Windowed Sinc Filters

Introductie 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

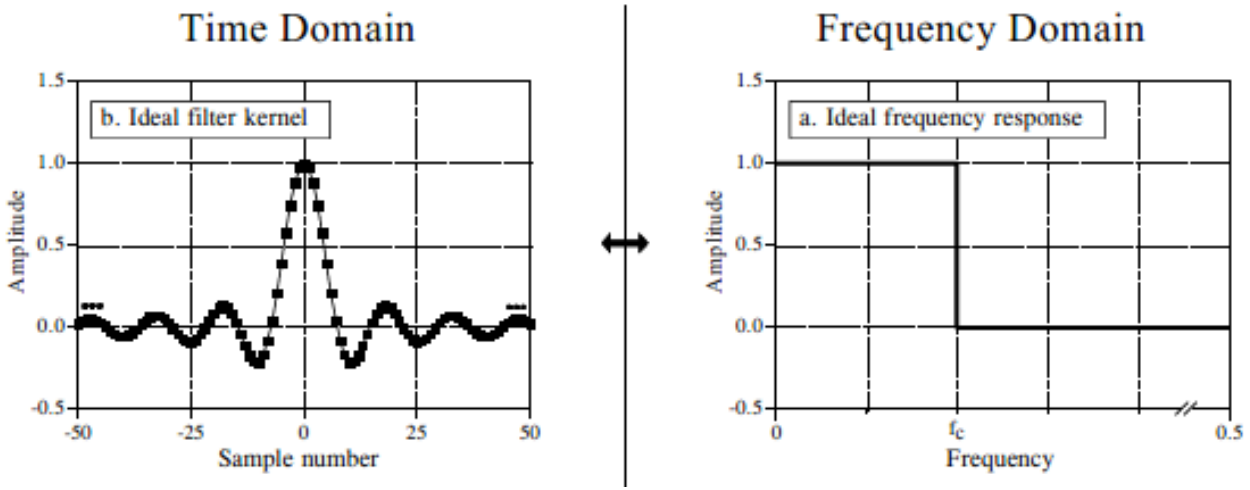
Strategie van Windowed-Sinc

Fig(a) Frequentierespons van ideale LDF

- Alle f's < fc worden doorgelaten met eenheidsamplitude
- Alle f's > fc worden geblokkeerd
- Doorlaatband perfect plat en de verzwakking in de stopband is oneindig groot
- De overgang tussen de twee is oneindig klein

Inverse Fourier Transformatie van deze ideale frequentieresponse levert de ideale filter-kernel op (zie fig (b) => sinc functie gegeven door :

$$h[i] = \frac{\sin(2\pi f_c i)}{i\pi}$$



Convolving een ingangssignaal met deze filter-kernel levert een perfect laagdoorlaatfilter op.
Probleem: sinc-functie loopt door naar plus en min oneindig => computerprobleem

Strategie van Windowed-Sinc

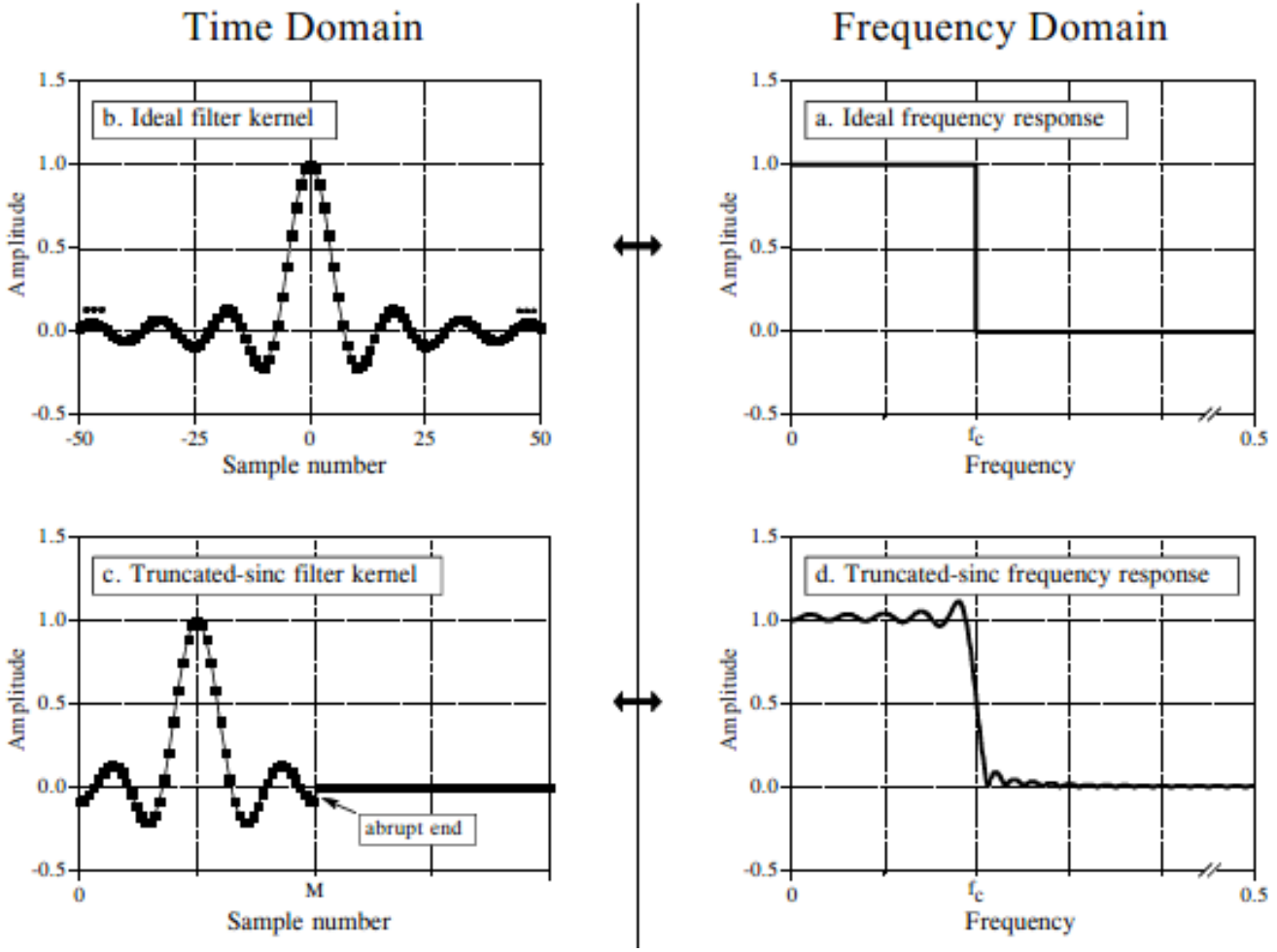
Omzeilen computerprobleem: 2 aanpassingen

1ste aanpassing: sinc-functie van (b) aanpassen aan sinc-functie van (c).

- Hoe? Sinc-functie afkappen op M+1 punten (met M een even getal) (alles buiten M+1 wordt vervangen door nullen

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

- Voordeel: filter-kernel bestaat enkel uit positieve indexen
- Nadeel : wijzigingen zorgen voor slechts benadering van ideale filter-kernel => geen ideale frequentieresponse hebben
- Vinden frequentie via Fourier Transformatie van signaal (c) naar (d) => rimpel in doorlaatband (nadeel) en slechtere demping in stopband
- Er is een eenvoudige methode om deze situatie te verbeteren



Strategie van Windowed-Sinc

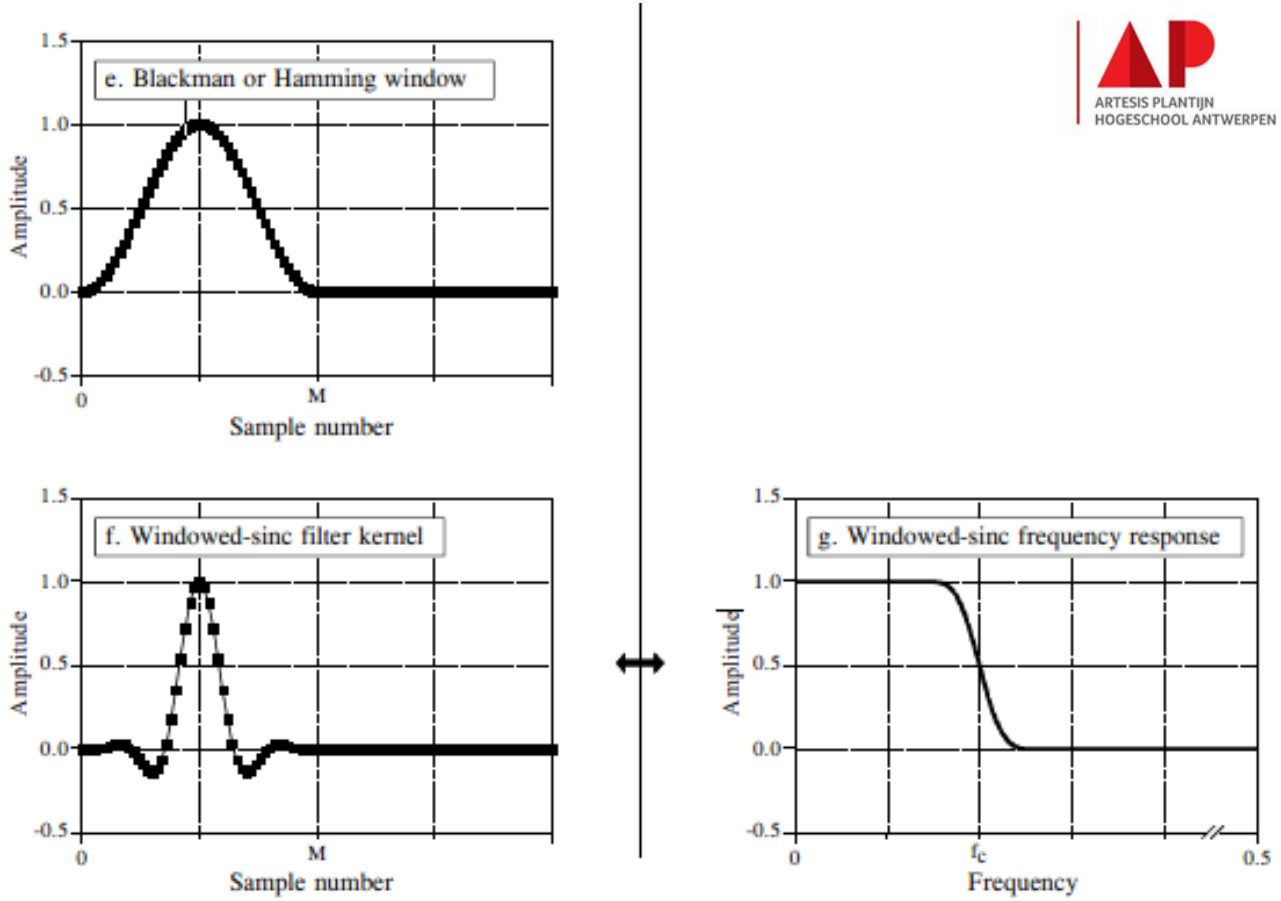
Oplossen slechtere karakteristieken door afkappen sinc-functie

Figuur (e) toont zogenaamd ***Blackman window***

Afgekapte sinc-functie met deze Blackman window vermenigvuldigen levert het windowed-sinc filterkernel (f)

Fig (g) toont het verbeterde frequentiekarakteristiek

- doorlaatband terug platter
- Sterk verbeterde stopbanddemping (niet meer zichtbaar in de grafiek)



Verschillende vensters zijn ontwikkeld sinds de jaren 50 om de windowed-sinc filter te verbeteren . ***Enkel Hamming-venster en Blackman-venster zijn echt bruikbaar.***

Hamming-window:

$$w[i] = 0.54 - 0.46\cos(2\pi i/M)$$

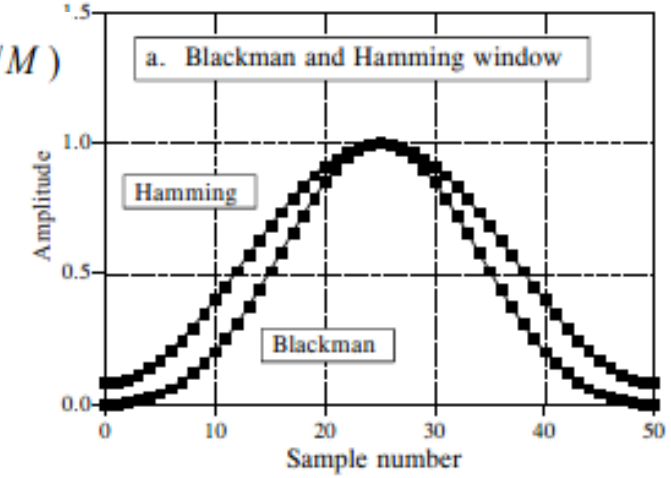
Blackman window:

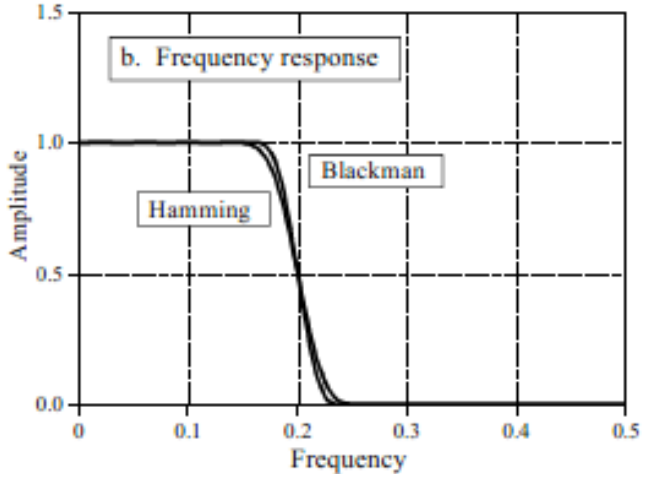
$$w[i] = 0.42 - 0.5\cos(2\pi i/M) + 0.08\cos(4\pi i/M)$$

Strategie van Windowed-Sinc

Blackman window: $w[i] = 0.42 - 0.5\cos(2\pi i/M) + 0.08\cos(4\pi i/M)$

Hamming window: $w[i] = 0.54 - 0.46\cos(2\pi i/M)$

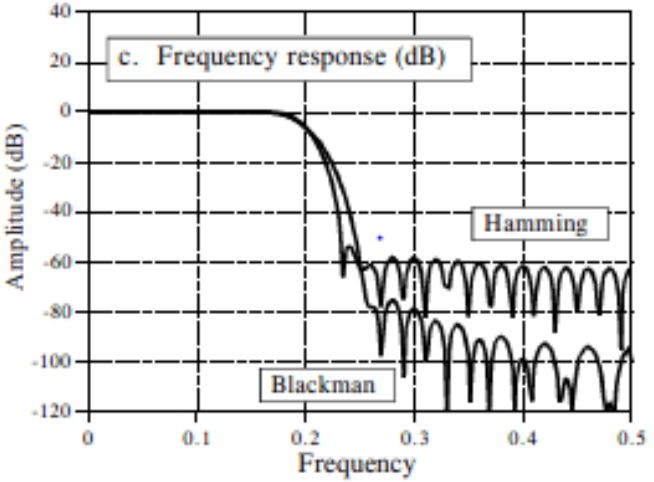




Figuur toont de vorm van deze twee windows voor M= 50 (totaal 51 punten in de curves)

Vergelijk Blackman – Hamming

- Hammingwindow is 20% snellere roll-of dan Blackman
- Blackman heeft een betere stopbandverzwakking (fig (c) (-74 dB (-0,02%) tegen -53 dB (-0,2%)
- Blackman heeft een rimpel in de doorlaatband (niet zichtbaar van ongeveer 0,02% terwijl Hamming een rimpel heeft van typisch 0,2%
- In het algemeen verdient Blackman de voorkeur op Hamming (trage roll-of is beter aanpasbaar dan een zwakke stopbandverzwakking

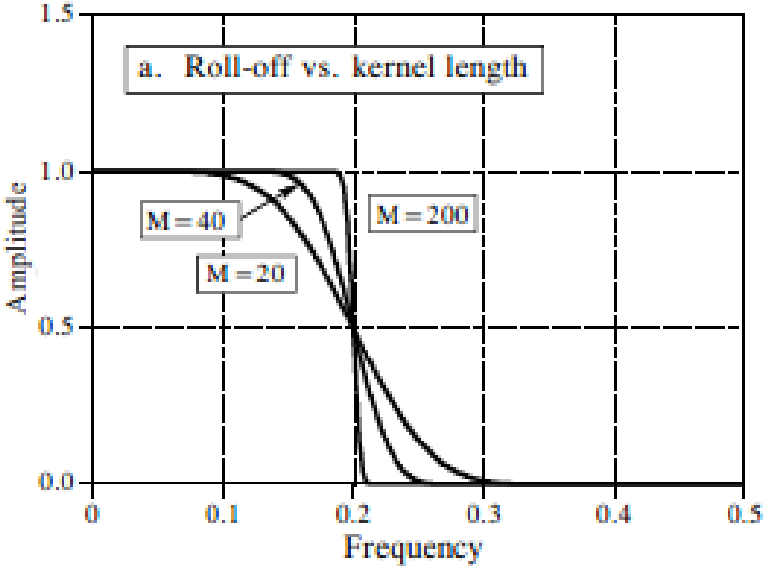


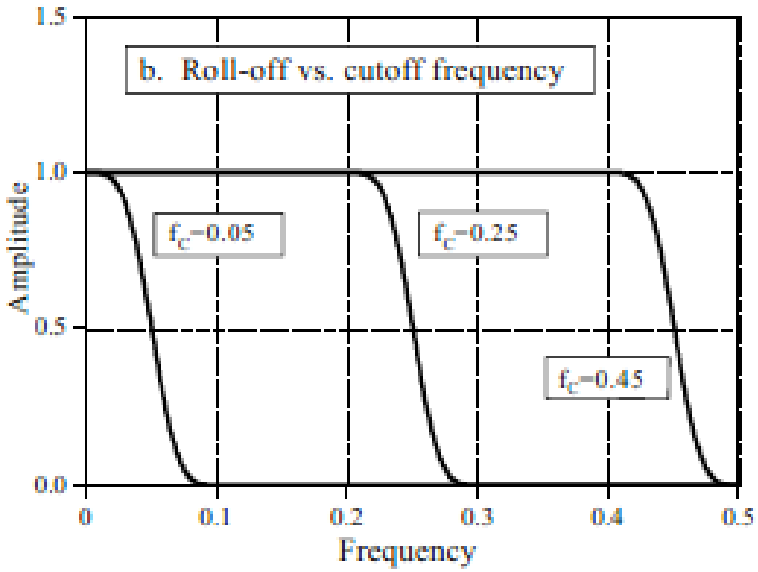
Ontwerp van de filter

Ontwerp windowed-sinc : 2 parameters van belang: fc (cutoff-frequentie) en M (lengte filter-kernel)

- Cutoff-frequentie: Uitgedrukt als een fractie van de samplerate en ligt tussen 0 en 0,5
- De waarde van M is bepalend voor de roll-off en wordt bepaald door M = 4/BW
- BW is de breedte in de overgang van doorlaatband naar stopband (van daar waar amplitude overgaat van praktisch 1 (99% van de curve) tot praktisch 0 (1% van de curve)

- Doorlaatband is bepaald door afstand tussen f=0 en f=0,5
- Fig (a) toont toont 3 LDF's met M gelijk aan 20, 40 en 200
- Fig (b) toont doorlaatbanden met fc = 0,05, 0,25 en 0,45





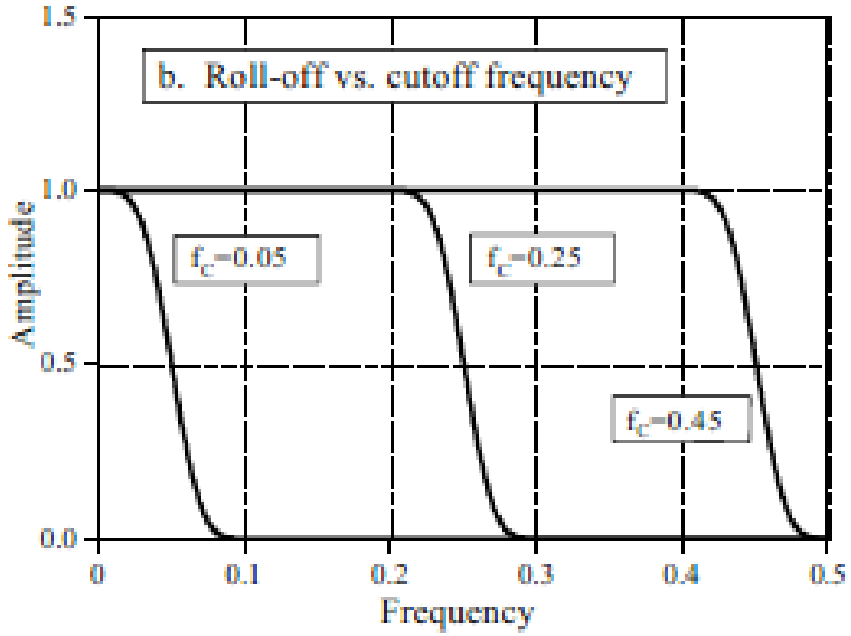
Ontwerp van de filter

Tijd nodig om convolutie uit te rekenen is evenredig met de lengte van de signalen => afweging nodig tussen de lengte M (filter-kernel) en BW (scherpte filter)

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

- ***Afsnijfrequentie van de doorlaatband wordt meestal bepaald op het halve amplitudepunt in plaats van op -3dB (0,707 amplitude)***
  - ***Waarom?*** Omdat de window-sinc frequentieresponse symmetrisch ligt tussen de doorlaatband en stopband
    - Vb. Hamming-window heeft een doorlaatrimpel van 0,2% en een identieke stopbandrimpel van 0,2%
  - Ander type filters vertonen deze symmetrie niet => geen voordeel mee. De symmetrie maakt windowed-sinc ideaal voor spectrale inversie (zie later)
  - Als fc en M geselecteerd zijn => filter-kernel kan berekend worden via volgende vergelijking:

$$h[i] = K \frac{\sin(2\pi f_c (i - M/2))}{i - M/2} \left[0.42 - 0.5\cos\left(\frac{2\pi i}{M}\right) + 0.08\cos\left(\frac{4\pi i}{M}\right) \right]$$



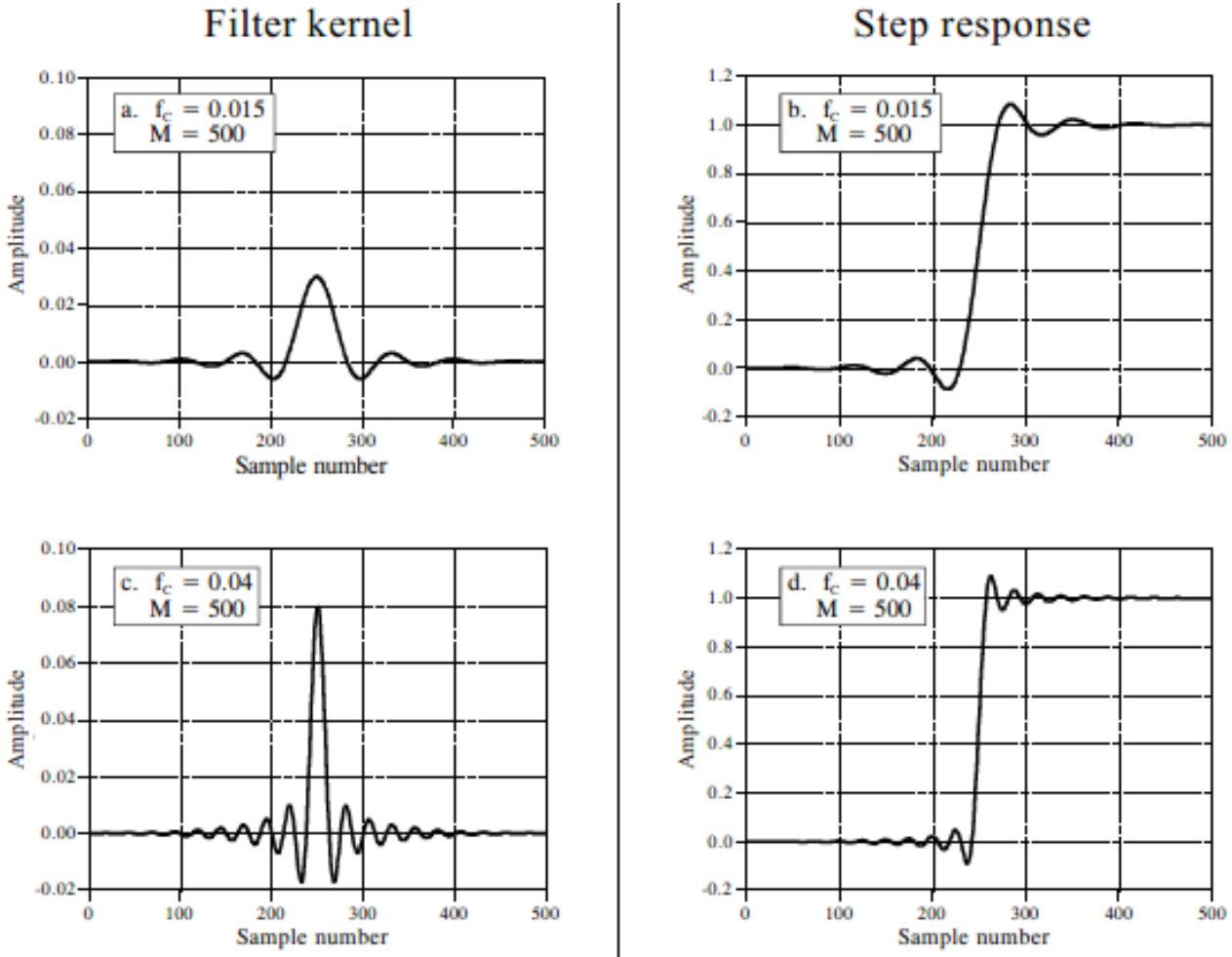
- Vgl bestaat uit sinc-functie, M/2-shift en Blackman window
- Om de filter de eenheidsamplitude te geven moet K zodanig gekozen worden dat de som van alle samples gelijk is aan 0 => Praktijk : negeer K tijdens de berekeningen en normaliseer alle samples waar nodig

Ontwerp van de filter

$$h[i] = K\,\frac{\sin(2\pi f_c\,(i - M/2))}{i - M/2}\left[0.42 - 0.5\cos\left(\frac{2\pi i}{M}\right) + 0.08\cos\left(\frac{4\pi i}{M}\right)\right]$$

- ***Stel M = 100*** (moet een even getal zijn)
  - Eerste punt in de filter-kernel is in de array locatie 0 terwijl het laatste punt zich bevind in de array op positie 100 => volledig signaal is 101 punten lang.
  - Het ***centrum van de symmetrie is op punt 50*** (dit is M/2)
  - De 50 punten links van punt 50 zijn symmetrisch tot de 50 punten rechts van puntpositie 50 (M/2)
  - ***Punt 0 heeft dezelfde waarde als punt 100; punt 49 dezelfde waarde als punt 51, enz…***
  - Als je een bepaald aantal samples nodig hebt in de filter om FFT te gebruiken, gewoon nullen aan het ene eind of het andere
  - Voorbeeld: ***M=100 => 101 punten; dichtstbijzijnde macht van 2 is 128 => 27 nullen toevoegen => filterkernel is 128 punten lang***

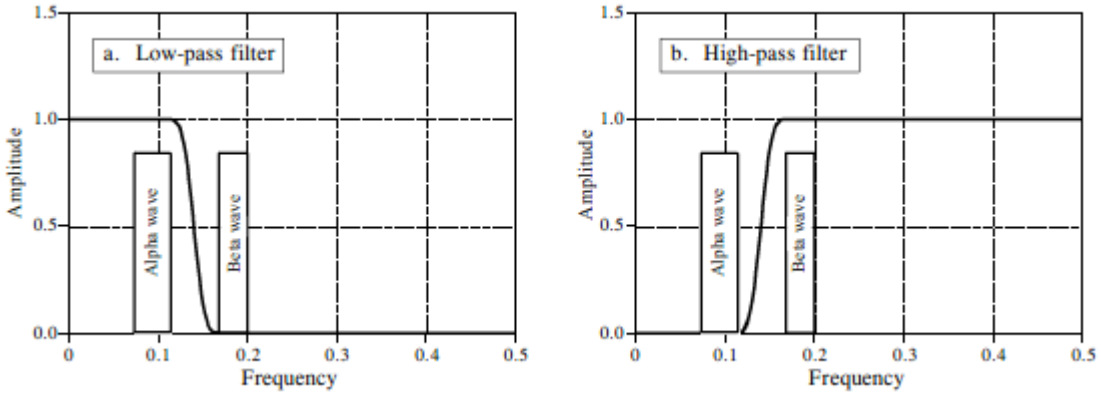
- Opmerking: goede respons in het frequentiebereik maar slecht in het tijdsdomein.



- Voorbeelden van windowed-sinc kernels en hun overeenkomstige stapresponsies
- Opmerking: ook de kleine samples links en rechts blijven belangrijk voor de performantie van de filter
- Floating point representatie is belangrijk om de kleine waarden te behouden

Voorbeelden van Windowed-Sinc filters

- 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 50 seconden 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



Voorbeelden van Windowed-Sinc filters

Hamming window: $w[i] = 0.54 - 0.46\cos(2\pi i/M)$

- 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.

```
100 'LOW-PASS WINDOWED-SINC FILTER
110 'This program filters 5000 samples with a 101 point windowed-sinc filter,
120 'resulting in 4900 samples of filtered data.
130 '
140 DIM X[4999]                    'X[ ] holds the input signal
150 DIM Y[4999]                    'Y[ ] holds the output signal
160 DIM H[100]                     'H[ ] holds the filter kernel
170 '
180 PI = 3.14159265
190 FC = .14                       'Set the cutoff frequency (between 0 and 0.5)
200 M% = 100                       'Set filter length (101 points)
210 '
220 GOSUB XXXX                     'Mythical subroutine to load X[ ]
230 '
240 '                              'Calculate the low-pass filter kernel via Eq. 16-4
250 FOR I% = 0 TO 100
260    IF (I%-M%/2) = 0 THEN H[I%] = 2*PI*FC
270    IF (I%-M%/2) <> 0  THEN H[I%] = SIN(2*PI*FC * (I%-M%/2)) / (I%-M%/2)
280    H[I%] = H[I%] * (0.54 - 0.46*COS(2*PI*I%/M%) )
290 NEXT I%
300 '
310 SUM = 0                        'Normalize the low-pass filter kernel for
320 FOR I% = 0 TO 100              'unity gain at DC
330    SUM = SUM + H[I%]
340 NEXT I%
350 '
360 FOR I% = 0 TO 100
370    H[I%] = H[I%] / SUM
380 NEXT I%
390 '
400 FOR J% = 100 TO 4999           'Convolve the input signal & filter kernel
410    Y[J%] = 0
420    FOR I% = 0 TO 100
430       Y[J%] = Y[J%] + X[J%-I%] * H[I%]
440    NEXT I%
450 NEXT J%
460 '
470 END
```

TABLE 16-1

Voorbeelden van Windowed-Sinc filters

Hamming window: $w[i] = 0.54 - 0.46\cos(2\pi i/M)$

ARTESIS PLANTIJN HOGESCHOOL ANTWERPEN

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

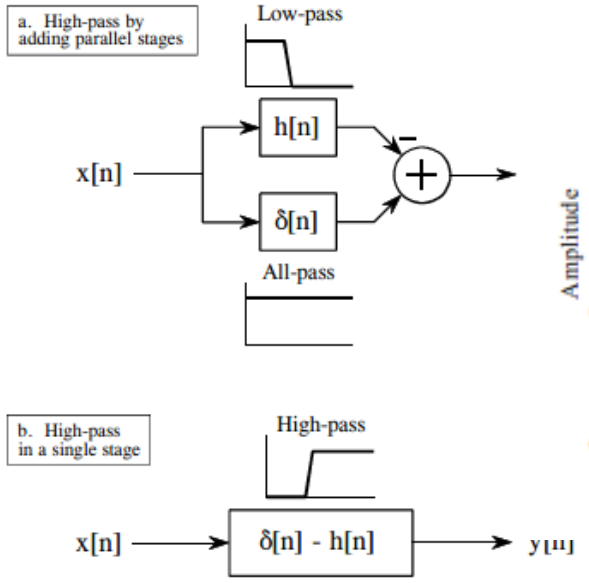
Keuze voor Hammingwindow voor dit vb.

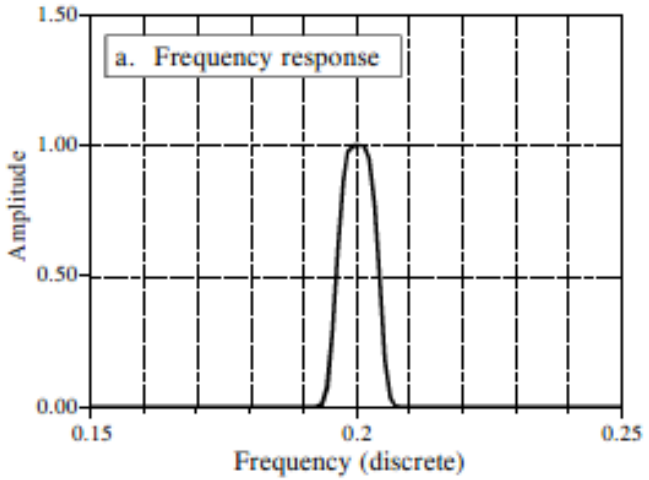
```
// LD-window synced filter
// voorbeeld 5000 samples filteren met een 101 punten windowed sinc filter
// resultaat levert 4900 gefilterde data op

int input_data [4999];  //ingangssamples opslaan
float output_data [4999]; // uitgangssamples
float kernel_H [100]; // kernel van de filter

PI = 3,14159265;
FC = 0,14 // afsnijfrequentie
M = 100  // Instellen filterlengte op 101 punten

Aanroepen subroutines om de input sampledata in te lezen in input_data

// berekenen LD filterkernel via vgl hamming window

for i=0 : 100
  if (i-M/2) == 0 then
    H(i) = 2 * PI * FC;
    end
  if (i-M/2) <> 0 then
    H(i) = sin(2*PI*FC*(i-M/2))/(i-M/2);
    end
  H(i) = H(i) * (0,54 - 0,46 * (cos(2 * PI * FC * (i-M/2))/(i-M/2);
end
```

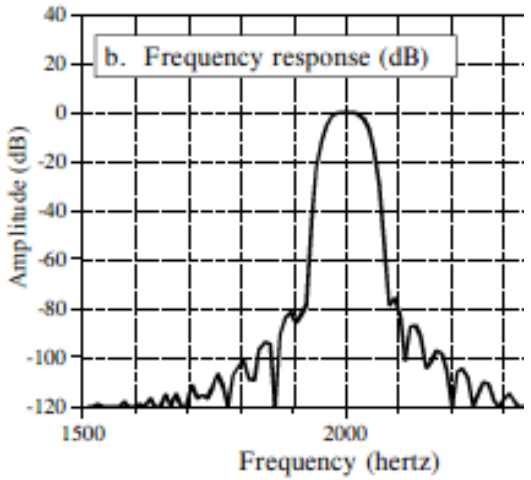
```
// berekenen LD filterkernel via vgl hamming window

for i=0 : 100
  if (i-M/2) == 0 then
    H(i) = 2 * PI * FC;
    end
  if (i-M/2) <> 0 then
    H(i) = sin(2*PI*FC*(i-M/2))/(i-M/2);
    end
  H(i) = H(i) * (0,54 - 0,46 * (cos(2 * PI * FC * (i-M/2))/(i-M/2);
end

//normaliseren LD-filter for eenheid^
SOM = 0;
for i=0 : 100
end

for i=0 : 100
  SOM = SOM + H(i);
  H(i) = H(i)/SOM;
end

// convolutie tussen het ingangssignaal en de kernel van de filter
for j=100 : 4999
  output_data(j) == 0;
  for i=0 : 100
    output_data(j) = output_data(j)+input_data(j-i) * H(i);
  end
end
```

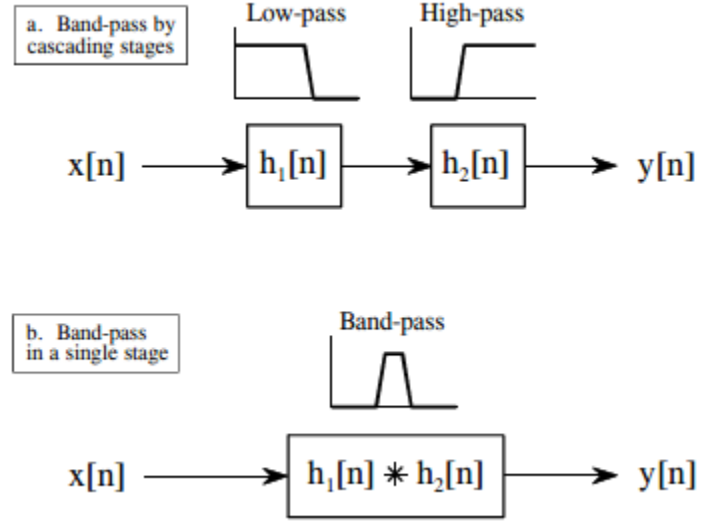
Ontwerp van banddoorlaatfilter

- Ontwerp filter:
  - Doel: isoleren signaaltoon in een audiosignaal, bv een duo-toon van een telefoontoets.
  - Stel dat het ***signaal is gedigitaliseerd met fs = 10 kHz*** en dat we een ***80 Hz band willen isoleren*** van frequenties die rond de 2 kHz liggen. (***band tussen 1960 Hz en 2040 Hz => genormaliseerde frequenties 0,196 en 0,204)***.
  - Om een transition BW te bekomen van 50 Hz => 50 Hz normaliseren =>50 Hz/10000Hz = 0,005 => M = 4/0,005 = 800 punten => filterkernel = 801 punten
  - Opbouw filter: LDF fc = 0,196; LDF fc = 0,204 daarna spectraal omgekeerd HDF;
  - 2 filters optellen => bandsperfilter => terug spectrale inversie => banddoorlaatfilter





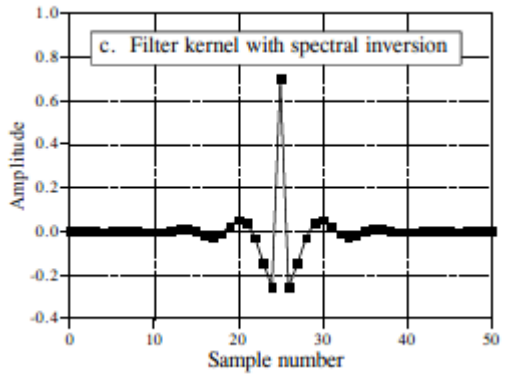


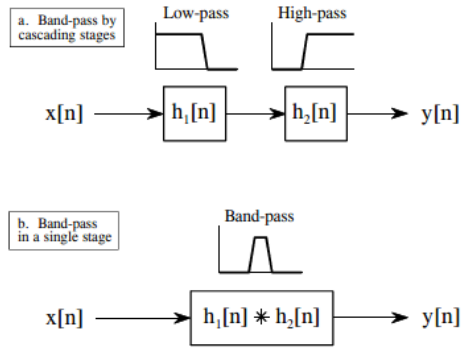


Ontwerp van banddoorlaatfilter

$$h[i] = K \frac{\sin(2\pi f_c (i - M/2))}{i - M/2} \left[0.42 - 0.5\cos\left(\frac{2\pi i}{M}\right) + 0.08\cos\left(\frac{4\pi i}{M}\right) \right]$$

```
100 'BAND-PASS WINDOWED-SINC FILTER
110 'This program calculates an 801 point band-pass filter kernel
120 '
130 DIM A[800]                      'A[ ] workspace for the lower cutoff
140 DIM B[800]                      'B[ ] workspace for the upper cutoff
150 DIM H[800]                      'H[ ] holds the final filter kernel
160 '
170 PI = 3.1415926
180 M% = 800                        'Set filter kernel length (801 points)
190 '
200 '                               'Calculate the first low-pass filter kernel via Eq. 16-4,
210 FC = 0.196                      'with a cutoff frequency of 0.196, store in A[ ]
220 FOR I% = 0 TO 800
230   IF (I%-M%/2) = 0 THEN A[I%] = 2*PI*FC
240   IF (I%-M%/2) <> 0  THEN A[I%] = SIN(2*PI*FC * (I%-M%/2)) / (I%-M%/2)
250   A[I%] = A[I%] * (0.42 - 0.5*COS(2*PI*I%/M%) + 0.08*COS(4*PI*I%/M%))
260 NEXT I%
270 '
280 SUM = 0                         'Normalize the first low-pass filter kernel for
290 FOR I% = 0 TO 800               'unity gain at DC
300   SUM = SUM + A[I%]
310 NEXT I%
320 '
330 FOR I% = 0 TO 800
340   A[I%] = A[I%] / SUM
350 NEXT I%
360 '                               'Calculate the second low-pass filter kernel via Eq. 16-4,
370 FC = 0.204                      'with a cutoff frequency of 0.204, store in B[ ]
380 FOR I% = 0 TO 800
390   IF (I%-M%/2) = 0 THEN B[I%] = 2*PI*FC
400   IF (I%-M%/2) <> 0  THEN B[I%] = SIN(2*PI*FC * (I%-M%/2)) / (I%-M%/2)
410   B[I%] = B[I%] * (0.42 - 0.5*COS(2*PI*I%/M%) + 0.08*COS(4*PI*I%/M%))
420 NEXT I%
430 '
440 SUM = 0                         'Normalize the second low-pass filter kernel for
450 FOR I% = 0 TO 800               'unity gain at DC
460   SUM = SUM + B[I%]
470 NEXT I%
480 '
490 FOR I% = 0 TO 800
500   B[I%] = B[I%] / SUM
510 NEXT I%
520 '
530 FOR I% = 0 TO 800               'Change the low-pass filter kernel in B[ ] into a high-pass
540   B[I%] = - B[I%]               'filter kernel using spectral inversion (as in Fig. 14-5)
550 NEXT I%
560 B[400] =  B[400] + 1
570 '
580 '
590 FOR I% = 0 TO 800               'Add the low-pass filter kernel in A[ ], to the high-pass
600   H[I%] = A[I%] + B[I%]         'filter kernel in B[ ], to form a band-reject filter kernel
610 NEXT I%                         'stored in H[ ] (as in Fig. 14-8)
620 '
630 FOR I% = 0 TO 800               'Change the band-reject filter kernel into a band-pass
640   H[I%] = -H[I%]                'filter kernel by using spectral inversion
650 NEXT I%
660 H[400] =  H[400] + 1
670 '                               'The band-pass filter kernel now resides in H[ ]
680 END
```





Ontwerp een windowfilter waarbij enkel de drums hoorbaar zijn van volgende mp3-file :

Streetdemonstration.mp3

Herhaal dit voor enkel de trompetten en enkel de drums.

Omzetten arduino voor lezen wav-files : https://www.instructables.com/id/Playing-Wave-file-using-arduino/