CURSUL 12

FILTRELE IIR ȘI TRECEREA DE LA ANALOGIC LA DIGITAL. EXEMPLE.
- Filtrul Butterworth trece-jos.
- Filtrul Chebyshev trece-jos.
- Filtrul eliptic trece-jos.

FILTRELE FIR: Designul optimal minimax.
- Răspunsul simetric/antisimetric la impuls și faza liniară.
- Algoritmul minimax pentru filtrul trece-jos.
- Exprimarea în decibeli a atenuării. Exemple comparative.

IMPLEMENTAREA FILTRELOR DIGITALE.
- Implementarea folosind limbaje de programare.
- Implementarea abstractă sub formă de schemă bloc.
- Secțiunea de ordin doi, sub forma directă I și forma directă II.
- Viteza de procesare și precizia procesării în timp real.
- MODELE DE ÎNTREBĂRI TIP GRILĂ PENTRU EXAMEN.

1

După ce în cursul precedent am identificat ce specificații tehnice trebuie furnizate, în general, pentru proiectarea unui filtru digital, și am dat argumente și contra-argumente pentru alegerea unui anumit tip de filtru (IIR sau FIR), vom parcurge cursul de azi analizând câteva tehnici de proiectare.

Filtrele IIR și trecerea de la analogic la digital

- Proiectarea filtrelor s-a făcut cu mult timp înainte ca procesarea digitală a semnalelor să apară.
- Există o mulțime de modele de filtre analogice, deja consacrate
- Există, însă, și metode de conversie a designului analogic într-o funcție de transfer rațională
- Multe pachete numerice oferă rutine special concepute pentru proiectarea filtrelor
- Proiectarea implică specificarea unor parametri și verificarea/testarea îndeplinirii cerințelor impuse

2

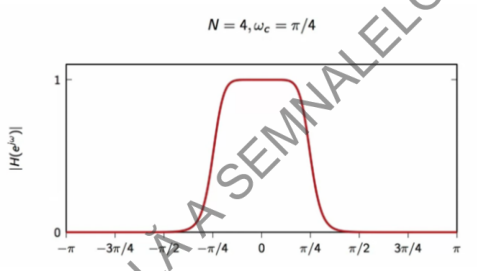
Ne concentrăm, pentru început, asupra filtrelor IIR.
Proiectarea filtrelor a căpătat consistență cu mult înainte ca procesarea digitală a semnalelor să apară și o mulțime de tipuri de filtre au fost dezvoltate, de-a lungul anilor, folosindu-se componente electronice discrete.
Există câteva metode consacrate care pot „traduce” designul analogic al unui filtru într-o funcție de transfer rațională.
Nu vom analiza detaliile ce țin de aceste metode, atâta timp cât majoritatea pachetelor numerice (precum Matlab, spre exemplu), vă oferă rutine „de-a gata”, care vă pot ajuta să proiectați filtre pe baza unui șablon clasic.
Proiectarea implică cel mai adesea o fază de încercare și de reglare a erorilor. Unii parametri pot fi specificați de la început, precum tipul de filtru și frecvența de tăiere dorită (sau frecvențele ce delimitează banda de tranziție). Apoi trebuie „ghiciți” alți parametri, cum ar fi ordinul filtrului. Se rulează rutina și se verifică dacă filtrul furnizat îndeplinește specificațiile. Dacă nu, probabil că va trebui să schimbăm și să mărim ordinul filtrului până când sunt îndeplinite specificațiile vizate.

Filtrul Butterworth trece-jos

Amplitudinea răspunsului în frecvență:
- Maxim plat
- Variație monotonă pe intervalul $[0, \pi]$

Parametri de proiectare
- Ordinul *N*
- Frecvența de tăiere

Valori de testat:
- Lățimea benzii de tranziție
- Eroarea benzii de trecere



3

Să analizăm acum câteva filtre clasice, ce s-au născut în lumea analogică.
Ne vom concentra pe prototipul de filtru trece-jos (lowpass). Rețineți, însă, că tehnicile de proiectare există și pentru celelalte tipuri de filtre.
Primul filtru la care ne vom opri este *filtrul Butterworth trece-jos*. Filtrul Butterworth are un răspuns în frecvență de amplitudine cu maxim plat și monotonă pe intervalul $[0, \pi]$.
Parametrii de proiectare sunt foarte simpli: ordinul filtrului și frecvența de tăiere.
Se rulează algoritmul și se obține un prototip. Testăm filtrul obținut în raport cu lățimea benzii de tranziție și eroarea benzii de trecere. Dacă vreunul dintre acești parametri nu îndeplinește specificațiile, se mărește ordinul și rulăm din nou algoritmul.

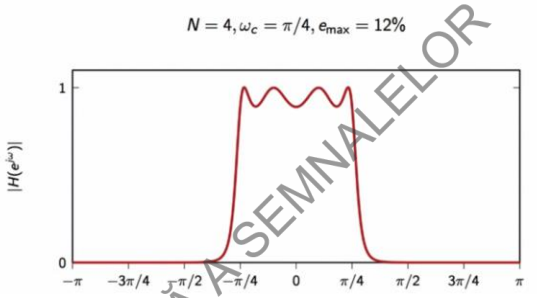
Răspunsul în frecvență al filtrului Butterworth arată ca în figură. Este, așa cum am spus, o curbă monotonă, care scade pe intervalul de la 0 la π. Acesta este un exemplu de filtru de ordinul 4, cu o frecvență de tăiere de $\pi/4$.

Filtrul Chebyshev trece-jos

Amplitudinea răspunsului în frecvență:
- Ripluri egale în banda de trecere
- Variație monotonă în banda de oprire

Parametri de proiectare
- Ordinul *N*
- Eroarea maximă a benzii de trecere
- Frecvența de tăiere

Valori de testat:
- Lățimea benzii de tranziție
- Eroarea benzii de oprire



4

Filtrul Chebyshev trece-jos are un răspuns în frecvență cu amplitudine având ripluri egale în banda de trecere (*equiripple*) și e monoton descrescătoare în banda de oprire.
Parametrii de proiectare sunt ordinul N al filtrului, eroarea maximă a benzii de trecere și frecvența de tăiere. Așadar, avem un parametru suplimentar în raport cu filtrul Butterworth.
Rulăm algoritmul și verificăm rezultatul în raport cu specificațiile filtrului, respectiv lățimea benzii de tranziție și eroarea în banda de oprire. Dacă vreunul dintre acești parametri nu îndeplinește specificațiile, creștem ordinul N și rulăm din nou algoritmul.

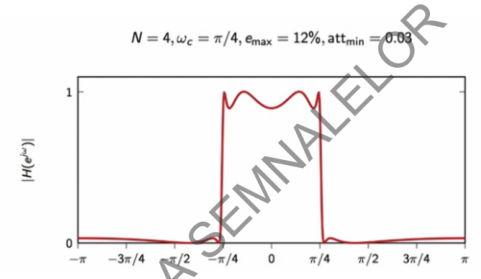
Răspunsul în frecvență al unui filtru Chebyshev arată ca în figură.
Avem, din nou, ordinul N=4, frecvența de tăiere $\pi/4$ și o eroare maximă în bandă de trecere de 12%.
În raport cu filtrul Butterworth, la acest filtru avem o bandă de tranziție mai abruptă, aceasta fiind recompensa pentru acceptarea unei erori cu ripluri egale în banda de trecere.

Filtrul eliptic trece-jos

Amplitudinea răspunsului în frecvență:
- Ripluri egale în benzile de trecere și de oprire

Parametri de proiectare
- Ordinul *N*
- Frecvența de tăiere
- Eroarea maximă a benzii de trecere
- Atenuarea minimă în banda de oprire

Valori de testat:
- Lățimea benzii de tranziție



5

Ultimul exemplu pe care îl analizăm este filtrul eliptic trece-jos. La un astfel de filtru riplurile sunt egale, atât în banda de trecere, cât și în banda de oprire. Parametrii de proiectare sunt: ordinul N, frecvența de tăiere, eroarea maximă a benzii de trecere și atenuarea minimă a benzii de oprire. Rulăm algoritmul și verificăm rezultatul în ceea ce privește lățimea benzii de tranziție.
Observați că acest tip de design ne permite să controlăm toți parametrii filtrului, cu excepția benzii de tranziție. Așadar, acest filtru este cel mai complex, dintre cele trei analizate.

Răspunsul în frecvență arată ca în figură, pentru specificațiile tehnice precizate.
Filtrul eliptic ne oferă cea mai abruptă bandă de tranziție pentru un ordin dat N, dintre toate filtrele analizate.

Trecerea de la domeniul analogic la domeniul digital

- Se aplică *transformarea filtrului prototip IIR* considerat ca filtru de plecare, în tipul de filtru dorit.

  În MATLAB, spre exemplu, există funcții care transformă un filtru trece-jos analogic în unul trece-jos (cu o altă frecvență de tăiere), trece-sus, trece-bandă sau oprește-bandă, cu frecvențele de tăiere dorite.

- Se realizează *discretizarea filtrului analogic*, prin determinarea funcției de transfer corespunzătoare filtrului digital (în domeniul timp se trece de la variabila continuă *t* la variabila discretă *n*).

  Există metode specifice, consacrate, care se utilizează în acest sens (metoda invarianței răspunsului la impuls, metoda transformării biliniare, metoda transformării în *z* adaptate, etc.).

6

Trecerea de la domeniul analogic la cel digital ține, în primul rând, de experiența celui care face proiectarea filtrului.
Două mențiuni trebuie făcute aici. Una ce ține de modul cum putem face *transformarea*, în domeniul analogic, a filtrului prototip în tipul de filtru dorit de noi, și respectiv cea cu privire la operațiunea de *discretizare* a filtrului obținut.

Observație importantă

Filtrele digitale IIR se pot proiecta și direct, în domeniul discret, fără a ne mai raporta la domeniul analogic.

- *Direct, în timp discret*: metoda de aproximare Padé, metoda celor mai mici pătrate
- *Direct, în domeniul frecvență*: metoda Yulke-Walker

7

Suntem datori să facem următoarea observație, enunțată în slide.

Nu insistăm asupra metodelor enumerate, ele putând fi studiate la laborator.

Filtrele FIR: design optimal minimax

Filtrele FIR sunt în exclusivitate digitale.
În anii 70, Parks și McClellan au dezvoltat un algoritm de proiectare optimală a filtrelor FIR.

- Fază liniară
- Eroare cu ripluri egale în banda de trecere și banda de oprire

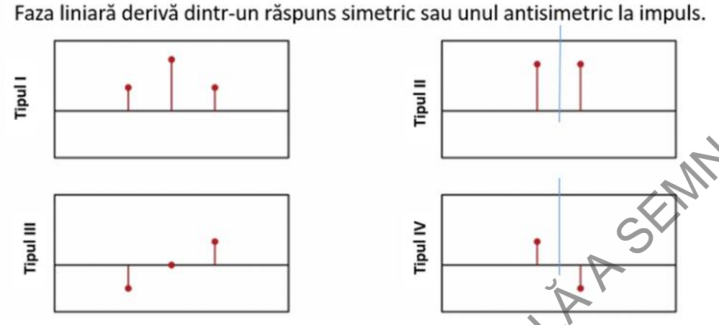
Algoritmul acționează minimizând eroarea maximă în banda de trecere și banda de oprire.

8

Să trecem în revistă acum metodele de proiectare ale filtrelor FIR. Proiectarea acestor tipuri de filtre, așa cum am mai spus, se bazează, în exclusivitate, pe procesarea semnalelor digitale. În cazul filtrelor FIR, funcția de transfer rațională nu mai este un raport de polinoame, ci doar un simplu polinom în z.
Există proceduri prin care se pot obține rezultate optime pentru un set dat de specificații tehnice.
În anii '70, Parks și McClellan au dezvoltat un algoritm de proiectare optimală a filtrelor FIR, prin care se obțin filtre cu fază liniară și
cu eroare egală, atât în banda de trecere, cât și în banda de oprire.
Algoritmul acționează prin *minimizarea erorii maxime* în banda de trecere și banda de oprire a funcției de transfer.

FAZĂ LINIARĂ

Faza liniară derivă dintr-un răspuns simetric sau unul antisimetric la impuls.



9

Liniaritatea fazei se realizează prin proiectarea unui răspuns la impuls, care este fie simetric, fie antisimetric. Ajungem la patru tipuri de filtre, în funcție de cum răspunsul la impuls este de lungime pară sau lungime impară, fiind simetric sau antisimetric.

Filtrele de tip I, probabil cele mai comune, au un răspuns la impuls de lungime N impară și sunt simetrice în jurul eșantionului central.

Filtrele de tip III au un număr N impar de eșantioane, sunt antisimetrice în raport cu cel central, care, desigur, este obligatoriu zero (altfel nu ar exista antisimetrie).

Filtrele de tip II și de tip IV sunt filtre simetrice, respectiv antisimetrice, dar ambele au un număr N par de eșantioane. Asta înseamnă că centrul de simetrie se află la egală distanță între eșantioanele centrale. Astfel, contribuția fiecărula este un factor de fază liniar non-întreg, dar egal cu jumătate.

Filtrul FIR de tipul I posedă fază liniară

$$h[C+n] = h[C-n]$$

$$h'[n] = h[n+C]$$
$$h'[n] = h'[-n]$$

$$H(z) = z^{-C}H'(z)$$

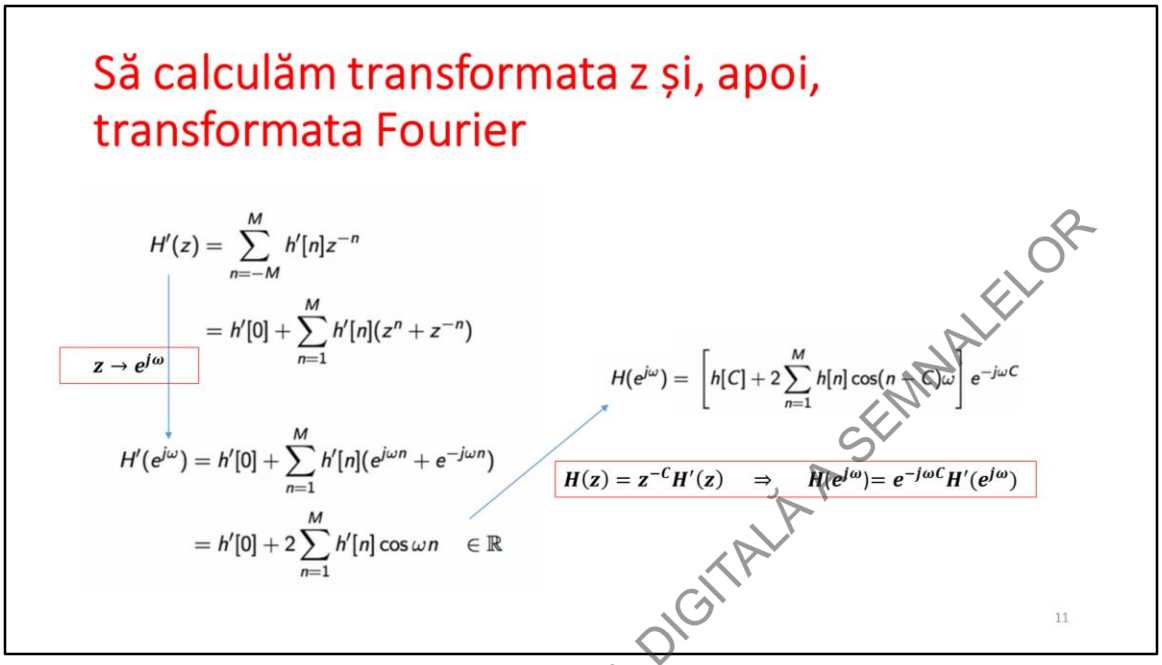
10

Să ne uităm mai detaliat la proprietățile fazei unui filtru FIR de tipul I.
Un filtru de tip I este un FIR simetric, de lungime impară.
Acolo unde eșantioanele sunt simetrice în jurul unui index C, relația de simetrie poate fi simplificată dacă deplasăm filtrul astfel încât eșantionul central să cadă în zero.
Definim, pentru asta, un filtru auxiliar, $h'[n] = h[n+C]$, astfel încât acest filtru va fi centrat acum în 0 și va fi simetric în raport cu 0, respectiv vom avea:

$$h'[n] = h'[-n].$$

Relația dintre transformatele z este:

$$H(z) = z^{-C}H'(z).$$



Dacă vom calcula transformata z a lui $H'(z)$, vom obține prima sumă din slide. Datorită simetriei, putem izola eșantionul central, $h'[0]$, și putem scrie restul sumei ca sumă de la *1* la *M*.

Dacă înlocuim acum z cu $e^{j\omega}$, obținem transformata Fourier a lui $h'[n]$, care se scrie ca în slide, după folosirea formulei pentru funcția cosinus. Așadar o cantitate pur reală. Aceasta înseamnă că transformata Fourier a lui $h'[n]$ este de fază zero.

Putem, apoi, obține transformata Fourier a lui $h[n]$, folosind formula de legătură dedusă:

$$H(z) = z^{-C}H'(z)$$

Am demonstrat, deci, că filtrele de tip I posedă fază liniară.

O demonstrație similară a liniarității fazei poate fi efectuată pentru toate celelalte trei tipuri de filtre FIR.

Algoritmul minimax pentru filtrul trece-jos

Amplitudinea răspunsului în frecvență:
- Ripluri egale în benzile de trecere și de oprire

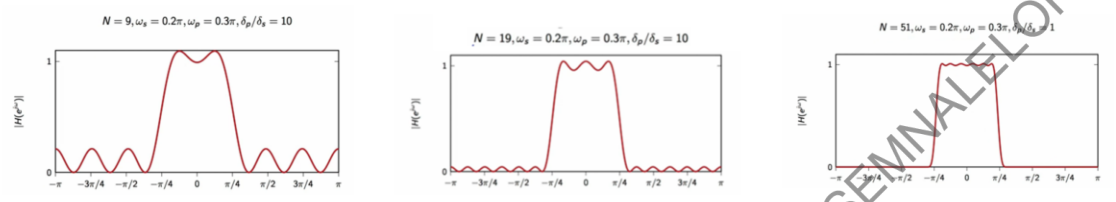
Parametri de proiectare
- Ordinul *N*
- Frecvența limită a benzii de trecere ω_p
- Frecvența limită a benzii de oprire ω_s
- Raportul de eroare δ_p/δ_s

Valori de testat:
- Eroarea maximă în banda de trecere
- Eroarea maximă în banda de oprire

12

După această demonstrație, analizăm algoritmul de proiectare Parks-McClellan, cunoscut și sub numele de *algoritmul de proiectare minimax*, în cazul filtrului trece-jos.
Reamintim că prin acest algoritm *se urmărește să se minimizeze eroarea maximă* permisă (în ceea ce privește amplitudinea răspunsului în frecvență al filtrului), atât în banda de trecere, cât și în banda de oprire.
Amplitudinea răspunsului în frecvență este caracterizată prin ripluri egale, atât în banda de trecere, cât și în banda de oprire.
Parametrii de proiectare sunt: ordinul N, marginea ω_p a benzii de trecere și marginea ω_s a benzii de oprire (adică lățimea benzii de tranziție), respectiv raportul dintre eroarea în banda de trecere și eroarea în banda de oprire: δ_p/δ_s.
Rulăm algoritmul și obținem un filtru cu care trebuie să verificăm eroarea maximă a benzii de trecere și eroarea maximă a benzii de oprire, specificate în cerințe.
Dacă oricare dintre ele depășește limitele, atunci trebuie să creștem ordinul N și să rulăm din nou algoritmul.

Exemplu: Filtru minimax trece-jos



13

Răspunsul în frecvență al filtrului minimax arată ca în figură. Este o caracteristică tipică, cu ripluri egale, și puteți vedea care au fost parametrii impuși, în acest caz, pentru N=9.

Priviți ce modificări se produc odată cu creșterea lui N, la valoarea 19, iar apoi la valoarea 51.

Putem vedea cum se obțin tranziții tot mai abrupte și erori tot mai mici în banda de trecere și banda de oprire.

Răspunsul în amplitudine exprimat în decibeli

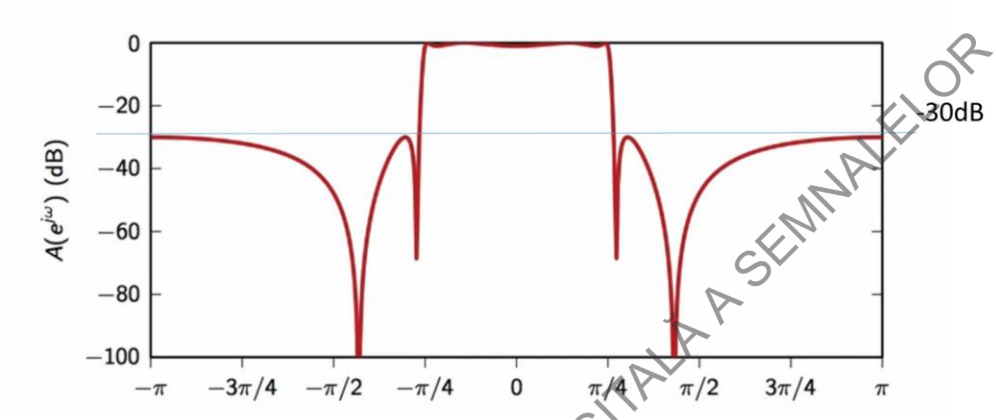
- Modalitate utilă în compararea atenuării dintre filtre
- Amplitudinea maximă în banda de trecere: G
- Atenuarea filtrului, exprimată în decibeli, este dată de:

$$A_{dB} = 20\log_{10}(|H(e^{j\omega})|/G)$$

14

O modalitate bună de a compara performanțele diferitelor filtre este de a exprima amplitudinea lor în decibeli.
Deci, dacă presupunem că G este amplitudinea maximă în banda de trecere, atunci atenuarea filtrului, exprimată în decibeli, este dată de formula din slide.

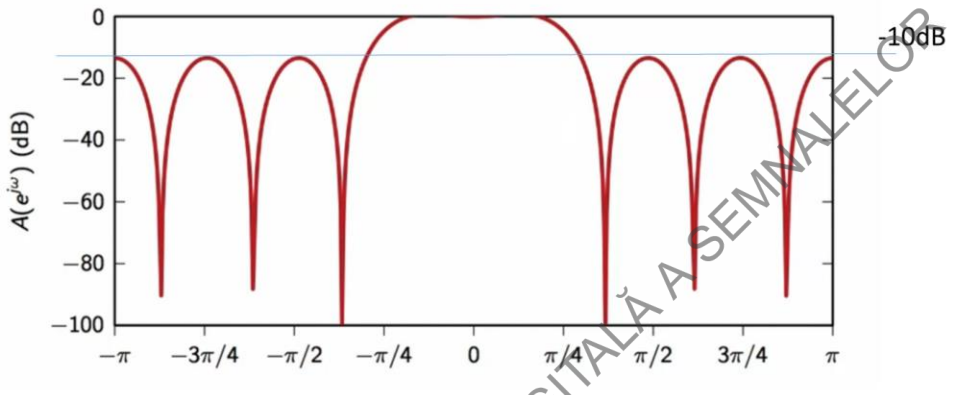
Filtru eliptic trece-jos de ordin 4, cu $\omega_c = \pi/4$, scară logaritmică



15

Cu noua notație introdusă, priviți cum arată reprezentarea grafică a atenuării pentru un filtru eliptic de ordin 4, cu frecvența de tăiere $\omega_c = \frac{\pi}{4}$. Putem observa aici o atenuare de aproximativ -30 dB în banda de oprire.

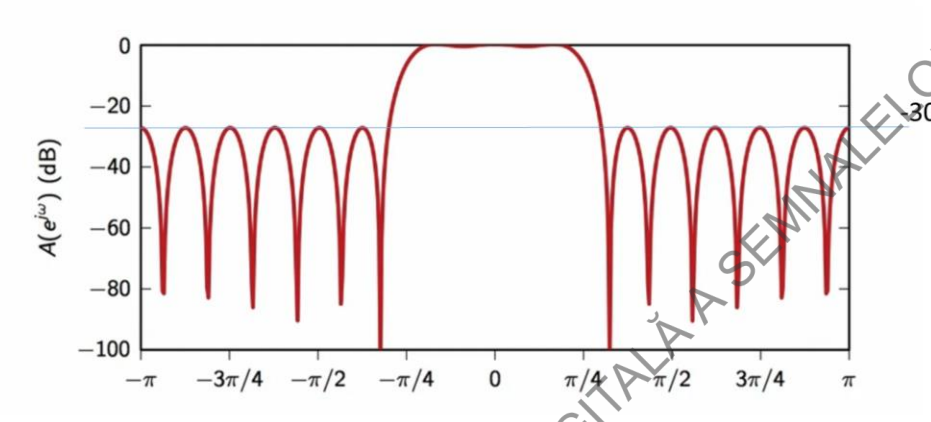
Filtru minimax trece-jos cu N=9, cu $\omega_c = \pi/4$, scară logaritmică



16

Un filtru trece-jos FIR minimax comparabil cu cel dat anterior (făcând comparația din punctul de vedere al costurilor de calcul), este cel pentru care s-a făcut reprezentarea din figură.
Pentru același cost (de calcul) obținem o atenuare care este cu siguranță mai slabă, pentru că este de ordinul a -10 dB (aproximativ).

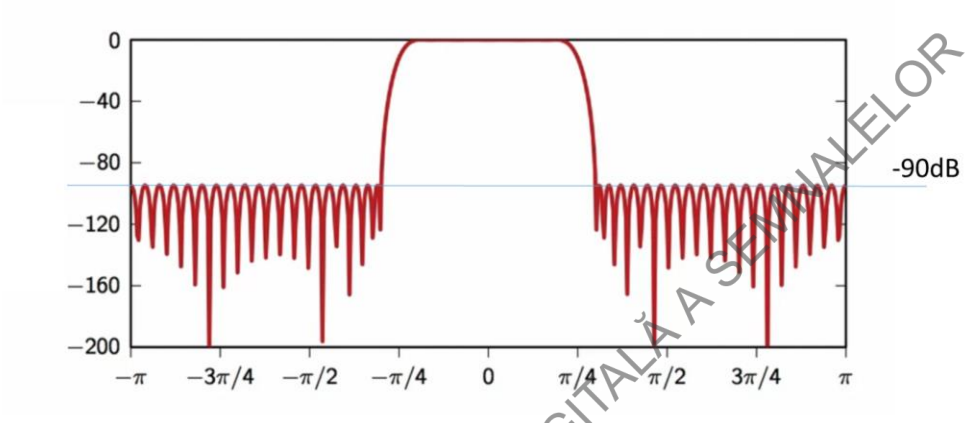
Filtru minimax trece-jos cu N=19, cu $\omega_c = \pi/4$, scară logaritmică



17

Pentru a obține o performanță comparabilă cu cea a filtrului eliptic, trebuie să mărim lungimea la cel puțin N=19. Aici am atins pragul de aproximativ -30 dB.

Filtru minimax trece-jos cu N=51, cu $\omega_c = \pi/4$, scară logaritmică



18

Filtrul pentru cazul N=51, pe care l-am văzut și mai înainte în reprezentarea obișnuită în domeniul frecvență, observăm că are o atenuare foarte, foarte bună. Ea este de aproximativ -90 dB. Dar, desigur, prețul pe care îl plătim acum nu este neglijabil, în ceea ce priveşte numărul de operaţii.

În cele din urmă, așa cum am menționat și anterior, în toate metodele descrise ne-am concentrat doar pe prototipuri de filtre trece-jos, dar putem folosi oricare dintre aceste modele pentru a obține și celelalte tipuri de filtre.

Prezentarea generală a problemei implementării filtrelor digitale

- Algoritmi utilizați pentru ecuațiile cu diferențe cu coeficienți constanți
- Schemă bloc
- Procesare în timp real

19

Am ajuns acum în punctul în care vom sublinia, pe scurt, unele probleme ce țin de implementarea filtrelor digitale.
Putem trece în limbaj cod, foarte ușor, orice ecuație cu diferențe cu coeficienți constanți, utilizând limbajul nostru de programare preferat, deoarece însăși ecuația reprezintă un algoritm în timp discret, care conține doar înmulțiri, adunări și întârzieri.
Mai interesant, putem reprezenta grafic această ecuație, așa cum am făcut-o deja, folosind scheme bloc. Acestea ne permit să gândim structura algoritmică într-un mod diferit și chiar realizează o optimizare a structurii filtrului, independentă de limbajul de programare folosit.
La implementarea filtrelor pentru aplicații în timp real, eficiența implementării este primordială. De aceea vom evidenția și anumite constrângeri, inerente procesării în timp real.
În ciuda abordărilor matematice, uneori abstracte, pe care le-am utilizat până acum, trebuie să fim conștienți că procesarea semnalelor digitale este o disciplină cât se poate de practică. Ceea ce am parcurs poate fi implementat, în bună parte, utilizând o platformă de calcul.

Un prieten vechi: *leaky integrator*

$$y[n] = (1 - \lambda)x[n] + \lambda y[n-1]$$

```
double Leaky(double x) {
  static const double lambda = 0.9;
  static double y = 0;

  y = lambda * y + (1 - lambda) * x;
  return y;
}
```

20

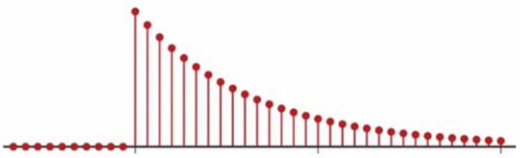
Revenim la un prieten mai vechi, *leaky integrator*, și îl prezentăm într-o formă ușor diferită.
Ne amintim ecuația cu diferențe cu coeficienți constanți, care descrie filtrul. Dăm o implementare a acesteia în cod C.

Modul în care implementăm filtrul presupune definirea unei funcții la care apelăm de fiecare dată atunci când sosește un eșantion de intrare, rezultând un eșantion de ieșire.
Funcția este de argument *double x*. În interiorul funcției avem o inițializare, iar apoi partea de calcul.
Vorbind despre inițializare, am definit valoarea pentru *lambda*, care este o valoare constantă.
Apoi definim *variabila locală y*, pe care o vom folosi pentru a păstra valoarea anterioară a ieșirii, pentru că trebuie să ne „amintim” această valoare atunci când apelăm funcția.
Este extrem de important să facem inițializarea. Prima dată când utilizăm funcția, vom presupune că filtrul era în repaus și că ieșirea a fost pe zero până la acel moment. Acesta este un punct foarte important atunci când ne ocupăm de implementarea practică.
Partea de calcul, ce urmează apoi, este doar o traducere simplă în C a ecuației cu diferențe cu coeficienți constanți.
Se va stoca, așadar, valoarea de ieșire în celula de memorie înainte de a se reveni.

Testarea

```
int main() {
  int n;
  for (n = 0; n < 20; n++)
    printf("%.4f ", Leaky(n==0 ? 1.0 : 0.0));
}
```

```
0.0500 0.0475 0.0451 0.0429 0.0407 0.0387 0.0368 0.0349 0.0332 0.0315
0.0299 0.0284 0.0270 0.0257 0.0244 0.0232 0.0220 0.0209 0.0199 0.0189
```



21

Putem acum testa dacă funcționează sau nu. Nu insistăm. Când rulăm, obținem ieșirea dată în slide, care, dacă facem reprezentarea grafică, este într-adevăr răspunsul la impuls al filtrului *leaky integrator*, sub formă de exponențială descrescătoare, așa cum ne așteptam.

Puncte cheie

- avem nevoie de o celulă de memorie pentru a stoca ieșirea anterioară
- trebuie să inițializam această celulă de memorie, înainte de a folosi filtrul pentru prima dată
- avem nevoie de două înmulțiri și o adunare pentru fiecare eșantion de ieșire

DE REȚINUT:
Costul unui filtru numeric este dependent de *numărul de operații* necesare obținerii fiecărui eșantion de ieșire, dar și de *capacitatea de memorare* de care avem nevoie în acest scop.

22

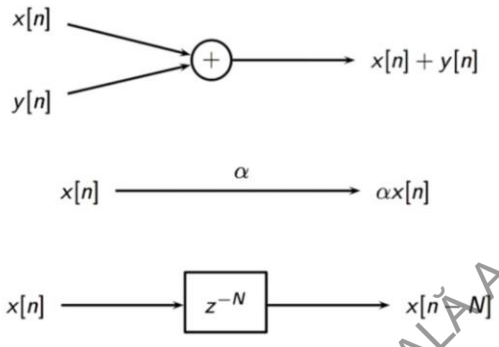
Punctele cheie pe care le reținem din implementarea *leaky integrator* în practică sunt următoarele:

- avem nevoie de o celulă de memorie pentru a stoca ieșirea anterioară
- apoi, trebuie să inițializam această celulă de memorie, înainte de a folosi filtrul pentru prima dată
- în ceea ce privește calculul, avem nevoie de două înmulțiri și o adunare pentru fiecare eșantion de ieșire.

Urmează implementarea în C. Dar, rețineți, limbajul utilizat este absolut irelevant atunci când vine vorba de implementarea unui filtru digital.

Ceea ce este important de subliniat este că, **în general**, costul unui filtru numeric este dependent de numărul de operații necesare obținerii fiecărui eșantion de ieșire, dar și de capacitatea de memorare de care avem nevoie.
Spre exemplu, în cazul filtrului cu *mediere mobilă,* pentru obținerea fiecărui eșantion de ieșire vom face M adunări (câte eșantioane anterioare trebuiesc luate în calcul) și o împărțire, dar vom avea nevoie și de M celule de memorie, pentru a stoca valorile eșantioanelor utilizate.

BLOCURI DE BAZĂ

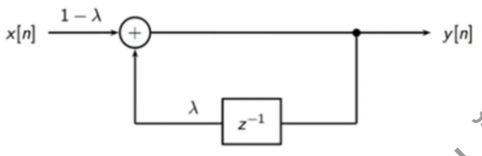


23

Am folosit, deja, de nenumărate ori blocurile de bază, abstracte, definite încă din primele cursuri și reprezentate în slide.
Cu aceste trei blocuri, am văzut că putem implementa orice ecuație cu diferențe cu coeficienți constanți.

Schema bloc a filtrului *leaky integrator*

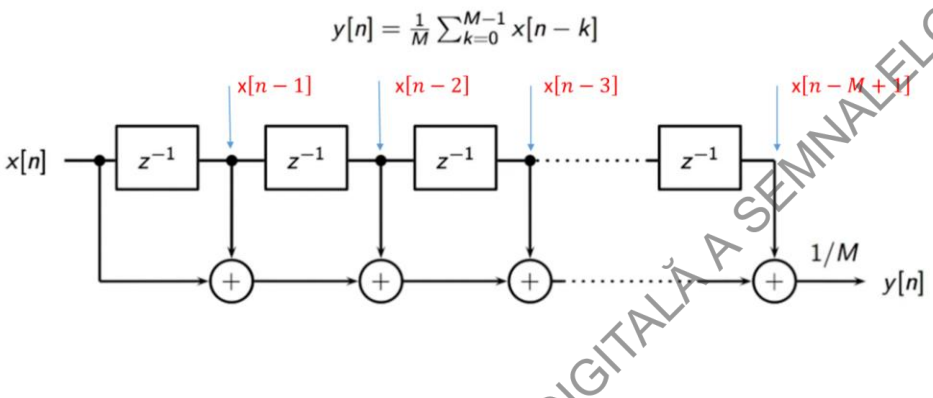
$$y[n] = \lambda y[n-1] + (1-\lambda)x[n]$$



24

Iată, de exemplu, filtrul *leaky integrator* dat prin schema sa bloc. Urmărim pe aceasta, din nou (a câta oară?), implementarea ecuației cu diferențe cu coeficienți constanți folosind blocurile de bază evidențiate.

Schema bloc a filtrului de *mediere mobilă*



Putem face același lucru cu filtrul de *mediere mobilă*.
Avem aici o serie de întârzieri și, prin fiecare bloc de întârziere, vom stoca o valoare anterioară a intrării, începând cu x[n-1] și până la x$[n - M + 1]$.
Se face însumarea lor: x[n]+x[n-1]+x[n-2]+...+ x[n-M+1],
după care împărțirea cu M.

Secțiunea de ordin doi

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}} = \frac{B(z)}{A(z)}$$

26

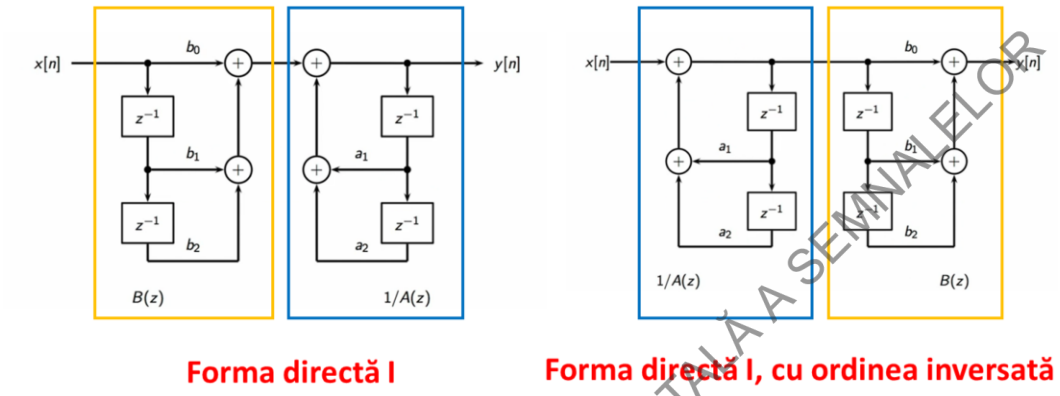
O altă structură interesantă pe care probabil o veți întâlni foarte des, dacă veți aprofunda domeniul procesării digitale a semnalelor, este ***secțiunea de ordinul doi.***

Aceasta corespunde la o funcție de transfer în care atât numărătorul, cât și numitorul, sunt polinoame de grad doi în z^{-1}. Polinoamele fiind cu coeficienți reali, atât rădăcinile numărătorului (zerourile), cât și rădăcinile numitorului (polii), dacă nu vor fi numere reale vor fi numere complex conjugate.

Vom avea, deci, doi poli și două zerouri.

Să remarcăm că orice funcție de transfer rațională poate fi scrisă ca produs de astfel de funcții, printr-o grupare convenabilă a factorilor la numărător și numitor. Produsul a N astfel de secțiuni se reduce, de fapt, la conectarea în cascadă a celor N structuri corespunzătoare.

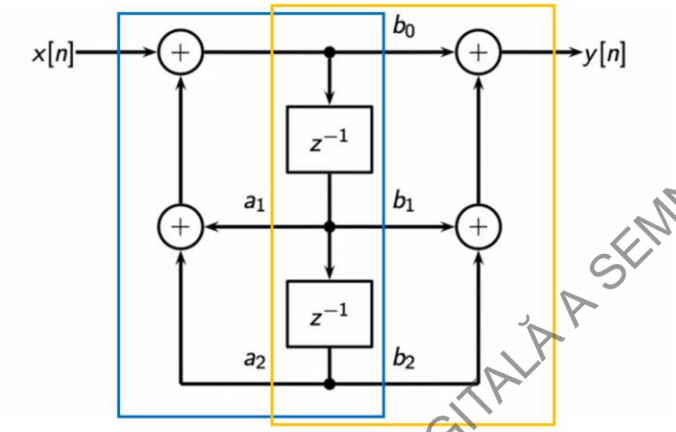
Schema bloc corespunzătoare secțiunii de ordin doi



Putem implementa secțiunea de ordinul al doilea, definită anterior, într-un mod foarte simplu, prin traducerea în limbajul schemei bloc a funcției de transfer.
Partea în chenar portocaliu, corespunde numărătorului funcției de transfer, iar cea în chenar albastru corespunde numitorului.

Aici apare partea interesantă. Folosind comutativitarea convoluției, putem inversa ordinea celor două subsecțiuni. Nimic nu se schimbă în ceea ce privește rezultatul. Dar acum, dacă analizăm structura, vedem că conținutul celulelor de întârziere va fi exact aceleași în orice moment (lucru care nu se întâmpla mai devreme).
Așadar, putem suprapune celulele de întârziere.

Secțiunea de ordin doi, forma directă II



28

Iată ce vom obține în acest mod: secțiunea de ordinul doi, în forma directă II. Avem, așadar, un *câștig net de două celule de memorie* (blocuri de întârziere) folosind acest tip de implementare. Puteți vedea că am reușit să optimizăm implementarea secțiunii de ordinul doi, fără a scrie cod în vreun limbaj de programare specific.

În sfârșit, câteva cuvinte despre procesarea semnalului în timp real. Să vedem cum este influențată viteza de procesare.

Dacă eșantioanele de pe intrare ajung la fiecare T secunde, fiecare eșantion de ieșire trebuie și el calculat în maxim T secunde, asta deși T poate fi de o valoare foarte, foarte mică. Spre exemplu, în audio, T este de obicei de 1/44.000 secunde.

Așadar, numărul de operaţii prin care eșantionul de ieșire este calculat devine un criteriu foarte important, iar alegerea filtrului va depinde de asta. Astfel devine important dacă vom alege să folosim un filtru IIR sau unul FIR.

Există și câteva trucuri comune ce pot fi aplicate în procesarea digitală a semanalelor. Unul dintre acestea este că se folosesc buffere circulare cu dimensiunea de forma 2^k, cu un *k* convenabil considerat, și, în aceste condiții, operația modulo se face mult mai rapid. De asemenea, se exploatează, atunci când este posibil, calculul în paralel cu mai multe procesoare, în loc să se forțeze creșterea spre limită a vitezei de calcul cu un singur procesor.

O altă problemă în cazul filtrelor reale este precizia procesării.
Unitățile de procesare vor avea întotdeauna o precizie aritmetică finită. Așa că *erorile de overflow* (când valoarea este atât de mare încât numărul de biți disponibil nu este suficient pentru reprezentarea valorii obținute) și cele de *underflow* (când valoarea este atât de mică încât numărul de biți disponibil nu este suficient pentru reprezentarea valorii obținute), pot reprezenta o reală problemă .
Putem utiliza filtre cu structuri mai complexe, pentru a combate unele dintre problemele de precizie. Asta, însă, în detrimentul eficacității.
Sunt folosite, de asemenea, și diverse trucuri pentru creșterea preciziei procesării, precum folosirea dublei precizii, divizarea structurii în secțiuni de ordin mai mic sau utilizarea virgulei mobile.

În loc de încheiere...

Literatura tehnică privind procesarea digitală a semnalelor este una cu adevărat vastă și, prin tot ce am parcurs noi până acum, putem spune că am răsfoit doar câteva pagini...

31

MODELE DE ÎNTREBĂRI TIP GRILĂ PENTRU EXAMEN

32

MODEL ÎNTREBARE 1

Dacă $x[n]$ este un semnal în timp discret, atunci valoarea lui $x[n]$ pentru un $n \notin \mathbb{Z}$ este:
a) zero
b) pozitivă
c) negativă
d) infinită
e) nu este definită

R: Răspunsul este e.

33

MODEL ÎNTREBARE 2

Se consideră semnalul discret de suport finit

$$x[n] = \{x[0] = 0, x[1] = 1, x[2] = 2, x[3] = 3 \text{ și } x[n] = 0, \text{în rest}\}.$$

Știind că acesta traversează un bloc de întârziere, care dintre următoarele secvențe ar putea fi culeasă pe ieșire?

a) 3, 2, 1, 0
b) 2, 3, 0, 0
c) 0, 0, 1, 2
d) 3, 0, 1, 2
e) 2, 3, 0, 1

R:

Ieșirea trebuie să fie identică cu intrarea (ordinea trebuie păstrată!), dar cu o întârziere oarecare. Pentru a), d) și e) condiția nu este îndeplinită. Pentru b), am avea semnal în avans $x[n+2]$. Singurul răspuns care corespunde, obținut pentru $x[n-2]$, este c.

34

MODEL ÎNTREBARE 3

Se consideră semnalul discret de suport finit dat de

$$x[n] = \{x[-1] = 2, x[0] = 4, x[1] = 0, x[2] = 3 \text{ și } x[n] = 0, \text{în rest}\}.$$

Care este expresia lui $x[n]$ scrisă cu ajutorul impulsului delta în timp discret $\delta[n]$?

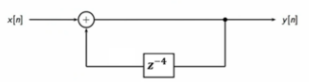
a) $x[n] = 2\,\delta[n] + 4\delta[n-1] + 3\delta[n-3]$
b) $x[n] = 2\,\delta[n+1] + 4\delta[n] + 3\delta[n-2]$
c) $x[n] = 2\,\delta[n] + 4\delta[n-1] + 3\delta[n-2]$
d) $x[n] = 2\,\delta[n-1] + 4\delta[n] + 3\delta[n+2]$
e) Niciunul dintre răspunsurile anterioare

R: Ținând cont de definiția lui $\delta[n]$, fără dificultate scriem:

$x[-1] = 2 = 2\,\delta[n+1]$, $x[0] = 4 = 4\delta[n], x[2] = 3 = 3\delta[n-2]$, în rest având $x[n] = 0$.
Răspunsul imediat este b.

35

MODEL ÎNTREBARE 4



Știind că semnalul de intrare este

$$x[n] = \{x[0] = 1, x[1] = 2, x[2] = 3, x[3] = 4 \text{ și } x[n] = 0, \text{în rest}\}$$

și considerând condiții inițiale nule pentru intrare, pentru ieșire și pentru conținutul blocului de întârziere, atunci:

a) $y[1001] = 1$
b) $y[1001] = 2$
c) $y[1001] = 3$
d) $y[1001] = 4$
e) $y[1001] = 0$

R:
Circuitul Karplus-Strong furnizează pe ieșire o secvență periodică în timp discret, de perioadă N=4, dată de: $y[n] = \begin{cases} 1, pentru\ n = 4k + 0 \\ 2, pentru\ n = 4k + 1 \\ 3, pentru\ n = 4k + 2 \\ 4, pentru\ n = 4k + 3 \end{cases}$ $cu\ k \epsilon \mathbb{Z}$

Cum: $1001 = 4 \cdot 250 + 1$, avem că $y[1001] = 2$ și răspunsul este b.

36

MODEL ÎNTREBARE 5

Care este răspunsul în frecvență al unui sistem descris de funcția de transfer

$$H(z) = \frac{1}{1 - 0{,}7z^{-1}}?$$

a) $\frac{e^{j\omega}}{e^{j\omega}-0{,}7}$

b) $\frac{e^{j\omega}}{e^{j\omega}+0{,}7}$

c) $\frac{e^{-j\omega}}{e^{-j\omega}-0{,}7}$

d) $\frac{e^{-j\omega}}{e^{-j\omega}+0{,}7}$

e) Niciunul dintre răspunsurile anterioare

R:

Pentru a obține răspunsul în frecvență al sistemului, adică transformata Fourier a răspunsului la impuls, este suficient să-l înlocuim în funcția de transfer *H(z)* pe z cu $e^{j\omega}$. Se obține răspunsul a.

MODEL ÎNTREBARE 6

Un SDLIT este cauzal dacă și numai dacă:

a) Răspunsul său la impuls, $h[n]$, este nenul pentru toate valorile pozitive ale lui *n*
b) Răspunsul său la impuls, $h[n]$, este zero pentru toate valorile pozitive ale lui *n*
c) Răspunsul său la impuls, $h[n]$, este nenul pentru toate valorile negative ale lui *n*
d) Răspunsul său la impuls, $h[n]$, este zero pentru toate valorile negative ale lui *n*
e) Niciunul dintre răspunsurile anterioare

R:

Definiția cauzalității oferă răspunsul corect imediat: d.

38

MODEL ÎNTREBARE 7

Să se calculeze rezultatul convoluției: $x[n] * \delta[n - n_0]$.
Răspunsul este:

a) $x[n + n_0]$
b) $x[n - n_0]$
c) $x[-n - n_0]$
d) $x[-n + n_0]$
e) Niciunul dintre răspunsurile anterioare

R: Se aplică definiția operației de convoluție și obținem

$$x[n] * \delta[n - n_0] = \sum_{-\infty}^{\infty} x[k]\delta[n - n_0 - k] = x[k]_{(n-n_0-k=0)} = x[n - n_0]$$

Răspunsul este b.

39

MODEL ÎNTREBARE 8

Care este transformata z a semnalului $\delta[n - n_0]$?

a) z^{n_0}
b) z^{-n_0}
c) $-z^{n_0}$
d) z^{n-n_0}
e) z^{n+n_0}

R:

$$X(z) = \sum_{n=-\infty}^{\infty} \delta[n - n_0] \cdot z^{-n} = z^{-n}|_{n-n_0=0} = z^{-n_0}$$

Răspunsul corect este b.

40

MODEL ÎNTREBARE 9

Un filtru numeric este descris de ecuația cu diferențe cu coeficienți constanți

$$y[n] = 0{,}5y[n-1] + 2x[n].$$

Funcția de transfer a filtrului este:

a) $\frac{2}{1+0{,}5z^{-1}}$

b) $\frac{0{,}5}{1+2z^{-1}}$

c) $\frac{0{,}5}{1-2z^{-1}}$

d) $\frac{2}{1-0{,}5z^{-1}}$

e) Niciunul dintre răspunsurile anterioare

R: Aplicând transformata z ecuației date se obține:

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

Răspunsul corect este d.

41

MODEL ÎNTREBARE 10

Un filtru numeric este caracterizat prin funcția de transfer

$$H(z) = \frac{1}{1 - 0{,}5z^{-1}} + \frac{2}{1 - 3z^{-1}}$$

Știind că filtrul este cauzal, regiunea de convergență RC este:

a) $|z| < 3$
b) $|z| > 3$
c) $|z| < 0{,}5$
d) $|z| > 0{,}5$
e) $|z| > 3{,}5$

R: Polii sunt $z_1 = 0{,}5$ și $z_2 = 3$. Cum filtrul este cauzal, avem că RC coincide cu regiunea din planul complex pentru care avem simultan $|z| > 0{,}5$ și $|z| > 3$. Intersecția celor două mulțimi este $|z| > 3$ și prin urmare răspunsul este b.

42