistas en el análisis de sistemas continuos (Práctica 3). Comandos: ***step***, ***lsim***

A obtener con Matlab:

- Gráfico de la señal de respuesta ante escalón de 10 unidades.
- Gráfico de la señal de respuesta ante la señal de entrada del ejemplo.

(Ambos gráficos deben incluir el nombre del alumno como título)