Filters met(IIR) en zonder tegenkoppeling (FIR)

- Berekening waarde uitgang enkel afhankelijk van de huidig waarde en een aantal vorige waarden van het inputsignaal.

- Berekening waarde uitgang gebeurt met zowel een aantal waarden afkomstig van de ingang als van de uitgang (terugkoppeling van uitgang naar ingang)

Filters met(IIR) en zonder tegenkoppeling (FIR)

- Er is geen terugkoppeling aanwezig van de uitgang naar de ingang.
- Hierdoor bestaat de gefilterde waarde van deze filter enkel met een eindig aantal waarden.
- De filter is hierdoor steeds stabiel.

- Hebben een terugkoppeling van de uitgang naar de ingang.
- Hierdoor bestaat de gefilterde waarde van deze filter theoretisch uit een oneindig aantal waarden
- Ten gevolge van deze terugkoppeling kunnen deze filters instabiel zijn.

Filters met(IIR) en zonder tegenkoppeling (FIR)

- Het werkingsprincipe komt overeen met het bepalen van het gemiddelde in de huidige en vorige waarden die voorzien zijn van gewichten met een factor b_k.

- Het werkingsprincipe komt overeen met het samenvoegen van een vorm van het bepalen van gemiddelde met zowel de huidige als met vorige in- en uitgangangswaarden.
- Zowel de in- als uitgangswaarden die hierbij betrokken zijn, zijn voorzien van gewichten.

Filters met(IIR) en zonder tegenkoppeling (FIR)

- FIR heeft beduidend meer coëfficiënten nodig (hoger orde) dan de IIR−filter om dezelfde eigenschappen te bekomen

- Hebben beduidend minder coëfficiënten nodig dan FIR om bepaalde filtereigenschappen, zoals demping en rimpel, te bekomen
- Door het feit dat ze weinig coëfficiënten nodig hebben vertonen deze filters maar een kleine signaalvertraging (wat een voordeel is bij online (real time)

Filters met(IIR) en zonder tegenkoppeling (FIR)

- Lineaire faseresponse

- De groepsvertraging is niet constant
- Hebben een niet-lineaire faseresponse

Filters met(IIR) en zonder tegenkoppeling (FIR)

- Magnitude en fase kunnen bepaald worden onafhankelijk van elkaar.

- IIR-filters zijn gelijkwaardig als de analoge filter

Transfertfunctie

- In analoge systemen beschreven via differentiaalvergelijkingen
- In digitale systemen beschreven aan de hand van verschilvergelijkingen

- ***Wat is een verschilvergelijking?***
  - ***Een verschilvergelijking is een rekenregel waarbij de huidige waarde van de output sequence y_n, en de huidige waarde van de input $x_n$en alle vorige waarden van de input- en output sequence worden gebruikt.***

  - Lineaire verschilvergelijkingen met constante coëfficiënten zijn de meest belangrijkste.
  - De coëfficiënten hebben de benaming $b_k$en a_k.

Verschilvergelijking

- ***FIR-verschilvergelijking***

  - ***Bevat enkel de ingangsignalen*** x_n , de uitgangssignalen y_n en de coëfficiënten b_k ***en*** a_k.

  - $y_n + a_1 y_{n-1} + a_2 y_{n-2} + \cdots + a_k y_{n-k} = b_0 x_n + b_1 x_{n-1} + b_2 x_{n-2} + \ldots + b_k x_{n-k}$

Transfertvergelijking

Voorbeeld FIR-transfertvergelijking

$$G_z = b_0 + b_1 z^{-1} + \ \ldots + b_n z^{-n}$$

De FIR-filter heeft geen noemer (geen terugkoppeling) in zijn transfertfunctie

Voorbeeld IIR-transfertvergelijking

$$G_z = \frac{b_0 + b_1 z^{-1} + \ \ldots + b_n z^{-n}}{1 + a_1 z^{-1} + \ \ldots + a_n z^{-n}}$$

Een IIR-filter heeft een gemeenschappelijke noemer in de transfertfunctie

Eigenschappen digitale filters

Afsnijfrequentie

- *Frequentie met nog de helft van het vermogen (-3 dB)*

Doorlaatband (pass band)

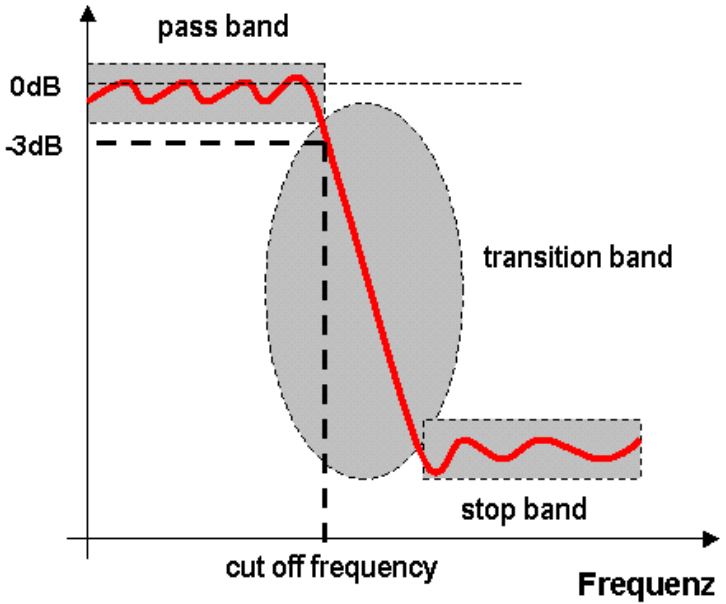
- *Zo vlak mogelijk om zo min mogelijk vervorming te bekomen*

Transistion band (overgangsband)

- *Moet zo klein mogelijk zijn*

Sperband

- *Zo hoog mogelijke dempingsverhouding (en eveneens vlak)*



Eigenschappen digitale filters

Filterorde

- *Hoe hoger de orde, hoe beter de filter*

Filtertype

- *Laagdoorlaat, hoogdoorlaat, banddoorlaat, bandsper*

Filter design procedure

- *Butterworth, Bessel, Chebycheff, …*

Sampling theorema en genormaliseerde frequenties

Gebruik van een bepaalde samplefrequentie f_s

- *kan frequenties evalueren tot aan* $\frac{f_s}{2}$ $(Nyquistfrequentie)$

In DSP f_s *dikwijls genormaliseerd naar frequentie* $1\ Hz$

- *Te evalueren frequenties tussen* $0\ Hz$ *en* $0,5\ Hz$

Sampling theorema en genormaliseerde frequenties

Voorbeeld:

- $f_s = 500\ Hz \Rightarrow fnyquist = 250\ Hz$
- $f_s = 10\ Hz \Rightarrow fnyquist = 5\ Hz$
- $f_s = 1\ Hz \Rightarrow fnyquist = 0,5\ Hz$

In digitale signaalprocessing wordt de samplefrequentie dikwijls genormaliseerd naar 1,0 Hz zodat de te samplen frequenties liggen tussen 0,0 fs en $0,5\ f_s$. (Dit betekent dat de afsnijfrequentie moet liggen tussen 0,0 fs en $0,5\ f_s$)

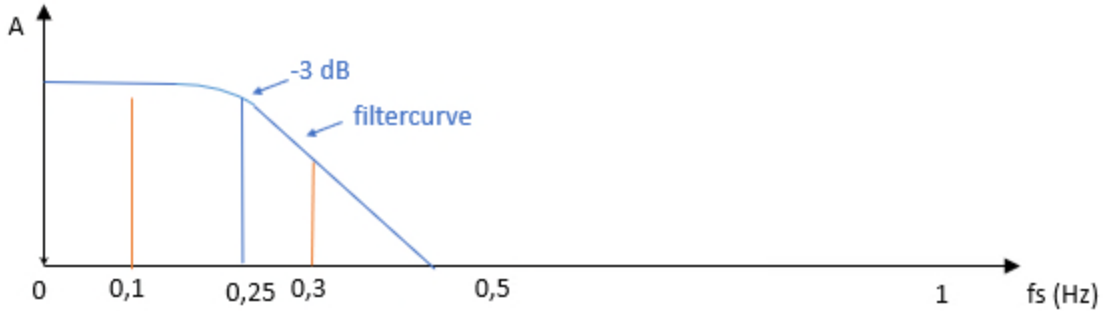
Sampling theorema en genormaliseerde frequenties

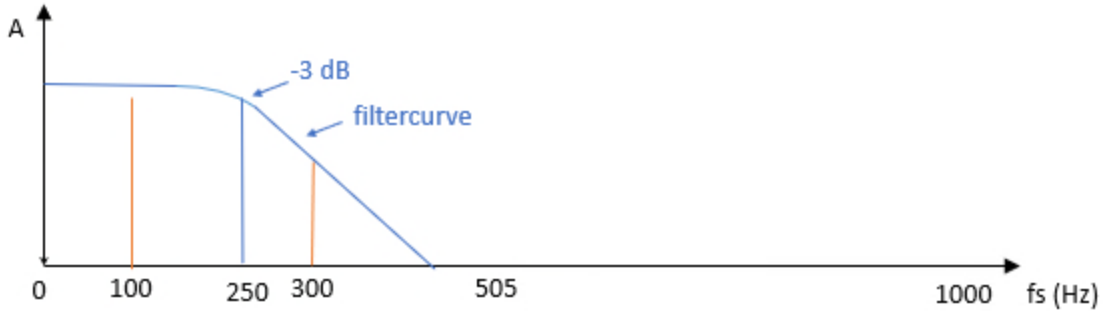
Voorbeeld:

Stel dat de kritische frequentie (afsnijfrequentie) ligt op 0,25 f_s (of 0,25 Hz)

- $f_s = 1\ kHz \Rightarrow f_c = 0,25 \times 1\ kHz = 250\ Hz$

- *Scilab-functies werken met genormaliseerde frequenties*





Eventjes overlopen

- Een ***frequentieselectieve filter*** wordt gebruikt om bepaalde delen van een frequentiespectrum door te laten ofwel te blokkeren.
  - Laagdoorlaat, hoogdoorlaat, banddoorlaat, bandsper

- Een ***digitale transfertfunctie*** wordt geschreven aan de hand van verschilvergelijkingen
  - Maakt gebruik van de huidige waarde van de output sequence y_n, en de huidige waarde van de input x_n en alle vorige waarden van de input- en output sequence

  - $\boldsymbol{voorbeeld\ FIR: G_z = b_0 + b_1 z^{-1} + \ ... + b_n z^{-n}}$

  - $\boldsymbol{voorbeeld\ IIR: G_z = \frac{b_0 + b_1 z^{-1} + ... + b_n z^{-n}}{1 + a_1 z^{-1} + ... + a_n z^{-n}}}$

Eventjes overlopen

- ***Eigenschappen van digitale filters***

  - Afsnijfrequentie (-3 dB)
  - Doorlaatband (zo vlak mogelijk)
  - Transitionband (zo klein mogelijk)
  - Sperband (zo veel mogelijk demping)
  - Filterorde (hoe hoger hoe beter)
  - Filtertype (LD, HD, BD, BS)
  - Filter design procedure (Butterworth, Bessel, Chebyscheff)

Eventjes overlopen

- ***Normalisatie***

  - ***In digitale signaalprocessing wordt de samplefrequentie dikwijls genormaliseerd naar 1,0 Hz zodat de te samplen frequenties liggen tussen 0,0 fs en*** $0,5\ f_s$.

  - ***Dit betekent dat de afsnijfrequentie moet liggen tussen 0,0 fs en*** $0,5\ f_s$

Voorbeeld : Toepassen van een filter op signalen

- Functie ***filter(num, dem, input)***
  - Hiermee kan je een digitale filter met een bepaalde transfertfunctie toevoegen
    - num : staat voor numerator (of teller van de transfertfunctie)
    - dem : staat voor denominator (of noemer van de transfertfunctie)
    - $input$: de te filteren samples

Voorbeeld : Toepassen van een filter op signalen

Stel een eenvoudig moving avarage FIR-filter waarbij de laatste twee ingangssignalen worden opgeteld en gedeeld door 2

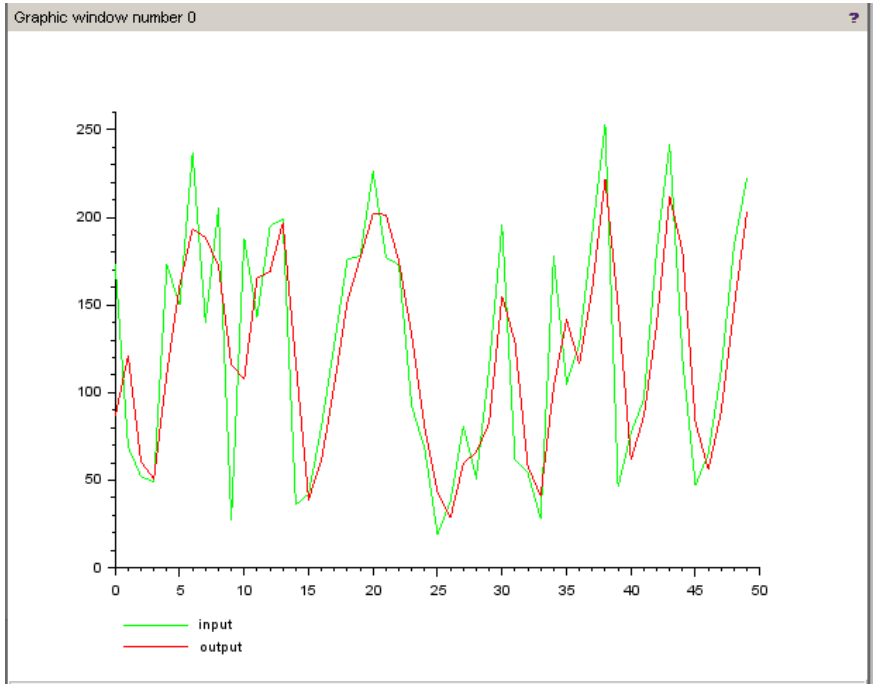
$y_{[n]} = 0,5\ x[n] + 0,5\ x_{[n-1]}$ ***(hogere filterorden kunnen bekomen worden door meer coëfficiënten te gebruiken)***

- ***Coëfficiënten voor dit filter :***
  - ***Teller : 0,5 0,5***
  - ***Noemer : 1 (FIR heeft geen coëfficiënten in de noemer => enkel 1***

- ***Programmatie van de filter:***
  - ***Coëfficiënten ingeven als vectoren (array)***
  - ***Ingangssignaal ingeven als vector (array)***

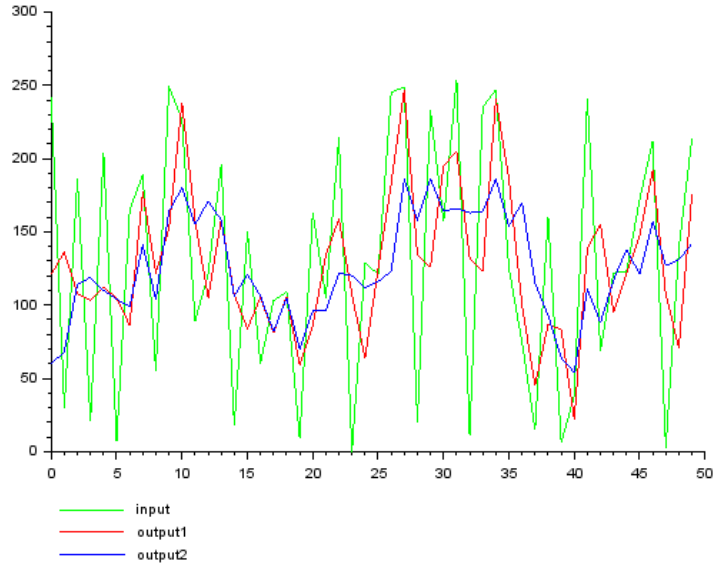
Voorbeeld : Toepassen van een filter op signalen

```
1  //·voorbeeld·van·FIR-filter:
2  //y(n))=0.5x(n)+0.5·x(n-1)
3  clc
4  clf
5
6  //·genereren·van·50·random·samples·(ADC-simulatie)
7  x_waarden·=·int(255*·rand(50,1))
8
9  //·definiëren·filter·met·twee·coëfficiënten·0.5·0.5
10 //·num·=·0.5·0.5;·dem·=·1·(FIR)·input·=·x_waarden
11
12 y_waarden·=·filter([0.5·0.5],[1],x_waarden)
13
14 //·plot·horizontaal·"n"·(aantal·waarden·-·samples)
15 n·=·[0·:·1·:·49]
16
17 //·vertikaal·amplitude·van·de·random·x_waarden
18 //·plot·met·legende·en·aanpassen·kleurstijl
19 plot2d·(n,[x_waarden,y_waarden],·leg·="input@output",...
20 ····style=[color("green"),·color("red")]);
21
```



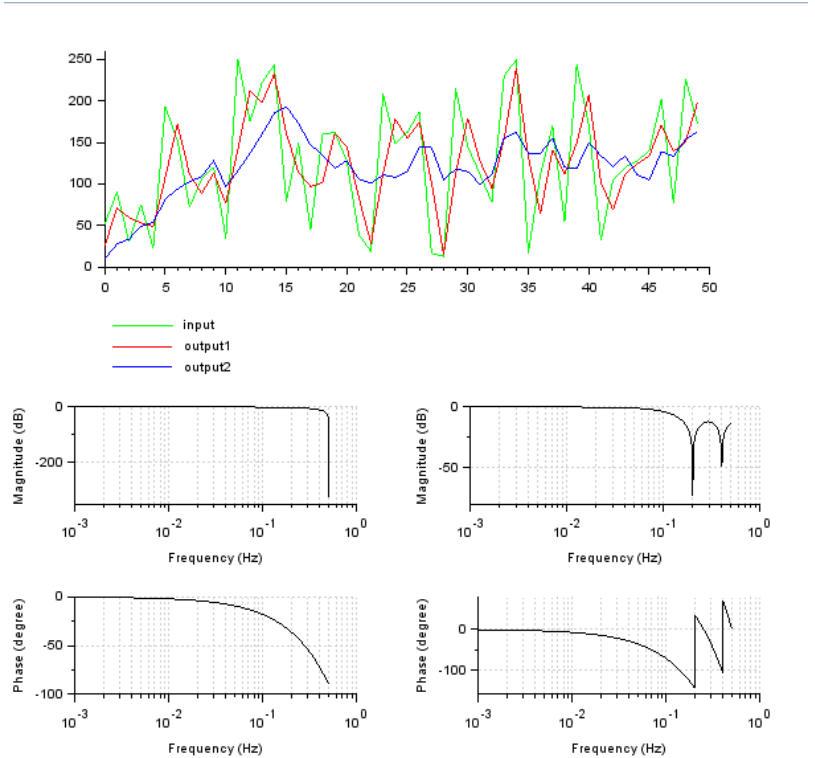
Voorbeeld : Vergelijk met een hogere orde filter

```
1  //·voorbeeld·van·FIR-filter:
2  //y(n))=0.5x(n)+0.5·x(n-1)
3  clc
4  clf
5
6  //·genereren·van·50·random·samples·(ADC-simulatie)
7  x_waarden·=·int(255*·rand(50,1))
8
9  //·definiëren·filter·met·twee·coëfficiënten·0.5·0.5
10 //·num·=·0.5·0.5;·dem·=·1·(FIR)·input·=·x_waarden
11
12 y1_waarden·=·int(filter([0.5·0.5],[1],x_waarden))
13
14 //definiëren·filter·met·4·coëfficiënten
15 y2_waarden·=·int(filter([0.25·0.25·0.25·0.25·],[1],x_waarden))
16 //·plot·horizontaal·"n"·(aantal·waarden·-·samples)
17 n·=·[0·:·1·:·49]
18
19 //·vertikaal·amplitude·van·de·random·x_waarden
20 //·plot·met·legende·en·aanpassen·kleurstijl
21 plot2d·(n,[x_waarden,y1_waarden,·y2_waarden],·leg·="input@output1@output2",...
22 ····style=[color("green"),·color("red"),·color("blue")]);
```



Voorbeeld : Vergelijk met een hogere orde filter

```
//-genereren-van-50-random-samples-(ADC-simulatie)
x_waarden-=-int(255*-rand(50,1))
//-definiëren-filter-met-twee-coëfficiënten-0.5-0.5
//-num-=-0.5-0.5;-dem-=-1-(FIR)-input-=-x_waarden
y1_waarden-=-int(filter([0.5-0.5],[1],x_waarden))
//definiëren-filter-met-4-coëfficiënten
y2_waarden-=-int(filter([0.2-0.2-0.2-0.2-0.2],[1],x_waarden))
//-plot-horizontaal-"n"-(aantal-waarden-=-samples)
n-=-[0-:-1-:-49]
subplot-(2,1,1)
//-vertikaal-amplitude-van-de-random-x_waarden
//-plot-met-legende-en-aanpassen-kleurstijl
plot2d-(n,[x_waarden,y1_waarden,-y2_waarden],-leg-="input@output1@output2",...
----style=[color("green"),-color("red"),-color("blue")]);
para_filter1-=-[0.5-0.5];
para_filter2-=-[0.2-0.2-0.2-0.2-0.2];
filter1_p-=-poly(para_filter1,-'z',-'coeff');
filter2_p-=-poly(para_filter2,-'z',-'coeff');
filter1_h-=-horner(filter1_p,(1/%z));
filter2_h-=-horner(filter2_p,(1/%z));
filter1_s-=-syslin('d',filter1_h);
filter2_s-=-syslin('d',filter2_h);
subplot(2,2,3)
bode-(filter1_s)
subplot(2,2,4)
bode-(filter2_s)
```



Filterontwerp volgens de “window”-methode

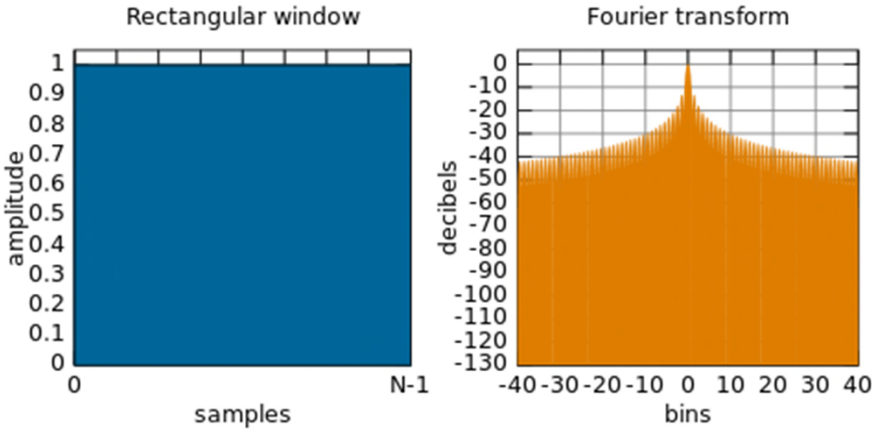
Waarvoor wordt een window gebruikt binnen het ontwerp van een digitale filter?

Om binnen het ontwerp met een ‘ideale’ impulsresponse (bv een sincfunctie die oneindig lang is) deze om te vormen naar een finitive impulse response

Voorbeelden van windows

Rectangular window *(boxcar of Dirichiet)*

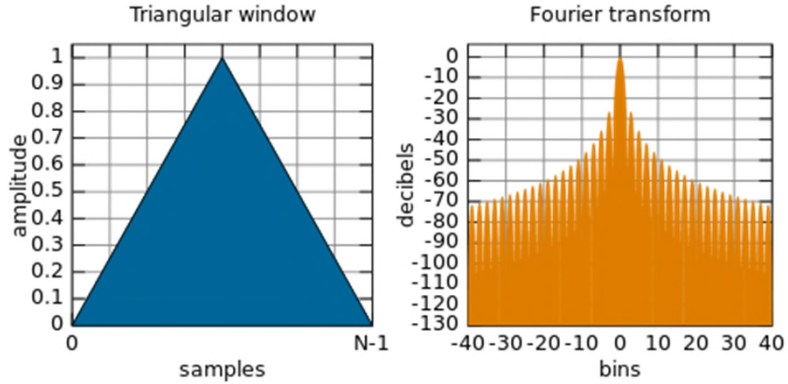
- *Eenvoudigste window-equivalent om alle waarden op nul te krijgen*
- *Alle waarden op nul behalve de waarden 'n' die doorgelaten worden*
- $w(n) = 1$*; met* $w(n)$ *de windowfunctie*
- *Nadeel filter:*
  - *Plotselinge overgang tussen wel doorgelaten en niet doorgelaten ➔ ontstaan ongewenste effecten in de DTFT (discrete time fourier transformatie)*



Voorbeelden van windows

Triangular window

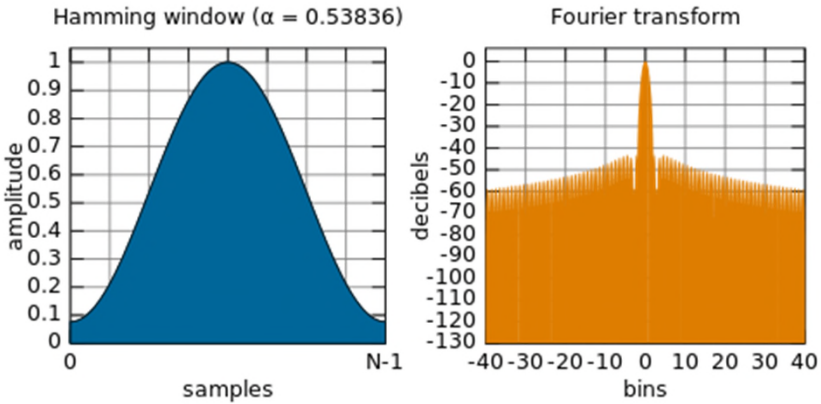
- $w(n) = 1 - n - \frac{\frac{N-1}{2}}{\frac{L}{2}}$
- ***met L gelijk aan $N, N+1$ of $N-1$***
- ***Kan gezien worden als de convolutie van twee $N/2$ brede reactangular windows***



Voorbeelden van windows

Hamming window

- **Windowfunctie :** $w(n) = \alpha - \beta cos\frac{2\pi n}{N-1}$
- **Met α = 0,54 en β= α -1 = 0,46**
- **Het window is zodanig geoptimaliseerd dat het maximum van de zijlob, die het dichtste bij het window ligt, een amplitude heeft van 1/5 van dat van het Hamming window**

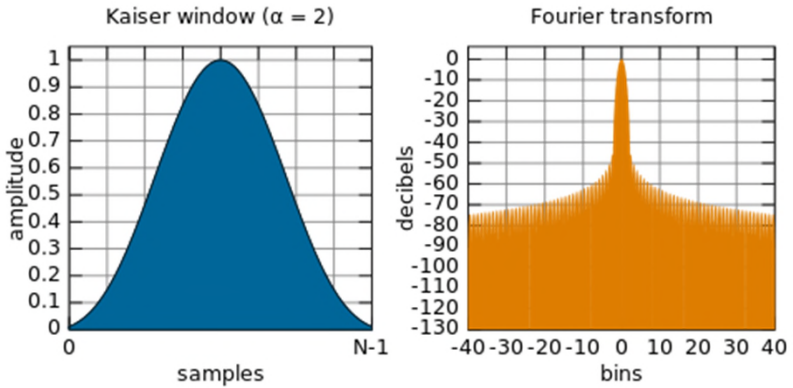


Hamming window, α = 0.53836 and β = 0.46164; B = 1.37. The original Hamming window would have α = 0.54 and β = 0.46; B = 1.3628.

Voorbeelden van windows

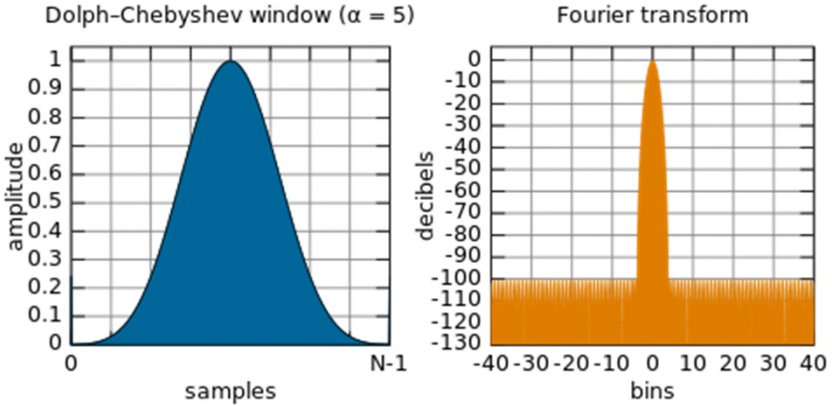
Kaiser window *(Kaiser-Bessel window)*

- Eenvoudige benadering van de DPSS window die gebruik maakt van Besselfuncties. DPSS staat voor Discrete Prolate Spheroidal Sequences
- Windowfunctie : $w(n) = \frac{I_0(\pi\sqrt{1-\left(\frac{2n}{N-1}-1\right)^2})}{I_0(\pi\alpha)}$
- Met I_0 de 0de orde gewijzigde Besselfunctie en α een variabele parameter die de afweging bepaalt tussen de hoofdlob en zijlobniveaus van het spectrale leakage pattern. Typische waarde voor α is drie.



Voorbeelden van windows

- **Dolph-Chebyshev**
  - Minimaliseert de chebyshev norm van de zijlobes voor een gegeven hoofdlobebreedte.
  - De window-functie is meestal gedefinieerd in termen van real-valued discrete Fouriertransformatie $W_0(k)$.

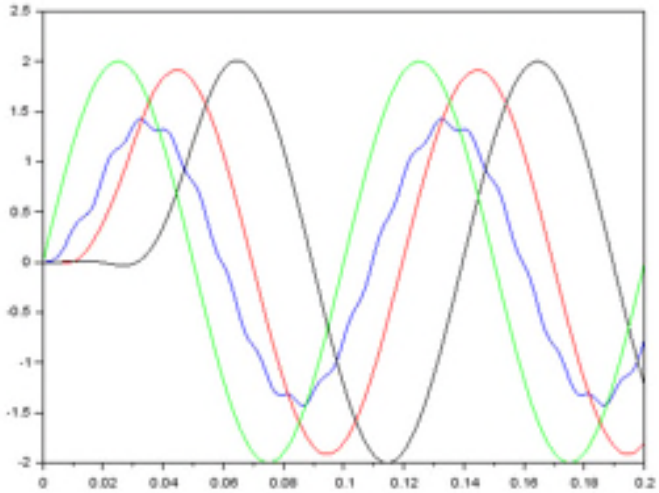


Waarvoor worden windows weer gebruikt bij digitale filters?

- Om binnen het ontwerp met een ‘ideale’ impulsresponse (bv een sincfunctie die oneindig lang is) deze om te vormen naar een finitive impulse response

Invloed van de orde op de filter

- ***Hoe lager de filterorde bij dezelfde afsnijfrequentie (40 Hz), hoe meer gedempt het signaal aan de uitgang verschijnt, en hoe meer zichtbaarder de hogere frequentiecomponenten.***
- **FIR-filter heeft een relatief hoge filterorde nodig om tot goede resultaten te bekomen**
- **Hoe hoger de orde, hoe groter de faseverschuiving**



Groen: originele sinus 10 Hz
Blauw : filteroutput 20ste orde filter
Rood : filteroutput 40ste orde filter
Zwart : filteroutput 80ste orde filter

Even overlopen …

- ***Hoe creëer je nu weer een FIR-filter met lineaire faseresponse in scilab?***

[coefficients, amplitude, frequency] = wfir(filter-type, filter-order, [fg1 fg2], windowtype, [par1 par2])

Wfir heeft volgende parameters nodig:

filter-type

Filter-orde

[fg1 fg2]

window-type : re, tr, hm, kr, ch (rectangular, triangular, hamming, kaiser, chebyshev)

[par1 par2]

De returnwaarden van de functie wfir() zijn:

coefficients : filtercoëfficiënten

amplitude : vector met lengte 256 met de amplitudewaarden

frequency : vector met lengte 256 met de frequenties in het gebied tussen 0 tot 0,5

Even overlopen …

- ***Hoe maak je scilab duidelijk dat het een filter is?***

```
[LD_coeff, amplitude, frequentie]  = ...
      wfir('lp', 40, [0.04 0], 'hm', [0 0]);

LD_polynoom = poly(LD_coeff, 'z', 'coeff’);

LD_functie = horner(LD_polynoom, (1/%z));

LD_lineair_system = syslin('d', LD_functie);
```

Even overlopen …

- ***Hoe maak je het resultaat van de filter zichtbaar in het tijdsdomein?***

- Via flts() kan de filter worden doorlopen met het testsignaal
  - LD_output = flts(testsign, LD_linear_system)

Even overlopen …

- ***Wat is de invloed van de orde en afsnijfrequentie op het uitgangssignaal van een FIR-filter met lineaire response?***

- **FIR-filter heeft een relatief hoge filterorde nodig om tot goede resultaten te bekomen**
  ➔Hoe hoger de orde, hoe groter de faseverschuiving

- **Hoe hoger de afsnijfrequentie, hoe meer invloed van de hogere frequenties in het signaal aanwezig zijn**
- **Hoe lager de afsnijfrequentie bij dezelfde filterorde, hoe gedempter het uitgangssignaal wordt**

Hoe kan je het spectrum van de filter zichtbaar maken in scilab?

- Met de functie fft() kan je een signaal decomposeren in de sinuscompenten waarmee dit signaal is samengesteld.