Escola Universitària
Politècnica de Mataró

Enginyeria Tècnica Industrial: Especialitat Electrònica Industrial

Projecte d'Estudi i Realització de Filtres Digitals per a Tractament de So

Ricard Marfil Camuñas
Joan Triadó Aymerich

TARDOR 2009

Resum

El projecte tracta sobre l'estudi i el coneixement dels mètodes i processos utilitzats en tractament de senyal i més concretament de les tècniques empleades en el filtratge de senyals de so discretes amb l'objectiu de proporcionar i establir l'aprenentatge dels diferents mecanismes necessaris per tal de controlar el seu comportament.

Mitjançant les eines de simulació i càlcul adequades, junt amb els coneixements que s'engloben en el món del tractament de senyals digitals, s'analitzarà el funcionament dels filtres discrets digitals, com es comporten i com modifiquen una senyal de so en unes circumstàncies determinades.

Com a finalitat última del projecte i després d'haver estudiat el comportament dels diferents tipus de filtre digital, es dissenyaran i programaran aquests filtres amb l'editor del Matlab i amb el Simulink, esbrinant els mètodes de disseny apropiats per tal d'aconseguir l'efecte desitjat.

Resumen

El proyecto trata sobre el estudio y el conocimiento de los métodos y procesos utilizados en tratamiento de señal y más concretamente de las técnicas empleadas en el filtraje de señales de sonido discretas con el objetivo de proporcionar y establecer el aprendizaje de los diferentes mecanismos necesarios para controlar sus comportamientos.

Mediante las herramientas de simulación y cálculo adecuadas, junto con los conocimientos que se engloban en el mundo del tratamiento de señales digitales, se analizará el funcionamiento de los filtros discretos digitales, como se comportan y como modifican una señal de sonido en unas circunstancias determinadas.

Como finalidad última del proyecto y después de haber estudiado el comportamiento de los diferentes tipos de filtro digital, se diseñarán y programarán estos filtros con el editor del Matlab y con el Simulink, averiguando los métodos de diseños apropiados para conseguir el efecto deseado.

Abstract

The project (deals with//aims to) the study and knowledge of methods and processes used in signal processing and more specifically, the techniques used in the filtering of discrete audio signals in order to provide and establish the learning of different methods to control their behaviour.

Through the appropriate simulation and calculation tools, along with the knowledge that fall into the world of digital signal processing, it will analyze the performance of discrete digital filters, how they behave and how they modify a sound signal in given circumstances.

As the ultimate purpose of the project and after studying the behaviour of different types of digital filters, it will design and program these filters with the Matlab and Simulink's editor, finding out the appropriate design methods to achieve the desired effect.

ÍNDEX

ÍNDEX DE FIGURES

1- Objecte del projecte

El projecte es fonamenta en l'estudi de les tècniques utilitzades en tractament de senyals i, de forma més concreta, en els processos empleats en el filtratge de senyals de so discretes amb l'objectiu de controlar el seu comportament en circumstàncies determinades.

L'estudi i el coneixement dels diferents mètodes en el tractament de senyals permetrà analitzar el funcionament dels filtres digitals, veure la seva resposta, ja sigui de forma freqüencial o temporal i esbrinar quin el seu funcionament.

Com a finalitat última, es dissenyaran i programaran aquests filtres amb les eines de càlcul i simulació adequades per aconseguir l'efecte pretès.

2 *Objecte del projecte*

2- Justificació del projecte

La realització del projecte i la feina feta durant tot aquest temps ha proporcionat un nivell de coneixement tècnic en el món del tractament de senyal i, de forma més específica, en el filtratge de senyals discretes que no es tenia anteriorment i que ha estat del tot satisfactori.

El desig i les ganes per aprendre un camp completament nou i ple de possibilitats justifiquen el treball realitzat i motiven a seguir estudiant de forma més encaminada al tema tractat.

3- Especificacions del projecte

L'objectiu del projecte és analitzar, estudiar i aprofundir sobre els sistemes de tractament de senyals discretes per aprendre com es comporten i com modifiquen una senyal de so digital en diferents circumstàncies.

Després de l'anàlisi i amb el coneixement apropiat, s'aprendrà a programar-los i dissenyar-los amb les eines de simulació i càlcul adequades.

Per tant, el contingut del projecte ha de complir les especificacions indicades i satisfer-les.

4- Importància del tractament de senyals

El tractament de senyals és una tecnologia relacionada amb un immens conjunt de disciplines, com l'oci, les comunicacions, la medicina o l'arqueologia. De forma rutinària veiem com a quotidianes les prestacions de sistemes d'entreteniment domèstic, com la televisió o l'equip d'àudio d'alta fidelitat, sistemes que sempre han estat fortament lligats al tractament de senyals.

El paper que juga el tractament de senyals en la nostra societat s'està accelerant, conduït en part per la convergència de les comunicacions i els computadors, tant en l'àrea del consum com en les aplicacions industrials. A més, la complexitat i sofisticació dels sistemes microelectrònics està creixent contínuament. A mesura que la tecnologia es vagi desenvolupant cada cop més, s'implementaran sistemes de tractament de senyals de baix cost, tamany reduïdíssim i baix consum de potència.

En conseqüència, la importància del tractament de senyals anirà augmentant i promourà avanços revolucionaris en algunes àrees d'aplicació.

5- Puntualitzacions

- Les senyals d'entrada seran sempre impulsos de diferents amplituds i senyals sinusoïdals. No es tractaran ni esglaons unitaris ni rampes ja que el tema a estudiar (senyals de so digitals) només hi intervenen senyals impulsionals.

- El motiu d'escollir sempre o gairebé sempre senyals d'àudio (generalment en MP3) mostrejades a 44.100 Hz és per el teorema de Nyquist, que diu que la freqüència de mostreig ha de ser més gran o igual que el doble de la freqüència màxima a mostrejar. En cas contrari, podria aparèixer l'efecte Aliasing, que és la inexacte reconstrucció de la senyal original per la falta de velocitat en el mostreig.

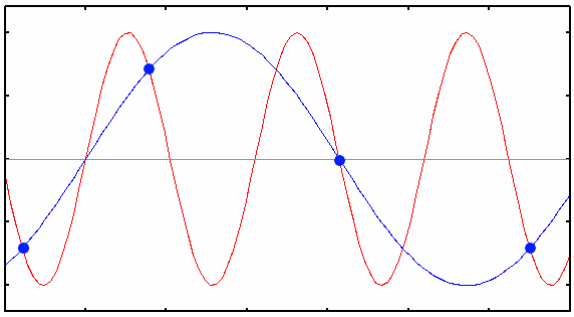


Fig. 5.1. Inexacte mostreig

Com que la freqüència màxima que pot escoltar la oïda humana són 20 kHz aproximadament, la f_m escollida compleix la seva funció sobradament.

$2 \cdot 20.000 = 40.000 \rightarrow f_m \geq 40.000 Hz \rightarrow f_m = 44.100 Hz$

- Tots els filtres són LTI's (Linear Time Invariant). Són lineals perquè la sortida de tots els filtres en un instant determinat la formaran combinacions de senyals d'entrada i sortida (retardades o no) sumades o restades entre elles. I són invariants en el temps perquè els paràmetres del sistema no canvien amb el temps i, per tant, una mateixa entrada ens donarà el mateix resultat en qualsevol moment.

6- Comportament temporal d'un sistema discret

Es disposa del següent sistema discret i es vol estudiar el seu comportament:

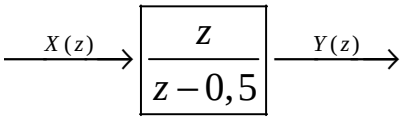


Fig. 6.1. Sistema discret

Es fan els càlculs següents per esbrinar el diagrama de blocs i l'equació en diferències que forma el sistema:

$$\frac{Y(z)}{X(z)} = \frac{z}{z-0,5}$$

$$X(z)\cdot z = z\cdot Y(z) - 0,5\cdot Y(z)$$

$$X(z) = Y(z) - 0,5\cdot Y(z)\cdot z^{-1} \rightarrow Y(z) = X(z) + 0,5\cdot Y(z)\cdot z^{-1}$$

Aquest és el diagrama de blocs:

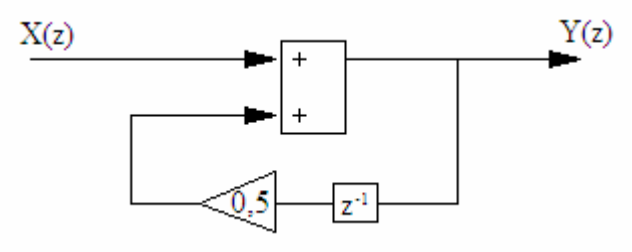


Fig. 6.2. Diagrama de blocs

Si la funció anterior es converteix en equació en diferències quedaria així:

$$\boxed{y(k) = x(k) + 0,5\cdot y(k-1)}$$ on k=1,2,3…

Com es pot veure, la sortida del filtre la forma la suma de la senyal d'entrada més una mostra anterior de la sortida atenuada a la meitat.

Si es connectessin els següents valors a l'entrada, el resultat seria aquest:

$$x(k) = \left\{-3.1 \cdot 10^{-5}, 3.1 \cdot 10^{-5}\right\}$$

$$y(1) = -3.1 \cdot 10^{-5}$$

$$y(2) = 1,55 \cdot 10^{-5}$$

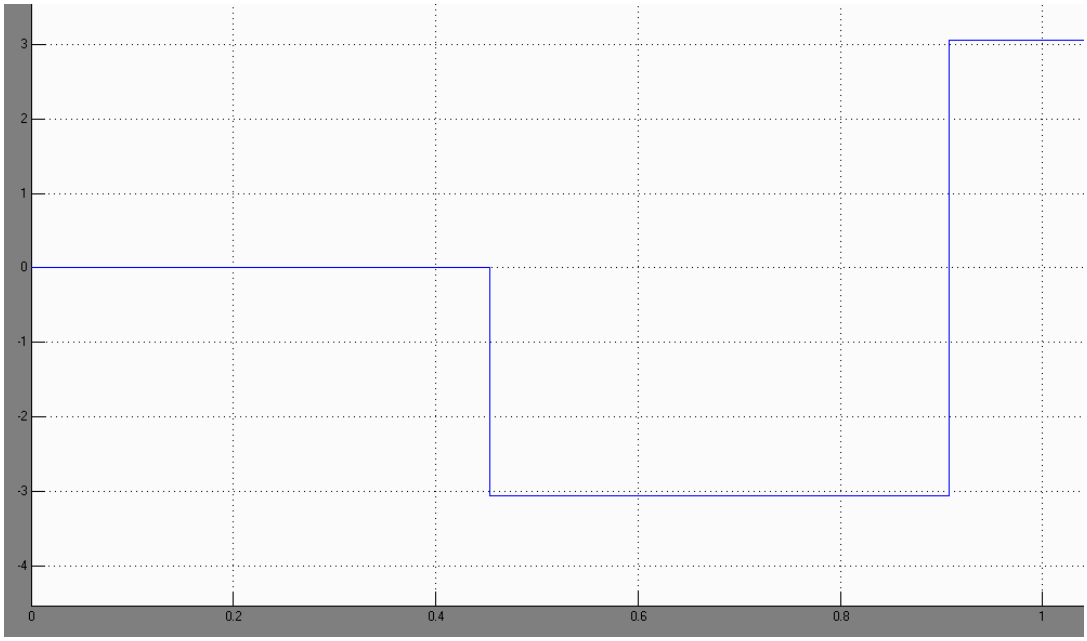


Fig. 6.3. Senyal d'entrada

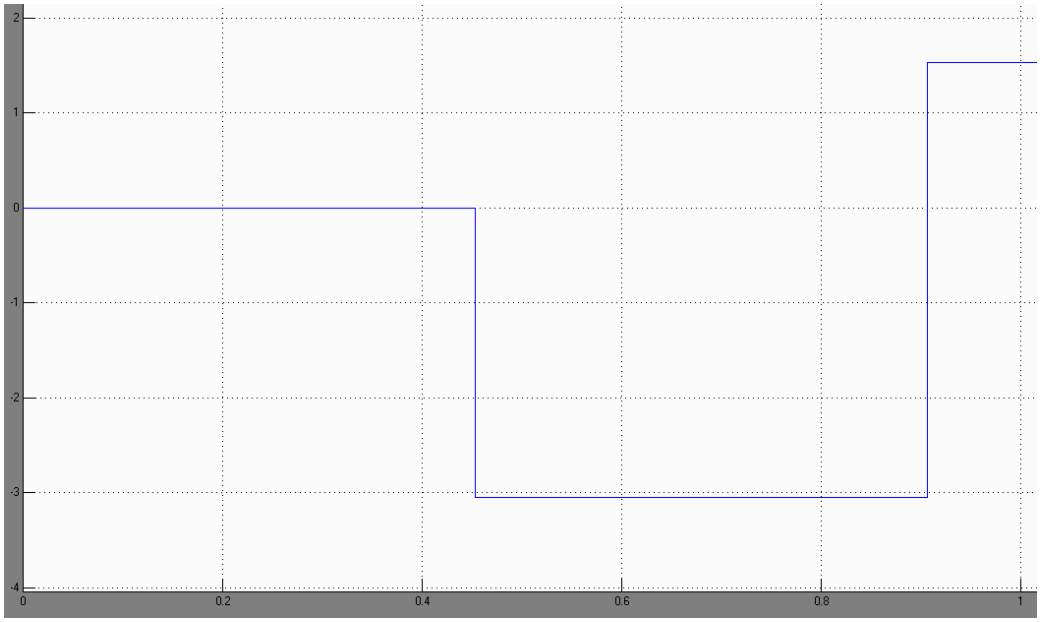


Fig. 6.4. Senyal filtrat

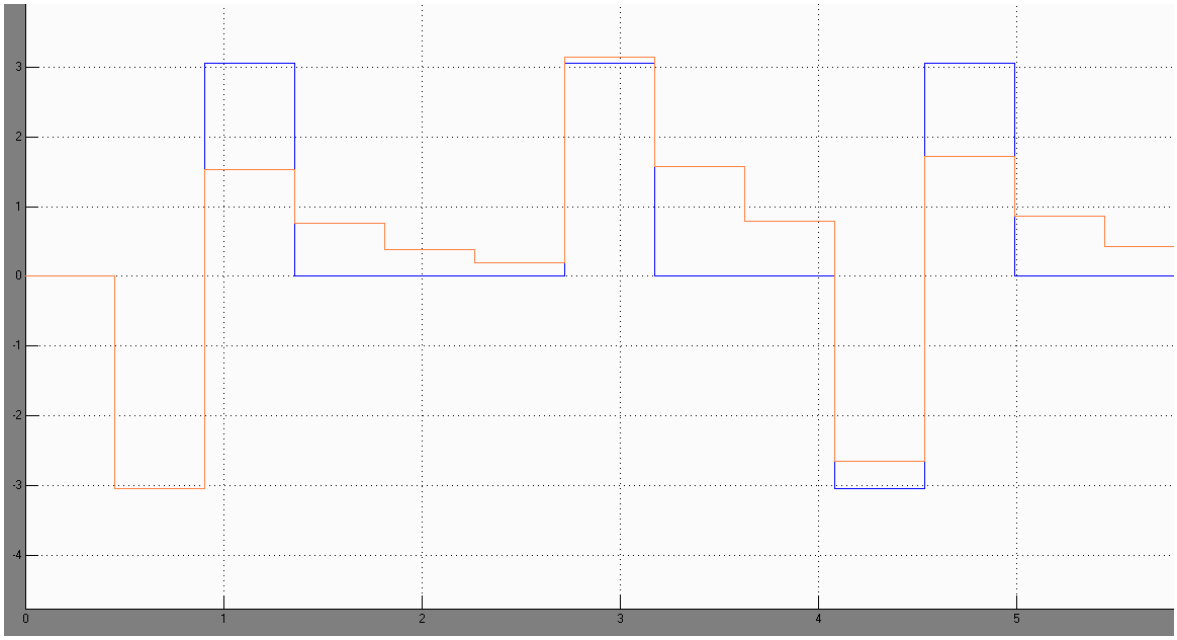


Fig. 6.5. Entrada (blau) i sortida (vermell)

Nota1: augmentar el grau de les z's del denominador provoca més retràs a la senyal de sortida.

Nota2: l'ordre del denominador ha de ser sempre més gran o igual que el del numerador. En cas contrari, l'equació en diferències mostraria que la sortida depèn d'entrades futures, quelcom impossible en un sistema físic real.

7- DFT (Discrete Fourier Transform)

La DFT és la transformada que permet a una senyal discreta passar al domini freqüencial, procés importantíssim en tots els temes relacionats amb el tractament de senyal. I la FFT (Fast Fourier Transform) és una forma més eficient i útil de calcular la DFT ja que estalvia operacions matemàtiques al processador.

S'utilitzarà sempre la FFT perquè així és com treballa el Matlab, però per estudiar el funcionament de la Transformada, ho farem amb la DFT.

Aquesta és la fórmula d'una DFT per N punts:

$$X_N[k] = \sum_{n=0}^{N-1} x_N[n] \cdot e^{-j(2\pi/N)k \cdot n}$$

Com que apareix component imaginari, es pot separar la fórmula en sumes de sinus i cosinus i després sumar-los:

$$X_N[k] = \sum_{n=0}^{N-1} x_N[n] \cdot \cos(-2\pi \cdot k \cdot n / N) + \sum_{n=0}^{N-1} x_N[n] \cdot \sin(-2\pi \cdot k \cdot n / N)$$

A continuació, es dissenyarà un petit programa amb l'editor del Matlab simulant una DFT:

Es disposa d'una entrada N=3 (3 valors complexes que passaran pel programa) i aquests valors estan guardats a un vector 'x'. S'inicialitzen les variables 'rea' (nombres reals) i 'ima' (nombres imaginaris), dient que seran una matriu 3x3 i els seus valors es posen a 0.

S'entra a un bucle *for* des de 0 fins a N-1. Aquest bucle representa les files de la matriu, per tant s'haurà de repetir 3 vegades. Dintre del primer bucle hi ha un altre 'for' que també es repetirà 3 vegades i seran les sumes de les parts real i imaginària de cada valor complex.

El resultat d'aquestes sumes separades s'ajuntaran en la variable 'g'. A partir d'aquí es repeteix el procés per fer les files que queden:

```
N=3;
x=[1-1j 2 3j];
rea=zeros(N);
ima=zeros(N);

for k=0:N-1
for n=0:N-1
rea(k+1)=rea(k+1)+x(n+1)*cos(-2*pi*k*n/N);
ima(k+1)=ima(k+1)+x(n+1)*sin(-2*pi*k*n/N);
end
g(k+1)=rea(k+1)+j*ima(k+1);
end
```

Si s'executa el programa, es comprovarà que la DFT dels valors indicats són els que tenim requadrats:

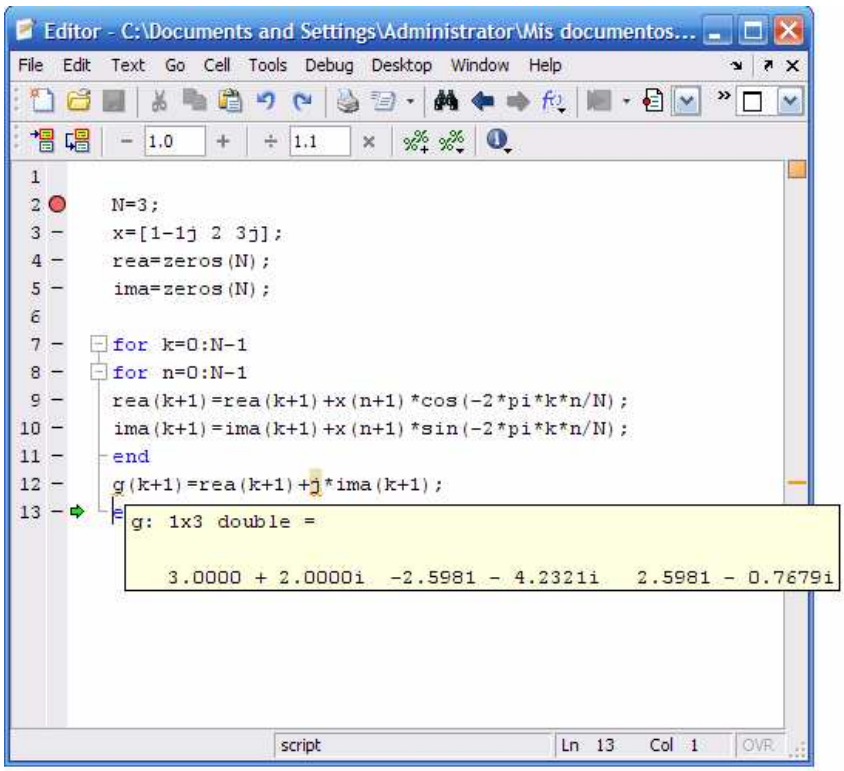


Fig. 7.1. DFT dels valors de 'x'

Per comprovar que els valors són correctes, s'introdueix al Matlab la funció 'fft':

>> fft(x);

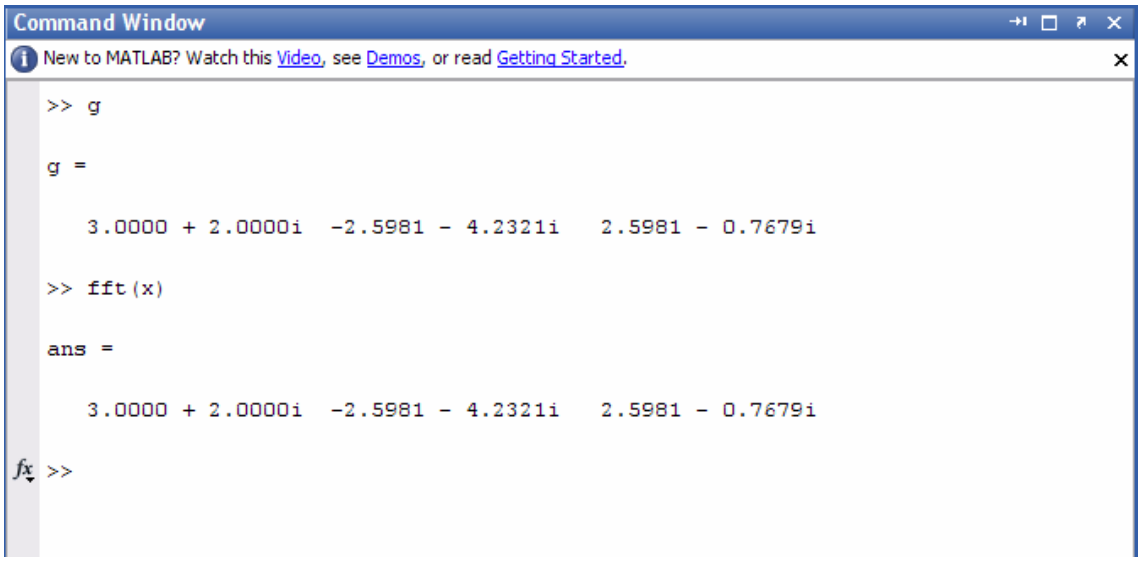


Fig. 7.2. Comprovació resultats

Els resultats són correctes.

8- Propietats dels filtres i dels sistemes en general

- *Memòria*: La sortida depèn d'entrades passades.

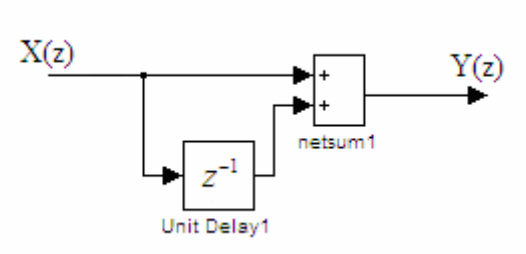


Fig. 8.1. Sistema amb memòria

- *Invertibilitat i sistemes inversos*: Un sistema és invertible si existeix un altre sistema que, col·locat en sèrie amb l'original, produeix una sortida equivalent al valor o valors de l'entrada.

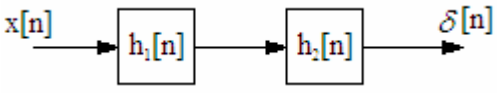


Fig. 8.2. Sistema invers

$$h_1[n] \cdot h_2[n] = \delta[n]$$
$$\delta[n] = x[n]$$

- *Causalitat*: En els sistemes causals (no anticipatius), la sortida només depèn d'entrades presents i passades, no de futures. Tots els sistemes físics reals són no anticipatius.

- *Estabilitat*: Un sistema és estable si la seva sortida a una entrada qualsevol no marxa cap a l'infinit. L'estabilitat la marcarà la posició dels pols del denominador. Si algun pol cau fora de la circumferència unitària el sistema serà inestable.

9- Tipus de filtres

Hi ha dos tipus de filtre i es distingeixen entre ells, entre d'altres coses, per la resposta a l'impuls:

- F.I.R. (Finite Impulse Response).
- I.I.R. (Infinite Impulse Response).

9.1- FIR

Filtres FIR també anomenats no recursius.

La resposta a un impuls (d'una mostra de llargada) de valor 1 (Delta de Dirac) és finita.

Els filtres FIR, a diferència dels IIR, no depenen de valors de sortida passats. Això està directament relacionat amb la presència de tots els pols a l'origen.

Estructura d'un FIR:

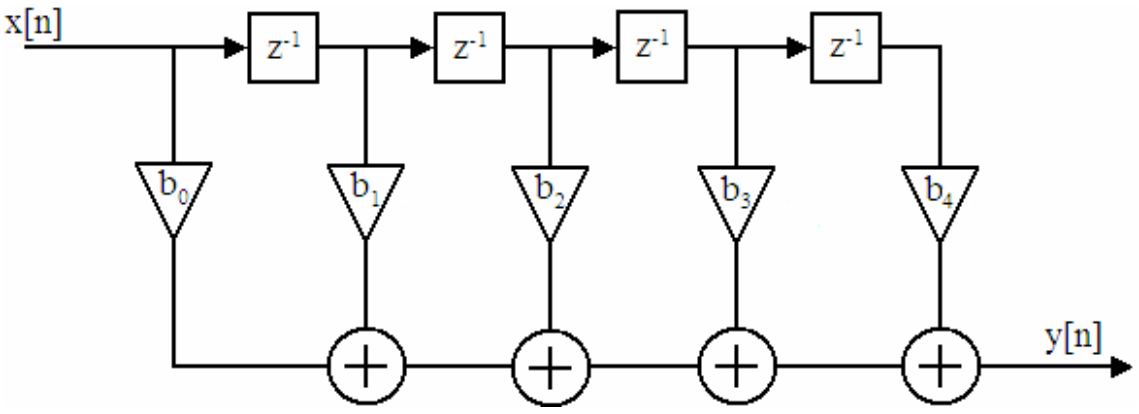


Fig. 9.1.1. Estructura d'un filtre FIR

Exemple: Comprovació que el següent filtre es tracta d'un FIR:

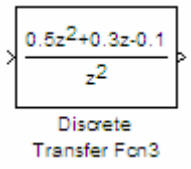


Fig 9.1.2. Funció transferència FIR

Es busca el diagrama de blocs amb els càlculs ja vistos:

$$\frac{Y(z)}{X(z)} = \frac{0,5z^2 + 0,3z - 0.1}{z^2} \rightarrow Y(z)\cdot z^2 = X(z)\cdot 0,5z^2 + X(z)\cdot 0,3z - X(z)\cdot 0,1$$

$$Y(z) = X(z)\cdot 0,5 + X(z)\cdot 0,5z^{-1} - X(z)\cdot 0,1z^{-2}$$

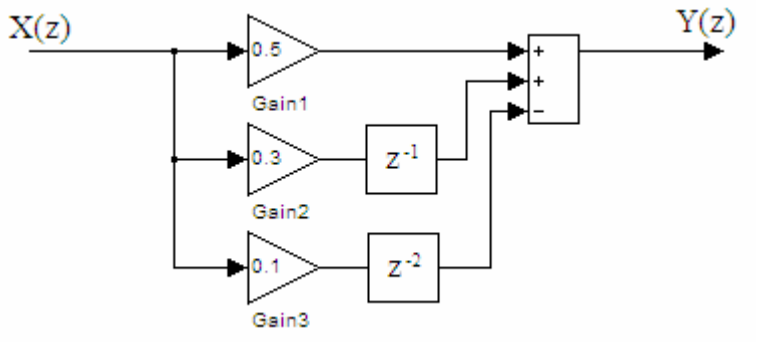


Fig. 9.1.3. Diagrama de blocs FIR

El filtre escollit té tots els pols a l'origen i després de fer el diagrama de blocs s'ha comprovat que la sortida no depèn de valors anteriors seus, sinó que només depèn de valors d'entrada actuals i passats.

Per comprovar que la resposta a l'impuls és finita s'utilitza el Simulink:

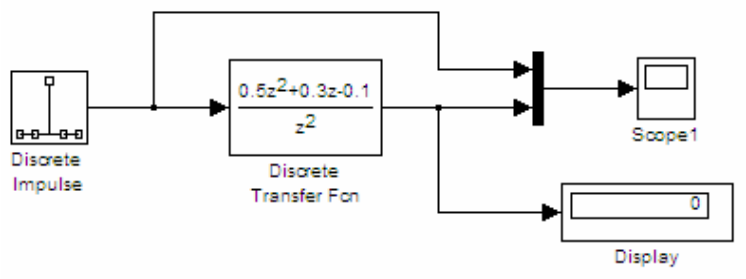


Fig. 9.1.4. Simulació del FIR

Després de simular a una freqüència de mostreig de 100 Hz, apareix el següent:

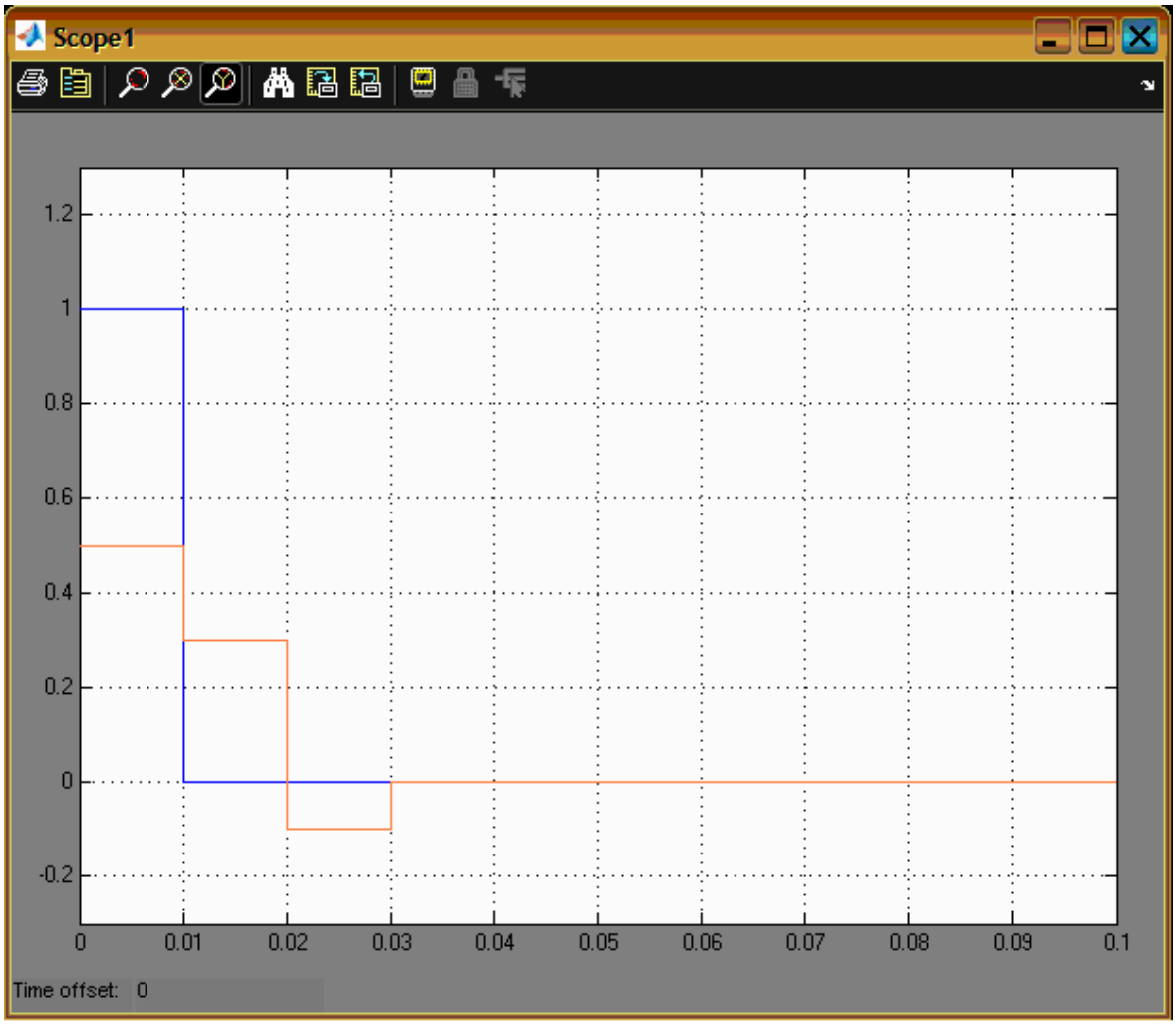


Fig. 9.1.5. Impuls (blau) i resposta a l'impuls (vermell)

La resposta a l'impuls està directament relacionada amb els valors dels coeficients del numerador. Això és una altra característica dels filtre FIR.

Es pot concloure que es tracta d'un FIR perquè la resposta a l'impuls és finita. En aquest cas, tarda 2 mostres a tornar al 0 del Delta de Dirac perquè el filtre és d'ordre 2.

9.2- IIR

La resposta a un Delta de Dirac és infinita.

Els filtres IIR depenen de valors anteriors de la seva sortida i es caracteritzen per la presència de pols fora de l'origen.

Estructura d'un IIR:

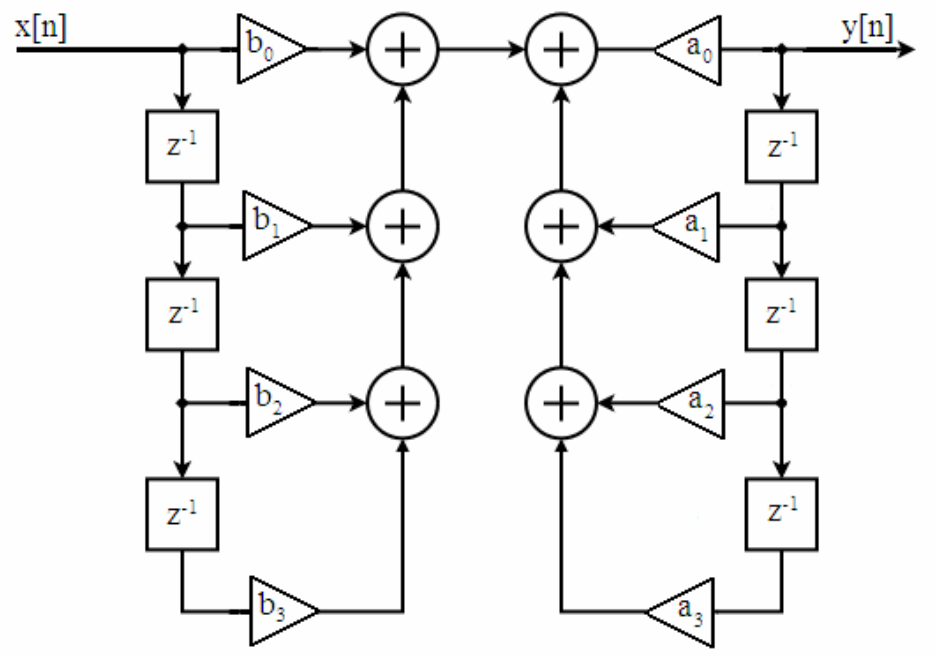


Fig. 9.2.1. Estructura d'un IIR

Exemple: Comprovació que el següent filtre es tracta d'un IIR:

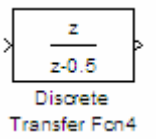


Fig. 9.2.2. Funció transferència IIR

$$\frac{Y(z)}{X(z)} = \frac{z}{z-0,5} \rightarrow Y(z)\cdot z - 0,5\cdot Y(z) = X(z)\cdot z$$

$$Y(z) = X(z) + 0,5\cdot Y(z)\cdot z^{-1}$$

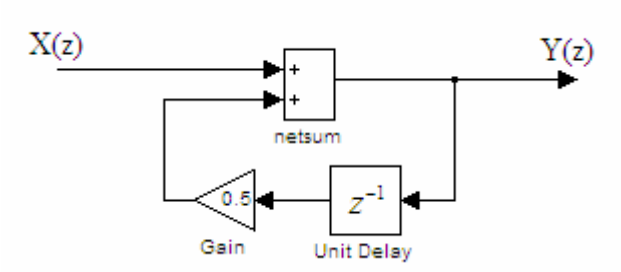


Fig. 9.2.3. Diagrama de blocs IIR

Aquest és el diagrama típic d'un filtre IIR. Es veu com la sortida té una realimentació d'una mostra anterior seva i atenuada a la meitat, provocada per la presència d'un pol fora de l'origen (z=0,5).

Comprovació que la resposta a l'impuls és infinita:

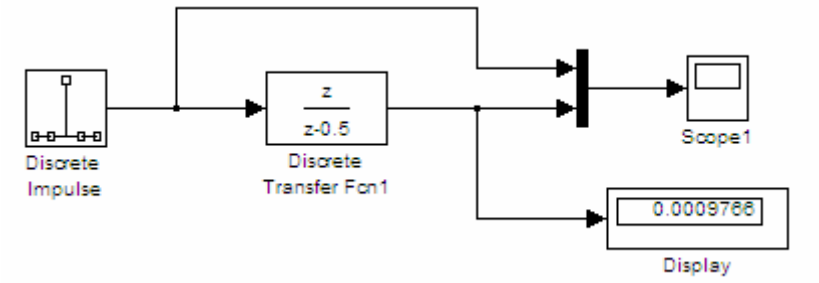


Fig. 9.2.4. Simulació del IIR

El bloc ‘Display’ apareix amb un nombre que no és 0, però s’acosta. Aquest nombre és l’últim valor agafat en el temps de simulació donat (0,1s). La informació que està donant el Display és que el filtre és capaç d’acostar-se a gairebé 1 mil·lèsima del 0 en un temps de 0,1s. Com que el temps de mostreig és de 0,01s, llavors la sortida ha assolit el valor de 0,0009766 en 10 mostres (0,01s ·10 = 0,1s).

Com més temps de simulació es doni al circuit, més s’acostarà aquest valor a 0.

Després de simular a 100 Hz, apareix el següent:

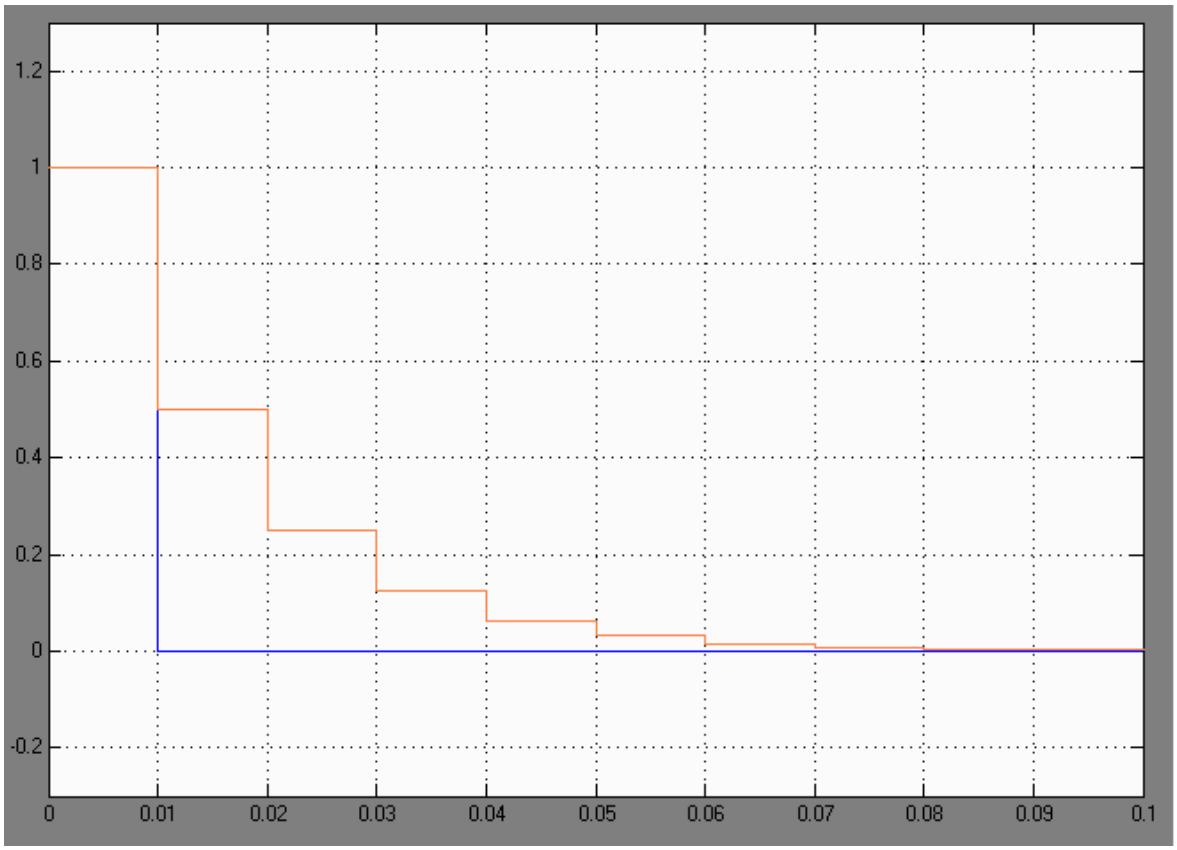


Fig. 9.2.5. Impuls (blau) i resposta a l’impuls (vermell)

Com s’ha explicat anteriorment, aquests tipus de filtres no retornen mai a 0. Si es fes un zoom a l’últim valor d’aquesta simulació apareixeria el valor del ‘Display’. Si es deixés més temps de simulació es comprovaria com el valor de sortida s’acostaria cada cop més a 0.

10- Convolució

En tractament de senyals, una convolució és el resultat de superposar dues senyals amb desplaçament. La convolució té múltiples aplicacions en diferents camps de la ciència i en el tema a tractar, té una importància fonamental. De fet, la fórmula de la Suma de Convolució permet determinar la resposta temporal d'un sistema a una entrada qualsevol a partir de la seva resposta a l'impuls.

Fórmula Suma de Convolució:

$$y[n] = \sum_{k=-\infty}^{+\infty} (x[k] \cdot h[n-k])$$

on y[n]: valor de la sortida del filtre en cada punt
x[k]: amplitud de les entrades
h[n-k]: resposta a l'impuls del filtre en cada punt

Exemple:

Disposem d'un filtre com el de la figura i es vol calcular la suma de convolució als punts indicats. A l'entrada es disposa de 3 valors (0,8 , 1,5 i 1,1) i se sap la resposta a l'impuls del sistema, com es pot veure al dibuix. Quina convolució hi haurà als punts 1, 2 , 3 i 4 de la sortida?
Nota: s'han agafat les quatre primeres respostes del sistema a l'impuls de les infinites que té (es tracta d'un IIR).

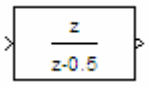
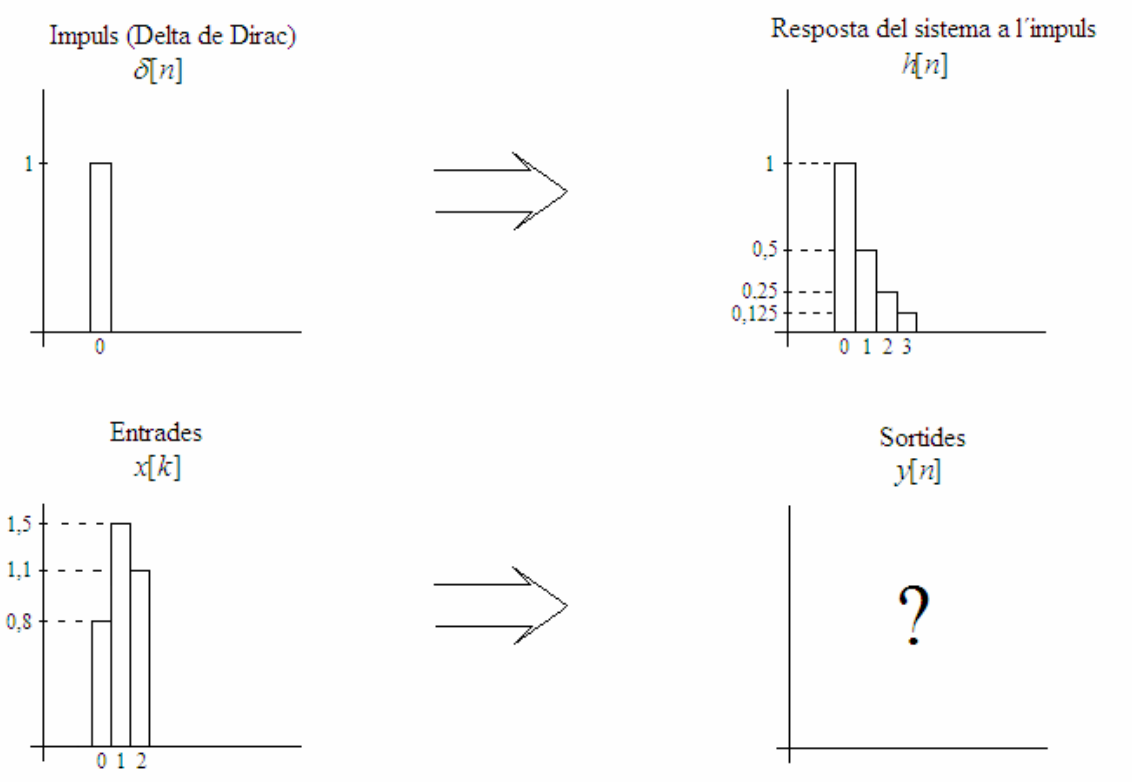


Fig. 10.1. Exemple per calcular la convolució



S'aplica la suma de convolució:

$$y[1] = (x[0] \cdot h[1-0]) + (x[1] \cdot h[1-1]) + (x[2] \cdot h[1-2])$$
$$y[1] = (0,8 \cdot 1) + (1,5 \cdot 0) + (1,1 \cdot 0) = 0,8$$

$$y[2] = (x[0] \cdot h[2-0]) + (x[1] \cdot h[2-1]) + (x[2] \cdot h[2-2])$$
$$y[2] = (0,8 \cdot 0,5) + (1,5 \cdot 1) + (1,1 \cdot 0) = 1,9$$

$$y[3] = (x[0] \cdot h[3-0]) + (x[1] \cdot h[3-1]) + (x[2] \cdot h[3-2])$$
$$y[3] = (0,8 \cdot 0,25) + (1,5 \cdot 0,5) + (1,1 \cdot 1) = 2,05$$

$$y[4] = (x[0] \cdot h[4-0]) + (x[1] \cdot h[4-1]) + (x[2] \cdot h[4-2])$$
$$y[4] = (0,8 \cdot 0,125) + (1,5 \cdot 0,25) + (1,1 \cdot 0,5) = 1,025$$

$$y[5] = (x[0] \cdot h[5-0]) + (x[1] \cdot h[5-1]) + (x[2] \cdot h[5-2])$$
$$y[5] = (0,8 \cdot 0) + (1,5 \cdot 0,125) + (1,1 \cdot 0,25) = 0,4625$$

Es pot fer la comprovació amb el Simulink:

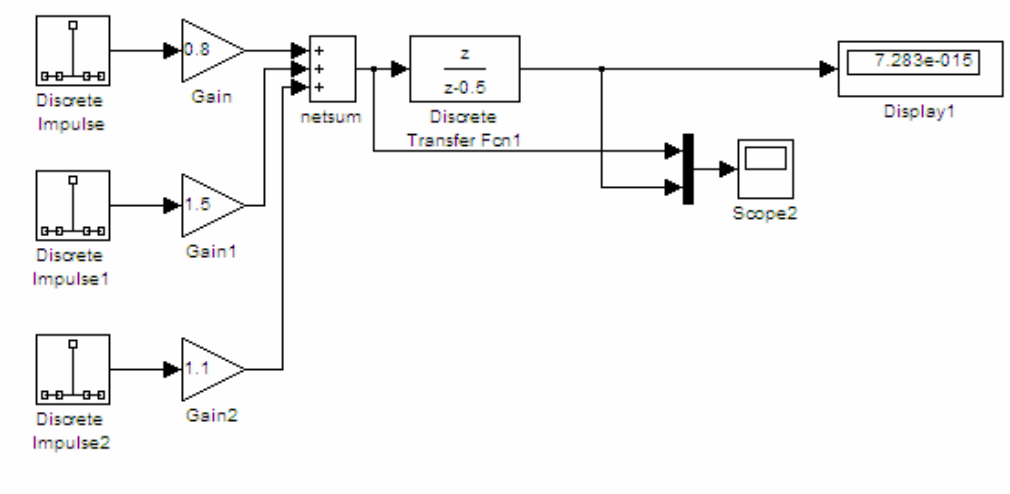


Fig. 10.2. Simulació del exemple donat

Doble clic al 'Scope':

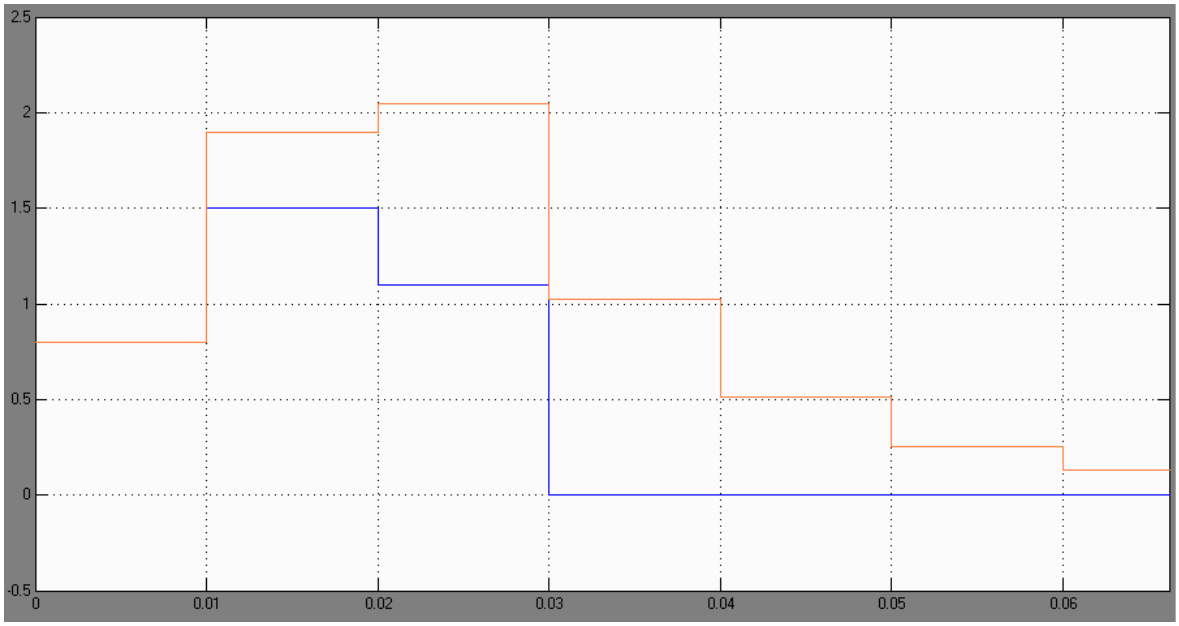


Fig. 10.3. Entrades (blau) i sortides (vermell)

Els resultats coincideixen.

11- Anàlisi freqüencial d'un filtre FIR per diagrama de Bode

Es vol analitzar el comportament del següent filtre:

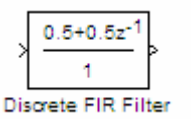


Fig. 11.1. Filtre FIR

Per veure el seu comportament freqüencial s'utilitza el Matlab i s'introdeixen les següents ordres:

```
>> num = [0.5 0.5];
>> den = [1 0];
>>H = tf(num,den,(1/44100));
>>bode (H)
```

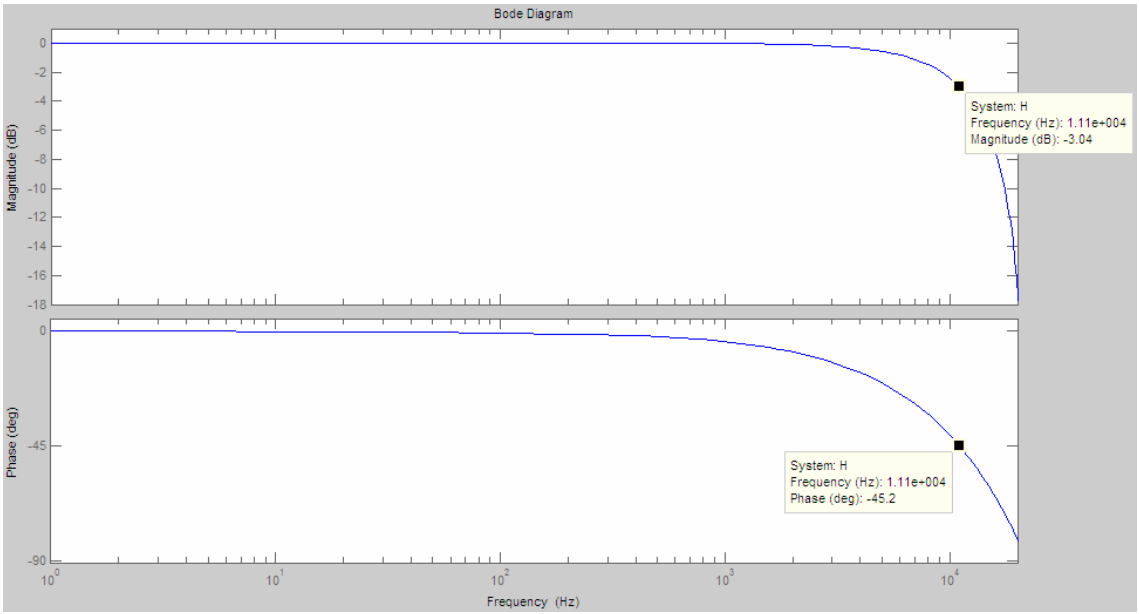


Fig. 11.2. Diagrama de Bode

El diagrama de Bode indica que el filtre es comporta com un passa-baixos. A freqüències de menys de 1000 Hz, no hi ha atenuació ni amplificació de la senyal d'entrada, amb un desfasament d'uns pocs graus a mesura que s'acosta als 1000 Hz.

En canvi, de 4000 Hz cap amunt, el Guany cau en picat i a partir dels 1000 Hz el desfasament augmenta considerablement. El punt marcat a la gràfica és la freqüència de tall aproximada, que ve d'aquí:

$f_{tall} \rightarrow 70{,}71\%$ (Guany de referència = $0dB$).

$20 \cdot \log(0{,}7071) = -3{,}01dB$

També se sap que aquest filtre de primer odre té un pendent de -20 dB/dècada degut a l'efecte del pol del denominador situat a l'origen de la circumferència unitària (característic dels FIRs).

Si es desitja visualitzar la situació dels pols i zeros, s'escriu el següent:

```
>> zplane(num,den)
```

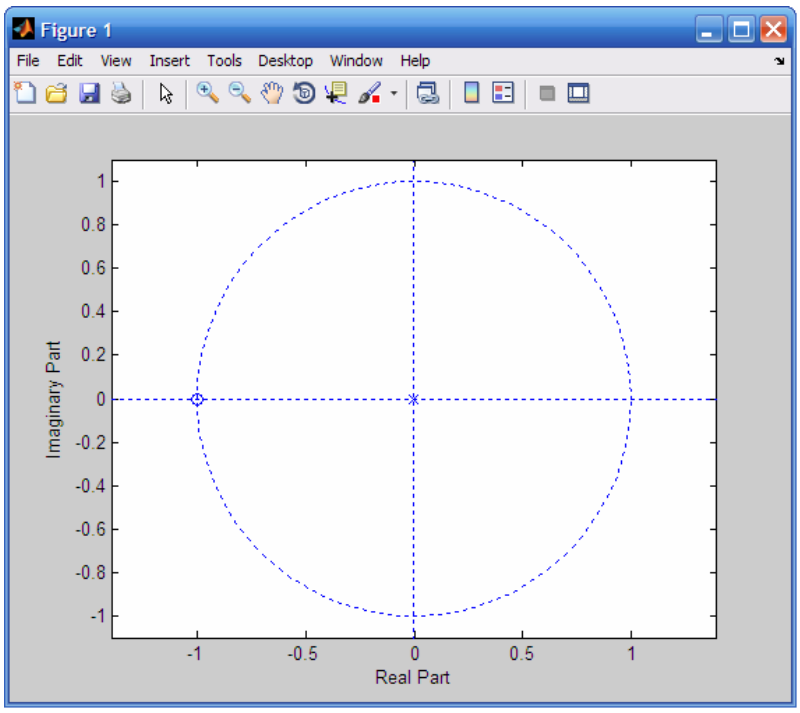


Fig. 11.3. Situació de pols i zeros

11.1- Efecte del FIR a una senyal d'entrada

Ara s'analitzarà freqüencialment l'anterior FIR per una entrada creada amb el programa 'Audacity' que varia 1.000 Hz per segon, des de 10.000 fins a 15.000 Hz. Es fa la FFT a la senyal d'entrada i es visualitza el mòdul de la FFT:

```
>> [y,Fs,bits] = wavread ('10000-15000');
>> affty = (abs(fft(y)));
>>plot (affty)
```

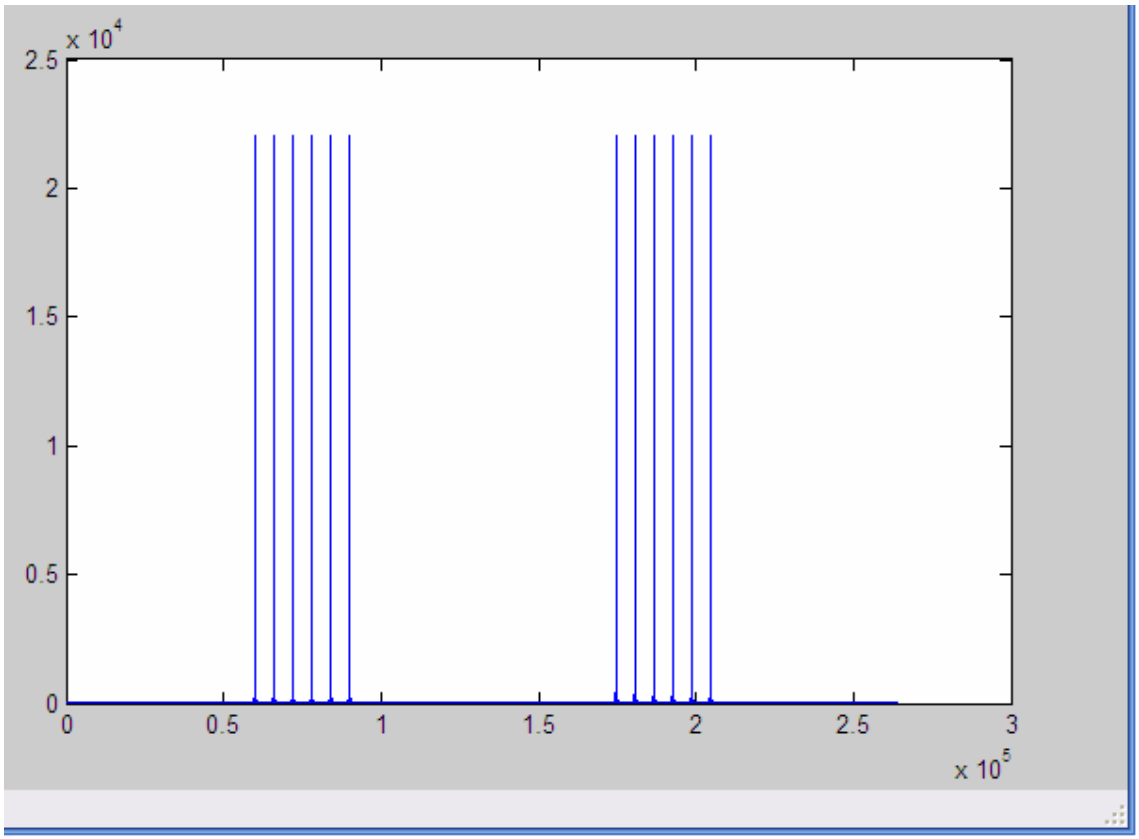


Fig. 11.1.1. Mòdul FFT de 'y'

La figura no mostra directament la informació en Hz i Volts, sinó en forma de períodes, com ve per defecte en el Matlab. També la imatge ve duplicada per la senyal simètrica.

Per veure la FFT de la manera que interessa fem el següent:

```
>> frequencia = [0 : 1/((264600-1)/44100) : 44100];
>> volts = (affty*2)/264600;
>>plot (frequencia,volts)
```

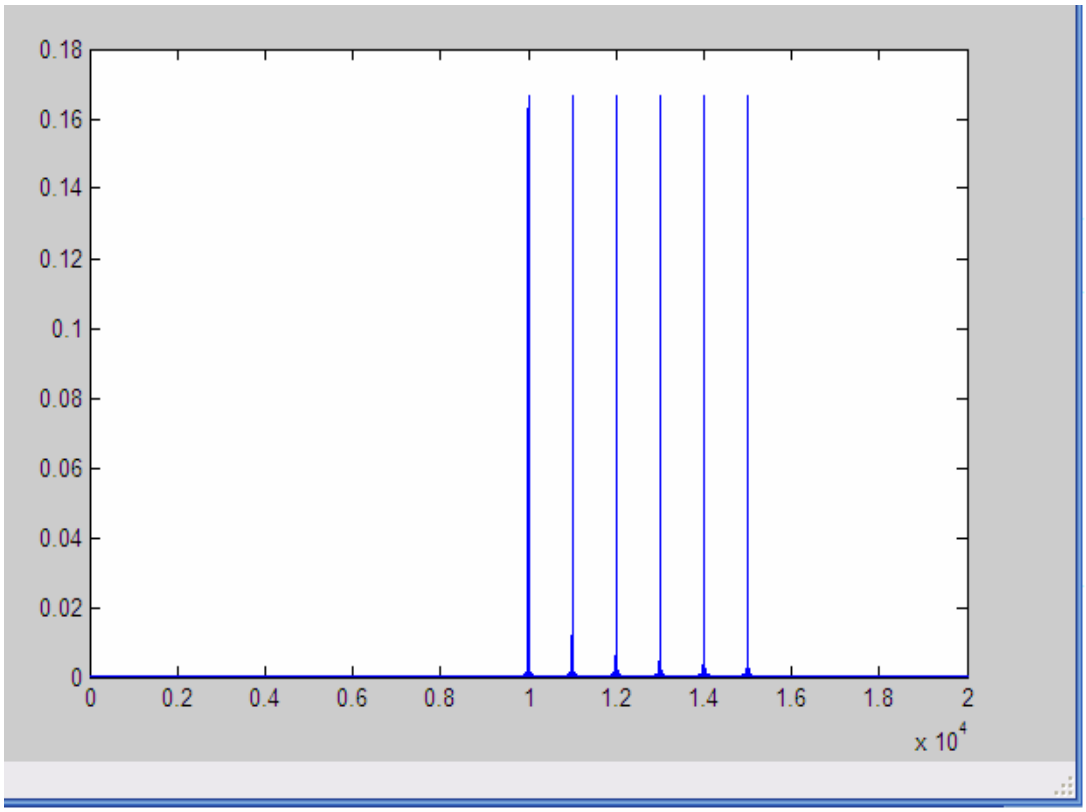


Fig. 11.1.2. Mòdul FFT de 'y' arreglat

Ara els eixos són els correctes i s'ha eliminat la senyal simètrica.

Ara es filtrarà la senyal amb el FIR anterior. Però abans, s'arreclarà la funció de transferència del filtre per evitar que hi hagi z's amb exponent negatiu (per facilitar la introducció de les instruccions al Matlab):

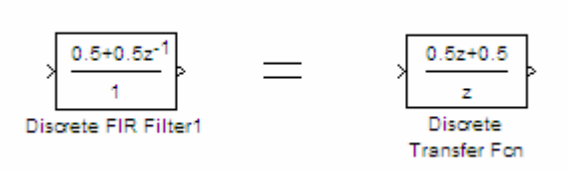


Fig. 11.1.3. Transformació a z's positives

S'introdueixen les dades al Matlab:

```
>> numerador = [0.5 0.5];
>>denominador = [1 0];
>> fir = filter(numerador, denominador, y);
>> afftfir = (abs(fft(fir)));
>>frequencia = [0 : 1/((264600-1)/44100) : 44100];
>> volts=(afftfir*2)/264600;
>> plot(frequencia, volts)
```

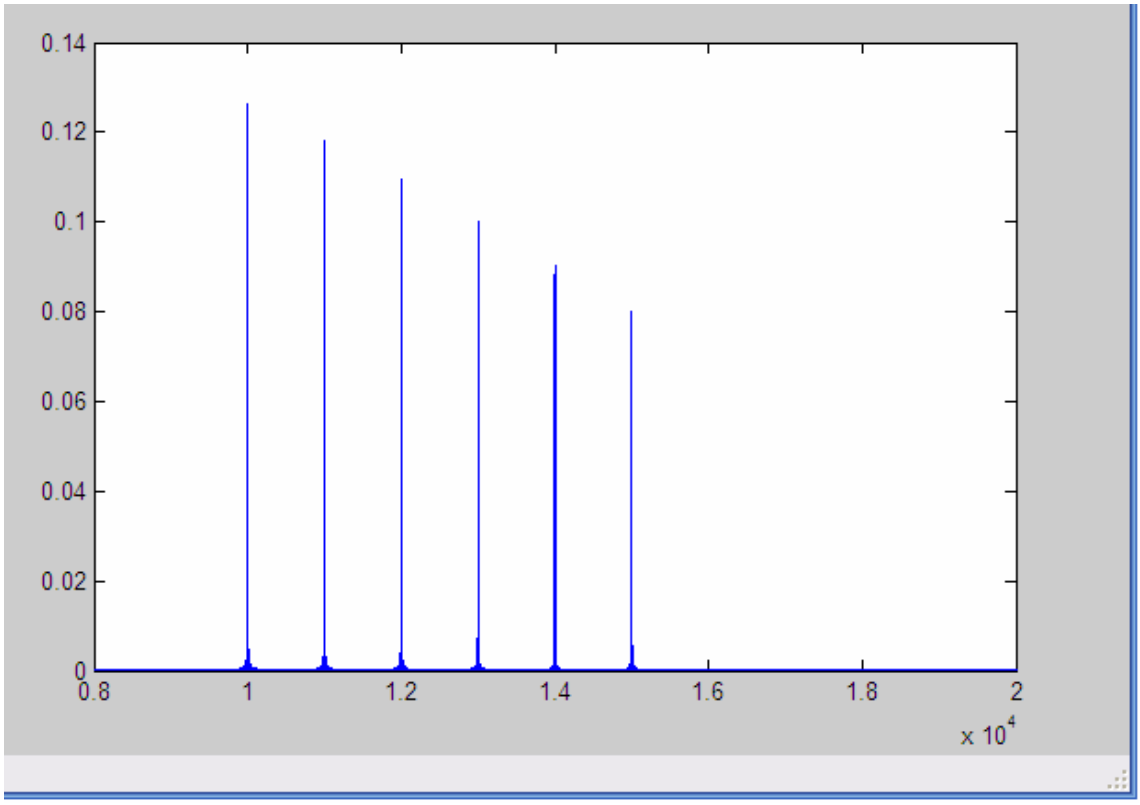


Fig. 11.1.4. Senyal filtrat

12- Anàlisi freqüencial d'un filtre IIR per diagrama de Bode

Estudi del comportament del següent filtre:

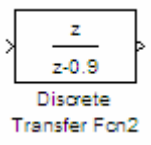


Fig. 12.1. Filtre IIR

Els paràmetres del Simulink són aquests:

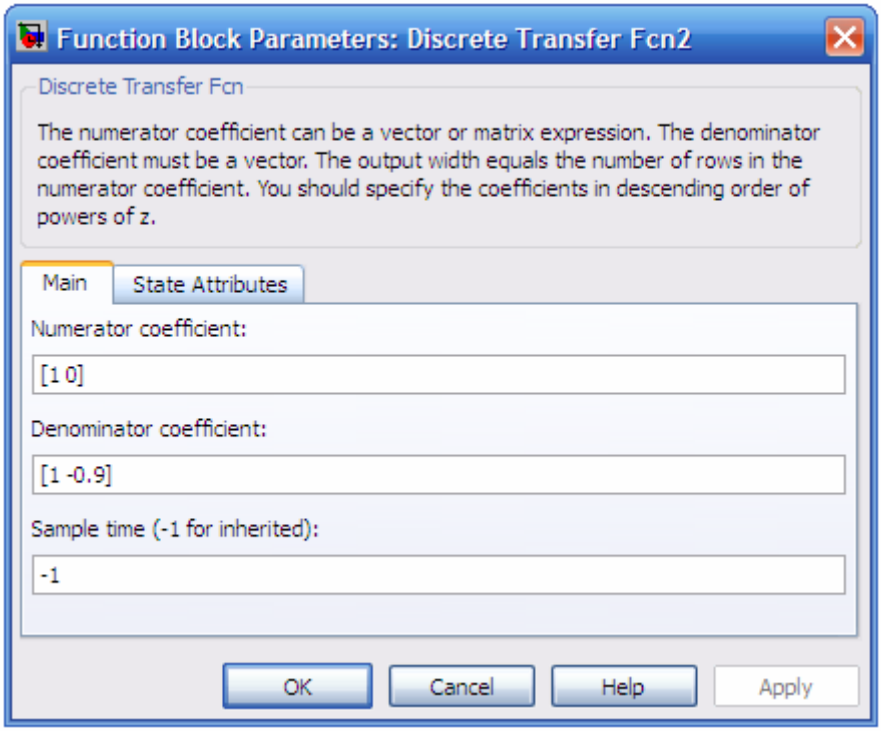


Fig. 12.2. Coeficients del sistema

El Sample time "-1" significa que el temps de mostreig del filtre és el mateix que el de la senyal d'entrada que se li connecta.

Per veure el Bode, s'escriu al Matlab:

```
>> num = [1 0];
```

```
>> den = [1 -0.9];
>> H = tf (num, den, (1/44100));
>> bode (H)
```

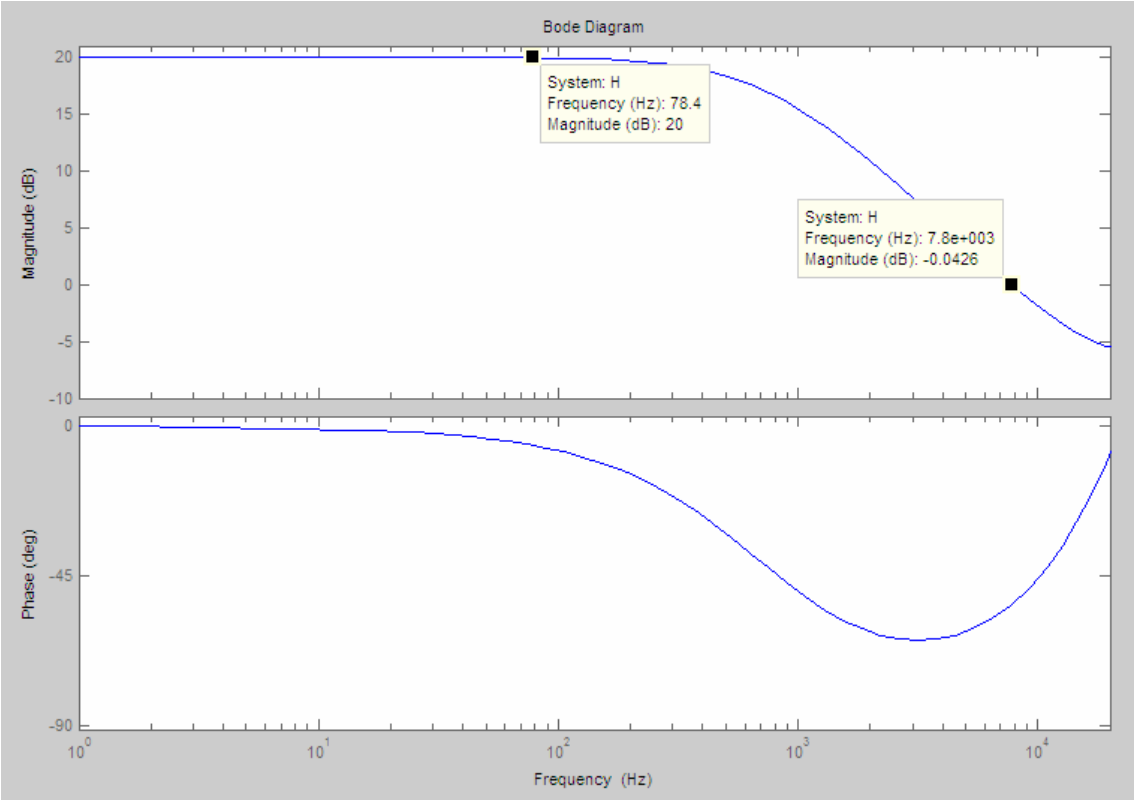


Fig. 12.3. Diagrama de Bode

Com es pot veure, el filtre escollit amplifica 20 dB entre els 1 i els 78 Hz i deixa d'amplificar als 7.800 Hz. A partir d'aquí, el filtre atenua el senyal de forma moderada.
Per tant, es tractaria d'un filtre amplificador de baixes freqüències i atenuador de molt altes freqüències.

Si es desitja que el filtre no amplifiqui a cap freqüència, se li pot posar un multiplicador que redueixi a 0 el seu guany.

$$\frac{z}{z-0,9} \rightarrow 1-0,9=0,1 \rightarrow k=0,1$$

Teorema del valor final:

S'agafa el valor que multiplica la z del numerador i se li resta el valor del pol del denominador. El resultat (0,1) és el factor que, multiplicat pel filtre, farà que aquest deixi de tenir guany. Per altra banda, és lògic que el valor sigui 0,1 perquè $20 \cdot \log 10 = 20dB$, per tant, per contrarestar un guany multiplicador de 10, farà falta un atenuador de 0,1.

Els nous paràmetres del Simulink seran aquests:

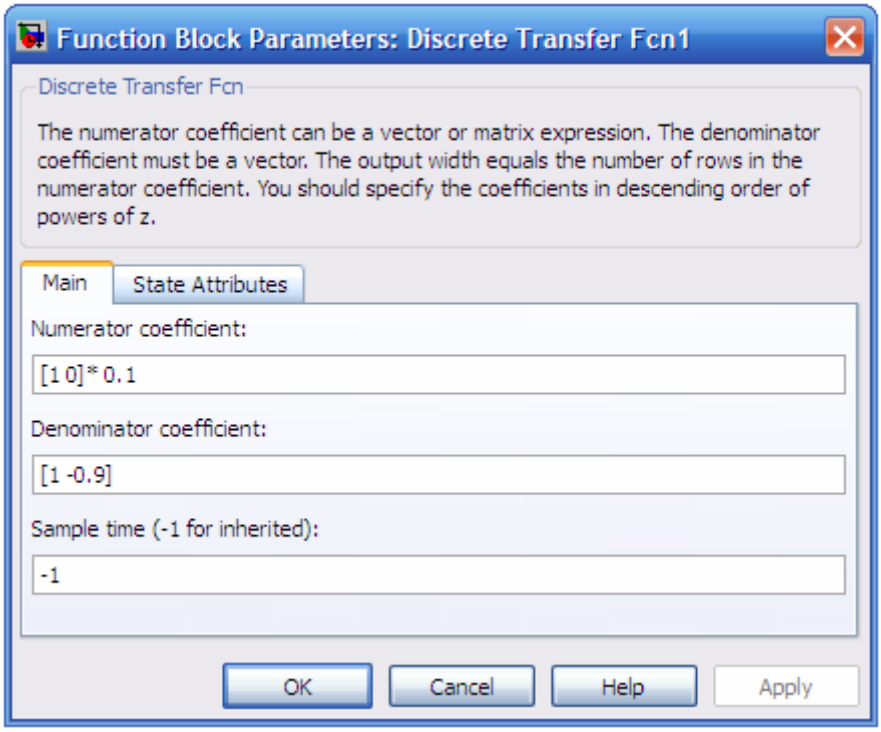


Fig. 12.4. Nous paràmetres del sistema

I les noves instruccions pel Matlab:

```
>> num = [0.1 0];
>> den = [1 -0.9];
>> H = tf (num, den, (1/44100));
>> bode (H)
```

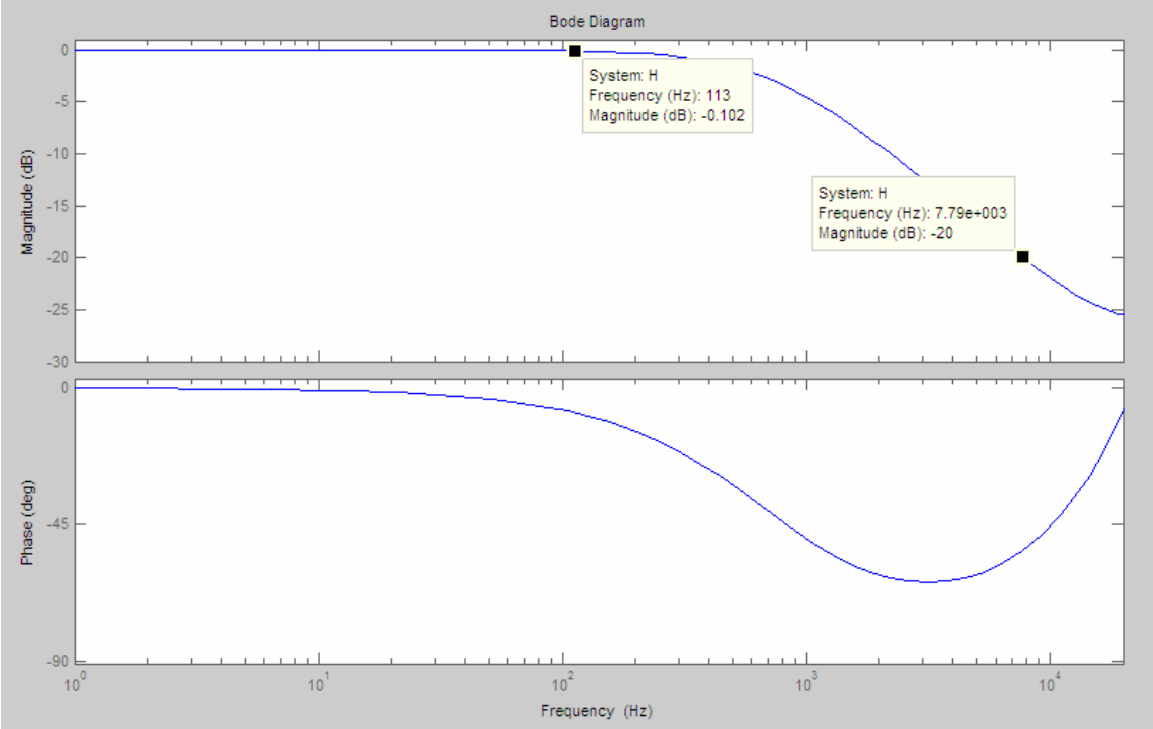


Fig. 12.5. Diagrama de Bode

Com es pot veure és el mateix diagrama que l'anterior però atenuat 20 dB. A partir de 113 Hz comença a tenir pèrdua significativa i com a anècdota, el punt on abans hi havia 0 dB (7.800 Hz), ara es tenen -20dB degut a l'atenuació que suposa multiplicar per 0,1 la funció de transferència.

L'ordre 1 del filtre fa que el pendent sigui de -20 dB/dècada. La situació de pol i del zero és aquesta:

>> zplane(num,den)

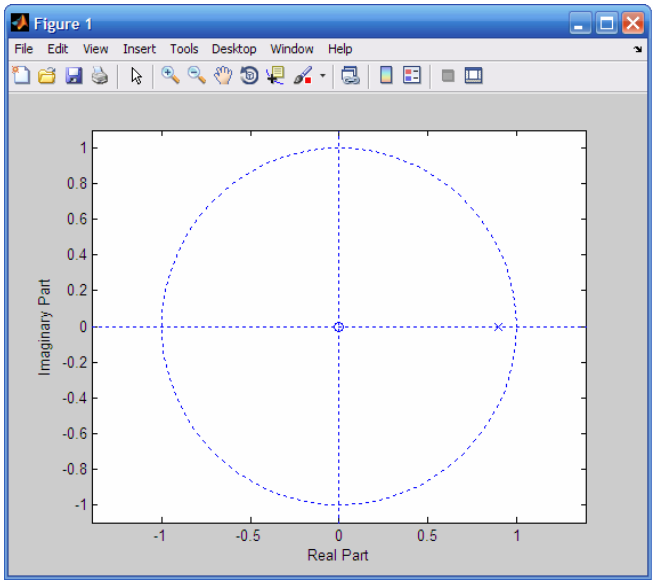


Fig. 12.6. Situació de pols i zeros

12.1- Efecte del IIR a una senyal d'entrada

Ara s'analitzarà freqüencialment l'anterior IIR per una senyal d'entrada creada amb el 'Audacity' que varia 1.000 Hz per segon, des de 1.000 fins a 5.000 Hz. Com s'ha fet abans, es fa la FFT a la senyal d'entrada i es visualitza el mòdul de la FFT:

```
>> [y,Fs,bits] = wavread ('1000-5000');
>> affty = (abs(fft(y)));
>>plot (affty)
```

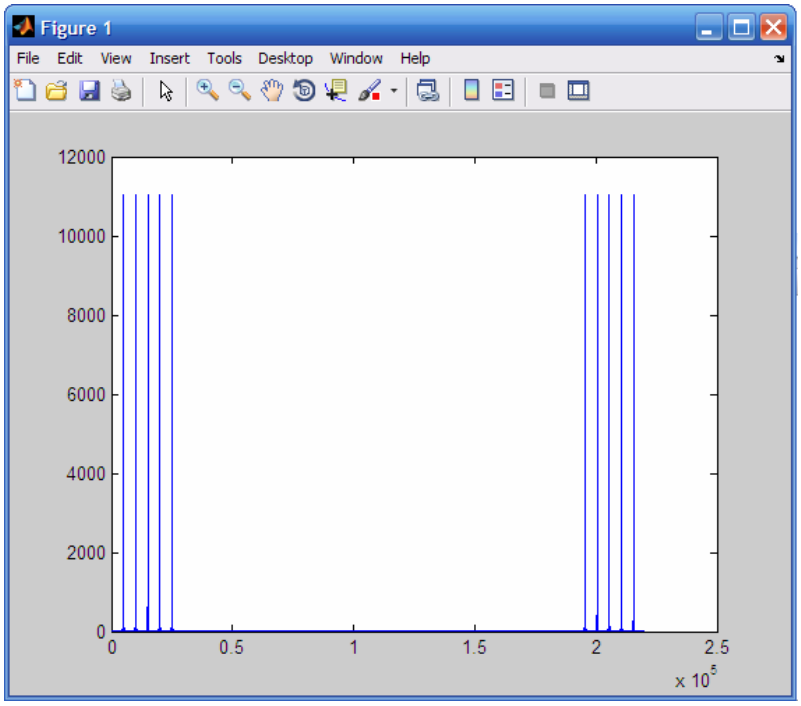


Fig. 12.1.1. Mòdul FFT de 'y'

Per veure la FFT de la manera que interessa fem el següent:

>> frequencia = [0 : 1/((264600-1)/44100) : 44100];

>> volts = (affty*2)/264600;

>>plot (frequencia,volts)

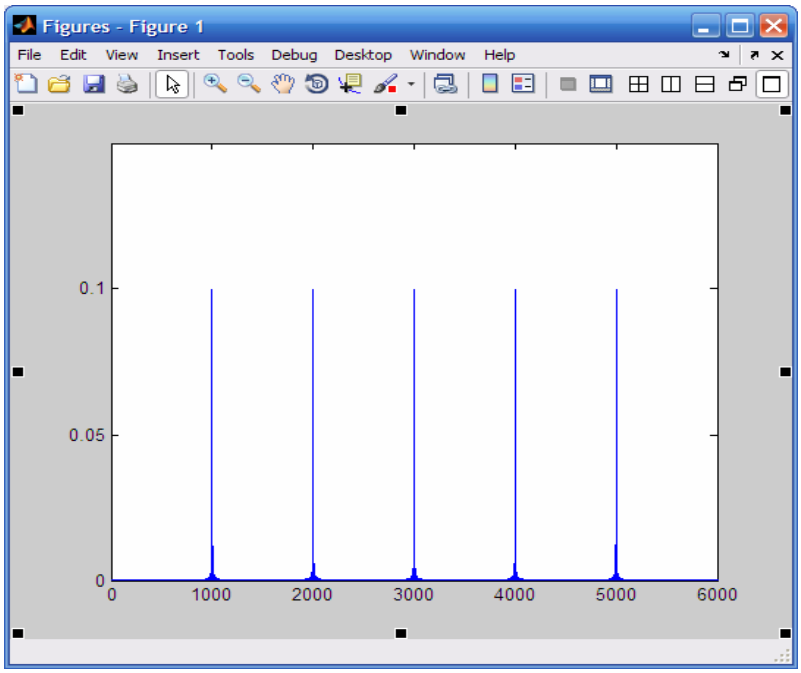


Fig. 12.1.2. Mòdul FFT de 'y' arreglat

S'introdueixen les dades al Matlab:

```
>> numerador = [1 0];
>>denominador = [1 -0.9];
>> fir = filter(numerador, denominador, y);
>> afftfir = (abs(fft(fir)));
>>frequencia = [0 : 1/((264600-1)/44100) : 44100];
>> volts=(afftfir*2)/264600;
>> plot(frequencia, volts)
```

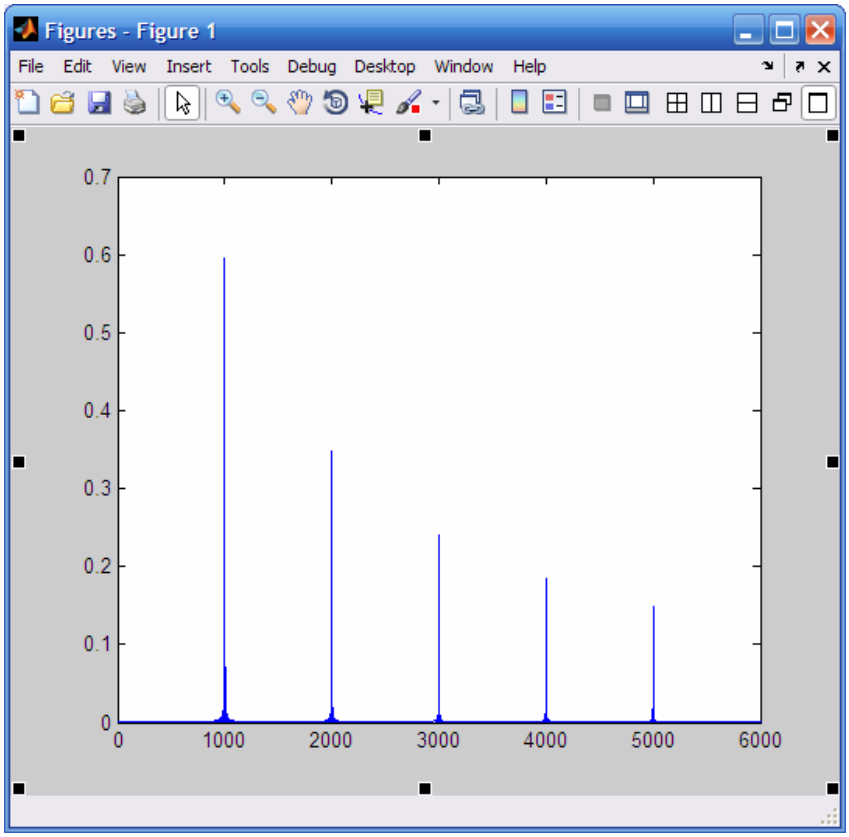


Fig. 12.1.3. Senyal filtrat

13- Disseny d'un FIR per finestra de Hamming

El Matlab té diverses mètodes a l'hora de dissenyar un filtre FIR amb unes especificacions determinades. De tots els mètodes disponibles, s'agafarà el "mètode de les finestres" per esbrinar els coeficients adients perquè el filtre faci el que se li demani.

El mètode que se segueix pel disseny és la convolució de la resposta a l'impuls d'un filtre ideal amb els coeficients d'una finestra, en aquest nostre cas, la finestra de Hamming.

Un filtre passa-baixos ideal té aquesta forma:

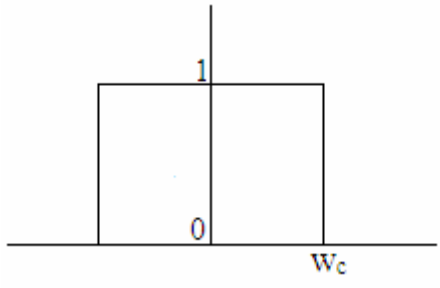


Fig. 13.1. Filtre passa-baixos ideal

La resposta a l'impuls d'aquest filtre ideal és la funció 'sinc':

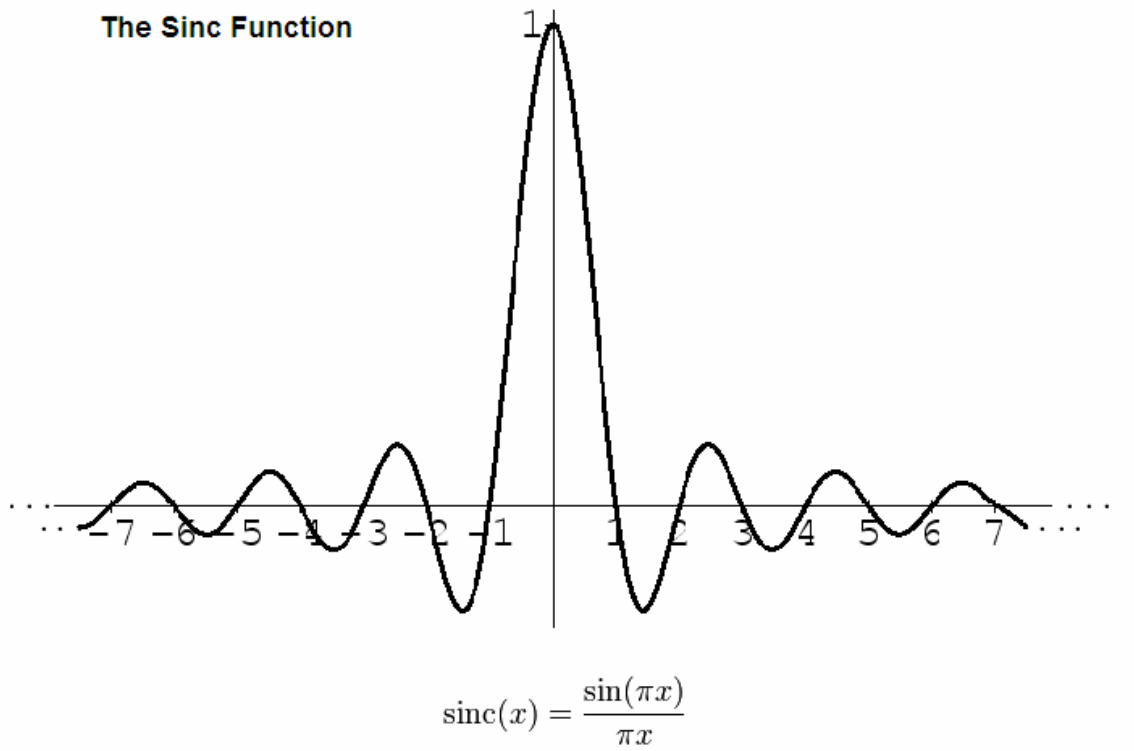


Fig. 13.2. Funció 'sinc'

Per altra banda, es disposa de l'equació de la finestra de Hamming. És aquesta:

$$w(n) = 0,54 - 0,46 \cdot \cos\left(\frac{2\pi n}{N}\right) \qquad \text{on } 0 \le n \le N$$

Si s'agafen 5 punts:

$$w(0) = 0,54 - 0,46 \cdot \cos\left(\frac{2\pi \cdot 0}{4}\right) = 0,08$$

$w(1) = 0,54$

$w(2) = 1$

$w(3) = 0,54$

$w(4) = 0,08$

El Matlab calcula els coeficients de Hamming amb aquesta instrucció:

>> hamming(L); on L=N-1

Fent un plot es veuria la finestra de Hamming de 5 punts.

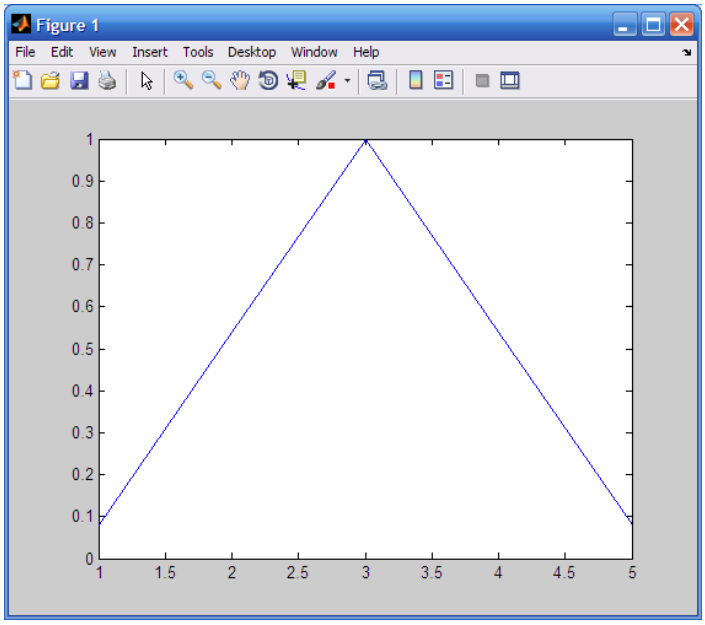


Fig. 13.3. Finestra de Hamming de 5 punts

I si s'agafen 300 punts:

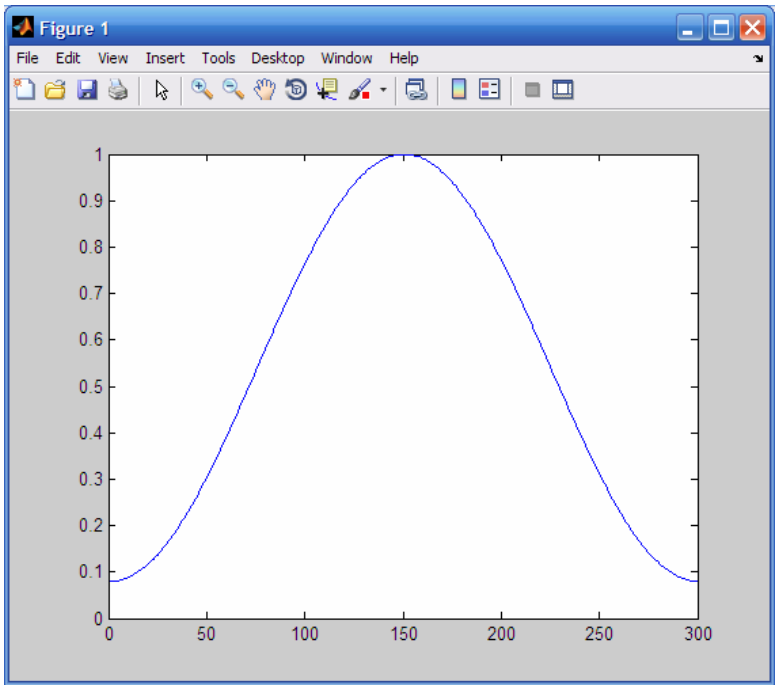


Fig. 13.4. Finestra de Hamming de 300 punts

El mètode que se segueix es basa en la multiplicació dels coeficients de Hamming i la funció sinc amb un desplaçament concret.

La raó d'utilitzar una finestra és perquè permet acotar la zona de freqüències desitjada de tal manera que si es desplaça la finestra de forma adequada s'aconsegueix l'efecte de deixar o no deixar passar el senyal.

Exemple: es vol dissenyar un filtre FIR passabanda entre 1000 i 5000 Hz d'un arxiu d'àudio, amb l'objectiu d'eliminar les freqüències baixes i les molt altes.

Abans de tot, s'ha de dir-li al Matlab l'ordre del filtre (que en realitat seran el nombre de coeficients que retornarà la funció) i la freqüència de mostreig. Després s'utilitzarà la funció fir1 (mètode de les finestres de Hamming) per esbrinar els coeficients dels zeros del filtre:

```
>> N=500;
>> Fs=44100;
>> Fny=Fs/2;
>> Coef=fir1(N,[1000 5000]/Fny, 'bandpass');
```

Nota: les freqüències s'han d'indicar normalitzades respecte la meitat de la freqüència de mostreig, és a dir, que hauran d'estar entre 0 i 1, sent 1 la meitat de la Fs.

A continuació, es fa la resposta freqüencial, indicant el numerador del sistema (Coef), el denominador (1, ja que es tracta d'un FIR), l'escala de l'eix de les freqüències (F) i la freqüència de mostreig.

```
>> num=Coef;
>> den=1;
>> F=0:1:10000;
>> freqz(num,den,F,Fs)
```

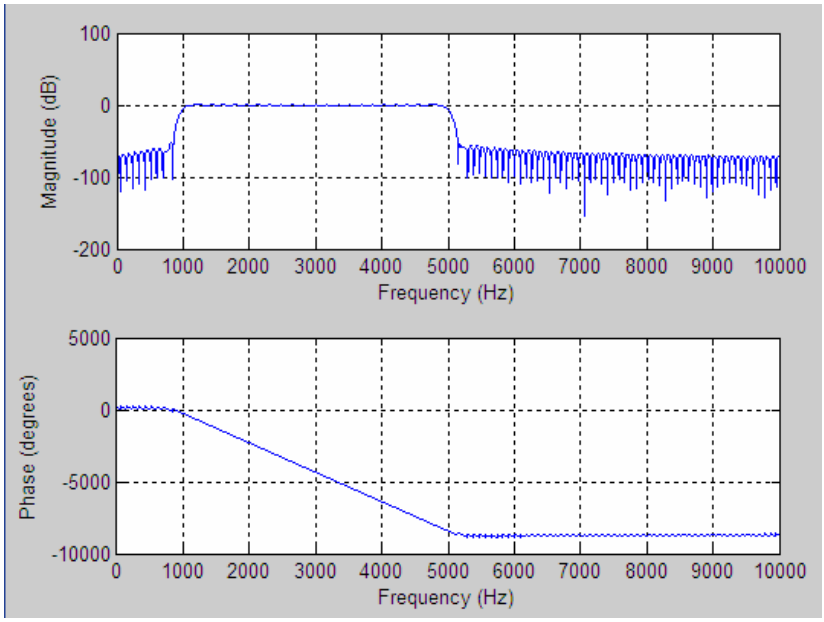


Fig. 13.5. Resposta freqüencial

També es pot fer el mòdul de la resposta freqüencial (donada en números complexes) i es tindria el mateix resultat però sense ser en dB:

```
>> Mrespf=abs(freqz(Coef,1,F,Fs));
>> plot(Mrespf)
```

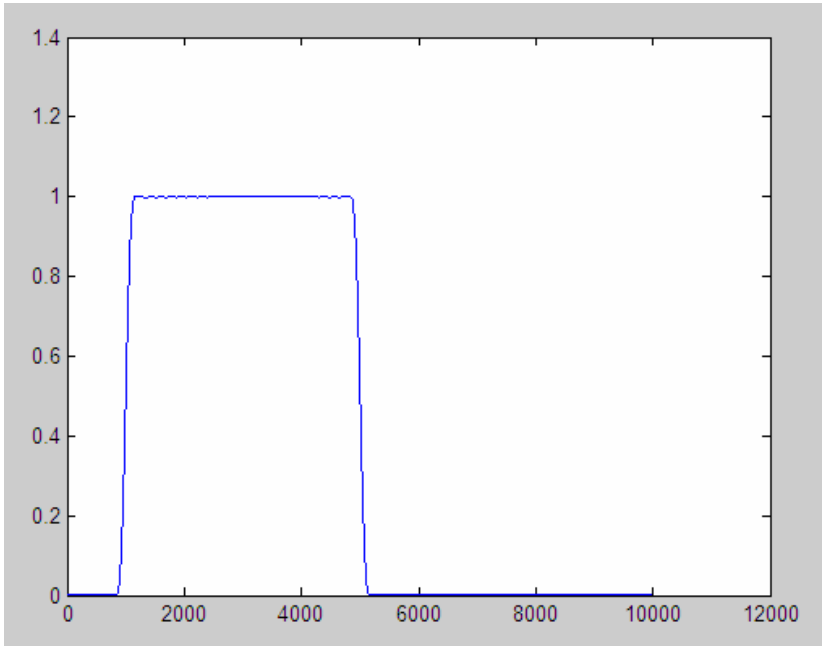


Fig. 13.6. Mòdul resposta freqüencial

Ara que s'ha comprovat que el filtre funciona com s'havia previst, es pot posar una entrada d'algun arxiu de so .wav i escoltar el nou so filtrat. Tot això es pot fer amb el Simulink:

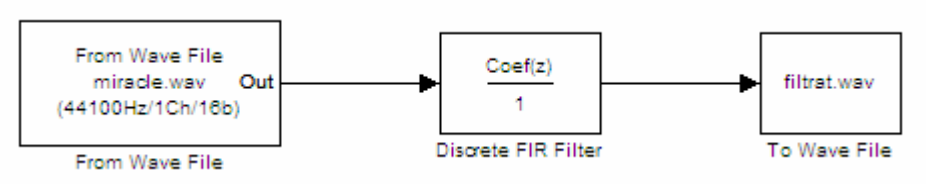


Fig. 13.7. Simulació i filtratge

En aquest cas s'ha agafat una cançó titulada 'miracle' i se l'ha passat pel filtre dissenyat. Cal dir que els coeficients que estaven guardats a la variable Coef (o num) del Workspace, s'han carregat al Simulink com es pot observar en la figura, és a dir, només posant-hi el nom de la variable 'Coef' al numerador del filtre FIR.

Nota: s'ha posat d'exemple el disseny d'un filtre FIR passabanda, però es pot fer un passa-baixos, un passa-alts o un elimina-banda d'aquesta manera:

```
>> baix=fir1(N,Wn,'low');      // Wn=freqüència de tall normalitzada respecte la meitat
                               freq. de mostreig. Com a exemple, s'hi posa 200/22050.
```

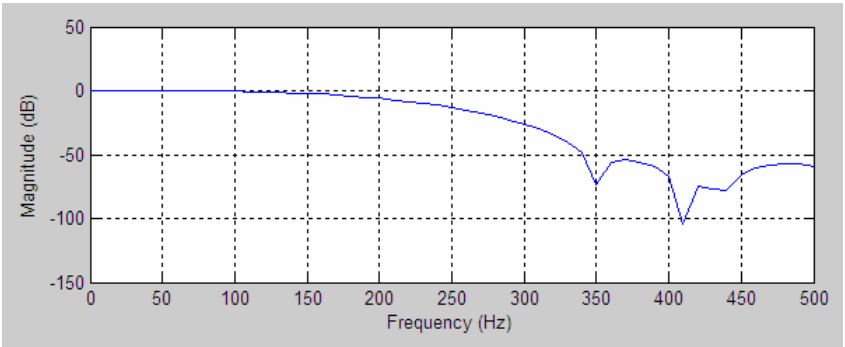


Fig. 13.8. Filtre passa-baixos

```
>> alts=fir1(N,Wn,'high');     // Com a exemple Wn=2000/22050.
```

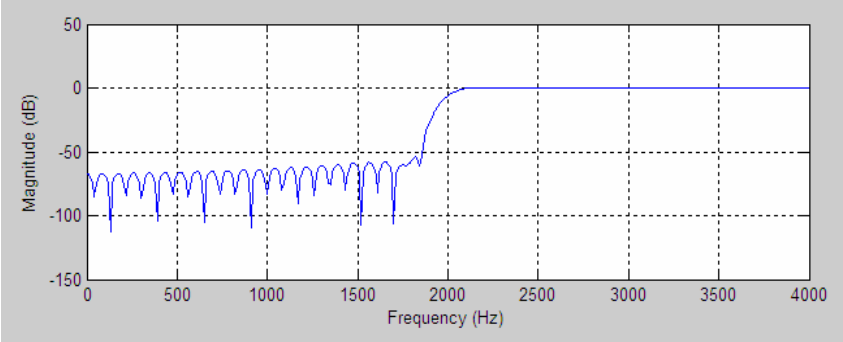


Fig. 13.9. Filtre passa-alts

```
>> ebanda=fir1(N,[1000 5000]/Fny, 'stop');
```

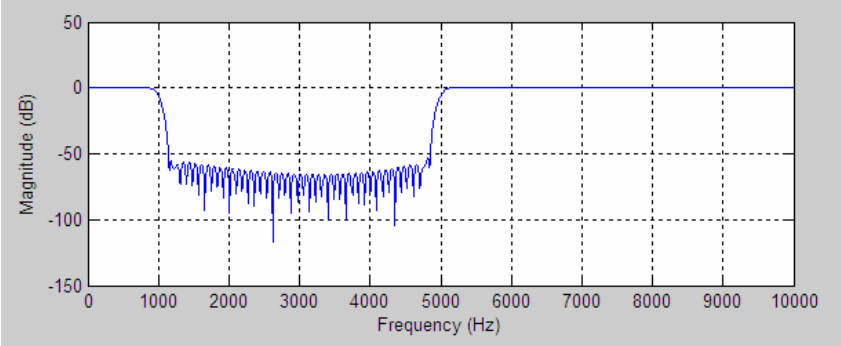


Fig. 13.10. Filtre elimina-banda

14- Disseny d'un IIR per aproximació de Butterworth

El Matlab permet dissenyar filtres IIR passant diversos paràmetres i que retorni l'ordre del filtre, la freqüència de tall o els coeficients de numerador i denominador adients per complir les especificacions. Tot això ho calcula utilitzant diverses aproximacions, com la de Butterworth, Chebyshev, Cauer o Bessel. Totes tenen els seus pros i contres, i per segons quines aplicacions podria ser una més útil que una altra. Aquí s'estudiarà l'aproximació de Butterworth, potser la més simple de totes.

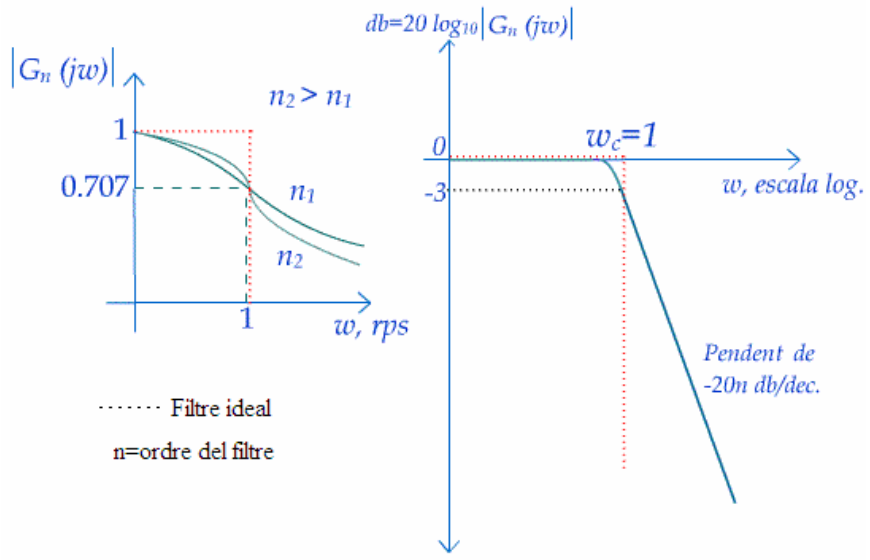
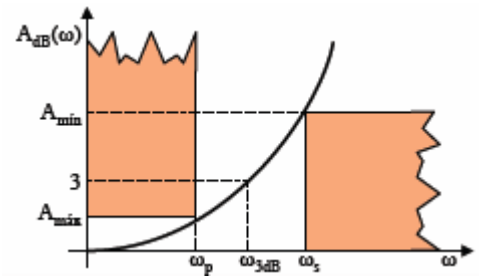


Fig. 14.1. Filtre passa-baixos ideal

Funció Guany (aproximació):

$$|G(j\omega)|^2 = \frac{1}{1+\omega^{2n}}$$

en decibels $\rightarrow A_{dB} = -20 \cdot \log|G(j\omega)| \rightarrow \boxed{A_{dB} = 10 \cdot \log(1+\omega^{2n})}$



ω_p = freq. Passabanda

ω_s = freq. Parabanda

ω_{3dB} = freq. a 3dB (freqüencia de tall)

$A_{mín}$ = atenuació passabanda

$A_{màx}$ = atenuació parabanda

Fig. 14.2. Atenuacions

En les atenuacions, es normalitzen les freqüències passabanda i parabanda respecte la freqüència de tall:

$$A_{màx} = 10 \cdot \log[1 + (\omega_p / \omega_{3dB})^{2n}] \rightarrow \omega_{3dB} = \frac{\omega_p}{\left(10^{A_{màx}/10} - 1\right)^{1/2n}}$$

$$A_{mín} = 10 \cdot \log[1 + (\omega_s / \omega_{3dB})^{2n}] \rightarrow \omega_{3dB} = \frac{\omega_s}{\left(10^{A_{mín}/10} - 1\right)^{1/2n}}$$

Si s'igualen les expressions i s'aïlla la n:

$$n = \frac{\log\left(\frac{10^{A_{mín}/10} - 1}{10^{A_{màx}/10} - 1}\right)}{2 \cdot \log\left(\frac{\omega_s}{\omega_p}\right)}$$

Amb aquesta fórmula es calcula l'ordre del filtre amb l'aproximació de Butterworth.

Exemple: volem saber l'ordre necessari per a un filtre passa-baixos amb les següents especificacions:

ω_p = 100;

ω_s = 1000;

$A_{màx}$ = 3; (ω_p = ω_{3dB})

$A_{mín} = 40$;

$$n = \frac{\log\left(\frac{10^{40/10}-1}{10^{3/10}-1}\right)}{2\cdot\log\left(\frac{1000}{100}\right)} = \frac{4}{2} \rightarrow \boxed{n=2}$$

L'ordre és el correcte perquè en una dècada hi ha un pendent de ± 40dB, per tant faran falta dos pols i dos zeros.

Amb el Matlab, es buscaria la n i ω_{3dB} d'aquesta manera:

```
>> Wp=100/22050;   (normalitzat respecte la meitat de la freqüència de Nyquist)
>> Ws=1000/22050;
>> Rp=3;
>> Rs=40;
>> [N,Wn]=buttord(Wp,Ws,Rp,Rs);
```

Retornaria N=2 i Wn=0,0045 (100/22050).

Ara que se sap l'ordre i la freqüència de tall del filtre, es busquen els seus coeficients:

La funció de transferència d'un filtre passa-baixos Butterworth de primer ordre, en la pla s és:

$$H(s) = \frac{\omega_c}{s+\omega_c} \quad \text{on } \omega_c = 2\pi f_c \text{ (} f_c \text{: freqüència de tall)}$$

Si es calcula la transformada bilineal per passar al pla z, quedaria:

$$H(z) = H(s)\Big|_{s=\frac{k(z-1)}{z+1}} \qquad \text{on } k = \frac{\omega_c}{\tan\left(\frac{\pi\cdot f_c}{F_s}\right)}$$

$$H(z)=\frac{\omega_c}{\frac{k(z-1)}{z+1}+\omega_c}=\frac{\omega_c\cdot z+\omega_c}{kz-k+\omega_c\cdot z+\omega_c}=\frac{\omega_c+\omega_c\cdot z^{-1}}{k-kz^{-1}+\omega_c+\omega_c\cdot z^{-1}}\rightarrow$$

$$\rightarrow H(z)=\frac{\frac{\omega_c}{k+\omega_c}+\frac{\omega_c}{k+\omega_c}\cdot z^{-1}}{1+\frac{\omega_c-k}{k+\omega_c}\cdot z^{-1}}$$

Ens quedarien els següents coeficients de numerador i denominador:

$$b_0=\frac{\omega_c}{k+\omega_c}\qquad b_1=\frac{\omega_c}{k+\omega_c}$$

$$a_1=\frac{\omega_c-k}{k+\omega_c}$$

a_0 (el terme més important del denominador sempre serà 1)

En el cas d'un filtre passa-baixos de segon ordre de Butterworth, els coeficients serien aquests:

$$b_0=\frac{\omega_c^2}{\omega_c^2+k^2+\sqrt{2}k\omega_c}\qquad b_1=\frac{2\omega_c^2}{\omega_c^2+k^2+\sqrt{2}k\omega_c}\qquad b_2=\frac{\omega_c^2}{\omega_c^2+k^2+\sqrt{2}k\omega_c}$$

$$a_1=\frac{2\omega_c^2-2k^2}{\omega_c^2+k^2+\sqrt{2}k\omega_c}\qquad a_2=\frac{\omega_c^2+k^2-\sqrt{2}k\omega_c}{\omega_c^2+k^2+\sqrt{2}k\omega_c}$$

En aquest exemple, s'aplicarien aquestes equacions amb el Matlab d'aquesta manera:

```
>> Fs=44100;
>> fc=100;
>> wc=2*pi*fc;
>> k=(wc)/(tan(pi*fc/Fs));
```

```
>> b0=(wc^2)/(wc^2+k^2+(sqrt(2)*k*wc))
   b0=5.024142299431053e-05
>> b1=(2*(wc^2))/(wc^2+k^2+(sqrt(2)*k*wc))
   b1=1.004828459886211e-04
>> b2=b0
   b2=5.024142299431053e-05
>> a1=((2*(wc^2))-(2*k^2))/(wc^2+k^2+(sqrt(2)*k*wc))
   a1= -1.979851542514359
>> a2=(wc^2+k^2-(sqrt(2)*k*wc))/(wc^2+k^2+(sqrt(2)*k*wc))
   a2 = 0.980052508206336
```

Per comprovar que els resultats són correctes, s'utilitza la funció 'butter' del Matlab, que calcula directament els coeficients del filtre:

```
>> N=2;
>> Wn=100/22050;   (normalitzat respecte la meitat de la freqüència de Nyquist)
>> [num,den]=butter(N,Wn,'low');
```

Els resultats serien num = [5.024142299431267e-05,1.004828459886253e-04,5.024142299431267e-05] i den = [1,-1.979851542514359,0.980052508206336], per tant, són correctes.

En aquest punt es pot mirar la situació dels zeros i dels pols en la circumferència de radi 1 del pla z:

```
>> z=roots(num);
   z = [-1   -1]
>> p=roots(den);
   p = [0.9899 + 0.001i   0.9899 – 0.001i]
>> zplane(z,p)
```

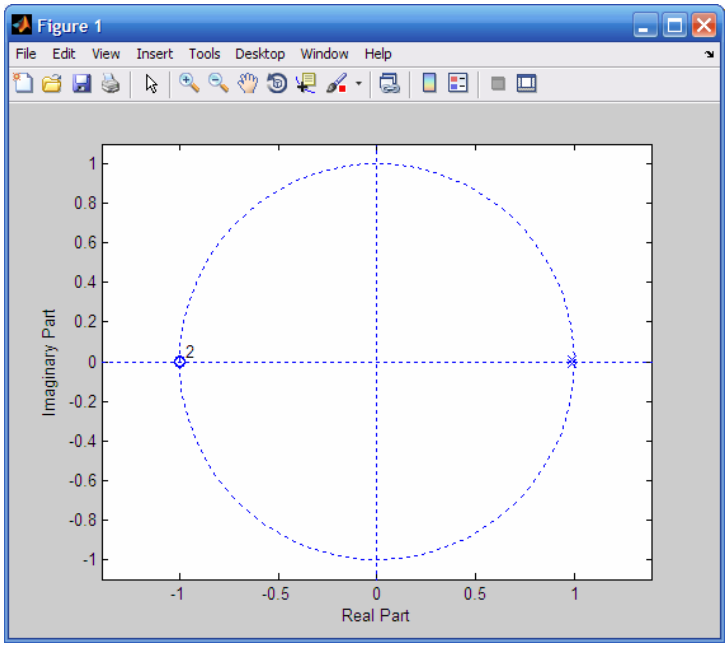


Fig. 14.3. Situació de pols i zeros

15- Disseny d'un efecte d'eco múltiple

Un efecte d'eco es produeix quan una senyal de so rebota i es torna a escoltar retardada i amb una intensitat més baixa. Amb el Simulink, es pot generar l'efecte d'eco desitjat simplement retardant el senyal original i atenuant-la de forma adequada.

Exemple:

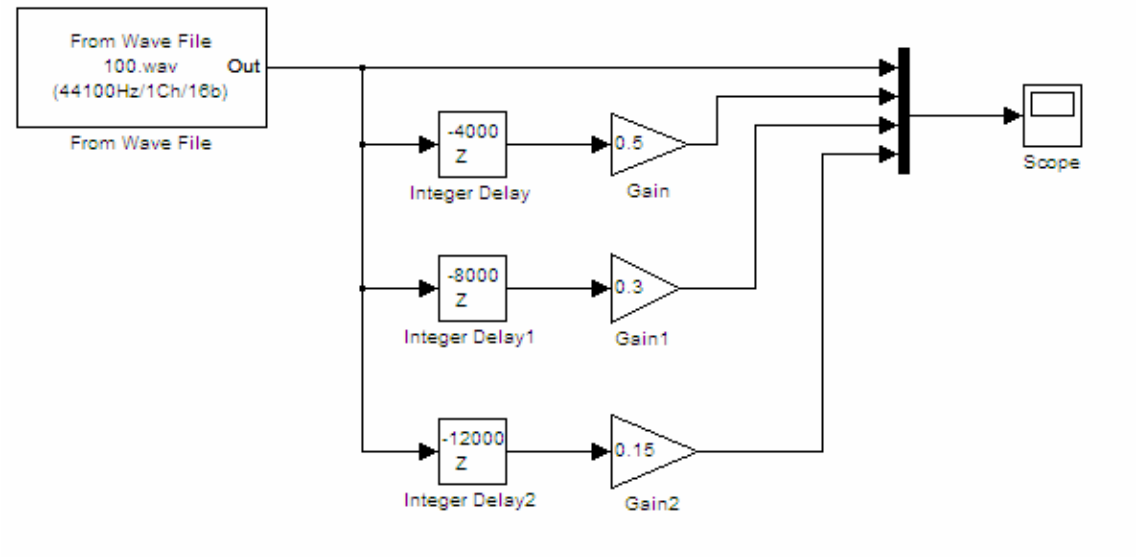


Fig. 15.1. Simulació d'un eco

Es disposa d'una senyal d'entrada sinusoïdal de 100 Hz invariant en el temps amb una $f_{mostreig}$ de 44.100 Hz. L'objectiu és simular un efecte d'eco múltiple.

La manera de fer-ho és aplicar retrassos de N mostres i atenuar el senyal baixant-li el guany.

Per saber el temps de retard es fa el següent:

$$f_m = 44100Hz \rightarrow T_m = \frac{1}{f_m} = \frac{1}{44100} = 2,2675737 \cdot 10^{-5} s$$

$$t_{retràs} = N_{mostres} \cdot T_m$$

per $N = 4000 \rightarrow t_{retràs} = 4000 \cdot 2,2675737 \cdot 10^{-5} = 90,7ms$

per $N = 8000 \rightarrow t_{retràs} = 8000 \cdot 2,2675737 \cdot 10^{-5} = 181,4ms$

per $N = 12000 \rightarrow t_{retràs} = 12000 \cdot 2,2675737 \cdot 10^{-5} = 272,1ms$

I per saber l'atenuació:

si $Guany = 0,5$ (reduïm l'amplitud del senyal a la meitat)

si $Guany = 0,3$ (reduïm l'amplitud de la senyal d'entrada un 70%)

si $Guany = 0,15$ (reduïm l'amplitud de la senyal d'entrada un 85%)

Podem fer una comprovació amb el Simulink del retard de N = 4000 mostres i Guany = 0,5 per una senyal d'entrada de 50 Hz:

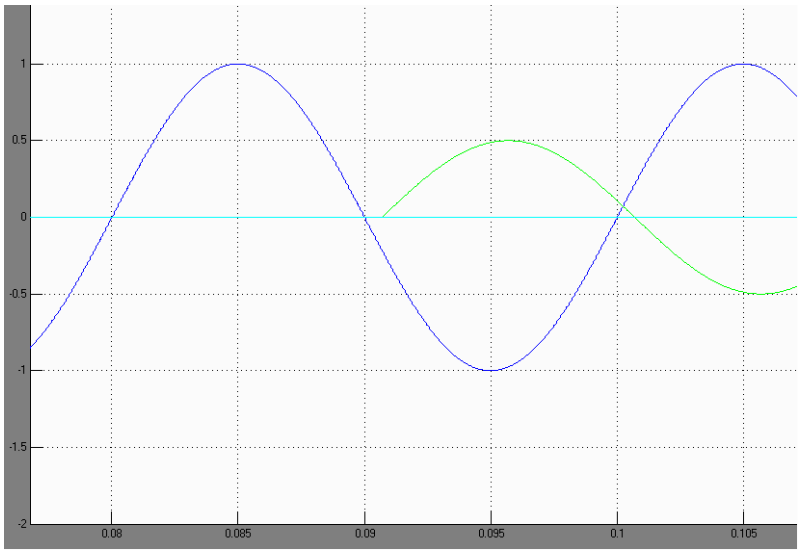


Fig. 15.2. Entrada (blau) i una de les sortides (verd)

16- Disseny d'un atenuador de pics màxims

Amb l'editor del Matlab, es dissenyarà un programa que atenuï els pics màxims d'una senyal de so qualsevol, amb l'objectiu d'evitar sons descompensats amb la resta del senyal.

Primer es carregarà la senyal de so .wav que estigui guardada en la carpeta del Matlab:

```
>> [y,Fs,bits]=wavread('senyalso');
```

Després es fa la FFT de 'y', que és la variable que conté tots els valors discrets de la senyal de so. Si es vol imprimir-ho per pantalla s'haurà de visualitzar la part absoluta (el mòdul) de la FFT:

```
>> ffty=fft(y);
>> plot(abs(ffty))
```

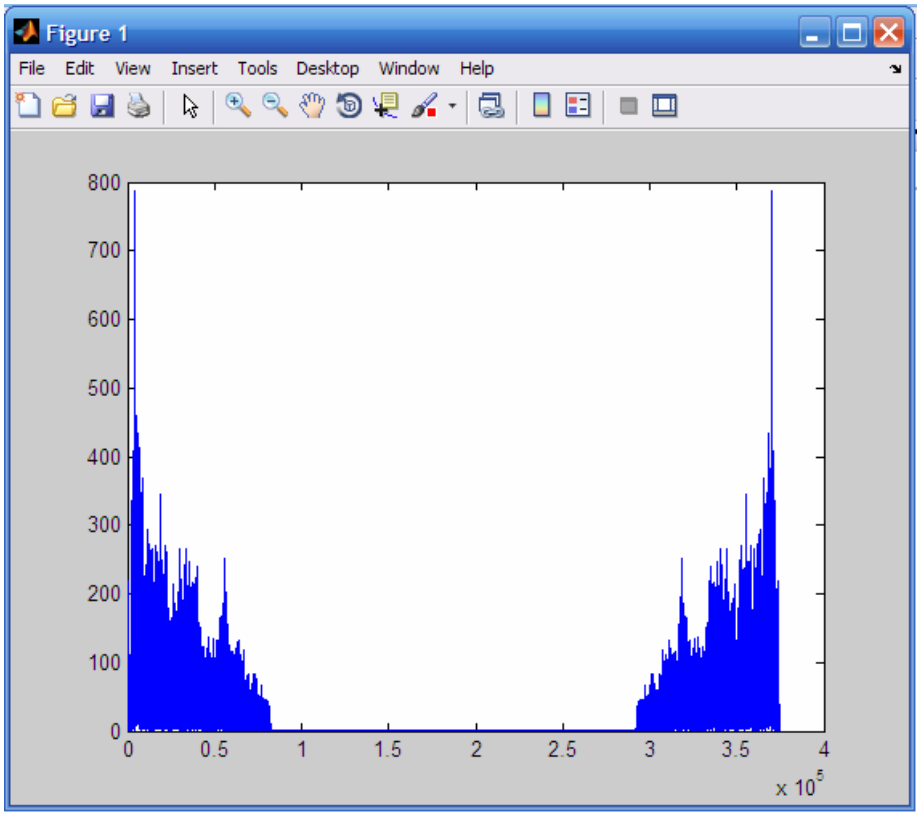


Fig. 16.1. FFT

Es pot comprovar com aquest senyal té pics molt alts que estan descompensats amb la resta del senyal. S'utilitzarà aquest exemple per il·lustrar el funcionament del programa dissenyat.

La forma del programa és aquesta:

```
function x=eliminavalors(array)
    x=array;
    s=sort(x,'descend');
    a=s(1:4);

    for i=(1:length(x))
        if(iguals(x(i),a))
            x(i)=((x(i-1)+x(i+1))/2);
        end
        array(i)=x(i);
    end
end

function trobat=iguals(real,ordenada)
    trobat=false;
    for k=(1:4)
        if(ordenada(k)==real)
            trobat=true;
        end
    end
end
```

Ara s'analitzarà pas per pas:

El programa comença recullint un array que se li passa pel Matlab. Aquest array serà la 'ffty'. L'array es guarda a 'x' i amb la funció 'sort', s'ordenen els seus valors de més alt a més baix.

```
function x=eliminavalors(array)
    x=array;
    s=sort(x,'descend');
```

Si s'executa el programa, es força un 'break point' i se situa el cursor a la variable 's', es veuran tots aquests valors ordenats:

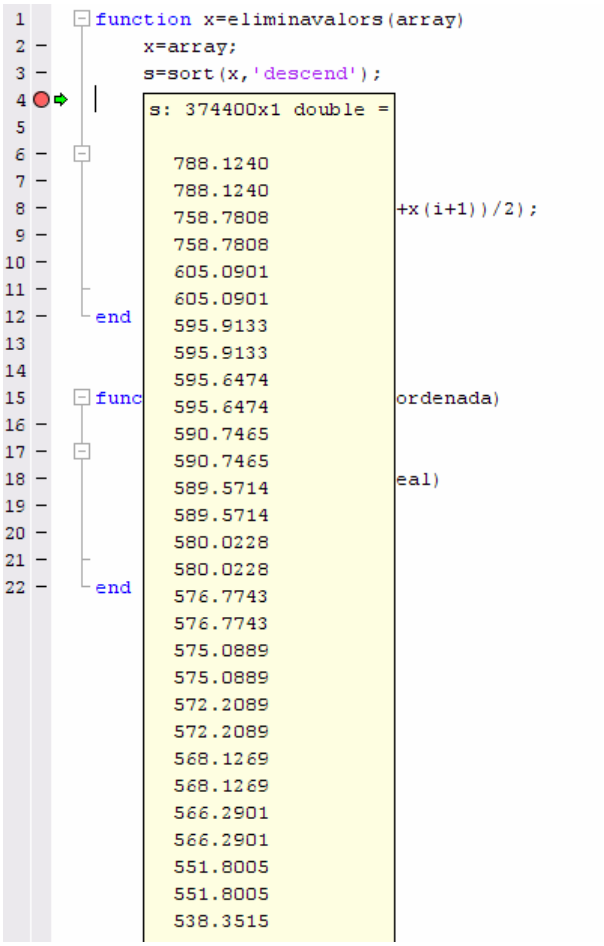


Fig. 16.2. Valors de l'array 'x' ordenats en 's'

Ara s'ha d'escollir quants d'aquests nombres s'atenuaran. En aquest exemple s'agafaran els quatre primers perquè la imatge anterior mostra que els quatre primers valors estan molt per sobre de la resta. Es posen aquests quatre valors a la variable 'a'.

```
    a=s(1:4);
```

A continuació, s'entra a un bucle 'for' que recorri tot l'array i dintre del for es crida a la funció 'iguals' que retornarà un booleà (true o false).

```
    for i=(1:length(x))
        if(iguals(x(i),a))
            x(i)=((x(i-1)+x(i+1))/2);
        end
        array(i)=x(i);
    end

function trobat=iguals(real,ordenada)
    trobat=false;
    for k=(1:4)
        if(ordenada(k)==real)
            trobat=true;
        end
    end
end
```

Aquesta funció fa el següent:

- S'entren com a paràmetres, els valors de l'array de tot el recorregut i els valors més alts.
- Es posa la variable 'trobat' a false inicialment.
- Es recorren els quatre valors que hi havia a 'a' (ara a 'ordenada').
- Si el valor de l'array en el punt 'i' del recorregut és igual a algun dels quatre valors de 'ordenada', llavors 'trobat' es converteix a true i retorna true a la crida de la funció.
- Si, en canvi, el valor de l'array no és igual a ningun del valors de 'ordenada', llavors 'trobat' continua a false i retorna false.

El booleà que retorni la funció 'iguals' farà que s'executi o no la línia de dins del 'if'.

```
if(iguals(x(i),a))
        x(i)=((x(i-1)+x(i+1))/2);
 end
array(i)=x(i);
```

En el cas que s'executi, voldria dir que s'ha trobat el punt on x(i) és igual a algun dels quatre valors de 'a'. Per tant, x(i) s'ha d'atenuar

.

El mètode escollit per atenuar és una mitja aritmètica entre el valor anterior i el següent de x(i), simplement per 'arrodonir' el resultat.

Quan ha acabat d'arrodonir, continua recorrent l'array i si no s'executa el contingut del if, es continua igualment recorrent l'array.

En aquest cas, s'executarà el if (com a true) 4 vegades, perquè hi ha quatre nombres a arrodonir i els punts d'arrodoniment (que es poden saber situant el cursor a sobre de la 'i' quan s'està executant) són aquests: 4265, 4267, 370135, 370137 (els dos últims són els simètrics).

El resultat de l'arrodoniment seria aquest:

Es crida al programa 'eliminavalors' i es guarda el resultat a la variable 'res'. Després es fa un 'plot' de la part absoluta:

```
>> res=eliminavalors(ffty);
>> plot(abs(res))
```

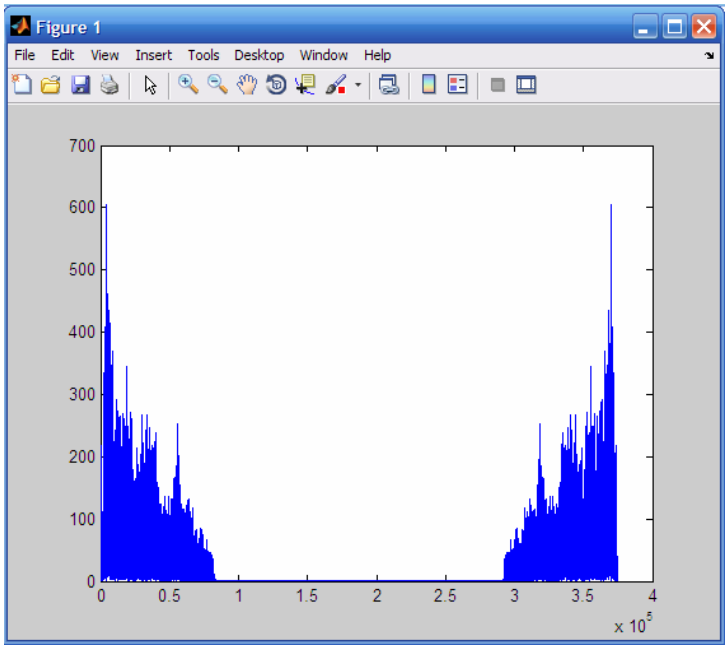


Fig. 16.3. Senyal filtrat

Si es fa un zoom als punts d'arrodoniment:

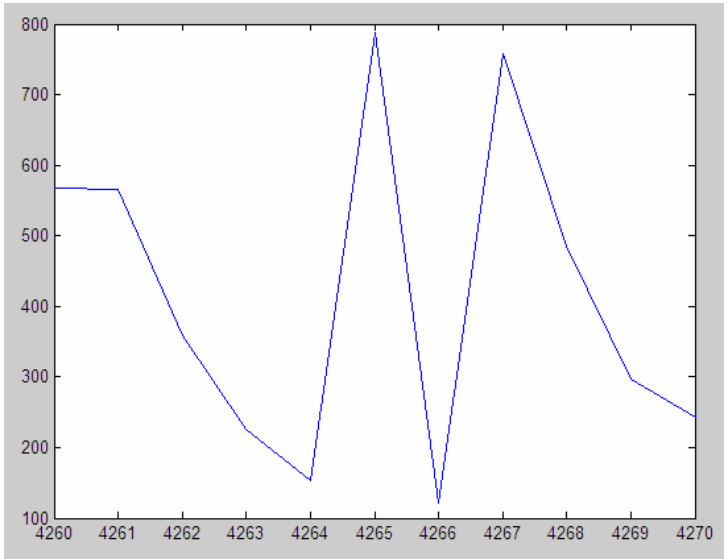


Fig. 16.4. Punts màxims abans d'arrodonir

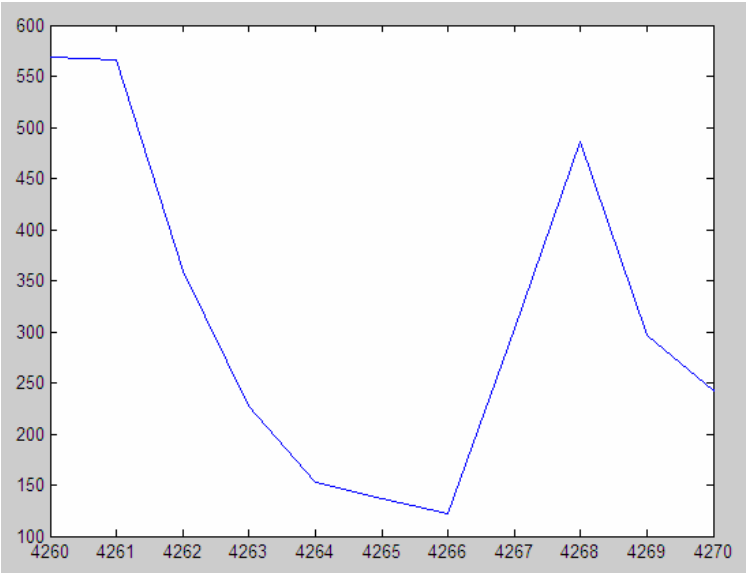


Fig. 16.5. Punts màxims arrodonits

Si es vol escoltar l'arxiu de so .wav que s'ha atenuat, s'ha d'escriure la següent línia:

```
>> wavwrite(ifft(res),Fs,'nouarxiu');
```

on la ifft és la inversa de la FFT (amb el nom de la variable on s'ha fet l'atenuació), la freqüència de mostreig que estava guardada en el Workspace quan es feia el 'wavread' i per últim i entre cometes, el nom que se li donarà al nou arxiu creat després de l'atenuació.

17- Estudi econòmic i pressupost

En aquest projecte en el qual no es realitza cap muntatge sinó que tot es simula a través d'eines informàtiques, l'estudi econòmic només té en compte els costos d'enginyeria i els costos de la instrumentació utilitzada.

17.1- Costos d'enginyeria

Concepte	Hores	Preu/hores (€)	Total (€)
Estudi i documentació	300	50	15.000
Redacció memòria	50	30	1.500

Total	16.500 €

17.2- Costos de instrumentació

Concepte	Hores d'utilització	Preu/hores (€)	Total (€)
Ordinador	300	0,5	150
Software Matlab	250	2	500
Software Audacity	20	0	0
Microsoft Office	50	1	50

Total	700 €

Pressupost total = CE + CI = 16.500 + 700 = 17.200 €

17- Conclusions

El treball desenvolupat durant tot aquest temps i la dedicació al projecte de l'estudi del tractament de senyals de so conclouen aquí. Arribats a aquest punt, es pot dir que s'ha complert amb l'objectiu previst al començament del projecte.

S'ha endinsat de ple en el món del tractament de senyals digitals, s'ha esbrinat com es comporta un sistema discret, tant temporal com freqüencialment, s'ha vist de què tracta una DFT i la seva importància dins de l'estudi de sistemes de tractament de senyals, els diferents filtres que existeixen i les seves diferències, s'ha assimilat conceptes com la convolució, l'estabilitat o la causalitat i s'ha après a dissenyar filtres, en el cas dels FIR's, utilitzant el mètode de les finestres de Hamming, i en el cas dels IIR, amb l'aproximació de Butterworth.

Després de tot el què s'ha treballat, es conclou amb l'acompliment dels objectius que s'havien proposat des de bon començament de forma satisfactòria.

18- Bibliografia

[1] Alan V. Oppenheim i Alan S. Willsky, *Señales y Sistemas.* S. Hamid Nawab, Segunda Edición, Prentice Hall.

[2] Alan V. Oppenheim i Ronald W. Schafer, *Tratamiento de Señales en Tiempo Discreto.* John R. Buck, Prentice Hall.

[3] Charles L. Phillips i H. Troy Nagle, Jr, *Sistemas de Control Digital.* Análisis y Diseño, Colección Ciencia Electrónica.

[4] Mark Kahrs i Karlheinz Brandenburg, *Applications of Digital Signal Processing to Audio and Acoustics*. Kluwer Academic Publishers.

[5] Ricard Villà, *Dinàmica de Sistemes*. CPDA.

[1] "Procesado digital de señal: base teórica. [Parte1]". *Miniwatt*. Vol. 25 No. 5:

[2] "Procesado digital de señal: base teórica. [Parte2]". *Miniwatt*. Vol. 25 No. 5:

[1] http://www.tsc.uvigo.es/BIO/Docencia/SCEE/Clases/Clase_07.pdf,
Funcions d'aproximació pel disseny de filtres.

[2] https://ccrma.stanford.edu/~jos/sasp/Window_Method.html,
Mètode de les finestres pel disseny de filtres.

[3] http://www2.dis.ulpgc.es/~obolivar/apuntes/tema5/tema5.htm,
Disseny de filtres FIR i IIR.

20- Annex

20.1- Tutorial de Sistemes Discrets amb el Matlab

20.1.1- Convolution

Commands covered: `conv`
`deconv`

To perform discrete time convolution, x[n]*h[n], define the vectors x and h with elements in the sequences x[n] and h[n]. Then use the command `y = conv(x,h)`

This command assumes that the first element in `x` and the first element in `h` correspond to n=0, so that the first element in the resulting output vector corresponds to n=0. If this is not the case, then the output vector will be computed correctly, but the index will have to be adjusted.

For example,

```
x = [1 1 1 1 1];
h = [0 1 2 3];
y = conv(x,h);
```

yields `y = [0 1 3 6 6 6 5 3]`. If `x` is indexed as described above, then y[0] = 0, y[1] = 1

In general, total up the index of the first element in `h` and the index of the first element in `x`, this is the index of the first element in `y`. For example, if the first element in `h` corresponds to n = -2 and the first element in `x` corresponds to n = -3, then the first element in `y` corresponds to n = -5.

Care must be taken when computing the convolution of infinite duration signals. If the vector `x` has length q and the vector `h` has length r, then you must truncate the vector `y` to have length min(q,r).

See the comments in Problem 3.7 of the textbook for additional information.

The command `conv` can also be used to multiply polynomials: suppose that the coefficients of a(s) are given in the vector `a` and the coefficients of b(s) are given in the vector `b`, then the coefficients of the polynomial a(s)b(s) can be found as the elements of the vector defined by `ab = conv(a,b)`.

The command `deconv` is the inverse procedure to the convolution. In this text, it is used as a means
of dividing polynomials. Given a(s) and b(s) with coefficients stored in `a` and `b`, then the coefficients of c(s) = b(s)/a(s) are found by using the command `c = deconv(b,a)`.

20.1.2- Transfer Function Representation

For a discrete-time transfer function, the coefficients are stored in descending powers of z or ascending powers of z-1. For example,

$$H(z) = \frac{2z^2+3z+4}{z^2+5z+6} = \frac{2+3z^{-1}+4z^{-2}}{1+5z^{-1}+6z^{-2}}3$$

then define the vectors as

```
num = [2 3 4];
den = [1 5 6];
```

20.1.3- Time Simulations

Commands Covered: `recur`
`conv`
`dstep`
`dimpulse`
`filter`

There are three methods to compute the response of a system described by the following recursive relationship.

$$y[n] + \sum_{i=1}^{N} a_i y[n-i] = \sum_{i=0}^{M} b_i x[n-i]$$

The first method uses the command `recur` and is useful when there are nonzero initial conditions. This command is available from the MathWorks ftp site and a shortened version is given in Figure C.5 of the textbook. The inputs to the function are the coefficients ai and bi stored in the vectors `a = [a1 a2 ... aN]` and `b = [b0 b1 ... bM]`, the initial conditions on x and on y are stored in the vectors `x0 = [x[n0-M], x[n0-M+1],...,x[n0-1]]` and `y0 = [y[n0-N], y[n0-N+1],...,y[n0-1]]`, and the time indices for which the solution needs to be calculated are stored in the vector `n` where n0 represents the first element in this vector. To use `recur`, type

```
y = recur(a,b,n,x,x0,y0);
```

The output is a vector `y` with elements y[n]; the first element of `y` corresponds to the time index n0.

For example, consider the system described by

y[n] - 0.6y[n-1] + 0.08y[n-2] = x[n-1]

where x[n] = u[n] and with initial conditions y[-1] = 2, y[-2] = 1, and x[-1] = x[-2] = 0. To compute the response y[n] for n = 0, 1,...,10, type

```
a = [-0.6 0.08]; b = [0 1];
x0 = 0; y0 = [1 2];
n = 0:10;
x = ones(1,11);
y = recur(a,b,n,x,x0,y0);
```

The vector `y` contains the values of y[n] for n = 0,1,...,10.

The second method to compute the response uses convolution and is useful when the initial conditions on y are zero. This method involves first finding the impulse response of the system, h[n], and then convolving h[n] with x[n] as discussed in Section 4.A. For example, consider the system described above with zero initial conditions, that is, y[-1]=y[-2]=0. The impulse response for this system is h[n] = 5[(0.4)n-(0.2)n]u[n]. The commands to compute y[n] are

```
n = 0:10;
x = ones(1,11);
h = 5*(0.4).^n - 5*(.02).^n;
y = conv(x,h);
y = y(1:length(n));
```

The vector `y` contains the values of y[n] for n = 0,1,...,10. Note that the vector was truncated to `length(n)` because both x[n] and h[n] are infinite duration signals. See the comments in Section 4.A regarding the convolution of infinite duration signals.

The third method of solving for the response requires that the transfer function of the system be known. The commands `dstep` and `dimpulse` compute the unit step response and the unit
impulse response, respectively while the command `filter` computes the response to initial conditions and to arbitrary inputs. The denominator coefficients are stored as `den = [1 a1 a2... aN]` and the numerator coefficients are stored as `num = [b0 b1 ... bM, 0 ... 0]` where there are N-M zeros padded on the end of the coefficients.

For example, consider the system given above with initial conditions y[-1] = y[-2] = 0. To compute the step response for n=0 to n=10, type the commands

```
n = 0:10;
num = [0 1 0]; den = [1 -0.6 0.08];
y = dstep(num,den,length(n));
```

The response can then be plotted using the `stem` plot. To compute the impulse response, simply replace `dstep` with `dimpulse` in the above commands.

To compute the response to an arbitrary input, store the input sequence in the vector `x`. The command `y = filter(num,den,x);` is used to compute the system response. If the system has nonzero initial conditions, the initial conditions can be stored in a vector v0. For a first order system where N=M=1, define `zi = [b1*x[-1]-a*y[-1]]`. For a second order system where N=M=2, define `zi = [b1*x[-1]+b2*x[-2]-a1*y[-1]-a2*y[-2], b1*x[-1]-a2*y[-1]]`. To compute the response with nonzero initial conditions, type

```
y = filter(num,den,x,zi);
```

For example, consider the previous system with the initial conditions y[-1] = 2 and y[-2] = 1 and input x[n] = u[n]. Type the following commands to compute y[n].

```
n = 0:10; x = ones(1,11);
num = [0 1 0]; den = [1 -0.6 0.08];
zi = [0.6*2-0.08*1, -0.08*2];
y = filter(num,den,x,zi);
```

20.1.4- Frequency Response Plots

Commands covered: `freqz`

The DTFT of a system can be calculated from the transfer function using `freqz`. Define the numerator and the denominator of the transfer function in `num` and `den`. The command

```
[H,Omega] = freqz(num,den,n,'whole');
```

computes the DTFT for n points equally spaced around the unit circle at the frequencies contained in the vector `Omega`. The magnitude of `H` is found from `abs(H)` and the phase of `H` is found from `angle(H)`. To customize the range for Ω, define a vector `Omega` of desired frequencies, for example `Omega = -pi:2*pi/300:pi` defines a vector of length 301 with values that range from -π to π. To get the DTFT at these frequencies, type

```
H = freqz(num,den,Omega);
```

20.1.5- Digital Filter Design

Commands covered: `bilinear`

`butter`
`cheby1`
`hamming`
`hanning`

The analog prototype method of designing IIR filters can be done by first designing an analog filter with the desired characteristics as shown in Section 3.D, then mapping the filter to the discrete-time domain. Store the numerator and denominator of the analog filter, H(s), in the vectors `num` and `den`, and let `T` be the sampling period. Then the numerator and denominator of the digital filter Hd(z) is found from the following command

```
[numd,dend] = bilinear(num,den,1/T)
```

Alternately, the commands `butter` and `cheby1` automatically design the analog filter and then use the bilinear transformation to map the filter to the discrete-time domain. Lowpass, highpass, bandstop, and bandpass filters can be designed using this method. The digital cutoff frequencies must be specified; these should be normalized by π. To design a digital lowpass filter based on the analog Butterworth filter, use the commands:

```
[num,den] = butter(n,Omegac)
```

where `n` is the number of poles and `Omegac` is the normalized digital cutoff frequency, $\Omega c = \omega cT/\pi$.

To design a highpass filter with cutoff frequency `Omegac`, use the commands

```
[num,den] = butter(n,Omegac,'high')
```

To design a bandpass filter with passband from `Omega1` to `Omega2`, define `Omega = [Omega1,Omega2]` and use the command

```
[num,den] = butter(n,Omega)
```

To design a bandstop filter with stopband from `Omega1` to `Omega2`, define `Omega = [Omega1, Omega2]` and use the command

```
[num,den] = butter(n,Omega,'stop')
```

The design for an nth order Type I Chebyshev filter is accomplished using the same methods as for `butter` except that "`butter`" is replaced by "`cheby1`":

```
[num,den] = cheby1(n,Omegac); % for a lowpass filter
[num,den] = cheby1(n,Omegac,'high'); % for a highpass filter
```

If `Omega` has two elements,

```
[num,den] = cheby1(n,Omega); % for a bandpass
[num,den] = cheby1(n,Omega,'stop'); % for a bandstop
```

The windows used in FIR filter design are given by

```
w = boxcar(N) % rectangular window
w = hamming(N)
w = hanning(N)
```

These commands are used to truncate the infinite impulse response of an ideal digital filter with the result being an FIR filter with length N.

The Signal Processing Toolbox also provides commands for computing the FIR filter directly. To obtain an FIR filter with length N and cutoff frequency Omegac (normalized by π) use the command

```
hd = fir1(N-1,Omegac)
```

The vector `hd` contains the impulse response of the FIR where `hd(1)` is the value of hd[0].

A length N highpass filter with normalized cutoff frequency Omegac is designed by using the command

```
hd = fir1(N-1,Omegac,'high')
```

A bandpass with passband from `Omega1` to `Omega2` is obtained by typing

`hd = fir1(N-1,Omega)`
where `Omega = [Omega1,Omega2]`.

A bandstop filter with stopband from `Omega1` to `Omega2` is obtained by typing
`hd = fir1(N-1,Omega,'stop')`
where `Omega = [Omega1,Omega2]`.

The `fir1` command uses the Hamming window by default. Other windows are obtained by adding an option of `'hanning'` or `'boxcar'` to the arguments; for example,
`hd = fir1(N-1,Omegac,'high',boxcar(N))`creates a highpass FIR filter with cutoff frequency `Omegac` using a rectangular window.

20.1.6- Digital Control Design
Commands covered: `bilinear`
`c2dm`
`hybrid`
An analog controller Gc(s) can be mapped to a digital controller Gd(z) using the bilinear transformation or the step response matching method. Store the numerator and denominator of Gc(s) in `num` and `den`. Then the numerator and denominator of Gd(z) is found from the bilinear transformation using the commands
`[numd,dend] = bilinear(num,den,1/T)`
where `T` is the sampling frequency.

To use the step invariant method, use the commands
`[numd,dend] = c2dm(num,den,T,'zoh')`

To simulate the response of a continuous-time plant with a digital controller, use the command `hybrid`, which is available at the MathWorks ftp site. Consider the block diagram in Figure 11.25.

The numerator and denominator coefficients of the plant are stored in `NGp` and `DGp`; the numerator and denominator coefficients of the controller are stored in `NGd` and `DGd`; the

reference input signal is stored in `r`; and the sampling time is stored in `T`. The increments in the time vector should selected to be the sampling time divided by an integer, for example, `t = 0:b:Tend` where there is some integer m such that bm=T. The command is used as

```
[y,ud] = hybrid(NGp,DGp,NGd,DGd,T,t,r);
```

The outputs of the command are the system response, `y`, and the control signal that is input to the plant, `ud`. The M-file contains a loop which computes the discrete-time control and then simulates the continuous-time plant for T seconds with the constant control. The process repeats for the next T second interval. The commands for `hybrid` are given below:

```
function [Y,UD] = hybrid(Np,Dp,Nd,Dd,T,t,U);
[Ac,Bc,Cc,Dc]=tf2ss(Np,Dp);
[Ad,Bd,Cd,Dd]=tf2ss(Nd,Dd);
nsam = T/(t(2)-t(1)); % # of integration pts per sample
% initialize
Y = 0;
UD = 0;
[ncr,ncc] = size(Ac);
xc0 = zeros(ncr,1);
[ndr,ndc] = size(Ad);
xdk = zeros(ndr,1);
kmax = fix(t(length(t))/T); % # of complete samples in t
for k = 0:kmax-1
% calculate control and output of zoh
ek = U(k*nsam+1) - Y(k*nsam+1);
xd = Ad*xdk + Bd*ek;
zoh = Cd*xdk + Dd*ek;
xdk = xd;
% integrate continuous-time plant with input
% of zoh for T seconds
udi = zoh*ones(nsam+1,1);
ti = t(k*nsam+1:(k+1)*nsam+1);
```

```
[yi,xi] = lsim(Ac,Bc,Cc,Dc,udi,ti,xc0);
xc0 = xi(nsam+1,:);
% augment vectors
Y = [Y;yi(2:nsam+1)];
UD = [UD;udi(2:nsam+1)];
end
if(kmax*nsam+1 < length(t))
% compute tail of simulation from t(kmax*nsam)
% to t_end
k = kmax;
% calculate control and output of zoh
ek = U(k*nsam+1) - Y(k*nsam+1);
xd = Ad*xdk + Bd*ek;
zoh = Cd*xdk + Dd*ek;
% integrate continuous-time plant with input of zoh
ti = t(k*nsam+1:length(t));
udi = zoh*ones(length(ti),1);
[yi,xi] = lsim(Ac,Bc,Cc,Dc,udi,ti,xc0);
% augment vectors
```

25

```
Y = [Y;yi(2:length(yi))];
UD = [UD;udi(2:length(udi))];
end
```