4– Average filter

Ing. Patrick Van Houtven

ARTESIS PLANTIJN
HOGESCHOOL ANTWERPEN

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.**

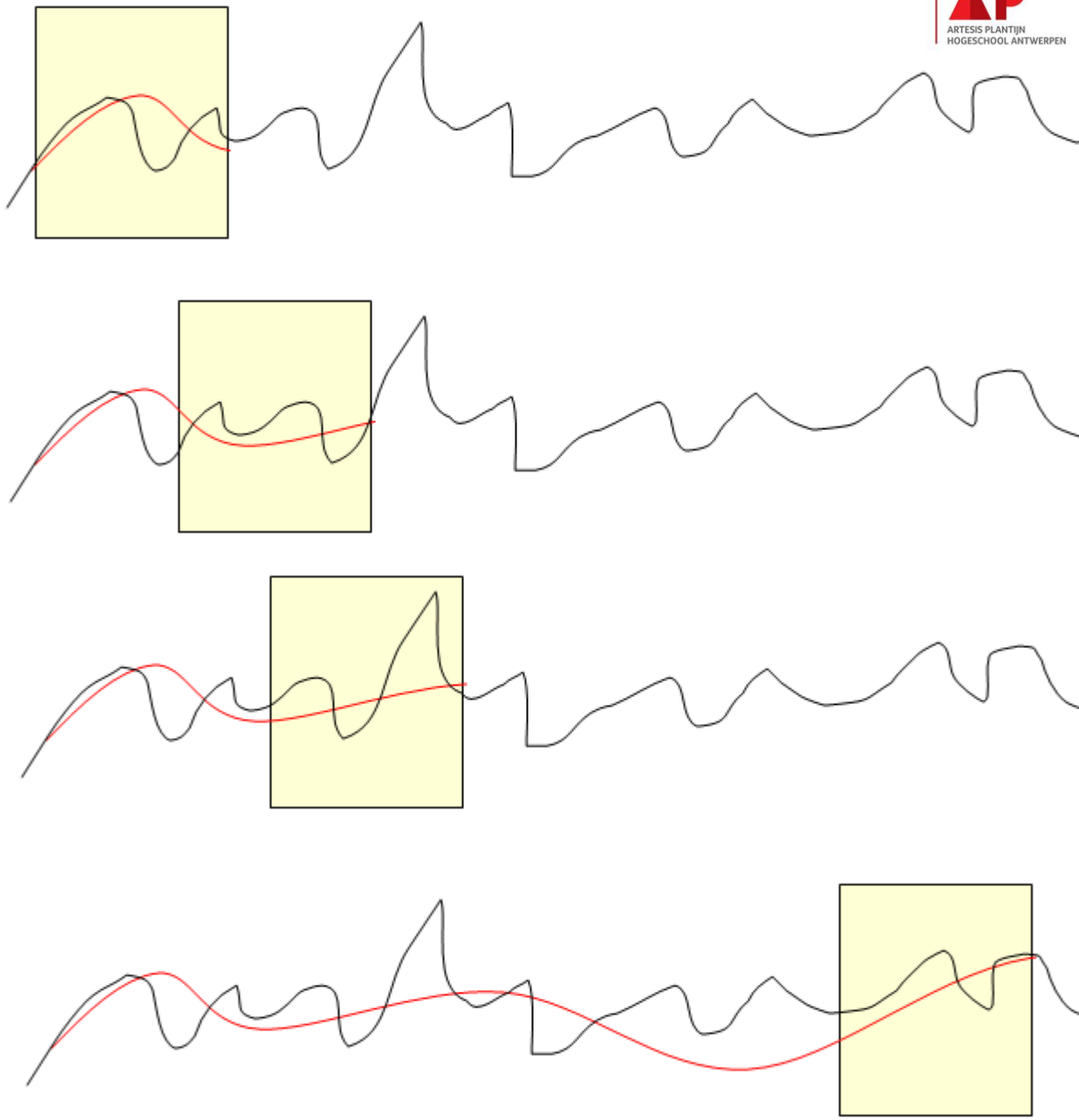
Aanverwante filters van Moving Average filters zijn onder andere de Gauss, Blackman, en multiplepass filter. Deze hebben (iets) betere prestaties in het frequentiedomein maar met nadeel dat ze meer rekentijd vergen om het resultaat te bekomen dan Moving Average.

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 weergeven als output voor bv. de centrale positie binnen dit venster.

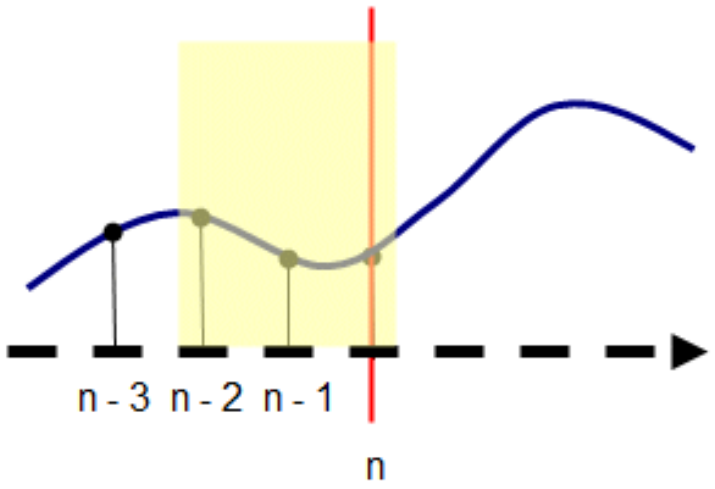
- Programmatorisch gezien kan het bewegen van het venster over de data verwezenlijkt worden met een for-loop. Na iedere berekening wordt het venster één positie verder verplaatst. (i = i+1)
- Bij het bepalen van het gemiddelde (avarage) worden enkel de berekeningen in het verleden en de huidige berekening 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)?



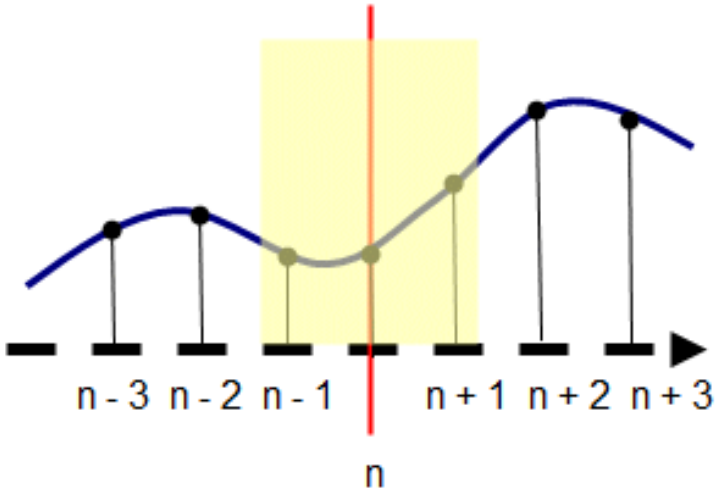
Gemiddelde gebruiken als filter

- 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.



$$\frac{X_{n-2} + X_{n-1} + X}{3}$$

ON-line



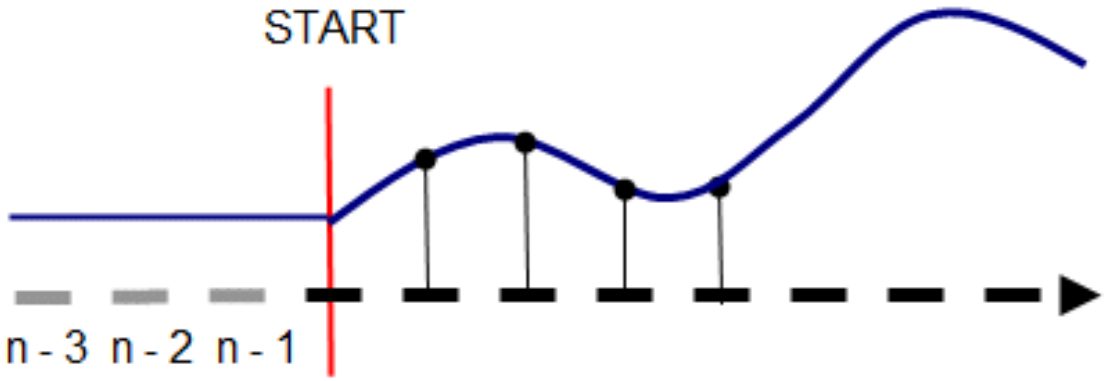
$$\frac{X_{n-1} + X + X_{n+1}}{3}$$

OFF-line

Gemiddelde gebruiken als 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)

Gemiddelde gebruiken als filter

Moving Average Filter

- De moving average filter is als volgt opgebouwd :

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

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

- Andere manier van opbouw:

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

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

- Voorbeeld :

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

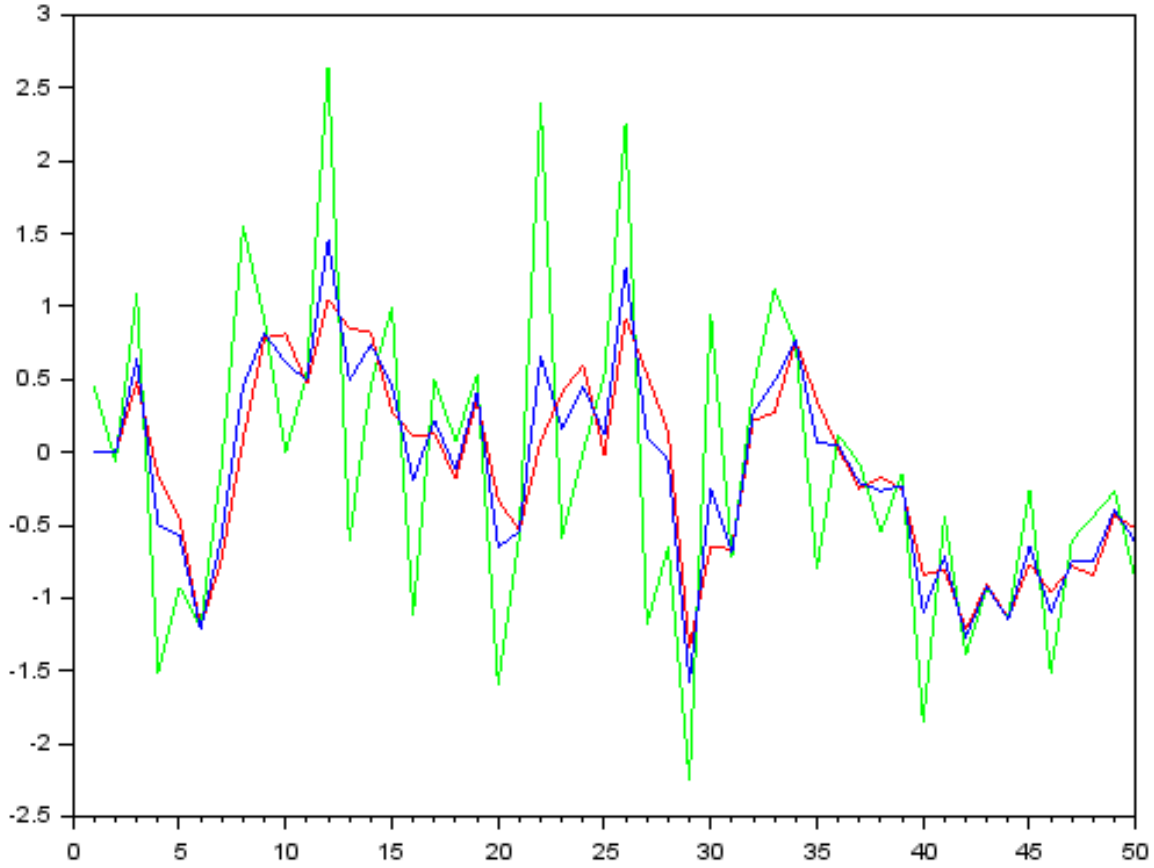
of:

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

Gemiddelde gebruiken als filter

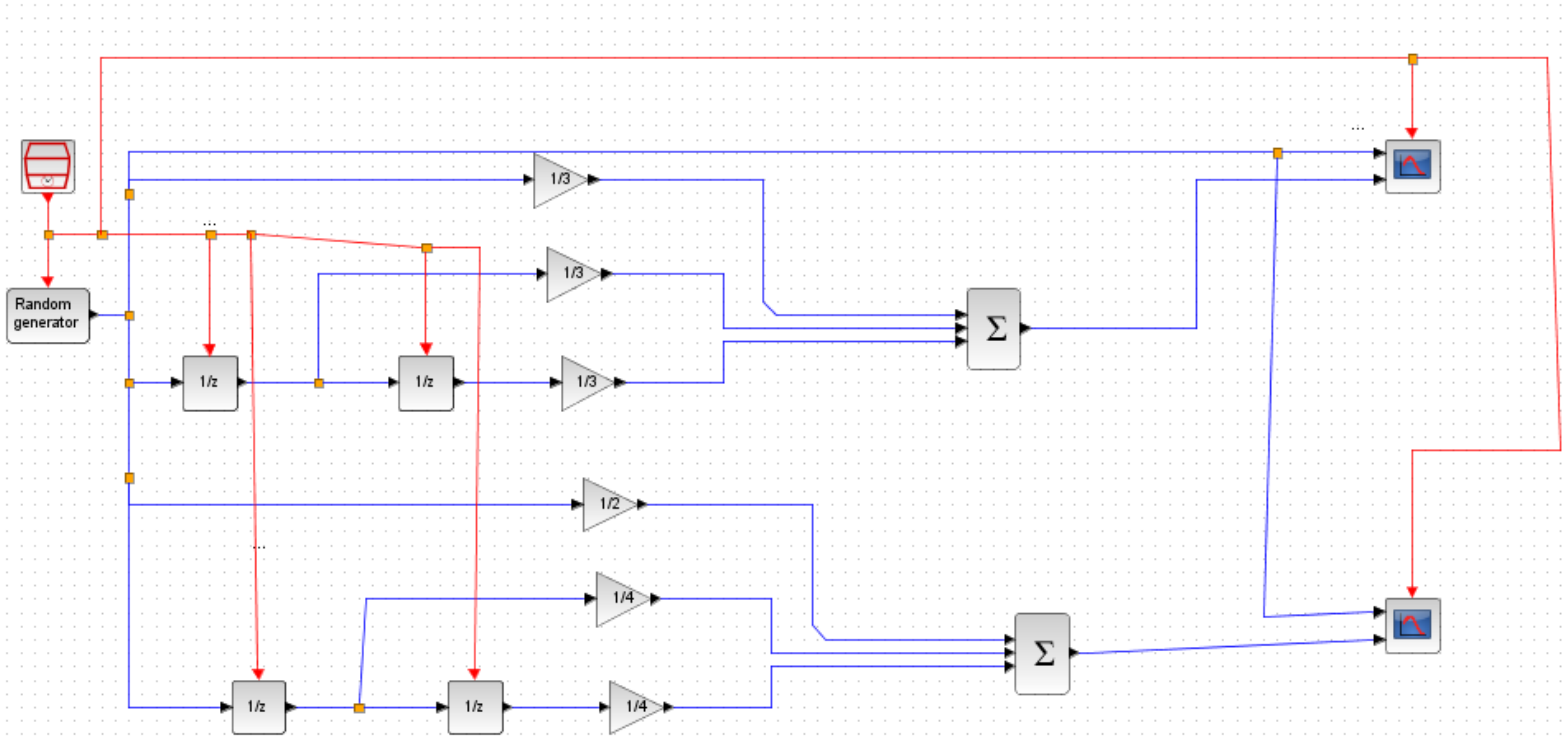
Voorbeeld Moving Average Filter

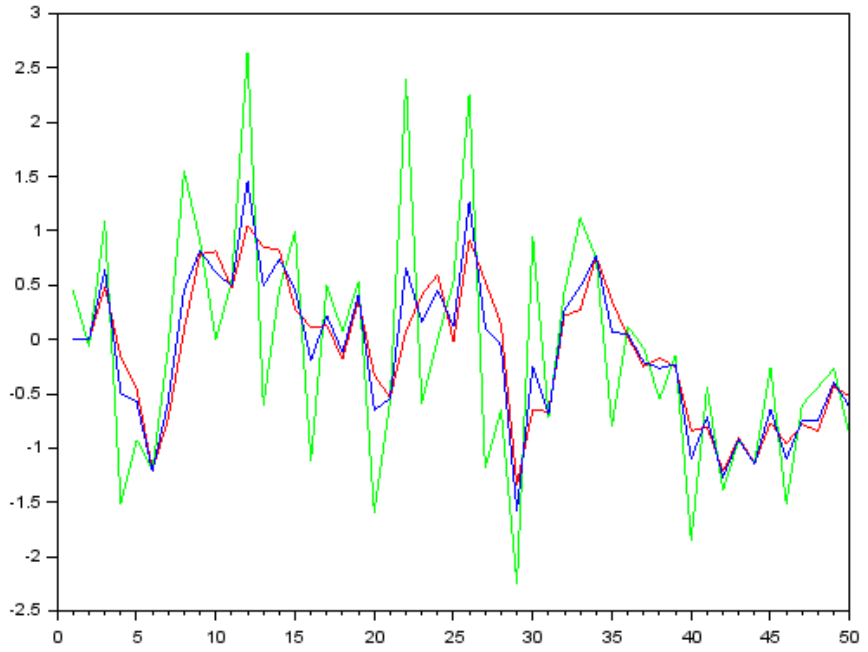
```
1  //Moving average filter op random 50 waarden
2  //opbouw met factoren 1.0
3  signal = rand(50,1,'normal')
4  filter_output = zeros (50,1)
5  //origineel signaal in groen  weergeven
6  plot(signal,'g')
7  //filter werking met factoren 0.33xn-2, 0.33xn-1 0.33 n
8  for i=3:length(signal)
9      filter_output(i)=0.33*signal(i-2)+ ...
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')
```



Gemiddelde gebruiken als filter

Voorbeeld Moving Average Filter in XCOS

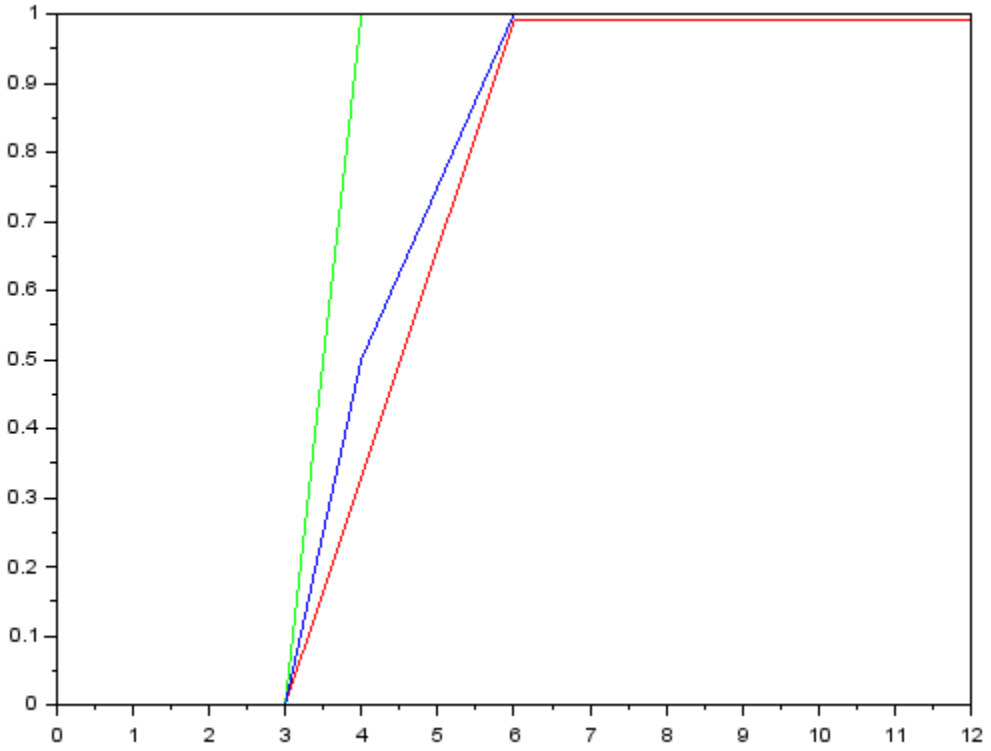




Gemiddelde gebruiken als filter

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.

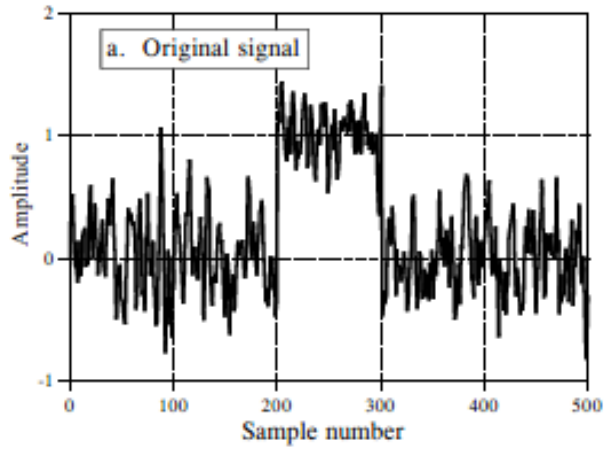
(a) Origineel pulssignaal met veel ruis op

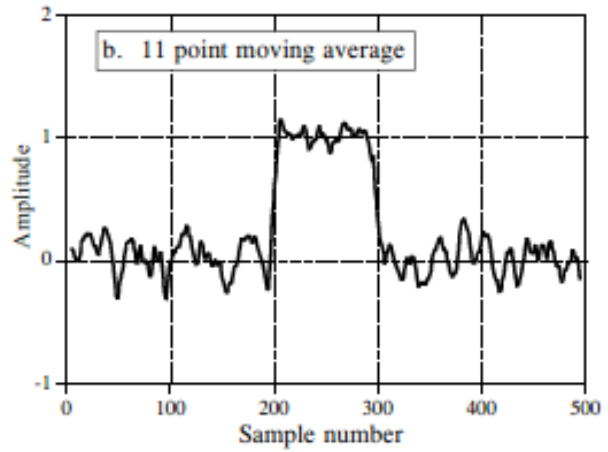
(b) Invloed van de filter vermindert de amplitude van de ruis (goed)

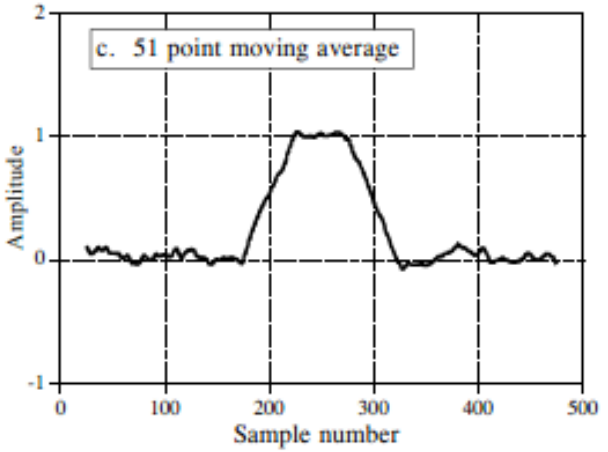
(c) maar verlaagt de scherpte van de randen (slecht)

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

Voorbeeld: 100 punt Moving Average Fiilter 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)

Gemiddelde gebruiken als 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)**

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

$$y = 0{,}25\,.Z^{-2} + 025.Z^{-1} + 0{,}50.Z$$

Hoe praktisch te werk gaan?

- De parameters van de moving avarage 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 coëfficënten (coeff) of wortels (roots)
- transferp is de naam van de polynominal

Gemiddelde gebruiken als filter

- Eens polynoom gevormd, staat deze in de representatie van z. Via de functie HORNER() wordt de polynomal 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:

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

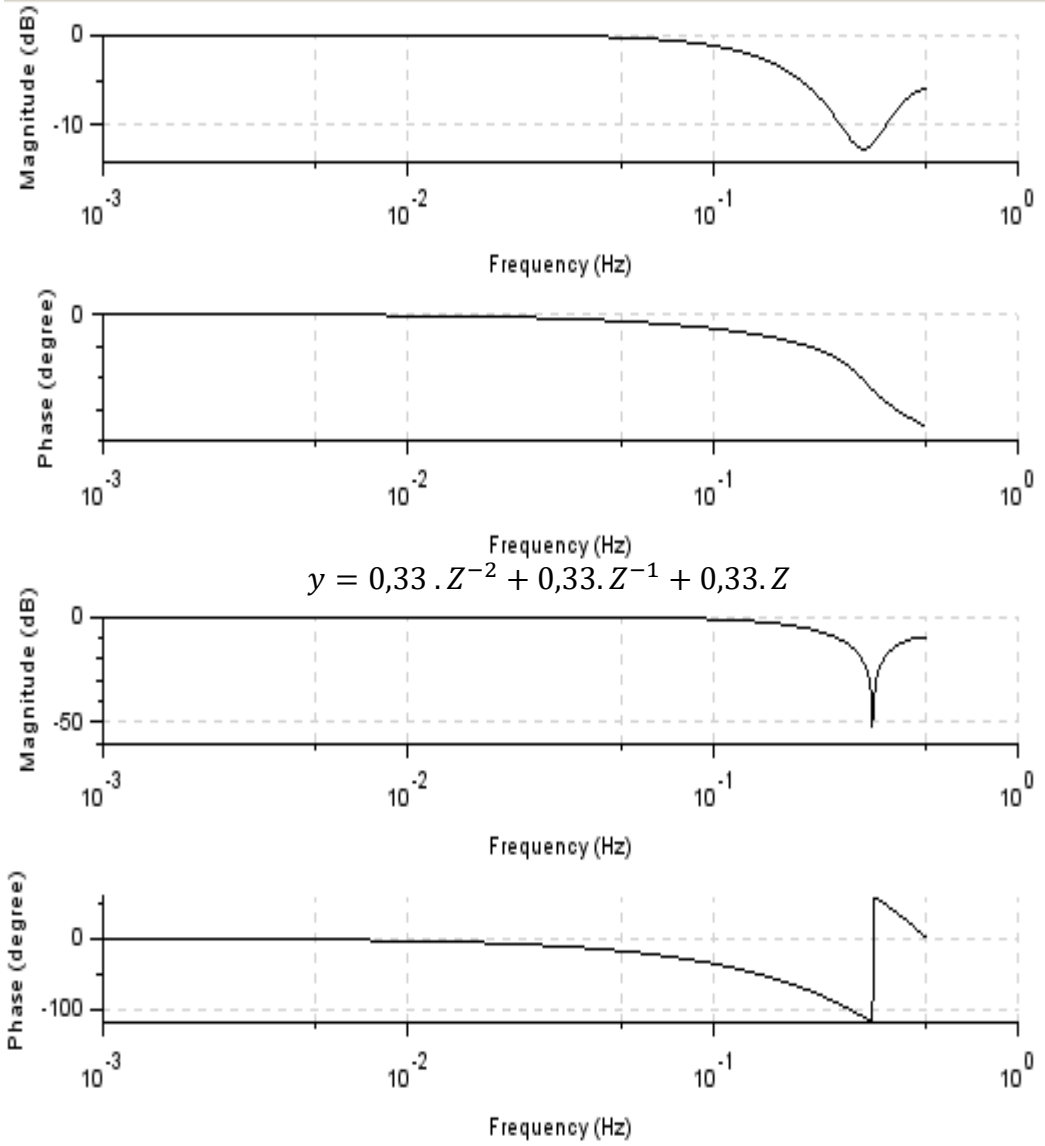
$$y = 0{,}33\,.Z^{-2} + 0{,}33.Z^{-1} + 0{,}33.Z$$

```
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)
```

Gemiddelde gebruiken als filter

```
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)
```

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

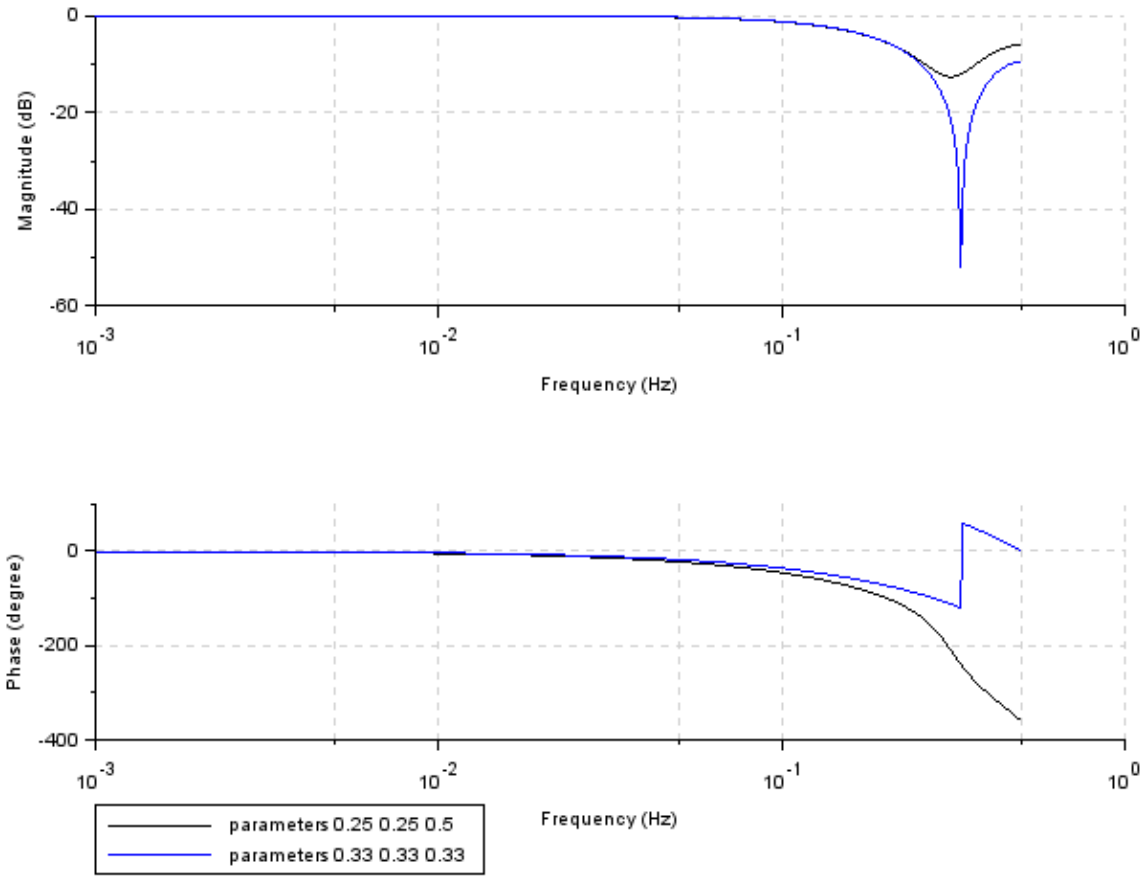


$$y = 0{,}33\,.Z^{-2} + 0{,}33.Z^{-1} + 0{,}33.Z$$

Gemiddelde gebruiken als filter

- Weergeven beide bodeplots over elkaar

```
//bodediagram van moving average filter
// filter y=0.25z-2 + 0.25 z-1 + 0.5 z
// opgave parameters van filter
parameter = [1/4 1/4 1/2]
// bepalen van polynoom
transferp = poly(parameter, 'z', 'coef')
//omvormen transfer poly in z via horner()naar 1/z
transferh = horner (transferp,(1/%z))
transfersys1 = syslin('d',transferh)
parameter = [1/3 1/3 1/3]
// bepalen van polynoom
transferp = poly(parameter, 'z', 'coef')
//omvormen transfer poly in z via horner()naar 1/z
transferh = horner (transferp,(1/%z))
//omzetten naar een discrete transferfunctie
transfersys2 = syslin('d',transferh)
//teken van bodediagrammen van beide filters en verduidelijking van de lijnen
bode([transfersys1; transfersys2],['parameters 0.25 0.25 0.5'; 'parameters 0.33 0.33 0.33'])
```

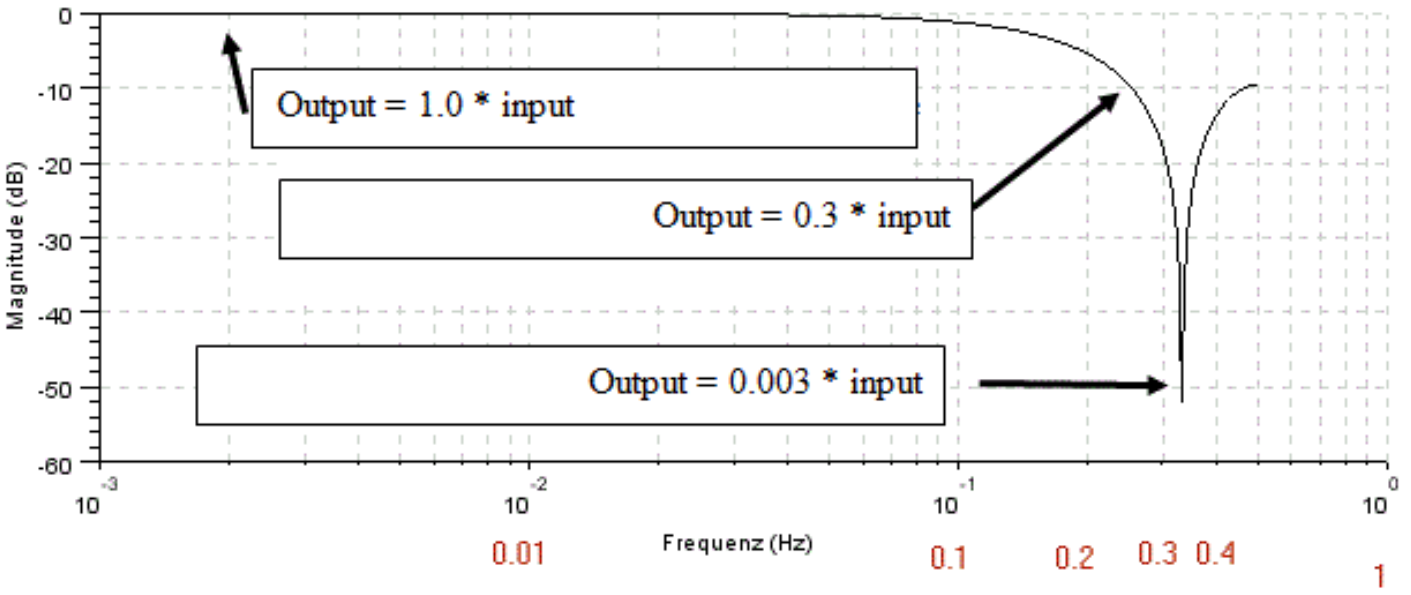


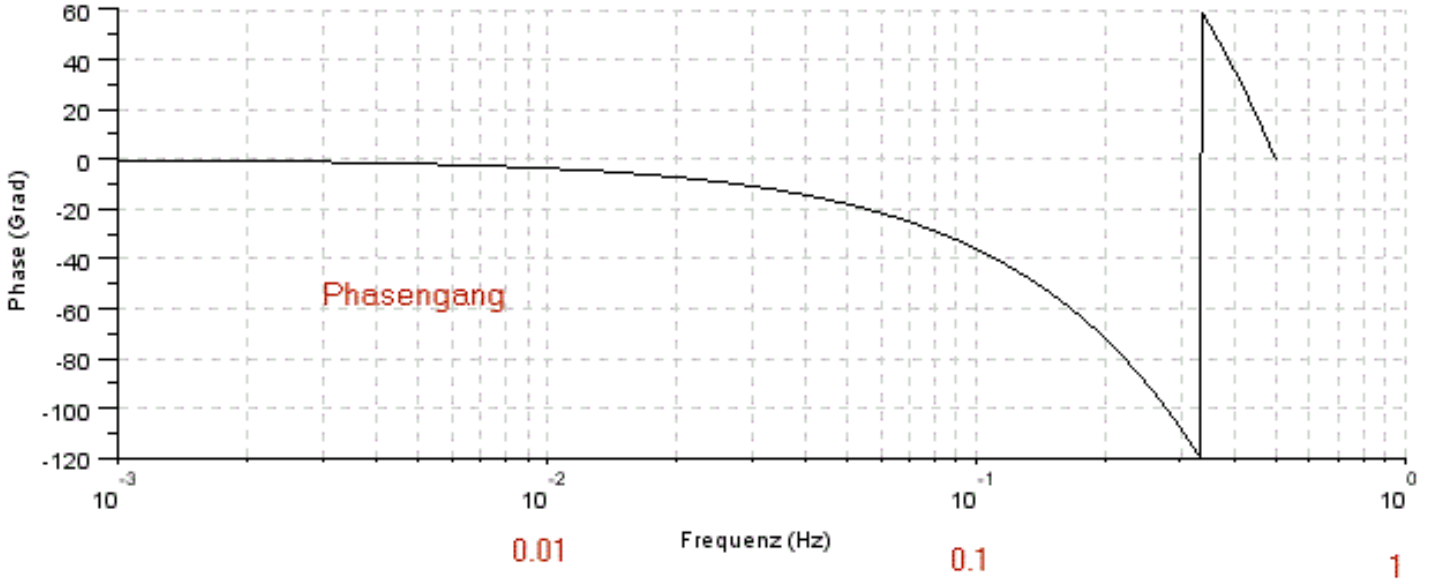
Gemiddelde gebruiken als filter

Interpretatie bodeplot

$[Vout/Vin]_{dB} = 20 \log (Vout/Vin) <=> Vout/Vin = 10^{dB/20}$

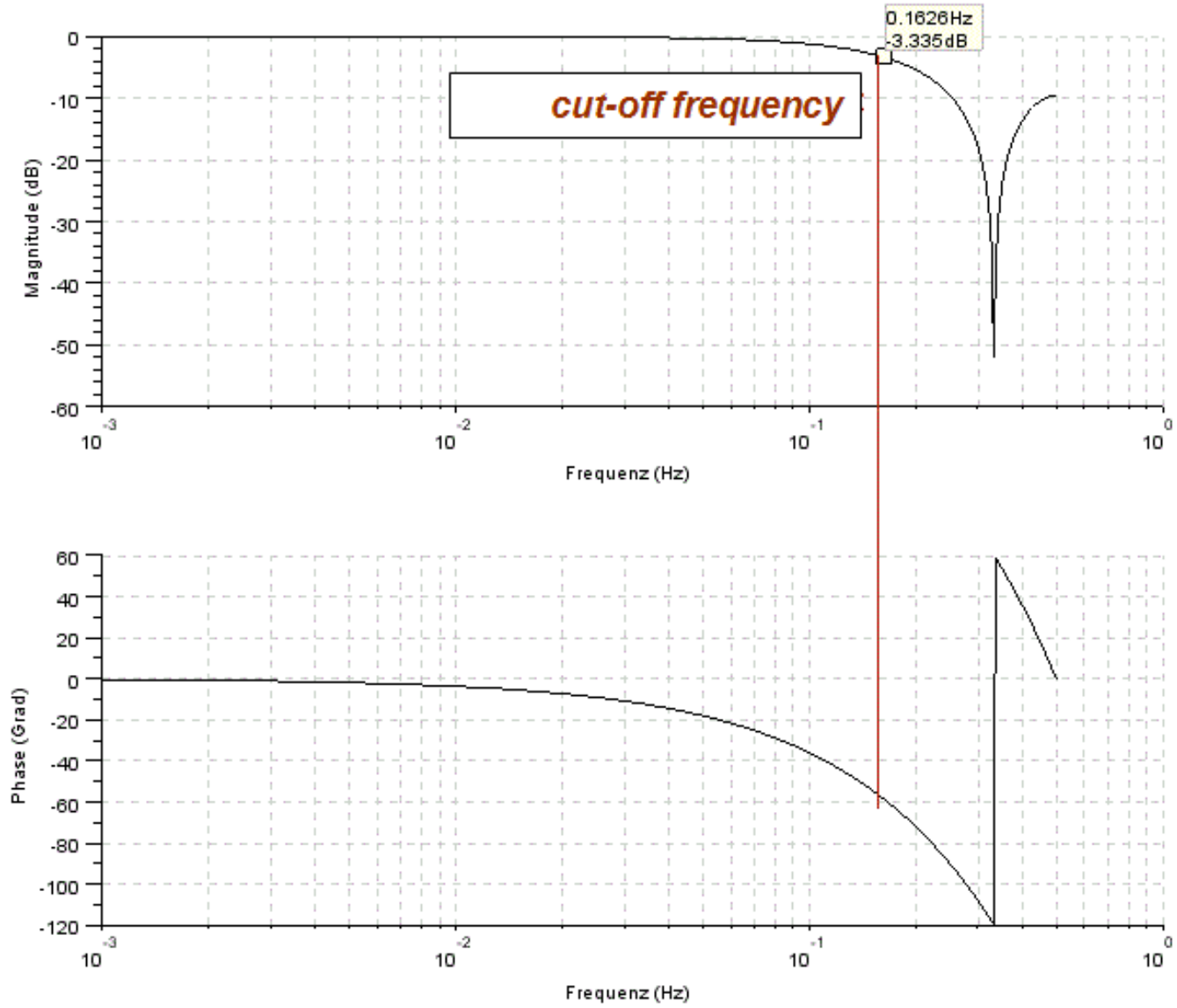
- Tot frequentie ongeveer 0,03 Hz is de versterking ongeveer 0 dB => versterking 1 en al deze frequenties worden niet verzwakt door de filter.
- Vanaf 0,1 Hz wordt het aantal dB negatief => verzwakking
- Tussen 0,3 Hz en 0,4 Hz de grootste verzwakking ongeveer -50 dB en daarna stijgt deze terug tot ongeveer -10 dB. Verklaren dit later waarom dit zo is.
- Bij -50 dB ($Vout/Vin = 10^{-50/20}$) 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 Vout nog 0,707(of -3dB) is van de sinusamplitude Vin.
- 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 x 8 kHz of 1280 Hz
  - Is de samplefrequentie gelijk aan 16 kHz dan is de afsnijfrequentie gelijk aan 0,16 x 16 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)

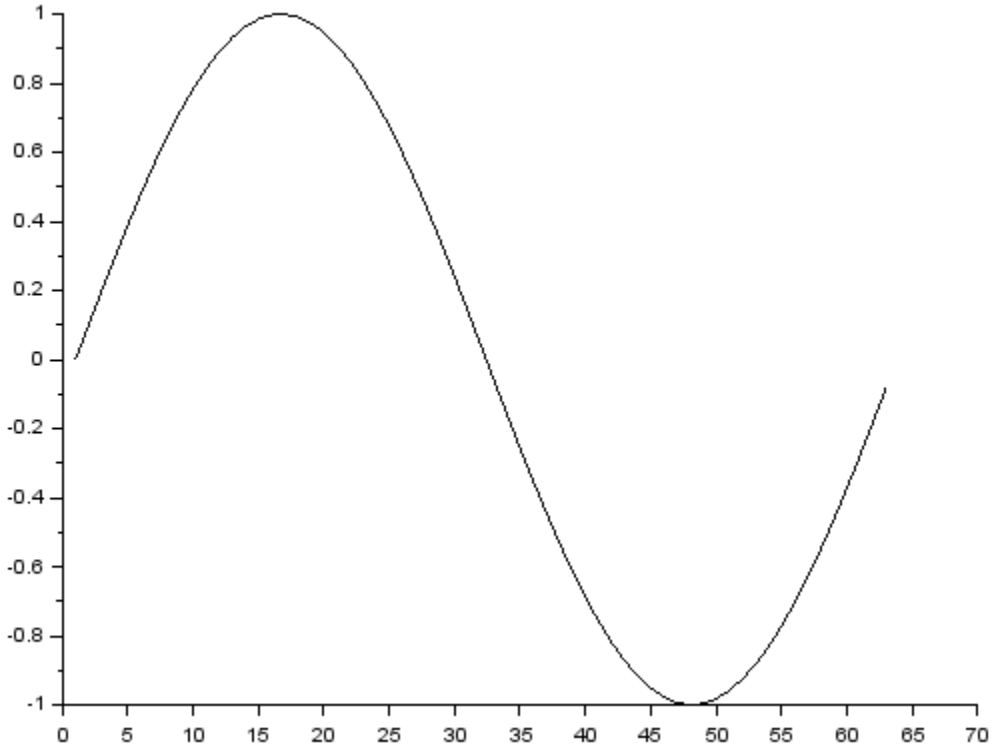


Gemiddelde gebruiken als filter

PLOT en PLOT2D

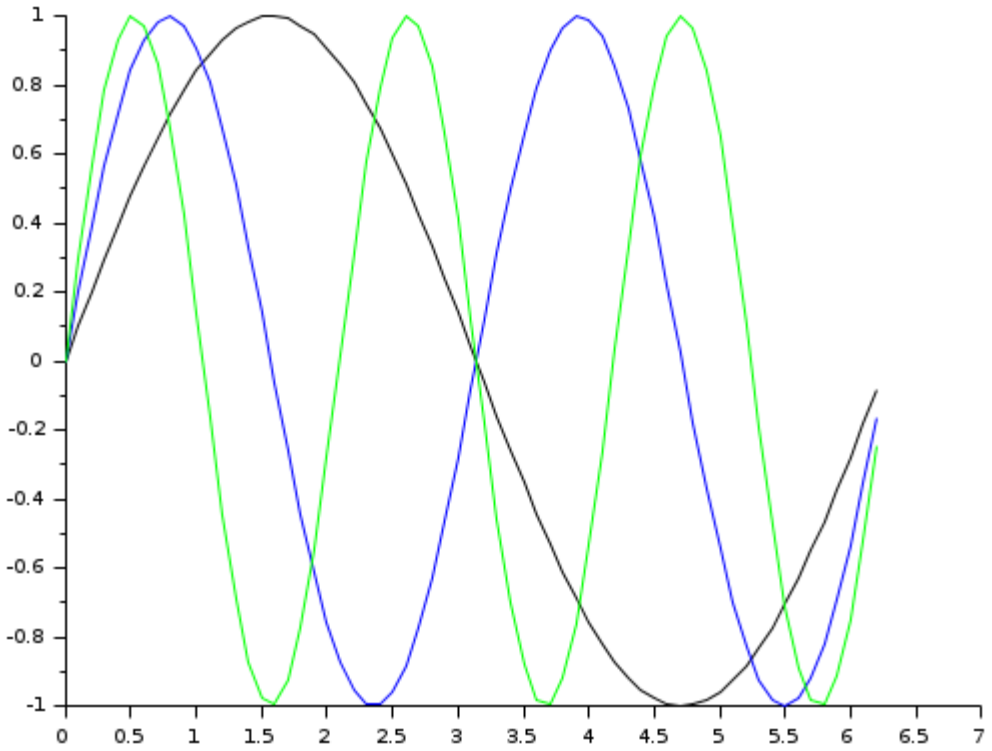
- De functie plot() hebben we tot nu toe gebruikt om signalen te visualiseren. Deze biedt echter niet zoveel mogelijkheden.
- plot2d()

```
1 // x initialisation
2 x=[0:0.1:2*%pi]';
3 //simple plot sinx met x
4 //varierend van 0 tot 6.28 (2pi)
5 //telkens stap 0.1 => 6.28/0.1 stappen
6 plot2d(sin(x));
```

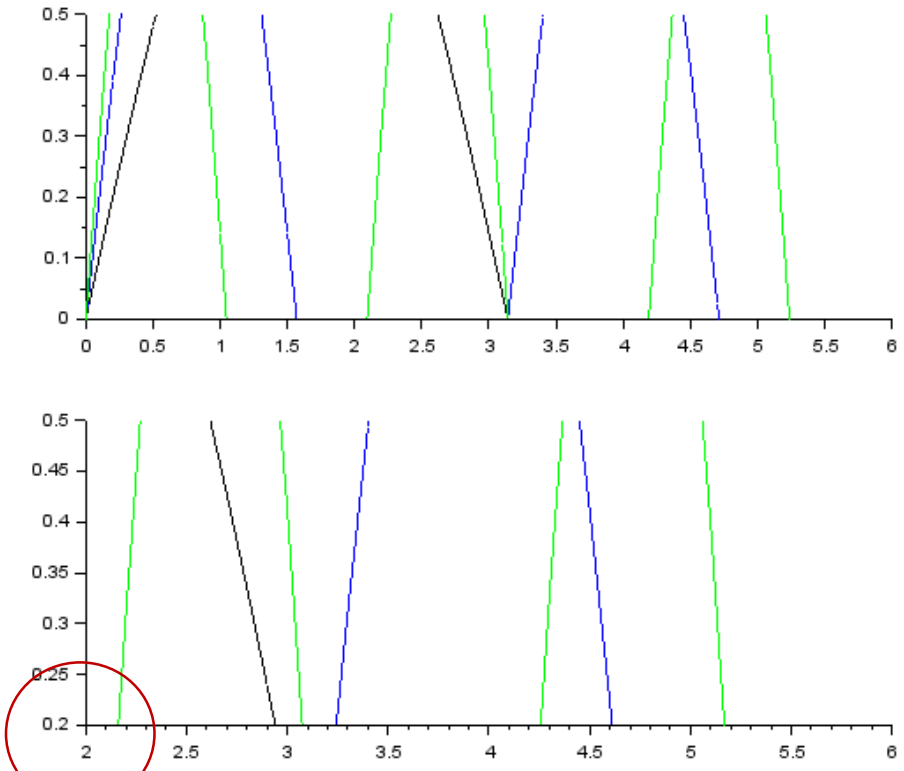


Gemiddelde gebruiken als filter

```
clf();
// verschillende sinussen op dezelfde tijdbasis
x=[0:0.1:2*%pi]';
plot2d(x,[sin(x) sin(2*x) sin(3*x)])
```



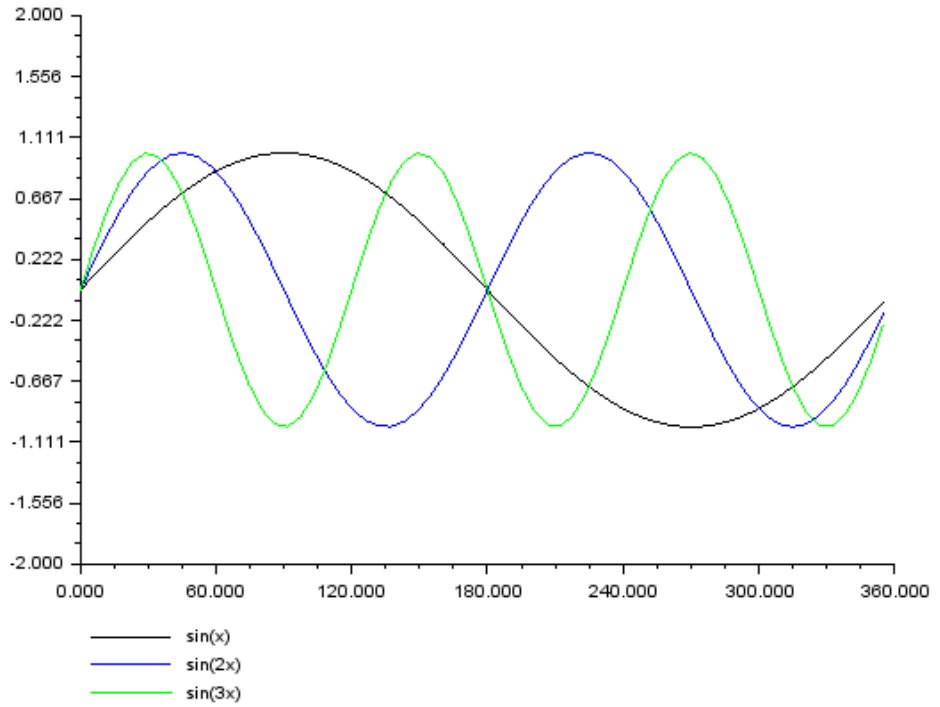

```
// multiple plot giving the dimensions of the frame
clf();
x=[0:0.1:2*%pi]';
subplot(2, 1, 1)
plot2d(x,[sin(x) sin(2*x) sin(3*x)],rect=[0,0,6,0.5]);
subplot(2, 1, 2)
plot2d(x,[sin(x) sin(2*x) sin(3*x)],rect=[2,0.2,6,0.5]);
```



Gemiddelde gebruiken als filter

- Plot2d() met legende en afbakening waarden op assen

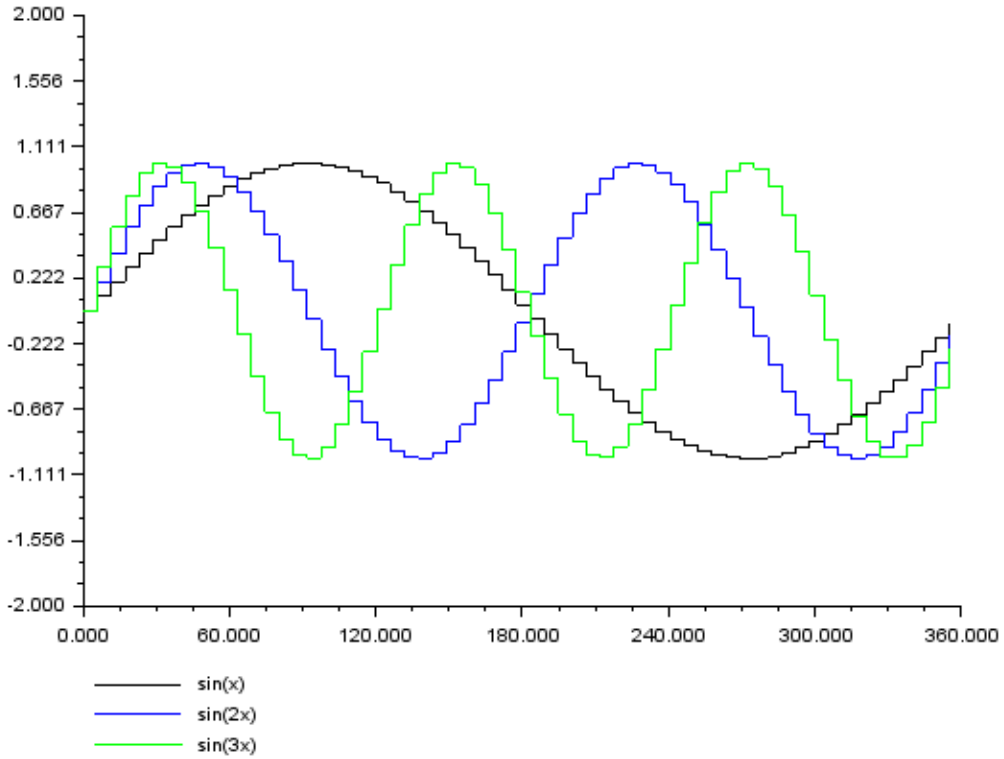
```
//multiple plot with captions and given tics + style
clf();
x=[0:0.1:2*%pi]';
//schaal in graden in plaats van radialen
graden = [0 :(0.1*360)/(2*%pi) :360]
//leg = legende met @ als afscheiding tussen waarden
//rec = gebied waartussen gegevens zichtbaar worden gemaakt
//rect = [min x, min y, max x, max y]
//nax = onderverdeling van de schaal
//nax = [aantal verdelingen tussen 2 waarden x, aantal x-waarden benoemd, ...
//na = ... aantal verdelingen tussen 2 waarden y, aantal y-waarden benoemd ]
plot2d(graden,[sin(x) sin(2*x) sin(3*x)],...
       [1,2,3],leg="sin(x)@sin(2x)@sin(3x)",nax=[3,7,2,10],rect=[0,-2,360,2]);

```



Gemiddelde gebruiken als filter

- Plot2d2() visualisering digitale sampling – maken trapspanning

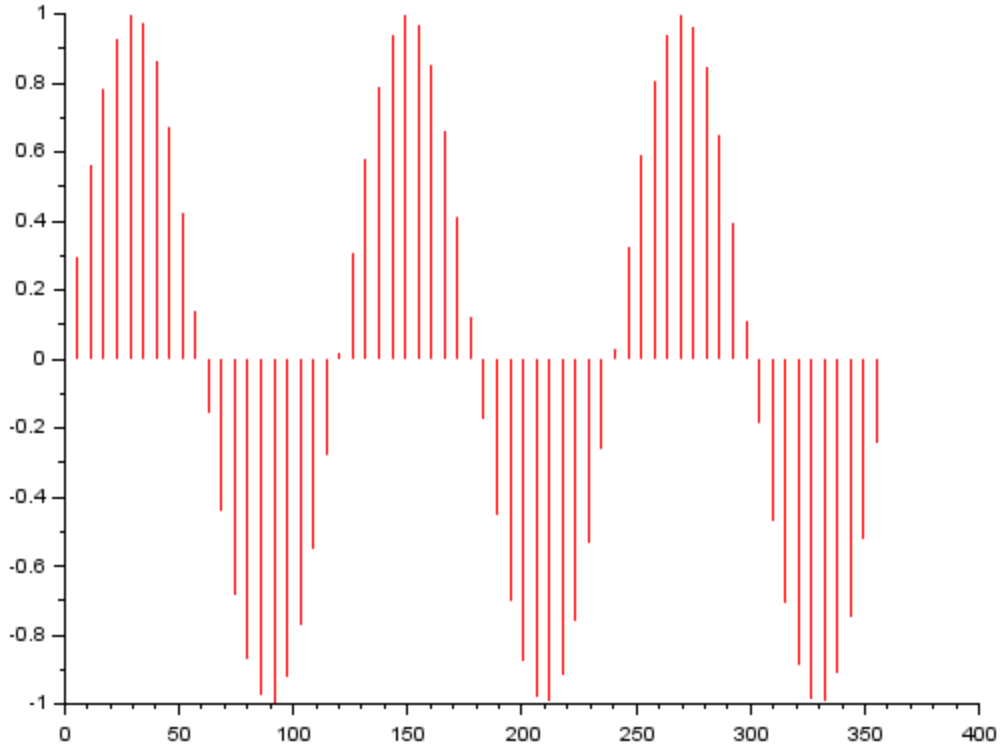
```
//plot2d2 kan gebruikt worden om digitale sampling te visualiseren
//of trapspanning te maken

plot2d2(graden,[sin(x) sin(2*x) sin(3*x) ],...
        [1,2,3],leg="sin(x)@sin(2x)@sin(3x)",nax=[3,7,2,10],rect=[0,-2,360,2]);
```



Gemiddelde gebruiken als filter

- Plot2d3() kan gebruikt worden voor lijnspectrum

```
21 //[5] optioneel voor keuze kleur van lijnspectrum
22 plot2d3(graden, sin(3*x) , [5]);
```



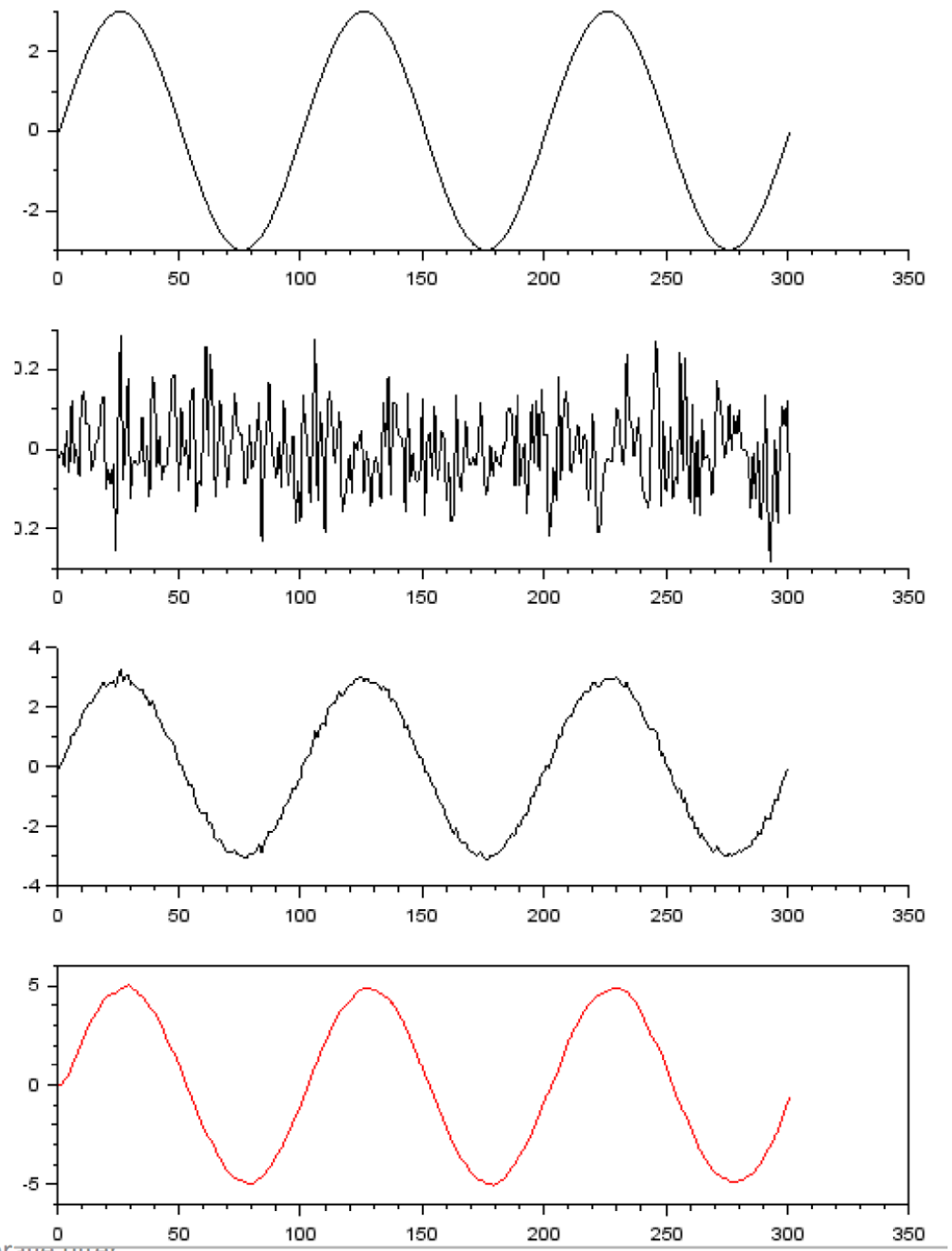
Gemiddelde gebruiken als filter

Ontwerp een moving average filter als ruisfilter.

Stel een sinussignaal van 100 Hz en amplitude 3 V. Hierop is een ruissignaal gesupperponeerd met een amplitude hoofdzakelijk rond 10 mv. (pieken kunnen hogere waarden bekomen)

Ontwerp een filter met minimaal 5 factoren om deze ruis te minimaliseren.

Bv. 0,33 0,33 0,33 0,33 0,33



```
//genereren van 100 Hz signaal fs = 10 kHz
t = 0 : 0.0001 : 0.03
sign = 3*sin(2*%pi*100*t)
subplot(4,1,1)
plot2d(sign)
//genereren van ruis random 200 mV
ruis =0.1* rand(length(sign),1,'normal')
subplot(4,1,2)
plot2d(ruis)
sign1 = sign + ruis'
subplot (4,1,3)
plot2d(sign1)
//filter werking met factoren 0.33xn-2, 0.33xn-1 0.33 n
for i=5:length(sign1)
    filter_output(i)=0.2*sign1(i-4) + 0.2*sign1(i-3)+...
    0.2*sign1(i-2)+ ...
     0.2*sign1(i-1)+ 0.2*sign1(i)
end
//filteroutput weergegeven in rood
subplot(4,1,4)
plot(filter_output,'r')

```