On-line implementatie van een lineair optimaal filter

On-line implementatie van een lineair optimaal filter

Nathan van de Wouw

WFW nr. : 93.086

Inhoudsopgave

Hoofdstuk 1

Filtertheorie

1.1 Inleiding

Het doel van systeemidentificatie is het vinden van een wiskundig model voor het systeem. De daarvoor benodigde informatie wordt uit metingen gehaald. Verder is een uitgangspunt dat er al een modelstructuur voorhanden is, bijv. verkregen uit toepassing van fysische wetten.

Het model bevat variabelen, die tijdsafhankelijk zijn, en parameters, die constant zijn. De bewegingsvergelijkingen,die deel uitmaken van de modelvergelijkingen, zijn vaak sterk niet-lineair in de modelvariabelen en lineair in de parameters. Echter, uit oogpunt van identificatie wordt de mate van niet-lineariteit nog verhoogd door de aanwezigheid van gecombineerde termen van modelvariabelen en parameters.

1.2 Het schattingsmodel

We gaan hier uit van een vorm van het schattingsmodel zoals die geïntroduceerd wordt in [1] Molengraft (1990). Hierbij gaan we ook meteen uit van de uitgebreide toestandsvorm. Door het gebruik van de uitgebreide toestandsvorm kan het schattingsmodel compacter beschreven worden.
De schatting van de uitgebreide toestand, $\hat{x}$, en de schatting van de uitgebreide ingang, $\hat{p}$, worden als volgt gedefiniëerd.

$$\hat{x} = \begin{bmatrix} \hat{s} \\ \hat{v} \\ \hat{\theta} \end{bmatrix} \quad ; \quad \hat{p} = \begin{bmatrix} \hat{u} \\ \hat{a} \end{bmatrix} \tag{1.1}$$

Waarbij :

$\hat{\mathbf{s}}$	n	kolom van graden van vrijheid (posities)
$\hat{\mathbf{v}}$	n	kolom, eerste afgeleide van $\mathbf{s}$ (snelheden)
$\hat{\mathbf{a}}$	n	kolom, eerste afgeleide van $\mathbf{v}$ (versnellingen)
$\mathbf{u}$	m	kolom met regelbare externe belastingen
θ	p	kolom met onbekende, constante parameters

Het in [1] Molengraft (1990) geïntroduceerde uitgebreide schattingsmodel ziet er nu als volgt uit:

$$\dot{\hat{x}} = A\hat{x} + B\hat{p} + \xi \tag{1.2}$$

$$Ez_m = f(\hat{x},t) + F(\hat{x},t)\hat{p} + \zeta \tag{1.3}$$

waarin geldt:

$$A = \begin{bmatrix} O & I & O \\ O & O & O \\ O & O & O \end{bmatrix} \; ; \; B = \begin{bmatrix} O & O \\ O & I \\ O & O \end{bmatrix} \; ; \; \xi = \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{bmatrix} \tag{1.4}$$

$$E = \begin{bmatrix} O \\ I \end{bmatrix} \; ; \; f(\hat{x},t) = \begin{bmatrix} h(\hat{x},t) \\ g(\hat{x},t) \end{bmatrix} \tag{1.5}$$

$$F(\hat{x},t) = \begin{bmatrix} H(\hat{x},t) & M(\hat{x},t) \\ R(\hat{x},t) & G(\hat{x},t) \end{bmatrix} \; ; \; \zeta = \begin{bmatrix} \zeta_1 \\ \zeta_2 \end{bmatrix} \tag{1.6}$$

Hierin stellen ξ en ζ de residuen en z_m de metingen voor.
Het probleem is nu het bepalen van zodanige $\hat{x}$ en $\hat{p}$ dat ξ en ζ geminimaliseerd worden.
De bovenstaande aanpak heeft twee belangrijke verschillen met methoden uit de literatuur.

- De ingangen **u** worden niet als exact gegeven beschouwd, maar worden hetzelfde behandeld als de rest van de modelvariabelen.
- De versnellingen **a** kunnen niet alleen uit de bewegingsvergelijking(en) bepaald worden, maar ook direct uit de versnellingsmetingen.

Opmerking Het feit dat er ook residuen zitten op de koppelingsvergelijkingen :

$$\dot{\hat{s}} = \hat{v} + \xi_1 \tag{1.7}$$

$$\dot{\hat{v}} = \hat{a} + \xi_2 \tag{1.8}$$

houdt in dat er een ontkoppeling (tot op zekere hoogte) mogelijk is tussen de snelheid en de verplaatsing en tussen de snelheid en de versnelling. Dit kan wenselijk zijn wanneer bijv. de snelheidsschatting al goed is en de versnellingsschatting nog niet. A.g.v. de eventuele ontkoppeling kan dan de versnellingsschatting verbeterd worden zonder dat de snelheidsschatting veel aangepast hoeft te worden.

1.3 Linearisatie

Voordat de **lineaire** optimale filtertheorie, beschreven in [1] Molengraft (1990), toegepast kan worden, zal het (niet-lineaire) systeem uiteraard gelineariseerd moeten worden.
De linearisatie moet plaatsvinden rond nominale waarden van de grootheden, $x = x_n$, $p = p_n$ en $z = z_n$. Laat verder δx , δp en δz de perturbaties van respectievelijk x, p en z rond x_n , p_n en z_n zijn.
Dus:

$$x = x_n + \delta x \tag{1.9}$$

$$p = p_n + \delta p \tag{1.10}$$

$$z = z_n + \delta z \tag{1.11}$$

De volgende twee vergelijkingen moeten nu gelineariseerd worden:

$$\dot{x} = Ax + Bp \tag{1.12}$$

$$Ez_m = f(x,t) + F(x,t)p \tag{1.13}$$

Er geldt:

$$f(x,t) = f(x_n,t) + \frac{\partial f(x_n,t)}{\partial x}\delta x \tag{1.14}$$

$$F(x,t) = F(x_n,t) + \frac{\partial F(x_n,t)}{\partial x}\delta x \tag{1.15}$$

De vergelijkingen (1.12) en (1.13) gaan hiermee over in :

$$(\dot{x}_n + \delta\dot{x}) = A(x_n + \delta x) + B(p_n + \delta p) \tag{1.16}$$

$$E(z_n + \delta z) = f(x_n,t) + \frac{\partial f(x_n,t)}{\partial x}\delta x \tag{1.17}$$

$$+[F(x_n,t) + \frac{\partial F(x_n,t)}{\partial x}\delta x](p_n + \delta p) \tag{1.18}$$

Uiteraard moeten de vergelijkingen (1.12) en (1.13) exact gelden voor de nominale waarden :

$$\dot{x}_n = Ax_n + Bp_n \tag{1.19}$$

$$Ez_n = f(x_n,t) + F(x_n,t)p_n \tag{1.20}$$

Uit het voorgaande volgen nu de voor de perturbaties geldende vergelijkingen :

$$\delta\dot{x} = A\delta x + B\delta p \tag{1.21}$$

$$E\delta z = [\frac{\partial f(x_n,t)}{\partial x} + \frac{\partial F(x_n,t)}{\partial x}p_n]\delta x + F(x_n,t)\delta p \tag{1.22}$$

Opmerking De tweede (en hogere) orde termen in de perturbaties zijn buiten beschouwing gelaten.

We hanteren nu de volgende naamgeving :

$$C = [\frac{\partial f(x_n,t)}{\partial x} + \frac{\partial F(x_n,t)}{\partial x}p_n] \tag{1.23}$$

$$D = F(x_n,t) \tag{1.24}$$

Het gelineariseerde systeemmodel ziet er nu als volgt uit :

$$\delta\dot{x} = A\delta x + B\delta p \tag{1.25}$$

$$E\delta z = C\delta x + D\delta p \tag{1.26}$$

Opmerking De specifieke keuzes van de nominale waardes x_n, p_n en z_n komen in de volgende paragraaf aan de orde.

1.4 Het lineaire optimale filter

Het linaire optimale filter zoals beschreven in Molengraft (1990) bestaat o.a. uit de volgende twee filtervergelijkingen :

$$\dot{\hat{x}} = A\hat{x} + B\hat{p} + QC^TV[Ez - C\hat{x} - D\hat{p}] \tag{1.27}$$

$$\hat{p} = [D^TVD]^{-1}D^TV[Ez - C\hat{x}] \tag{1.28}$$

met Q de oplossing van de volgende matrix Riccati vergelijking :

$$\dot{Q} = L_{12} + L_{11}Q + QL^T{}_{11} - QL_{21}Q \tag{1.29}$$

met :

$$\begin{aligned} L_{11} &= A - B(D^TVD)^{-1}(D^TVC) \\ L_{12} &= B(D^TVD)^{-1}B^T + W^{-1} \\ L_{21} &= C^TVC - (C^TVD)(D^TVD)^{-1}(D^TVC) \end{aligned}$$

Uiteraard horen bij de vergelijkingen (1.27) en (1.29) geschikte begincondities.
Deze filtervergelijkingen gelden voor een lineair (of gelineariseerd) systeem : zie vergelijkingen (1.25) en (1.26).
Wanneer we nu uitgaan van een gelineariseerd systeem met perturbaties $\delta x, \delta p$ en δz dan zien de filtervergelijkingen er als volgt uit :

$$\delta\dot{x} = A\delta x + B\delta p + QC^TV[E\delta z - C\delta x - D\delta p] \tag{1.30}$$

$$\delta p = [D^TVD]^{-1}D^TV[E\delta z - C\delta x] \tag{1.31}$$

Verder geldt :

$$\dot{x}_n = Ax_n + Bp_n \tag{1.32}$$

Sommeer nu de vergelijkingen (1.30) en (1.32).
Dit resulteert in :

$$\dot{\hat{x}} = A\hat{x} + B\hat{p} + QC^TV[E\delta z - C\delta x - D\delta p] \tag{1.33}$$

Nu moet er nog een keuze gemaakt worden voor de nominale waardes x_n, p_n en z_n. Voor de nominale toestand x_n, waar om gelineariseerd wordt, kiezen we nu de momentane geschatte toestand. Dit heeft als gevolg dat de perturbatie van x nul is :

$$x_n = \hat{x} \Rightarrow \delta x = 0 \tag{1.34}$$

De filtervergelijkingen worden nu :

$$\dot{\hat{x}} = A\hat{x} + B\hat{p} + QC^TV[E(z - z_n) - D\delta p] \tag{1.35}$$

$$\delta p = [D^TVD]^{-1}D^TVE(z - z_n) \tag{1.36}$$

$$\dot{Q} = L_{12} + L_{11}Q + QL^T{}_{11} - QL_{21}Q \tag{1.37}$$

Nu moeten er nog keuzes gemaakt worden voor p_n en z_n.
Gebruik nu dat de bewegingsvergelijkigen en de meetvergelijkingen exact moeten gelden voor de nominale waardes :

$$Ez_n = f(x_n, t) + F(x_n, t)p_n \tag{1.38}$$

Nu er een keuze gemaakt is voor x_n kunnen geschikte p_n en z_n, d.m.v. een iteratief proces, m.b.v. (1.38) worden bepaalt.
Het iteratieve proces ziet er als volgt uit :

$$z_{n0} = (E^T E^{-1}) E^T [f(x_n, t) + F(x_n, t) p_{n0}] \quad (1.39)$$

$$p_{n0} = (F^T F^{-1}) F^T [E z_{n0} - f(x_n, t)] \quad (1.40)$$

$$z_{n1} = \ldots \quad (1.41)$$

Dit moet dan doorlopen worden totdat een bepaalde mate van convergentie optreedt. In de praktijk blijkt convergentie snel (vaak binnen 2 of 3 iteraties) op te treden. De fout die hier gemaakt wordt mag in ieder geval niet beperkend zijn voor de nauwkeurigheid.
Voor p_{n0} moet nu nog een keuze worden gemaakt. Als beginwaarde p_{n0} kan de vorige schatting voor p: $\hat{p}$ gekozen worden.
Nu kunnen dus nieuwe schattingen voor **x** en **p** (en **Q**) bepaalt worden door de 2 differentiaalvergelijkingen simultaan op te lossen én $\hat{p} = p_n + \delta p$ te berekenen. Voor het oplossen van de differentiaalvergelijkingen is uiteraard een integratie-procedure nodig. In het geval van een on-line implementatie van het filter zal dit dan een discrete integratieregel moeten zijn. Dit is dan ook het onderwerp van de volgende paragraaf.

1.5 De integratiemethode

In deze paragraaf zal een integratiemethode besproken worden, die de integratie van de vergelijkingen (1.35) en (1.37) zal verzorgen. De basis van de integratiemethode is een Euler schema. Hieraan is nog een correctie toegevoegd in de vorm van een tweede orde Runge-Kutta schema

Uitleg Stel er geldt $t_i < t_0 < t_{i+1}$ en $t_{i+1} - t_i = \Delta t$. De meting $z_i = z(t_i)$ is op het tijdstip t_0 dus al beschikbaar en z_{i+1} nog niet. Verder hebben we ook al schattingen voor x_i en p_i. Het doel is nu het verkrijgen van schattingen voor x_{i+1} en p_{i+1}.Welke grootheden beschikbaar moeten zijn voor het tijdstip t_{i+1} is uiteraard afhankelijk van welke grootheden in de gebruikte regelwet teruggekoppeld worden. We gaan er hier even van uit dat er in de regeling geen versnellings- en ingangsterugkoppeling plaatsvindt. Dit houdt dus in dat alleen $\hat{x}_{i+1}$ (en niet $\hat{p}_{i+1}$) beschikbaar moet zijn voor het tijdstip t_{i+1}. Nu bepalen we een schatting voor x_{i+1} d.m.v. een Eulerstap (zie figuur 1) :

$$\hat{x}_{i+1} = \hat{x}_i + \Delta t \, f_i \quad (1.42)$$

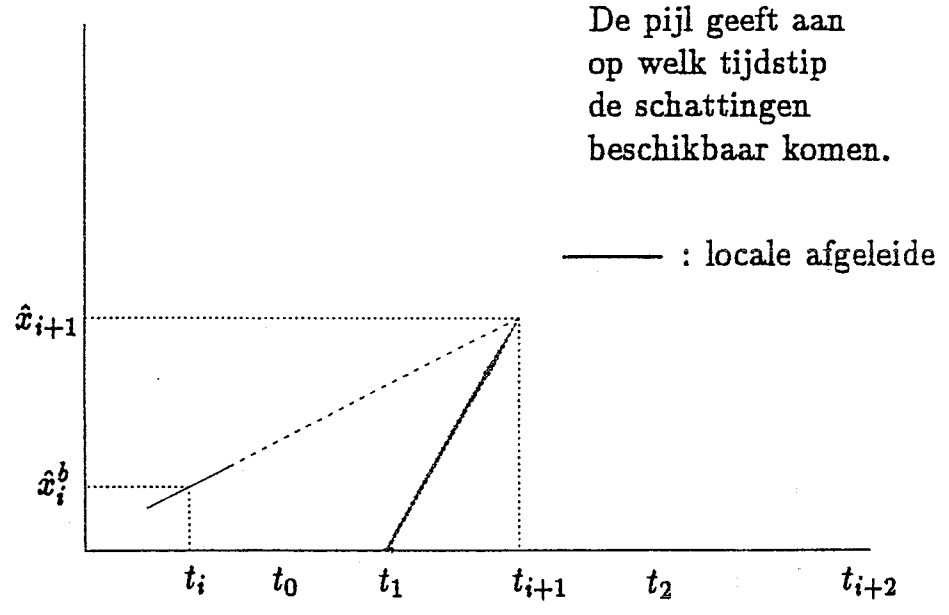
waarbij f_i de afgeleide van x op het tijdstip t_i is.
Uiteraard wordt binnen de integratie-procedure ook p_{i+1} bepaald (deze is ook nodig om f_i te bepalen, zie vergelijking (1.35)). Echter deze schatting van p_{i+1} is nog gebaseerd op een linearisatie om $\hat{x}_i$ en niet om $\hat{x}_{i+1}$.(Deze $\hat{p}_{i+1}$ is eigenlijk meer een schatting voor p_i). Later wordt nog een betere schatting voor p_{i+1} bepaald, waarbij dan gebruik gemaakt wordt van z_{i+1} en van een linearisatie om de met de Eulerstap bepaalde $\hat{x}_{i+1}$. De berekening van $\hat{x}_{i+1}$ neemt $(t_1 - t_0)$ seconden in beslag, dus $\hat{x}_{i+1}$ is op $t = t_1$ beschikbaar. Op dat moment kan de regelwet toegepast worden. Het is nu afhankelijk van de regelwet of al op t_1 of pas op t_{i+1} de berekening van u_{i+1} kan starten. We nemen nu aan dat de meting z_{i+1} gebruikt wordt in de regelwet. Op het tijdstip t_{i+1} kan de berekening van u_{i+1} dan beginnen. Wanneer u_{i+1} nu op

t_2 beschikbaar is dan komt deze $(t_2 - t_{i+1})$ te laat beschikbaar. We nemen hier aan dat de berekening van u_{i+1} betrekkelijk eenvoudig is, zodat $(t_2 - t_{i+1})$ klein is t.o.v. Δt. De nieuwe ingang wordt dus maar net te laat toegepast. Het grootste gedeelte van Δt wordt gebruikt voor het oplossen van de filter-vergelijkingen.

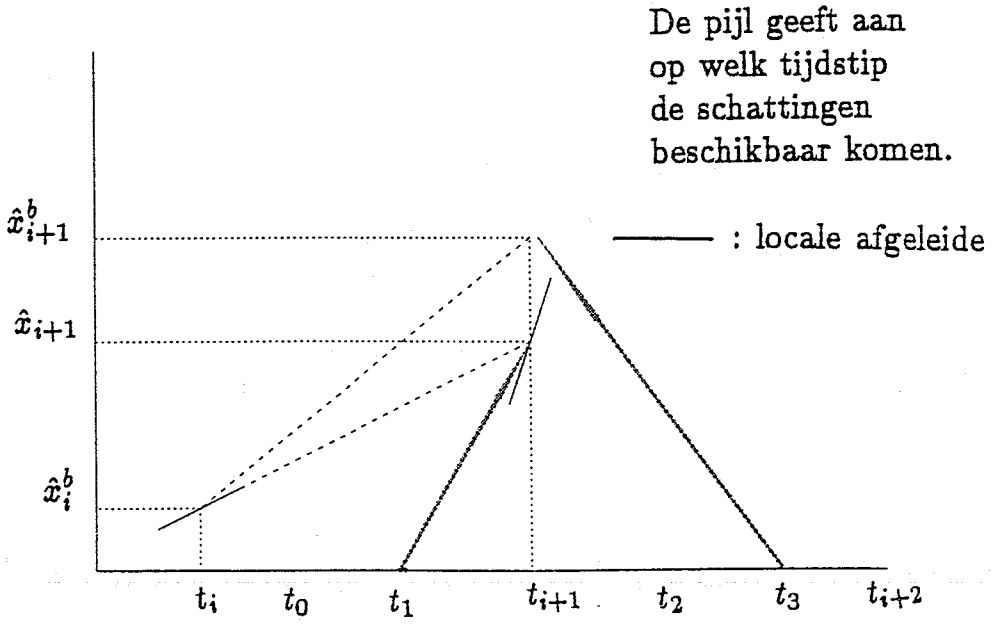
Nu kan op t_2, omdat nu ook z_{i+1} bekend is, alsnog (bij wijze van correctie) een tweede orde Runge-Kutta schema gebruikt worden om een betere schatting $\hat{x}^b_{i+1}$ te bepalen. Deze $\hat{x}^b_{i+1}$ wordt dan wel niet direct in de regelwet gebruikt, maar dient wel als een (verbeterd) beginpunt voor de volgende Eulerstap. Zo wordt de nauwkeurigheid van de integratie verhoogd.
Nu wordt $\hat{x}^b_{i+1}$ als volgt bepaald :

$$\hat{x}^b_{i+1} = \hat{x}_i + \frac{\Delta t}{2}(f_i + f_{i+1}) \tag{1.43}$$

waarbij f_{i+1} de afgeleide van x op het tijdstip t_{i+1} is.
Deze berekening neemt $(t_3 - t_2)$ seconden in beslag. Dus op t_3 komt $\hat{x}^b_{i+1}$ beschikbaar (zie figuur 2). Tijdens de berekening van f_{i+1} wordt ook een nieuwe, goede schatting voor p_{i+1}, $\hat{p}^b_{i+1}$, bepaald, die wel gebaseerd is op de meting z_{i+1} en op een linearisatie om $\hat{x}_{i+1}$. Als uiteindelijke schattingen van x en p op t_i gebruiken we nu $\hat{x}_i$ en $\hat{p}^b_i$: Dus $\hat{p}^b_i$ en niet $\hat{p}_i$ omdat $\hat{p}^b_i$ echt bij $\hat{x}_i$ hoort.
Vanaf t_3 kan nu weer dezelfde cyclus starten als vanaf t_0.



Figuur 1.1: De Eulerstap



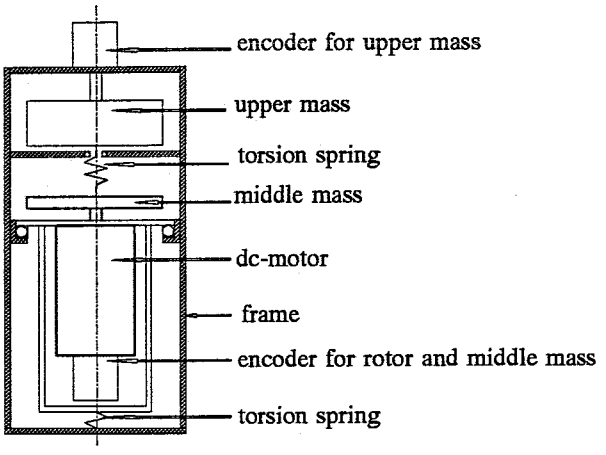
Figuur 1.2: De Runge-Kuttastap

Hoofdstuk 2

Toepassing v/d filtertheorie

2.1 Het experimentele systeem

Het systeem is een manipulator met 3 graden van vrijheid, zie figuur 2.1. Het doel is nu



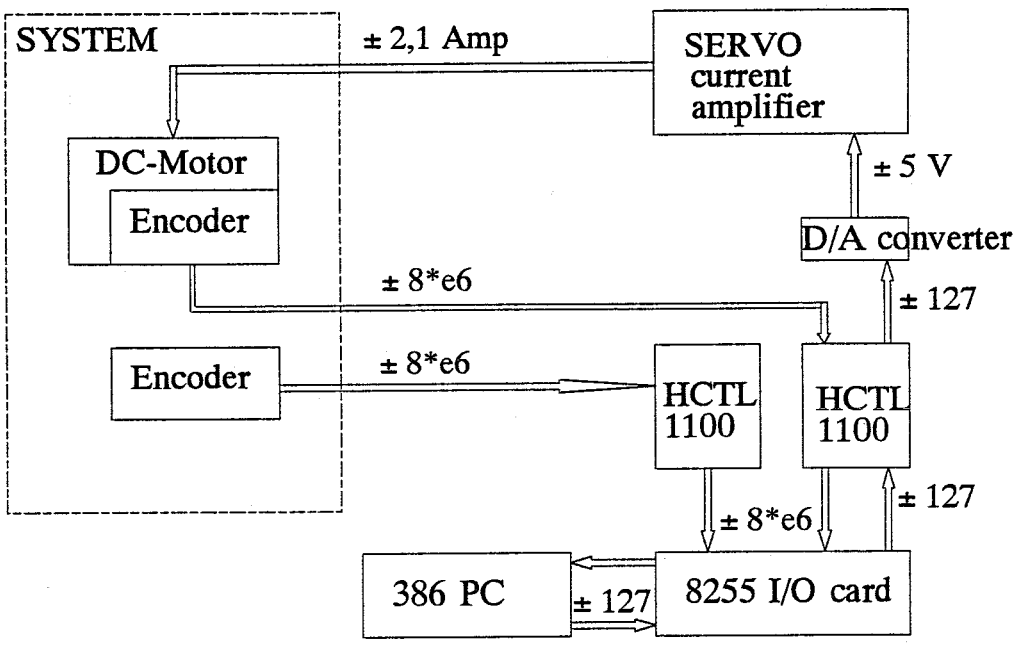
Figuur 2.1: Het experimentele systeem

het reconstrueren van de toestand van het systeem, bestaande uit 2 posities, 2 snelheden en eventueel nog enige parameters. De verkregen schatting van de toestand kan dan gebruikt worden in een regeling. De regeling op zich is hier echter niet het onderwerp van onderzoek. De middelste massa bestaat uit een schijf verbonden met de rotor van de gelijkstroom-motor. De stator en de buis zijn middels een veer verbonden met het frame van het systeem. Deze veer kan vervangen worden door een starre verbinding. Bij het modelleren van het systeem wordt de verbinding star verondersteld. De hoofdkarakteristieken van het systeem zijn als volgt :

- Een permanent magnetische gelijkstroom-motor.
- Een stroomversterker.
- 2 optische hoek-encoders, die incrementeel de positie van de bovenste massa en de rotorpositie meten .

- 2 veren met verstelbare stijfheden.
- De systeemidentificatie (en de regeling) worden gerealiseerd m.b.v. een microcomputer.

In figuur 2.2 is een schema van de hardware afgebeeld.



Figuur 2.2: Schema van de hardware-schakeling

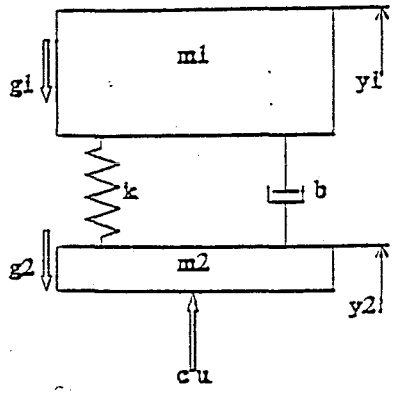
2.2 Het systeemmodel

Het systeem wordt gemodelleerd als een schakeling van twee massa's, een veer en een demper. De verbinding tussen de stator en het frame wordt star verondersteld. In figuur (2.3) is het systeemmodel schematisch weergegeven. Hierin stellen de symbolen het volgende voor:

cu: het motorkoppel
$\mathbf{g_1}, \mathbf{g_2}$: verstoringskoppels a.g.v. wrijving
$\mathbf{m_1}, \mathbf{m_2}$: respectievelijk de massa van de bovenste en de onderste massa
b : demperconstante
k : veerconstante
$\mathbf{y_1}, \mathbf{y_2}$: respectievelijk de verplaatsing van de bovenste en de onderste massa

De bewegingsvergelijkingen behorende bij dit model zijn nu :

$$m_1 \ddot{y}_1 = F_k + F_b - g_1 \tag{2.1}$$



Figuur 2.3: Het systeemmodel

$$m_2\ddot{y}_2 = -F_k - F_b - g_1 + F_m \tag{2.2}$$

waarin geldt :

$$F_k = k(y_2 - y_1) \tag{2.3}$$

$$F_b = b(\dot{y}_2 - \dot{y}_1) \tag{2.4}$$

$$F_m = cu \tag{2.5}$$

In vergelijking (2.5) stelt u de input voor en c een factor die de motorconstante, de versterkingsfactor van de versterker en de overdrachtsfactoren van de andere hardware-componenten in rekening brengt. Omdat c niet gemakkelijk te schatten is worden de vergelijkingen (2.1) en (2.2) met c geschaald. De modelvergelijkingen komen er dan als volgt uit te zien.

$$\frac{m_1}{c}\ddot{y}_1 = \frac{k}{c}(y_2 - y_1) + \frac{b}{c}(\dot{y}_2 - \dot{y}_1) - \frac{g_1}{c} \tag{2.6}$$

$$\frac{m_2}{c}\ddot{y}_2 = \frac{k}{c}(y_1 - y_2) + \frac{b}{c}(\dot{y}_1 - \dot{y}_2) - \frac{g_2}{c} + u \tag{2.7}$$

Nu moet het in hoofdstuk 1 besproken filter toegepast worden op het systeem.

Opmerking Voorlopig wordt in het model wrijving niet meegenomen. In de vergelijkingen (2.6) en (2.7) vallen dan de termen $\frac{g_1}{c}$ en $\frac{g_2}{c}$ weg.

Voor de uitgebreide toestand x, de uitgebreide ingang p en de metingen z_m is de volgende keuze gemaakt:

$$x = \begin{bmatrix} y_1 \\ y_2 \\ \dot{y}_1 \\ \dot{y}_2 \\ \frac{k}{c} \end{bmatrix} \quad ; \quad p = \begin{bmatrix} u \\ \ddot{y}_1 \\ \ddot{y}_2 \end{bmatrix} \quad ; \quad z_m = \begin{bmatrix} y_1 \\ y_2 \\ u \end{bmatrix} \tag{2.8}$$

De matrices **A,B,E,f,F** in de vergelijkingen (1.2) t/m (1.3) zien er voor dit concrete systeem als volgt uit:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad ; \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.9}$$

$$E = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

$$f = \begin{bmatrix} x_5(x_1 - x_2) + \frac{b}{c}(x_3 - x_4) \\ x_5(x_2 - x_1) + \frac{b}{c}(x_4 - x_3) \\ x_1 \\ x_2 \\ 0 \end{bmatrix} \quad ; \quad F = \begin{bmatrix} 0 & \frac{m_1}{c} & 0 \\ 1 & 0 & \frac{m_2}{c} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{2.11}$$

waarin x_i de i_{de} component uit de uitgebreide toestandsvector x is.
Na linearisatie van dit uit identificatie-oogpunt niet-lineaire systeem verkrijgen we de matrices **C** en **D** (zie vergelijkingen (1.23) t/m (1.26)) :

$$C = \begin{bmatrix} \hat{x}_5 & -\hat{x}_5 & \frac{b}{c} & -\frac{b}{c} & (\hat{x}_1 - \hat{x}_2) \\ -\hat{x}_5 & \hat{x}_5 & -\frac{b}{c} & \frac{b}{c} & (\hat{x}_2 - \hat{x}_1) \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad ; \quad D = F \tag{2.12}$$

Opmerking In **C** staan geschatte grootheden, omdat in paragraaf 1.4 de keuze is gemaakt om te lineariseren om de geschatte toestand.

Met deze informatie kunnen nu de filtervergelijkingen (1.35) t/m (1.37) toegepast worden op het systeem. Uiteraard zal er dan nog wel eerst een keuze voor de weegmatrices **V** en **W** en voor de beginschattingen $\hat{\mathbf{Q}}_0$ en $\hat{\mathbf{x}}_0$ gemaakt moeten worden.

2.3 Een simulatie

2.3.1 Inleiding

Alvorens onze aandacht te richten op een on-line implementatie van het lineaire, optimale filter op het in de voorgaande paragrafen besproken systeem wordt, bij wijze van een test, eerst nog een simulatie uitgevoerd. In deze simulatie is, terwille van de eenvoud, de wrijving buiten beschouwing gelaten (net als in de vorige paragraaf). Deze simulatie is uitgevoerd met het pakket Matlab.

2.3.2 Resultaten van de simulatie

Het Matlab-programma, dat de simulatie verzorgt is te vinden in de bijlagen 1a t/m 1c. Het programma bestaat uit 3 delen:

- Het hoofdprogramma (bijlage 1a)
- Een subroutine, waarin **p** en de afgeleides van **x** en **Q** berekend worden (bijlage 1b)
- Een subroutine, waarin gegevens t.a.v. de generatie van de metingen berekend worden (bijlage 1c)

Opmerking In de simulatie is bij het genereren van de metingen ook uitgegaan van een systeem zonder wrijving. In het filter worden dus exacte bewegingsvergelijkingen gebruikt.

Nu nog even iets over de keuze van de weegmatrices en de begincondities:

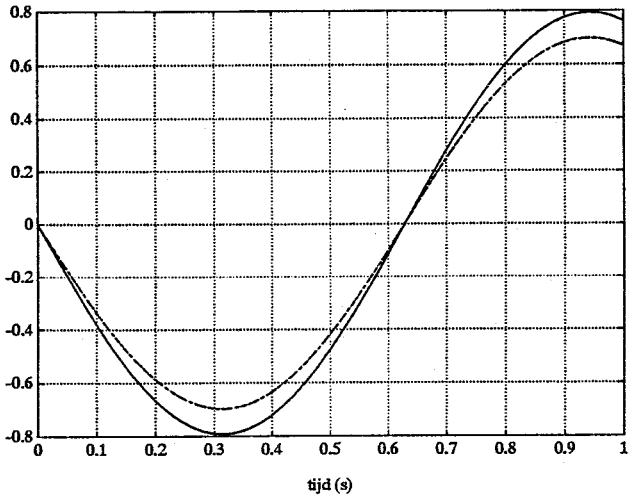
- De begincondities: Voor de verplaatsingen en de snelheden zijn exacte begincondities genomen (dit is in het geval van een simulatie nl. mogelijk). Voor de op **c** geschaalde stijfheidsparameter $k^* = \frac{k}{c}$ is een slechtere beginschatting genomen ($\hat{k}_0 = 900\ [\frac{Nmrad^{-1}}{Nm}]$ terwijl voor de exacte k^* geldt: $k^* = 978[\frac{Nmrad^{-1}}{Nm}]$) om te kijken of de parameter toch goed geschat wordt door het filter. In de beginschatting van **Q** is alleen het element, dat correspondeert met de betrouwbaarheid van de beginschatting van **k** ongelijk 0. Dit element is groot genomen om het mogelijk te maken dat het filter $\hat{k}$ in de eerste stappen veel aanpast.
- De weegmatrices: Bij de keuze van de weegmatrices is als uitgangspunt genomen dat er veel waarde gehecht moet worden aan de bewegingsvergelijkingen (die waren immers exact) en aan de metingen. Ook wordt er, zij het in iets mindere mate, veel waarde gehecht aan de koppelingsvergelijkingen (1.7) en (1.8). Er wordt echter veel vrijheid gegeven aan de schatting van **k**, omdat we in principe niet weten hoe goed of slecht de beginschatting $\hat{k}_0$ is. Verder zijn bij herhaaldelijk draaien van de simulatie **V**, **W** en $\hat{\mathbf{Q}}_0$ nog wat aangepast teneinde de residuen te minimaliseren. De uiteindelijke weegmatrices en begincondities zijn te vinden in het Matlab-hoofdprogramma (bijlage 1a).

De reconstructies van **x** en **p** zijn grafisch weergegeven in de figuren (2.4) t/m (2.8).
N.B. Er is hier uitgegaan van $\hat{x}_i$ en $\hat{p}_i^b$ zoals beschreven in paragraaf 1.5.

In deze grafieken is te zien dat **x** en **p** goed gereconstrueerd worden. De grafieken van de metingen en de schattingen vallen nagenoeg overal over elkaar heen. De (geschaalde) stijfheidswaarde van 978 $[\frac{Nmrad^{-1}}{Nm}]$ wordt ook netjes teruggevonden. Om de resultaten nu beter te kunnen beoordelen worden nu de verschillende residuen bekeken. Een aantal residuen is grafisch weergegeven in de figuren (2.9) t/m (2.14).

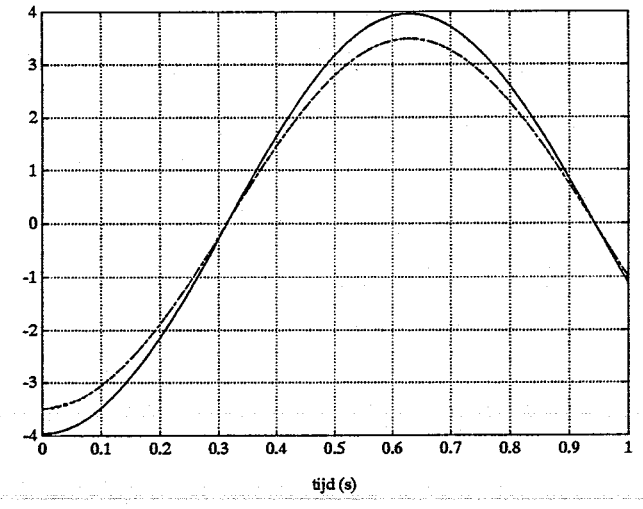
Opmerking Uiteraard zal in werkelijkheid niet zoiets als een residu op de snelheids- of versnellingsschatting bepaald kunnen worden, wanneer er geen snelheids- of versnellingsmetingen zijn verricht. Echter in de simulatie hebben we de metingen zelf gegenereerd en kunnen we wel op deze wijze een controle op het filter uitvoeren.

De relatieve groottes van de residuen op **s** en **v** zijn ± 0.1 %. Dit is precies de grootte van de gebruikte tijdstap (0.001). Dit strookt met de verwachte integratie-nauwkeurigheid bij een 1^e orde integratie-methode (zoals de Euler intgratie-methode); deze is nl. van de orde van de



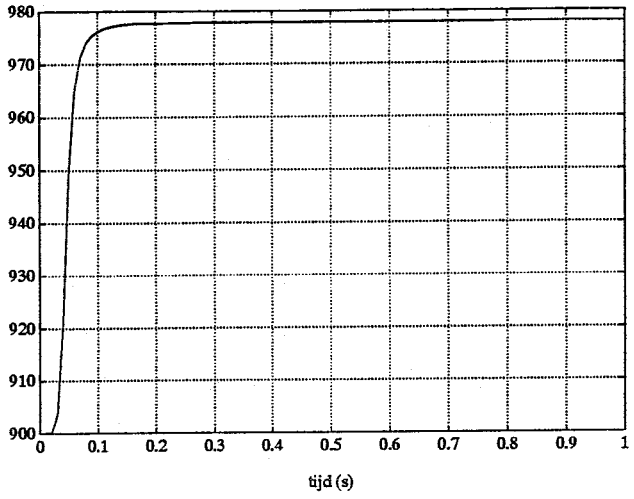
verplaatsingsmeting m_1 : —— ; *verplaatsingsschatting* m_1 : . . . . . .
verplaatsingsmeting m_2 : – – – – ; *verplaatingsschatting* m_2 : – · – · – · –

Figuur 2.4: verplaatsingen [rad]

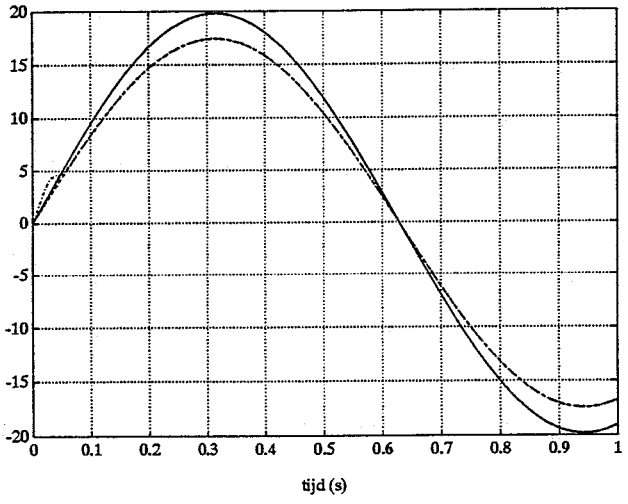


snelheidsmeting m_1 : —— ; *snelheidsschatting* m_1 : . . . . . .
snelheidsmeting m_2 : – – – – ; *snelheidsschatting* m_2 : – · – · – · –

Figuur 2.5: snelheden $[\frac{rad}{s}]$

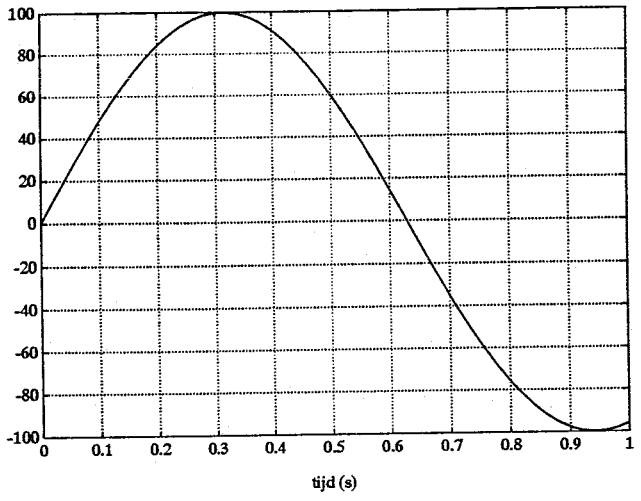


Figuur 2.6: stijfheid $\hat{k}^*$ $[\frac{Nmrad^{-1}}{Nm}]$



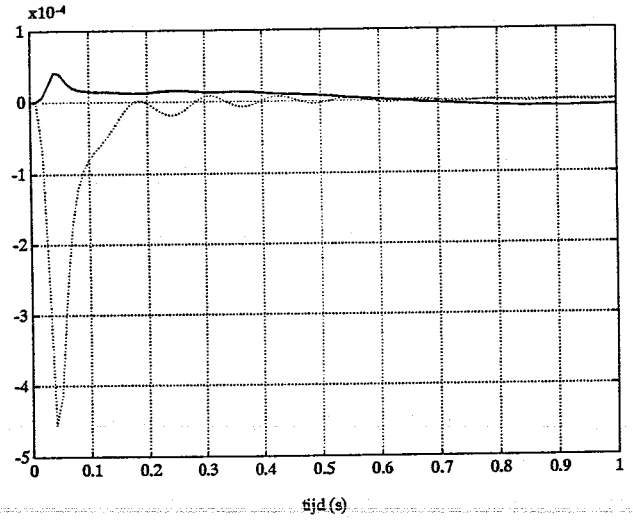
versnellingsmeting m_1 : —— ; *versnellingsschatting* m_1 : .....
versnellingsmeting m_2 : – – – – ; *versnellingsschatting* m_2 : – · – · – · –

Figuur 2.7: versnellingen $[\frac{rad}{s^2}]$



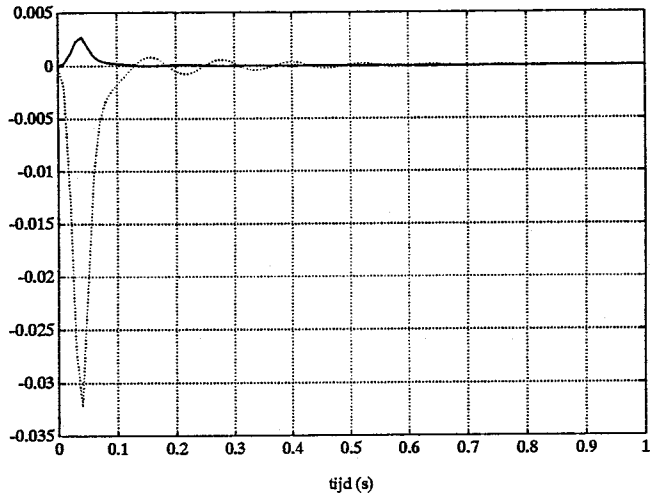
ingangsmeting : —— ; *ingangsschatting* : .....

Figuur 2.8: ingang [-]



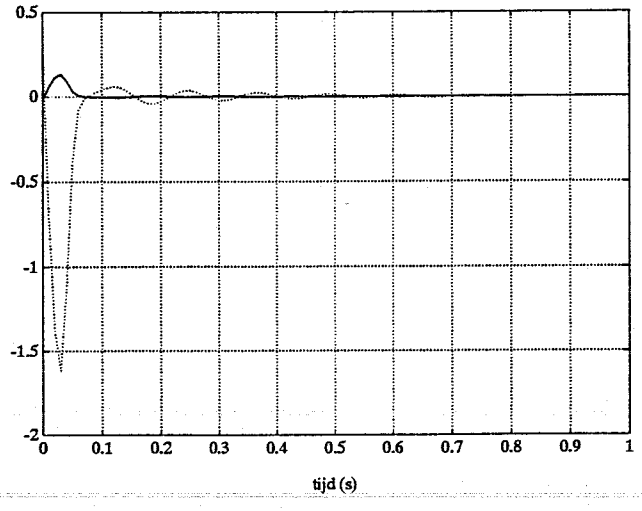
residuenverpl. m_1 : —— ; *residuenverpl.* m_2 : .....

Figuur 2.9: residuen op de verplaatsingen [rad]



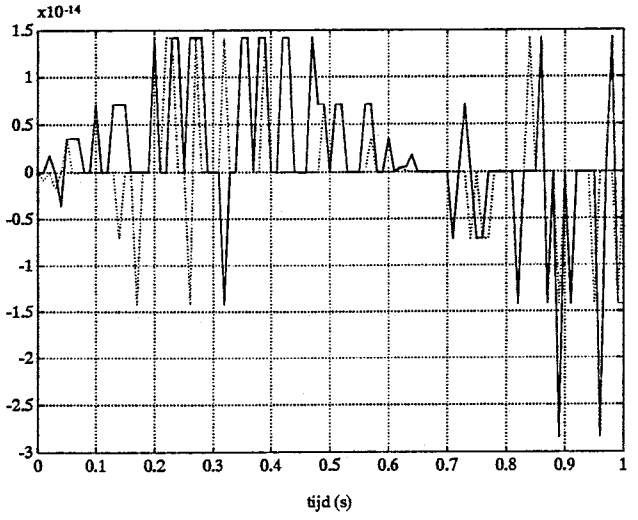
residuensnelh. m_1 : —— ; *residuensnelh.* m_2 : .....

Figuur 2.10: residuen op de snelheden $[\frac{rad}{s}]$



residuenversnel. m_1 : —— ; *residuenversnel.* m_2 : .....

Figuur 2.11: residuen op de versnellingen $[\frac{rad}{s^2}]$

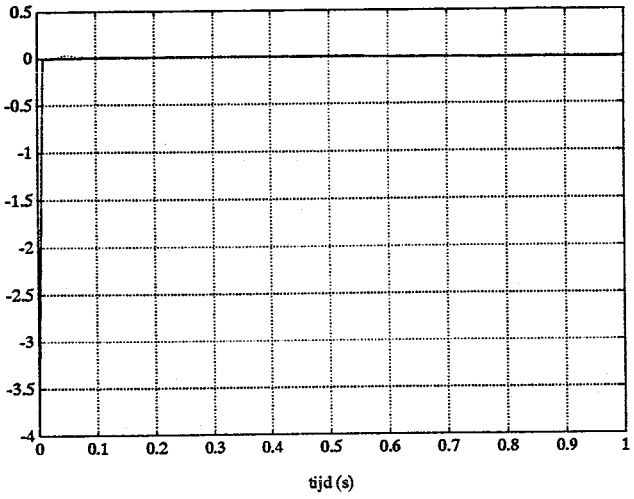


residubew.vgl. (2.6) : ——*residubew.vgl.* (2.7) : .....

Figuur 2.12: residuen op de bewegingsvergelijkingen [-]

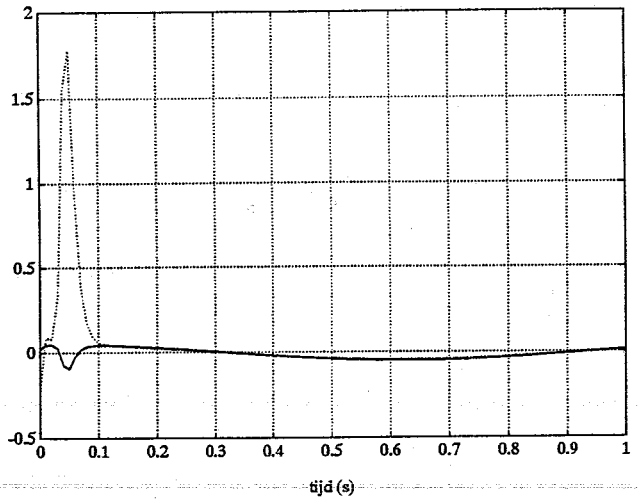
bij integratie gebruikte stapgrootte.
De relatieve grootte van de residuen op **a** is na 0.1 seconden ook ± 0.1 %. Het residu op de ingangsschatting is verwaarloosbaar klein (orde 10^{-16}). De oorzaak hiervan is me nog niet geheel duidelijk, daar er t.a.v. de ingangsmeting geen grotere weging is toegepast dan bij de verplaatsingsmetingen.
Ook de residuen op de bewegingsvergelijkingen zijn zeer klein. Verder zijn nog de residuen op de koppelingsvergelijkingen (1.7) en (1.8) bekeken. In een werkelijke situatie zijn deze residuen de enige controle op de snelheids- en versnellingsschattingen. In de figuren (2.13) en (2.14) is te zien dat deze residuen een relatieve grootte van ± 1 % hebben. Er is nog geprobeerd ze, door aanpassing van de weegmatrices, in dezelfde orde te krijgen als de residuen op de metingen, maar dit is niet echt gelukt.

Opmerking Uit een nadere studie van verscheidene simulatieresultaten bleek dat $\hat{x}_i$ en $\hat{x}_i^b$ niet veel van elkaar verschillen. Wanneer echter $\hat{p}_i$ en $\hat{p}_i^b$ bekeken worden, is een zeer groot verschil waar te nemen. Dit komt doordat, zoals in paragraaf 1.5 besproken, $\hat{p}_i$ eigenlijk bij $\hat{x}_{i-1}$ hoort en niet bij $\hat{x}_i$. Door $\hat{p}_i$ en $\hat{p}_i^b$ te vergelijken, vergelijk je dus eigenlijk grootheden, die bij verschillende tijdstippen horen. De Runge-Kutta stap is dus echt wel noodzakelijk voor zover hierin $\hat{p}_i^b$ berekend wordt. Wanneer de Runge-Kutta stap als geheel weggelaten wordt, worden de **x**-schattingen veel slechter dan $\hat{x}_i$. Dit kan zowel aan het slechtere uitgangspunt t.a.v. de schatting van **x** voor de volgende Eulerstap als aan de slechtere nominale waardes van **p** en **z** (t.b.v. de linearisatie), die als gevolg daarvan ontstaan, liggen.



residukop.vgl.snelheid m_1 : —— ; *residukop.vgl.snelheid* m_2 : .....

Figuur 2.13: residuen op de snelheidskoppelingsvergelijking $[\frac{rad}{s}]$



residukop.vgl.versnelling m_1 : —— ; *residukop.vgl.versnelling* m_2 : .....

Figuur 2.14: residuen op de versnellingskoppelingsvergelijking $[\frac{rad}{s^2}]$

2.4 Een on-line implementatie van het filter

Wanneer het filter on-line geïmplementeerd gaat worden, moet het programma ook in 'real-time' functioneren. D.w.z. dat één filterloop in het programma binnen de tijdstap, waarvan de grootte enerzijds afhankelijk is van de toegepaste integratie methode en de vereiste nauwkeurigheid en anderzijds van de snelheid van de dynamica van het systeem, doorlopen moet worden.
Nu is er gekozen om te programmeren in C++ omdat:

- hierdoor het de rekentijd behoorlijk verkort wordt t.o.v. berekeningen met Matlab.
- wanneer enkele 'library-files', verzorgd door Jos Banens, toegevoegd worden er binnen C++ op een 'Matlab-achtige' manier geprogrammeerd kan worden (dit is o.a. gemakkelijk bij matrixvermenigvuldigingen). Ook is op eenvoudige wijze communicatie met Matlab mogelijk.

Verder wordt verwezen naar documentatie: [2] Tools for Control Experiments (TCE) door J.Banens.
Een listing van het programma, dat de on-line implementatie van het filter verzorgt is te vinden in bijlage 2. Het programma bestaat uit een C++-file: het hoofdprogramma (bijlage 2a) en een Matlab-file, waarin de invoer voor het C++-programma wordt verzorgd (bijlage 2b).

Nu bleek uit experimenten dat, ook na eficiënter programmeren, met de aanwezige hardware (AT386 16MHz) minimaal een tijdstap van 0.04 seconden nodig is om een filterloop te doorlopen. Deze tijdstap is veel te groot om, bij de gebruikte integratie methode en de snelle dynamica van het huidige systeem (met name van belang is de grote stijfheid in het systeem), een convergerend, laat staan een nauwkeurig, resultaat te krijgen. Een tijdstap, die nodig is om convergentie en een nauwkeurig resultaat te garanderen, ligt in de orde van 0.005 seconden (dit volgde uit simulaties). Nu zou natuurlijk een langzame beweging bekeken kunnen worden, waarbij ook nog zoveel mogelijk voor een langzaam reagerend filter gezorgd moet worden. Dit laatste zou gedaan kunnen worden door weinig waarde te hechten aan de metingen door deze niet sterk te wegen. Echter, de dynamica van het systeem blijft in de filtervergelijkingen aanwezig. Ook hiermee werd dus geen convergerend resultaat verkregen. Er waren enkele oplossingen mogelijk:

- Het gebruiken van snellere hardware, bijv. een AT486 i.p.v. de nu aanwezige AT386. Dit bleek echter op korte termijn niet mogelijk.
- Een ander systeem, met een tragere dynamica, bekijken, zodat met de grotere stapgrootte gerekend kan worden.

Er is gekozen voor het 'oude' systeem, waarbij de veer vastgedraaid is, zodat bij het 'nieuwe' systeem, dat hierna het gereduceerde systeem genoemd zal worden, de massa's m_1 en m_2 als een star geheel gemodelleerd kunnen worden. De snelle dynamica is nu uit het systeem. De toestandsvector wordt door deze keuze ook nog korter, waardoor het aantal te integreren differentiaalvergelijkingen behoorlijk wordt verkleind. Het systeem is nu dus gereduceerd tot een massa, waarop wrijving van een nog onbekende vorm werkt. Het model van het gereduceerde systeem is in figuur (3.1) zichtbaar gemaakt.

Opmerking Het C++-programma zou nog iets efficiënter geprogrammeerd kunnen worden door zelf alle vermenigvuldigingen uit te schrijven. Echter hiermee zou de factor 10 winst in rekentijd absoluut niet gehaald worden, terwijl het programma er totaal onoverzichtelijk door zou worden. Wel zou de rekentijd nog iets verkort kunnen worden door in C++ gebruik te maken van 'sparse-matrices'. Echter deze tijdwinst zou niet afdoende zijn. Het draaien van het programma voor het oorspronkelijke systeem met snellere hardware blijft natuurlijk een aanbeveling.

Hoofdstuk 3

Het gereduceerde systeem, gefilterd en geregeld

3.1 Inleiding

Het filter wordt nu alsnog 'on-line' geïmplementeerd op het gereduceerde systeem. Opgemerkt zij dat in de filtervergelijkingen nu foutieve bewegingsvergelijkingen gebruikt worden. Fouten in de gemodelleerde bewegingsvergelijkingen zijn:

- De modellering van de wrijving is niet correct, althans niet volledig naar verwachting. Verwacht wordt nl. dat ook viskeuze wrijving een rol zal spelen aangezien de wrijving ook in de lagers optreedt. Verder kan een wrijvingsbijdrage geleverd worden door het contact tussen een stelschroefje en de onderste massa.
- Het feit, dat de veer vastgedraaid is, wil niet zeggen, dat er totaal geen rek meer in de veer zit. Deze, eventueel aanwezige, zéér hoge stijfheid is ook niet gemodelleerd (eventueel aanwezige demping is ook niet gemodelleerd).

Er kan nu bestudeerd worden of het filter in deze situatie toch goede resultaten levert.
Er zijn 2 soorten experimenten uitgevoerd:

- Experimenten met een open sturing.
- Experimenten met een regeling.

Alleen de resultaten van de experimenten met de regeling zullen besproken worden, daar de resultaten van de experimenten met de open sturing geen extra informatie verschaffen.

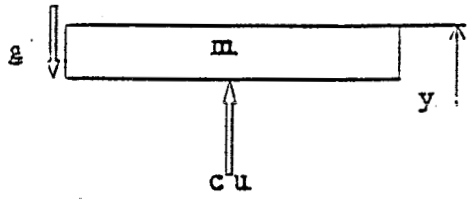
3.2 Het systeemmodel

In figuur (3.1) is het systeemmodel afgebeeld. De bewegingsvergelijking bij dit model is:

$$m\ddot{y} = cu - W \tag{3.1}$$

waarin voor de wrijvingskracht **W** geldt:

$$W = W_s W_g = W_g sign(v) \tag{3.2}$$



Figuur 3.1: Het systeemmodel: massa met wrijving

W_g stelt hierin nu de grootte van de wrijving voor. Schalen we de bewegingsvergelijking weer op **c** dan volgt:

$$\frac{m}{c}\ddot{y} = u - \frac{W}{c} = u - \frac{W_g}{c}sign(v) \tag{3.3}$$

waarbij voor $\frac{m}{c}$ geldt $\frac{m}{c} = \frac{m_1+m_2}{c}$.
W_g wordt nu als parameter meegenomen in de uitgebreide toestand **x**. Voor de uitgebreide toestand **x**, de uitgebreide ingang **p** en de metingen z_m is de volgende keuze gemaakt:

$$x = \begin{bmatrix} y \\ \dot{y} \\ \frac{W_g}{c} \end{bmatrix} \quad ; \quad p = \begin{bmatrix} u \\ \ddot{y} \end{bmatrix} \quad ; \quad z_m = \begin{bmatrix} y \\ u \end{bmatrix} \tag{3.4}$$

De matrices **A,B,E,f,F** in de vergelijkingen (1.2) t/m (1.3) zien er voor dit concrete systeem als volgt uit:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad ; \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{3.5}$$

$$E = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.6}$$

$$f = \begin{bmatrix} x_3 sign(x_2) \\ x_1 \\ 0 \end{bmatrix} \quad ; \quad F = \begin{bmatrix} -1 & \frac{m}{c} \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \tag{3.7}$$

waarin x_i de i_{de} component uit de uitgebreide toestandsvector x is.
Na linearisatie van dit uit identificatie-oogpunt niet-lineaire systeem verkrijgen we de matrices **C** en **D** (zie vergelijkingen (1.23) t/m (1.26)) :

$$C = \begin{bmatrix} 0 & 0 & sign(x_2) \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad ; \quad D = F \tag{3.8}$$

Opmerking De 0 op de (1,2)-positie in **C** is alleen correct voor $x_2 \neq 0$. Voor de afgeleide van $f(1) = x_3 sign(x_2)$ naar x_2 geldt nl. dat deze nul is voor $x_2 \neq 0$ en oneindig voor $x_2 = 0$. In de matrix **C** is hiermee dus geen rekening gehouden. Later zal dus bekeken moeten worden of dit het verkrijgen van goede resultaten in de weg staat. Met deze informatie kunnen nu weer de filtervergelijkingen (1.35) t/m (1.37) toegepast worden op het systeem. Uiteraard zal er dan nog wel eerst een keuze voor de weegmatrices **V** en **W** en voor de beginschattingen $\hat{\mathbf{Q}}_0$ en $\hat{\mathbf{x}}_0$ gemaakt moeten worden.

3.3 De toegepaste regeling

Er wordt nu in de experimenten meteen een regeling toegepast om te testen of de resultaten, die het filter levert, van een dusdanige kwaliteit zijn, dat deze geschikt zijn om als invoer voor een succesvolle regeling te dienen.

3.3.1 De regelwet

Met de bewegingsvergelijking als uitgangspunt (zie vergelijking (3.1) is voor de volgende vorm van de ingang gekozen:

$$u = \frac{m}{c}\ddot{y}_d + \frac{\hat{W}}{c} sign\hat{\dot{y}} + P(y_d - \hat{y}) + D(\dot{y}_d - \hat{\dot{y}}) \tag{3.9}$$

Met **P,D** > 0. Hierin karakteriseren y_d,$\dot{y}_d$ en $\ddot{y}_d$ de gewenste baan. Merk op dat de ingang bestaat uit een **PD**-gedeelte en een gedeelte met wrijvingscompensatie en een versnellingsterm. Wanneer vergelijking (3.9) ingevuld wordt in de bewegingsvergelijking volgt de foutvergelijking:

$$m(\ddot{y}_d - \ddot{y}) + D(\dot{y}_d - \dot{y}) + P(y_d - y) = 0 \tag{3.10}$$

Voor een goede keuze van **P** en **D** geldt dan $y \rightarrow y_d$. Nu wordt **u** als volgt in 2 delen opgesplitst:

$$u_s = \frac{m}{c}\ddot{y}_d + \frac{\hat{W}}{c} sign(\hat{\dot{y}}) \tag{3.11}$$

$$u_c = P(y_d - \hat{y}) + D(\dot{y}_d - \hat{\dot{y}}) \tag{3.12}$$

Beschouw nu de bewegingsvergelijkingen in de volgende vorm:

$$\dot{x}_r = \chi(x_r, u, t) \tag{3.13}$$

Met:

$$x_r = \begin{bmatrix} y \\ \dot{y} \end{bmatrix} \tag{3.14}$$

Deze x_r splitsen we nu op: $x_r = x_{r_d} + x_{r_c}$. In x_{r_d} zijn de gewenste grootheden vertegenwoordigd.

$$\Rightarrow \dot{x}_r = \dot{x}_{r_d} + \dot{x}_{r_c} = \chi(x_{r_d} + x_{r_c}, u_s + u_c, t) \tag{3.15}$$

$$\Rightarrow \dot{x}_{r_c} = \chi(x_{r_d} + x_{r_c}, u_s + u_c, t) - \chi(x_d, u_s, t) \tag{3.16}$$

$$\Rightarrow \dot{x}_{r_c} \simeq \frac{\partial \chi}{\partial x_r} \Big|_{\substack{x_r = x_{r_d} \\ u = u_s}} x_{r_c} + \frac{\partial \chi}{\partial u} \Big|_{\substack{x_r = x_{r_d} \\ u = u_s}} u_c \tag{3.17}$$

$$\Rightarrow \dot{x}_{r_c} \simeq Ax_{r_c} + Bu_c \tag{3.18}$$

Voor het onderhavige geval geldt voor de matrices **A** en **B** het volgende:

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad ; \quad B = \begin{bmatrix} 0 \\ \frac{c}{m} \end{bmatrix} \tag{3.19}$$

terwijl voor u_c de volgende terugkoppeling gehanteerd wordt:

$$u_c = -Lx_{r_c} = L(x_{r_d} - x) \tag{3.20}$$

$$u_c = [PD]\left(\begin{bmatrix} y_d \\ \dot{y}_d \end{bmatrix} - \begin{bmatrix} y \\ \dot{y} \end{bmatrix}\right) \tag{3.21}$$

waarbij **L** de (stationaire waarde van de) terugkoppelmatrix is, waarvoor geldt:

$$L = R^{-1}B^T P_r \tag{3.22}$$

met P_r de oplossing van de onderstaande matrix-Riccati vergelijking:

$$A^T P_r + P_r A + Q - P_r R^{-1} B^T P_r = 0 \tag{3.23}$$

waarin **R** de weegfactor van de regelinspanning en**Q** de weegmatrix voor de toestandsfout voorstellen.

3.4 Resultaten van de 'on-line' toepassing van het filter met regeling

Het programma, waarvan in bijlage 3 een listing te vinden is, dat het voorgaande numeriek implementeert, bestaat weer uit 2 delen,nl. :

- Een **C++**-file : Het hoofdprogramma (bijlage 3a).
- Een Matlab-file, waarin de invoer voor het hoofdprogramma wordt verzorgd (bijlage 3b).

Nu er een kleiner aantal differentiaalvergelijkingen geïntegreerd hoeft te worden, blijkt dat met de aanwezige hardware een minimale tijdstap van 0.025 seconden haalbaar is. Daar de snelle dynamica uit het systeem verdwenen is, is dit voldoende om t.a.v. de integratie convergentie te garanderen.

Verder zal nu weer ingegaan worden op de keuzes van de beginschattingen en weegmatrices.

- De beginschattingen: De beginschattingen zijn te vinden in de listing van de Matlab-file (bijlage 3b). Voor de verplaatsing en de snelheid is beginschatting 0 genomen.
  Motivatie: De beweging begint altijd in positie 0 [rad] (dit is uiteraard een keuze) en voor de snelheid is bewust een foutieve beginschatting genomen om te testen of en hoe snel het filter in deze situatie reageert. Uit eerdere schattingen van de op **c** geschaalde wrijvingsparameter bleek dat deze in de orde van 10 $[\frac{N}{Nm}]$ ligt. Deze waarde is hier als beginschatting genomen. In de beginschatting van **Q** is alleen het element m.b.t de betrouwbaarheid van de beginschatting van de wrijving groot gekozen. Het filter krijgt

hierdoor de vrijheid om de wrijvingsschatting in de eerste stappen snel aan te passen. Met de 0 op de(2,2)-positie in $\hat{Q}_0$ wordt het filter eigenlijk slecht voorgelicht over de betrouwbaarheid van de beginschatting van **v**. Normaal gesproken zou bij een slechte beginschatting een hoge waarde in **Q** gekozen worden, echter door dit niet te doen kan bekeken worden hoe snel het filter de **Q** aanpast om daarmee de **x**-schattingen in de juiste te wijzingen (zie resultaten).

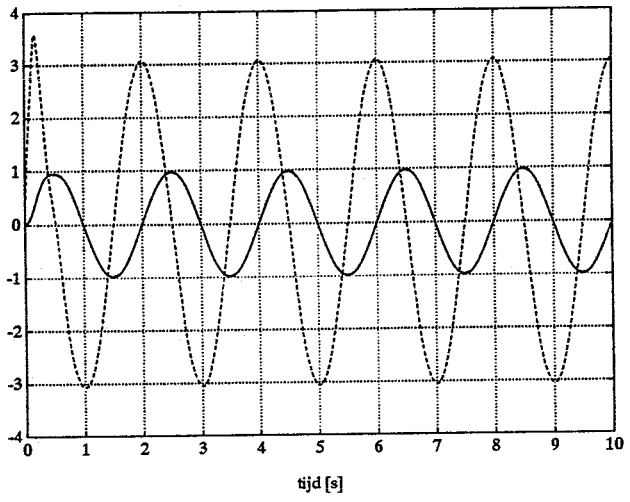
- De weegmatrices
  - T.a.v. het filter:
    De weegmatrices **V** en **W** zijn te vinden in de listing van de Matlab-file (bijlage 3b). Uitgangspunt bij de keuze van **V** en **W** dat het vertrouwen in de bewegingsvergelijking klein is en dat de wrijvingsschatting veel vrijheid moet krijgen. Dit laatste is nodig omdat hiermee de modelleringsfouten gecompenseerd moeten worden. Het is nl. zeer waarschijnlijk dat de wrijvingsparameter niet een zuiver Coulombse wrijvingsparameter voorstelt. Hierop zal later nog verder op worden ingegaan. Eerlijkheid gebied te zeggen dat de uiteindelijke **V** en **W** gekozen zijn op grond van een 'trial and error'-proces, waarbij het criterium was het over de gehele linie minimaliseren van de residuen.
  - T.a.v. de regeling:
    De weegmatrices **R** en **Q** zijn te vinden in de listing van de Matlab-file (bijlage 3b). Als uitgangspunt bij de keuze van de componenten van **R** en **Q** is voor een beschouwing van de ordegroottes van de grootheden, die corresponderen met de **R**,**Q**-componenten, gekozen.

    N.B. Volgens dit principe is ook geprobeerd tot een goede keuze voor **V** en **W** te komen, echter in dat geval leidde dat niet tot goede resultaten.
    Daarna is eerst de verhouding tussen de groottes van **R**en **Q** aangepast om wat meer regelinspanning toe te laten (kleinere **R** c.q. grotere **Q**). Dit bleek een beter regelresultaat (kleinere volgfout) te bevorderen. Ook werd een nog iets grotere P-actie toegelaten ((1,1)-component van **Q** vergroten) om een nog beter regelresultaat te bewerkstelligen. Dit laatste werd steeds doorgevoerd tot net voordat de ingang zo snel fluctueerde, dat het systeem alleen nog maar wat trilde. Zo werd een bevredigend regelresultaat bereikt.

Eerst zal nu een experiment, waarbij niet voor de grootst mogelijke P-actie in de regeling gekozen is om niet al te wilde ingangssignalen te creëren, besproken worden (de bijbehorende **Q** is in de listing van het Matlab-programma te vinden).
De resultaten van het experiment:
De reconstructies van **x** en **p** zijn grafisch weergegeven in de figuren (3.2) t/m (3.5).

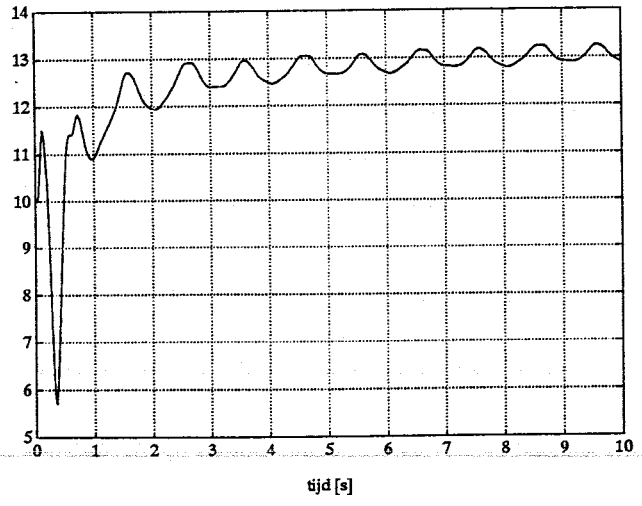
N.B. Er is hier weer uitgegaan van $\hat{x}_i$ en $\hat{p}_i^b$ zoals beschreven in paragraaf (1.5).

T.a.v. de verplaatsing **s** en de ingang **u** kan a.d.h.v. de figuren (3.2) en (3.4) geconcludeerd worden dat de reconstructies goed lijken te zijn. Er zal echter nog wel naar de residuen gekeken moeten worden. Voor de controle op de kwaliteit van de reconstructies van **v** en **a** is dit zeker noodzakelijk. In figuur (3.3) is te zien dat de schatting van de wrijvingsparameter

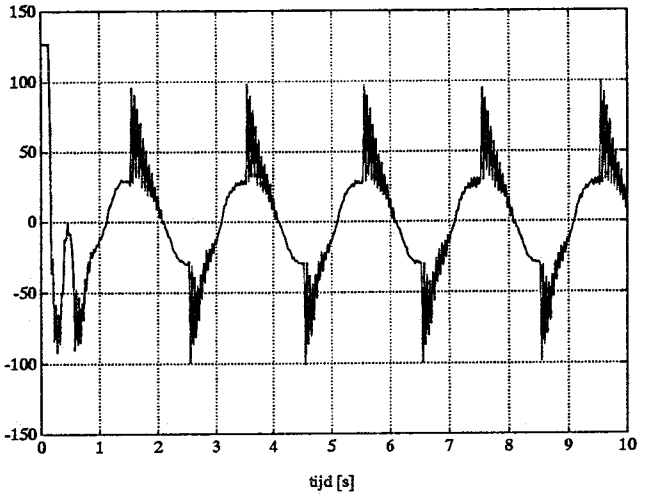


verplaatsingsschatting : —— ; *verplaatsingsmeting* : ..... *snelheidsschatting* : – – – –

Figuur 3.2: verplaatsing en snelheid ([rad] $\frac{rad}{s}$)

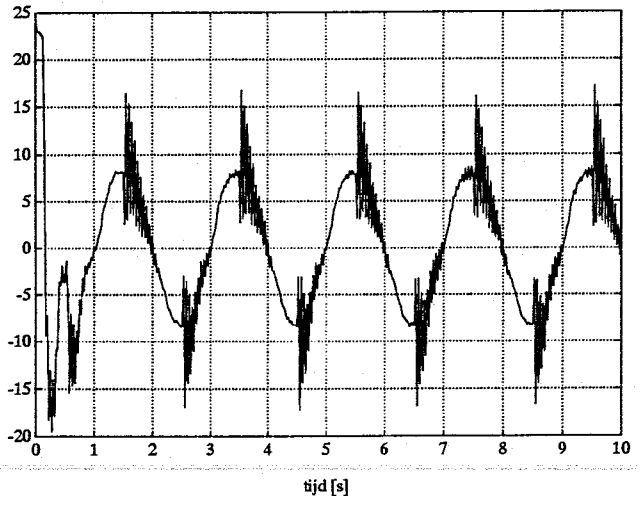


Figuur 3.3: schatting Coulombse wrijvingsparameter $\frac{W_g}{c}$ [-]

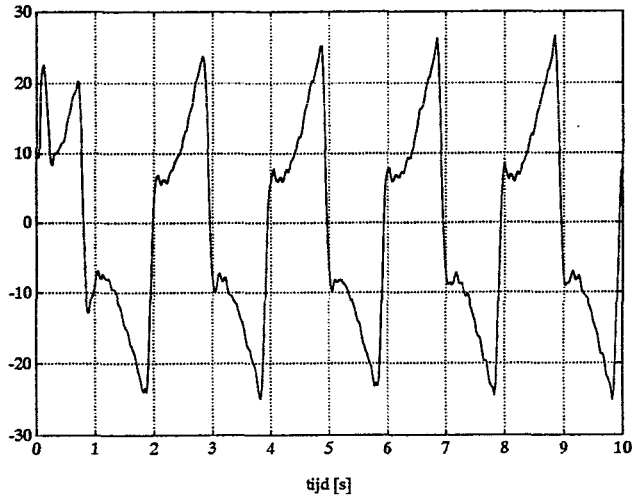


ingangsschatting : —— ; *ingangsmeting* : .....

Figuur 3.4: ingang [-]



Figuur 3.5: versnelling $[\frac{rad}{s^2}]$

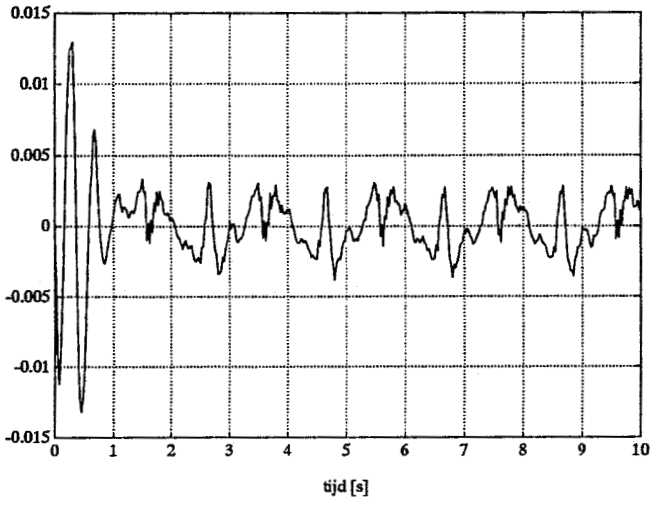


Figuur 3.6: Schatting wrijvingsparameter met tekenwisselingen [-]

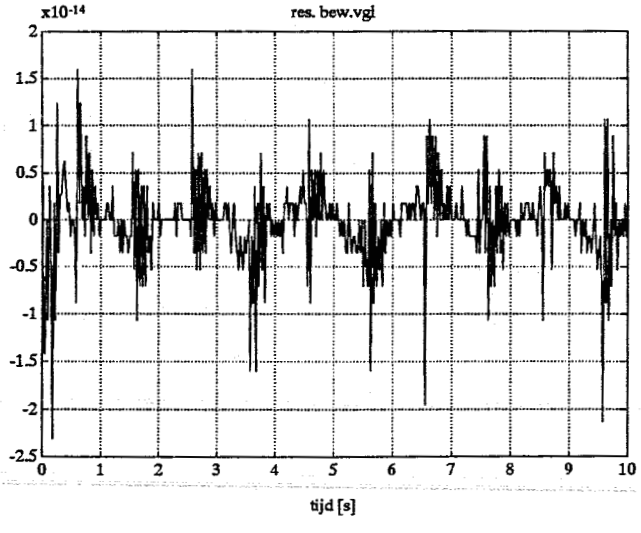
inderdaad in de orde van 10 blijft. Globaal bekeken gaat de schatting ook naar een constant niveau. Duidelijk aanwezig zijn echter de golven op de reconstructie. Dat deze variatie op de wrijvingsschatting blijft bestaan, kan verklaard worden door in te zien dat deze schatting de niet correcte modellering van de wrijving en van het systeem als geheel continu moet compenseren.

Om toch iets beter inzicht in de in het systeem aanwezige wrijvingsvormen is de wrijving in het filtermodel ook een keer als constante meegenomen, d.w.z. dat in deze parameter ook de tekenwisselingen zitten. De schatting van deze parameter is in figuur (3.6) afgebeeld. interessanter is, is dat er een duidelijke snelheidsafhankelijkheid van de wrijving in figuur (3.6) te herkennen is. Er is dus een soort viskeuze wrijving in het systeem aanwezig. Dit wil niet zeggen dat de viskeuze wrijvingsparameter dan een constante zou moeten zijn. Deze parameter op zich zou nl. ook nog afhankelijk van de snelheid kunnen zijn, aangezien, wanneer de figuren (3.6) en (3.2) gecombineerd worden, een iets meer dan dan proportioneel verloop van de schatting van de wrijvingsparameter als functie van de snelheid te herkennen is.

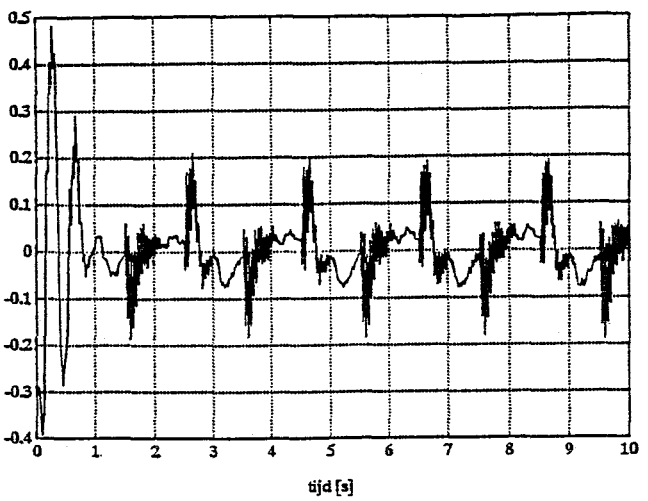
Nu zullen de residuen bekeken worden. Een aantal residuen is grafisch afgebeeld in de figuren (3.7) en (3.8). Hierin is te zien dat het residu op de verplaatsing in de orde van 0.0025 [rad] blijft ($\pm 0.25\%$ reconstructiefout). Dit is wel bevredigend aangezien de tijdstapgrootte in de Eulerintegratie 0.025 seconden is. Duidelijk is dat op dit moment de fouten, die tijdens de integratie gemaakt worden, bepalend zijn voor de totale fout van het filter. Wanneer met een kleinere stapgrootte gewerkt zou kunnen worden zou de fout, die bij de linearisatie gemaakt wordt, wel eens bepalend kunnen worden. Het residu op de ingangsschatting is zéér klein. Dit zou verklaard kunnen worden door het feit dat de metingen van $\mathbf{u}$, die gebruikt worden, berekende waarden zijn (daar zit dus geen meetfout op). Het residu op de bewegingsvergelijking is ook erg klein. Tot zover zijn de resultaten dus prima. De residuen



Figuur 3.7: residu op de verplaatsing [rad]



Figuur 3.8: residu op de bewegingsvergelijking [-]

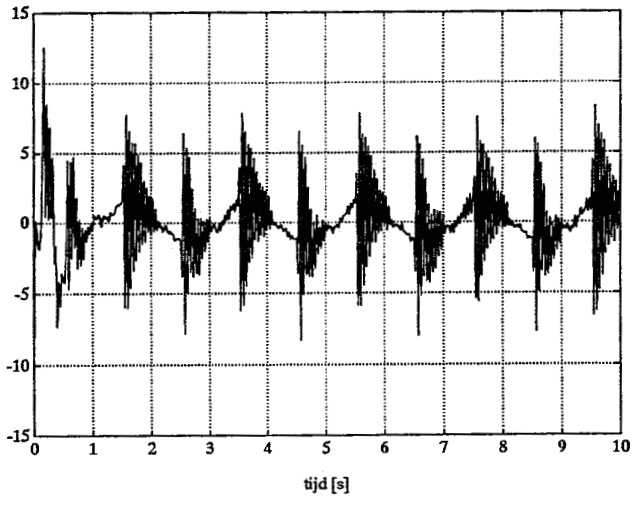


Figuur 3.9: residu op de snelheids-koppelingsvergelijking $[\frac{rad}{s}]$

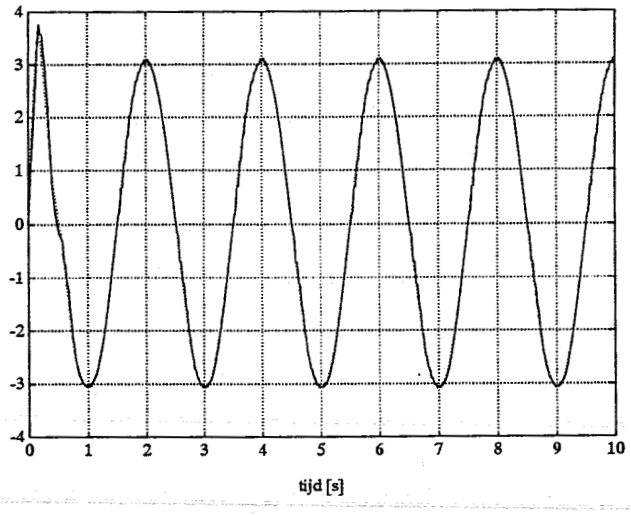
op de koppelingsvergelijkingen zijn wel groter dan verwacht (zie figuren (3.9) en (3.10)). In de figuren (3.11) en (3.12) is echter te zien dat de resultaten toch niet zo slecht zijn. Het grote residu op de versnellings-koppelingsvergelijking is te verklaren door het feit dat het filter de snelle veranderingen in a, a.g.v. het snel fluctueren van de door de regeling voorgeschreven ingang, met de gebruikte stapgrootte van 0.025 seconden niet goed genoeg kan volgen. Een verbetering (ook t.a.v. de snelheids-koppelingsvergelijking) zou dus bewerkstelligd kunnen worden door de stapgrootte te verkleinen. Hier is echter wel snellere hardware voor nodig. Ook is te zien dat de gevolgen van de foute keuze van de beginschatting van v snel verdwijnen. Over het algemeen zijn de resultaten van het filter (bij de gebruikte stapgrootte voor de integratie) dus wel bevredigend.

Verder moet nu nog het regelresultaat bekeken worden. Hiermee kan dan bekeken worden of de filterresultaten goed genoeg zijn om als invoer voor een succesvolle regeling te kunnen dienen. Het regelresultaat is zichtbaar gemaakt in de figuren (3.13) en (3.14). hieruit blijkt dat de volgfout $\pm 2\%$ is. Aangezien het feit dat in het filter al met een stapgrootte van 0.025 seconden gewerkt wordt (en dus daar al een fout in de orde van procenten verwacht mag worden), kan dit resultaat als bevredigend beschouwd worden.

Er zijn ook nog experimenten uitgevoerd met maximaal mogelijke P-actie, maar dat leverde nauwelijks een beter regelresultaat op, terwijl de ingang zo snel en hevig ging fluctueren dat de fout op de versnellings-koppelingsvergelijking alleen nog maar groter werd. Bij kleinere stapgroottes zou wel geprobeerd kunnen worden het regelresultaat te verbeteren door de weegmatrices van de regeling te wijzigen.

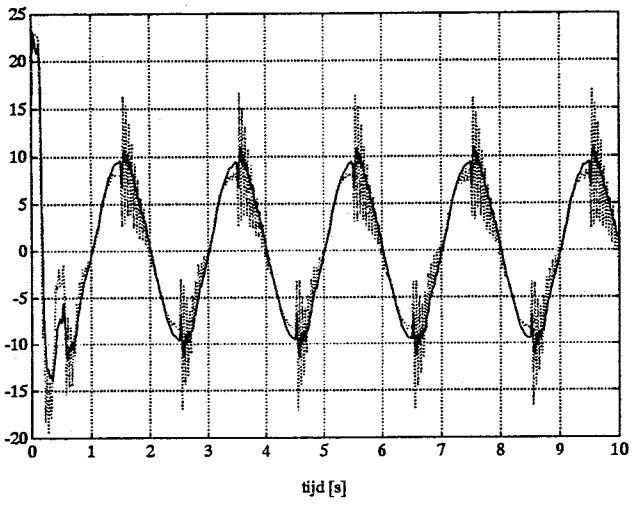


Figuur 3.10: residu op de versnellings-koppelingsvergelijking $[\frac{rad}{s^2}]$



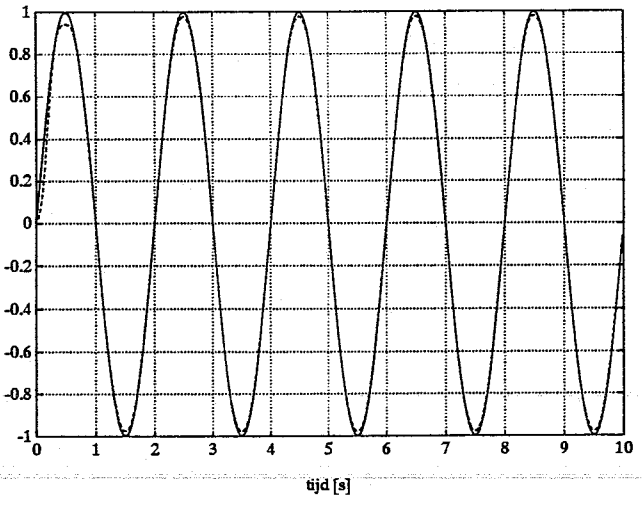
$\dot{\hat{s}}$: —— ; $\hat{v}$: .....

Figuur 3.11: snelheidsschattingen $[\frac{rad}{s}]$



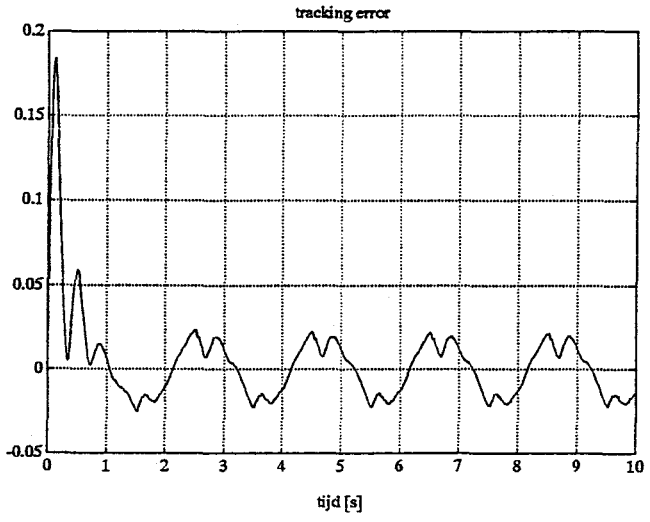
$\dot{\hat{v}}$: —— ; $\hat{a}$: .....

Figuur 3.12: versnellingsschattingen $[\frac{rad}{s^2}]$



gewenstebaan : —— ; *verplaatsingsmeting* : .....

Figuur 3.13: gewenste en doorlopen baan [rad]



Figuur 3.14: de volgfout [rad]

Opmerkingen

- De filterresultaten van de experimenten met de open sturing waren ook goed, waarbij ook de residuen op de koppelingsvergelijkingen klein waren (in de orde van 1%), omdat de ingang nu wat 'netter' verliep: $u = sin(\pi t)$.
- Daar de verkregen resultaten bevredigend zijn en geen vreemde of onverwachte resultaten op de tijdstippen waarop $x_2 = 0$ verkregen zijn kan er geconcludeerd worden dat het meenemen van een foutieve **C**-matrix voor de snelheid $x_2 = 0$ geen duidelijk nadelige gevolgen heeft t.a.v. de resultaten van de experimenten.

Hoofdstuk 4

Conclusies en aanbevelingen

Geconcludeerd kan worden dat het lineaire, optimale filter geschikt is voor 'on-line'-implementatie op niet-lineaire systemen. Wel is gebleken dat het programma dat het filter numeriek verzorgd relatief omvangrijk wordt. Hierdoor moet men óf zorgen voor hardware, die snel genoeg is om de berekeningen in de filter-loop binnen een tijdstapgrootte, die nauwkeurige filterresultaten kan garanderen, uit te voeren óf genoegen nemen met minder nauwkeurige resultaten (zoals hier het geval was). Hieruit vloeien enkele aanbevelingen voort:

- Het programma voor het gereduceerde systeem (zie bijlage 3) zou nog iets efficiënter geprogrammeerd kunnen worden door o.a. het gebruik van 'sparse-matrices'. Ook zal snellere hardware gebruikt moeten worden. Dan zouden kleinere stapgroottes gebruikt kunnen worden. Hiermee kunnen de residuen dan overeenkomstig verkleind worden, waardoor het filter nog geschikter wordt voor 'on-line'-implementatie met regelingen.
- idem, maar dan voor het oorspronkelijke systeem (zie bijlage 2). Het is aan te bevelen het filter op verschillende niet-lineaire systemen te testen.
- Uiteraard zou het ook goed zijn om, wanneer het filter echt nauwkeurige resultaten levert, zorgvuldiger gekozen regelwetten toe te passen.

Bibliografie

[1] M.J.G. van de Molengraft, Identification of non-linear mechanical systems for control application. -[S.1] proefschrift TUE. - I11, Eindhoven,(1990).

[2] J. Banens, documentatie bij Tools for Control Experiments (TCE), Technische Universiteit Eindhoven.

Bijlage 1a

```
%**********************   STAGSV.m   **********************
%        twee-massa-veer-demper-systeem

clear

%
%+++++++++++++++++    GEGEVENS    ++++++++++++++++++++++++
%

% mogelijkheid tot inbreng ruis
  percz1=input('geef ruispercentage in verpl.meting massa1:  ');
  percz2=input('geef ruispercentage in verpl.meting massa2:  ');
  percu=input('geef ruispercentage in ingangsmeting:  ');
  perca=input('geef ruispercentage in versn.meting:  ');


%    systeemparameters

m1=4.7;
m2=0.39;
k=978;
b=2.7;

%    excitatie-gegevens

excs

Au=100;
omega=5;

z10=A1*(1+(percz1/100)*randn(1));
z20=A2*(1+(percz2/100)*randn(1));
v10=B1*omega;
v20=B2*omega;
a10=-A1*((omega)^2);
a20=-A2*((omega)^2);
u0=(percu/100)*randn(1);
m0=[z10
    z20
    u0];
zz1=z10;
zz2=z20;
vv1=v10;
vv2=v20;
aa1=a10;
aa2=a20;
FF=u0;
mm=m0';

% gegevens tijdsdiscretisatie

deltat=0.001;
te=1;
t0=0;
t=[0:deltat:te]';

%gegevens t.b.v. dvs.m

eps1=1e-6;
eps2=1e-6;

%    beginwaardes

x=[z10 z20 v10 v20 900]';
```

```
xx=x';
xxe=x';
xxdot1=[0 0 0 0 0]';
xxdot2=[0 0 0 0 0]';
invR0=[0 0 0 0 0
       0 0 0 0 0
       0 0 0 0 0
       0 0 0 0 0
       0 0 0 0 1e8];
Q=invR0;
QQ=Q;
f0=[x(1)*x(5)-x(2)*x(5)+b*x(3)-b*x(4)
   -x(1)*x(5)+x(2)*x(5)-b*x(3)+b*x(4)
         x(1)
         x(2)
         0                ];
F0=[0 m1 0
   -1 0 m2
    0 0 0
    0 0 0
    1 0 0];


%    systeem- & weegmatrices

A=[0 0 1 0 0
   0 0 0 1 0
   0 0 0 0 0
   0 0 0 0 0
   0 0 0 0 0];
B=[0 0 0
   0 0 0
   0 1 0
   0 0 1
   0 0 0];
E=[0 0 0
   0 0 0
   1 0 0
   0 1 0
   0 0 1];
V=[1e5 0 0 0 0
   0 1e5 0 0 0
   0 0 1e4 0 0
   0 0 0 1e4 0
   0 0 0 0 1e4];

W=[1e3 0 0 0 0
   0 1e3 0 0 0
   0 0 1e3 0 0
   0 0 0 1e3 0
   0 0 0 0 1e-2];

% beginschatting p

  p=(inv(F0'*V*F0))*F0'*V*(E*m0-f0);
  pp=p';
  p0=p;
  ps=p;
  rr=[0 (u0-p0(1)) 0 0 (u0-p0(1))];

% beginschatting y (array in integratie-methode)

    y0=[Q(1,1);Q(1,2);Q(1,3);Q(1,4);Q(1,5);Q(2,2);Q(2,3);Q(2,4);Q(2,5)
```

```
        Q(3,3);Q(3,4);Q(3,5);Q(4,4);Q(4,5);Q(5,5);x(1);x(2);x(3);x(4);x(5)];
    y=y0;

%
%+++++++++++++++++++   OPTIMALE FILTER  ++++++++++++++++++++++++
%

iter=0;

for tj=t(2,1):deltat:t(length(t),1) ,

%
%+++++++++++++++  GENERATIE METINGEN  ++++++++++++++++++
%

    F=(Au*sin(omega*tj))*(1+(percu/100)*randn(1));
    FF=[FF;F];

    z1=B1*sin(omega*tj)+A1*cos(omega*tj)*(1+(percz1/100)*randn(1));
    zz1=[zz1;z1];

    z2=B2*sin(omega*tj)+A2*cos(omega*tj)*(1+(percz2/100)*randn(1));
    zz2=[zz2;z2];

    v1=-A1*omega*sin(omega*tj)+B1*omega*cos(omega*tj);
    vv1=[vv1;v1];

    v2=-A2*omega*sin(omega*tj)+B2*omega*cos(omega*tj);
    vv2=[vv2;v2];

    a1=-B1*((omega)^2)*sin(omega*tj)-A1*((omega)^2)*cos(omega*tj);
    aa1=[aa1;a1];

    a2=-B2*((omega)^2)*sin(omega*tj)-A2*((omega)^2)*cos(omega*tj);
    aa2=[aa2;a2];

    m=[z1
       z2
       F];
    mm=[mm;m'];

% opslag oude y

    yp=y;

%
%  eerste aanroep functiebeschrijving
%  Eulerstap

    tm=tj-deltat;
    dvs

    f1=yprime;
    xdot1=[f1(16) f1(17) f1(18) f1(19) f1(20)]';
    y=yp+deltat*f1;

    Q=[y(1) y(2) y(3) y(4) y(5)
       y(2) y(6) y(7) y(8) y(9)
       y(3) y(7) y(10) y(11) y(12)
       y(4) y(8) y(11) y(13) y(14)
       y(5) y(9) y(12) y(14) y(15)];

    x=[y(16) y(17) y(18) y(19) y(20)]';
```

```
%
%  tweede aanroep functiebeschrijving
%
%  bepaling p behorende bij y(na Eulerstap) en
%  verbeterde y-schatting

      tm=tj;
      dvs

      f2=yprime;
      xdot2=[f2(16) f2(17) f2(18) f2(19) f2(20)]';

% bepaling nieuwe schatting y

      y=yp+deltat*((f1+f2)/2);

      p=ps;
      xe=[y(16);y(17);y(18);y(19);y(20)];

     xx=[xx;x'];
     pp=[pp;p'];
     xxe=[xxe;xe'];
     xxdot1=[xxdot1;xdot1'];
     xxdot2=[xxdot2;xdot2'];

% residuen bepaling

fn=[x(1)*x(5)-x(2)*x(5)+b*x(3)-b*x(4)
   -x(1)*x(5)+x(2)*x(5)-b*x(3)+b*x(4)
         x(1)
         x(2)
         0                ];
Fn=[0 m1 0
   -1 0 m2
    0 0 0
    0 0 0
    1 0 0];

    r=E*m-fn-Fn*p;
    rr=[rr;r'];

    iter=iter+1;
    iter

end

% reduceren v/h aantal data-punten

    mmr=mm(1:10:length(mm),:);
    vv1r=vv1(1:10:length(vv1));
    vv2r=vv2(1:10:length(vv2));
    aa1r=aa1(1:10:length(aa1));
    aa2r=aa2(1:10:length(aa2));
    tr=t(1:10:length(t));
    xxr=xx(1:10:length(xx),:);
    xxdot1r=xxdot1(1:10:length(xxdot1),:);
    xxdot2r=xxdot2(1:10:length(xxdot2),:);
    rrr=rr(1:10:length(rr),:);
    ppr=pp(1:10:length(pp),:);



% data t.b.v plotjes
```

```
save tr.mat tr /ascii
save xxr.mat xxr /ascii
save ppr.mat ppr /ascii
save rrr.mat rrr /ascii
save mmr.mat mmr /ascii
save vv1r.mat vv1r /ascii
save vv2r.mat vv2r /ascii
save xxdot1r.mat xxdot1 /ascii
save xxdot2r.mat xxdot2 /ascii
save aa1r.mat aa1r /ascii
save aa2r.mat aa2r /ascii
```

Bijlage 1b

```
%  beschrijving dv's


      Qh=[y(1) y(2) y(3) y(4) y(5)
          y(2) y(6) y(7) y(8) y(9)
          y(3) y(7) y(10) y(11) y(12)
          y(4) y(8) y(11) y(13) y(14)
          y(5) y(9) y(12) y(14) y(15)];
      xh=[y(16);y(17);y(18);y(19);y(20)];

      f=[y(20)*y(16)-y(17)*y(20)+b*(y(18)-y(19))
        -y(20)*y(16)+y(17)*y(20)-b*(y(18)-y(19))
          y(16)
          y(17)
          0];
      F=[0 m1 0
          -1  0 m2
          0 0 0
          0 0 0
          1 0 0];

% bepaling nominale waarden waarom linearisatie plaatsvindt

      pl(:,1)=ps;
      zl(:,1)=(inv(E'*E))*E'*(f+F*pl(:,1));
      pl(:,2)=(inv(F'*F))*F'*(E*zl(:,1)-f);
      zl(:,2)=(inv(E'*E))*E'*(f+F*pl(:,2));
      k=2;
      while ((((pl(1,k)-pl(1,k-1)) >= eps1)  |
      ((zl(1,k)-zl(1,k-1)) >= eps2))|(((pl(2,k)-pl(2,k-1)) >= eps1)  |
      ((zl(2,k)-zl(2,k-1)) >= eps2)) | (((pl(3,k)-pl(3,k-1))>= eps1) |
      ((zl(3,k)-zl(3,k-1)) >= eps2))) & (k<=20) ,
           k=k+1;
           pl(:,k)=(inv(F'*F))*F'*(E*zl(:,k-1)-f);
           zl(:,k)=(inv(E'*E))*E'*(f+F*pl(:,k));
      end
      pll=pl(:,k);
      zll=zl(:,k);

% matrices

      C=[y(20) -y(20) b -b (y(16)-y(17))
          -y(20) y(20) -b b (y(17)-y(16))
          1     0     0     0     0
          0     1     0     0     0
          0     0     0     0     0];
      D=[0 m1 0
          -1  0 m2
          0 0 0
          0 0 0
          1 0 0];

      L11=A-B*(inv(D'*V*D))*(D'*V*C);
      L12=B*(inv(D'*V*D))*B'+inv(W);
      L21=C'*V*C-(C'*V*D)*(inv(D'*V*D))*(D'*V*C);

% metingen

      mig=tm/deltat;
      z=(mm(mig+1,:))';

bepaling v/d afgeleiden en nieuwe schatting p
```

```
varp=(inv(D'*V*D))*D'*V*E*(z-zll);
ps=pll+varp;

Qdot=L12+L11*Qh+Qh*L11'-Qh*L21*Qh;
xdot=A*xh+B*ps+Qh*C'*V*(E*(z-zll)-D*varp);

yprime=[Qdot(1,1);Qdot(1,2);Qdot(1,3);Qdot(1,4);Qdot(1,5)
        Qdot(2,2);Qdot(2,3);Qdot(2,4);Qdot(2,5);Qdot(3,3)
        Qdot(3,4);Qdot(3,5);Qdot(4,4);Qdot(4,5);Qdot(5,5)
        xdot(1);xdot(2);xdot(3);xdot(4);xdot(5)];
```

Bijlage 1c

```
% excitatie F=Au*sin(omega*t)
% bepaling informatie t.b.v. generatie metingen

Au=100;
m1=4.7;
m2=0.39;
b=2.7;
k=978;
omega=5;

AA=[(-omega*b) (-((omega)^2)*m1+k) (omega*b) (-k)
    (-((omega)^2)*m1+k) (omega*b) (-k) (-omega*b)
    (omega*b) (-k) (-omega*b) (k-((omega)^2)*m2)
    (-k) (-omega*b) (k-((omega)^2)*m2) (omega*b)];

bb=[0;0;Au;0];

xx=inv(AA)*bb;

A1=xx(1);
B1=xx(2);
A2=xx(3);
B2=xx(4);
```

Bijlage 2a

```
/*      filt02.c
        =====
Nathan v/d Wouw // Rene v d Molengraft.
Uses type mat.

*/

#include "tce.h"
#include "mat.h"
#include "exp.h"
#include <stdio.h>      /*Standaard input/output functies       */
//#include <math.h>

/*      Globale declaraties                                     */
/*      In deze implementatie moeten alle globale variabelen   */
/*      hieronder gedeclareerd worden.                          */

mat     u(1,1),x(5,1),xe (3,1),xt(3,1),z(2,1);
int     n;                      /* aantal acties       */
double  T;                      /* Sample tiid [s]      */
double  sys[2];                 /* van IN2.MAT          */
double  wait[3] = { 0, 0, 0 };  /* Wait counts          */

mat     invE(3,5),F(5,3),invF(3,5),A(5,5),At(5,5),B(5,3),C(5,5),Ct(5,5),
        L11(5,5),L12(5,5),L11t(5,5),L21(5,5),invD(3,3),Dt(3,5),D(5,3),V(5,5),
        Bt(3,5),invDt(3,3),E(5,3),Qdot(5,5);

mat     f(5,1),pl(3,1),pl_old(3,1),zl(3,1),zl_old(3,1),
        ps(3,1),zm(3,1),varp(3,1),xdot(5,1);

mat     yp(20,1),f1(20,1),f2(20,1),p(3,1),xdot1(5,1),xdot2(5,1),xp(5,1),
        zml(3,1),Ez(5,1),Ax(5,1),Bps(5,1),Dvp(5,1),r(5,1),fn(5,1);
mat     Q(5,5),K1(5,5),K2(5,5),K3(3,3),I(5,5),Qdot1(5,5),Qdot2(5,5),Qp(5,5);

double  b,eps1,m1,m2;
int     k;

// shorthands (afkortingen)

#define tsi(who)        ts_init(who,n,#who)
#define gm(who)         who = ml_get_mat(#who)
#define pm(who)         ml_put_mat(who,#who)
#define ms(who)         mat_show(who)

// bepaling afgeleiden en nieuwe schatting p

void    DVS(void) {

        f.p[0]  = x.p[4]*(x.p[0] - x.p[1]) + b*(x.p[2] - x.p[3]);
        f.p[1]  = x.p[4]*(x.p[1] - x.p[0]) - b*(x.p[2] - x.p[3]);
        f.p[2]  = x.p[0];
        f.p[3]  = x.p[1];

        pl_old  = ps;
        zl_old  = invE*(F*pl_old+f);
        pl      = invF*(E*zl_old - f);
        zl      = invE*(f + F*pl);
        k = 2;
        while (( ( (pl.p[0] - pl_old.p[0]) >= eps1 ) ||
        ( (pl.p[1] - pl_old.p[1]) >= eps1 )  ||
        ( (pl.p[1] - pl_old.p[1]) >= eps1 ) ||
        ( (zl.p[0] - zl_old.p[0]) >= eps1 ) ||
        ( (zl.p[1] - zl_old.p[1]) >= eps1 )  ||
        ( (zl.p[1] - zl_old.p[1]) >= eps1 ) )  && (k <= 4) ) {
```

```
        k = k + 1;                                          // beveiliging
        pl_old  = pl;
        zl_old  = zl;
        pl      = invF*(E*zl_old - f);
        zl      = invE*(f + F*pl);
        }

        C.p[0]  = x.p[4];
        C.p[1]  = -x.p[4];
        C.p[5]  = -x.p[4];
        C.p[6]  =  x.p[4];
        C.p[20] = x.p[0] - x.p[1];
        C.p[21] = x.p[1] - x.p[0];

        Ct   = tran(C);

        L11  = A-K1*C;
        L21  = Ct*K2*C;
        L11t = tran(L11);

        // metingen: z u
        zm.p[0] = z.p[0];
        zm.p[1] = z.p[1];
        zm.p[2] = u.p[0];

        varp = K3*(zm-zl);
        ps   = pl + varp;

        zml =zm - zl;

        Ez.p[0] = 0;
        Ez.p[1] = 0;
        Ez.p[2] = zml.p[0];
        Ez.p[3] = zml.p[1];
        Ez.p[4] = zml.p[2];

        Ax.p[0] = x.p[2];
        Ax.p[1] = x.p[3];

        Bps.p[2] = ps.p[1];
        Bps.p[3] = ps.p[2];

        Dvp.p[0] = m1*varp.p[1];
        Dvp.p[1] = m2*varp.p[2]-varp.p[0];
        Dvp.p[4] = varp.p[0];

// bepaling afgeleides

        Qdot = L12 + (2*L11 - Q*L21)*Q;
        Qdot = 0.5*(Qdot + tran(Qdot)); // (waarborging symmetrie)
        xdot = Ax + Bps + Q*(Ct*(V*(Ez - Dvp)));

}
```

```
// De Eulerstap

void    PRED(void) {

        DVS();
        xdot1 = xdot;
        Qdot1 = Qdot;
        x += T*xdot1;
        Q += T*Qdot1;
        xe = x;
```

```
}

// Runge-Kutta

void    CORR(void) {
        DVS();
        x += T*0.5*(xdot - xdot1);
        Q += T*0.5*(Qdot - Qdot1);
        p = ps;

        fn.p[0] = xe.p[4]*(xe.p[0]-xe.p[1])+b*(xe.p[2]-xe.p[3]);
        fn.p[1] = -xe.p[4]*(xe.p[0]-xe.p[1])-b*(xe.p[2]-xe.p[3]);
        fn.p[2] = xe.p[0];
        fn.p[3] = xe.p[1];

        r = E*zm-fn-F*p;
}

// constante matrices
void    INIT(void) {

        K1 = B*invD*Dt*V;
        K2 = V*(I-D*invD*Dt*V);
        K3 = invD*Dt*V*E;
}

// sturing
void    C_law(void) {
        u.p[0] = 200*xt.p[0];
}

// draait het experiment
void    run(void) {
        int     i;

        INIT();

        exp_init(z,u,T,wait);

        ts_get_all();           /*haal eerste referentie data */
printf("n: %6.2d\n",n);
        for (i=0; i<n+1; i++) {
                PRED();         /* Eulerstap                    */
                exp_get();      /* haal z                       */
                C_law();        /* sturing                      */
                exp_put();      /* geeft u                      */
                CORR();         /* Runge-Kutta                  */
                ts_put_all();   /* geeft tijdreeksen            */
                ts_get_all();   /* haalt volgende referentie data */

        }
        exp_fini();
                printf("Wait counts: min=%3.0lf, max=%3.0lf, mean=%3.1lf\n",
                wait[0],wait[1],wait[2]);
}

void    epilog(void) {
        ml_open("out_filt",1);
        ml_put_vec(sys,2,"sys");
        ml_put_vec(wait,3,"wait");

        ts_save_all(1);
        ml_close();
```

```
}

int     main(void) {
        sys[0] = sys[1] = 0;
        b = 2.7;

        /* haal data van Matlab                                    */

        ml_open("in_filt",0);
        ml_get_vec(sys,2,"sys");
        T = sys[0];
        n = (int)(sys[1]/T+.5);          /* tijdstappen  */
        ts_load(xt,"xt");
        eps1=ml_get_scalar("eps1",0.000001);
        m1=ml_get_scalar("m1",4.7);
        m2=ml_get_scalar("m2",0.39);
        gm(K);
        gm(A);gm(At);gm(B);gm(Bt);gm(C);gm(D);gm(Dt);
        gm(F);gm(f);gm(V);gm(L12);gm(invD);gm(invDt);gm(E);
        gm(invE);gm(invF);gm(x);gm(Q);gm(p);gm(ps);gm(y);gm(I);
        gm(Ez);gm(Ax);gm(Bps);gm(Dvp);
        ml_close();

        /* Creer tijdreeks registratie omgeving  */
        tsi(z); tsi(u); tsi(x); tsi(Q); tsi(p);tsi(Qdot);tsi(xdot);tsi(varp);

        run();
        epilog();

        return 0;
}

/*      end of filt02.c    */
```

Bijlage 2b

```
clear;echo off
%=========================================================================
%                  PROGRAMMA experiment
%=========================================================================

%VARIABELEN

load in_filt

dT=0.04;
Teind=1-dT;t=(0:dT:Teind);
%Teind is precies de periode tijd v.h. ref. signaal
sys=[dT Teind]; %nodig in experiment

% MODEL

%gegevens t.b.v sturing

amp=0.1;f=0.05;w=f*2*pi;

% !!!!! LET OP !!!!! GEBRUIK TRANS !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
xt=[amp*cos(w*t);-amp*w*sin(w*t);-amp*w*w*cos(w*t)]';
% !!!!! LET OP !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

keuze=input('#0 all, #1: friction, #2: controller, #3: kalman ,#4: break  ');
if keuze==0, kalm3;cont_st;fric_st; elseif keuze==1, fric_st; elseif keuze==2,
elseif keuze==3, kalm3; elseif keuze==4, break; end

% save nodige variabelen

disp('De huidige eindtijd voor het experiment is:'),disp(Teind)
T = input('geef indien nodig nieuwe eindtijd:    ');
if T ~=Teind,
   Teind = T; sys = [dT Teind];
   disp('nieuwe eindtijd is'),disp(Teind)
end;

sys,pause

%  parameters

m1=4.7;
m2=0.39;
b=2.7;

eps1=1e-6;

% matrices

A=[0 0 1 0 0
   0 0 0 1 0
   0 0 0 0 0
   0 0 0 0 0
   0 0 0 0 0];
At=A';
B=[0 0 0
   0 0 0
   0 1 0
   0 0 1
   0 0 0];
Bt=B';
C=[zeros(2,5)
   1 0 0 0 0
   0 1 0 0 0
```

```
   0 0 0 0 0];
D=[0 m1 0
  -1 0 m2
   0 0 0
   0 0 0
   1 0 0];
Dt=D';

F=D;
f=zeros(5,1);

% weegmatrices

V=[1e5 0 0 0 0
   0 1e5 0 0 0
   0 0 1e5 0 0
   0 0 0 1e5 0
   0 0 0 0 1e5]

W=1e3*[1 0 0 0 0
   0 1 0 0 0
   0 0 1 0 0
   0 0 0 1 0
   0 0 0 0 1e-9]

invD=inv(D'*V*D);
invDt=invD';
E=[0 0 0
   0 0 0
   1 0 0
   0 1 0
   0 0 1];

invE=inv(E'*E)*E';
invF=inv(F'*F)*F';

I=eye(5);

Ez=[0;0;0;0;0];
Ax=[0;0;0;0;0];
Bps=[0;0;0;0;0];
Dvp=[0;0;0;0;0];

%    beginwaardes

m0=[0;0;0];
x=[0 0 0 0 978]';
invR0=[0 0 0 0 0
       0 0 0 0 0
       0 0 0 0 0
       0 0 0 0 0
       0 0 0 0 .1];
Q=invR0;
```

```
f0=[x(1)*x(5)-x(2)*x(5)+b*x(3)-b*x(4)
   -x(1)*x(5)+x(2)*x(5)-b*x(3)+b*x(4)
         x(1)
         x(2)
         0              ];

L12=B*(inv(D'*V*D))*B'+inv(W);

% beginschatting p
```

```
  p=(inv(F'*V*F))*F'*V*(E*m0-f0);
  p0=p;
  ps=p;

% beginschatting y (array in integratie-methode)

    y0=[Q(1,1);Q(1,2);Q(1,3);Q(1,4);Q(1,5);Q(2,2);Q(2,3);Q(2,4);Q(2,5)
        Q(3,3);Q(3,4);Q(3,5);Q(4,4);Q(4,5);Q(5,5);x(1);x(2);x(3);x(4);x(5)];
    y=y0;

% residuen

 fn=[x(1)*x(5)-x(2)*x(5)+b*x(3)-b*x(4)
   -x(1)*x(5)+x(2)*x(5)-b*x(3)+b*x(4)
         x(1)
         x(2)
         0              ];
Fn=[0 m1 0
   -1 0 m2
    0 0 0
    0 0 0
    1 0 0];

    r=E*m-fn-Fn*p;

% save gegevens

save in_filt
clear

%EXPERIMENT

!filt01
clear;echo off

%RESULTATEN

rest_fil
end
```

Bijlage 3a

```
/*      filt05.c
        =====
Nathan v/d Wouw // Rene v d Molengraft.
Uses type mat.

*/

#include "tce.h"
#include "mat.h"
#include "exp.h"
#include <stdio.h>      /*Standaard input/output functies       */
#include <math.h>

/*      Globale declaraties                                     */
/*      In deze implementatie moeten alle globale variabelen hieronder   */
/*      globaal gedeclareerd worden.                                */

mat     u(1,1),us(1,1),uc(1,1),x(5,1),xe(3,1),xt(3,1),z(2,1);
int     n;                      /* aantal acties              */
double  T;                      /* Sample tijd [s]            */
double  sys[2];                 /* van IN2.MAT                */
double  wait[3] = { 0, 0, 0 };  /* Wait counts                */

mat     invE(2,3),F(3,2),invF(2,3),A(3,3),At(3,3),B(3,2),C(3,3),Ct(3,3),
        L11(3,3),L12(3,3),L11t(3,3),L21(3,3),invD(2,2),Dt(2,3),D(3,2),V(3,3),
        Bt(2,3),invDt(2,2),E(3,2),Qdot(3,3);

mat     f(3,1),pl(2,1),pl_old(2,1),zl(2,1),zl_old(2,1),
        ps(2,1),zm(2,1),varp(2,1),xdot(3,1);

mat     yp(20,1),f1(9,1),f2(9,1),p(2,1),xdot1(3,1),xdot2(3,1),xp(3,1),
        zml(2,1),Ez(3,1),Ax(3,1),Bps(3,1),Dvp(3,1),r(3,1),fn(3,1);
mat     Q(3,3),K1(3,3),K2(3,3),K3(2,2),I(3,3),Qdot1(3,3),Qdot2(3,3),Qp(3,3);
mat     xd(1,1),xdp(1,1),xdpp(1,1),L(1,2);
double  b,eps1,m,Ws,Aq;
int     k;

// shorthands (afkortingen)

#define tsi(who)        ts_init(who,n,#who)
#define gm(who)         who = ml_get_mat(#who)
#define pm(who)         ml_put_mat(who,#who)
#define ms(who)         mat_show(who)

//bepaling afgeleiden en nieuwe schatting p

void    DVS(void) {
        // Prediction of the state

        //   bepaling van de richting v/d wrijvingskracht
        Ws = 0;
        if (x.p[1] != 0){
        Ws= (x.p[1])/(fabs(x.p[1]));
        }


        f.p[0]  = x.p[2]*Ws;
        f.p[1]  = x.p[0];

        //  bepaling nominale waardes p,z (t.b.v. linearisatie)
        pl_old  = ps;
        zl_old  = invE*(F*pl_old+f);
        pl      = invF*(E*zl_old - f);
        zl      = invE*(f + F*pl);
```

```
        k = 2;
        while (( ( (pl.p[0] - pl_old.p[0]) >= eps1 ) ||
        ( (pl.p[1] - pl_old.p[1]) >= eps1 )   ||
        ( (pl.p[1] - pl_old.p[1]) >= eps1 ) ||
        ( (zl.p[0] - zl_old.p[0]) >= eps1 ) ||
        ( (zl.p[1] - zl_old.p[1]) >= eps1 )   ||
        ( (zl.p[1] - zl_old.p[1]) >= eps1 ) )  && (k <= 4) ) {
        k = k + 1;                                  // beveiliging
        pl_old  = pl;
        zl_old  = zl;
        pl      = invF*(E*zl_old - f);
        zl      = invE*(f + F*pl);
        }

        C.p[6]  =  Ws;

        Ct      = tran(C);

        L11  = A-K1*C;
        L21  = Ct*K2*C;
        L11t = tran(L11);

        // metingen: z u

        zm.p[0] = z.p[1];
        zm.p[1] = u.p[0];

        varp = K3*(zm-zl);
        ps   = pl + varp;

        zml =zm - zl;

        Ez.p[1] = zml.p[0];
        Ez.p[2] = zml.p[1];

        Ax.p[0] = x.p[1];

        Bps.p[1] = ps.p[1];

        Dvp.p[0] = -varp.p[0]+m*varp.p[1];
        Dvp.p[2] = varp.p[0];

        //  bepaling afgeleides Q,x
        Qdot = L12 + (2*L11 - Q*L21)*Q;
        Qdot = 0.5*(Qdot + tran(Qdot)); (waarborging symmetrie)
        xdot = Ax + Bps + Q*(Ct*(V*(Ez - Dvp)));

}

// De Eulerstap

void    PRED(void) {

        xp = x;
        Qp = Q;
        DVS();
        xdot1 = xdot;
        Qdot1 = Qdot;
        x += T*xdot1;
        Q += T*Qdot1;
        xe = x;
}

void    CORR(void) {
```

```
        DVS();
        xdot2 = xdot;
        Qdot2 = Qdot;
        x = xp + T*(0.5*(xdot1 + xdot2));
        Q = Qp + T*(0.5*(Qdot1 + Qdot2));
        p = ps;


        fn.p[0] = xe.p[2]*Ws;
        fn.p[1] = xe.p[0];
        fn.p[2] = 0;

        r = E*zm-fn-D*p;
}

// constante matrices

void    INIT(void) {

        K1 = B*invD*Dt*V;
        K2 = V*(I-D*invD*Dt*V);
        K3 = invD*Dt*V*E;
}

// PD-regelwet

void    C_law(void) {

  us.p[0] =m*xdpp.p[0] + x.p[2]*Ws;
  uc.p[0] = L.p[0]*xd.p[0] + L.p[1]*xdp.p[0] -(L.p[0]*x.p[0] + L.p[1]*x.p[1]);
  u.p[0] = us.p[0] + uc.p[0];
}

// Draait het experiment

void    run(void) {

        int     i;

        INIT();

        exp_init(z,u,T,wait);

        ts_get_all();           /*haal eerste referentie data   */
printf("n: %6.2d\n",n);
        for (i=0; i<n+1; i++) {
                PRED();
                exp_get();      /* haal z                       */
                C_law();        /* regelwet                     */
                exp_put();      /* geeft u                      */
                CORR();
                ts_put_all();   /* geeft alle tijdreeksen       */
                ts_get_all();   /* haal volgende referentie data*/
        }
        exp_fini();
                printf("Wait counts: min=%3.0lf, max=%3.0lf, mean=%3.1lf\n",
                wait[0],wait[1],wait[2]);
}

void    epilog(void) {
        ml_open("out_filt",1);
        ml_put_vec(sys,2,"sys");
        ml_put_vec(wait,3,"wait");
```

```
        ts_save_all(1);
        ml_close();
}

int     main(void) {
        sys[0] = sys[1] = 0;

        /* haal data van Matlab                          */

        ml_open("in_filt",0);
        ml_get_vec(sys,2,"sys");
        T = sys[0];
        n = (int)(sys[1]/T+.5);          /* tijdstappen  */
        ts_load(xt,"xt");
        ts_load(xd,"xd");
        ts_load(xdp,"xdp");
        ts_load(xdpp,"xdpp");
        eps1=ml_get_scalar("eps1",0.000001);
        m=ml_get_scalar("m",0.39);
        gm(K);
        gm(A);gm(At);gm(B);gm(Bt);gm(C);gm(D);gm(Dt);
        gm(F);gm(f);gm(V);gm(L12);gm(invD);gm(invDt);gm(E);
        gm(invE);gm(invF);gm(x);gm(Q);gm(p);gm(ps);gm(I);
        gm(Ez);gm(Ax);gm(Bps);gm(Dvp);gm(L);
        ml_close();

        /* Creer tiijdreeks registratie omgeving  */
  tsi(z); tsi(u); tsi(x); tsi(Q); tsi(p);tsi(r);tsi(xdot2);tsi(xdot1);tsi(xd);
  tsi(xdp);tsi(xdpp);

        run();
        epilog();

        return 0;
}

/*      end of filt05.c    */
```

Bijlage 3b

```
clear;echo off
%=========================================================================
%                  PROGRAMMA experiment
%=========================================================================
%VARIABELEN

load in_filt

dT=0.025;
f=0.5;
Teind=(1/f)-dT;t=(0:dT:Teind+.1*dT);
Teind is precies de periode tijd v.h. ref. signaal
sys=[dT Teind]; %nodig in experiment

% MODEL

% gegevens t.a.v. gewenste baan

amp=0.2;w=f*2*pi;
Aq=1;

% !!!!! LET OP !!!!! GEBRUIK TRANS !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
xt=[amp*cos(w*t);-amp*w*sin(w*t);-amp*w*w*cos(w*t)]';
xd=[Aq*sin(w*t)]';
xdp=[Aq*w*cos(w*t)]';
xdpp=[-(w^2)*Aq*sin(w*t)]';
% !!!!! LET OP !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

keuze=input('#0 all, #1: friction, #2: controller, #3: kalman ,#4: break  ');
if keuze==0, kalm3;cont_st;fric_st; elseif keuze==1, fric_st; elseif keuze==2,
elseif keuze==3, kalm3; elseif keuze==4, break; end

% save nodige variabelen

disp('De huidige eindtijd voor het experiment is:'),disp(Teind)
T = input('geef indien nodig nieuwe eindtijd:    ');
if T ~=Teind,
   Teind = T; sys = [dT Teind];
   disp('nieuwe eindtijd is'),disp(Teind)
end;

sys,pause
%    systeemparameters

m=5.09;

%********** regeling ********

AA=[0 1
    0 0];
BB=[0
    (1/m)];
QQ=[7.5*(5/Aq)^2 0
    0 (5/(Aq*w))^2];
RR=0.01*(1/(Aq*w^2))^2;
[L,S]=lqr(AA,BB,QQ,RR);

%********** filter**********

eps1=1e-6;


% matrices
```

```
A=[0 1 0
   0 0 0
   0 0 0];
At=A';
B=[0 0
   0 1
   0 0];
Bt=B';
C=[0 0 0
   1 0 0
   0 0 0];
D=[-1 m
   0 0
   1 0];
Dt=D';

F=D;
f=zeros(3,1);

V=[1e-7 0 0
   0 1e-1 0
   0 0 1e-7]

W=1e3*[1e3 0 0
   0 1e6 0
   0 0 1e-9]

invD=inv(D'*V*D);
invDt=invD';
E=[0 0
   1 0
   0 1];

invE=inv(E'*E)*E';
invF=inv(F'*F)*F';

I=eye(3);

Ez=[0;0;0];
Ax=[0;0;0];
Bps=[0;0;0];
Dvp=[0;0;0];

%    beginwaardes

m0=[0;0];
x=[0 0 10]';
invR0=[0 0 0
       0 0 0
       0 0 1e8];
Q=invR0;

f0=[x(3)*sign(x(2))
         x(1)
         0        ];

L12=B*(inv(D'*V*D))*B'+inv(W);

% beginschatting p

  p=(inv(F'*V*F))*F'*V*(E*m0-f0);
  p0=p;
  ps=p;
```

```
% residuen op schatting na Eulerstap

f0=[x(3)*sign(x(2))
          x(1)
          0          ];
Fn = F;
r=E*m0-f0-Fn*p;

% save gegevens
save in_filt

clear

%EXPERIMENT

!filt01
clear;echo off

%RESULTATEN

rest_fil
end
```