

Ingeniería Electrónica 5°

LABORATORIO

**SISTEMAS DE CONTROL EN
TIEMPO REAL**

Práctica III (Voluntaria)

*Implementación de Filtros Digitales
FIR*

Objetivos

La aplicación de técnicas de tratamiento de señales analógicas de entrada y salida es muy usual en SCTR. Un ejemplo lo podemos tener en el caso del tratamiento de señales procedentes de sensores que entregan señales analógicas.

Una de las posibles soluciones es tratar estos problemas mediante técnicas digitales. Como ejemplo de tratamiento de señales, vamos a comenzar por utilizar la aplicación más usual: el filtrado digital. En esta primera incursión, vamos a implementar solo filtrado FIR.

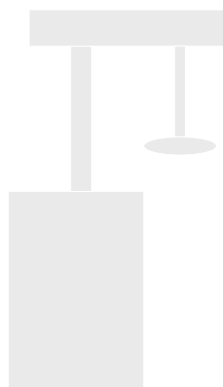
El objetivo es diseñar un Paquete ‘Ada.Digital.Filtros.FIR’ que incluya un repertorio de funciones de filtrado digital.

Requisitos

No es necesario el conocimiento de realización de filtros, es decir, los coeficientes y orden de los filtros se suponen ya calculados con otras herramientas, por ejemplo Matlab. Nos limitamos a realizar las funciones de filtrado, es decir, a usar los filtros.

Plazos de Entrega

Hasta el último día de clase.



Teoría

Los filtros digitales se categorizan en FIR (Respuesta al impulso finita), y IIR (Respuesta al impulso infinita), o de forma equivalente, filtros que sólo tienen ceros, o filtros que tienen ceros y polos.

Considerando $x[n]$ la secuencia de muestras de entrada, $y[n]$ la de salida, se definen las ecuaciones en diferencias para el caso del filtrado FIR:

FIR

$$y[n] = b_0 * x[n] + b_1 * x[n-1] + \dots + b_{N-1} * x[n-(N-1)]$$

- $x[n]$ es la entrada actual, $x[n-1]$ la entrada previa, y así sucesivamente.
- $y[n]$ representa la salida actual.
- b_m representa el coeficiente de orden m .

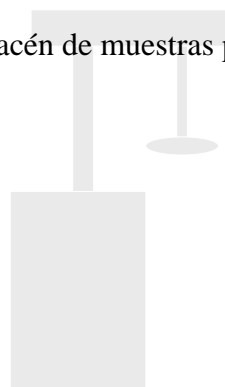
El filtro sería de orden $N-1$.

¿Qué datos necesitamos conocer previamente?

Solamente los coeficientes del filtro b_m que como dijimos se han calculado previamente utilizando otro software diferente.

El problema del filtrado se reduce simplemente a una realización de sumas de productos de coeficientes estáticos por valores de muestras de entrada actuales y pasadas. Necesitamos por tanto dos almacenes de datos:

- Almacén de coeficientes.
- Almacén de muestras previas y actual.



A continuación se presenta una posible organización de datos en memoria:

B_{N-1}	t_0	t_1
B_{N-2}	$x[n-(N-1)]$	$x[n]$
	$x[n-(N-2)]$	$x[n-(N-1)]$
	$x[n-(N-3)]$	$x[n-(N-2)]$
	...	$x[n-(N-3)]$
B_2
B_1
B_0	$x[n-1]$...
	$x[n]$	$x[n-1]$

Como se puede observar, la tabla de coeficientes es estática. En cambio, la tabla que aloja las muestras de entrada va rotando, de manera que si para t_0 la última posición era $x[n]$, en t_1 pasa a ser considerada $x[n-1]$.

Para calcular el valor filtrado $y[n]$ se obtiene la entrada actual $x[n]$, que podría ser la procedente por ejemplo del generador de tonos de la práctica anterior, se realizan las operaciones de la ecuación, y se obtiene el primer valor de salida del filtrado. Se actualiza la tabla de muestras $x[n]$ para el siguiente ciclo de filtrado...y así sucesivamente.

Desarrollo

El objetivo es crear un Tipo Abstracto recogido bajo el paquete Ada.Digital.Filtros.FIR. Este ADT(Abstract Data Type), me permite crear múltiples instancias de este tipo. De manera que puedo crear múltiples filtros preparados para ser utilizados con diferentes coeficientes cada uno de ellos.

¿Qué operaciones debe incluir el paquete?

- *Inicialización del Filtro.* Es decir, carga de coeficientes en la tabla de coeficientes del filtro, definir orden del filtro, apuntadores de tabla de coeficientes al inicio, e inicialización de la tabla de muestras(se puede usar la función definida en el siguiente punto).

- *Reset del filtro.* Poner a 0 la tabla de muestras de entrada e inicializar los apuntadores de la tabla de muestras al inicio.
- *Función de operación de filtrado.* El filtro ya inicializado tras crear el objeto, se usa con esta función, recibe una muestra de entrada y devuelve el valor de salida filtrado.

¿Qué tipo de datos debo utilizar?

FLOAT.

Consejos...

Como puedo cargar filtros de diferentes longitudes, las tablas podrán tener diferentes longitudes en función del orden del filtro cargado. Por ello, lo ideal es utilizar arrays dinámicos.

Para probar el filtrado podemos incorporar nuestro paquete **generador de tonos**. Lo modificamos si es necesario para que incluya un función que devuelva muestras del tono...este será nuestro $x[n]$.

Para visualizar el resultado podemos reutilizar la función de representación del paquete **generador de tonos**.

Datos previos

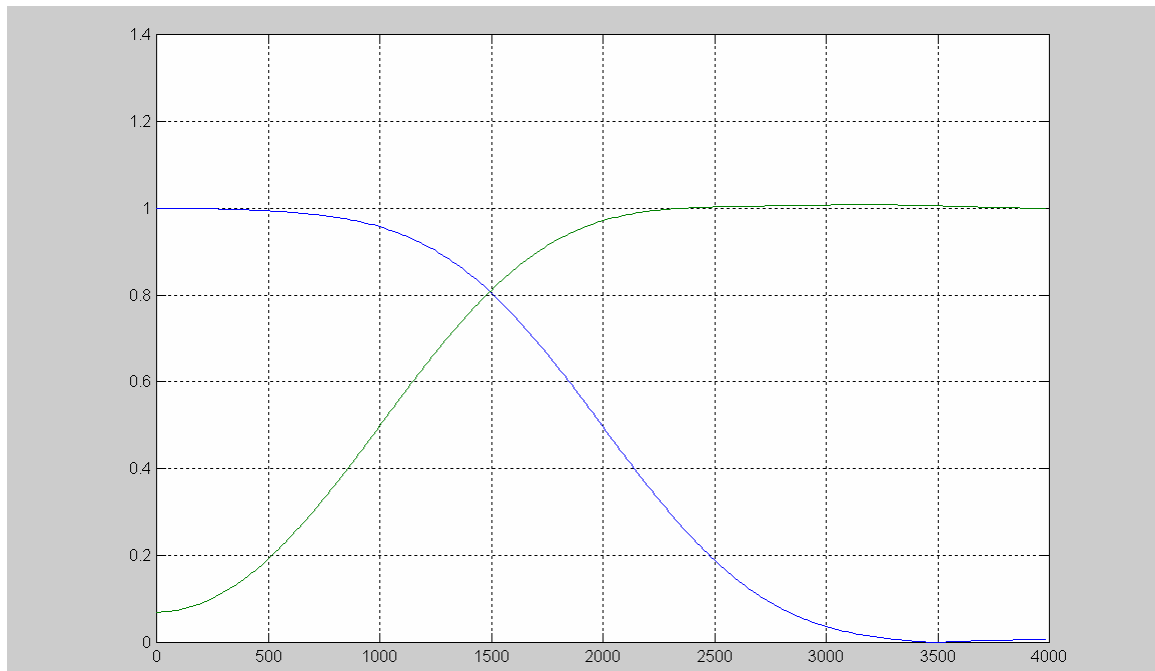
A continuación os doy los coeficientes de dos filtros FIR, el primero de orden 10 paso bajo y el segundo de orden 10 paso alto. Represento también las respuestas en frecuencia de los dos. Los coeficientes comienzan con b_0 .

FIR1 paso bajo:

0.00506032, -3.25051e-018, -0.0419429, 1.321e-017, 0.288485, 0.496795,
0.288485, 1.321e-017, -0.0419429, -3.25051e-018, 0.00506032

FIR2 paso alto:

0.00361583, -8.21176e-018, -0.0299701, -0.109006, -0.206136, 0.753033,-
0.206136, -0.109006, -0.0299701, -8.21176e-018, 0.00361583



Respuesta en Frecuencia

La frecuencia de muestreo de referencia es de 8000Hz, con lo que podemos introducir señales de hasta 4000Hz. Estos valores sirven de ejemplo a la hora de generar tonos de frecuencias diferentes para probar los filtros.

Funcionamiento

Debe ser posible desde un programa principal crear el filtro o filtros. Leer muestras del tono o tonos, ir pasándolas al filtro, leer el resultado, y representar en pantalla.

