

Practicumverslag voor Data Assimilatie

Remko Moerkerken, 9575336, remkom@ch.twi.tudelft.nl

Sebastiaan, 9121164, sebastia@ch.twi.tudelft.nl

3 juni 2003

1 Inleiding

Bij het modelleren en simuleren van een situatie kun je met behulp van echte data het verloop van de simulatie verbeteren. Probleem is echter dat zowel de gemeten data als het model zelf zeer waarschijnlijk ruis bevatten wat de simulatie geen goed doet. Bij normale situaties wordt alleen rekening gehouden met modelruis, maar nauwelijks met ruis in de gemeten data. Een methode om hier wel rekening mee te houden is om gebruik te maken van Kalman filtering.

2 Model

Het model waar we mee gaan werken ziet er als volgt uit:

$$\begin{aligned}\mathbf{x}_{k+1} &= F\mathbf{x}_k + G\mathbf{w}_k \\ z_k &= M\mathbf{x}_k + v_k\end{aligned}\tag{1}$$

Hier zijn

$$F = \begin{pmatrix} a & 0,2 & b \\ c & 1,0 & d \\ 0,3 & e & f \end{pmatrix} \quad G = \begin{pmatrix} 1,0 \\ 0,5 \\ 0,3 \end{pmatrix} \quad M = (0,5 \quad 1,5 \quad 0,7)\tag{2}$$

en \mathbf{w}_k en v_k zijn gemodelleerde ruis met respectievelijk varianties Q en R .

Van het systeem wordt geëist dat het bestuurbaar en waarneembaar is. Waarneembaarheid garandeert dat de variantie in de toestand waarin het systeem zich bevindt kan worden waargenomen in de metingen. Bestuurbaarheid houdt in dat de modelruis \mathbf{w}_k invloed heeft op alle componenten van \mathbf{x}_k . Als we aan deze twee eigenschappen voldoen weten we dat ons systeem stabiel is en hiermee is ook het

Kalmanfilter dat we later gaan toepassen stabiel. Een derde eis die ons opgelegd is dat voor de eigenwaarden van F moet gelden dat $0,9 < \lambda_i < 1,0$. Het systeem zal dus een trage convergentie kennen.

Om aan deze laatste eis te voldoen hebben wij naar het karakteristiek polynoom gekeken van F . Na enig rekenwerk volgde dat deze gelijk is aan

$$-\lambda^3 + (1 + a + f)\lambda^2 - (a + f + af - 0,3b - 0,2c - de)\lambda + af + bce + 0,06d - 0,3b - ade - 0,2cf = 0 \quad (3)$$

Verder weten we dat voor de eigenwaarden λ_i moet gelden dat

$$\begin{aligned} (\lambda_1 - \lambda)(\lambda_2 - \lambda)(\lambda_3 - \lambda) &= 0 \\ -\lambda^3 + (\lambda_1 + \lambda_2 + \lambda_3)\lambda^2 - (\lambda_1\lambda_2 + \lambda_1\lambda_3 + \lambda_2\lambda_3)\lambda + \lambda_1\lambda_2\lambda_3 &= 0 \end{aligned} \quad (4)$$

Aangezien we weten dat de eigenwaarden moeten voldoen aan $0,9 < \lambda_i < 1,0$ legt dit de volgende 3 eisen op de parameters op:

$$\begin{aligned} 2,7 < 1 + a + f < 3 \\ 2,43 < a + f + af - 0,3b - 0,2c - de < 3 \\ 0,729 < af + bce + 0,06d - 0,3b - ade - 0,2cf < 1 \end{aligned} \quad (5)$$

Merk op dat dit noodzakelijke voorwaarden zijn, doch niet voldoende. Dit garandeerd namelijk niet dat het karakteristiek polynoom alleen reëelwaardige oplossingen geeft, wat wel noodzakelijk is voor ons probleem. Om analytisch te onderzoeken voor welke waarden dit het geval is zouden we naar de methode van Cardano kunnen kijken. Dit levert echter teveel onoverzichtelijke eisen op. We hebben besloten om binnen bovenstaande voorwaarden een aantal mogelijkheden te proberen. Dit leverde ons de volgende matrix F op:

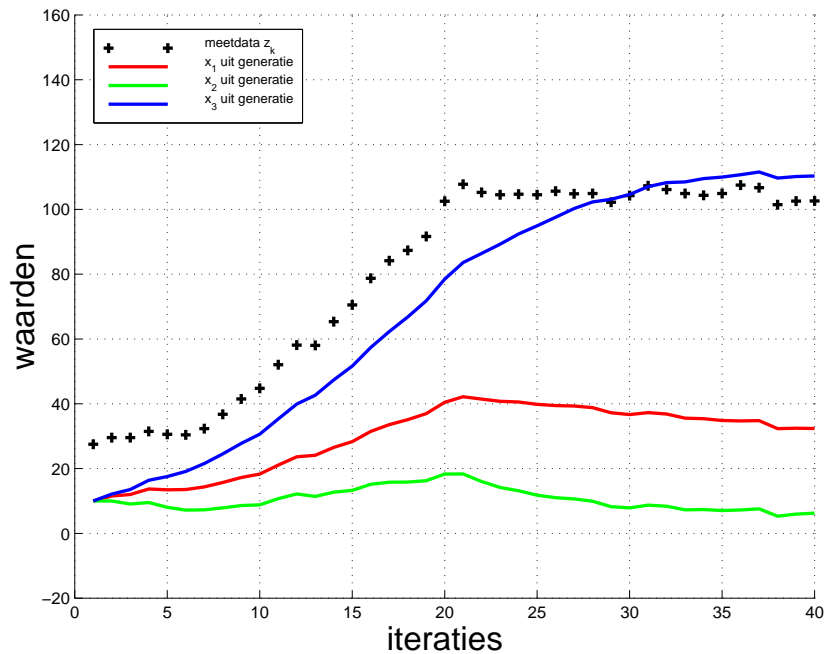
$$F = \begin{pmatrix} 0,95 & 0,2 & 0,0002 \\ -0,0003 & 1,0 & 0 \\ 0,3 & 0 & 0,91 \end{pmatrix} \quad (6)$$

Deze levert de eigenwaarden $\lambda_1 = 0,9086$, $\lambda_2 = 0,9527$ en $\lambda_3 = 0,9988$. Deze matrix maakt het systeem ook waarneembaar en bestuurbaar aangezien zowel de matrix $(G \quad FG \quad F(FG))$ als de matrix $(M^T \quad FM^T \quad F(FM^T))$ beide rang 3 hebben.

3 Generatie van de meetdata

Omdat we geen echte data hebben zijn we genoodzaakt om deze kunstmatig te genereren. Dit gebeurt aan de hand van systeem 1. Door een beginwaarde en ruis toe te

voegen proberen we de ruis aan de data toe te voegen die je in werkelijkheid zou hebben. Door een beginwaarde te kiezen en de modelruis mee te nemen kunnen wij data genereren die, net als echte meetdata, niet de exacte waarden van een ruisloos model weergeeft. We hebben gekozen voor varianties $Q = 1$ en $R = 0,7$ en als beginwaarde $\mathbf{x}_0 = (10 \ 10 \ 10)^T$. We krijgen dan het verloop zoals in figuur 1.

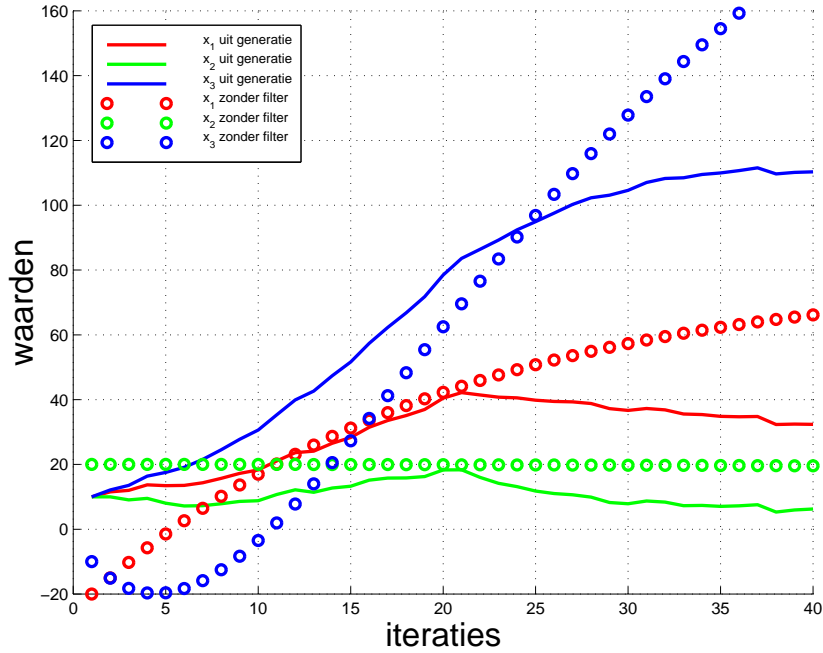


Figuur 1: Gegenerateerde data

4 Kalman filter

Het doel is om zonder bekendheid van de beginwaarde en de varianties van de ruis die we voor het genereren van de data in de vorige paragraaf gebruikt hebben de waarden in figuur 1 zo goed mogelijk te benaderen. Zouden we geen rekening houden met de ruis en een willekeurige (meestal foute) beginwaarde kiezen dan krijgen we het verloop te zien zoals in figuur 2. De door ons gekozen beginwaarde is $\tilde{\mathbf{x}}_0 = (-20 \ 20 \ -10)^T$. Deze gebruiken wij in het vervolg van het verslag. We zien dat zonder rekening te houden met de ruis en de meetdata het model de werkelijkheid niet benaderd.

Om dit wel te doen gebruiken we Kalmanfiltering. Deze filter houdt zowel rekening met de verwachte ruis alswel meetdata en de beginwaarde. De verwachte ruis beïnvloed het filter in de mate waarin het zich aanpast aan de hand van de meetdata



Figuur 2: Benadering zonder filter

z_k . Als je grote ruis verwacht zal het filter zich minder enthousiast aanpassen dan wanneer je weinig ruis verwacht.

De Kalmanfiltertechniek bestaat uit 2 stappen. Eerst wordt er een tijdsupdate gedaan. Deze is min of meer gelijk aan een stap doen zonder rekening te houden met ruis en dergelijke.

$$\tilde{\mathbf{x}}_{k+1} = F\mathbf{x}_k \quad (7)$$

Deze waarde wordt aan de hand van de verwachte ruis en de gemeten (bekende) data aangepast. Hiervoor berekenen we een nieuwe covariantiematrix P_k en Kalman gain K_k :

$$\begin{aligned} \tilde{P}_{k+1} &= F\tilde{P}_kF^T + GQG^T \\ K_{k+1} &= \tilde{P}_{k+1}M^T(M\tilde{P}_{k+1}M^T + R)^{-1} \end{aligned} \quad (8)$$

Daarna wordt de aanpassing aan \mathbf{x}_{k+1} gedaan met

$$\begin{aligned} \mathbf{x}_{k+1} &= \tilde{\mathbf{x}}_{k+1} + K_{k+1}(z_{k+1} - M\tilde{\mathbf{x}}_{k+1}) \\ P_{k+1} &= (I - K_{k+1}M)\tilde{P}_{k+1} \end{aligned} \quad (9)$$

Dit algoritme hebben we in Matlab geïmplementeerd zoals te vinden is in appendix A.4.

De beginwaarde $\tilde{\mathbf{x}}_0$ kan heel verkeerd of juist redelijk goed gekozen worden. Dit is afhankelijk van de bekendheid met het probleem. Met de covariantiematrix P_0 kan invloed uitgeoefend worden in welke mate de beginwaarde belangrijk of correct is. Als je denkt dicht in de buurt te zitten van de echte beginwaarde \mathbf{x}_0 dan is de eenheidsmatrix een goede keuze. Indien je denkt dat de beginwaarde erg verkeerd gekozen is dan moet P_0 vrij groot genomen worden.

5 Simulatieresultaten

Om gevoel te krijgen op welke manier deze parameters invloed hebben op de manier van convergeren hebben we een aantal simulaties gedraaid de parameters te vinden in tabel 1.

P_0	Q	R	figuur
I	1	0,7	3
$\begin{pmatrix} 5000000 & 0 & 0 \\ 0 & 15000000 & 0 \\ 0 & 0 & 7000000 \end{pmatrix}$	1	0,7	4
I	10	10	5
I	0,001	0,001	6
I	0,001	10	7
I	10	0,001	8

Tabel 1: Onderzochte parameters

Omdat niet alle verschillen erg duidelijk zijn hebben we een grafiek gemaakt van de fouten tussen de gefilterde oplossing en de echte oplossing. Om de uitwerking van bepaalde keuzen voor de filterparameters duidelijk te zien moet dus ook gekeken worden naar figuur 9 op pagina 9. Hierin is alleen gekeken naar de 3e component van \mathbf{x}_k omdat in deze component alle veranderingen het hardste doorwerken.

Als we rekening houden met de verwachte ruis en de meetdata in acht nemen (en dus de Kalmanfilter gebruiken) zien we dat de data netjes in de buurt blijft van de echte data, zie figuur 3. Na 25 iteraties zie je dat het model redelijk is geconvergeert naar de echte data. We zien dat de simulatie uiteindelijk dichterbij de echte data komt.

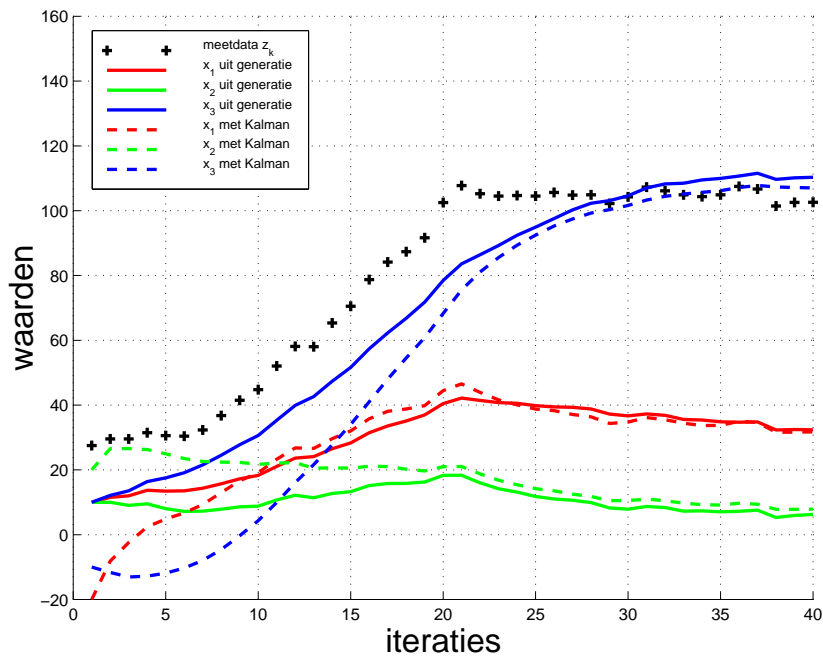
Als we voor een grote begincovariantiematrix P_0 kiezen zien we in de eerste 10 iteraties een paar grote slingers. De beginwaarde is van minder belang en het algoritme richt zich primair op de bekende meetdata, zie figuur 4.

Voor figuur 5 verwachtten we grote ruis, terwijl de werkelijke ruis veel kleiner was. We zien geen verschil met de andere figuren.

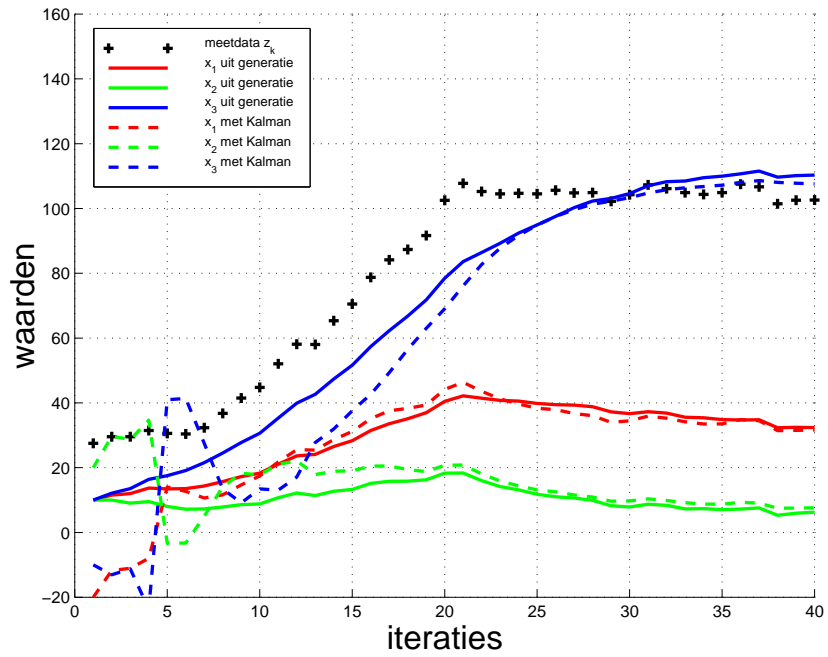
Wanneer er weinig ruis wordt verwacht is het verloop zoals in figuur 6. De gefilterde data past zich vrij snel aan aan de verwachting en de gemeten data. R is namelijk klein en daardoor is de Kalman gain (zie vergelijking 8) groter en daarmee de invloed van de meetdata.

Als we deze Kalman gain klein willen houden en dus alleen een grote ruis verwachten in de gemeten data (zie figuur 7) zien we, zij het in mindere mate, hetzelfde verloop als in figuur 2 waarin we geen rekening hielden met de gemeten data.

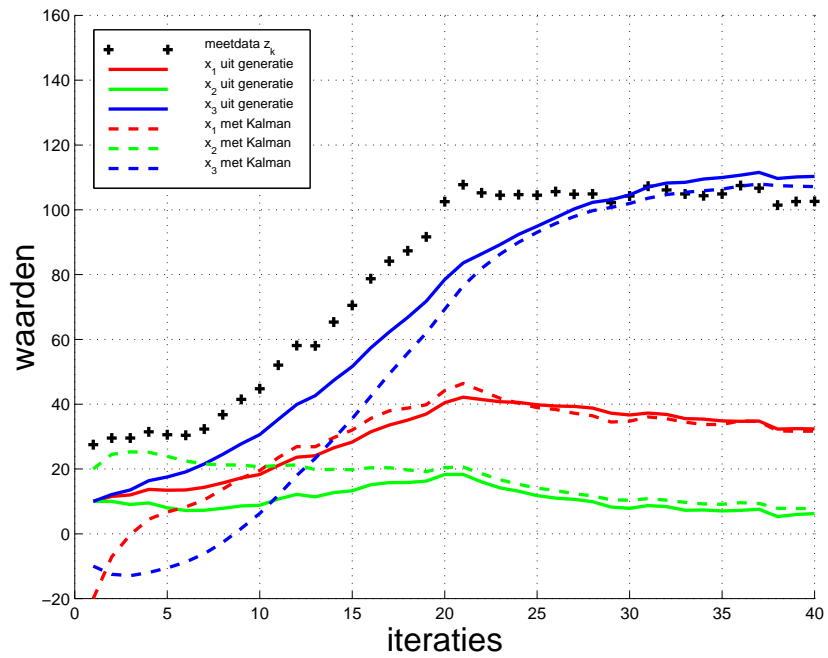
De laatste situatie die we bekeken hebben is wanneer we weinig ruis in de gemeten data verwachten en weinig waarde hechten aan de voorgaande stap. Een grote Q en kleine R zorgen er beide voor dat de Kalman gain groot wordt en de gemeten data een nog belangrijkere peiler wordt. Dit is te zien in figuur 8



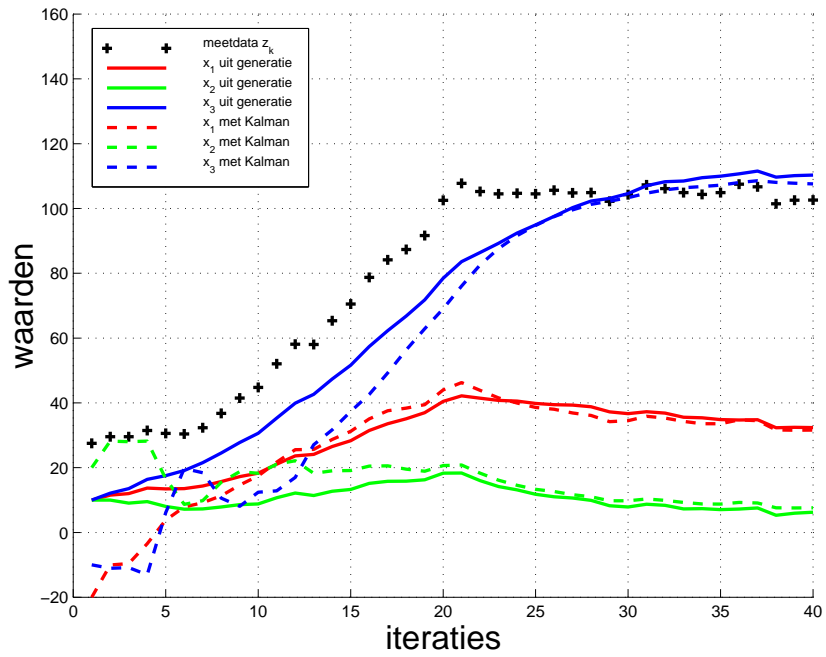
Figuur 3: Gewoon gefilterd met verwachte ruis die erin zit



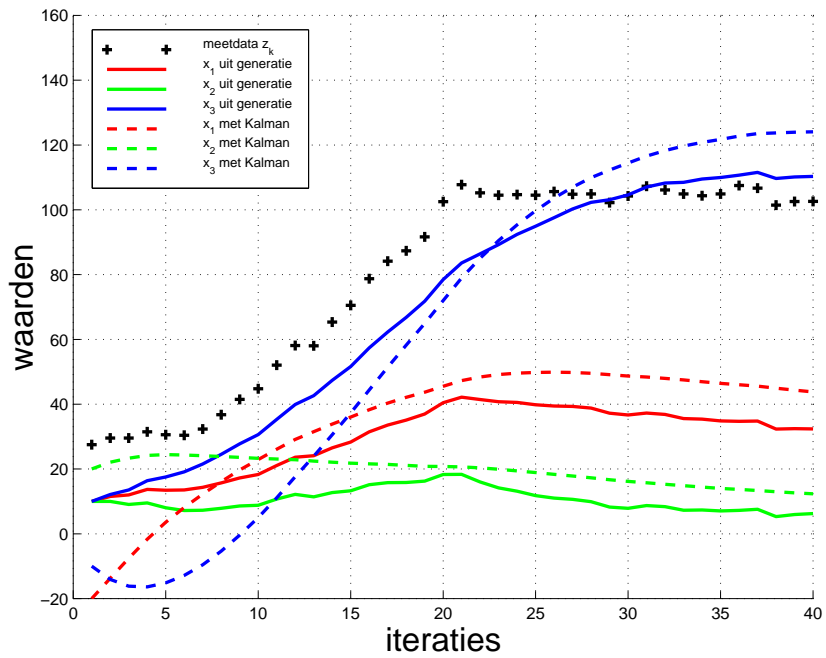
Figuur 4: De invloed van P_0



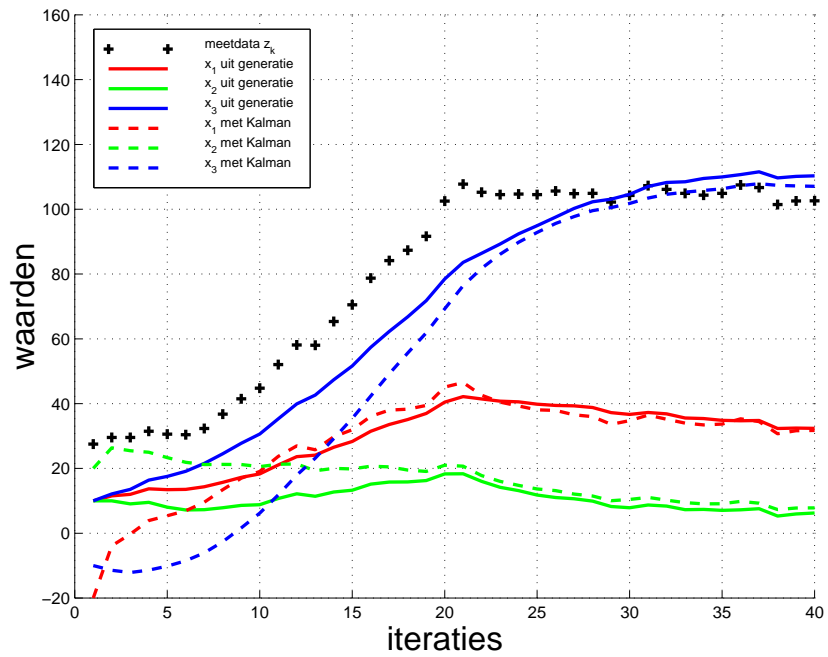
Figuur 5: Verwacht grote ruis in zowel meetdata als model



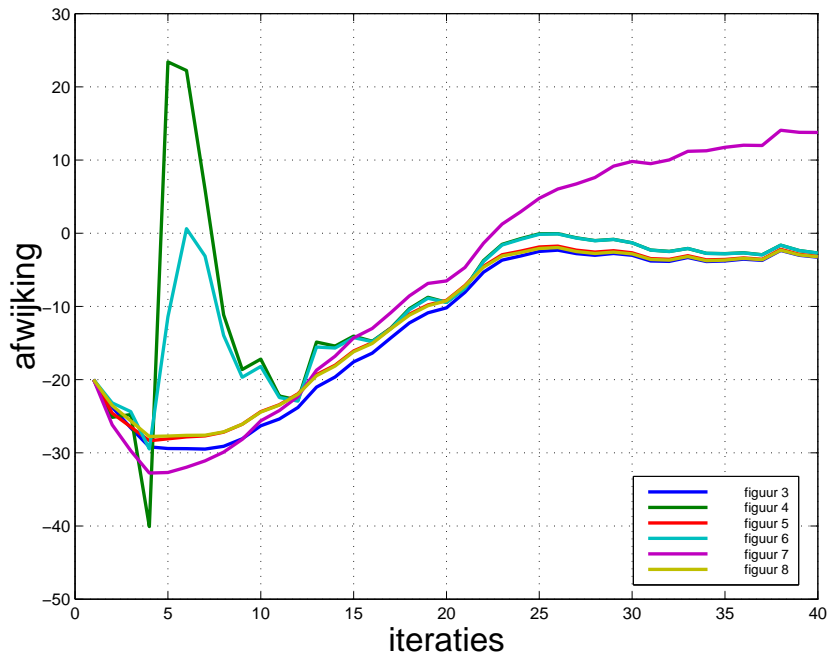
Figuur 6: Verwacht kleine ruis in zowel meetdata als model



Figuur 7: Verwacht veel ruis in de meetdata



Figuur 8: Verwacht veel ruis in het model



Figuur 9: Verschil tussen de echte oplossing en de verschillende gefilterde oplossingen met betrekking tot de 3e component van \mathbf{x}_k

6 Conclusie

We zien dat het gebruik van een filter onmisbaar is. Zonder filter is er geen leidraad waarnaar het iteratieve proces zich kan richten.

Drie punten die in acht genomen en goed moeten worden onderzocht voor een succesvol gebruik van de filter zijn:

- hoe goed is de beginvoorwaarde?
- wat is de te verwachte modelruis?
- hoe goed is de gemeten data?

Na onze experimenten zien we dat wanneer de eerste twee punten verkeerd worden verondersteld het effect op lange termijn nihil is. Goed gemeten data is, zoals ook te verwachten is, een noodzaak voor een goede benadering.

A Matlab sources

A.1 Vragen *a* en *b*

```
1 % onderdeel a: constructie en controle matrix F
2 a=0.95;
3 b=0.0002;
4 c=-0.0003;
5 d=0.000;
6 e=0.000;
7 f=0.91;
8
9 F=[a 0.2 b; c 1 d; 0.3 e f];
10
11 % eigenwaarden ok?
12 eig(F)
13
14 % maak waarneembaarheids- en bestuurbaarheidsmatrices
15 G = [1;0.5;0.3];
16 M = [0.5 1.5 0.7];
17
18 C= [G F*G F*(F*G)];
19 O= [M' F*M' F*(F*M')];
20
21 % rank=3?
22 rank(C)
23 rank(O)
24
25
26 % onderdeel b: genereren van data
27 K=40;
28 Q=1;
29 R=0.7;
30 w=sqrt(Q)*randn(1,K);
31 v=sqrt(R)*randn(1,K);
32 x=zeros(3,K);
33 z=zeros(1,K);
34
35 % de 'echte' beginwaarde
36 x(:,1) =[10;10;10];
37
38 for k=1:K-1
```

```

39     z(1,k)=M*x(:,k)+v(k);
40     x(:,k+1)=F*x(:,k)+w(k);
41     end
42     z(1,K)=M*x(:,K)+v(K);
43     clf
44
45     % voor plaatjes van onze gebruikte data
46     %load meet2.dat x z
47     %x=meet2(1:3,:);
48     %z=meet2(4,:);
49
50     % maak plaatje
51     hold on
52     set(plot([1:K],z,'k+'),'LineWidth',2)
53     set(plot([1:K],x(1,:),'r'),'LineWidth',2)
54     set(plot([1:K],x(2,:),'g'),'LineWidth',2)
55     set(plot([1:K],x(3,:),'b'),'LineWidth',2)
56     set(gcf,'Position',[0 0 760 570])
57     grid on
58     set(xlabel('iteraties'),'FontSize',20)
59     set(ylabel('waarden'),'FontSize',20)
60     set(gca,'YLim',[-20,160])
61     legend('meetdata\z_{k}','x_{1}\_uit\_generatie','x_{2}\_uit\_generatie','x_{3}\_uit\_generatie',2)
62     % title('Generatie van de data en meetwaarden')
63
64     shg
65
66     print -depsc2 pic01.eps
67
68     % bewaar de data
69     save meet3.dat x z - ascii
70
71     pause
72
73     % voer het model uit zonder filter (dus zonder rekening te houden
74     % met ruis en meetdata)
75     clf
76     xv=zeros(3,K);
77     xv(:,1)=bvw;
78     for k=1:K-1
79         xv(:,k+1)=F*xv(:,k);
80     end

```

```

81
82 % en maak ook hier weer een plaatje van
83 hold on
84 set(plot ([1: K],x (1,:) ', 'r'), 'LineWidth',2)
85 set(plot ([1: K],x (2,:) ', 'g'), 'LineWidth',2)
86 set(plot ([1: K],x (3,:) ', 'b'), 'LineWidth',2)
87 set(plot ([1: K],xv (1,:) ', 'or'), 'LineWidth',2)
88 set(plot ([1: K],xv (2,:) ', 'og'), 'LineWidth',2)
89 set(plot ([1: K],xv (3,:) ', 'ob'), 'LineWidth',2)
90 set(gcf, 'Position' ,[0 0 760 570])
91 legend('x_{1} uit generatie ', 'x_{2} uit generatie ', 'x_{3} uit generatie ',
        'x_{1} zonder filter ', 'x_{2} zonder filter ', 'x_{3} zonder filter ',2)
92 grid on
93 set(xlabel (' iteraties '), 'FontSize',20)
94 set(ylabel ('waarden'), 'FontSize',20)
95 set(gca, 'YLim',[-20,160])
96
97 print -depsc2 pic01a.eps

```

A.2 Vraag c en d

```

1 % vraag c
2 P=eye(3);
3 X=doekalman(P,1,0.7);
4 % title (' Gewoon gefilterd ');
5 shg
6 print -depsc2 pic02.eps
7
8
9 %vraag d, verschillende P, Q en Rs
10
11 pause
12 % stap 1: bekijk invloed van P
13 P =[0.5 0 0; 0 1.5 0; 0 0 0.7]*10000000;
14 X (2,:)=doekalman(P,1,0.7);
15 % title (' Grote P ');
16 print -depsc2 pic03.eps
17
18 pause
19 % stap 2: P=I, verwacht grote ruis
20 P=eye(3);
21 X (3,:)=doekalman(P,10,10);
22 % title (' Grote ruis ');

```

```

23 print –depsec2 pic04.eps
24
25 pause
26 % stap 3: P=I, verwacht kleine ruis
27 P=eye(3);
28 X(4,:)=doekalman(P,0.001,0.001);
29 % title (' Kleine ruis ')
30 print –depsec2 pic05.eps
31
32 pause
33 % stap 4: P=I, verwacht grote ruis in meetdata
34 P=eye(3);
35 X(5,:)=doekalman(P,0.001,10);
36 % title (' Grote ruis meetdata ')
37 print –depsec2 pic06.eps
38
39 pause
40 % stap 5: P=I, verwacht grote ruis in generatie
41 X(6,:)=doekalman(P,10,0.001);
42 % title (' Grote ruis generatie ')
43 print –depsec2 pic07.eps
44 pause
45 clf
46
47 set(plot(1:length(X),X(1,:),1:length(X),X(2,:),1:length(X),X(3,:),1:length
(X),X(4,:),1:length(X),X(5,:),1:length(X),X(6,:)), 'LineWidth',2)
48
49 set(gcf, ' Position ' ,[0 0 760 570])
50 legend(' figuur 3', ' figuur 4', ' figuur 5', ' figuur 6', ' figuur 7', ' figuur 8',4)
51 set(xlabel(' iteraties '), 'FontSize',20)
52 set(ylabel(' afwijking '), 'FontSize',20)
53 grid on
54 print –depsec2 fouten.eps

```

A.3 Initiator voor de filter

```

1 function X3=doekalman(P,Q,R)
2 clf
3
4 % maak matrices
5 a=0.95;
6 b=0.0002;
7 c=-0.0003;

```

```

8 d=0.000;
9 e=0.000;
10 f=0.91;
11
12 F=[a 0.2 b ; c 1 d ; 0.3 e f];
13 G =[1;0.5;0.3];
14 M =[0.5 1.5 0.7];
15
16 % laad de data in
17 load meet2.dat x z
18 x=meet2(1:3,:);
19 z=meet2(4,:);
20 K=length(z);
21
22 X3=zeros(1,K);
23
24 % vaste beginwaarde
25 bvw=[-20;20;-10];
26
27 % doe de Kalmanfilter
28 X=kalmanfilter(P,Q,R,K,bvw,F,M,G,z);
29
30 % en maak een mooi plaatje
31 hold on
32 set(plot ([1: K],z, 'k+'), 'LineWidth',2)
33 set(plot ([1: K],x (1,:) ', 'r'), 'LineWidth',2)
34 set(plot ([1: K],x (2,:) ', 'g'), 'LineWidth',2)
35 set(plot ([1: K],x (3,:) ', 'b'), 'LineWidth',2)
36 set(plot ([1: K],X (1,:) ', 'r--'), 'LineWidth',2)
37 set(plot ([1: K],X (2,:) ', 'g--'), 'LineWidth',2)
38 set(plot ([1: K],X (3,:) ', 'b--'), 'LineWidth',2)
39 set(gcf, 'Position', [0 0 760 570])
40 legend('meetdata_z_{k}', 'x_{1}_uit_generatie', 'x_{2}_uit_generatie', 'x_{3}_uit_generatie', 'x_{1}_met_Kalman', 'x_{2}_met_Kalman', 'x_{3}_met_Kalman',2)
41 grid on
42 set(xlabel(' iteraties '), 'FontSize',20)
43 set(ylabel(' waarden'), 'FontSize',20)
44 set(gca, 'YLim', [-20,160])
45
46 X3=X(3,:)-x(3,:);

```

A.4 Kalmanfilter algoritme

```
1 % Kalman filter functie met verschillende P, Q en R
2 % met K stappen beginnende bij X0 en matrices F, M, G
3 % en meetdata z
4
5 function X=kalmanfilter(P,Q,R,K,X0,F,M,G,z)
6
7 X=zeros(3,K);
8 Xer=zeros(3,K);
9 X(:,1)=X0;
10
11 for k=1:K-1
12     %time update
13     Xer(:,k+1)=F*X(:,k);
14     P=F*P*F'+G*Q*G';
15
16     %Kalman gain
17     Kg=P*M'*inv(M*P*M'+R);
18
19     %measurment update
20     X(:,k+1)=Xer(:,k+1)+Kg*(z(k+1)-M*Xer(:,k+1));
21     P=(eye(3)-Kg*M)*P;
22 end
```