

Introductie Moving Average Filter (voortschrijdend gemiddelde filter)

Meest voorkomende filter in digitale signaalbewerking vermits het de gemakkelijkste digitale filter is om op te bouwen;

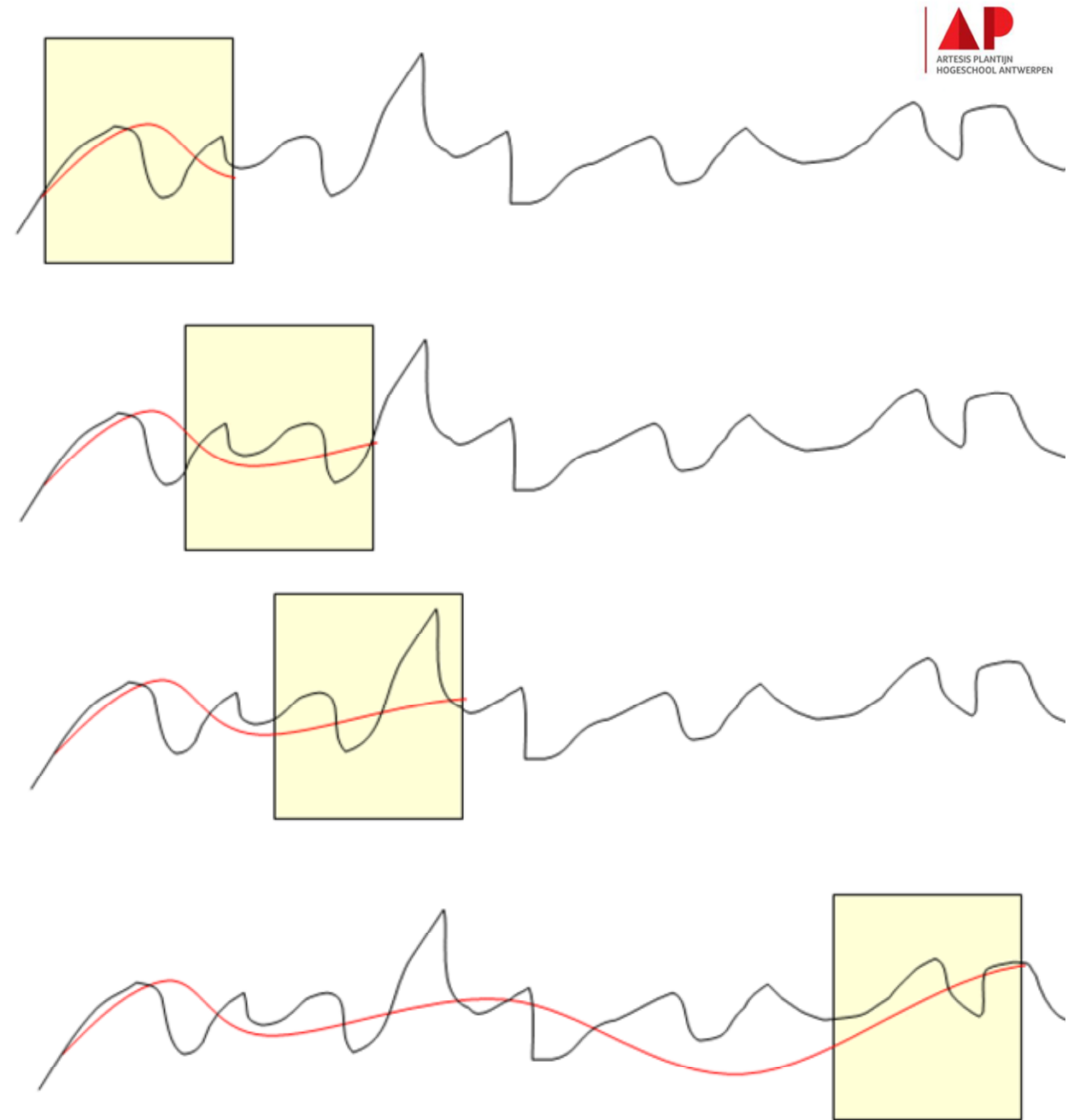
De filter is optimaal voor het verwijderen van random ruis terwijl een scherpe stap-response behouden blijft.

- Dit maakt het tot een **geschikte filter voor tijdsdomein gecodeerde signalen**.
- Moving Average Filter is de **slechtste filter voor frequentiedomein gecodeerde signalen, met weinig vermogen om een bepaalde frequentieband af te scheiden van anderen**.

Aanverwante filters van Moving Average filters zijn onder andere de Gauss, Blackman, en multiplepass filter. Deze hebben (iets) betere prestaties in het frequentiedomein maar met nadeel dat ze meer rekentijd vergen om het resultaat te bekomen dan Moving Average.

Gemiddelde gebruiken als filter

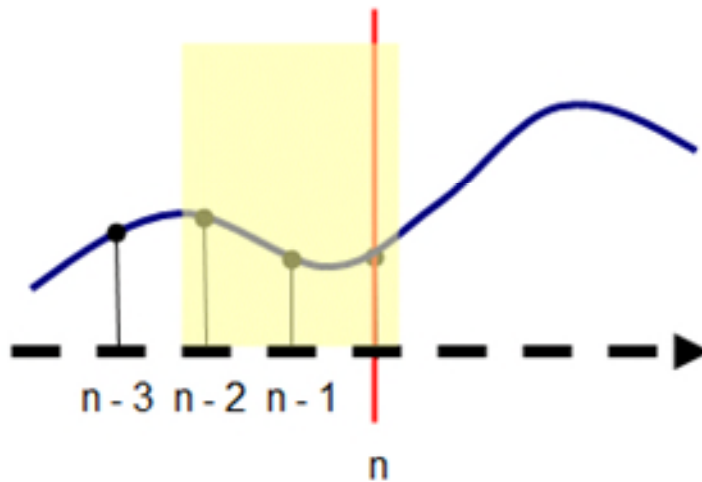
- “Moving processing” over data is het belangrijkste proces voor een digitale filter.
- In dit voorbeeld wordt een gemiddelde waarde bepaald aan de hand van een bewegend venster.
- Van de data binnen het venster wordt de gemiddelde waarde bepaald en deze wordt weergegeven als output voor bv. de centrale positie binnen dit venster.
- Programmatorisch gezien kan het bewegen van het venster over de data verwezenlijkt worden met een for-loop. Na iedere berekening wordt het venster één positie verder verplaatst. ($i = i+1$)
- Bij het bepalen van het gemiddelde (average) worden enkel de berekeningen in het verleden en de huidige berekening in rekening gebracht.
- De vraag is hoeveel waarden terug in het verleden gebruik je en kan je ook gebruik maken van waarden die verder op gelegen zijn (waarden die in de toekomst bepaald worden)?



Moving Average filter

Gemiddelde gebruiken als filter

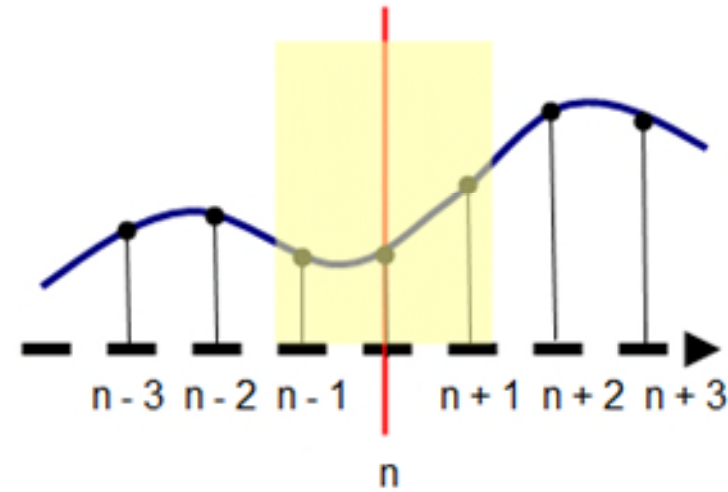
- Voor de berekening van de huidige waarde, deze uit het verleden of in de toekomst kunnen n samplewaarden gebruikt worden.
- De toekomstige waarden kunnen enkel gebruikt worden in OFF-line berekeningen. Enkel dan is er beschikking over de toekomstige waarden.
- Bij ON-line berekeningen kunnen enkel de waarden uit het verleden en de huidige waarde in de berekeningen worden opgenomen.



$$\frac{x_{n-2} + x_{n-1} + x_n}{3}$$

3

ON-line



$$\frac{x_{n-1} + x_n + x_{n+1}}{3}$$

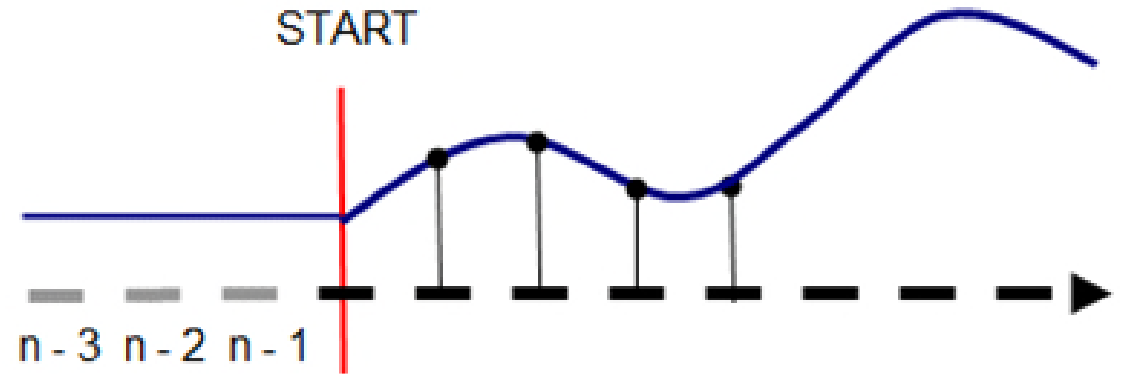
3

OFF-line

Moving Average filter

Start/Stop probleem.

- Voor de eerste berekening zijn er geen waarden in het verleden. Nemen we hier de voorgaande waarden gelijk aan 0, dan wordt er reeds een grote fout gecreëerd tijdens het startmoment.
 - Hiervoor zijn verschillende oplossingen. De meest eenvoudige is de onbekende waarden dezelfde te nemen als de eerste waarde die gekend is voor deze onbekende waarden. (zie naaststaande figuur)
 - Wanneer vervolgens de volgende waarden genomen worden heb je een probleem met de laatste waarde
-
- Hoeveel waarden uit het verleden en/of de toekomst neem je best mee in de berekeningen?
 - Antwoord : hoe meer waarden, hoe beter MAAR hou rekening met hoe meer waarden je neemt, hoe meer computervermogen je nodig hebt om de berekeningen uit te laten voeren. (Offline maakt dit niet veel uit maar Online wel omdat dan de berekeningen in real-time moeten gebeuren)



Moving Average Filter

- De moving average filter is als volgt opgebouwd :

$$y = \frac{1,0 \cdot x_{n-2} + 1,0x_{n-1} + 1,0 x}{3}$$

- Alle in beschouwing genomen waarden in het verleden hebben de factor 1,0 bij deze filter.

- Andere manier van opbouw:

$$y = 0,33 \cdot x_{n-2} + 0,33x_{n-1} + 0,33x$$

- Je kan de filter ook opbouwen met factoren (gewichten)

- Voorbeeld :

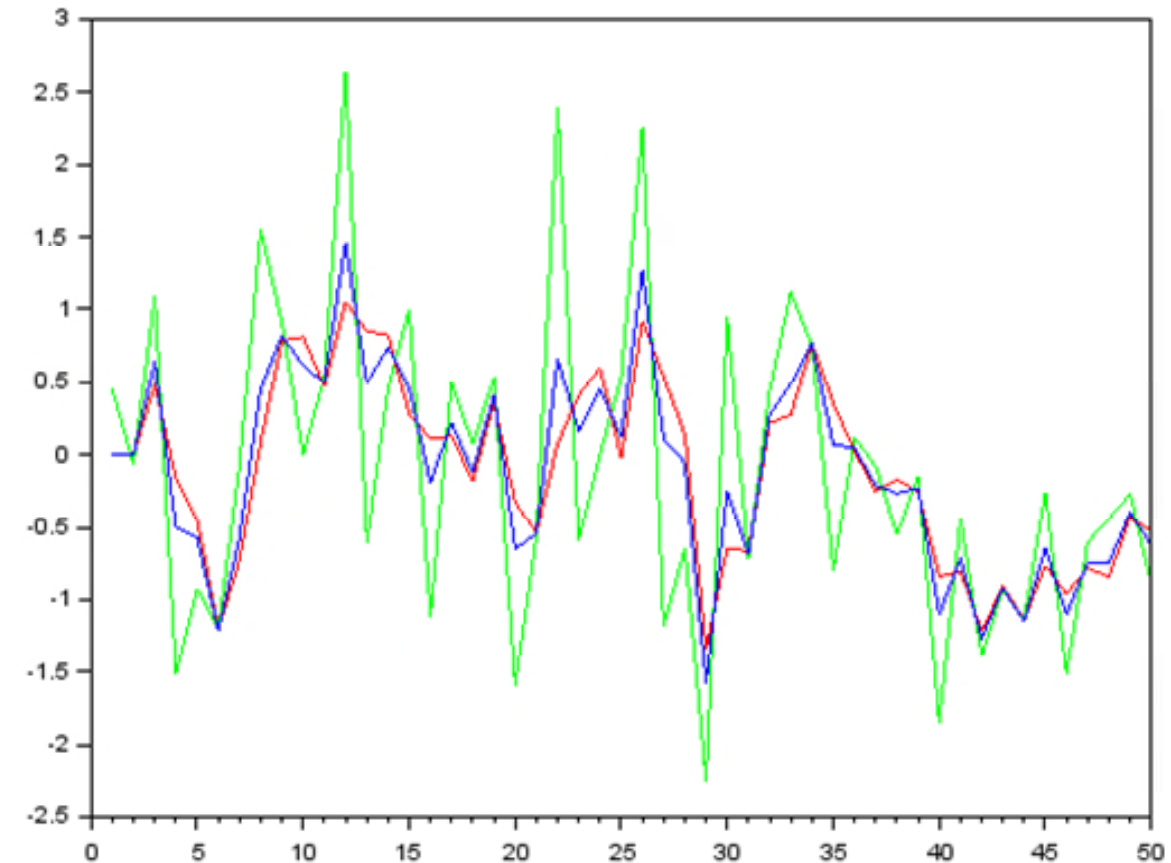
- $$y = \frac{0,75 \cdot x_{n-2} + 0,75x_{n-1} + 1,5 x}{3}$$

of:

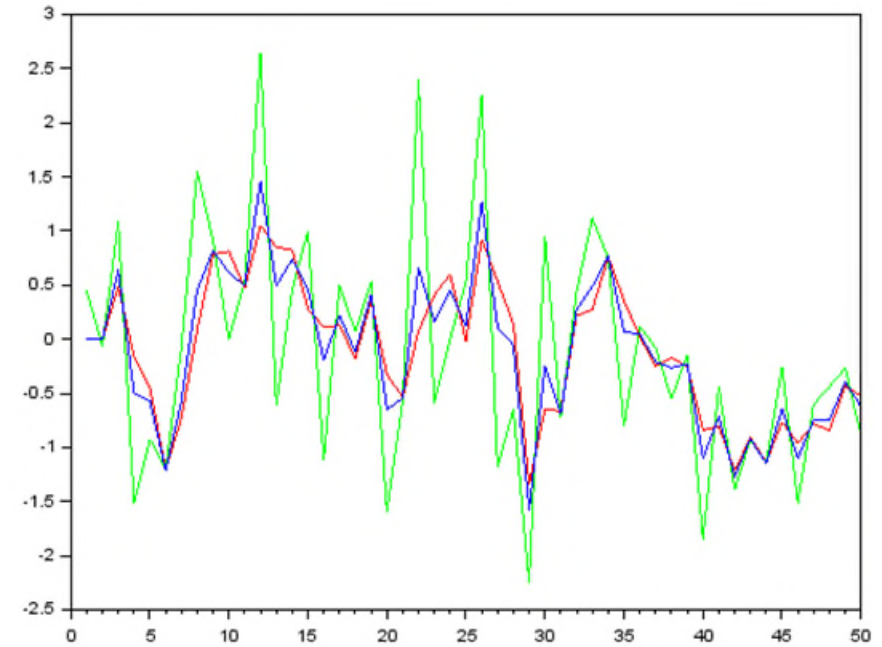
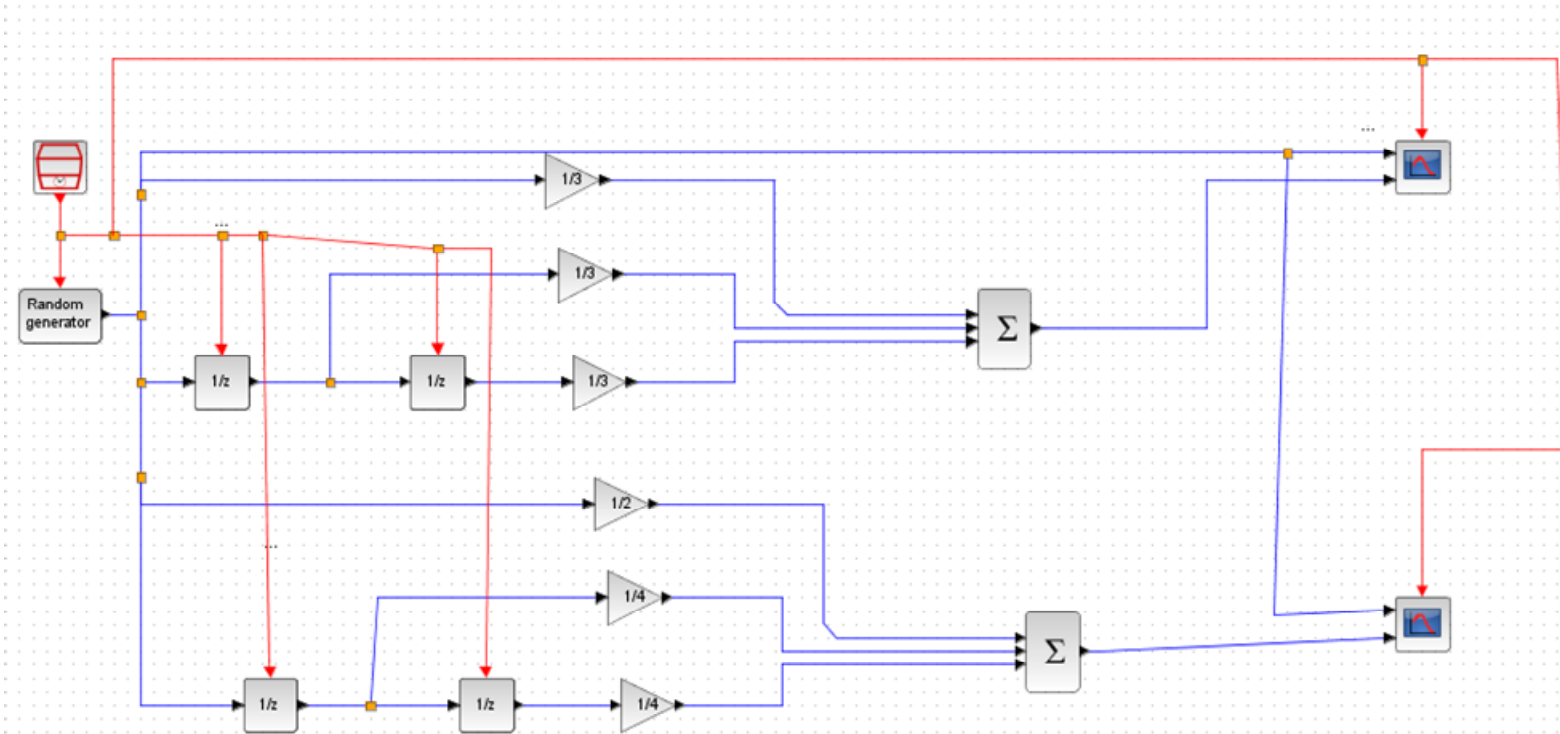
- $$y = 0,25 \cdot x_{n-2} + 0,25x_{n-1} + 0,50x$$

Voorbeeld Moving Average Filter

```
1 //Moving average filter op random 50 waarden
2 //opbouw met factoren 1.0
3 signal = rand(50,1,'normal')
4 filter_output = zeros(50,1)
5 //origineel signaal in groen weergeven
6 plot(signal,'g')
7 //filter werking met factoren 0.33xn-2, 0.33xn-1 0.33·n
8 for i=3:length(signal)
9     filter_output(i)=0.33*signal(i-2)+...
10     0.33*signal(i-1)+0.33*signal(i)
11 end
12 //filteroutput weergegeven in rood
13 plot(filter_output,'r')
14 //filter werking met factoren 0.25xn-2, 0.25xn-1 0.50·n
15 filter_output2 = zeros(50,1)
16 for i=3:length(signal)
17     filter_output2(i)=0.25*signal(i-2)+...
18     0.25*signal(i-1)+0.50*signal(i)
19 end
20 //filteroutput2 (met verschillende factoren of gewichten)
21 //weergegeven in blauw
22 plot(filter_output2,'b')
```

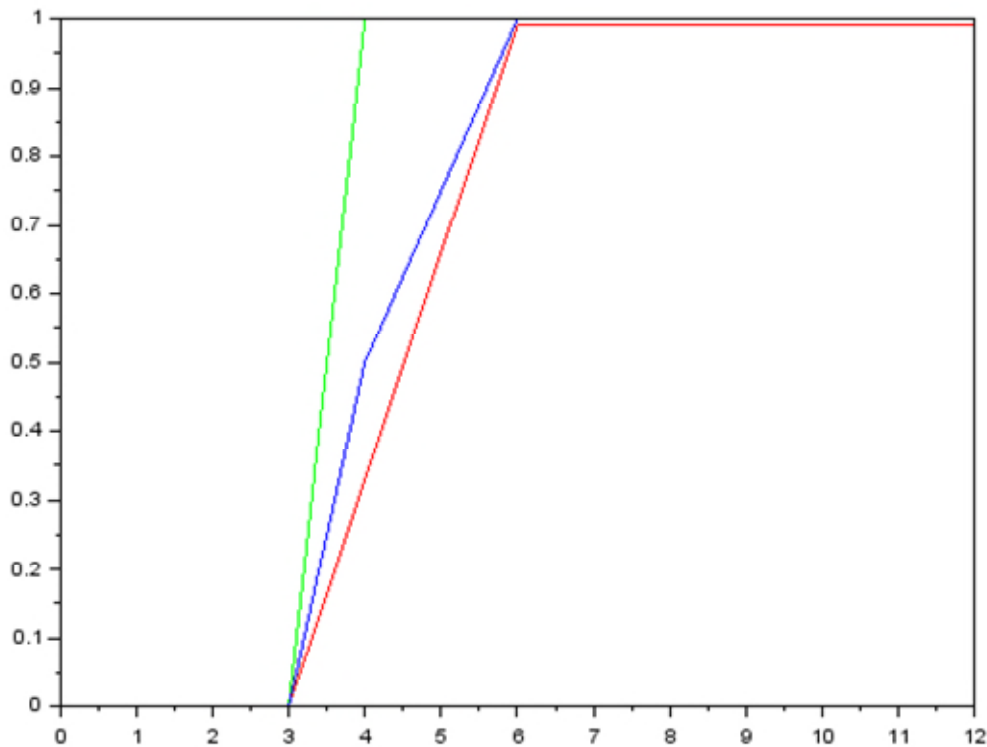


Voorbeeld Moving Average Filter in XCOS



Stapresponse

- Met de stapresponse kan gemakkelijk het verschil tussen de transfertfuncties worden aangetoond.
- Stel terug beide filters van voorgaand programma waaraan nu een stapresponse wordt aangelegd



```
1 //Moving average filter met stapresponse
2 //opbouw met factoren 1.0
3 clc
4 stap = [0 0 0 1 1 1 1 1 1 1 1 1]
5 filter_output = zeros(length(stap),1)
6 //origineel signaal in groen weergeven
7 plot(stap,'g')
8 //filter werking met factoren 0.33xn-2, 0.33xn-1 0.33·n
9 for i=3:length(stap)
10     filter_output(i)=0.33*stap(i-2)+...
11     0.33*stap(i-1)+0.33*stap(i)
12 end
13 //filteroutput weergegeven in rood
14 plot(filter_output,'r')
15 //filter werking met factoren 0.25xn-2, 0.25xn-1 0.50·n
16 filter_output2 = zeros(length(stap),1)
17 for i=3:length(stap)
18     filter_output2(i)=0.25*stap(i-2)+...
19     0.25*stap(i-1)+0.50*stap(i)
20 end
21 //filteroutput2 (met verschillende factoren of gewichten)
22 //weergegeven in blauw
23 plot(filter_output2,'b')
```

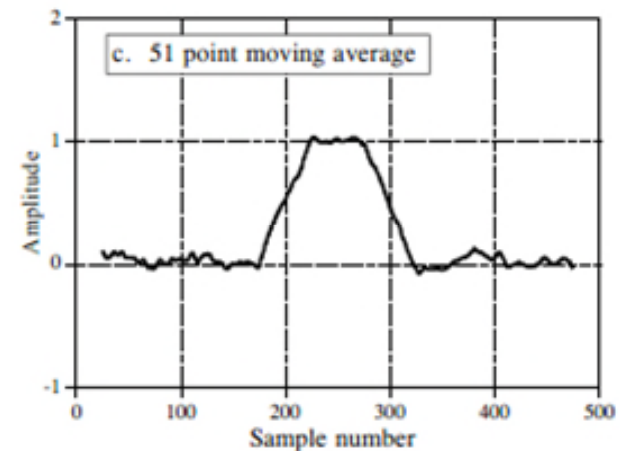
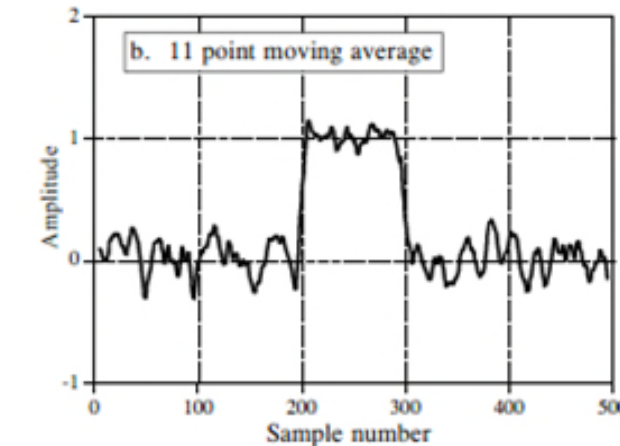
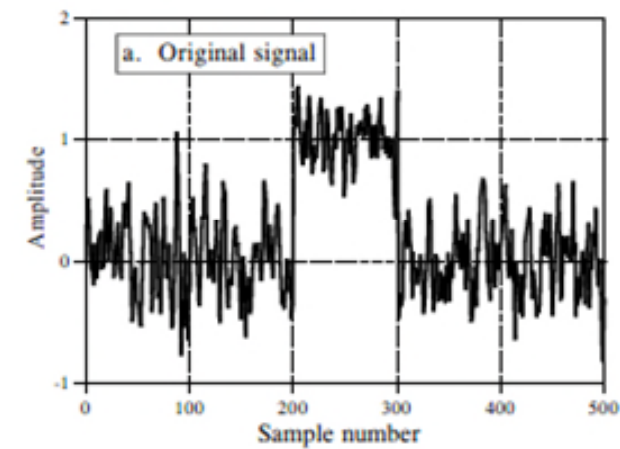

Ruisonderdrukking versie stapresponse

Moving Average Filter is vooral geschikt voor onderdrukking van witte ruis terwijl de scherpste stapresponsie behouden blijft.

- (a) Origineel pulssignaal met veel ruis op
- (b) Invloed van de filter vermindert de amplitude van de ruis (goed)
- (c) maar verlaagt de scherpte van de randen (slecht)

De hoeveelheid ruisonderdrukking is gelijk aan de vierkantswortel van het aantal punten in het gemiddelde.

Voorbeeld: 100 punt Moving Average Filter vermindert de ruis met een factor 10



Ruisonderdrukking versie stapresponse

Waarom Moving Average is een goede oplossing?

Stel dat we een filter willen ontwerpen, met naast ruisonderdrukking ook verscherping van de randscherpte (flanken)

Stel dat we de randscherpte versterken door te specificeren dat er elf punten zijn in het stijgen van de stapresponse .

Hoe kiezen we deze 11 punten zodat de ruis in het uitgangssignaal zo klein mogelijk wordt?

De ruis op het signaal dat we proberen te verminderen is willekeurig => geen zin om speciale gewichten op bepaalde plaatsen toe te passen. Beste ruisonderdrukking bekomen door alle samples gelijkwaardig te behandelen (dus moving average filter)

Frequentieresponse

- Praktisch worden filters getest in het frequentiedomein en bijvoorbeeld via een bodediagram. Hiervoor gebruiken we de functie **bode()** om een inzicht te verkrijgen in de digitale filter.
- De **bode()**-functie heeft een transferfunctie nodig in het gesampled gebied waarbij de individuele coëfficiënten met Z^{-1} , Z^{-2} , ... gekend zijn als transferparameters
- Eén samplevertraging is Z^{-1} en twee samplevertragingen wordt door Z^{-2} voorgesteld enz ...
- De twee beschouwde moving average filters hebben een werking als een LD-filter en hebben de volgende transferfunctie (de samplefrequentie is genormaliseerd naar 1)

$$y = 0,33 \cdot Z^{-2} + 0,33 \cdot Z^{-1} + 0,33 \cdot Z$$

$$y = 0,25 \cdot Z^{-2} + 0,25 \cdot Z^{-1} + 0,50 \cdot Z$$

Hoe praktisch te werk gaan?

- De parameters van de moving average filter worden als een rijvector opgeslagen (parameters). Per definitie is de eerste parameter gelijk aan Z ; de tweede Z^{-1} ; de derde Z^{-2} enz ...
- Deze rijvector is nog niet de transferfunctie die we binnen Scilab nodig hebben. We moeten Scilab nog duidelijk maken dat dit een transferfunctie is.
- Via de functie **POLY()** definiëren we een polynomial op basis van de vector parameter als volgt :

transferp = poly(parameter, 'z', 'coeff')

- Parameter : de opgeslagen parameters in een rijvector
- 'z' : soort van transferfunctie . (discreet voorgesteld door z en continu voorgesteld door s)
- 'coeff' : weergeven of het gaat om coëfficiënten (coeff) of wortels (roots)
- transferp is de naam van de polynomial

- Eens polynoom gevormd, staat deze in de representatie van z . Via de functie **HORNER()** wordt de polynomial omgevormd van z -representatie naar $1/z$ representatie waardoor de benodigde transferfunctie wordt bekomen. Aldus wordt **transferh** bekomen als functie in $1/z$ representatie.

`transferh = horner (transferp, (1/%z))`

- Nu moeten we enkel scilab nog duidelijk maken dat de functie die we beschrijven een discrete transfer functie is. Dit kunnen we doen met de parameter 'd' in de functie SYSLINK(). Het antwoord van de functie SYSLINK() is dat er een lijst verschijnt die volledig het lineair systeem beschrijft.

`transfersys = syslin ('d', transferh)`

- Het bodediagram kan weergegeven worden als volgt:

`bode (transfersys)`

- Voorbeeld programma voor bodeplots van de twee moving average filters met transferfuncties:

$$y = 0,25 \cdot Z^{-2} + 0,25 \cdot Z^{-1} + 0,5 \cdot Z$$

$$y = 0,33 \cdot Z^{-2} + 0,33 \cdot Z^{-1} + 0,33 \cdot Z$$

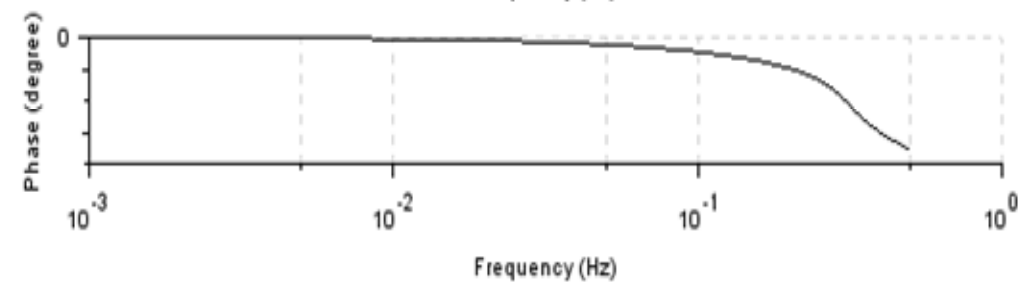
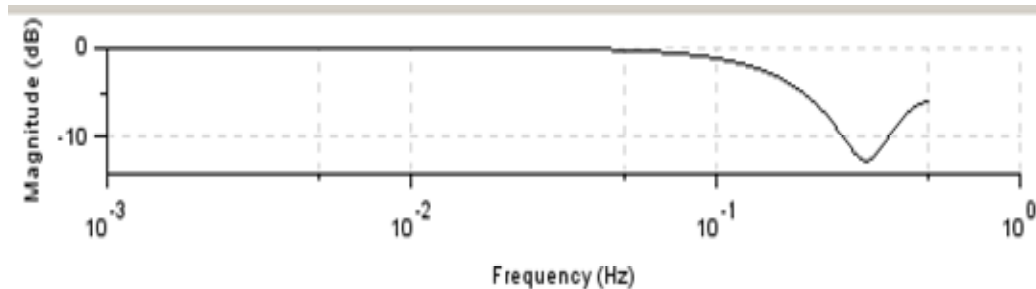
```
1 //bodediagram.van.moving.average.filter
2 //.filter.y=0.25z-2.+0.25.z-1.+0.5.z
3 //.opgave.parameters.van.filter
4 subplot(2,1,1)
5 parameter=[1/4 1/4 1/2]
6 //.bepalen.van.polynoom
7 transferp=poly(parameter,'z','coef')
8 //omvormen.transfer.poly.in.z.via.horner()naar.1/z
9 transferh=horner(transferp,(1/%z))
10 transfersys=syslin('d',transferh)
11 bode(transfersys)
12 subplot(2,1,2)
13 parameter=[1/3 1/3 1/3]
14 //.bepalen.van.polynoom
15 transferp=poly(parameter,'z','coef')
16 //omvormen.transfer.poly.in.z.via.horner()naar.1/z
17 transferh=horner(transferp,(1/%z))
18 //omzetten.naar.een.discrete.transferfunctie
19 transfersys=syslin('d',transferh)
20 //teken.van.bodediagram.van.de.filter
21 bode(transfersys)
```

```

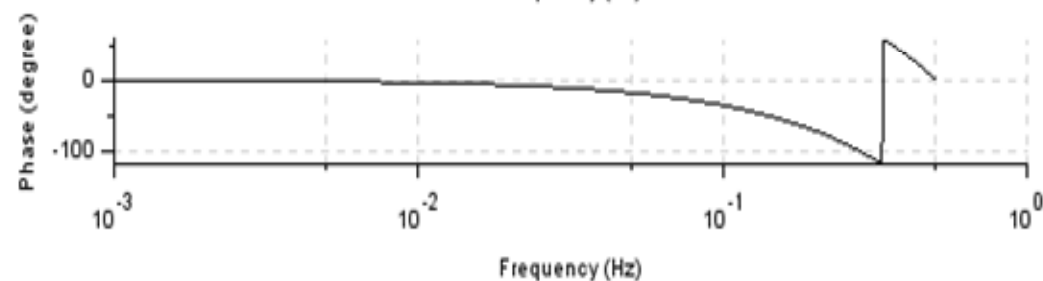
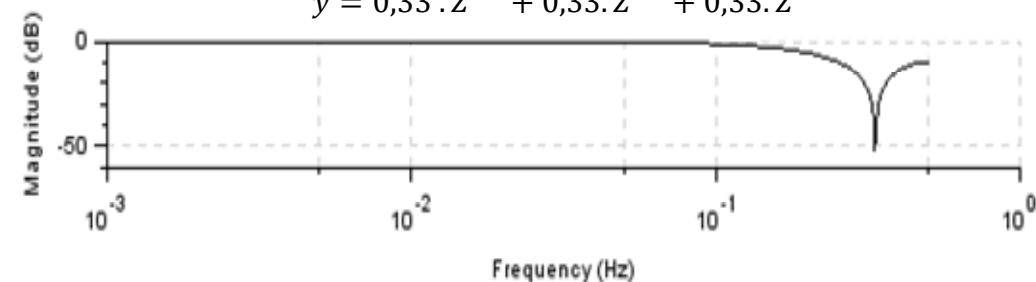
1 //bodediagram.van.moving.average.filter
2 //.filter.y=0.25z-2.+0.25.z-1.+0.5.z
3 //opgave.parameters.van.filter
4 subplot(2,1,1)
5 parameter=[1/4 1/4 1/2]
6 //.bepalen.van.polynoom
7 transferp=poly(parameter,'z','coef')
8 //omvormen.transfer.poly.in.z.via.horner()naar.1/z
9 transferh=horner(transferp,(1/%z))
10 transfersys=syslin('d',transferh)
11 bode(transfersys)
12 subplot(2,1,2)
13 parameter=[1/3 1/3 1/3]
14 //.bepalen.van.polynoom
15 transferp=poly(parameter,'z','coef')
16 //omvormen.transfer.poly.in.z.via.horner()naar.1/z
17 transferh=horner(transferp,(1/%z))
18 //omzetten.naar.een.discrete.transferfunctie
19 transfersys=syslin('d',transferh)
20 //teken.van.bodediagram.van.de.filter
21 bode(transfersys)

```

$$y = 0,25.Z^{-2} + 0,25.Z^{-1} + 0,5Z$$

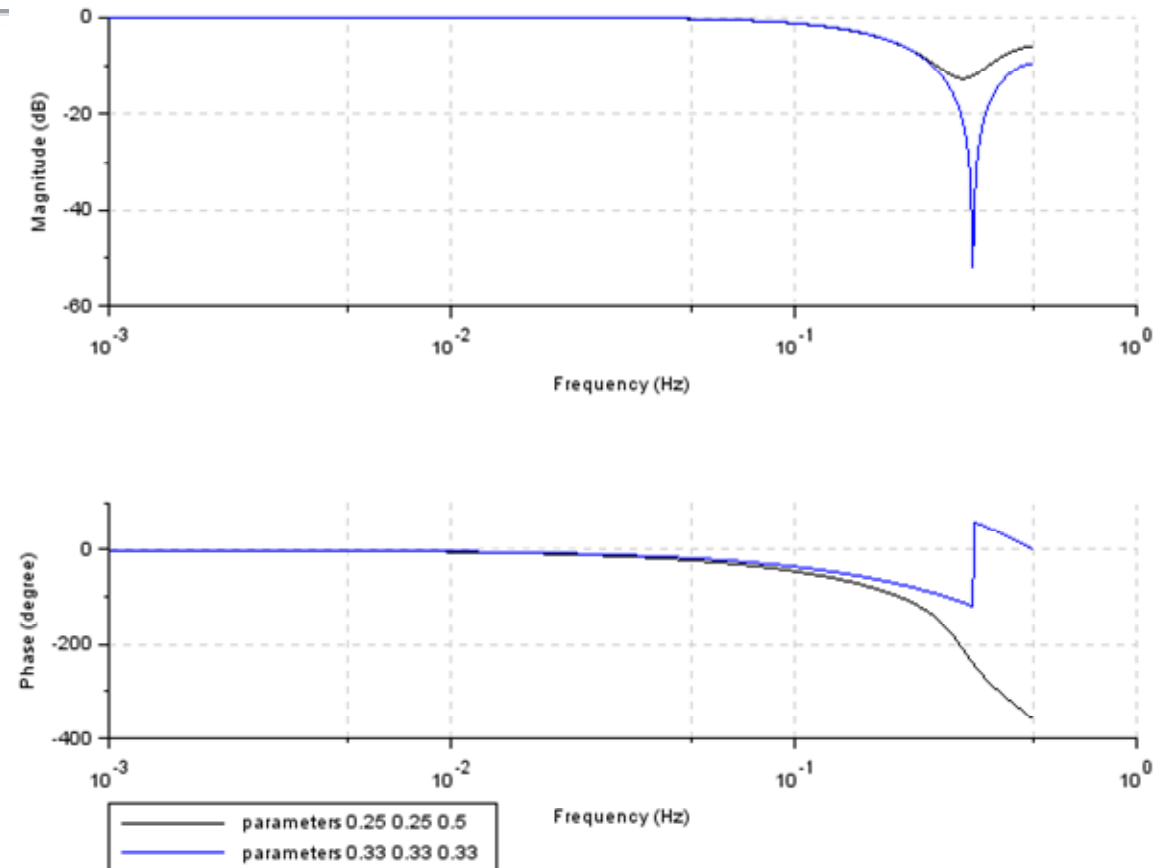


$$y = 0,33.Z^{-2} + 0,33.Z^{-1} + 0,33.Z$$



- Weergeven beide bodeplots over elkaar

```
1 //bodediagram van moving average filter
2 //filter y=0.25z^-2 + 0.25z^-1 + 0.5z^0
3 //opgave parameters van filter
4 parameter = [1/4 1/4 1/2]
5 //bepalen van polynoom
6 transferp = poly(parameter, 'z', 'coef')
7 //omvormen transfer poly in z via horner() naar 1/z
8 transferh = horner (transferp, (1/%z))
9 transfersys1 = syslin('d', transferh)
10 parameter = [1/3 1/3 1/3]
11 //bepalen van polynoom
12 transferp = poly(parameter, 'z', 'coef')
13 //omvormen transfer poly in z via horner() naar 1/z
14 transferh = horner (transferp, (1/%z))
15 //omzetten naar een discrete transferfunctie
16 transfersys2 = syslin('d', transferh)
17 //teken van bodediagrammen van beide filters en verduidelijking van de lijnen
18 bode([transfersys1; transfersys2], ['parameters 0.25 0.25 0.5'; 'parameters 0.33 0.33 0.33'])
```



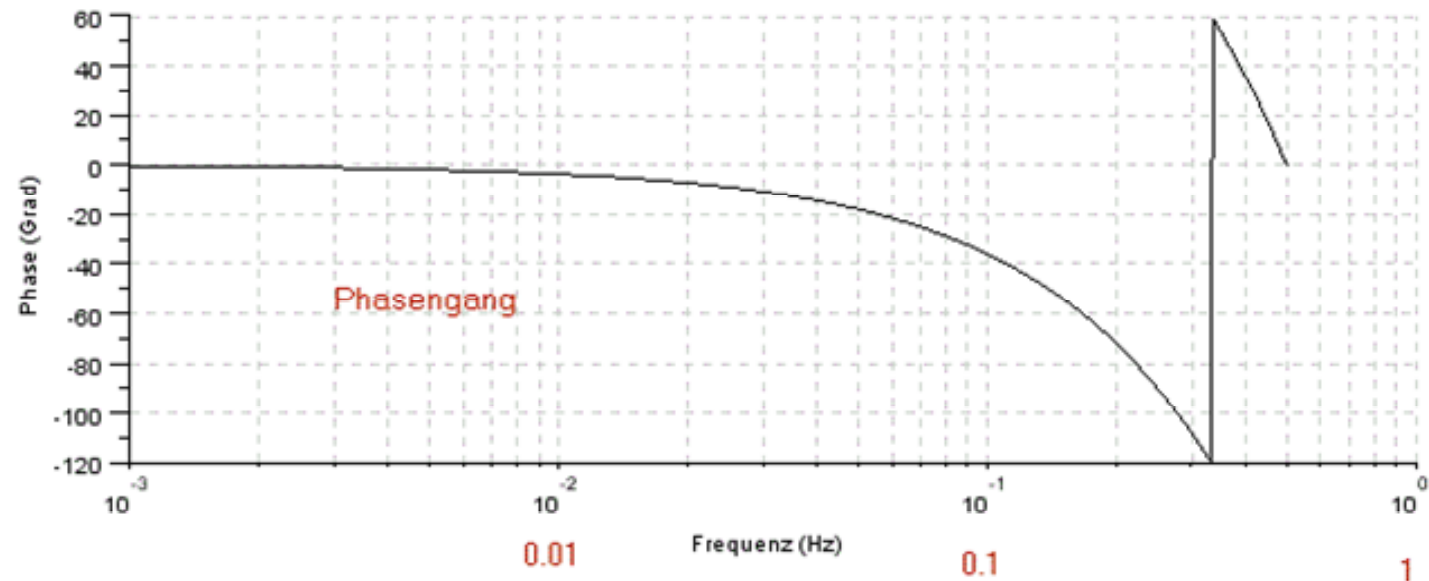
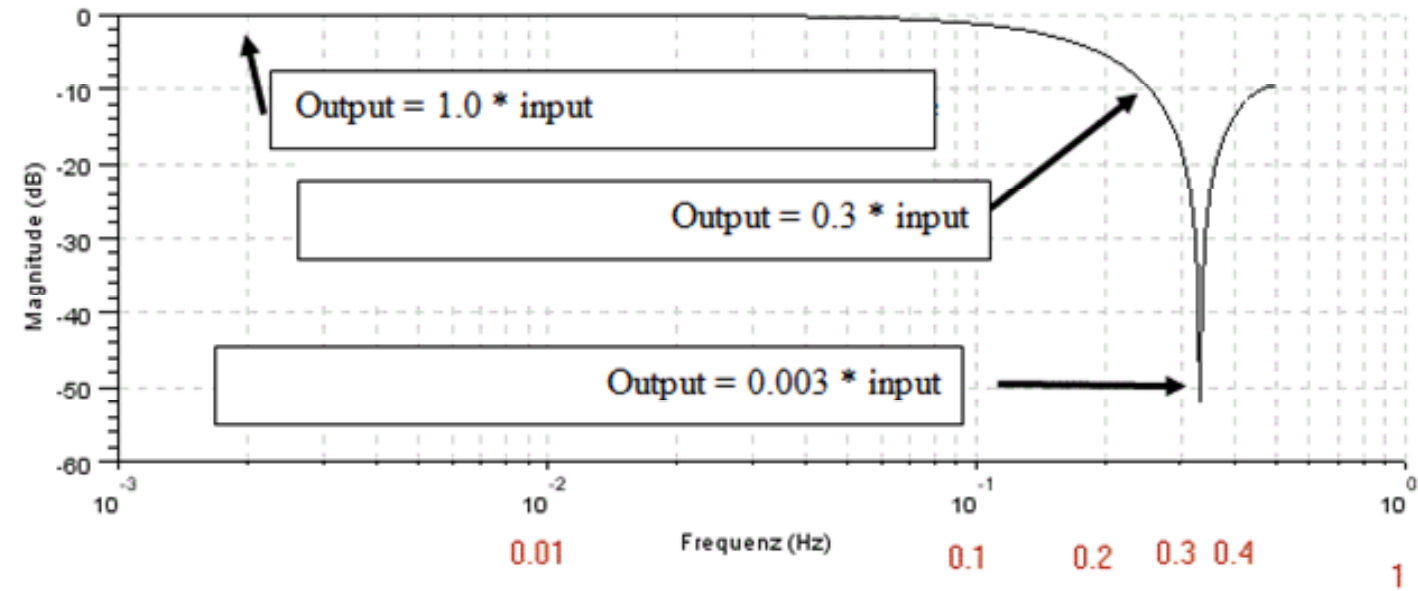
Gemiddelde gebruiken als filter



Interpretatie bodeplot

$$[V_{out}/V_{in}]_{dB} = 20 \log (V_{out}/V_{in}) \Leftrightarrow V_{out}/V_{in} = 10^{dB/20}$$

- Tot frequentie ongeveer 0,03 Hz is de versterking ongeveer 0 dB => versterking 1 en al deze frequenties worden niet verzwakt door de filter.
- Vanaf 0,1 Hz wordt het aantal dB negatief => verzwakking
- Tussen 0,3 Hz en 0,4 Hz de grootste verzwakking ongeveer -50 dB en daarna stijgt deze terug tot ongeveer -10 dB. Verklaren dit later waarom dit zo is.
- Bij -50 dB ($V_{out}/V_{in} = 10^{-50/20}$) verkrijg je een verzwakking van 0,003 => uitgang is zo goed als 0.
- Bij -10 dB is de verzwakking 0,316



Gemiddelde gebruiken als filter

- **Afsnijfrequentie of cut-off frequentie** van de moving average filter is de frequentie waarbij de sinusamplitude van V_{out} nog 0,707(of -3dB) is van de sinusamplitude V_{in} .
- De afsnijfrequentie van de moving average waarde met de parameter 1/3, 1/3 en 1/3 is ongeveer 0,16 Hz.
- Wat betekent dit?
 - Als de samplefrequentie gelijk is aan 8 kHz dan is de afsnijfrequentie van deze filter gelijk aan $0,16 \times 8 \text{ kHz}$ of 1280 Hz
 - Is de samplefrequentie gelijk aan 16 kHz dan is de afsnijfrequentie gelijk aan $0,16 \times 16 \text{ kHz}$ of 2560 Hz
- De **faseresponse** geeft weer hoeveel graden de sinusgolf aan de uitgang verschoven is ten opzichte van de sinusgolf aan de ingang. Hoe hoger de frequentie, hoe hoger de faseverschuiving.
- Onervaren gebruikers negeren dikwijls deze faseverschuiving. Nochtans is ze belangrijk omdat ze een grote invloed heeft op het uitgangssignaal (komen in een volgend hoofdstuk over digitale filter op terug)

