

Introductie moving average filter

- ***Meest voorkomende filter in DSP***
- ***Optimaal voor verwijderen van random ruis terwijl een scherpe stapresponse behouden blijft***
- ***Zeer geschikt voor tijdsdomein, slechtste filter voor frequentiedomein***
 - ***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

- 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.

Gemiddelde gebruiken als filter

- *Werken met factor 1,0*

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

Andere manier van opbouw :

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

Gemiddelde gebruiken als filter

- *Werken met factoren (gewichten)*

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

Andere manier van opbouw :

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

Voorbeeld opbouw moving average filter

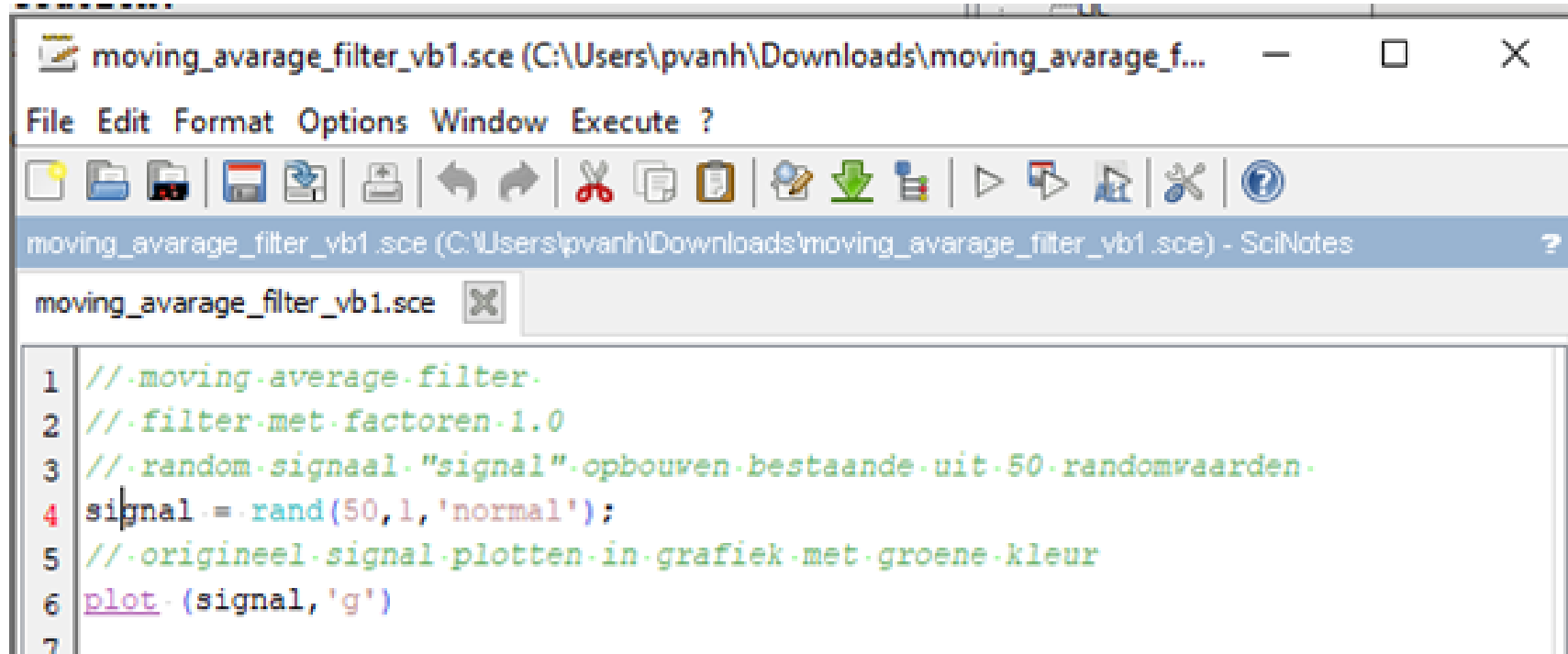
Stel dat we een vergelijking willen maken tussen de filter met gelijke gewichten
 $y = 0,33 \cdot x_{n-2} + 0,33x_{n-1} + 0,33x$ en de filter met verschillende gewichten:
 $y = 0,25 \cdot x_{n-2} + 0,25x_{n-1} + 0,50x$

We maken de vergelijking met een signaal bestaande uit 50 random samples

De code schrijven we in Scilab. Hiervoor gebruiken we SciNotes als editor
(terug te vinden onder het hoofdmenu *Applications*)

signal = rand(50,1,'normal');

Hoe in scilab een “signaal” opbouwen, bestaande uit 50 randomwaarden?



The screenshot shows the Scilab SciNotes application window. The title bar reads "moving_avarage_filter_vb1.sce (C:\Users\pvanh\Downloads\moving_avarage_f...". The menu bar includes "File", "Edit", "Format", "Options", "Window", and "Execute ?". The toolbar contains various icons for file operations and execution. The active window is titled "moving_avarage_filter_vb1.sce (C:\Users\pvanh\Downloads\moving_avarage_filter_vb1.sce) - SciNotes". The script content is as follows:

```
1 //.moving-average-filter.  
2 //.filter-met-factoren-1.0  
3 //.random-signaal-"signal"-opbouwen-bestaande-uit-50-randomwaarden-  
4 signal = rand(50,1,'normal');  
5 //.origineel-signal-plotten-in-grafiek-met-groene-kleur  
6 plot (signal,'g')  
7
```

```
signal = rand(50,1,'normal');  
plot (signal,'g')
```

Hoe in scilab een “signaal” opbouwen, bestaande uit 50 randomwaarden?

clf

```
signal = rand(50,1,'normal');  
plot (signal,'g')
```

Parameter voor rand()

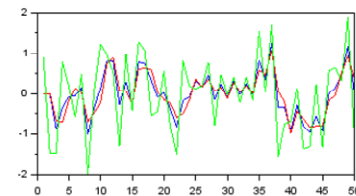
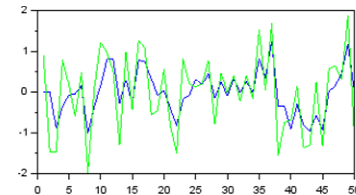
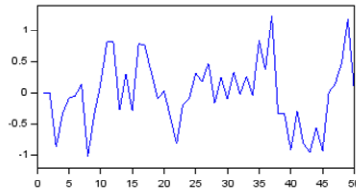
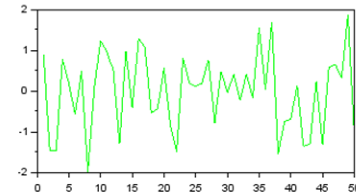
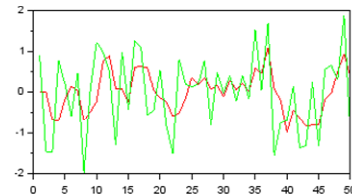
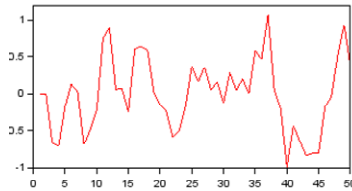
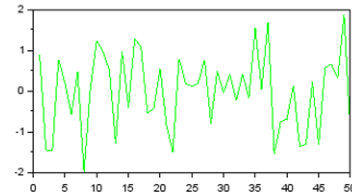
‘normal’ = verdeling volgens gausachtige curve

‘uniform’ = gelijkmatige verdeling

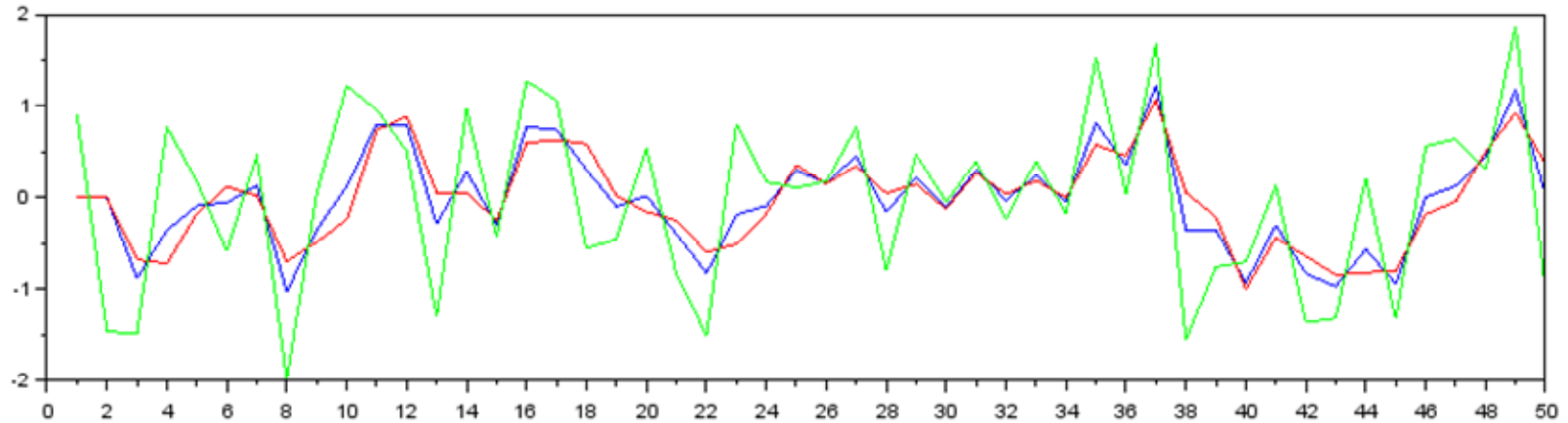
Vergelijk : $y = 0,33 \cdot x_{n-2} + 0,33x_{n-1} + 0,33x$ met $y = 0,25 \cdot x_{n-2} + 0,25x_{n-1} + 0,50x$

```
filter_output = zeros(50,1)
// filter werking met factoren 0,33 x[n-2] 0,33 x[n-1] 0,33
for i=3:length (signal)
    filter_output(i) = 0.33 * signal(i-2)+0.33 * signal (i-1) + 0.33 *signal(i);
end
subplot (2,1,2)
plot (filter_output, 'r');

filter_output = zeros(50,1)
// filter werking met factoren 0,33 x[n-2] 0,33 x[n-1] 0,33
for i=3:length (signal)
    filter_output(i) = 0.25 * signal(i-2)+0.25 * signal (i-1) + 0.50 *signal(i);
end
subplot (2,1,2)
plot (filter_output, 'r');
```



Vergelijk : $y = 0,33 \cdot x_{n-2} + 0,33x_{n-1} + 0,33x$ met $y = 0,25 \cdot x_{n-2} + 0,25x_{n-1} + 0,50x$



Stapresponse van de Moving Average Filter

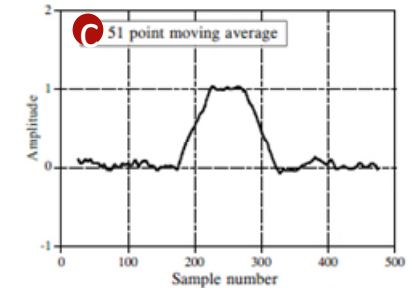
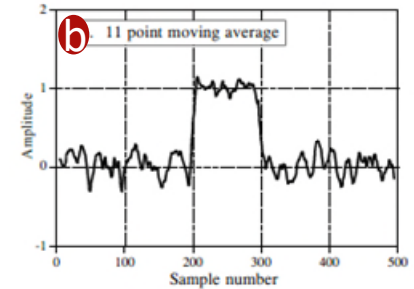
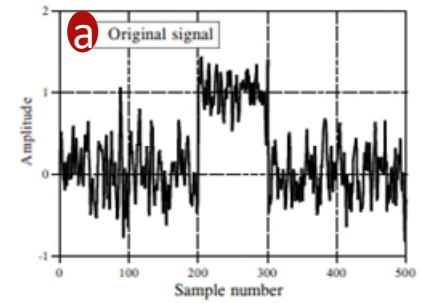
- Met de stapresponse kan gemakkelijk het verschil tussen de transfertfuncties worden aangetoond.

```
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 via Moving Average Filter

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 scherpthe van de randen (slecht)



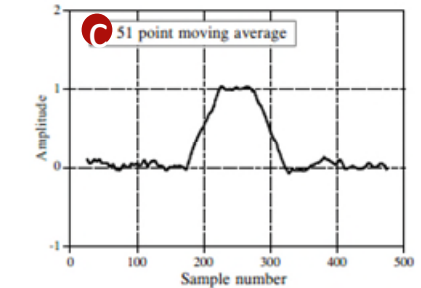
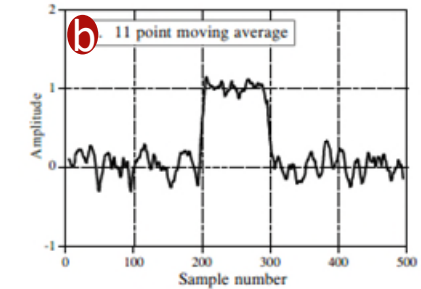
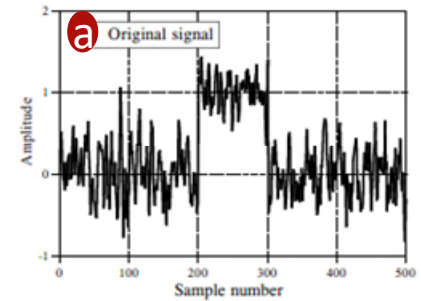
Ruisonderdrukking via Moving Average Filter

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

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 wordt bekomen door alle samples gelijkwaardig te behandelen (dus moving average filter)



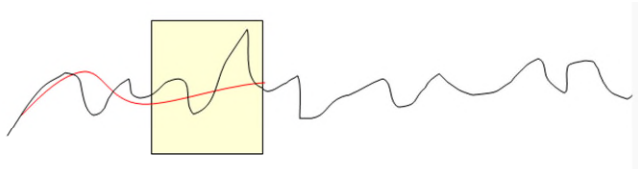
Eventjes herhalen

Moving Average Filter is optimaal voor het verwijderen van random ruis terwijl een scherpe stapresponse behouden blijft

Zeer geschikt voor filtering in het tijdsdomein

Niet geschikt voor filtering in het frequentiedomein

Eventjes herhalen



- ***Bewegen van venster kan met for-loop verwezenlijkt worden***
- ***Bepalen van het gemiddelde, enkel berekeningen van het huidige en een aantal van het verleden in rekening gebracht***

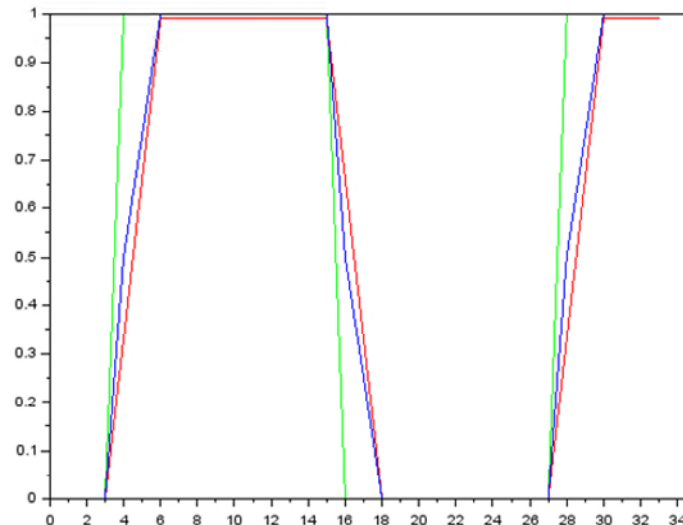
```
clf
signal = rand(50,1,'normal');

filter_output = zeros(50,1)

for i=3:length (signal)
    filter_output(i) = 0.33 * signal(i-2)+0.33 * signal (i-1) + 0.33 *signal(i);
end
```

Eventjes herhalen

- *Som van de gewichten (factoren) is steeds gelijk aan 1 (100 %)*
 - $y = 0,10 \cdot x_{n-2} + 0,20x_{n-1} + 0,70x$
 - $y = 0,20 \cdot x_{n-2} + 0,20x_{n-1} + 0,20x + 0,20 \cdot x_{n+1} + 0,20 \cdot x_{n+2}$
- *Moving Average Filter is vooral geschikt voor onderdrukking van witte ruis terwijl de scherpste stapresponsie behouden blijft.*



Frequentieresponse Moving Average Filter

- *Praktisch : filters in het frequentiedomein getest (bv. via bodediagram)*
- *In scilab gebruik maken van bode() –functie*
- *Om bode() te kunnen gebruiken zijn eerst een aantal omvormingen noodzakelijk*

Frequentieresponse Moving Average Filter

- *Stel filter met volgende gewichten : $y = 0,25 \cdot x_{n-2} + 0,25x_{n-1} + 0,50x$*
 - *Doorlopen for-lus*
 - ➔ *samples tijdsdomein in y-vector*
 - ➔ *transferfunctie nodig om deze om te zetten naar frequentiedomein*
 - *hierin zijn de transferparameters Z^{-1}, Z^{-2}, \dots*
- ➔ $y = 0,25 \cdot x_{n-2} + 0,25x_{n-1} + 0,50x \Rightarrow y = 0,25 \cdot Z^{-2} + 0,25 \cdot Z^{-1} + 0,50 \cdot Z$ (f_s genormaliseerd naar 1)

Frequentieresponse Moving Average Filter

- *Hoe praktisch te werk gaan?*

- *Parameters moving average filter opslaan als rijvector ($Z, Z^{-1}, Z^{-2}, Z^{-3}, \dots, Z^{-n}$)*
- *De rijvector is niet automatisch omgezet => eerst scilab duidelijk maken dat de rijvector een veelterm of polynoom bevat*
- **Eerste stap:** *scilab duidelijk maken dat de rijvector een polynoom bevat via de poly()-functie*
 - **transferp = poly(parameter, 'z', coeff)**

naam voor de
polynoom (vrij te
kiezen)

Weergeven of het gaat om coëfficiënten (coeff) of
wortels (roots)

Soort van transferfunctie ('z' discreet; 's' continu)

De opgeslagen parameters in een rijvector

Frequentieresponse Moving Average Filter

- *Hoe praktisch te werk gaan?*

- **Tweede stap** : *polynoom omvormen van z-representatie naar 1/z-representatie => transferfunctie wordt bekomen*
- *Via horner() gebeurt de omvorming van z naar 1/z*
- ***transferh = horner(transferp, (1/%z))***

naam voor de 1/z-
representatie (vrij te
kiezen)

Aangeven 1/z-representatie gewenst

De polynoom die gevormd is van de
samples uit het tijdsdomein

Frequentieresponse Moving Average Filter

- *Hoe praktisch te werk gaan?*

- **Derde stap:** *scilab duidelijk maken dat de functie die beschreven wordt een discrete transferfunctie is*
- *Via syslin() en de parameter 'd' kan dit verwezenlijkt worden*
- ***transfersys = syslin('d',transferh)***

naam voor de
discrete
transferfunctie (vrij te
kiezen)

Vector met de 1/z-representatie

Aangeven discrete transferfunctie maken

Frequentieresponse Moving Average Filter

- *Hoe praktisch te werk gaan?*
 - *Vierde stap: bodediagram maken van de moving average filter*
 - *bode(transfersys)*

Bodediagram van een filter weergeven in scilab:

- *Stap 1: de uitgangsvector met de samplewaarden in het tijdsdomein van de filter omzetten naar een polynoom in z-representatie*
- *Stap 2 : polynoom omvormen van z-representatie naar 1/z representatie*
- *Stap 3: duidelijk maken dat de 1/z-representatie een discrete transferfunctie is.*
- *Stap 4 : gebruik van de bodefunctie*

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



x_{n-2}, x_{n-1}, x_n

poly(uitgangsvector, 'z',coeff)



z^3, z^2, z

horner(polynoom, (1/%z))



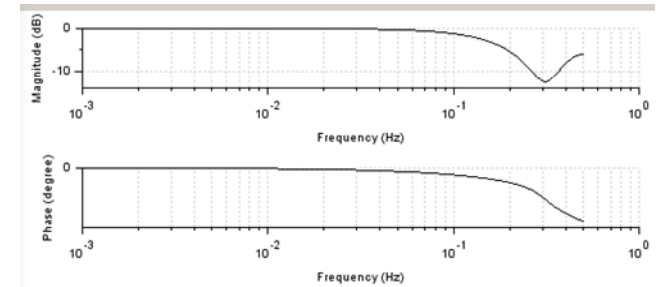
z^{-3}, z^{-2}, z^{-1}

syslin('d', 1/Z-representatievector)



filtercoëfficiënten

Bode(syslin_omgetransformeerde_vector)

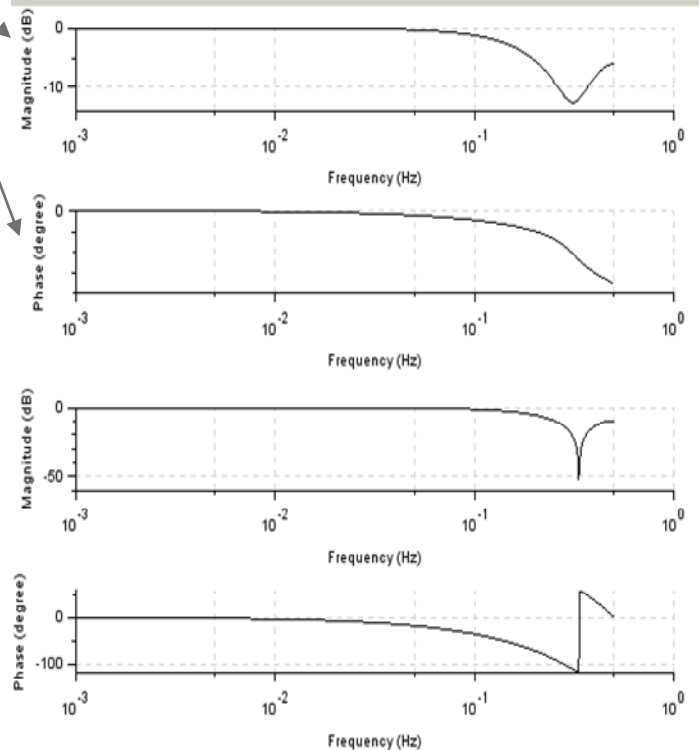


frequentieresponse

Codevoorbeeld:

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

```
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=sslin('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=sslin('d',transferh)
20 //teken.van.bodediagram.van.de.filter
21 bode(transfersys)
```



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