- Veel coëfficiënten => trager
- Gebruik van wfir()
- ***[coefficients, amplitude, frequency] = wfir(filter-type, filter-order, [fg1 fg2], windowtype, [par1 par2])***

- Minder coëfficiënten voor dezelfde orde als FIR, maar onstabieler en niet-lineaire faseverschuiving
- Gebruik van iir()
- ***iir(filterorde,'filtertype','filterdesign'[fg1 fg2],[par1 par2])***

Hoe ontwerp je een IIR-filter in scilab?

IIR_filter = iir(filterorde,'filtertype','filterdesign'[fg1 fg2],[par1 par2])

- ***filterorde*** : orde van filter
- ***filtertype*** : 'lp', 'hp', 'bp' en 'sb'
- ***filterdesing*** : 'butt', 'cheb1', 'cheb2' en 'ellip'
- ***[fg1, fg2]***
  - 'lp' en 'hp' :$0 < fg1 < 0.5$; $fg2 = 0$
  - 'bp' en 'sb' : $fg1 = f_{cl}; fg2 = f_{ch}$

- ***[par1, par2] :***
  - par1: enkel voor cheb1 filters
  - par2: enkel voor cheb2 filters
  - par1 en par2 enkel voor ellip filters

Invulling:

- cheb1 : 1-par1<rimpel<1 in doorlaatband (passband)
- cheb2 : 0<rimpel<par2 in bandsperband (stopband)
- ellip :
  - 1-par1 < rimpel < 1 in doorlaatband
  - 0<rimpel<par2 in bandsperband

IIR_filter : bevat een single input en single output van de discrete transfertfunctie van de filter

Hoe ontwerp je een IIR-filter in scilab?

[p, g, z]= iir(filterorde,'filtertype','filterdesign'[fg1 fg2],[par1 par2])

- p : vector met de getransformeerde filterpolen
- z : vector met de getransformeerde nullen
- g : scalair getransformeerde filterversterking

Even overlopen : IIR-filter

IIR_filter = iir(filterorde,'filtertype','filterdesign'[fg1 fg2],[par1 par2])

[p, g, z]= iir(filterorde,'filtertype','filterdesign'[fg1 fg2],[par1 par2])

- p : vector met de getransformeerde filterpolen
- z : vector met de getransformeerde nullen
- g : scalair getransformeerde filterversterking

Even overlopen : IIR-filter

IIR_filter = iir(filterorde,'filtertype','filterdesign'[fg1 fg2],[par1 par2])

Tijdsdomeinresponse :

LD_response : flts(testsignaal, iir_filter)

Frequentiedomeinresponse :

[hm, fr] = frmag(sys,npts) vb: [hm,fr]=frmag(IIR_filter, 128)

- **hm : vector van magnitude van de frequentieresponse**
- **fr : aantal punten in de genormaliseerde frequentieresponse die geevalueerd worden binnen het genormaliseerd frequentiedomein**
- **sys : transferfunctie, polynoon of de vector met polynoomcoëfficiënten**
- **npts : integer, het aantal punten in de frequentieresponse**

Introductie tot windowed-sinc filters

Gebruik: scheiden van een band van frequenties van elkaar

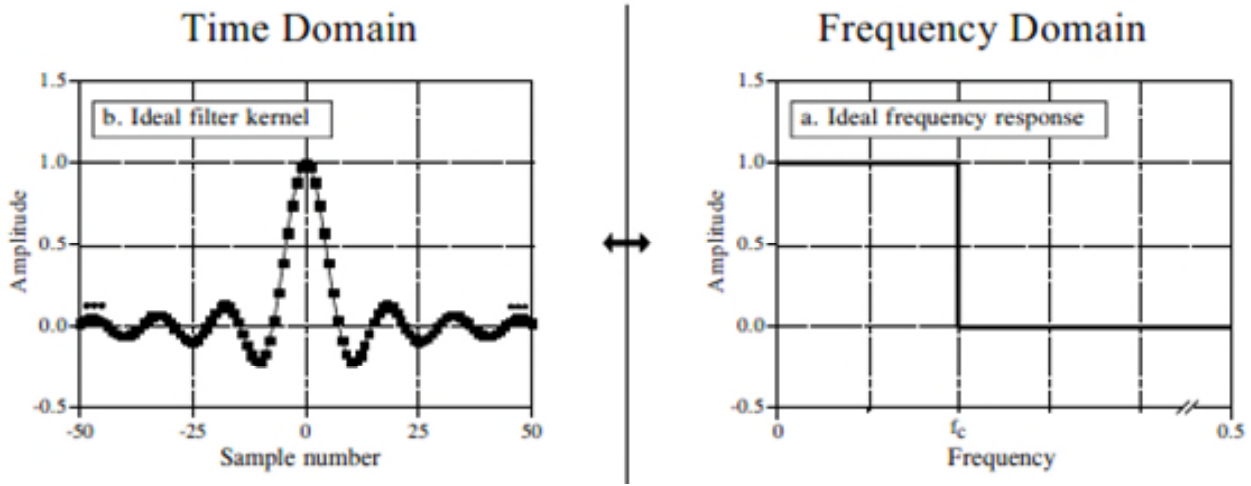
Zijn erg stabiel en leveren goede tot zeer goede prestaties

Nadeel : slechte prestaties in het tijdsdomein: overmatige rimpel en overschrijding van de stap-respons

Zijn eenvoudig te programmeren indien ze uitgevoerd worden met een standaard convolutie maar zijn langzaam in uitvoering

Met FFT verbeteren deze filters drastisch in rekensnelheid

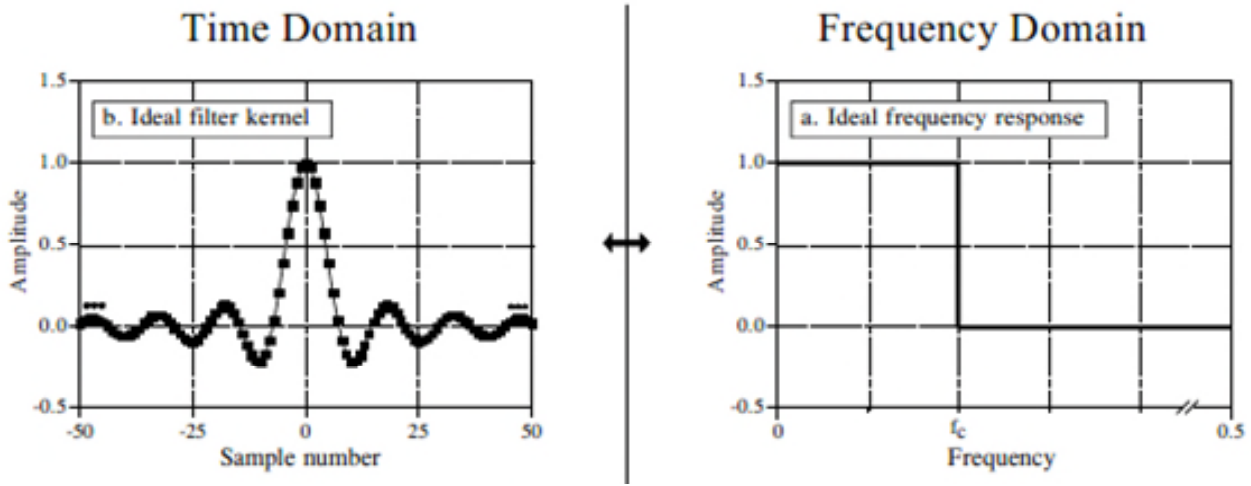
Frequentieresponse van een ideale LD-filter



$f < f_c$: *doorgelaten met eenheidsamplitude*

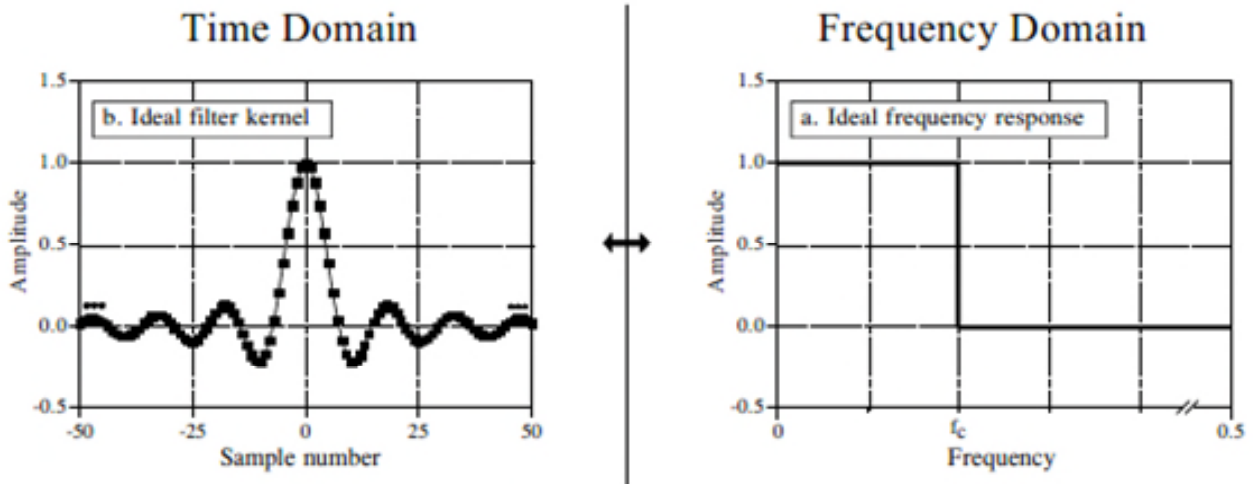
$f > f_c$: *geblokkeerd*

Frequentieresponse van een ideale LD-filter



Doorlaatband perfect plat
Verzwakking stopband oneindig groot
Overgang tussen doorlaatband en stopband is oneindig klein

Frequentieresponse van een ideale LD-filter

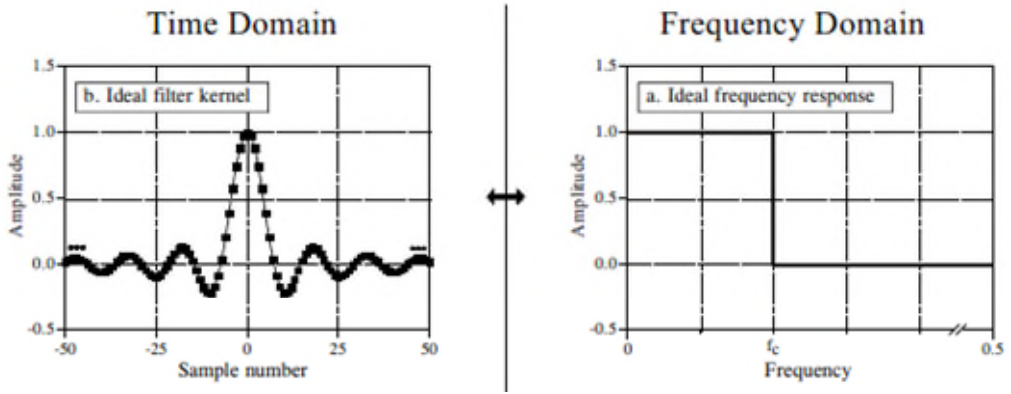


Inverse Fouriertransformatie van ideale frequentieresponse levert de ideale filterkernel op (figuur b)

Perfecte LD-filter

$$h_{(i)} = \frac{\sin(2\pi f_c i)}{i\pi}$$

Frequentieresponse van een ideale LD-filter



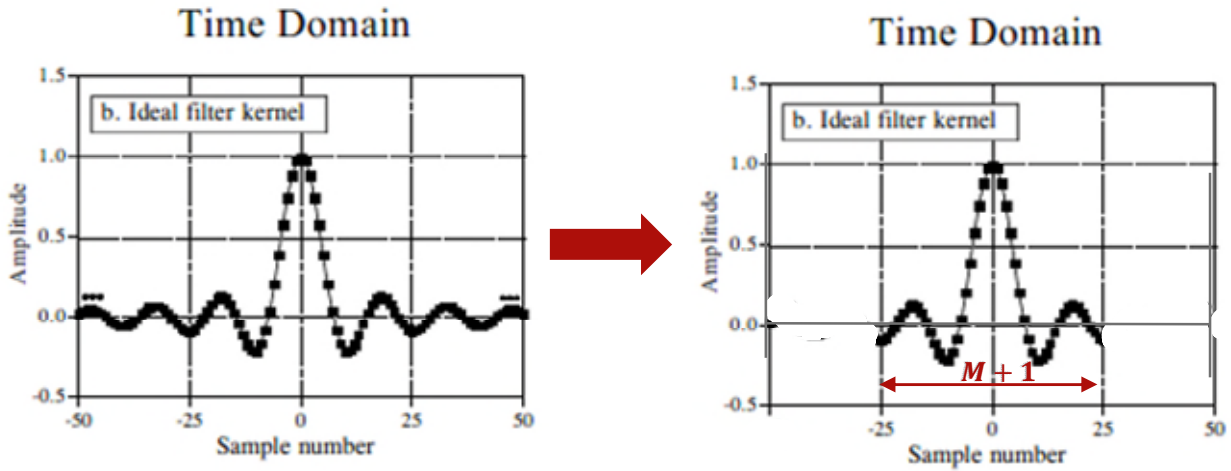
$$h_{(i)} = \frac{\sin(2\pi f_c i)}{i\pi}$$

Probleem: sinc-functie loopt door naar plus en min oneindig
➔ computerprobleem

Strategie van Windowed-Sinc

Omzeilen computerprobleem

1ste aanpassing: sinc-functie afkappen op M+1 punten (M is een even getal en alles buiten $M + 1$ wordt door nullen vervangen)

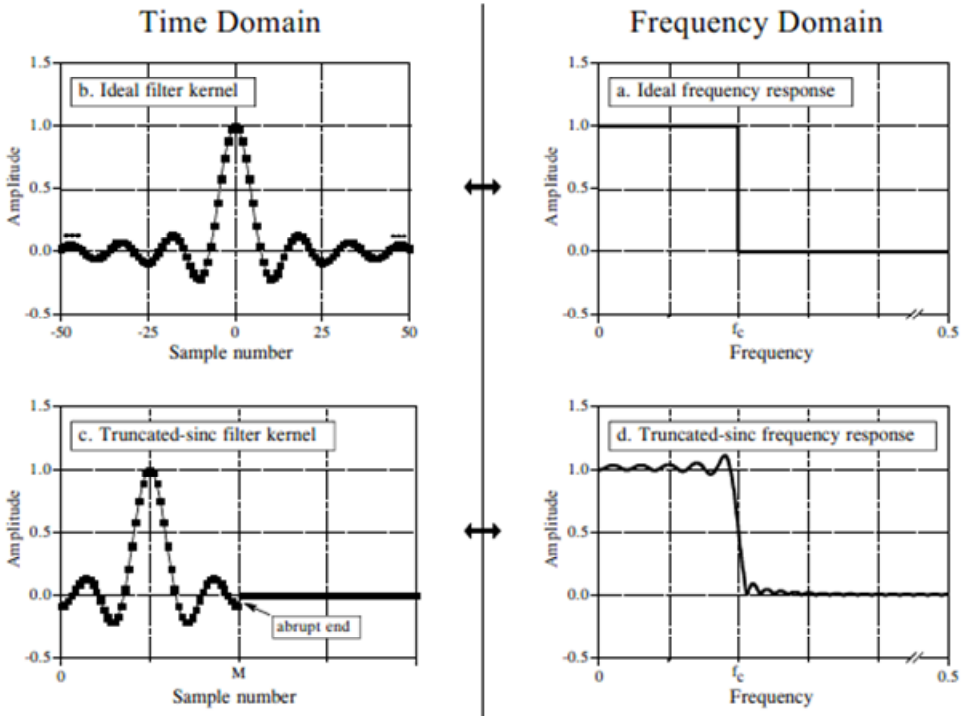


Strategie van Windowed-Sinc

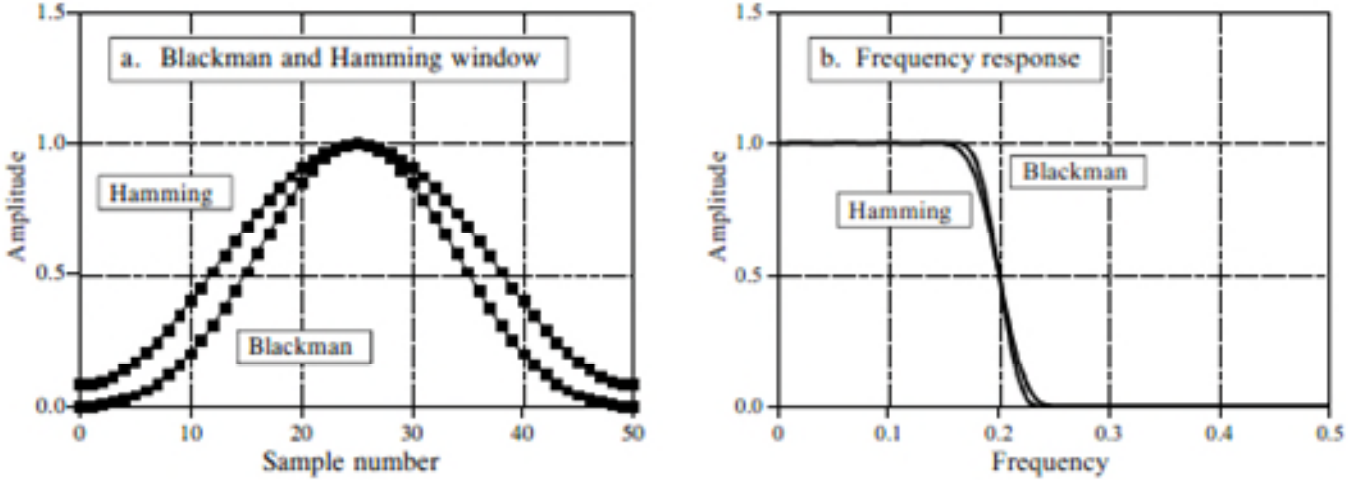
Omzeilen computerprobleem

2de aanpassing: hetgeen overblijft van de sinc-functie verplaatsen naar rechts zodat deze loopt van 0 tot M

- *Voordeel :*
  - *Kernel bestaat uit enkel positieve indexen*
  - *Nadeel:*
    - *Geen ideale frequentietransponse meer maar een benadering => rimpel in doorlaatband en slechtere demping in stopband*



Blackman versus Hamming window

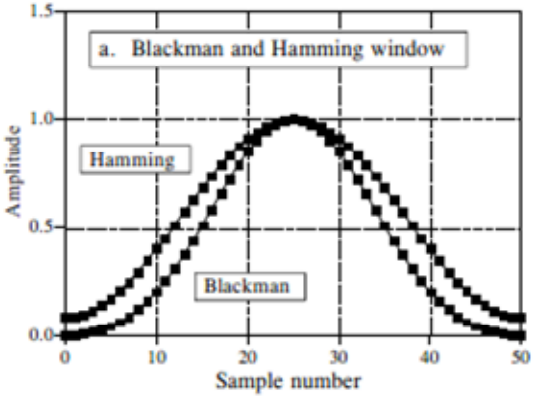


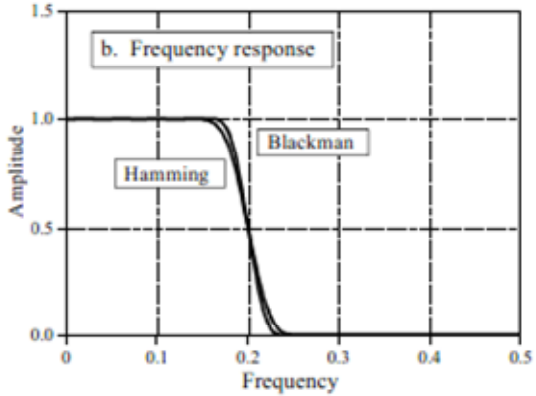
- *Blackman window:* $w[i] = 0{,}42 - 0{,}5\cos\left(\frac{2\pi i}{M}\right) + 0{,}08\cos\left(\frac{4\pi i}{M}\right)$
- *Hamming window:* $w[i] = 0{,}54 - 0{,}46\cos(\frac{2\pi i}{M})$

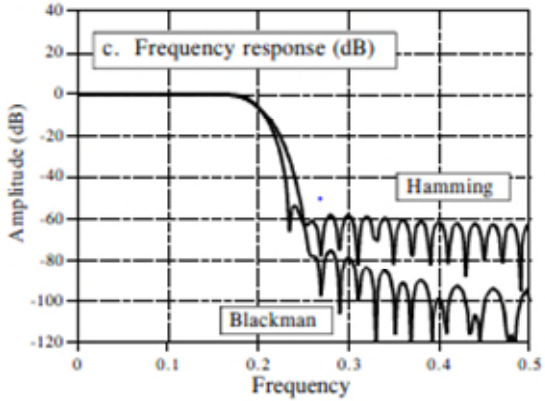
Blackman versus Hamming window

- *Roll-of*
  - *Hamming heeft een 20% snellere roll-off dan Blackman*
- *Stopbandverzwakking*
  - *Blackman (-74 dB) of 0,02%*
  - *Hamming (-53 dB) of 0,2 %*
- *Rimpel doorlaatgebied*
  - *Blackman 0,02%; Hamming 0,2 %*

Blackman verdient voorkeur op Hamming (trage roll-of geter aanpasbaar dan een zwakke stopbandverzwakking)





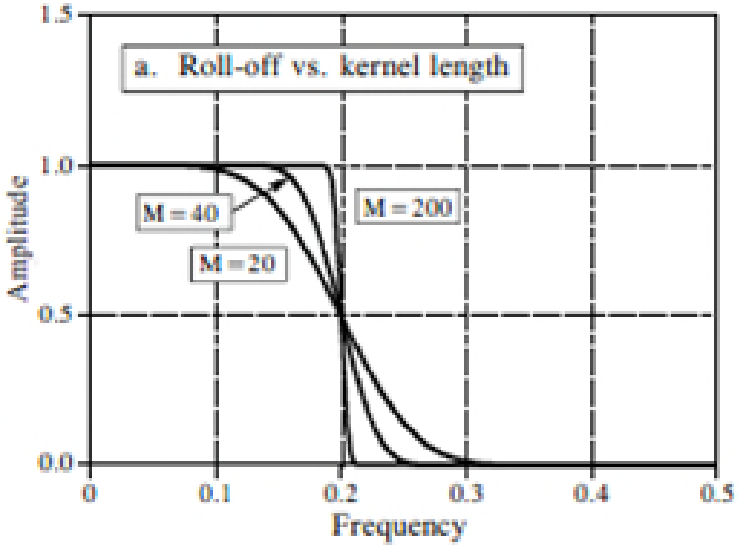


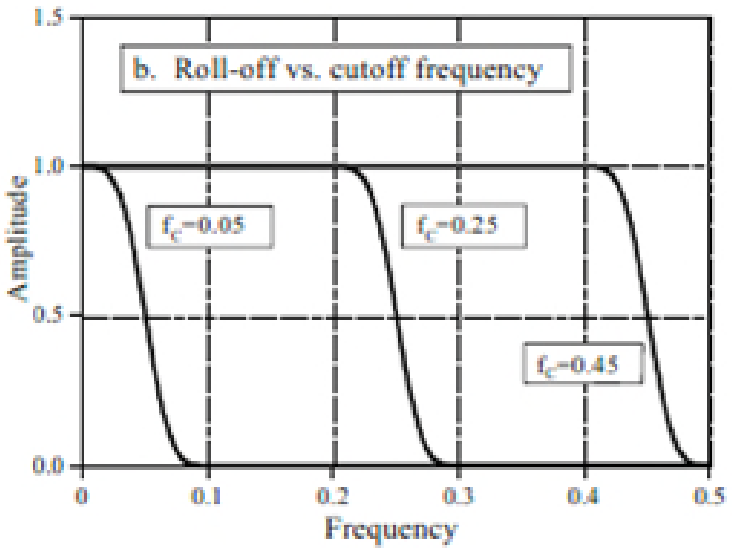
Waar dien je rekening mee te houden bij een window-sync filterontwerp?

- *Belangrijke parameters:*
  - *Afsnijfrequentie*
    - *Wordt uitgedrukt als een fractie van de samplerate : (0 <f_c<0,5)*
  - *De waarde van M*
    - *Bepalend voor de roll-of*
    - *$M = \frac{4}{BW}$; BW is de breedte van de overgang van doorlaatband naar stopband*

Waar dien je rekening mee te houden bij een window-sync filterontwerp?

$M = \frac{4}{BW}$; *BW is de breedte van de overgang van doorlaatband naar stopband*





Voorbeelden:

- *Fig(a) : 3 LD-filters met* M gelijk aan 20, 40 en 200
- *Fig(b) : doorlaatbanden met* fc = 0,05, 0,25 en 0,45

Waarmee rekening houden bij het filterontwerp?

De rekentijd om de convolutie uit te rekenen is afhankelijk van de lengte van de signalen en de grootte van de filterkernel ➔ ***afweging maken tussen de lengte M (filterkernel) en de BW*** (scherpte van de filter)

$$M = \frac{4}{BW}$$

Afsnijfrequentie doorlaatband meestal bepaald op halve amplitudepunt ipv -3 dB-punt

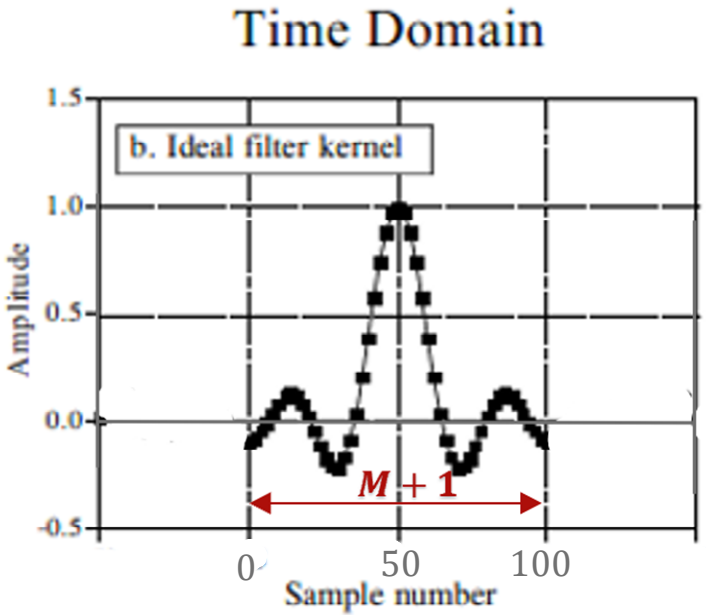
- ***Waarom?***
  - ***Window-sinc frequentieresponse ligt symmetrisch tussen doorlaatband en sperband => faselineariteit***

Hoe een filter ontwerpen?

- Stap 1 : bepalen type filter en de afsnijfrequentie(s)
- *Stap 2 : Bepalen aantal punten M tussen banddoorlaar en bandstop*
- *Stap 3 : Berekenen van de filterkernel afhankelijk van het type window*
  - *Voorbeeld:*
    - *Filterkernel die een combinatie is van een sinc-functie, Blackman-window en een $M/2$-schift*
    - $h[i] = K \frac{\sin\left(2\pi f_c\left(i-\frac{M}{2}\right)\right)}{i-\frac{M}{2}} [0.42 \ - 0.5\cos\left(\frac{2\pi i}{M}\right) + 0.08\cos\left(\frac{4\pi i}{M}\right)]$
    - *K wordt zodanig gekozen dat de som van alle samples gelijk is aan 0*
      - *Praktisch: negeer K tijdens de berekeningen en normaliseer alle samples waar nodig*

Veronderstel we ontwerpen een filter en kiezen $M = 100$. Wat houdt dat in?

- ***Kernel bestaat uit $M + 1 = 101$ punten***
  - **Eerste punt in de filterkernel is in de array op positie 0**
  - **Laatste punt in de filterkernel is in de array op positie 100**
- ***Het centrum van de symmetrie is op punt 50***
  - **Centrum is M/2**
  - **50 linkse punten van $M/2$ zijn symmetrisch ten opzichte van de 50 rechtse punten**
- ***Als een bepaald aantal samples nodig zijn om FFT te gebruiken vul je het tekort met nullen aan***
  - **Vb: $M = 100$ ➔ 101 punten; dichts bijzijnde macht van 2 is 128 ➔ 27 nullen toevoegen zodat een FFT met 128 punten kan gebruikt worden**

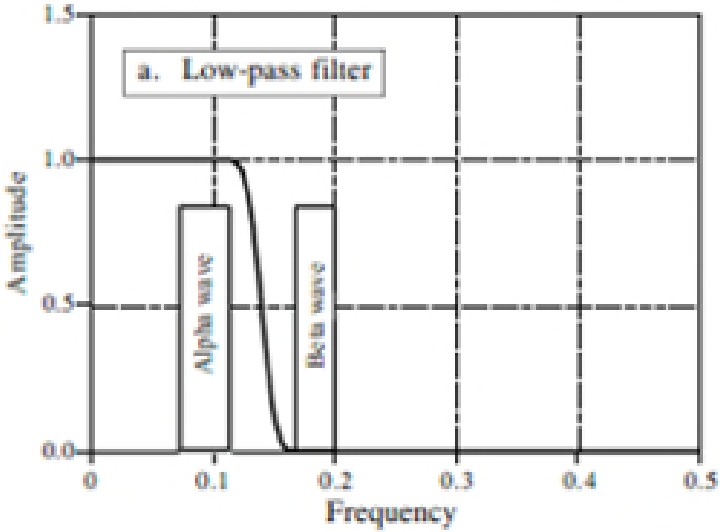


Voorbeeld van ontwerp windowed-sync filter.

- Een EEG (Elektro-encefalografie) is een meting van de elektrische activiteit van de hersenen.
  - Wordt gedetecteerd als mV-signalen en verschijnen op elektroden die aan het hoofd bevestigd zijn.
  - EEG is resultaat van een groot aantal van deze elektrische pulsen
- Verschillende frequenties in het EEG kunnen worden geïdentificeerd voor verschillende mentale toestanden
  - Ogen gesloten, ontspannen levert een signaal op tussen 7 en 12 Hz (alfa ritme genoemd)
  - Openen van ogen en rond kijken levert een bèta ritme op met frequenties tussen 17 en 20 HZ
- Signalen EEG via analoge elektronica versterkt en vervolgens gedigitaliseerd met sampling rate van 100 samples per seconde (fs = 100Hz)
- Verwerven van gegevens voor 50seconden levert een signaal op van 50 x 100 = 5000 punten
- Doel van de filter:
  - Alfa ritmes scheiden van bèta ritmes via een laagdoorlaatfilter met afsnijfrequentie 14 Hz of 14/100Hz = 0,14

Voorbeeld van ontwerp windowed-sync filter.

- Ontwerp filter:
  - Afsnijfrequentie fc = 14 Hz => normaliseren fc/fs = 14 Hz / 100 Hz = 0,14
  - Alfasignalen liggen tussen 7 en 12 Hz; bèta tussen 17 en 20 HZ => afstand is 5 HZ (kiezen BW = 4 HZ => normalisatie 0,04
  - M = 4 / 0,04 = 100 => 101 punten nodig voor symmetrische opbouw
  - Keuze voor Hammingwindow voor dit vb.



$$h[i] = K\frac{\sin\left(2\pi f_c\left(i-\frac{M}{2}\right)\right)}{i-\frac{M}{2}}(0.54-0{,}46\cos(\frac{2\pi i}{M})$$

Voorbeeld van ontwerp windowed-sync filter.

```
PI = 3,14159265;
//genormaliseerde afsnijfrequentie
FC = 0.14
// instellen kernel op 100 punten (totaal aantal = 101)
M=100

//bepalen LD filterkernal hammingwindow
H = zeros(1:101)
for i=1 : 101
    if (i-M/2-1) ==0 then
        H(i) = 2 * PI * FC
    end
    if (i-M/2-1) <> 0 then
        H(i) =sin(2*PI*FC*(i-M/2-1))/(i-M/2-1);
    end
    H(i) = H(i)*(01.54 - 0.46*cos(2*PI*(i-1)/M));
end
```

−1 vermits in scilab de eerste positie van een vector 1 is in plaats van 0

$$h[i] = K\frac{\sin\left(2\pi f_c\left(i-\frac{M}{2}\right)\right)}{i-\frac{M}{2}}(0.54-0{,}46\cos(\frac{2\pi i}{M})$$

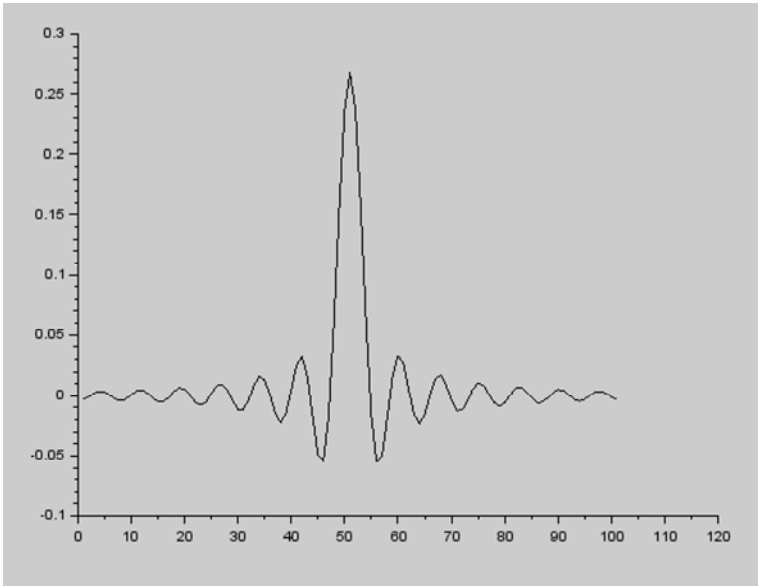
$$h[i] = K\frac{\sin\left(2\pi f_c\left(i-\frac{M}{2}\right)\right)}{i-\frac{M}{2}}(0.54-0{,}46\cos(\frac{2\pi i}{M})$$

$$h[i] = K\frac{\sin\left(2\pi f_c\left(i-\frac{M}{2}\right)\right)}{i-\frac{M}{2}}(0.54-0{,}46\cos(\frac{2\pi i}{M})$$

Voorbeeld van ontwerp windowed-sync filter.

```
//normaliseren LD-filter voor eenheid (bepalen van K in kernelformule)
som = 0
for i=1 : 101
    som = som + H(i)
end
for i=1 : 101
    H(i) = H(i)/som
end
```

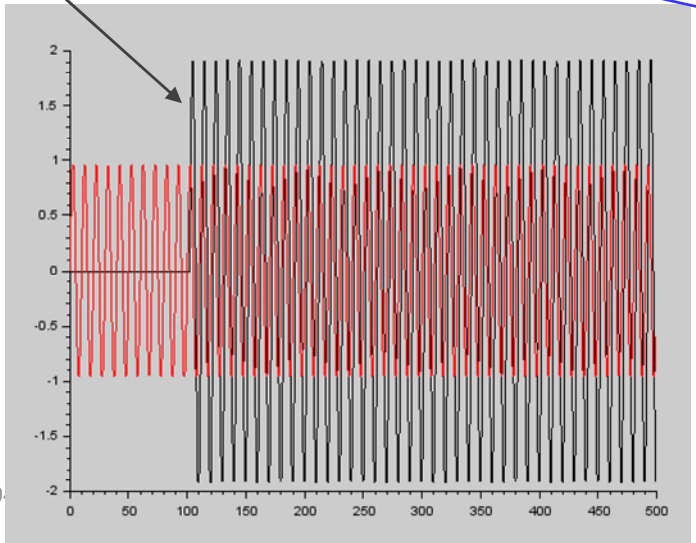
$$h[i] = \boldsymbol{K}\frac{\sin\left(2\pi f_c\left(i-\frac{M}{2}\right)\right)}{i-\frac{M}{2}}(0.54-0{,}46\cos(\frac{2\pi i}{M}))$$

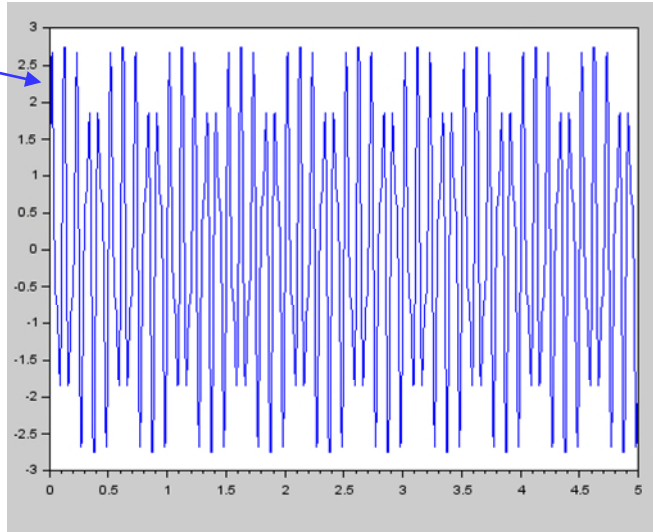


Voorbeeld van ontwerp windowed-sync filter.

```
41 // convolutie tussen ingangssignaal en kernel van de filter
42 output = zeros(1:499)
43 for j = 102 : 499
44     for i=1 : 101
45         .....
46         output(j) = output(j)+ testsignaal(j-i)*H(i)
47     end
48 end
```

1ste ingangssample waar de volledige 101-punten filterkernel op kan bewerken





Wat als we de 18 Hz willen uitfilteren ?

- Ontwerpen van een HD-windowed-sinc filter
- Hoe doe je dat?

➢ Stap 1: ontwerp van LD-filter

➢ Stap 2 : spectrale inversie van de kernel

  ➢ Alle punten in de kernel inverteren

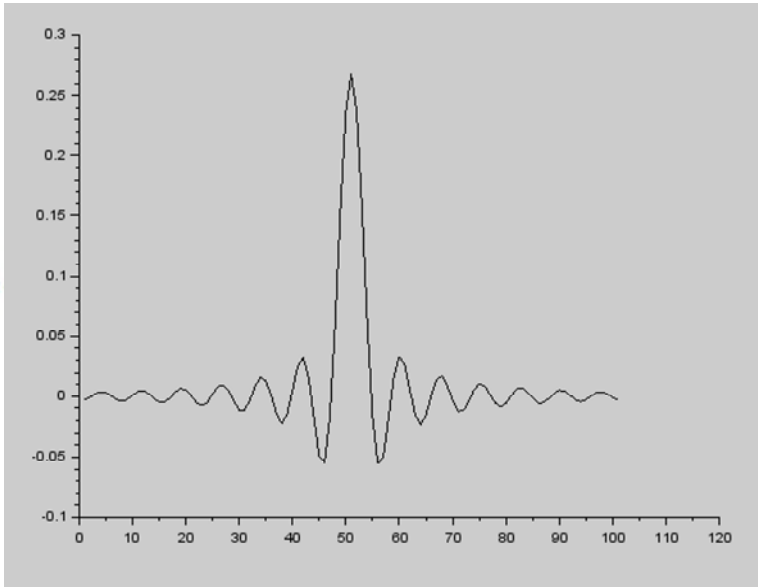
  ➢ Middelste punt (M/2) de eenheidspuls (1) bijtellen

Hetzelfde voorbeeld als de LD-filter maar nu 18 Hz filteren

- **Stap 1: Ontwerp van 14 Hz LD-filter-kernel**

$$h[i] = K\frac{\sin\left(2\pi f_c\left(i-\frac{M}{2}\right)\right)}{i-\frac{M}{2}}(0.54-0{,}46\cos(\frac{2\pi i}{M})$$

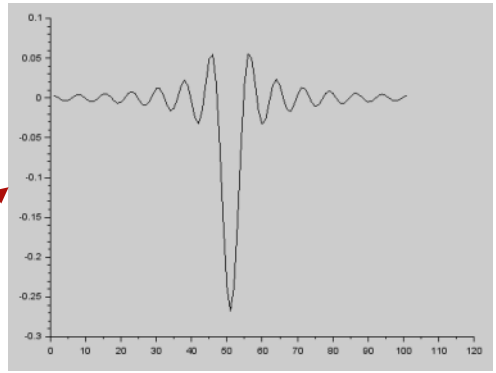
```
21 //bepalen LD filterkernal hammingwindow
22 H = zeros(1:101)
23 for i=1 : 101
24     if (i-M/2-1) ==0 then
25         H(i) = 2 * PI * FC
26     end
27     if (i-M/2-1) <> 0 then
28         H(i) =sin(2*PI*FC*(i-M/2-1))/(i-M/2-1);
29     end
30     H(i) = H(i)*(01.54 - 0.46*cos(2*PI*(i-1)/M));
31 end
32 //normaliseren LD-filter voor eenheid
33 som = 0
34 for i=1 : 101
35     som = som + H(i)
36 end
37 for i=1 : 101
38     H(i) = H(i)/som
39 end
```

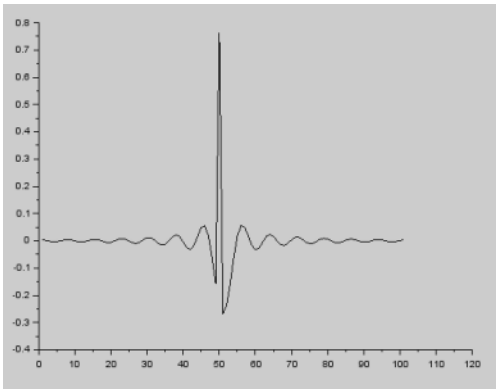


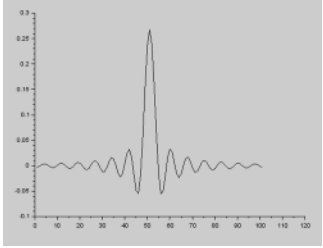
Hetzelfde voorbeeld als de LD-filter maar nu 18 Hz filteren

- **Stap 2: Spectrale inversie LD-kernel om HD-kernel te bekomen**

```
57 HD = zeros(1:101)
58 for i=1 :101
59     HD(i) = -H(i)
60 end
61 HD(50)=HD(50)+1
```





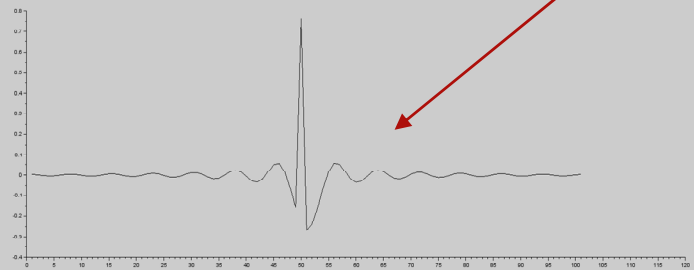


Resultaat HD-filter

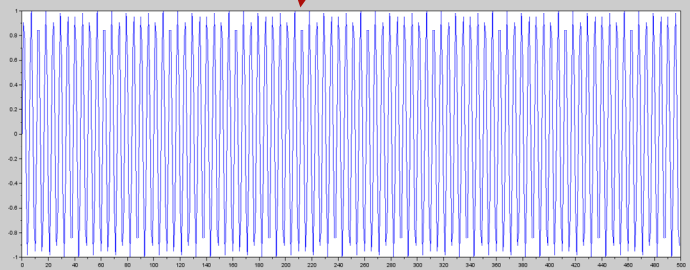
- **Uitvoeren convolutie**

```
67 out_HD = zeros(1:499)
68 for j = 102 : 499
69     for i=1 : 101
70         out_HD(j) = out_HD(j)+ testsignaal(j-i)*HD(i)
71     end
72 end
```

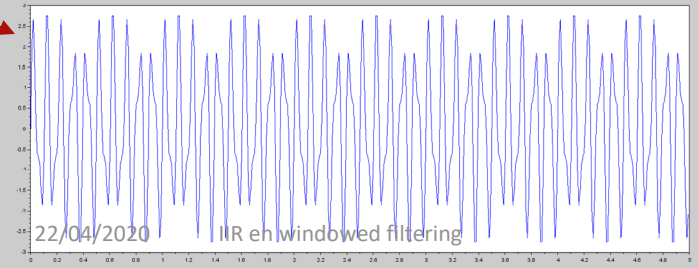
kernel



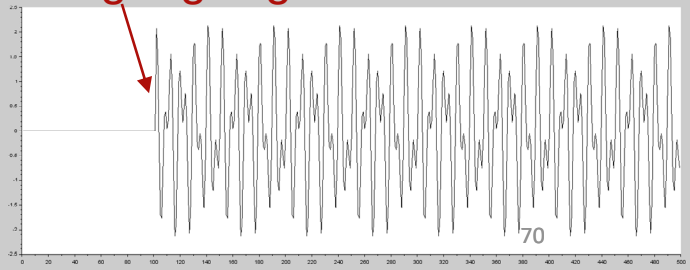
Sinus 18 Hz



testsignaal



uitgangssignaal



Hoe bandfilter maken?

- Stap 1: Bepalen van genormaliseerde afsnijfrequenties
- Stap 2: Bepalen van het aantal punten M
- Bereken de LD-filter kernel met de laagste afsnijfrequentie
- Normaliseer de LD-filterkernel met de laagste afsnijfrequentie
- Bereken de LD-filterkernel met de hoogste afsnijfrequentie
- Normaliseer de LD-filterkernel met de hoogste afsnijfrequentie
- Vorm de LD-filterkernel met de hoogste afsnijfrequentie om tot HD-filterkernel (via spectrale inversie en 1 bijtellen op punt M/2)
- Tel de LD- en HD-filterkernels op waarmee een bandsperfilter wordt bekomen.
- Voer convolutie uit met de kernel van de bandsperfilter
- Vorm de bandsperfilter om (via spectrale inversie en 1 optellen bij het punt M/2) om een banddoorlaatfilter te bekomen
- Voer convolutie uit met de kernel van de banddoorlaatfilter