Evenwichtshandhaving

Stageverslag

" EVENWICHTSHANDHAVING "

Een modellering m.b.v. het multibody-pakket DADS

In opdracht van:
Pauline Evers & Mariëlle Snijders
Buro BMGT, T.U. Eindhoven

Onder begeleiding van:
Ir. H Giesen & dr.ir. A Sauren

Geschreven door:
Henrie de Haas
Student WMT, id.nr. 234075
Faculteit Werktuigbouwkunde
Technische Universiteit Eindhoven
Zomer/Herfst 1991

Rapport WFW nr. 92.004

" EVENWICHTSHANDHAVING "

Een modellering m.b.v. het multibody-pakket DADS

Met vallen en opstaan
Word je groot
En leer je
Leven

Samenvatting

In het onderzoek naar het functioneren van het evenwichtssysteem bij ouderen, uitgevoerd door twee bewegingswetenschappers van de Rijks Universiteit Limburg, bleek er behoefte te zijn aan een computermodel.
In dit model is de mens gemodelleerd als een staande slinger met rotatieveren en -dempers in de scharnieren. De verstoring van het evenwicht vindt plaats door de voor te schrijven verplaatsing op voetniveau volgens een scheve-sinus-functie.

Een enkelslingermodel kan het gemeten gedrag niet goed genoeg benaderen, dus is er verder gewerkt met het dubbelslingermodel met scharnieren op enkel- en heupniveau. De slingerdelen zijn als homogene balken gemodelleerd en in beide scharnieren zijn zowel rotatieveren en rotatiedempers aangebracht.
Het dubbelslingermodel is geïmplementeerd in het multibody programma-pakket DADS vanwege de overzichtelijkheid en de eenvoud om het werkelijke experiment met behulp van een DADS-model te simuleren.
Gezien het feit dat DADS niet geschikt is voor een parameterschatting is het dubbelslingermodel tevens in PC-Matlab gemodelleerd.
Beide modellen leveren dezelfde responsies op.

Het PC-Matlab-model is gebruikt om de invloed van modelparameters, zoals veerstijfheden, dempingsconstanten, massa's en massaverdeling te bepalen. Tevens is dit model gebruikt om de invloed van de 2^e en hogere orde termen in de bewegingsvergelijkingen te bepalen.
Met behulp van het DADS-model zijn er "meetgegevens" gegenereerd waarmee de PC-Matlab-routines voor parameterschatting getest zijn.

Het blijkt dat de hogere-orde termen nauwelijks invloed hebben op het modelgedrag.
De PC-Matlab-routines voor parameterschatting zijn niet geschikt om voor dit dubbelslingermodel tot betrouwbare resultaten te komen.

Ondanks de onbetrouwbaarheid in de werkelijke meetgegevens en de daaruit berekende modelgegevens rijst het vermoeden dat een tripelslingermodel tot een realistischere weergave van de werkelijkheid leidt.
Beschrijving van dat model in PC-Matlab en gebruik van een Kalman-filter of een geschikte fitprocedure zou tot een betrouwbare parameterschatting kunnen leiden.
De uit meerdere (nauwkeurigere) metingen bepaalde parameters maken het DADS-model dan een betrouwbaar simulatiemodel.

Voorwoord

Deze stageopdracht heb ik gedurende de zomer 1991 gedaan op de Technische Universiteit Eindhoven. De opdracht maakt deel uit van het afstudeerwerk van de twee bewegingswetenschappers, die in opdracht van buro BMGT (Biomedische en Gezondheids Techniek) onderzoek doen naar de evenwichtshandhaving van ouderen. Doel is om het door hun uitgevoerde pilot-experiment met een computermodel te simuleren.
Het beschrijven van dat model en het modelleren in DADS is in goede samenwerking met de bewegingswetenschappers Pauline Snijders en Mariëlle Evers, hun begeleider Maurice Heijnen en mijn begeleiders Hub Giesen en Fons Sauren vlot verlopen. Ik wil hen hierbij bedanken voor de hulp en samenwerking.
De benodigde schatting van de modelparameters heb ik uitgevoerd met behulp van PC-Matlab. Voor de hulp en tips bij het gebruik van PC-Matlab en de optimization toolbox daarin wil R. Huisman en B. de Jager bedanken.
Ondanks de prettige samenwerking en goede begeleiding bleek het fitten van de meetgegevens een groter probleem te zijn dan vooraf gedacht. Het gewenste doel heb ik dan ook niet bereikt, namelijk een goed werkend simulatiemodel opstellen. Toch hoop ik het inzicht in en het gedrag van het simulatiemodel op een bruikbare manier beschreven te hebben, zodat de opdracht alsnog voltooid kan worden.
Mogelijk heeft de samenwerking een positieve bijgedrage geleverd aan het afstudeerwerk van Pauline en Mariëlle, doordat het opstellen van een simulatiemodel om een meer abstracte probleembenadering vraagt.
Mijn eventuele opvolger in het opstellen van een werkbaar simulatiemodel wil ik veel succes toewensen, waarbij dit stageverslag hem op de goede weg zou kunnen helpen.

Inhoudsopgave

Hoofdstuk 1 **Inleiding**

Deze stage-opdracht maakt deel uit van een onderzoek naar het functioneren van het evenwichtssysteem bij ouderen. Dat onderzoek wordt uitgevoerd als afstudeeropdracht door twee bewegingswetenschappers van de Rijks Universiteit Limburg. Het onderzoek bestaat uit twee delen, te weten een literatuuronderzoek en een pilot-experiment.

1. Literatuuronderzoek

Op basis van eerder verricht literatuuronderzoek naar de gangbare theorieën en opvattingen met betrekking tot het fysiologisch functioneren van het evenwichtssysteem in het algemeen, wordt het nu toegespitst op de oudere mens.
Allereerst wordt nagegaan welke veranderingen als gevolg van toenemende leeftijd binnen de afzonderlijke deelsystemen, te weten het visuele, het vestibulaire, het proprioceptieve en het oppervlakkig sensibele systeem optreden. Daarnaast vindt er een inventarisatie plaats van het reeds verrichte onderzoek op het gebied van ouderen en hun evenwicht.

Uit de theoretische concepten over het functioneren van het evenwichtssysteem bij ouderen wordt een hypothetisch model ontwikkeld waaruit toetsbare stellingen met betrekking tot de proprioceptie worden afgeleid. De stellingen worden getoetst in een pilot-experiment.

2. Pilot-experiment

Doelstelling: Het ontwikkelen van een experimenteel instrumentarium, waarmee inzicht wordt verkregen in het functioneren van het evenwichtssysteem bij ouderen, met nadruk op de bijdrage van het proprioceptieve systeem.

Om enig inzicht te krijgen in het functioneren van het ontwikkelde instrumentarium nemen aan het pilot-experiment proefpersonen van middelbare leeftijd deel.

Probleemstelling: Welk sensorisch systeem levert de primaire bijdrage aan de evenwichtshandhaving bij een horizontale verstoring op het niveau van de voeten?

Vraagstellingen:
1. *Op welk tijdstip en bij welke verstoring vindt er een verplaatsing van het hoofd plaats?*
2. *Op welk tijdstip en bij welke verstoring treedt er EMG-activiteit op in de spieren van het onderbeen?*
3. *Is het mogelijk om op basis van de vraagstellingen 1 en 2 antwoord te geven op de vraag of een gesignaleerde hoofdbeweging een passieve reactie is op de verstoring op voetniveau, danwel veroorzaakt wordt door een zenuwsignaal vanuit de onderste extremiteiten?*
4. *Kan de monosynaptische strekreflex een functionele bijdrage leveren aan de handhaving van het evenwicht?*

Methode: Proefpersonen van middelbare leeftijd staan op een hydraulisch in voor- en achterwaartse richting te activeren platform. De verstoring heeft een maximale versnelling van 3 m/s^2 en een maximale amplitude van 0.05 m. Rondom het platform wordt een buizenstelsel geplaatst om de veiligheid van de proefpersonen te garanderen.

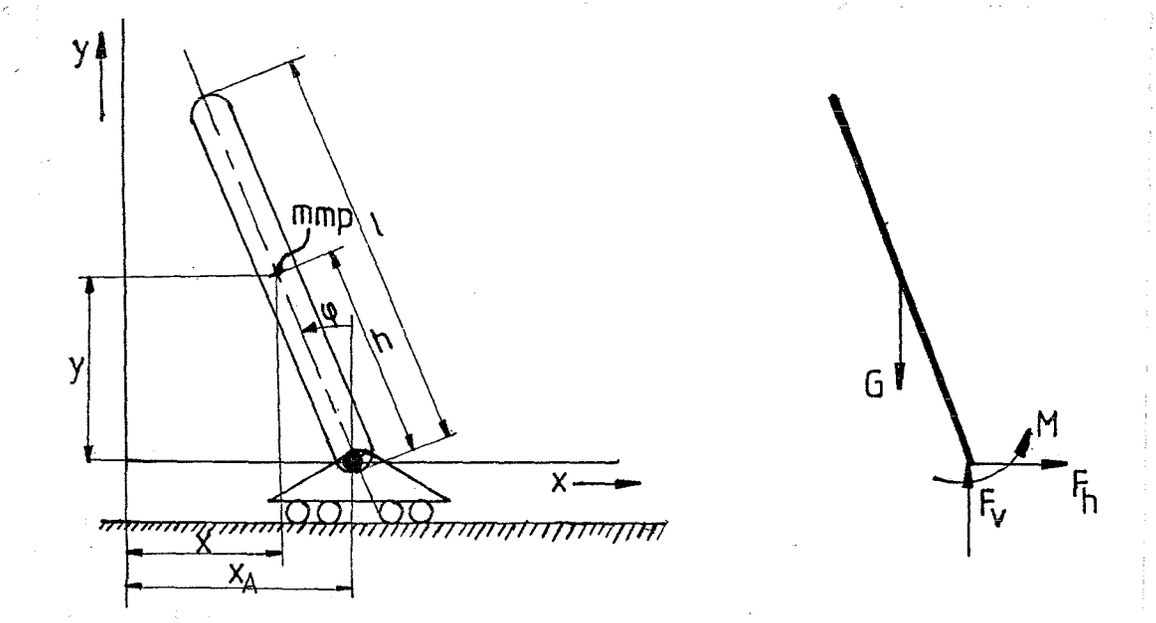
Meetapparatuur: - EMG-apparatuur
- versnellingsopnemers
- video-apparatuur

Een computersimulatie met behulp van het programmapakket DADS is nodig om een idee te krijgen over de ordegrootte van de verstoringen en optredende lichamelijke reacties. Het opstellen van dat computermodel en het implementeren in DADS is het doel van deze stage.

Op grond van de metingen aan een eerste proefpersoon dient het model in overeenstemming gebracht te worden met de experimentele resultaten. Hiertoe moeten de modelparameters geschat worden, zodanig dat er met het model realistische simulaties uitgevoerd kunnen worden.
Dit kan dan geverifieerd worden aan de hand van andere reeksen resultaten. Indien volstaan kan worden met één set parameters, is het simulatiemodel algemeen bruikbaar.

Hoofdstuk 2 ***De enkelslinger***

Het enkelslingermodel is in de onderstaande figuur weergegeven, zowel in samenstelling alsook in losgesneden toestand met de erop werkende krachten en momenten. De bewegingsvergelijking van dit model zal in paragraaf 2.1 afgeleid worden. In paragraaf 2.2 wordt de modellering met behulp van het programmapakket DADS beschreven. In paragraaf 2.3 worden de modelgegevens met de meetgegevens vergeleken.



figuur 2.1 Het enkelslingermodel

2.1 <u>De bewegingsvergelijking</u>

De tweede wet van Newton en de impulsmoment vergelijking ten opzichte van het massamiddelpunt zien er als volgt uit:

$$F_h = m * \ddot{x}$$

$$F_v - m * g = m * \ddot{y} \qquad (1)$$

$$h * \cos\varphi * F_h + h * \sin\varphi * F_v + M = J * \ddot{\varphi}$$

Volgens figuur 2.1 met voor M en G ingevuld:

$$G = m * g$$

$$M = -b * \dot{\varphi} - k * \varphi \qquad (2)$$

Voor de positie, snelheid en versnelling van het massamiddelpunt geldt:

$$x = x_A - h * \sin\varphi$$

$$\dot{x} = \dot{x}_A - h * \dot{\varphi} * \cos\varphi$$

$$\ddot{x} = \ddot{x}_A - h * (\ddot{\varphi} * \cos\varphi - \dot{\varphi}^2 * \sin\varphi) \qquad (3)$$

$$y = h * \cos\varphi$$

$$\dot{y} = -h * \dot{\varphi} * \sin\varphi$$

$$\ddot{y} = -h * (\ddot{\varphi} * \sin\varphi + \dot{\varphi}^2 * \cos\varphi)$$

Hiermee volgt voor F_h en F_v:

$$F_h = m * (\ddot{x}_A - h * (\ddot{\varphi} * \cos\varphi - \dot{\varphi}^2 * \sin\varphi)) \qquad (4)$$

$$F_v = m * g - m * h * (\ddot{\varphi} * \sin\varphi + \dot{\varphi}^2 * \cos\varphi)$$

De bewegingsvergelijking ontstaat door eliminatie van F_h en F_v:

$$(J + m * h^2) * \ddot{\varphi} + b * \dot{\varphi} + k * \varphi - m * g * h * \sin\varphi = m * h * \cos\varphi * \ddot{x}_A \qquad (5)$$

Voor het lineariseren van de bewegingsvergelijking rond de (instabiele) evenwichtsstand geldt:

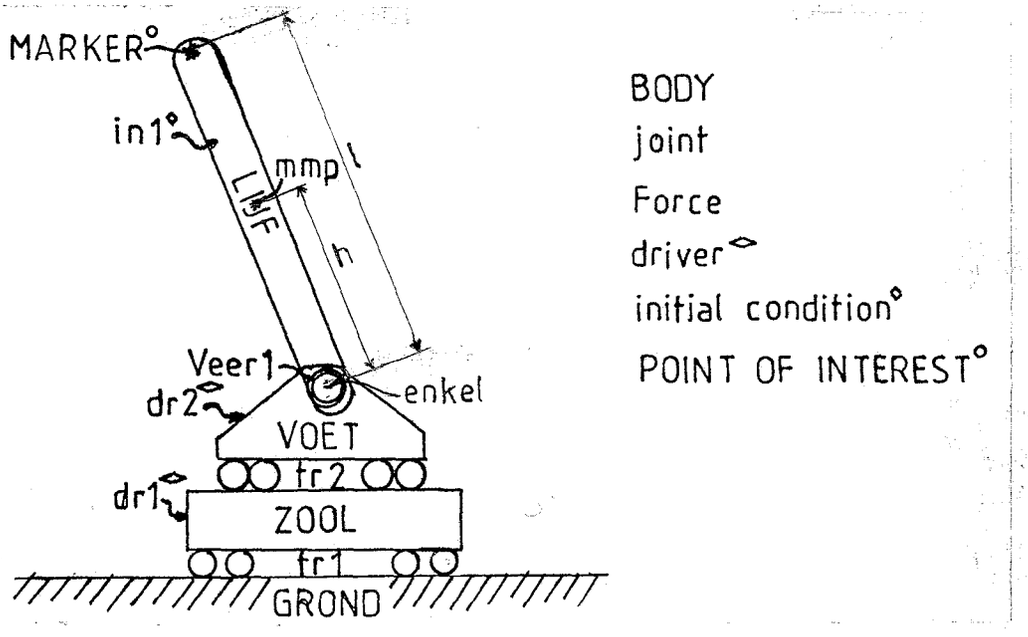
$$\varphi \ll 1$$
$$\sin\varphi \approx \varphi \qquad (6)$$
$$\cos\varphi \approx 1$$

Daarmee luidt de gelineariseerde bewegingsvergelijking:

$$(J + m * h^2) * \ddot{\varphi} + b * \dot{\varphi} + (k - m * g * h) * \varphi = m * h * \ddot{x}_A \qquad (7)$$

2.2 De modellering m.b.v. DADS

De enkelvoudige slinger is in DADS heel eenvoudig te modelleren.
Het ingangssignaal is in dit geval x_o de verplaatsing van de voet. Het signaal is een scheve sinusfunctie (optelling van een sinusfunctie en een polynoom), die niet rechtstreeks in DADS te modelleren is. De scheve sinusfunctie is daarom opgesplitst in zijn harmonisch en polynomisch deel. Door twee dummy massa's te definiëren en toepassen van twee drivers is het ingangssignaal te modelleren. Op de ene dummy staat een driver die het harmonische deel van de scheve sinusfunctie vertegenwoordigt en op de andere dummy staat een driver die het polynomisch deel representeert. Beginpositie is een rechtopstaande slinger in rust. Als begincondities gelden daarom: de hoek van het slingerlichaam ten opzichte van de vaste wereld en de hoeksnelheid van dat lichaam zijn 0.
Met de juiste veerstijfheid en dempingsconstante van de veer en demper in het enkelgewricht zou de gemeten responsie in het model gerepresenteerd kunnen worden.



figuur 2.2 Het DADS dubbelslingermodel

Het model bestaat uit 4 body's, de vaste wereld, het slingerlichaam en de twee dummy massa's. Het polynoom-deel van de scheve sinus staat op de body genaamd VOET en het harmonische deel staat op de body genaamd ZOOL.
De modelgegevens zijn afkomstig van metingen aan de proefpersoon en berekende waarden, die met behulp van het programma GEBOD verkregen zijn.
Het slingerlichaam wordt vanwege de eenvoud benaderd met een homogene balk.

Body LIJF:
massa m = 63 kg
lengte l = 1.68 m
hoogte zwaartepunt h = 0.84 m
massatraagheidsmoment $J = 1/12*m*l^2 = 14.82\ kgm^2$

Zowel body VOET als ZOOL hebben een verwaarloosbare massa en massatraagheidsmoment, body GROND is de vaste wereld.
Zie ook de DADS invoerfile in bijlage 5.

2.3 Het model versus de meetgegevens

Het ingangssignaal dat in het experiment gebruikt is, ziet er als volgt uit:

$$x_A = V*t - A*\sin(\omega*t)$$

$$V = a_{max} / \omega$$

$$A = V / \omega$$

$$\omega = \sqrt{(2*\pi*a_{max} / s_{max})} \qquad (8)$$

$$a_{max} = 3\ m/s^2$$

$$s_{max} = 0.04\ m$$

$$x_A = 0.1382*t - 0.00637*\sin(21.71*t)$$

Een schatting voor de veerstijfheid wordt gevonden uit een beschouwing van de statische evenwichtsstand van het menselijk lichaam, aangenomen dat de lijn door het zwaartepunt van de mens en de enkel 5 graden verdraaid is ten opzichte van de loodlijn door de enkel op de voetzool (zie bijvoorbeeld het Polytechnisch Zakboekje, onderdeel antropometrie).

$$1/2*l*\sin\varphi_{st}*m*g = k*\varphi_{st}$$

Hieruit volgt dus voor de veerstijfheid: (9)

$$k = 1/2*l*\sin\varphi_{st}*m*g / \varphi_{st} = 518.4\ N/rad$$

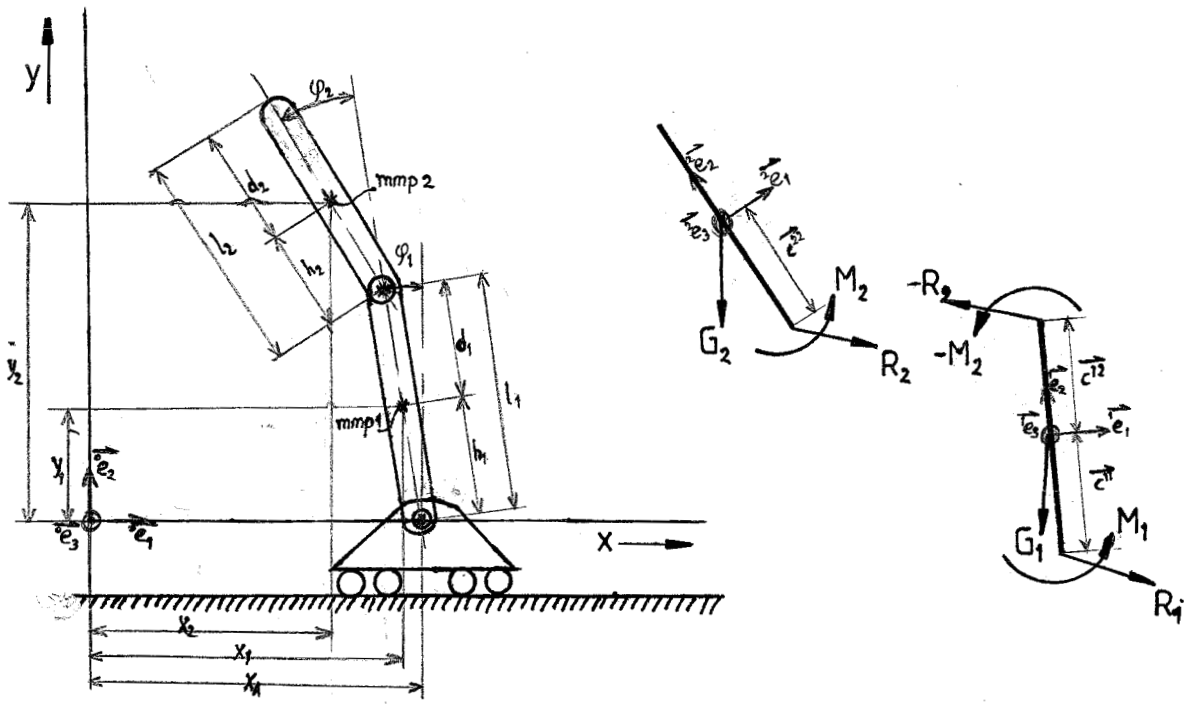
Na deze waarde ingevuld te hebben voor de veerstijfheid van de enkel is de horizontale en vertikale verplaatsing van de marker met behulp van het programma DADS uitgerekend.

Het resultaat van deze berekening, zie bijlage 2, komt niet overeen met de meetwaarden, zie bijlage 1. De berekende verplaatsingen zijn kleiner dan de gemeten waarden. Verkleining van de veerstijfheid of vergroting van de dempingsconstante zou een overeenkomst tussen gemeten en berekende verplaatsing op kunnen leveren.

Deze manier van zoeken naar een passend model (de trial and error methode) is niet de meest voor de hand liggende methode om het model te fitten op de praktijkgegevens.

Na bespreking met alle betrokkenen, wordt geconcludeerd dat het enkelvoudige slingermodel niet het meest geschikte model is voor een goede beschrijving van de praktijk. Er wordt besloten om verder te werken met het dubbelslingermodel, waarbij de verbinding tussen hoofd en romp star verondersteld wordt.

Het model van de dubbelslinger is in de onderstaande figuur weergegeven, zowel in overzicht alsook losgesneden met invoering van (snede-) krachten en momenten. De bewegingsvergelijking van dit model wordt in paragraaf 3.1 afgeleid. In paragraaf 3.2 zal dan de modellering in het programmapakket DADS beschreven worden, waarna in paragraaf 3.3 het model in PC-Matlab beschreven wordt. In paragraaf 3.4 zullen beide model implementaties met elkaar vergeleken worden.



figuur 3.1 Het dubbelslingermodel

3.1 De bewegingsvergelijkingen

De zwaartekrachten en momenten zijn als volgt t.o.v. de inertiaalbasis te schrijven:

$$\overline{G_1} = m_1 * \begin{bmatrix} 0 \\ -g \\ 0 \end{bmatrix}$$

$$\overline{G_2} = m_2 * \begin{bmatrix} 0 \\ -g \\ 0 \end{bmatrix} \qquad \textbf{(10)}$$

$$M_1 = -b_1 * \dot{\varphi}_1 - k_1 * \varphi_1$$

$$M_2 = -b_2 * \dot{\varphi}_2 - k_2 * \varphi_2$$

Voor de posities van de massamiddelpunten t.o.v. de inertiaalbasis en de afgeleiden daarvan geldt het volgende:

$$\overline{r_1} = \begin{bmatrix} x_A - h_1 * \sin\varphi_1 \\ h_1 * \cos\varphi_1 \\ 0 \end{bmatrix}$$

$$\overline{\dot{r}_1} = \begin{bmatrix} \dot{x}_A - h_1 * \dot{\varphi}_1 * \cos\varphi_1 \\ -h_1 * \dot{\varphi}_1 * \sin\varphi_1 \\ 0 \end{bmatrix}$$

$$\overline{\ddot{r}_1} = \begin{bmatrix} \ddot{x}_A + h_1 * (\dot{\varphi}^{2_1} * \sin\varphi_1 - \ddot{\varphi}_1 * \cos\varphi_1) \\ -h_1 * (\dot{\varphi}^{2_1} * \cos\varphi_1 + \ddot{\varphi}_1 * \sin\varphi_1) \\ 0 \end{bmatrix}$$

$$\overline{r_2} = \begin{bmatrix} x_A - l_1 * \sin\varphi_1 - h_2 * \sin(\varphi_1 + \varphi_2) \\ l_1 * \cos\varphi_1 + h_2 * \cos(\varphi_1 + \varphi_2) \\ 0 \end{bmatrix} \qquad \textbf{(11)}$$

$$\overline{\dot{r}_2} = \begin{bmatrix} \dot{x}_A - l_1 * \dot{\varphi}_1 * \cos\varphi_1 - h_2 * (\dot{\varphi}_1 + \dot{\varphi}_2) * \cos(\varphi_1 + \varphi_2) \\ -l_1 * \dot{\varphi}_1 * \sin\varphi_1 - h_2 * (\dot{\varphi}_1 + \dot{\varphi}_2) * \sin(\varphi_1 + \varphi_2) \\ 0 \end{bmatrix}$$

$$\overline{\ddot{r}_2} = \begin{bmatrix} \ddot{x}_A + l_1 * (\dot{\varphi}^{2_1} * \sin\varphi_1 - \ddot{\varphi}_1 * \cos\varphi_1) \\ - h_2 * (\ddot{\varphi}_1 + \ddot{\varphi}_2) * \cos(\varphi_1 + \varphi_2) + h_2 * (\dot{\varphi}_1 + \dot{\varphi}_2)^2 * \sin(\varphi_1 + \varphi_2) \\ -l_1 * (\dot{\varphi}_1^2 * \cos\varphi_1 + \ddot{\varphi}_1 * \sin\varphi_1) \\ - h_2 * (\dot{\varphi}_1 + \dot{\varphi}_2)^2 * \cos(\varphi_1 + \varphi_2) - h_2 * (\ddot{\varphi}_1 + \ddot{\varphi}_2) * \sin(\varphi_1 + \varphi_2) \\ 0 \end{bmatrix}$$

Gegeven de relaties tussen de verschillende bases volgt voor de rotatiematrices:

$$^0\overrightarrow{e} = \underline{C^1} \; ^1\overrightarrow{e}$$

$$^1\overrightarrow{e} = \underline{C^2} \; ^2\overrightarrow{e} \qquad \textbf{(12)}$$

$$^0\overrightarrow{e} = \underline{C^1} \; \underline{C^2} \; ^2\overrightarrow{e} = \underline{R^2} \; ^2\overrightarrow{e}$$

$$\underline{C^1} = \begin{bmatrix} \cos\varphi_1 & -\sin\varphi_1 & 0 \\ \sin\varphi_1 & \cos\varphi_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(13)

$$\underline{C^2} = \begin{bmatrix} \cos\varphi_2 & -\sin\varphi_2 & 0 \\ \sin\varphi_2 & \cos\varphi_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\underline{R^2} = \underline{C^1} * \underline{C^2} =$$

$$= \begin{bmatrix} \cos\varphi_1 * \cos\varphi_2 - \sin\varphi_1 * \sin\varphi_2 & -\cos\varphi_1 * \sin\varphi_2 - \sin\varphi_1 * \cos\varphi_2 & 0 \\ \sin\varphi_1 * \cos\varphi_2 + \cos\varphi_1 * \sin\varphi_2 & -\sin\varphi_1 * \sin\varphi_2 + \cos\varphi_1 * \cos\varphi_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(14)

Voor de positie in componenten t.o.v. de inertiaalbasis van de scharnierpunten t.o.v. de massamiddelpunten van de slingerlichamen geldt:

$$\overline{c^{11}} = \underline{C^1} * \begin{bmatrix} 0 \\ -h_1 \\ 0 \end{bmatrix} = -h_1 * \begin{bmatrix} -\sin\varphi_1 \\ \cos\varphi_1 \\ 0 \end{bmatrix}$$

$$\overline{c^{12}} = \underline{C^1} * \begin{bmatrix} 0 \\ d_1 \\ 0 \end{bmatrix} = d_1 * \begin{bmatrix} -\sin\varphi_1 \\ \cos\varphi_1 \\ 0 \end{bmatrix} \qquad \textbf{(15)}$$

$$\overline{c^{22}} = \underline{R^2} * \begin{bmatrix} 0 \\ -h_2 \\ 0 \end{bmatrix} = -h_2 * \begin{bmatrix} -\sin(\varphi_1 + \varphi_2) \\ \cos(\varphi_1 + \varphi_2) \\ 0 \end{bmatrix}$$

De bewegingsvergelijkingen volgen uit figuur 3.1 door gebruik van de 2e wet van Newton, de impulsmoment-stelling en eliminatie van de verbindingskrachten R_1 en R_2.

$$1: \; m_1 * \ddot{\vec{r}}_1 = \vec{G}_1 + \vec{R}_1 - \vec{R}_2$$

$$2: \; m_2 * \ddot{\vec{r}}_2 = \vec{G}_2 + \vec{R}_2$$

$$1: \; M_1 \vec{e}_3 - M_2 \vec{e}_3 + \vec{c}^{11} \; X \; \vec{R}_1 - \vec{c}^{12} \; X \; \vec{R}_2 = J_1 * \dot{\omega}_1 \vec{e}_3$$

$$2: \; M_2 \vec{e}_3 + \vec{c}^{22} \; X \; \vec{R}_2 = J_2 * \dot{\omega}_2 \vec{e}_3$$

(16)

$$(M_1 - M_2)\,\vec{e}_3 + m_1 * \vec{c}^{11} X \ddot{\vec{r}}_1 + m_2 * (\vec{c}^{11} - \vec{c}^{12}) X \ddot{\vec{r}}_2 +$$
$$-\vec{c}^{11} X (\vec{G}_1 + \vec{G}_2) + \vec{c}^{12} X \vec{G}_2 = J_1 * \dot{\omega}_1 \vec{e}_3$$

$$M_2 \vec{e}_3 + \vec{c}^{22} X (m_2 * \ddot{\vec{r}}_2 - \vec{G}_2) = J_2 * \dot{\omega}_2 \vec{e}_3$$

(17)

Invullen van de posities van de massamiddelpunten en scharnierpunten in de bewegingsvergelijkingen levert:

$$(m_1 * h_1^2 + m_2 * l_1^2 + m_2 * l_1 * h_2 * \cos\varphi_2 + J_1) * \ddot{\varphi}_1 +$$
$$+ (m_2 * l_1 * h_2 * \cos\varphi_2) * \ddot{\varphi}_2 - m_2 * l_1 * h_2 * \sin\varphi_2 * (\dot{\varphi}_1 + \dot{\varphi}_2)^2 +$$
$$- (m_2 * l_1 + m_1 * h_1) * g * \sin\varphi_1 + b_1 * \dot{\varphi}_1 + k_1 * \varphi_1 - b_2 * \dot{\varphi}_2 +$$
$$-k_2 * \varphi_2 = (m_1 * h_1 + m_2 * l_1) * \cos\varphi_1 * \ddot{x}_A$$

$$(m_2 * h_2 * l_1 * \cos\varphi_2 + m_2 * h_2^2 + J_2) * \ddot{\varphi}_1 + (m_2 * h_2^2 + J_2) * \ddot{\varphi}_2 +$$
$$+ m_2 * h_2 * l_1 * \sin\varphi_2 * \dot{\varphi}_1^2 - h_2 * m_2 * g * \sin(\varphi_1 + \varphi_2) +$$
$$+ b_2 * \dot{\varphi}_2 + k_2 * \varphi_2 = m_2 * h_2 * \cos(\varphi_1 + \varphi_2) * \ddot{x}_A$$

(18)

Met de aanname dat de hoeken klein zijn mogen de cosinussen en sinussen vervangen worden door 1 respectievelijk de hoek zelf.

$$(m_1 * h_1^2 + m_2 * l_1^2 + m_2 * l_1 * h_2 + J_1) * \ddot{\varphi}_1 + m_2 * l_1 * h_2 * \ddot{\varphi}_2 +$$
$$-m_2 * l_1 * h_2 * \varphi_2 * (\dot{\varphi}_1 + \dot{\varphi}_2)^2 - (m_2 * l_1 + m_1 * h_1) * g * \varphi_1 +$$
$$+ b_1 * \dot{\varphi}_1 + k_1 * \varphi_1 - b_2 * \dot{\varphi}_2 - k_2 * \varphi_2 = (m_1 * h_1 + m_2 * l_1) * \ddot{x}_A$$

$$(m_2 * h_2 * l_1 + m_2 * h_2^2 + J_2) * \ddot{\varphi}_1 + (m_2 * h_2^2 + J_2) * \ddot{\varphi}_2 +$$
$$+ m_2 * h_2 * l_1 * \varphi_2 * \dot{\varphi}_1^2 - h_2 * m_2 * g * (\varphi_1 + \varphi_2) +$$
$$+ b_2 * \dot{\varphi}_2 + k_2 * \varphi_2 = m_2 * h_2 * \ddot{x}_A$$

(19)

Indien de snelheden ook klein zijn mogen de niet lineaire termen verwaarloosd worden.
De bewegingsvergelijkingen zien er dan als volgt uit:

$$(m_1 * h_1^2 + m_2 * l_1^2 + m_2 * l_1 * h_2 + J_1) * \ddot{\varphi}_1 +$$
$$m_2 * l_1 * h_2 * \ddot{\varphi}_2 - (m_2 * l_1 + m_1 * h_1) * g * \varphi_1 +$$
$$+ b_1 * \dot{\varphi}_1 + k_1 * \varphi_1 - b_2 * \dot{\varphi}_2 - k_2 * \varphi_2$$
$$= (m_1 * h_1 + m_2 * l_1) * \ddot{x}_A$$

$$(m_2 * h_2 * l_1 + m_2 * h_2^2 + J_2) * \ddot{\varphi}_1 + (m_2 * h_2^2 + J_2) * \ddot{\varphi}_2 +$$
$$b_2 * \dot{\varphi}_2 + k_2 * \varphi_2 - h_2 * m_2 * g * (\varphi_1 + \varphi_2) = m_2 * h_2 * \ddot{x}_A$$

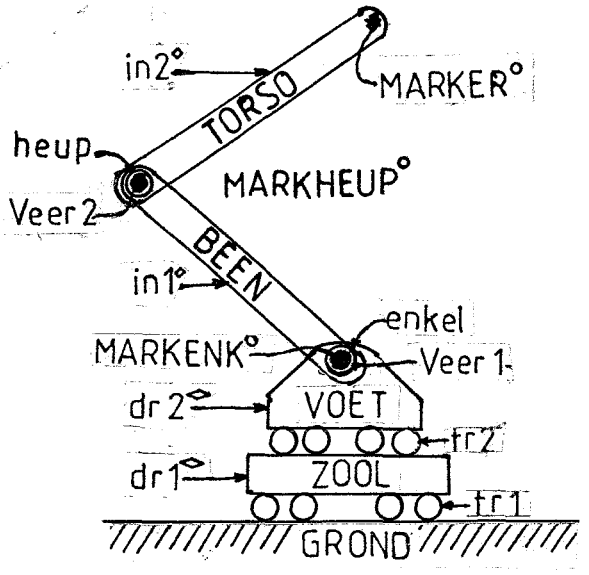
(20)

Zie ook bijlage IV voor een controleberekening m.b.v. Lagrange.

3.2 De modellering m.b.v. DADS

Met behulp van DADS is het mogelijk om een eenvoudig en gebruikersvriendelijk simulatiemodel op te stellen. Veranderingen in gegevens, zoals bijvoorbeeld de massa en lengte van een proefpersoon, zijn gemakkelijk in het DADS-model door te voeren. Daarbij heeft DADS veel mogelijkheden om de resultaten overzichtelijk te presenteren.

Het DADS-model bestaat uit 5 body's, te weten de vaste wereld GROND, de twee slingerlichamen BEEN en TORSO en de dummy massa's VOET en ZOOL. Het ingangssignaal wordt via de twee drivers DR1 en DR2 en de twee dummy massa's op het model overgebracht (op dezelfde manier als bij de enkelslinger). In de scharnierpunten zijn rotatieveren en -dempers aangebracht (Zie figuur3.2).



figuur 3.2 Het DADS dubbelslingermodel

Body BEEN:

massa m_1 = 31.6 kg
lengte l_1 = 0.89 m
*massamiddelpunt h_1 = 1/2*l_1 = 0.445 m*
*massatraagheidsmoment J_1 =1/12*m_1*l_1^2 =2.0859 kgm^2*

Body TORSO:

massa m_2 = 31.4 kg
lengte l_2 = 0.79 m
*massamiddelpunt h_2 = 1/2*l_2 = 0.395 m*
*massatraagheidsmoment J_2 = 1/12*m_2*l_2^2 = 1.6331 kgm^2*

Driver DR1:

type driver: X
*driving function: POLYNOOM ; -0.1382*t*

Driver DR2:

type driver: X-DIFFERENCE
*driving function: HARMONISCH ; -0.00637*sin(21.71*t)*

RSDA VEER1:

veerstijfheid k_1 = 500 Nm/rad
dempingsconstante b_1 = 100 Nms/rad

RSDA VEER2:

veerstijfheid k_2 = 150 Nm/rad
dempingsconstante b_2 = 25 Nms/rad

Het polynomisch deel van het ingangssignaal wordt als negatieve waarde ingevoerd vanwege het feit dat driver DR2 het verschil tussen de beide dummy's voorschrijft. Het signaal wat aan de enkel wordt aangeboden is dus het harmonisch signaal minus het polynomisch signaal.
De body's VOET en ZOOL (dummy's) hebben een verwaarloosbare massa en massatraagheidsmoment, terwijl body GROND verbonden is met de vaste wereld.
De overige modelgegevens zijn gemeten aan de proefpersoon en/of berekend m.b.v. GEBOD. De slingerlichamen worden als homogene balken gemodelleerd.

Zie voor de berekende responsie bijlage 3 en voor de DADS-invoerfiles bijlage 5.

3.3 Responsiebepaling m.b.v. PC-Matlab

Het multibody pakket DADS is niet geschikt om parameterschattingen mee uit te voeren. PC-Matlab heeft daarvoor wel de mogelijkheden. Vandaar dat het dubbelslingermodel ook in PC-Matlab geïmplementeerd is. Om de implementatie mogelijk te maken wordt gebruik gemaakt van de toestandsbeschrijving.

De in paragraaf 3.1 afgeleide gelineariseerde bewegingsvergelijkingen ($\phi << 1$, zie formule 19), die het wiskundige model vormen zijn overzichtelijk in matrix-vorm te schrijven:

$$\underline{A} * \begin{bmatrix} \ddot{\varphi}_1 \\ \ddot{\varphi}_2 \end{bmatrix} + \underline{B} * \begin{bmatrix} \dot{\varphi}_1 \\ \dot{\varphi}_2 \end{bmatrix} + \underline{C} * \begin{bmatrix} \varphi_1 \\ \varphi_2 \end{bmatrix} + \underline{D} * \begin{bmatrix} \varphi_2 * (\dot{\varphi}_1 + \dot{\varphi}_2)^2 \\ \varphi_2 * \dot{\varphi}_1^2 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \qquad \begin{matrix} (21,1) \\ (21,2) \end{matrix}$$

In deze formule geldt voor de matrices $\underline{A}$, $\underline{B}$, $\underline{C}$ en $\underline{D}$

$$\underline{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} J_1 + m_1 h_1^2 + m_2 l_1^2 + m_2 l_1 h_2 & m_2 l_1 h_2 \\ J_2 + m_2 h_2^2 + m_2 l_1 h_2 & J_2 + m_2 h_2^2 \end{bmatrix}$$

$$\underline{B} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} b_1 & -b_2 \\ 0 & b_2 \end{bmatrix} \qquad \textbf{(22)}$$

$$\underline{C} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} = \begin{bmatrix} k_1 - (m_2 l_1 + m_1 h_1) * g & -k_2 \\ -m_2 h_2 g & k_2 - m_2 h_2 g \end{bmatrix}$$

$$\underline{D} = \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix} = \begin{bmatrix} -m_2 l_1 h_2 & 0 \\ 0 & m_2 l_1 h_2 \end{bmatrix}$$

en voor de kolom $\overline{f}$:

$$\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} (m_1 h_1 + m_2 l_1) * \ddot{x}_A \\ m_2 h_2 \ddot{x}_A \end{bmatrix} \qquad \textbf{(23)}$$

Voor de toestandsbeschrijving, waarin de 2e-orde differentiaalvergelijkingen omgeschreven worden naar een stelsel 1e-orde differentiaalvergelijkingen, is het eenvoudiger om de bewegingsvergelijkingen in algemene vorm te herschrijven.

$$\dot{\overline{y}} = \overline{f}(\overline{y}, \overline{\theta}, \overline{u})$$

$\overline{y} \ni$ *toestandsgrootheden*

$$\overline{\theta} \ni k_1, k_2, b_1 \text{enb}_2 \qquad \textbf{(24)}$$

$$\overline{u} \ni \ddot{x}_A$$

$a_{22} * (21,1) - a_{12} * (21,2)$

$$\begin{bmatrix} (a_{22}a_{11} - a_{12}a_{21})\ddot{\varphi}_1 + a_{22}b_{11}\dot{\varphi}_1 + (a_{22}b_{12} - a_{12}b_{22})\dot{\varphi}_2 + \\ + (a_{22}c_{11} - a_{12}c_{21})\varphi_1 + (a_{22}c_{12} - a_{12}c_{22})\varphi_2 + a_{22}d_{11}\varphi_2(\dot{\varphi}_1 + \dot{\varphi}_2)^2 + \\ -a_{12}d_{22}\varphi_2\dot{\varphi}_1^2 = a_{22}f_1 - a_{12}f_2 \\ \\ a_{21}\ddot{\varphi}_1 + a_{22}\ddot{\varphi}_2 + b_{22}\dot{\varphi}_2 + c_{21}\varphi_1 + c_{22}\varphi_2 + d_{22}\varphi_2\dot{\varphi}_1^2 = f_2 \end{bmatrix} \qquad \textbf{(25)}$$

Een eenvoudige toestandsbeschrijving volgt nu door het invoeren van de volgende toestandsgrootheden:

$$
\begin{aligned}
y_1 &= a_{21}\varphi_1 + a_{22}\varphi_2 \\
y_2 &= \varphi_1 \\
y_3 &= \dot{\varphi}_1 \\
y_4 &= a_{21}\dot{\varphi}_1 + a_{22}\dot{\varphi}_2
\end{aligned}
\qquad \textbf{(26)}
$$

Hiermee is de toestand te beschrijven als:

$$\dot{y}_1 = y_4$$

$$\dot{y}_2 = y_3$$

$$
\begin{aligned}
\dot{y}_3 = 1/DET(A)\ [&-a_{22}b_{11} * y_3 - (b_{12} - a_{12}b_{22}/a_{22}) * (y_4 - y_3) + \\
&-(a_{22}c_{11} - a_{12}c_{21}) * y_2 - (c_{12} - a_{12}c_{22}/a_{22}) * (y_1 - a_{21}y_2) + \\
&+ a_{12}d_{22}/a_{22} * (y_1 - a_{21}y_2) * y_3^2 + d_{11} * (y_1 - a_{21}y_2) * \\
&* (y_4^2/a_{22}^2 + (1 - a_{21}/a_{22})^2 y_3^2 + 2/a_{22} * (1 - a_{21}/a_{22}) * y_3 * y_4 + \\
&+ a_{22}f_1 - a_{12}f_2]
\end{aligned}
$$

$$
\begin{aligned}
\dot{y}_4 = &-b_{22}/a_{22} * (y_4 - a_{21}y_3) - c_{21}y_2 - c_{22}/a_{22} * (y_1 - a_{21}y_2) + \\
&-d_{22}/a_{22} * (y_1 - a_{21}y_2) + f_2
\end{aligned}
$$

(27)

Dit stelsel niet lineaire 1e-orde differentiaalvergelijkingen is met behulp van de PC-Matlab routine ODE23 (of ODE45 voor nauwkeurigere berekening) op te lossen. Deze routine gebruikt de 2e- en 3e-orde (respectievelijk 4e- en 5e-orde) Runge-Kutta integratie methode.

Zie bijlage 6 voor de programma's voor het berekenen van de coëfficienten (NLSYST.M), de implementatie van de toestandsbeschrijving (NLTOEST.M) en de gebruikte PC-Matlab routines.

Voor de uiteindelijke responsiebepaling in PC-Matlab zijn de modelgegevens ingevuld in de bewegingsvergelijkingen. Na een correcte invulling volgt hieruit voor de matrices $\underline{A}$, $\underline{B}$, $\underline{C}$ en $\underline{D}$ en de kolom met belastingen $\overline{f}$:

$$\underline{A} = \begin{bmatrix} 44.2541 & 11.0387 \\ 17.5710 & 6.5323 \end{bmatrix}$$

$$\underline{B} = \begin{bmatrix} 100 & -25 \\ 0 & 25 \end{bmatrix}$$

$$\underline{C} = \begin{bmatrix} 87.9422 & -150 \\ -121.6319 & 28.3681 \end{bmatrix}$$

$$\underline{D} = \begin{bmatrix} -11.0387 & 0 \\ 0 & 11.0387 \end{bmatrix}$$

$$f_1 = 126.024 * \sin(21.71 * t)$$

$$f_2 = 37.209 * \sin(21.71 * t)$$

In de simulatie is verder gebruikt:
De begintijd t_0 = 0 s
De eindtijd t_{final} = 0.2 s
Beginvoorwaarden y_0 = [0 ; 0 ; 0 ; 0]

Voor een nauwkeurige berekening is ODE45 gebruikt, met een hoge tolerantie;
*De tolerantie tol = 1 * e -10.*

Het resultaat is identiek aan de DADS-responsie, zie bijlage 3.

3.4 Het DADS-model versus het PC-Matlab-model

Zoals al opgemerkt in paragraaf 3.3 levert de responsiebepaling via PC-Matlab geen zichtbare verschillen op met de responsie via DADS (zie bijlage 3). Beide modellen laten een positieve hoekverdraaiing zien van de benen en een in eerste instantie negatieve hoekverdraaiing van het bovenlichaam, waarbij positief tegen de draairichting van de wijzers van een klok in gericht is en de verstoring voorwaarts (positief) gericht is. Dit gedrag voldoet aan het verwachte bewegingspatroon.

De modelbeschrijving in PC-Matlab is dus te gebruiken om de modelparameters in het DADS-model te schatten. Het PC-Matlab-model is ook geschikt om het modelgedrag te onderzoeken.

Hoofdstuk 4 ***Parametervariatie in het dubbelslingermodel***

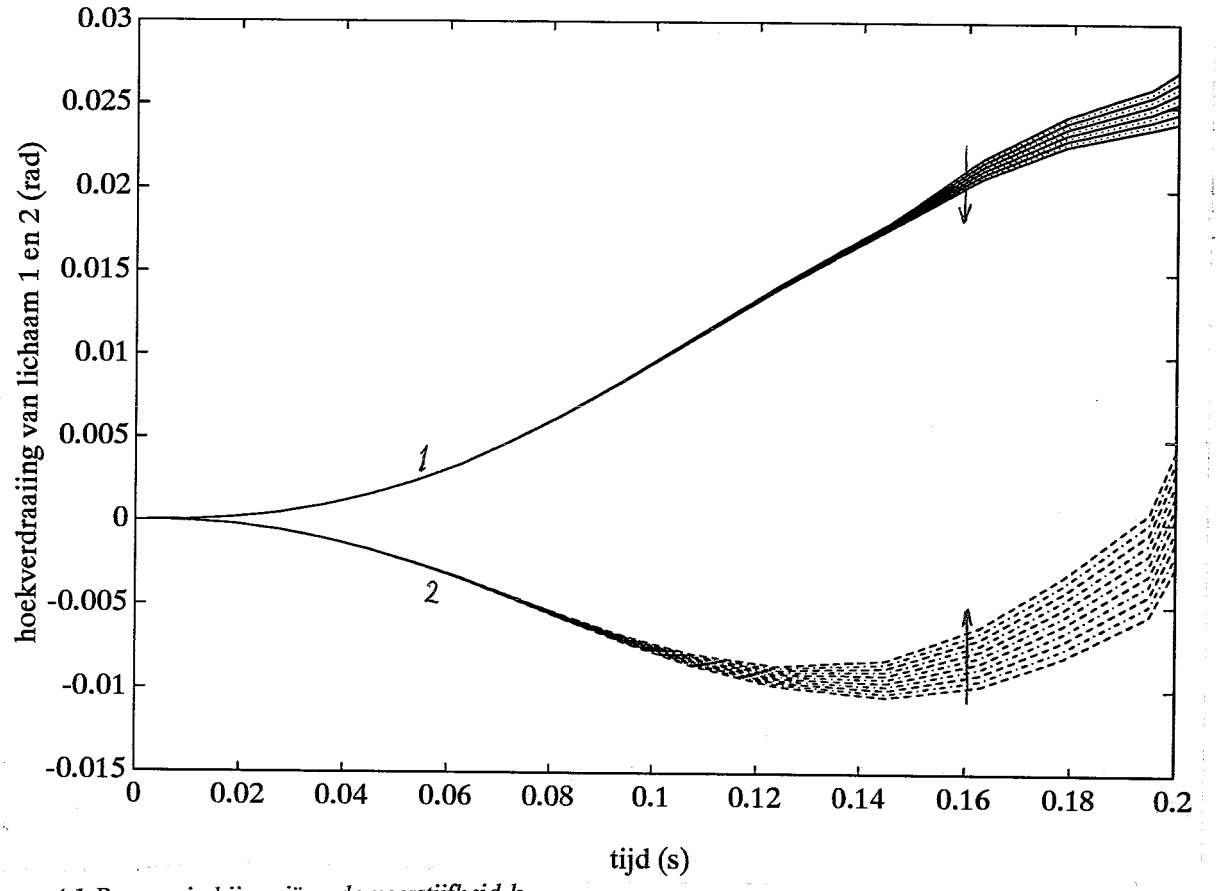
Om gevoel te krijgen voor het gedrag van het dubbelslingermodel is er een parameterstudie uitgevoerd. Door een stapsgewijze variatie van de parameters in het PC-Matlab-model (zie paragraaf 3.3) is uit de berekende responsie de invloed van de desbetreffende parameter te bepalen.
De volgende parameters zijn gevariëerd: de veerstijfheid in het enkelgewricht (k_1), de dempingsconstante in het enkelgewricht (b_1), de veerstijfheid in het heupgewricht (k_2), de dempingsconstante in het heupgewricht (b_2), de massa van het onderste slingerlichaam (m_1), de massa van het bovenste slingerlichaam (m_2), de ligging van het massamiddelpunt van het onderste slingerlichaam (h_1), de ligging van het massamiddelpunt van het bovenste slingerlichaam (h_2), het massatraagheidsmoment van het onderste slingerlichaam (J_1) en het massatraagheidsmoment van het bovenste slingerlichaam (J_2).

De resultaten van deze parametervariaties worden in de volgende paragrafen besproken. In paragraaf 4.1 de variatie van veerstijfheden, in paragraaf 4.2 de variatie van dempingsconstanten, in paragraaf 4.3 de variatie van de slingermassa's, in paragraaf 4.4 de variatie in de ligging van het massamiddelpunt, in paragraaf 4.5 de variatie van de massatraagheidsmomenten en in paragraaf 4.6 wordt het lineaire met het niet-lineaire model vergeleken.

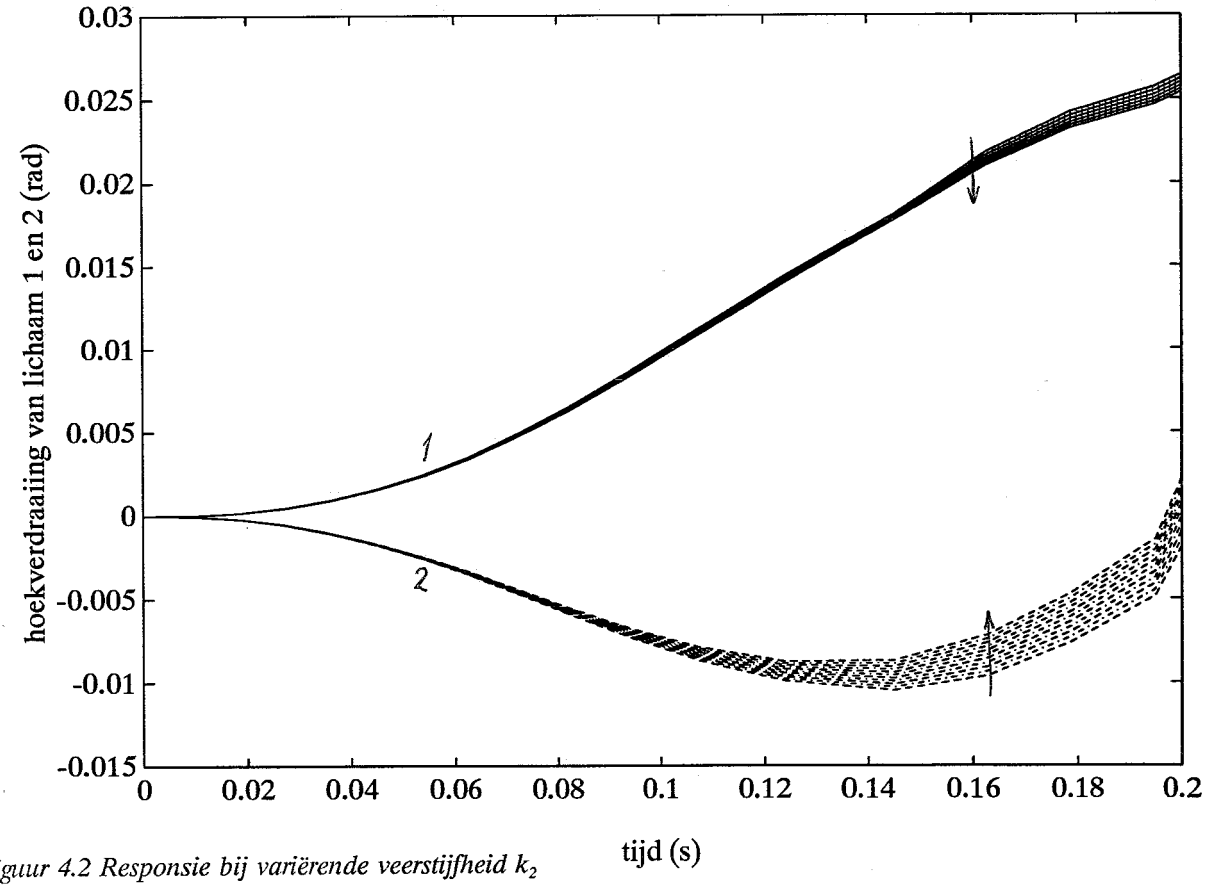
4.1 Variatie van de veerstijfheden

In het model van de dubbelslinger heeft variatie van de veerstijfheid in zowel het enkel- alsook in het heupgewricht niet veel invloed op de hoekverdraaiing van beide slingerlichamen. In deze parameterstudie is de waarde van de veerstijfheid in het enkelgewricht gevarieerd van 200 tot 1025 Nm/rad en de veerstijfheid in het heupgewricht van 50 tot 270 Nm/rad. De variatie is toegepast in stappen van 75 respectievelijk 20 Nm/rad.
De resultaten zijn te zien in de figuren 4.1 en 4.2, waarin de hoekverdraaiingen (rad) van slingerlichamen 1 en 2 zijn uitgezet tegen de tijd (s). Bij groter wordende veerstijfheid worden de hoekverdraaiingen iets kleiner. Door een kleine nauwkeurigheid (snelle rekentijd) zijn de responsies enigszins hoekig op het einde van de simulatie.

Beide veerstijfheden hebben bij de optredende kleine hoekverdraaiingen dezelfde (kleine) invloed op het modelgedrag.



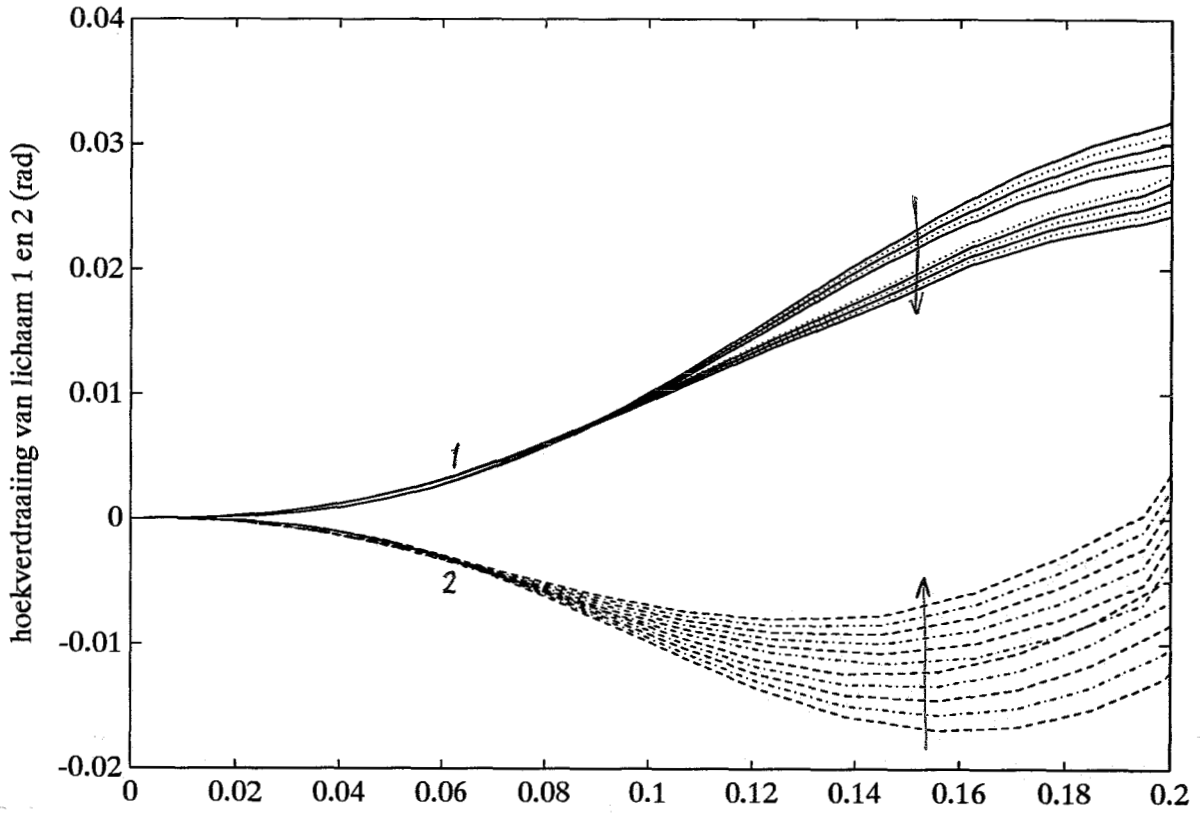
figuur 4.1 Responsie bij variërende veerstijfheid k_1



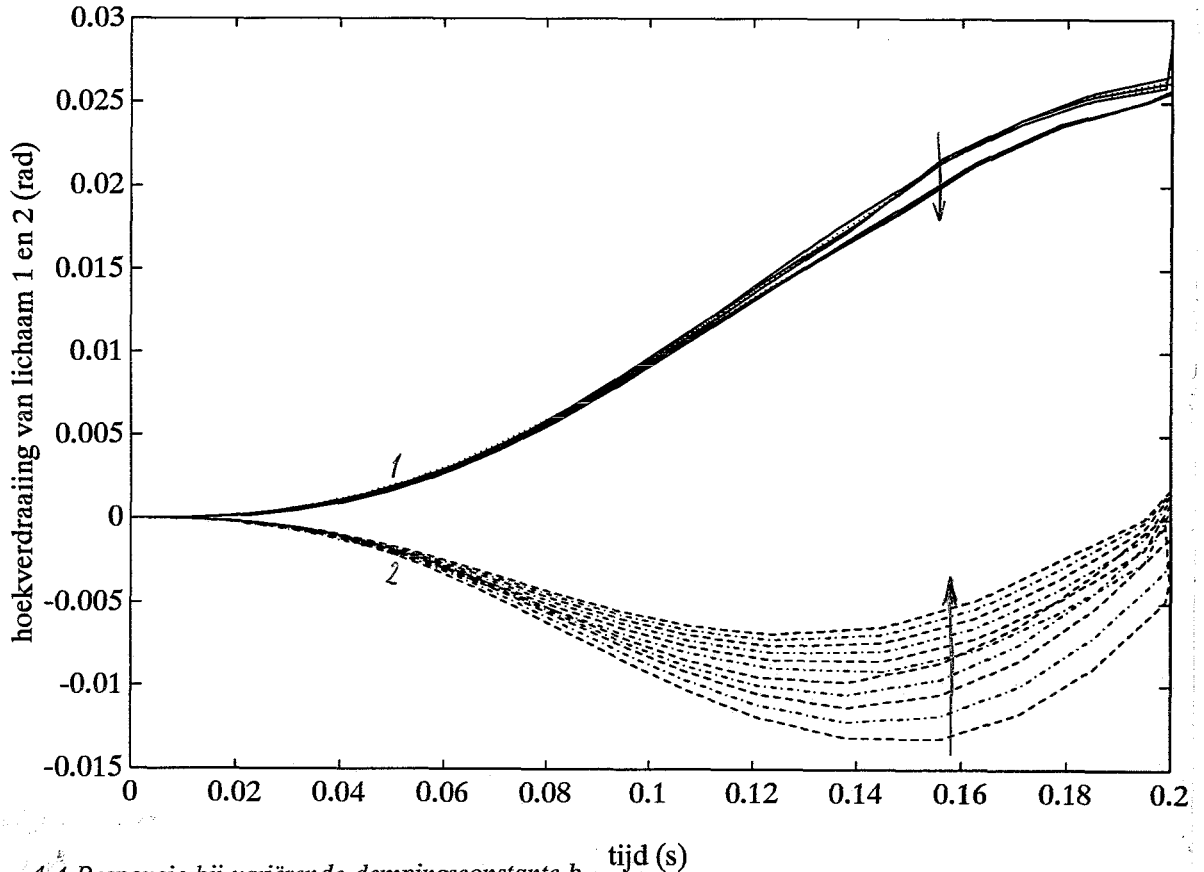
figuur 4.2 Responsie bij variërende veerstijfheid k_2

4.2 Variatie van de dempingsconstanten

De rotatiedempers die in de beide gewrichten gesitueerd zijn hebben een grotere invloed op het gedrag van het systeem dan de rotatieveren.
Zoals in de figuren 4.3 en 4.4 te zien is, wordt door vergroting van de demping de hoekverdraaiing van met name slingerlichaam 2 gedempt. De variatie in dempingsconstanten die is toegepast is voor de demper in het enkelgewricht 25 tot 135 Nm/rad/s en voor de demper in het heupgewricht 10 tot 48,5 Nm/rad/s, in stappen van 10 respectievelijk 3,5 Nm/rad/s.
Ook in deze figuren zijn de responsies enigszins hoekig vanwege de kleine nauwkeurigheid.



figuur 4.3 Responsie bij variërende dempingsconstante b_1

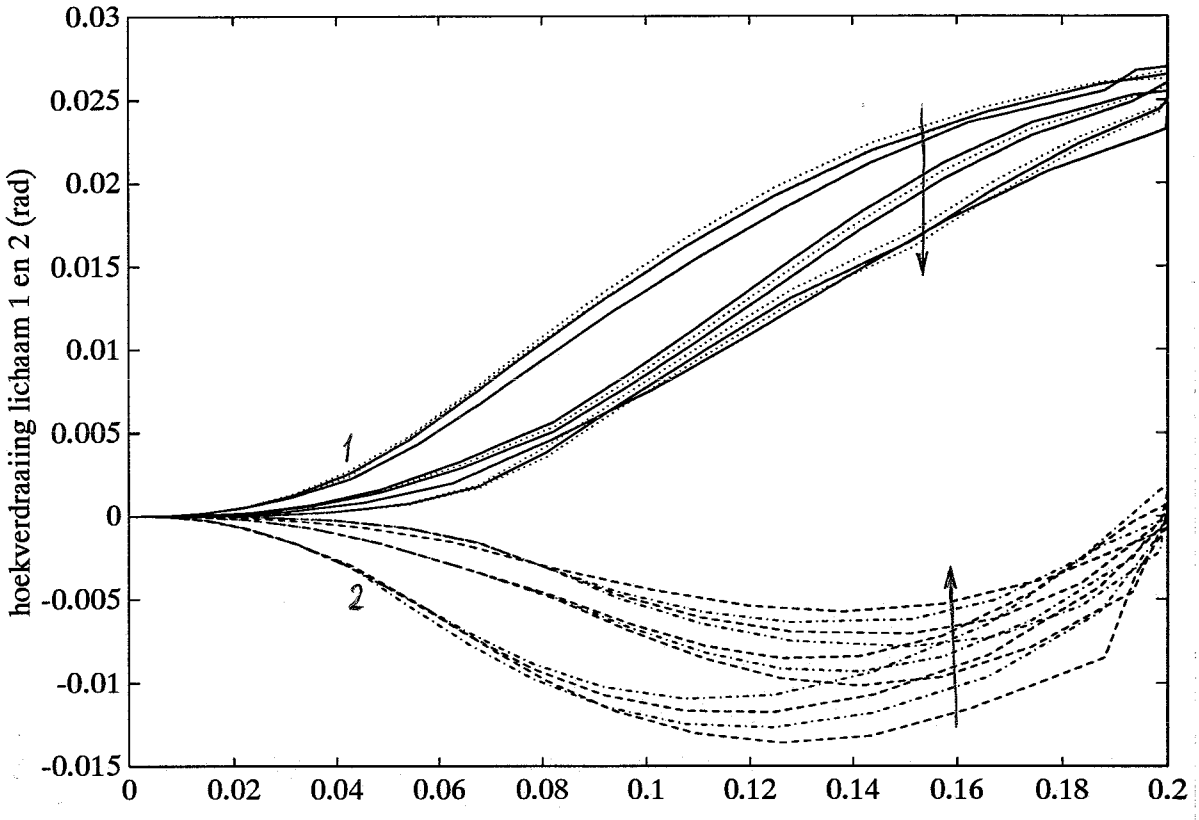


figuur 4.4 Responsie bij variërende dempingsconstante b_2

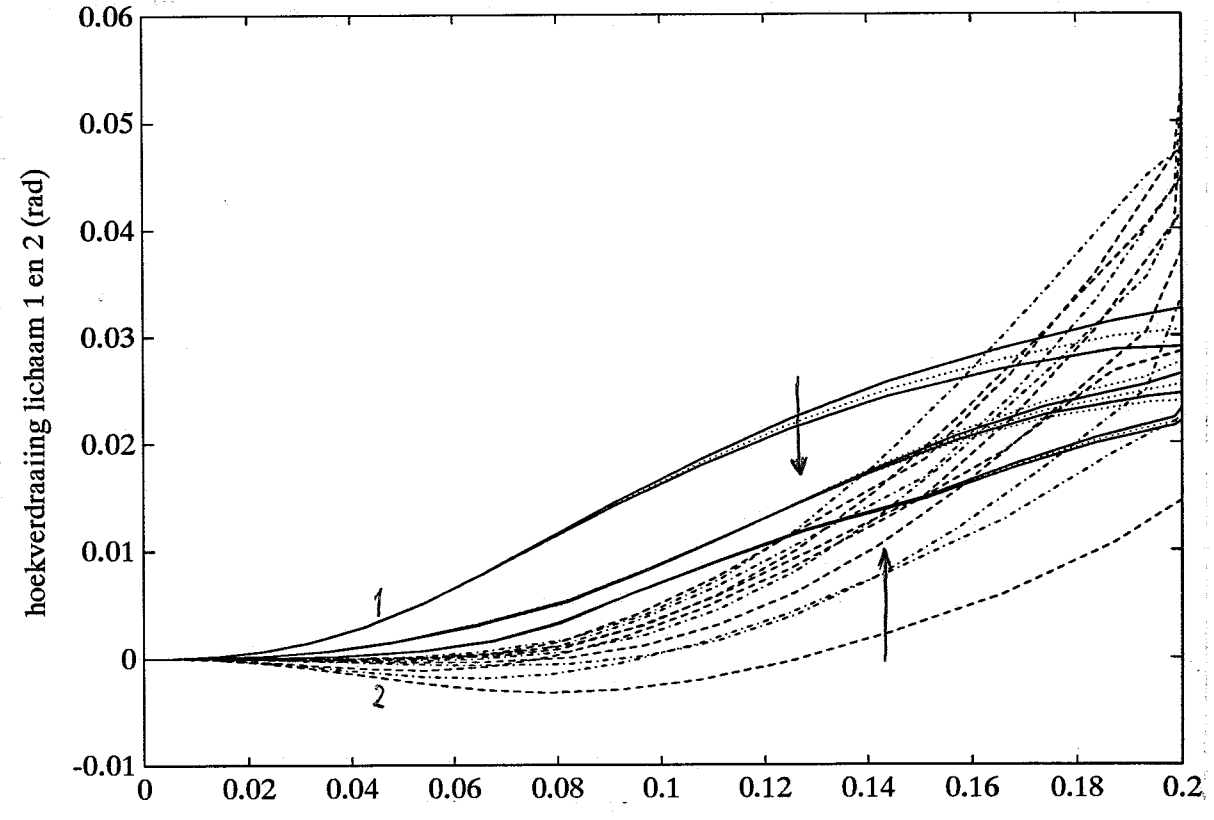
4.3 Variatie van de slingermassa's

Het gewicht van een proefpersoon is een parameter die van invloed kan zijn op het modelgedrag. Voor het opstellen van een goed simulatiemodel is het van belang die invloed te kennen.
In dit model hebben beide slingerlichamen een homogeen verdeelde massa, die van invloed is op het massatraagheidsmoment ten opzichte van het massamiddelpunt van beide lichamen. Deze invloed is terug te vinden in een vergroting van het massatraagheidsmoment bij groter wordende massa. De slingerlichamen worden benaderd met homogene balken en voor het massatraagheidsmoment ten opzichte van het massamiddelpunt geldt dus $J = m * l^2 / 12$.
De slingermassa's zijn gevarieerd van 20 tot 47,5 kg in stappen van 2,5 kg. De invloed van deze massaveranderingen is te zien in de figuren 4.5 en 4.6.
Het blijkt dat bij groter wordende massa de curves van de hoekverdraaiingen een platter verloop krijgen . Uit de beide figuren blijkt tevens dat vergroting van de massa van het bovenste lichaam een veel sterkere reactie teweeg brengt dan vergroting van de massa van het onderste lichaam. Het lichaam wordt dubbelgevouwen, zoals ook te verwachte is bij een 'topzwaar' persoon.

Het hoekige verloop met de piek op het einde van de simulatie is te wijten aan de lage nauwkeurigheid, waarmee deze variatie doorgerekend is. De sprong in het de responsies bij groter wordende massa's is vreemd.



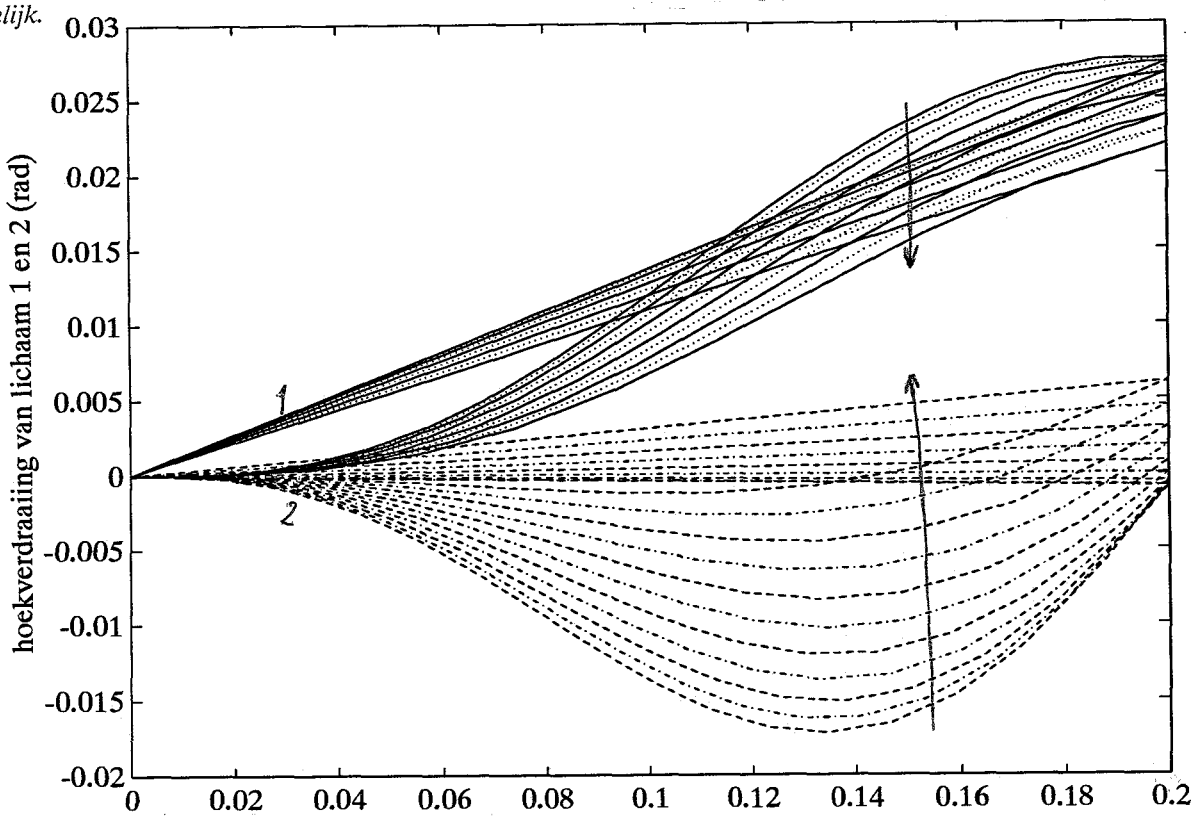
figuur 4.5 Responsie bij variërende slingermassa m_1



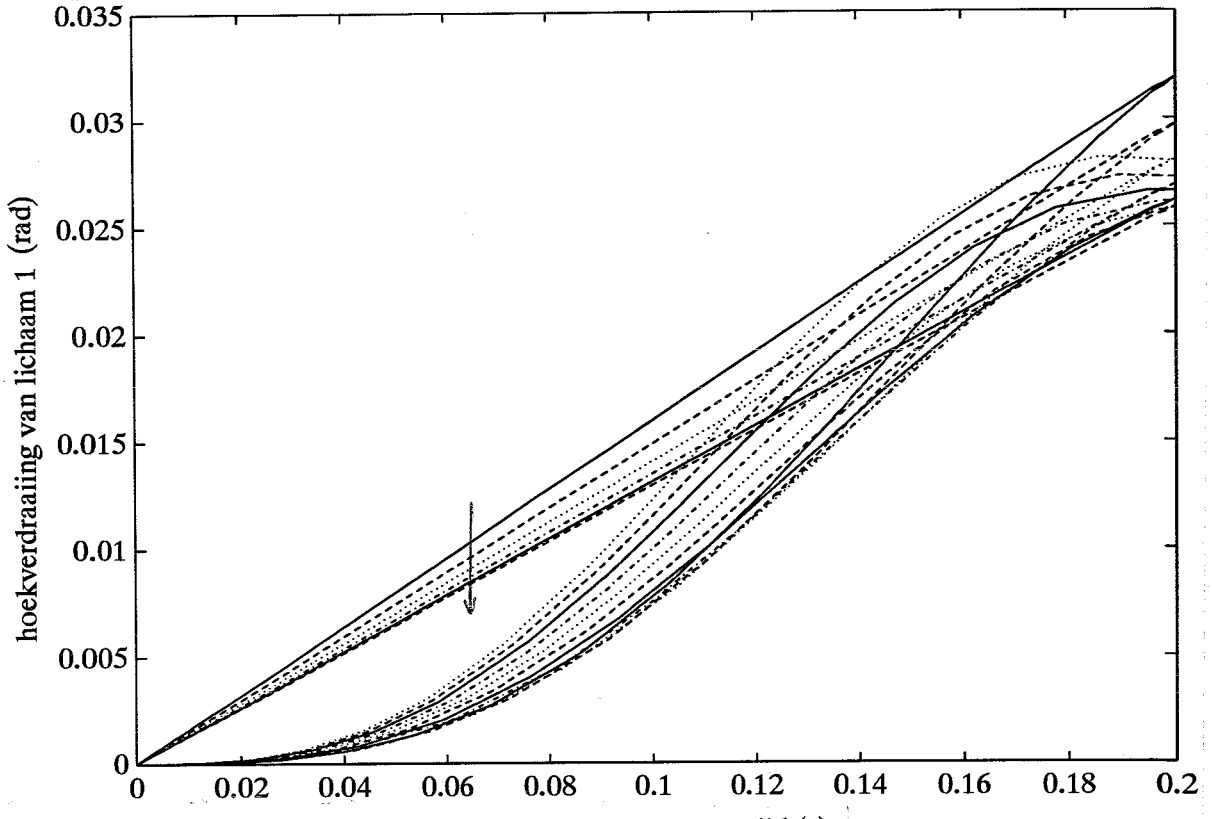
figuur 4.6 Resopnsie bij variërende slingermassa m_2

4.4 Variatie van de ligging van het massamiddelpunt

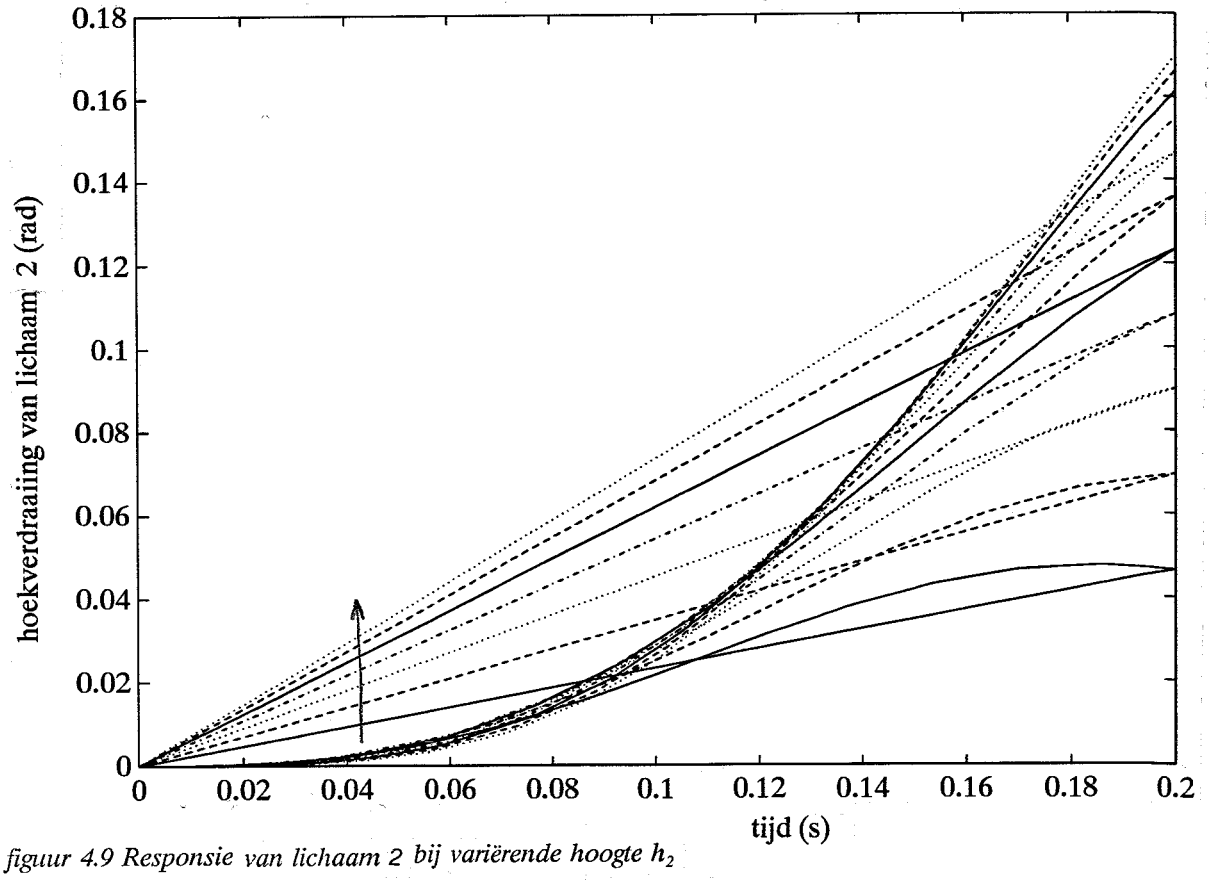
In dit model is de ligging van het massamiddelpunt vastgelegd door de hoogte boven het gewricht (draaiingsas) in vertikale positie op te geven. Vanwege de uniform veronderstelde massaverdeling en de modellering als starre staaf, is de hoogteparameter h de helft van de slingerlengte. De veronderstelling dat de massa uniform over het lichaam is verdeeld is niet in overeenkomst met de werkelijkheid. De aanname dat de lichaamsdelen te zien zijn als starre staven is eveneens een vereenvoudiging van de werkelijkheid. Het massamiddelpunt van elk lichaamsdeel bevindt zich dus waarschijnlijk niet op de halverwege dat lichaamsdeel. Door variatie in de ligging van het massamiddelpunt wordt er inzicht verkregen over de invloed van deze vereenvoudigingen op het modelgedrag.
Voor lichaam 1 is de hoogte in stappen van 0,055 m gevarieerd van 0,15 tot 0,755 m. De resultaten van deze variatie zijn te zien in figuur 4.7. Voor lichaam 2 is de hoogte in stappen van 0,045 m gevarieerd van 0,125 tot 0,62 m. De resultaten van deze variatie zijn te zien in de figuren 4.8 en 4.9. Voor de duidelijkheid zijn de hoekverdraaiingen van slingerlichaam 1 en 2 bij variatie van h_2 apart weergegeven.
Ook nu worden de curves van de hoekverdraaiingen platter van vorm. Voor het bovenste slingerlichaam wordt de hoekverdraaiing positief bij verleggen van het massamiddelpunt naar boven, zodat hetzelfde gedrag optreedt als bij een 'topzwaar' persoon (zie paragraaf 4.3).
Het massatraagheidsmoment wordt bij deze variatie constant gehouden. De slingerlengte blijft eveneens gelijk.



figuur 4.7 Responsie bij variërende hoogte h_1



figuur 4.8 Responsie van lichaam 1 bij variërende hoogte h_2

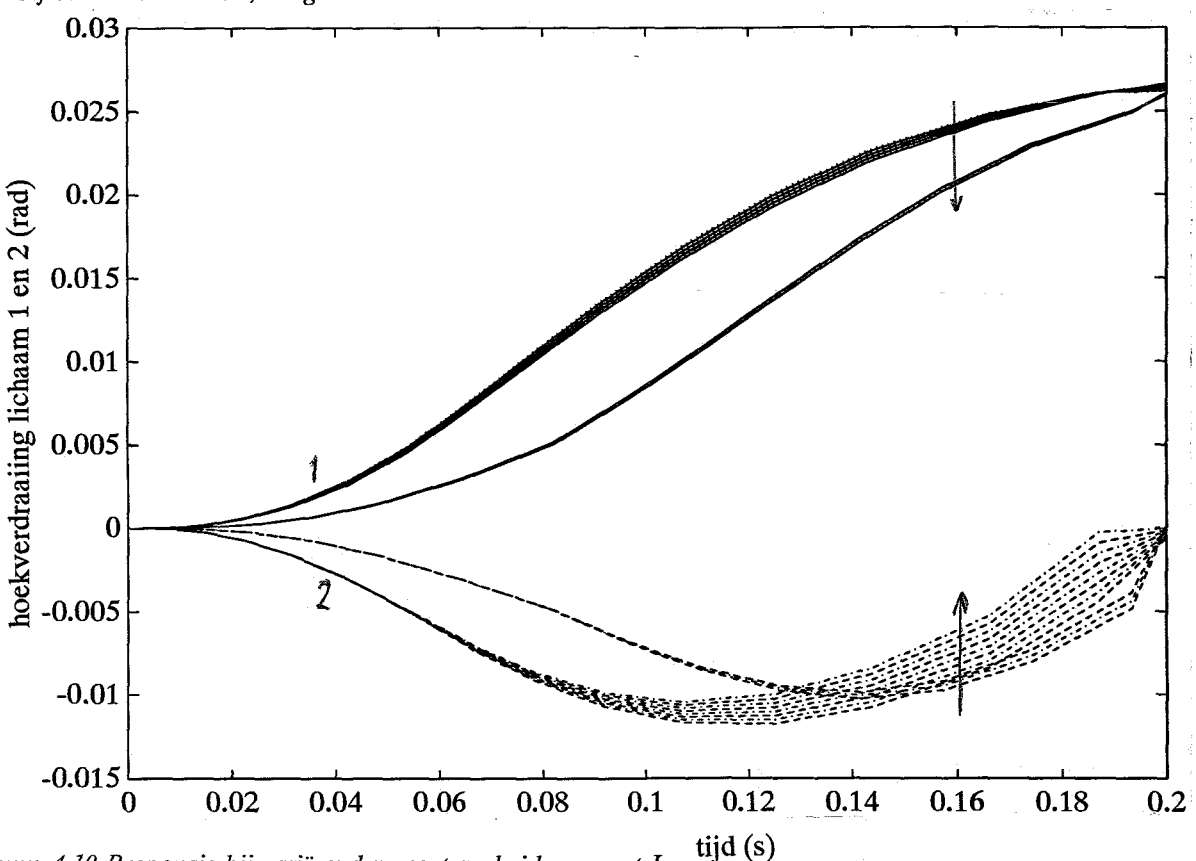


figuur 4.9 Responsie van lichaam 2 bij variërende hoogte h_2

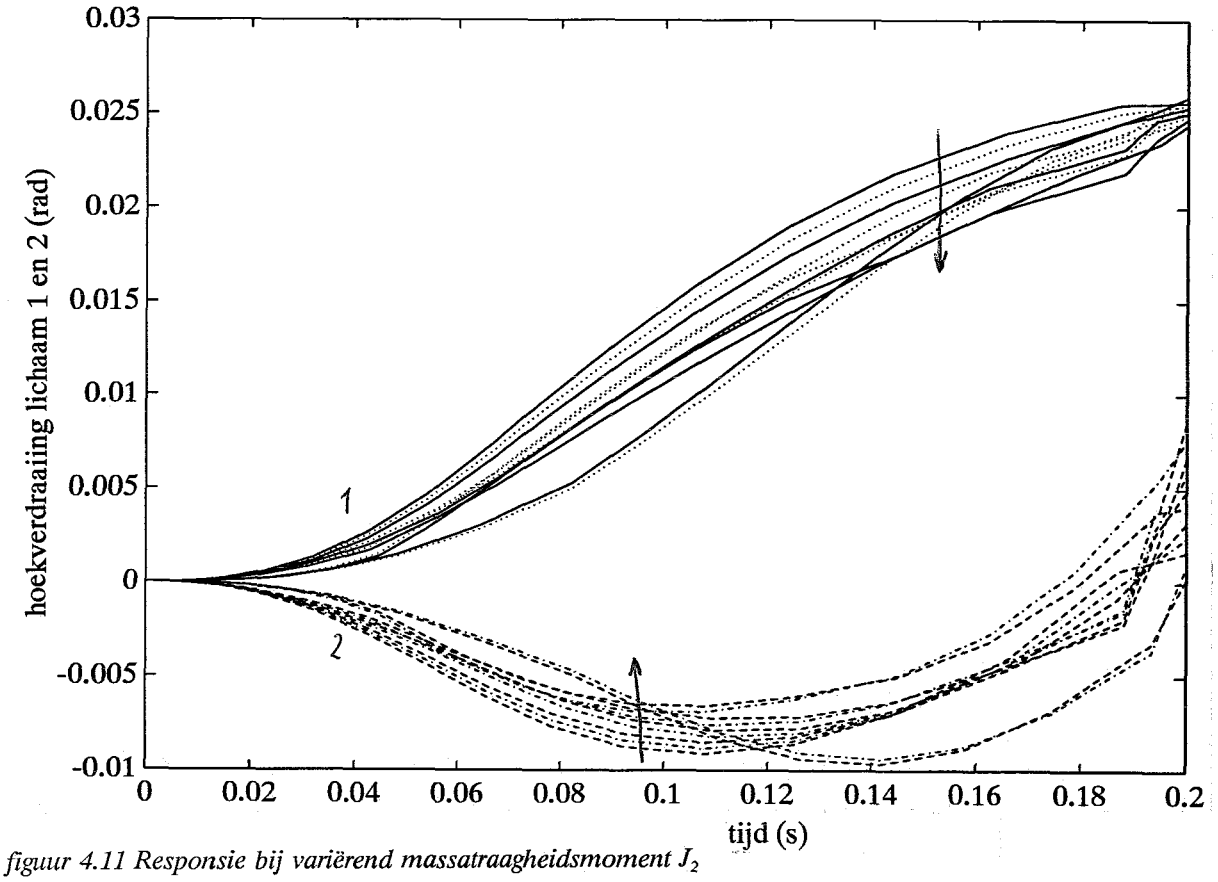
4.5 Variatie van het massatraagheidsmoment

De massatraagheidsmomenten ten opzichte van de massamiddelpunten van de beide slingerlichamen in dit model zijn constant. De grootte van die massatraagheidsmomenten is uitgerekend met de aanname dat de massa homogeen verdeeld is en beide slingerlichaamen met een starre staaf benaderd kunnen worden. Deze beide aannames zijn een sterke vereenvoudiging van de werkelijkheid. De grootte van het massatraagheidsmoment van elk lichaamsdeel in dit model is niet bekend. Om de invloed van de grootte van het massatraagheidsmoment ten opzichte van het massamiddelpunt te kennen (en dus het belang van een correcte bepaling ervan) wordt deze parameter in het dubbelslingermodel gevarieerd.
In deze parameterstudie verandert alleen het massatraagheidsmoment ten opzichte van het massamiddelpunt van elk slingerlichaam in het model, dus de ligging van het massamiddelpunt blijft hetzelfde (halverwege het slingerlichaam).
De grootte van het massatraagheidsmoment van lichaam 1 is gevarieerd van 0,4 tot 2,05 kgm^2 in stappen van 0,15 kgm^2. De grootte van het massatraagheidsmoment van lichaam 2 is gevarieerd van 0,35 tot 1,78 kgm^2 in stappen van 0,13 kgm^2.
De resultaten zijn te zien in de figuren 4.10 en 4.11.

Vergroting van de massatraagheidsmomenten ten opzichte van de massamiddelpunten heeft niet veel invloed op de respnsie van het model. Bij groter wordend massatraagheidsmoment worden de hoekverdraaiingen bij beide slingerlichamen iets kleiner. Vreemd is dat er in de verandering van de responsies bij variatie van J_1 een sprong voorkomt bij de waarde 1,75 kgm^2. Bij variatie van J_2 treedt zo'n zelfde sprong in de responsies op bij een waarde van 1,49 kgm^2.



figuur 4.10 Responsie bij variërend massatraagheidsmoment J_1

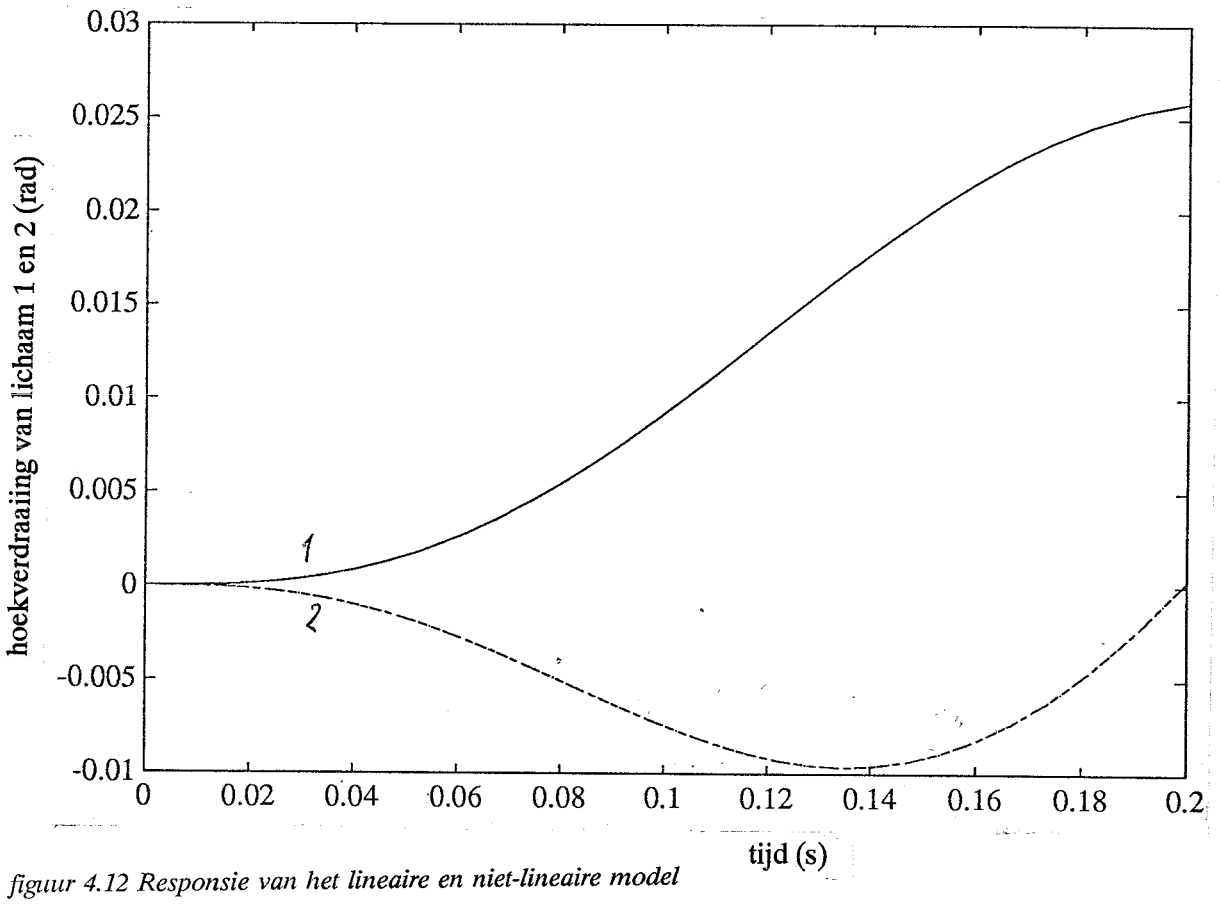


figuur 4.11 Responsie bij variërend massatraagheidsmoment J_2

4.6 Het lineaire versus het niet-lineaire model

In de parameterstudie is gebruik gemaakt van het niet-lineaire model van de dubbelslinger, beschreven in hoofdstuk 3. Met "niet-lineair" wordt hier bedoeld het model met de hogere orde termen in de variabelen, maar wel gelineariseerd rond de evenwichtsstand, zie formule 19.
Het verschil in rekentijd tussen het lineaire en het niet lineaire model was niet zo groot dat die rekentijd een beperkende factor was. In een fitprocedure is de gewonnen rekentijd door het lineaire model te gebruiken wel van nut, door de herhaling van het doorrekenen van het model. Het blijkt voor de hoekverdraaiingen die optreden nauwelijks verschil te maken of er met het lineaire of het niet-lineaire model gerekend wordt (zie figuur 4.12). Pas bij grotere hoekverdraaiingen, wanneer ook de linearisatie rond de evenwichtsstand te grote afwijkingen oplevert is gebruik van het lineaire model discutabel.

Zie bijlage 7 voor de programma's waarmee deze parametervariatie uitgevoerd zijn.



figuur 4.12 Responsie van het lineaire en niet-lineaire model

Hoofdstuk 5 **Parameterschatting**

De parameterschatting is uitgevoerd met behulp van PC-Matlab, waarin standaard fitprocedures beschikbaar zijn. De kleinste kwadraten methode wordt automatisch gebruikt bij uitrekenen van een overbepaald stelsel (matrix-) vergelijkingen. Andere fitprocedures zijn aanwezig in de toolboxes waar de faculteit werktuigbouwkunde over beschikt.
In de volgende paragrafen zullen de fitprocedures die gebruikt zijn beschreven worden. In paragraaf 5.1 de procedure voor het enkelslingermodel, in paragraaf 5.2 de procedures voor het dubbelslingermodel en in paragraaf 5.3 de procedure en resultaten van het fitten van de meetwaarden.

5.1 Parameterschatting bij het enkelvoudige slinger-model

Het enkelvoudige slinger-model is beschreven in hoofdstuk 2. De in paragraaf 2.1 afgeleide lineaire bewegingsvergelijking vormt het wiskundige model waarmee gefit is. Deze (matrix-) vergelijking heb is geïmplementeerd in PC-Matlab.
De hoekverdraaiing, de hoeksnelheid en de hoekversnelling zijn de variabelen in deze bewegingsvergelijking. De te fitten parameters maken deel uit van de coëfficienten voor de variabelen in de bewegingsvergelijking. Door het invullen van de berekende waarden voor de hoekverdraaiing (uit de gemeten horizontale en vertikale verplaatsingen) en de daarmee berekende hoeksnelheid en hoekversnelling per tijdstip ontstaat een overbepaald stelsel vergelijkingen. Dit overbepaalde stelsel vergelijkingen wordt in PC-Matlab automatisch met een kleinste kwadraten methode opgelost. Als oplossing geeft het programma dan de geschatte waarden voor de veerstijfheid en dempingsconstante. Hiermee wordt de fout tussen de berekende en gemeten waarde geminimaliseerd.
Het programma waarin deze procedure beschreven is, is bijgevoegd in bijlage 8.

Deze fitprocedure levert een correcte schatting op, zoals uitgetest is op met DADS gegenereerde meetgegevens.

5.2 Parameterschatting bij het dubbelslingermodel

Het dubbelslingermodel is beschreven in hoofdstuk 3. Het wiskundige model bestaat uit 2 gekopppelde 2^e orde differentiaalvergelijkingen, die in paragraaf 3.1 afgeleid zijn. Deze vergelijkingen zijn in matrix-vorm geïmplementeerd in PC-Matlab.
Behalve de kleinste kwadraten methode zijn er in de optimization-toolbox van PC-Matlab enkele functies beschikbaar om willekeurige niet-lineaire functies te fitten op meetgegevens. De drie functies die voor dit probleem geschikt zijn, zijn FMINS, FMINU en CONSTR. FMINS gebruikt de Nelder-Mead simplex methode, FMINU gebruikt de quasi-Newton methode en CONSTR is een functie waarbij beginvoorwaarden opgegeven moeten worden en maakt gebruik van Sequential Quadratic Programming. Deze vier methoden worden in de volgende subparagrafen besproken.

5.2.1 De kleinste kwadraten methode

Ook nu zijn de hoekverdraaiing, hoeksnelheid en hoekversnelling de variabelen en zijn de veerstijfheden en dempingsconstanten parameters in de coëfficienten van de bewegingsvergelijkingen. Door invullen van de gemeten en daaruit berekende waarden wordt dit overbepaalde stelsel met een kleinste kwadraten methode opgelost. Zie bijlage 9 voor de programmabeschrijving.

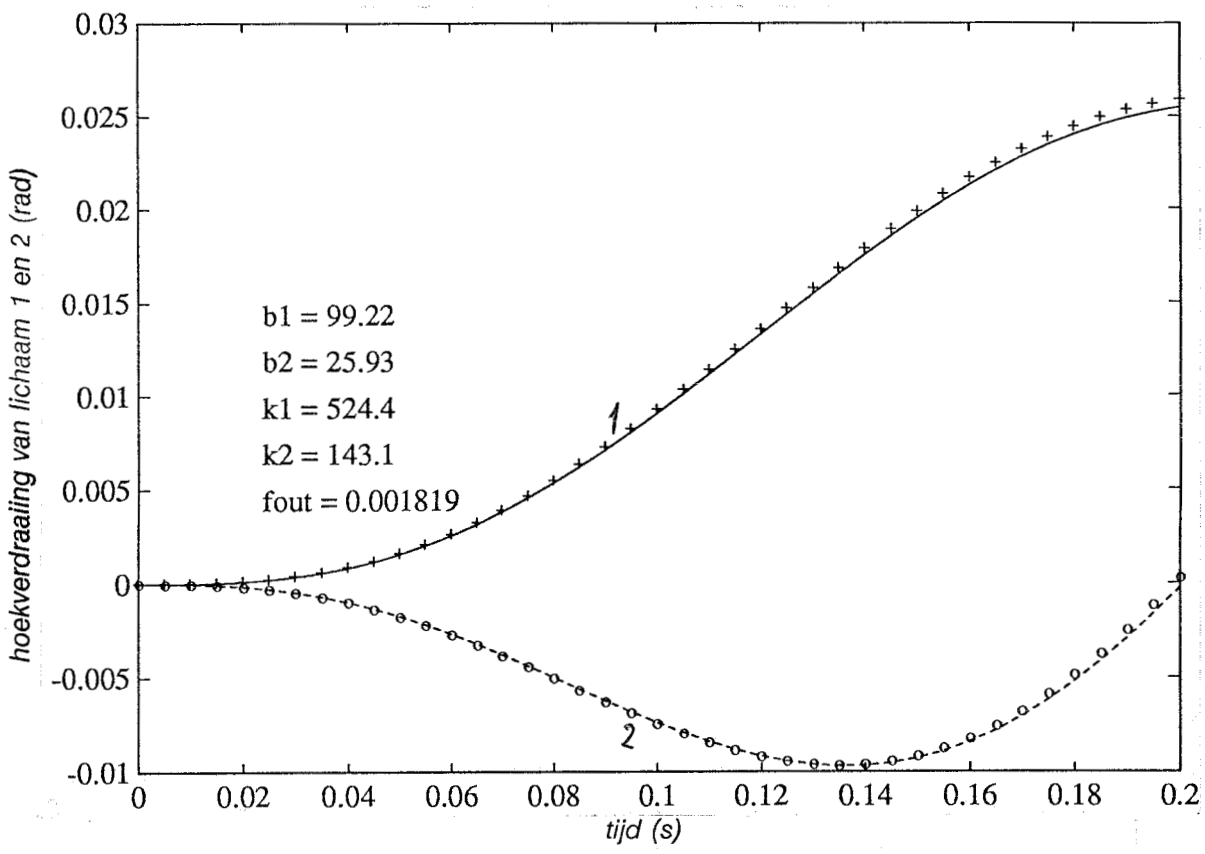
Op deze manier is de koppeling van beide differentiaalvergelijkingen niet expliciet voorgeschreven, zodat de uitkomst onzinnig kan zijn. Om dit te voorkomen is er een procedure gebruikt, waarbij de twee bewegingsvergelijkingen afzonderlijk berekend worden. Door eerst de bewegingsvergelijking van het bovenste slingerlichaam door te rekenen zijn de waarden van b_2 en k_2 bekend en kunnen die waarden in de bewegingsvergelijking van het onderste slingerlichaam ingevuld worden, waarna de overige parameters geschat worden. Zie bijlage 10 voor de programmabeschrijving.
In deze procedure is het nodig om per parameter in te kunnen geven of de desbetreffende parameter bekend is en hoe groot hij dan is. Alle bekende delen uit de bewegingsvergelijkingen worden dan naar het rechterlid van die vergelijkingen gebracht, zodat de te schatten parameters overblijven in het linkerlid van die vergelijkingen. Dit overbepaalde stelsel vergelijkingen wordt dan volgens een kleinste kwadraten methode opgelost.

De beide hiervoor beschreven oplosprocedures blijken verkeerde schattingen op te leveren, indien gebruik wordt gemaakt van m.b.v. DADS gegenereerde meetgegevens, waarvan de bijbehorende parameters bekend zijn. Negatieve veerstijfheden en dempingsconstanten blijken ook tot de oplossingsmogelijkheden te behoren, hetgeen wel wiskundig verantwoord, maar fysisch gezien onzin is. Deze procedures zijn dus niet geschikt om de meetgegevens mee te fitten.

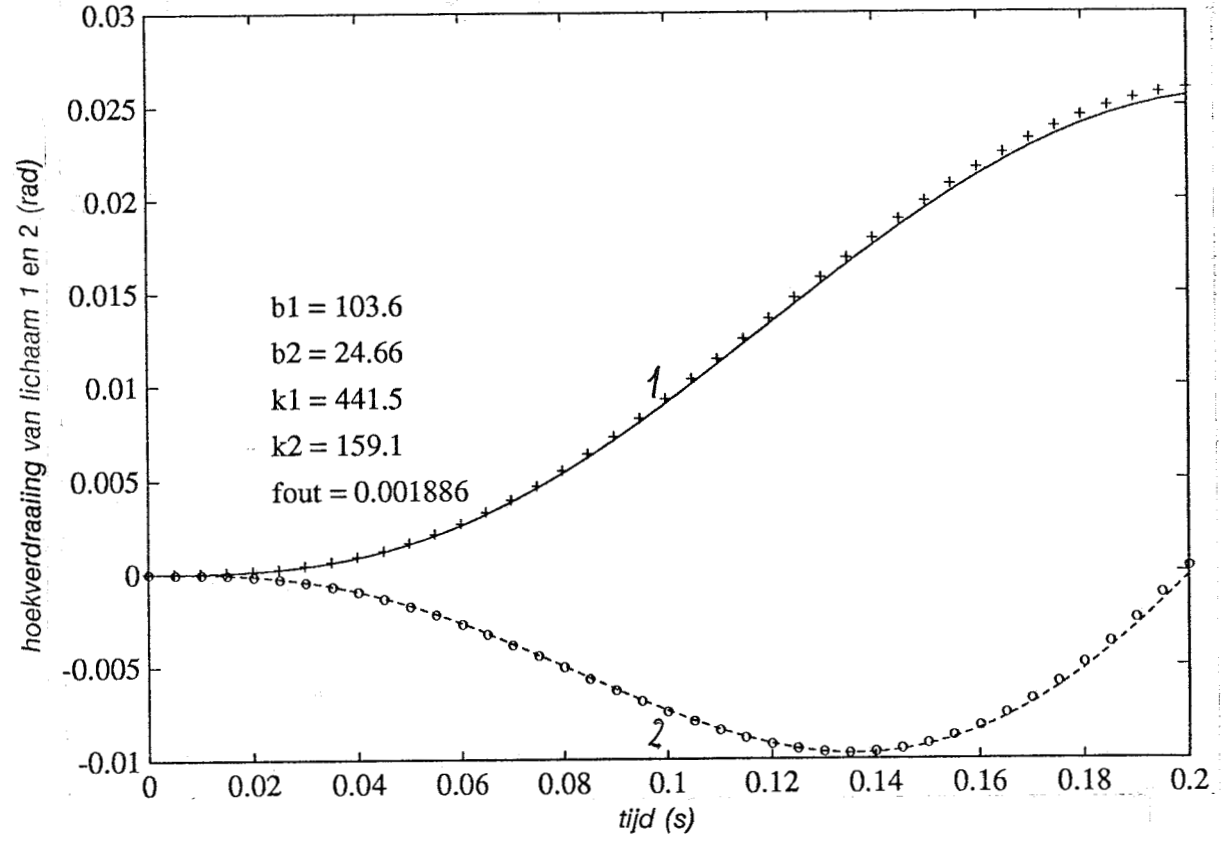
5.2.2 De Nelder-Mead simplex methode

De procedure FMINS uit de optimization toolbox van PC-Matlab gebruikt de Nelder-Mead simplex methode om een scalaire niet-lineaire functie te minimaliseren. Om deze procedure toe te passen op het fitprobleem wordt er een functie geïntroduceerd die de fout tussen gemeten en berekende waarden berekend, afhankelijk van de te fitten parameters. De Nelder-Mead simplex methode is beschreven in bijlage 11 en de gebruikte procedure is beschreven in bijlage 12.
Als fout wordt de norm van de gemeten minus de berekende waarde genomen.
Deze procedure levert niet het globale minimum van de te minimaliseren functie. Zoals te zien is in de figuren 5.1, 5.2 en 5.3 geeft deze procedure niet gelijke oplossingen bij verschillende beginschattingen en geeft een correcte beginschatting niet snel de goede oplossing.

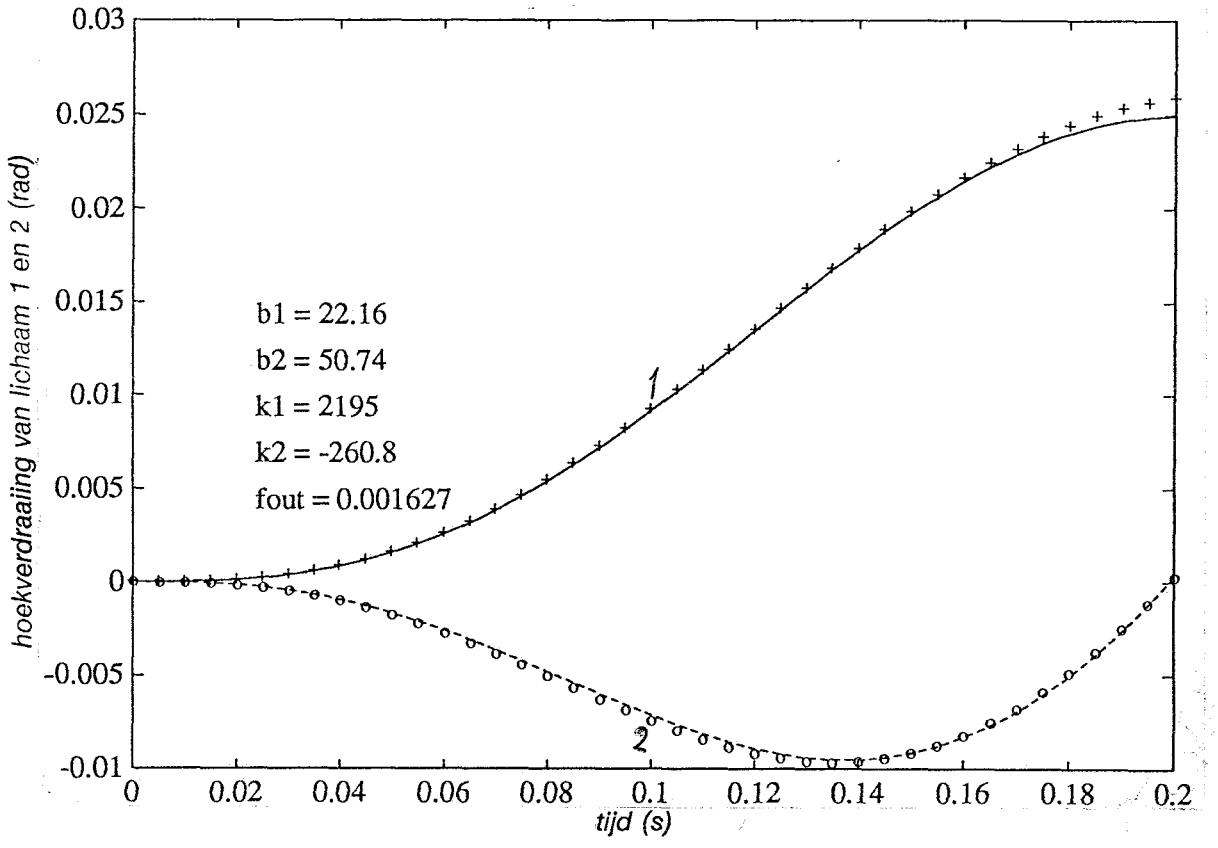
De m.b.v. DADS gegenereerde "meetgegevens" zijn berekend met de volgende waarden voor respectievelijk b_1, b_2, k_1 en k_2: 100 Nms/rad, 25 Nms/rad, 500 Nm/rad en 150 Nm/rad. De tolerantie ligt in de ordegrootte van 1.e -8.



figuur 5.1 Fitresultaat van FMINS bij de correcte beginwaarden [100, 25, 500, 50]



figuur 5.2 Fitresultaat van FMINS bij beginwaarden [90, 30, 550, 165]



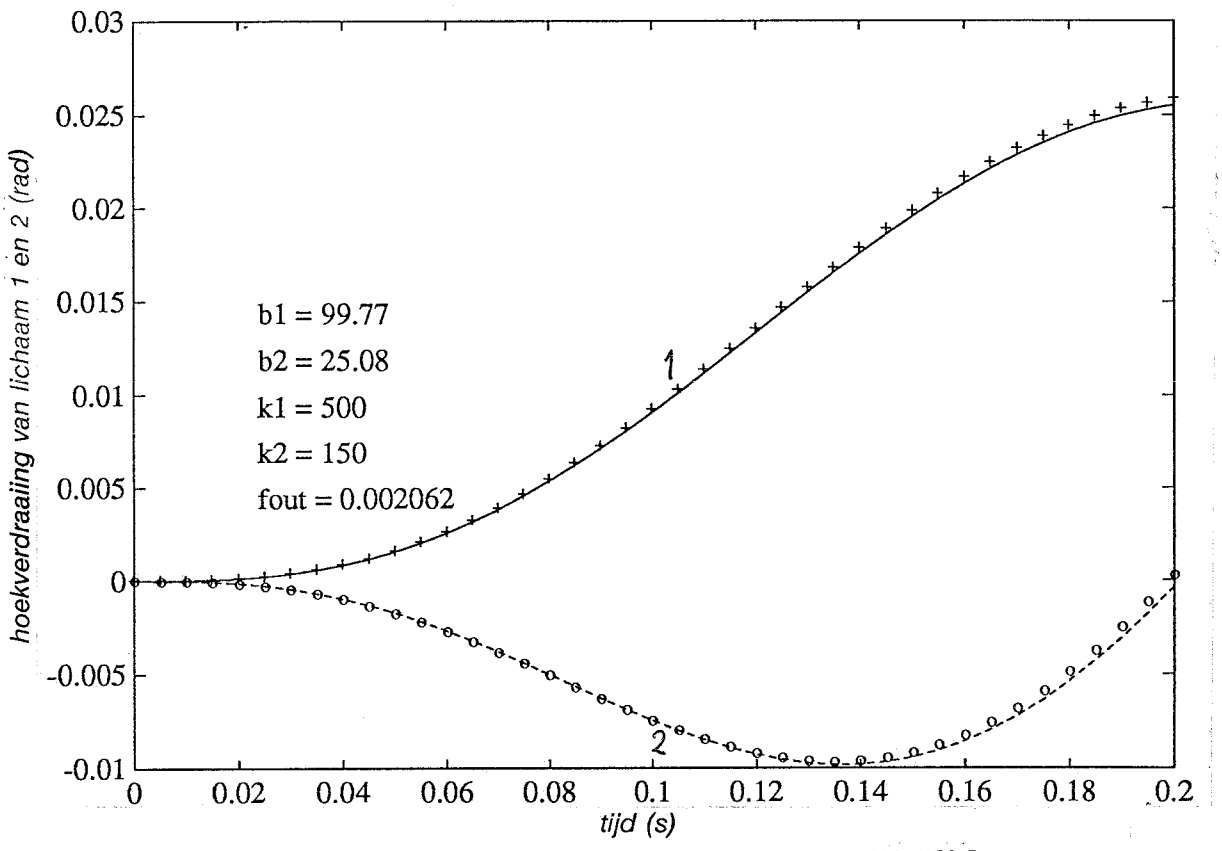
figuur 5.3 Fitresultaat van FMINS bij beginwaarden [15, 75, 349, 199]

5.2.3 De quasi-Newton methode

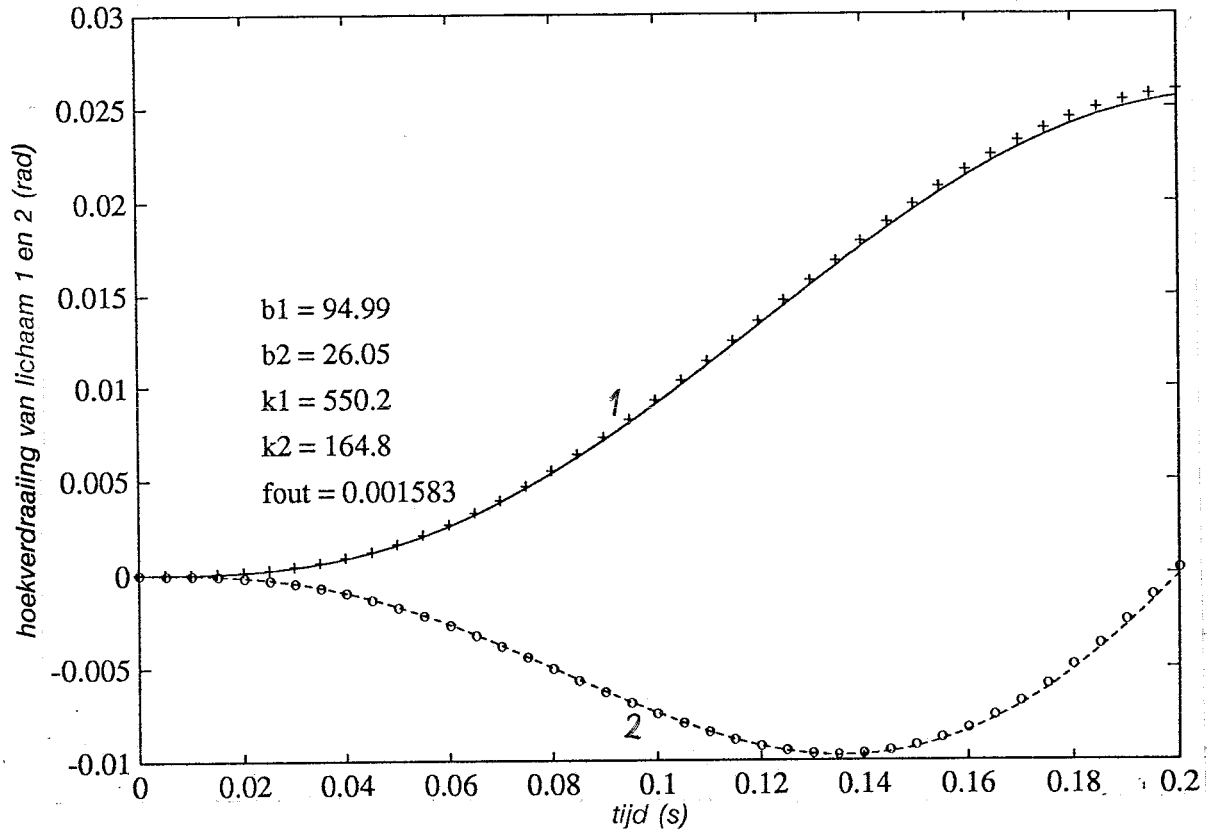
De procedure FMINU uit de optimization toolbox van PC-Matlab gebruikt de quasi-Newton methode om een scalaire niet-lineaire functie te minimaliseren. Voor deze procedure is dezelfde foutfunctie gebruikt als ook bij FMINS toegepast is. De procedure FMINU is nader beschreven in bijlage 13.
Ook deze procedure is uitgetest met de door DADS gegenereerde meetgegevens (zie paragraaf 5.2.2). Afhankelijk van de beginschatting is FMINU bruikbaar of niet. De optimalisatie wordt onderbroken doordat er singulariteiten optreden in de berekening, waarna er minder berekende tijdstippen zijn dan gemeten waarden. Hierdoor ontstaan er problemen bij het berekenen van de fout en dus wordt de optimalisatie onderbroken. Dit probleem zou te ondervangen zijn door aanpassing van de foutfunctie, hoewel de kans bestaat dat er weinig bruikbare punten overblijven. Aangezien er nog andere methoden bestaan om te fitten, is hier van afgezien.

5.2.4 De Sequential Quadratic Programming methode

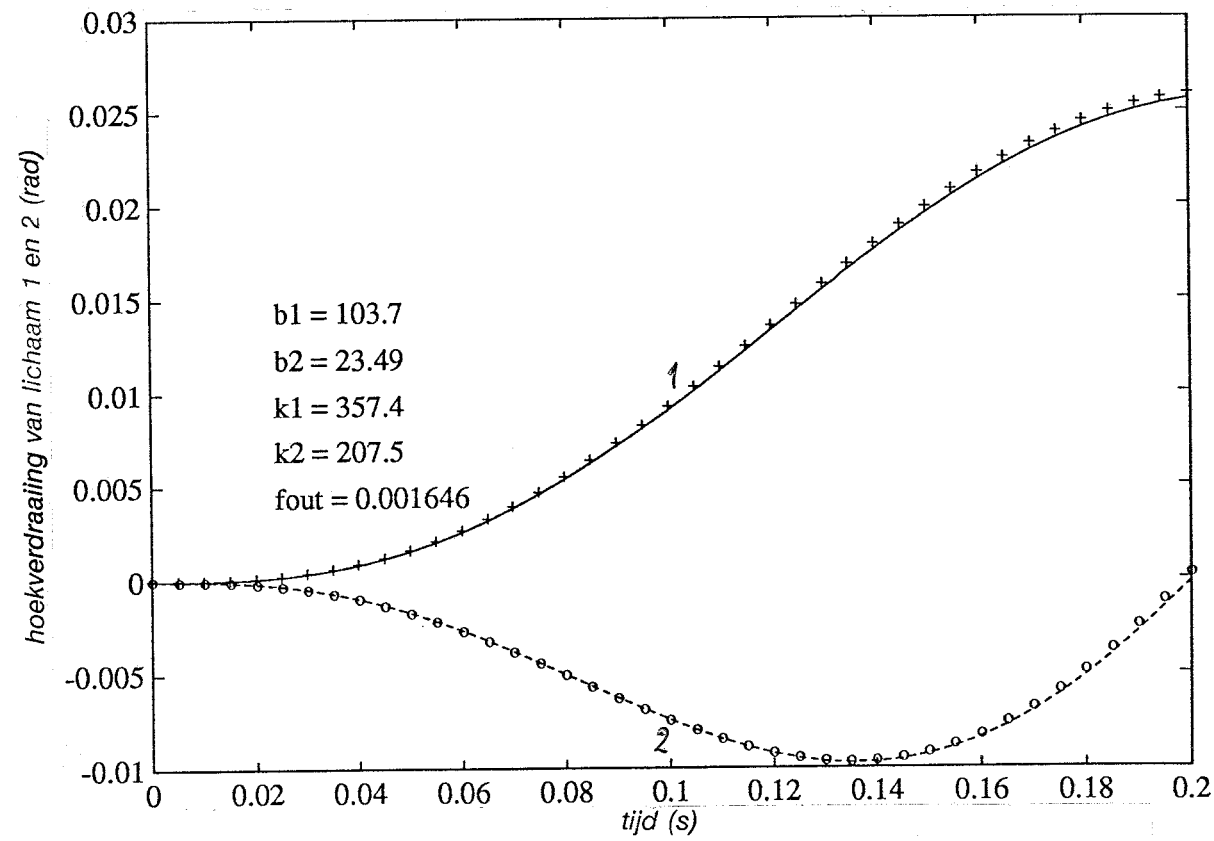
De procedure CONSTR uit de optimization toolbox van PC-Matlab gebruikt Sequential Quadratic Programming om een multivariabele niet-lineaire functie met randvoorwaarden te minimaliseren. Wederom is dezelfde foutfunctie gebruikt als bij FMINS en FMINU. De voorgeschreven randvoorwaarden zijn dat alle parameters groter dan of gelijk aan 0 moeten zijn.
De gebruikte procedures staan beschreven in bijlage 14.
Uit de resultaten van de parameterschatting op de door DADS gegenereerde meetgegevens (zie paragraaf 5.2.2) blijkt deze procedure niet het globale minimum te schatten. De resultaten van deze tests zijn weergegeven in de figuren 5.4, 5.5 en 5.6.



figuur 5.4 Fitresultaat van CONSTR bij de correcte beginwaarden [100, 25, 500, 150]



figuur 5.5 Fitresultaat van CONSTR bij beginwaarden [90, 30, 550, 165]

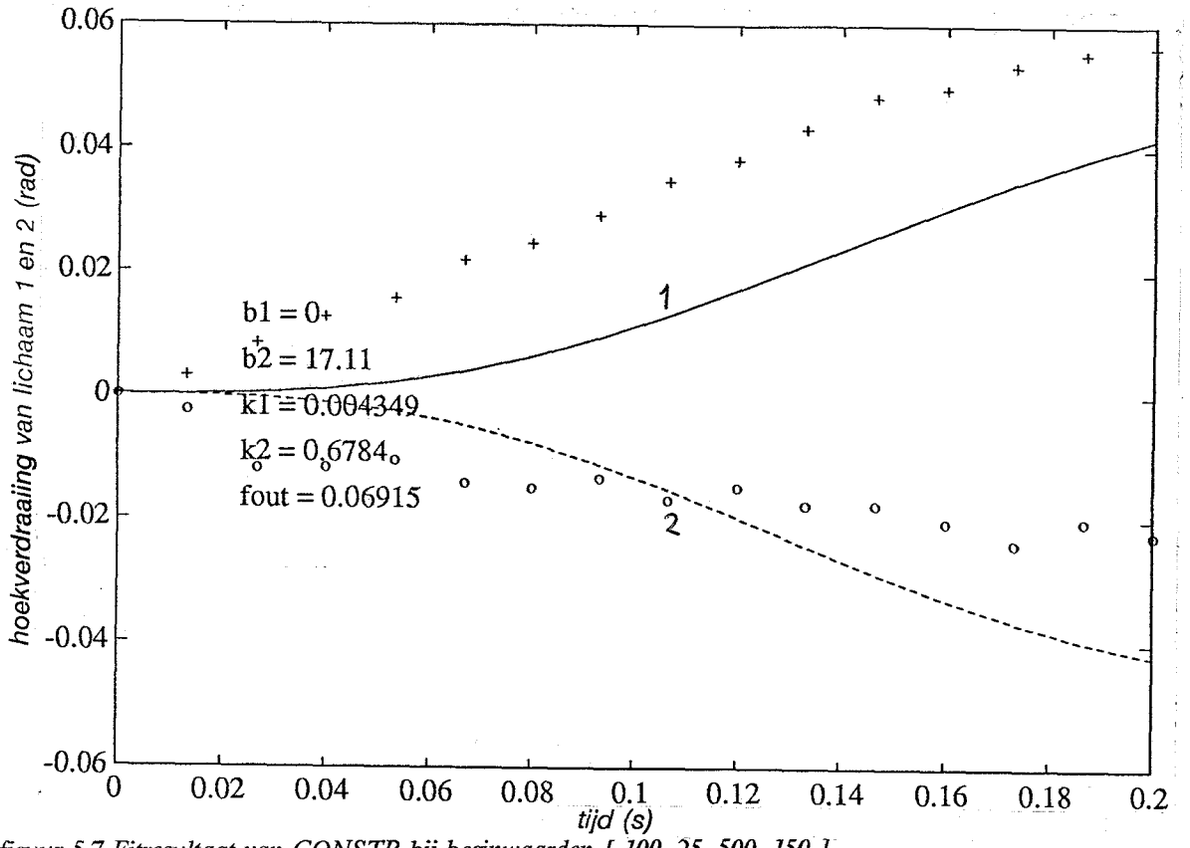


figuur 5.6 Fitresultaat van CONSTR bij beginwaarden [15, 75, 349, 199]

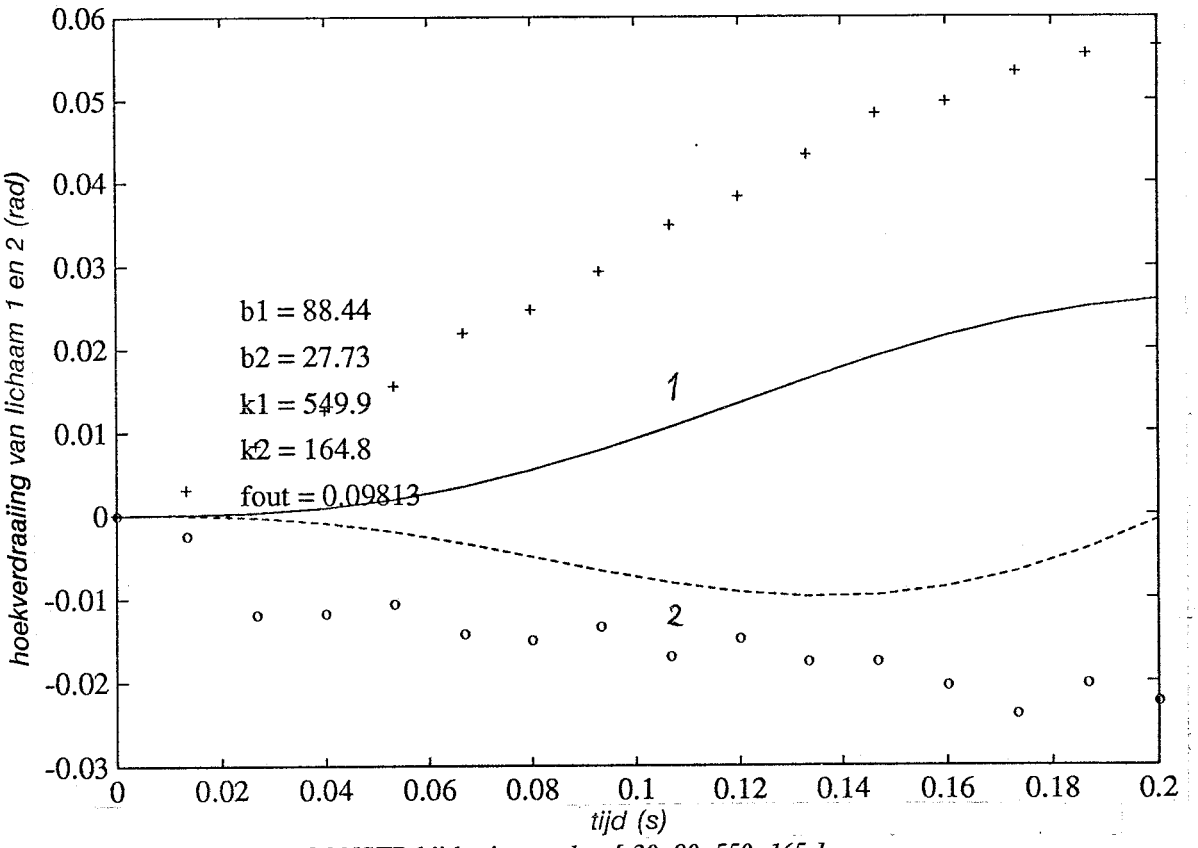
5.3 Parameterschatting op basis van de meetgegevens

De gegevens die door de twee bewegingswetenschappers gemeten zijn bestaan uit horizontale en vertikale verplaatsingen uitgedrukt in pixels. Bij deze gegevens behoren ook de gegevens van een vast referentiekader, zodat de maten in pixels om te rekenen zijn naar de standaard eenheid meter. Die meetgegevens (in meters) zijn gebruikt om de hoekverdraaiing van de 'slingerlichamen' te berekenen.
Voor deze berekening zijn alleen de meetgegevens in horizontale richting gebruikt aangezien daar de minste afwijkingen in verwacht kunnen worden. Doordat namelijk het referentiekader te ver buiten het vlak van beweging geplaatst is, zijn de omrekenfactoren te klein. Hierdoor zijn de meetgegevens buiten het centrum van het beeld onbetrouwbaar. De relatief kleine horizontale beweging heeft zich echter rond het midden van het beeld afgespeeld, waardoor de afwijking in die gegevens minimaal verondersteld mag worden.

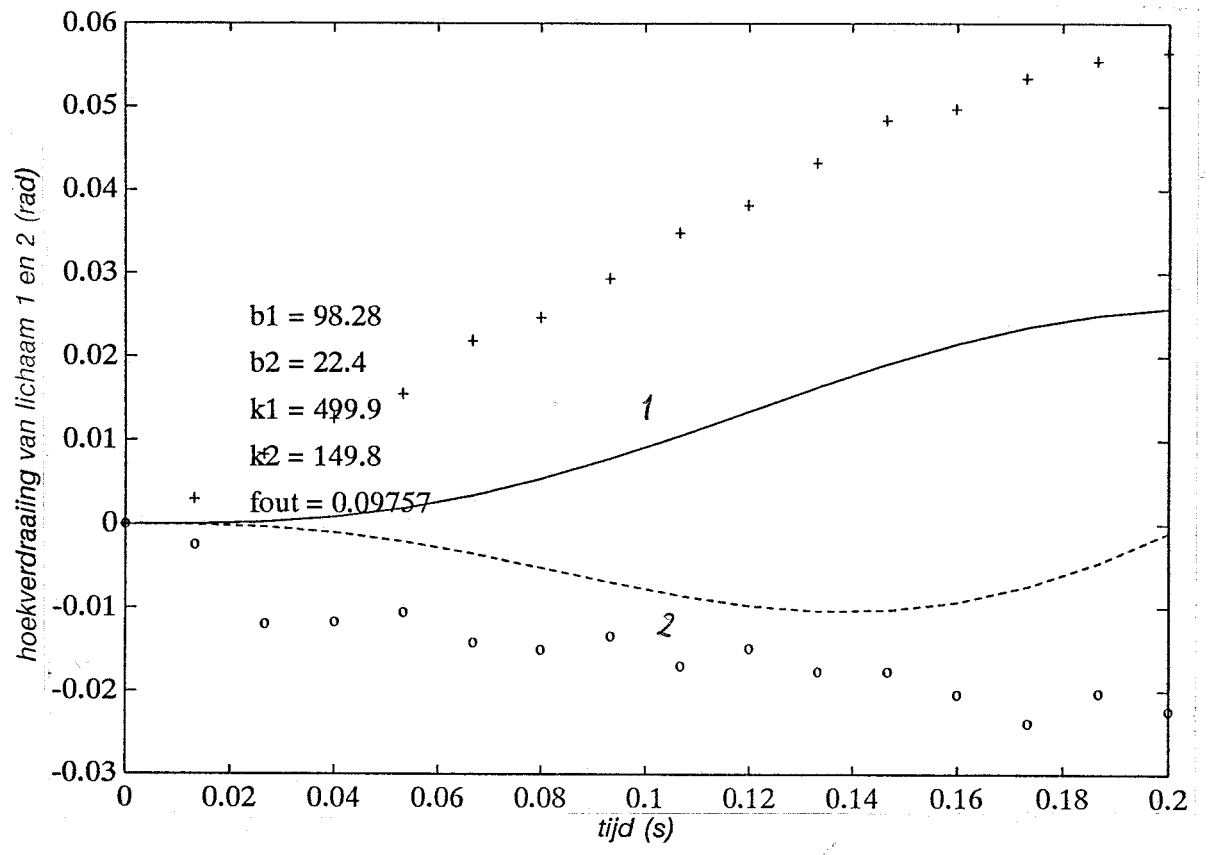
Het wiskundige model van de dubbelslinger (beschreven in paragraaf 3.1) is via de verschillende methoden (beschreven in paragraaf 5.2) gefit op de in de proefopstelling gemeten waarden, die omgerekend zijn naar hoekverdraaiingen. Vanwege de onbetrouwbaarheid van de fitprocedures is er met verschillende beginschattingen geprobeerd een goed passende oplossing te verkrijgen.
Uit de resultaten blijkt, zie de figuren 5.7 t/m 5.9, dat het dubbelslingermodel niet past op de meetresultaten. De gemeten hoekverdraaiingen zijn groter dan de berekende hoekverdraaiingen, zelfs in een systeem zonder rotatieveren en -dempers. Slechts met negatieve dempingsconstanten is het mogelijk de gemeten hoekverdraaiingen te realiseren, zoals in de figuren 5.10 t/m 5.12 te zien is.
Dit resultaat zou kunnen betekenen dat een actuator, in de vorm van spierwerking, de responsie op de verstoring vergroot. Op grond van het eveneens gemeten EMG (Electro-Myo-Gram) lijkt dit echter niet aannemelijk te zijn.



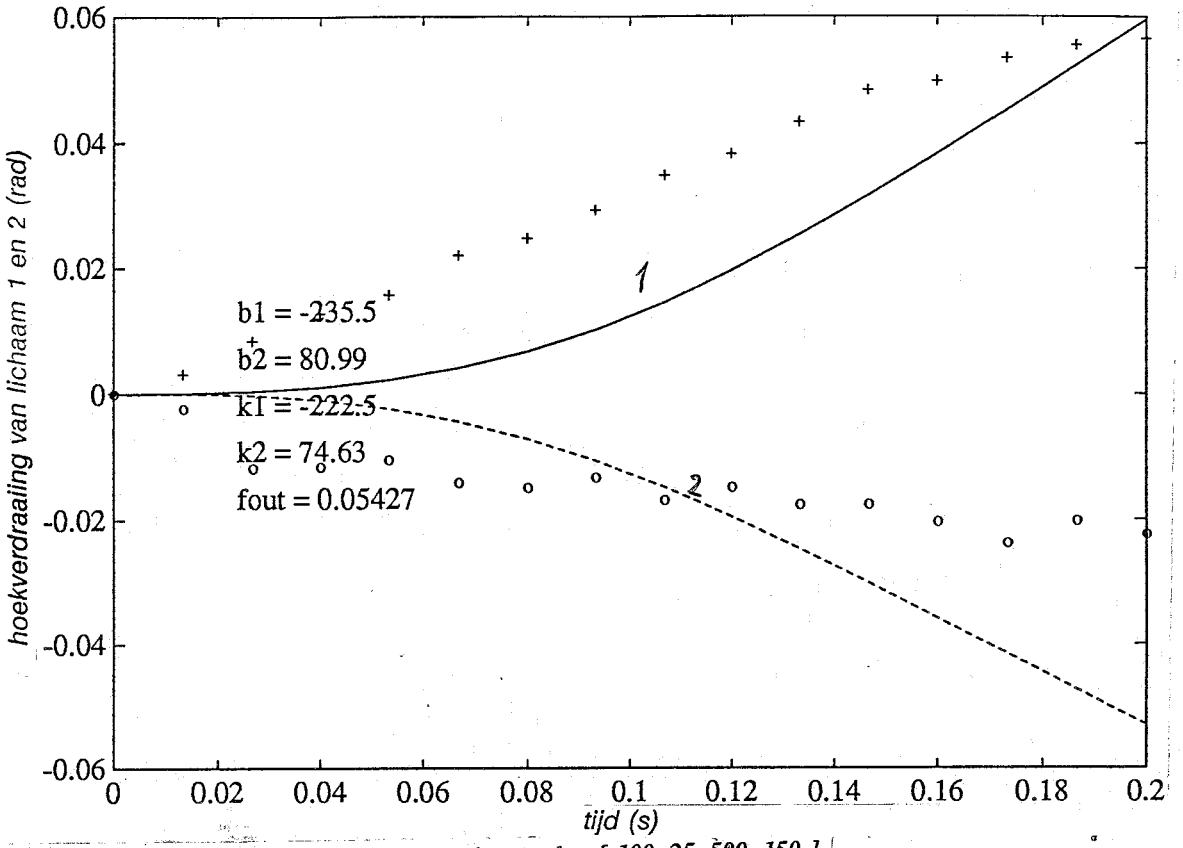
figuur 5.7 Fitresultaat van CONSTR bij beginwaarden [100, 25, 500, 150]



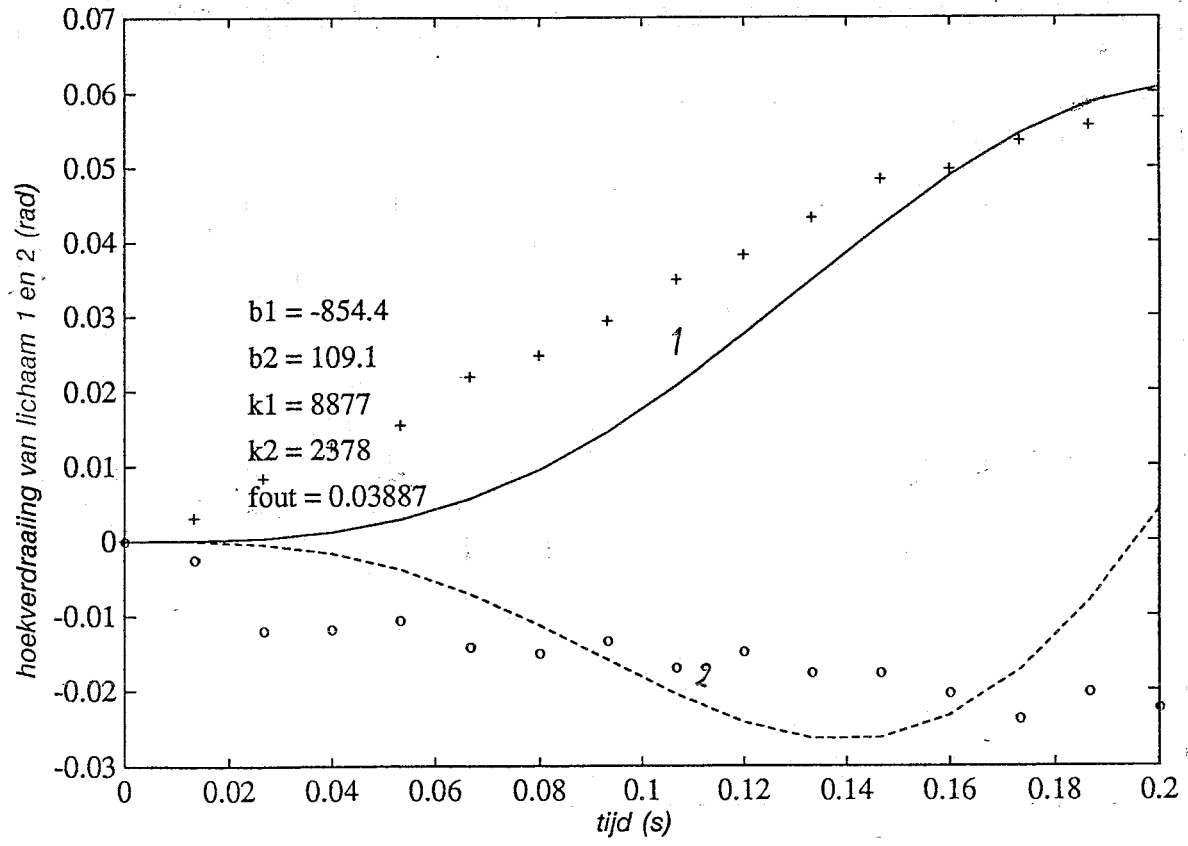
figuur 5.8 Fitresultaat van CONSTR bij beginwaarden [30, 90, 550, 165]



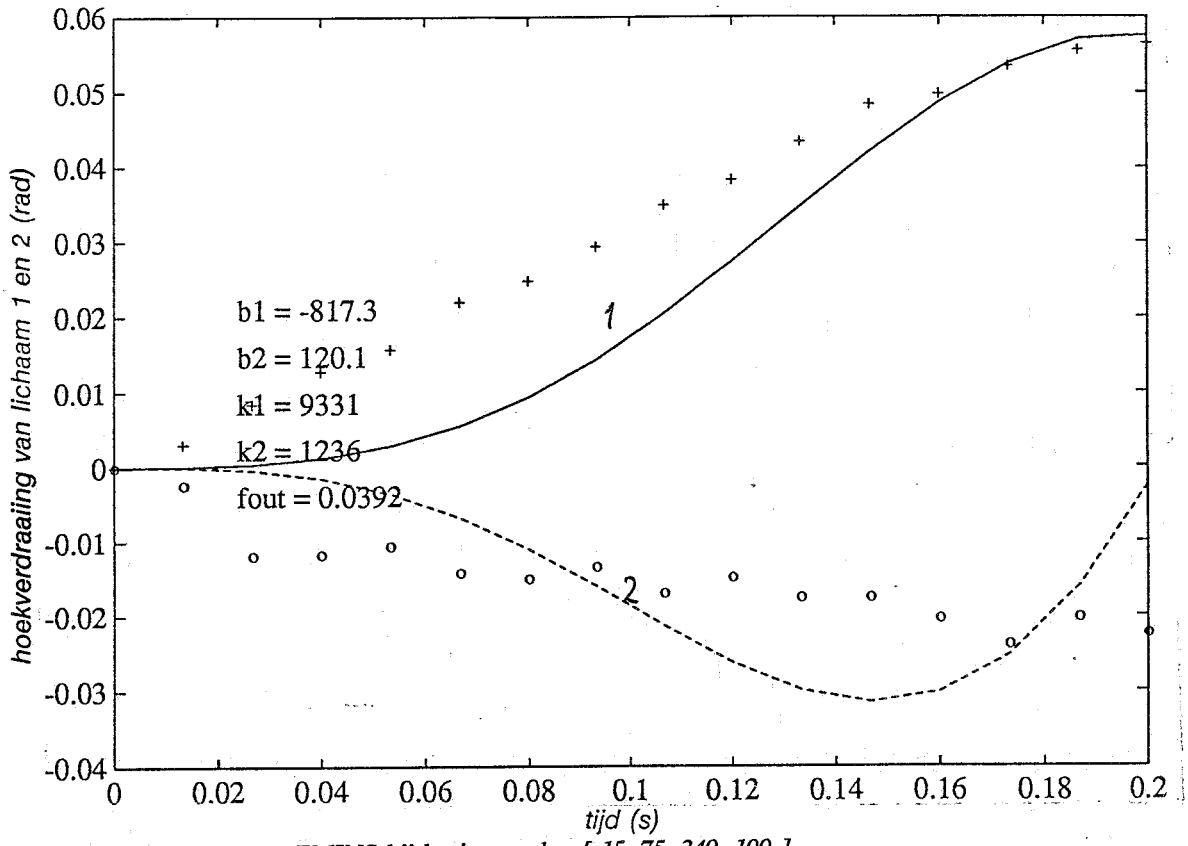
figuur 5.9 Fitresultaat van CONSTR bij beginwaarden [15, 75, 349, 199]



figuur 5.10 Fitresultaat van FMINS bij beginwaarden [100, 25, 500, 150]



figuur 5.11 Fitresultaat van FMINS bij beginwaarden [30, 90, 550, 165]

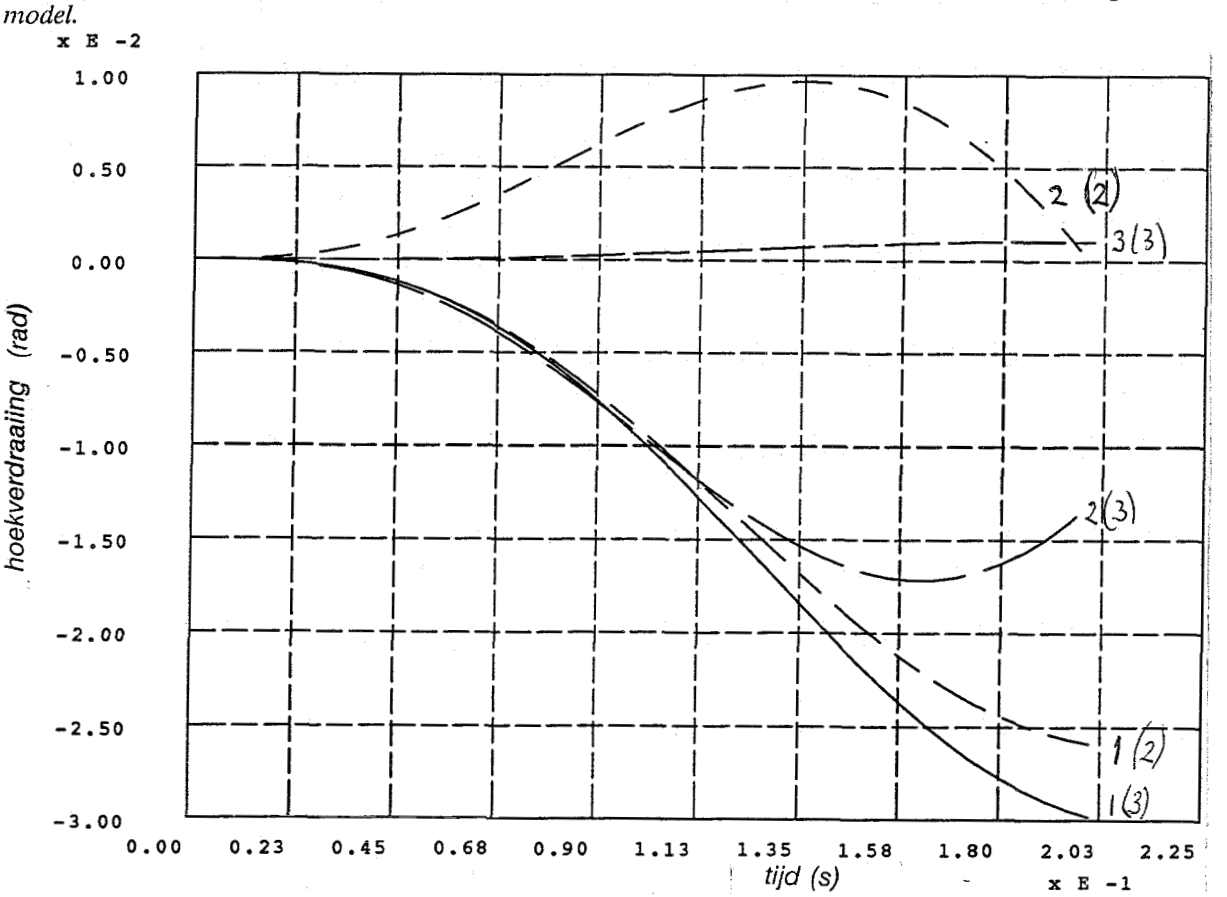


figuur 5.12 Fitresultaat van FMINS bij beginwaarden [15, 75, 349, 199]

In dit hoofdstuk worden de fouten en onzekerheden die in deze opdracht voorkomen besproken. In paragraaf 6.1 worden de tekortkomingen van het slingermodel besproken, in paragraaf 6.2 wordt ingegaan op de fitprocedures en in paragraaf 6.3 worden de onzekerheden van de meetgegevens beschouwd.

6.1 Het slingermodel

Het slingermodel is een grove benadering van het menselijk lichaam, dat afhankelijk van het aantal slingerlichamen meer of minder geschikt is voor modellering van het evenwichtsexperiment. Uit de meetresultaten is af te leiden dat de grootste hoekverdraaiingen plaats vinden in het enkel- en heupgewricht. Dit geldt dan vooral net nadat de verstoring plaats heeft gevonden. De invloed van een als afzonderlijk slingerlichaam gemodelleerd hoofd blijkt echter een grote invloed te hebben op het dynamisch gedrag van de overige slingerlichamen, zoals te zien is in figuur 6.1. In deze figuur zijn de resultaten van een simulatie met een tripelslingermodel en met een dubbelslingermodel zichtbaar.
De daarin gebruikte waarden voor de dempingsconstanten en veerstijfheden zijn als volgt: b_1 = 100 Nms/rad, b_2 = 25 Nms/rad, b_3 = 10 Nms/rad, k_1 = 500 Nm/rad, k_2 = 150 Nm/rad en k_3 = 50 Nm/rad. Uiteraard komen b_3 en k_3 alleen voor in het tripelslingermodel.
De hoekverdraaiing van het lijf, het middelste slingerlichaam, is nu veel groter dan bij het dubbelslingermodel.



figuur 6.1 Responsies van het tripel- en dubbelslingermodel

Zoals al in hoofdstuk 4 is opgemerkt is de aanname dat elk slingerlichaam met een homogene balk gemodelleerd kan worden niet in overeenstemming met de realiteit. Afhankelijk van het aantal slingerlichamen (en dus de gemodelleerde lichaamsdelen die dan daartoe behoren) is deze benadering beter. De massaverdeling (vooral de ligging van het massamiddelpunt) heeft een grote invloed op het modelgedrag en kan het dubbelslingermodel beter geschikt maken.
De aanname dat de slingerlichamen tussen de gewrichten, zoals ze in DADS gemodelleerd zijn, star zijn is een sterke vereenvoudiging. In werkelijkheid zullen de knieën en zeker ook de rug invloed uitoefenen op de reactie van het lichaam. Zie de gemeten responsie in bijlage 1, waaruit blijkt dat de marker op het hoofd van de proefpersoon eerst omhoog verplaatst alvorens omlaag te bewegen. De elasticiteit van de botten zelf zal echter niet van grote invloed zijn op het gedrag in de korte tijdsduur van het experiment.
De modellering van lineaire rotatieveren en -dempers in de gewrichten is eveneens discutabel vanwege de grote invloed die ze hebben op het dynamisch gedrag. Levende weefsels blijken sowieso vaak niet-lineaire eigenschappen te hebben. Voor een realistische invoering van niet-lineariteiten zou echter het gedrag van het spierskeletstelsel beter bekend moeten zijn. Daarbij is het voor de helderheid van en inzicht in het model duidelijker om de niet-lineariteiten niet te implementeren.

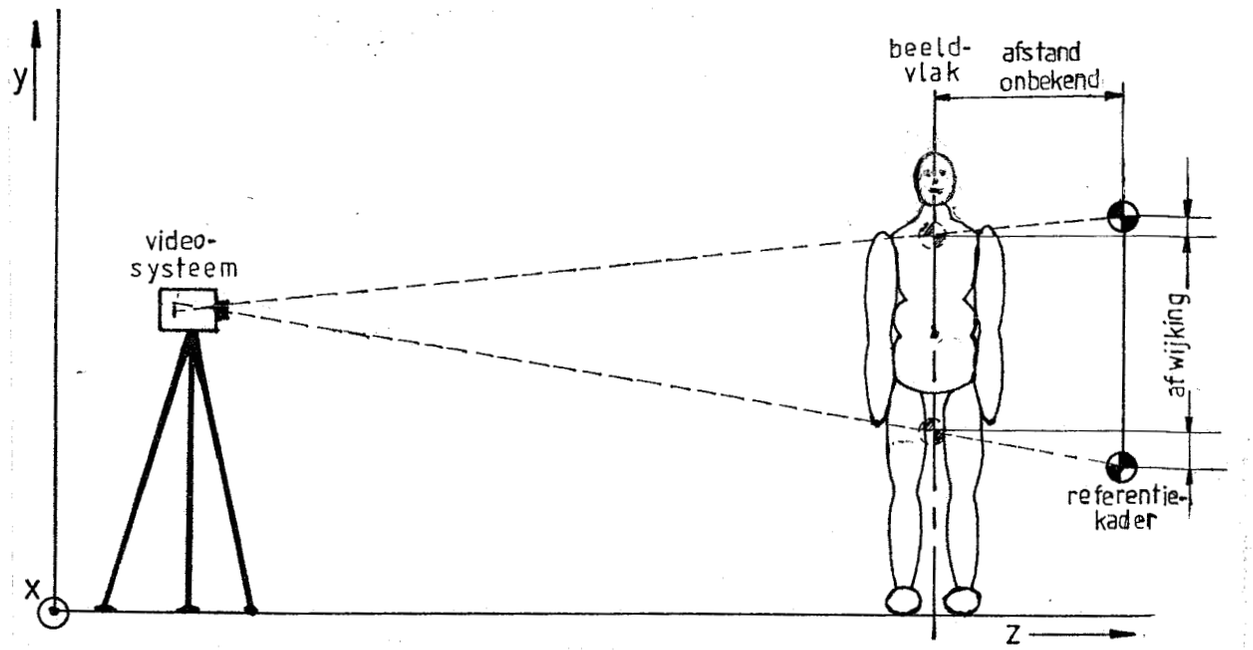
6.2 De fitprocedures

De gebruikte fitprocedures zijn, zoals al bleek in hoofdstuk 5, onbetrouwbaar. Geen van de gebruikte procedures is in staat het globale minimum van de fitfunctie te bepalen. De resultaten van de parameterschatting zijn dan ook niet alleszeggend. Bij de interpretatie van die resultaten is het van belang de startwaarden te kennen, evenals een globaal inzicht in het gedrag van het model te hebben. Zie de hoofdstukken 4 en 5.

6.3 De meetgegevens

De fitprocedure is toegepast op slechts een set meetgegevens en dien ten gevolge niet representatief voor het gehele experiment. De meetgegevens, horizontale en vertikale verplaatsingen van markers gemeten in pixels, zijn bovendien niet nauwkeurig. Deels ligt dit aan de onnauwkeurigheid (ruis) van het gebruikte videosysteem waarmee de verplaatsingen bepaald zijn en voor het overige wordt er met het omrekenen van pixels naar meters en van markerverplaatsingen naar hoekverdraaiingen van lichaamsdelen een fout geïntroduceerd. De onbetrouwbaarheid van deze meting is echter niet bepaald.
Het vaste kader met daarop de markers met een tussenafstand van 1 meter stond niet in het vlak van beweging, maar circa 1 meter erachter. De hieruit bepaalde omrekeningsfactor is dus te klein, vanwege de niet loodrechte hoek bij het afbeelden. Slechts in het centrum van het beeld is de omrekeningsfactor bij benadering goed maar naar de randen toe wordt de afwijking echter steeds groter. Dit is weergegeven in figuur 6.2. Deze afwijking zou te berekenen zijn als alle afstanden en posities bekend zouden zijn, hetgeen helaas niet het geval is. Deze fout is bij alle meetsets aanwezig, hoewel niet bij alle even groot.

Bij de omrekening van verplaatsingen naar hoekverdraaiingen is gebruik gemaakt van de horizontale verplaatsingen (de kleinst mogelijke fout) en de afstand tussen de markers. Hierbij is ook de afstand tussen de markers niet tijdens het experiment gemeten, maar is die achteraf nagemeten bij de proefpersoon. In het minst gunstigste geval kan de fout bij deze omrekening verdubbelen.
De fouten die bij de berekening van de te fitten meetgegevens optreden maken een betrouwbare uitspraak over de mogelijkheid de juiste parameters te schatten onmogelijk, als er al een betrouwbare fitprocedure zou zijn.



figuur 6.2 Vertekening van het referentiekader bij de gebruikte opstelling

* *Betreffende het enkel- of dubbelslingermodel*

Het enkelslingermodel is een te eenvoudige weergave van de realiteit om de beweging van met name het hoofd in beeld te brengen. Het is dan ook niet geschikt om als simulatiemodel voor het evenwichtsexperiment te gebruiken.

Het dubbelslingermodel is waarschijnlijk niet realistisch genoeg om als simulatiemodel te gebruiken, indien aangenomen mag worden dat het 95%-betrouwbaarheidsinterval niet te groot is. De optredende hoekverdraaiingen zijn te klein (ook zonder enige veerwerking en demping) om de 'gemeten' waarden te benaderen, hoewel met een andere massaverdeling grotere hoekverdraaiingen te simuleren zijn. Het dubbelslingermodel kan ook voldoen indien er een actuatorwerking in een of beide gewrichten wordt verondersteld.

Het gedrag van meervoudige slingermodellen is moeilijk te voorspellen. Het tripelslingermodel heeft een ander gedrag (grotere hoekverdraaiingen) en kan mogelijkerwijs beantwoorden aan de gemeten realiteit.

* *Betreffende het lineaire of niet-lineaire model*

De 'gemeten' hoekverdraaiingen zijn zo klein (<0.06 rad), dat linearisatie van de goniometrische functies een fout oplevert die verwaarloosbaar is ten opzichte van de overige fouten. De optredende hoeksnelheden zijn eveneens zo klein dat het weglaten van niet lineaire termen een verwaarloosbare fout introduceert.
De in de overige experimenten gebruikte verstoringen geven geen aanleiding tot het gebruik van het niet-lineaire model.

* *Algemene conclusies*

PC-Matlab is geschikt voor het doorrekenen van het wiskundige model. Hiermee wordt dezelfde responsie bepaald als met het programma DADS.
De in PC-Matlab aanwezige fitprocedures (die ik gebruikt heb) zijn niet geschikt voor het fitten van het dubbelslingermodel (en tripelslingermodel) op de meetgegevens.
Er is niet met enige zekerheid te zeggen of een model past op de meetgegevens, zolang de hoekverdraaiingen niet correct te berekenen zijn.

* *Aanbevelingen*

PC-Matlab is geschikt voor de parameterschatting van het dubbelslingermodel en het tripelslingermodel indien er een geschikte fitprocedure in geïmplementeerd wordt. Parameterschatting via de toestandsbeschrijving en het gebruik van een Kalman-filter zou ook tot goede resultaten kunnen leiden.
Onderzocht kan worden of het dubbelslingermodel met een andere massaverdeling (massamiddelpunt dichter bij het heupgewricht) wel aan de gemeten responsies kan voldoen. Indien dat niet mogelijk is kan het tripelslingermodel uitkomst bieden. Het wiskundige model van de dubbelslinger dient daartoe uitgebreid worden naar een tripelslingermodel.
Het tripelslingermodel is al in DADS gemodelleerd en wacht op goed geschatte modelparameters.

Een betrouwbare verificatie van het slingermodel is mogelijk indien de meetgegevens op reële schaal in meters uitgedrukt kunnen worden en de onnauwkeurigheid in die gegevens bekend is. Hiervoor is het nodig het videosysteem nogmaals op te stellen om de goede omrekeningsfactor van pixels naar meters te bepalen. Het berekenen van de hoekverdraaiingen kan dan nauwkeuriger gebeuren door gebruik te maken van zowel de horizontale alsook de vertikale markerverplaatsingen. De juiste markerafstanden zijn dan ook te bepalen uit de meetgegevens.

Aangezien het hoofd alleen versnellingen registreert en niet de verplaatsingen verdient het de voorkeur om voor de parameterschatting en de verificatie gebruik te maken van de met behulp van versnellingsopnemers gemeten versnellingen van het hoofd. Mogelijkerwijs is het daarmee wel mogelijk om een van de standaard PC-Matlab fitprocedures te gebruiken met een betrouwbaar resultaat.

Geursen J.B, Altena D., Massen C.H & Verduin M. (1976) " A model of the standing man for the description of his dynamic behaviour ".
Agressologie 1976, 17, B:62-69

Kodde L., Caberg H.B., Mol J.M.F. & Massen C.H. (1982) " An application of mathematical models in posturography ".
J. Biomed. Engng. 1982, Vol. 4, January

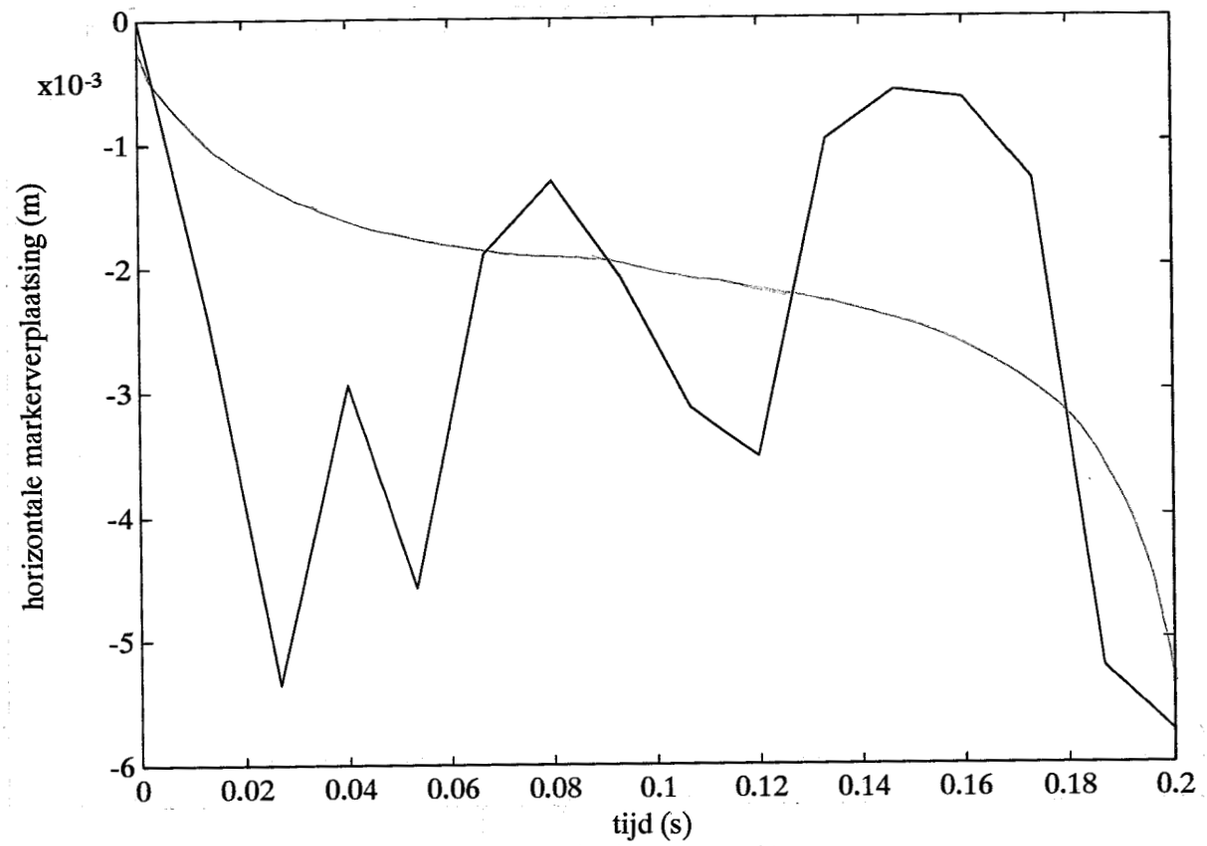
Kearney R.E. & Hunter I.W. (1990) " System identification of human joint dynamics "
J. Biomed. Engng. 1990, Vol. 18, Issue 1

Baughman L.D. (1983) " Development of an interactive computer program to produce body description data "
(GEBOD)
Report Air Force Aerospace Medical Research Laboratory, AFAMRL-TR-83-058, july 1983

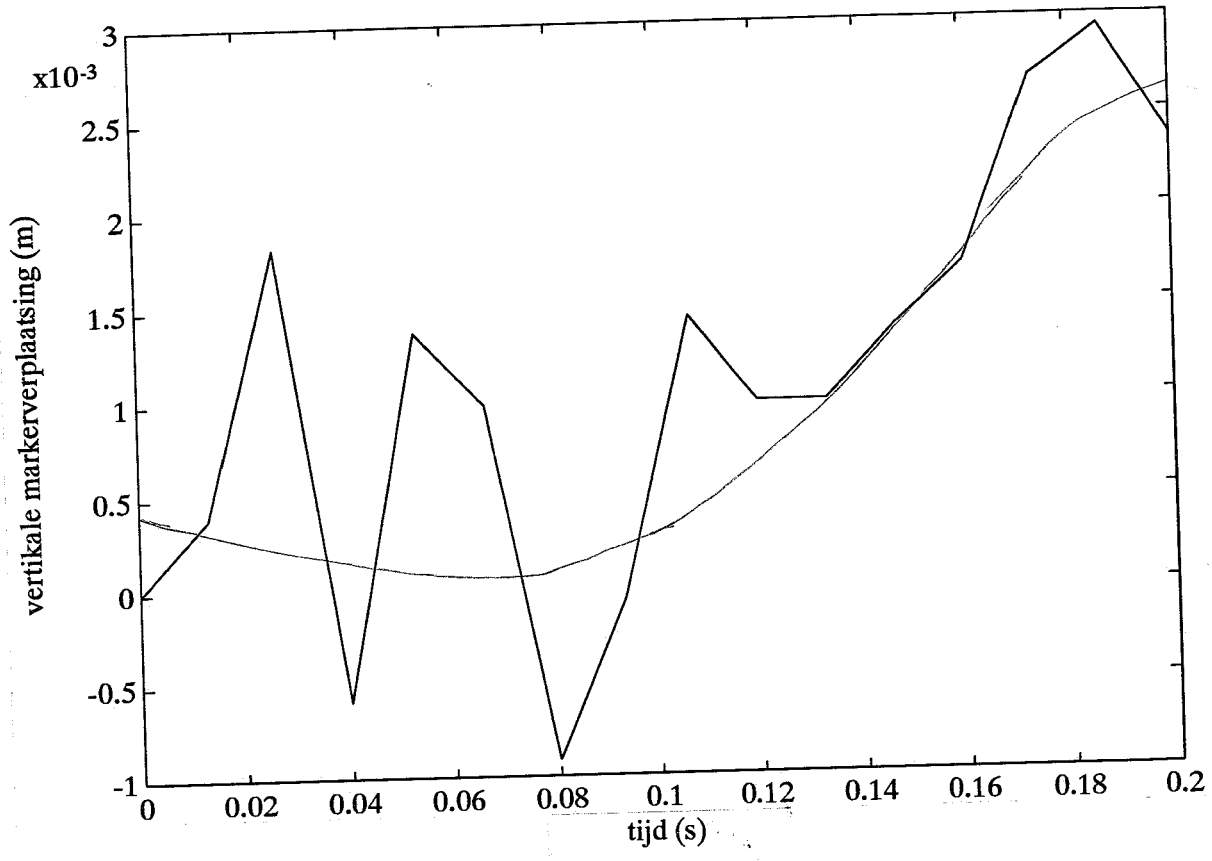
Sauren A. (1988) " Multibody dynamica "
Technische Universiteit Eindhoven, dictaatnr. 4659, september 1990

BIJLAGE 1

Horizontale en vertikale verplaatsing van het hoofd



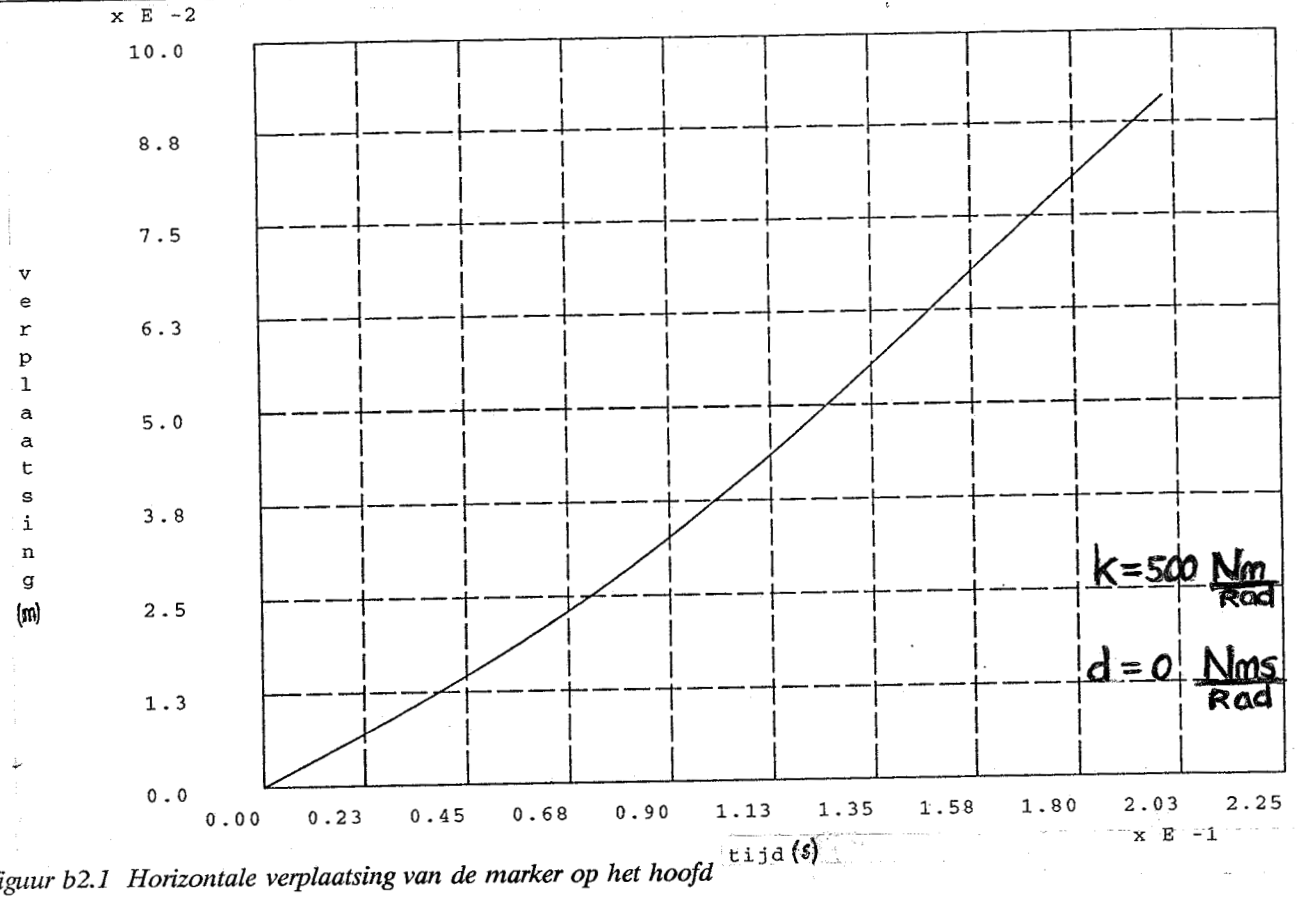
figuur b1.1 Horizontale verplaatsing van de marker op het hoofd



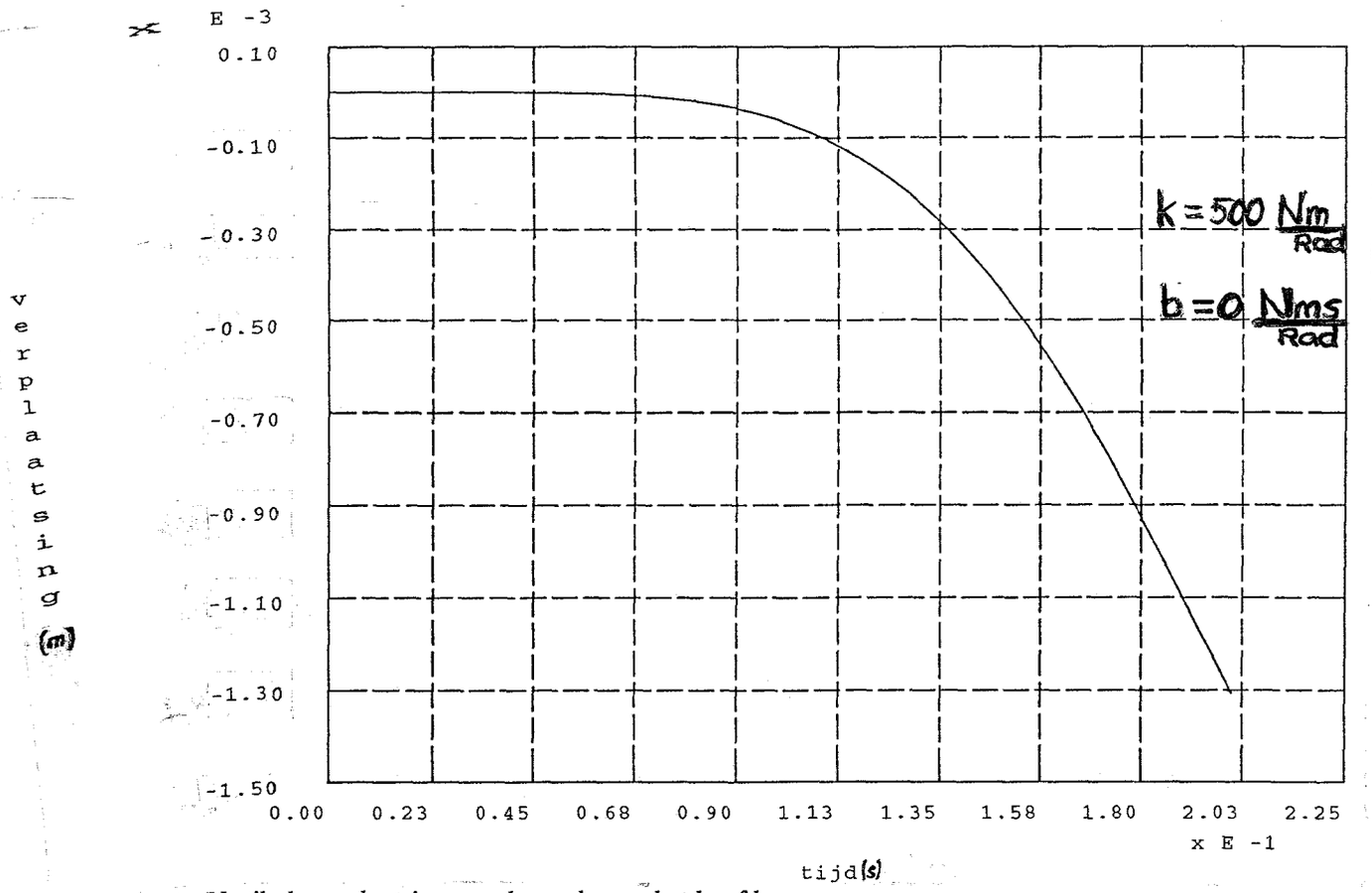
figuur b1.2 Vertikale verplaatsing van de marker op het hoofd

BIJLAGE 2

Horizontale en vertikale verplaatsing van het hoofd, zoals berekend met DADS

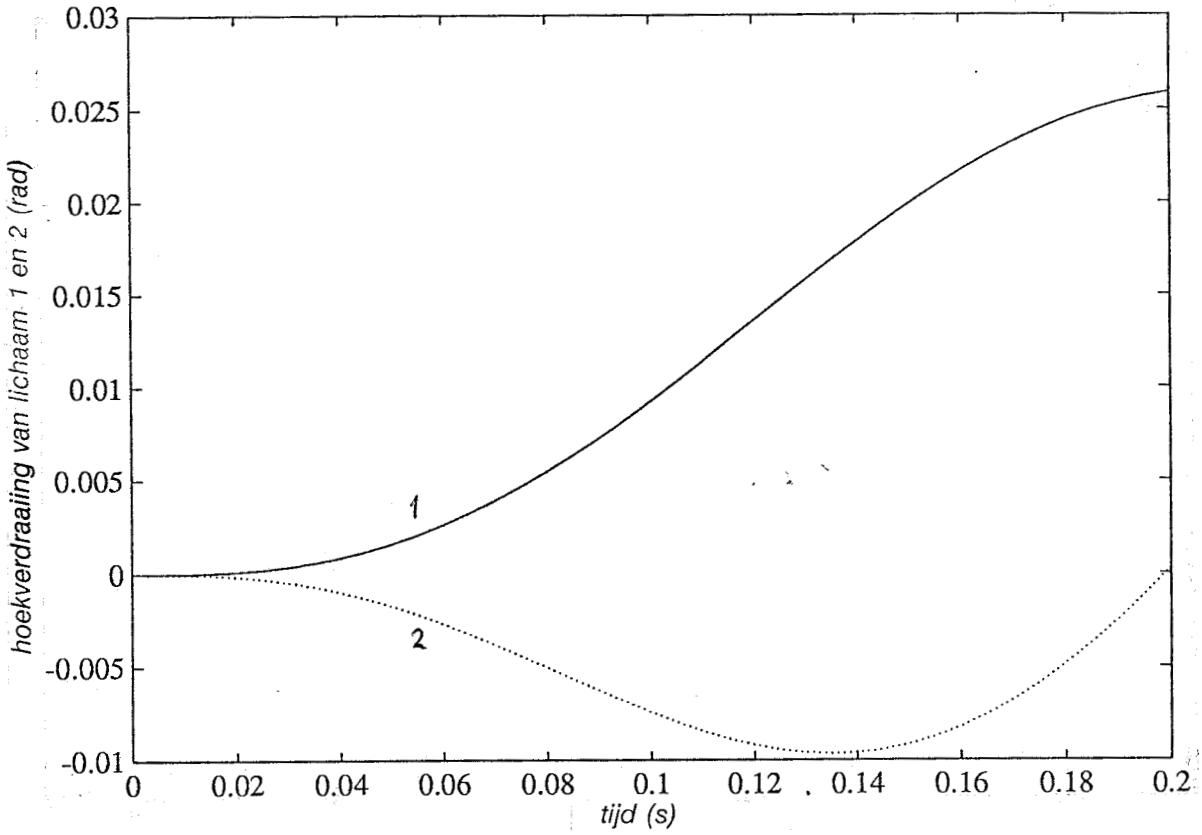


figuur b2.1 Horizontale verplaatsing van de marker op het hoofd



figuur b2.2 Vertikale verplaatsing van de marker op het hoofd

De hoekverdraaiing van het DADS-dubbelslingermodel



figuur b3.1 Responsie van het dubbelslingermodel met constanten: [100,25,500,150]

Controleberekening m.b.v. de vergelijkingen van Lagrange

Voor de kinetische energie (T) geldt: (index = nummer lichaam)

$$T_1 = \tfrac{1}{2} \int_{M_1} \dot{\vec{r}}_1 \cdot \dot{\vec{r}}_1 \, dm + \tfrac{1}{2} J_1 \vec{\omega}_1 \cdot \vec{\omega}_1$$

$$T_2 = \tfrac{1}{2} \int_{M_2} \dot{\vec{r}}_2 \cdot \dot{\vec{r}}_2 \, dm + \tfrac{1}{2} J_2 \vec{\omega}_2 \cdot \vec{\omega}_2$$

$$T = T_1 + T_2$$

Met de bekende $\vec{r}_1$ en $\vec{r}_2$ volgen na eenmaal differentiëren $\dot{\vec{r}}_1$ en $\dot{\vec{r}}_2$

$$\dot{\vec{r}}_1 = \begin{bmatrix} \dot{x}_A - h_1 \dot{\varphi}_1 \cos\varphi_1 \\ -h_1 \dot{\varphi}_1 \sin\varphi_1 \\ 0 \end{bmatrix}$$

$$\dot{\vec{r}}_2 = \begin{bmatrix} \dot{x}_A - l_1 \dot{\varphi}_1 \cos\varphi_1 - h_2 (\dot{\varphi}_1 + \dot{\varphi}_2) \cos(\varphi_1 + \varphi_2) \\ -l_1 \dot{\varphi}_1 \sin\varphi_1 - h_2 (\dot{\varphi}_1 + \dot{\varphi}_2) \sin(\varphi_1 + \varphi_2) \\ 0 \end{bmatrix}$$

Waarmee voor T volgt:

$$T = \tfrac{1}{2}(m_1 + m_2)\dot{x}_A^2 - (m_1 h_1 \dot{\varphi}_1 \cos\varphi_1 + m_2 l_1 \dot{\varphi}_1 \cos\varphi_1 + m_2 h_2 (\dot{\varphi}_1 + \dot{\varphi}_2) \cdot \cos(\varphi_1 + \varphi_2)) \dot{x}_A + \tfrac{1}{2}(m_1 h_1^2 + m_2 l_1^2)\dot{\varphi}_1^2 + \tfrac{1}{2} J_1 \dot{\varphi}_1^2 + \tfrac{1}{2} m_2 h_2^2 (\dot{\varphi}_1 + \dot{\varphi}_2)^2 + \tfrac{1}{2} J_2 (\dot{\varphi}_1 + \dot{\varphi}_2)^2 + m_2 l_1 h_2 \dot{\varphi}_1 (\dot{\varphi}_1 + \dot{\varphi}_2) \cdot (\cos\varphi_1 \cdot \cos(\varphi_1 + \varphi_2) + \sin\varphi_1 \sin(\varphi_1 + \varphi_2))$$

De gegeneraliseerde vrijheidsgraden zijn φ_1 en φ_2 :

$$\underline{q} = \begin{bmatrix} \varphi_1 \\ \varphi_2 \end{bmatrix} \qquad \dot{\underline{q}} = \begin{bmatrix} \dot{\varphi}_1 \\ \dot{\varphi}_2 \end{bmatrix}$$

$$T_{,\dot{\underline{q}}} = \begin{bmatrix} -(m_1 h_1 \cos\varphi_1 + m_2 l_1 \cos\varphi_1 + m_2 h_2 \cos(\varphi_1+\varphi_2))\dot{x}_A + (m_1 h_1^2 + m_2 l_1^2)\dot{\varphi}_1 + \\ + J_1 \dot{\varphi}_1 + m_2 h_2^2 (\dot{\varphi}_1 + \dot{\varphi}_2) + J_2 (\dot{\varphi}_1 + \dot{\varphi}_2) + m_2 l_1 h_2 \cos\varphi_2 (\dot{\varphi}_1 + \dot{\varphi}_2) + \\ m_2 l_1 h_2 \cos\varphi_2 \dot{\varphi}_1 \\ \\ - m_2 h_2 \cos(\varphi_1+\varphi_2)\dot{x}_A + m_2 h_2^2 (\dot{\varphi}_1 + \dot{\varphi}_2) + J_2 (\dot{\varphi}_1 + \dot{\varphi}_2) + \\ m_2 l_1 h_2 \cos\varphi_2 \dot{\varphi}_1 \end{bmatrix}$$

$$\frac{d}{dt} T_{,\dot{\underline{q}}} = \begin{bmatrix} -(m_1 h_1 \cos\varphi_1 + m_2 l_1 \cos\varphi_1 + m_2 h_2 \cos(\varphi_1+\varphi_2))\ddot{x}_A + (m_1 h_1 \dot{\varphi}_1 \sin\varphi_1 + \\ m_2 l_1 \dot{\varphi}_1 \sin\varphi_1 + m_2 h_2 (\dot{\varphi}_1+\dot{\varphi}_2)\sin(\varphi_1+\varphi_2))\dot{x}_A + (m_1 h_1^2 + m_2 l_1^2)\ddot{\varphi}_1 + \\ J_1 \ddot{\varphi}_1 + m_2 h_2^2 (\ddot{\varphi}_1 + \ddot{\varphi}_2) + J_2 (\ddot{\varphi}_1 + \ddot{\varphi}_2) - m_2 l_1 h_2 \dot{\varphi}_2 \sin\varphi_2 (\dot{\varphi}_1 + \dot{\varphi}_2) + \\ m_2 l_1 h_2 \cos\varphi_2 (\ddot{\varphi}_1 + \ddot{\varphi}_2) + m_2 l_1 h_2 \cos\varphi_2 \ddot{\varphi}_1 - m_2 l_1 h_2 \dot{\varphi}_2 \sin\varphi_2 \dot{\varphi}_1 \\ \\ - m_2 h_2 \cos(\varphi_1+\varphi_2)\ddot{x}_A + m_2 h_2 (\dot{\varphi}_1 + \dot{\varphi}_2)\sin(\varphi_1+\varphi_2)\dot{x}_A + \\ m_2 h_2^2 (\ddot{\varphi}_1 + \ddot{\varphi}_2) + J_2 (\ddot{\varphi}_1 + \ddot{\varphi}_2) + m_2 l_1 h_2 \cos\varphi_2 \ddot{\varphi}_1 - m_2 l_1 h_2 * \\ \sin\varphi_2 \dot{\varphi}_2 \dot{\varphi}_1 \end{bmatrix}$$

$$T_{,\underline{q}} = \begin{bmatrix} (m_1 h_1 \dot{\varphi}_1 \sin\varphi_1 + m_2 l_1 \dot{\varphi}_1 \sin\varphi_1 + m_2 h_2 (\dot{\varphi}_1 + \dot{\varphi}_2)\sin(\varphi_1+\varphi_2))\dot{x}_A \\ \\ m_2 h_2 (\dot{\varphi}_1 + \dot{\varphi}_2)\sin(\varphi_1+\varphi_2)\dot{x}_A - m_2 l_1 h_2 \dot{\varphi}_1 (\dot{\varphi}_1 + \dot{\varphi}_2)\sin\varphi_2 \end{bmatrix}$$

Voor de potentiële energie (V) geldt:

$$V = m_1 h_1 g \cos\varphi_1 + m_2 g (l_1 \cos\varphi_1 + h_2 \cos(\varphi_1+\varphi_2)) + \tfrac{1}{2} k_1 \varphi_1^2 + \tfrac{1}{2} k_2 \varphi_2^2$$

Voor de gegeneraliseerde momenten (Q^*) geldt:

$$Q^* = \begin{bmatrix} -b_1 \dot{\varphi}_1 \\ -b_2 \dot{\varphi}_2 \end{bmatrix}$$

Volgens Lagrange geldt:

$$\frac{d}{dt} T_{,\dot{q}} - T_{,q} + V_{,q} = Q^*$$

$$\begin{bmatrix} (-(m_1 h_1 + m_2 l_1)\cos\varphi_1 + m_2 h_2 \cos(\varphi_1+\varphi_2))\ddot{x}_A + \\ (m_1 h_1^2 + m_2 l_1^2 + J_1 + m_2 h_2^2 + J_2 + 2 m_2 l_1 h_2 \cos\varphi_2)\ddot{\varphi}_1 + \\ (m_2 h_2^2 + J_2 + m_2 l_1 h_2 \cos\varphi_2)\ddot{\varphi}_2 - m_2 l_1 h_2 \sin\varphi_2 (\dot{\varphi}_2^2 + 2\dot{\varphi}_1 * \dot{\varphi}_2) \\ \\ -m_2 h_2 \cos(\varphi_1+\varphi_2)\ddot{x}_A + (m_2 h_2^2 + J_2 + m_2 l_1 h_2 \cos\varphi_2)\ddot{\varphi}_1 + \\ (m_2 h_2^2 + J_2)\ddot{\varphi}_2 + m_2 l_1 h_2 \sin\varphi_2 \dot{\varphi}_1^2 \end{bmatrix} +$$

$$- \begin{bmatrix} m_1 h_1 g \sin\varphi_1 + m_2 g (l_1 \sin\varphi_1 + h_2 \sin(\varphi_1+\varphi_2)) - k_1 \varphi_1 \\ \\ m_2 h_2 g \sin(\varphi_1+\varphi_2) - k_2 \varphi_2 \end{bmatrix} =$$

$$\begin{bmatrix} -b_1 \dot{\varphi}_1 \\ -b_2 \dot{\varphi}_2 \end{bmatrix}$$

Bewegingsvergelijking ⑤ volgt hieruit door de onderste vergelijking van de bovenste vergelijking af te trekken en vergelijking ⑥ is identiek aan de onderste vergelijking.

BIJLAGE 5

De gebruikte invoerfiles voor de berekeningen met DADS

TEST.VB2

```
ANALYSIS
CREATE SYSTEM.DATA
  UNITS                    := 'SI'
  ANALYSIS.TYPE            := 'DYNAMIC'
  STARTING.TIME            := '0.0'
  ENDING.TIME              := '0.2'
  PRINT.INTERVAL           := '0.01'
  GRAVITY.SEA.LEVEL        := '9.80665'
  X.GRAVITY                := '0.0'
  Y.GRAVITY                := '-1.0'
  SCALE.GRAVITY.COEF       := '1.0'
  MATRIX.OPERATIONS        := 'SPARSE'
  REDUNDANCY.CHECK         := 'TRUE'
  LU.TOL                   := '1.0D-12'
  ASSEMBLY.TOL             := '1.0D-3'
  BYPASS.ASSEMBLY          := 'FALSE'
  OUTPUT.FILE              := 'BINARY'
  REFERENCE.FRAME          := 'LOCAL'
  DEBUG.FLAG               := 'TRUE'
UP
CREATE DYNAMIC.DATA
  REACTION.FORCES          := 'FALSE'
  FORCE.COORDINATES        := 'GLOBAL'
  PRINT.METHOD             := 'INTERPOLATED'
  MAX.INT.STEP             := '0.005'
  SOLUTION.TOL             := '0.001'
  INTEGRATION.TOL          := '0.001'
UP
UP
FORCE
CREATE RSDA
  NAME                     := 'VEER1'
  JOINT.NAME               := 'ENKEL'
  ORIENTATION.ANGLE        := '0.0'
  SPRING.CONSTANT          := '500'
  DAMPING.COEFFICIENT      := '100'
  ACTUATOR.TORQUE          := '0.0'
  CURVE.SPRING             := 'NONE'
  CURVE.DAMPER             := 'NONE'
  CURVE.ACTUATOR           := 'NONE'
  ANGULAR.UNITS            := 'RADIANS'
UP
CREATE TSDA
  NAME                     := 'TSDA1'
  BODY.1.NAME              := 'VOET'
  BODY.2.NAME              := 'ZOOL'
  SPRING.CONSTANT          := '1.0D12'
  FREE.LENGTH.SPRING       := '0.001'
  DAMPING.COEFFICIENT      := '0.0'
  ACTUATOR.FORCE           := '0.0'
```

```
  P.ON.BODY.1                  := ( 0.0, 0.0 )
  P.ON.BODY.2                  := ( 0.0, 0.0 )
  Q.ON.BODY.1                  := ( 1.0, 0.0 )
  Q.ON.BODY.2                  := ( 1.0, 0.0 )
  CURVE.SPRING                   := 'NONE'
  CURVE.DAMPER                   := 'NONE'
  CURVE.ACTUATOR                 := 'NONE'
  NODE.1                       := '0'
  NODE.2                       := '0'
UP
UP
JOINTS
CREATE REVOLUTE.JOINT
  NAME                         := 'ENKEL'
  BODY.1.NAME                    := 'LIJF'
  BODY.2.NAME                    := 'VOET'
  P.ON.BODY.1                  := ( 0, -0.84 )
  P.ON.BODY.2                  := ( 0.0, 0.0 )
  Q.ON.BODY.1                  := ( 1, -0.84 )
  Q.ON.BODY.2                  := ( 1.0, 0.0 )
  NODE.1                       := '0'
  NODE.2                       := '0'
UP
CREATE TRANSLATIONAL.JOINT
  NAME                         := 'TR1'
  BODY.1.NAME                    := 'ZOOL'
  BODY.2.NAME                    := 'GROND'
  P.ON.BODY.1                  := ( 0.0, 0.0 )
  P.ON.BODY.2                  := ( 0.0, 0.0 )
  Q.ON.BODY.1                  := ( 1.0, 0.0 )
  Q.ON.BODY.2                  := ( 1.0, 0.0 )
  NODE.1                       := '0'
  NODE.2                       := '0'
UP
CREATE TRANSLATIONAL.JOINT
  NAME                         := 'TR2'
  BODY.1.NAME                    := 'VOET'
  BODY.2.NAME                    := 'ZOOL'
  P.ON.BODY.1                  := ( 0.0, 0.0 )
  P.ON.BODY.2                  := ( 0.0, 0.0 )
  Q.ON.BODY.1                  := ( 1.0, 0.0 )
  Q.ON.BODY.2                  := ( 1.0, 0.0 )
  NODE.1                       := '0'
  NODE.2                       := '0'
UP
UP
CREATE BODY
  NAME                         := 'GROND'
  CENTER.OF.GRAVITY              := ( 0.0, 0.0 )
  PHI                        := '0.0'
  FIXED.TO.GROUND                := 'TRUE'
  MASS                       := '1.0'
  INERTIA                      := '1.0'
  XG.FORCE                       := '0.0'
  YG.FORCE                       := '0.0'
  TORQUE.CONSTANT                := '0.0'
```

```
  CURVE.XGF                := 'NONE'
  CURVE.YGF                := 'NONE'
  CURVE.TORQUE             := 'NONE'
  OUTLINE.SHAPE            := 'NONE'
  SHAPE.CENTER             := ( 0.0, 0.0 )
  ANGULAR.UNITS            := 'RADIANS'
  FLEXIBLE                 := 'FALSE'
  SUPERELEMENT             := 'FALSE'
UP
CREATE BODY
  NAME                     := 'LIJF'
  CENTER.OF.GRAVITY        := ( 0.0, 0.0 )
  PHI                      := '0.0'
  FIXED.TO.GROUND          := 'FALSE'
  MASS                     := '63'
  INERTIA                  := '59.3'
  XG.FORCE                 := '0.0'
  YG.FORCE                 := '0.0'
  TORQUE.CONSTANT          := '0.0'
  CURVE.XGF                := 'NONE'
  CURVE.YGF                := 'NONE'
  CURVE.TORQUE             := 'NONE'
  OUTLINE.SHAPE            := 'NONE'
  SHAPE.CENTER             := ( 0.0, 0.0 )
  ANGULAR.UNITS            := 'RADIANS'
  FLEXIBLE                 := 'FALSE'
  SUPERELEMENT             := 'FALSE'
UP
CREATE BODY
  NAME                     := 'VOET'
  CENTER.OF.GRAVITY        := ( 0.0, 0.0 )
  PHI                      := '0.0'
  FIXED.TO.GROUND          := 'FALSE'
  MASS                     := '0.001'
  INERTIA                  := '0.0001'
  XG.FORCE                 := '0.0'
  YG.FORCE                 := '0.0'
  TORQUE.CONSTANT          := '0.0'
  CURVE.XGF                := 'NONE'
  CURVE.YGF                := 'NONE'
  CURVE.TORQUE             := 'NONE'
  OUTLINE.SHAPE            := 'NONE'
  SHAPE.CENTER             := ( 0.0, 0.0 )
  ANGULAR.UNITS            := 'RADIANS'
  FLEXIBLE                 := 'FALSE'
  SUPERELEMENT             := 'FALSE'
UP
CREATE BODY
  NAME                     := 'ZOOL'
  CENTER.OF.GRAVITY        := ( 0.0, 0.0 )
  PHI                      := '0.0'
  FIXED.TO.GROUND          := 'FALSE'
  MASS                     := '0.001'
  INERTIA                  := '0.0001'
  XG.FORCE                 := '0.0'
  YG.FORCE                 := '0.0'
```

```
  TORQUE.CONSTANT          := '0.0'
  CURVE.XGF                := 'NONE'
  CURVE.YGF                := 'NONE'
  CURVE.TORQUE             := 'NONE'
  OUTLINE.SHAPE            := 'NONE'
  SHAPE.CENTER             := ( 0.0, 0.0 )
  ANGULAR.UNITS            := 'RADIANS'
  FLEXIBLE                 := 'FALSE'
  SUPERELEMENT             := 'FALSE'
UP
CREATE INITIAL.CONDITION
  NAME                     := 'IN1'
  BODY.1.NAME              := 'LIJF'
  BODY.2.NAME              := 'NONE'
  TYPE.INITIAL.COND        := 'PHI'
  INITIAL.VALUE            := '0.0'
  TIME.DERIVATIVE          := '0.0'
  P.ON.BODY.1              := ( 0.0, 0.0 )
  P.ON.BODY.2              := ( 0.0, 0.0 )
  EXTRA.COORD              := '0'
  ANGULAR.UNITS            := 'RADIANS'
UP
CREATE POINT.OF.INTEREST
  NAME                     := 'MARKER'
  BODY.NAME                := 'LIJF'
  P.ON.BODY                := ( 0.0, 0.8 )
  NODE                     := '0'
UP
CREATE DRIVER
  NAME                     := 'DR1'
  BODY.1.NAME              := 'VOET'
  BODY.2.NAME              := 'ZOOL'
  TYPE.DRIVER              := 'X.DIFF'
  DRIVING.FUNCTION         := 'POLYNOMIAL'
  FUNCTION.PARAMETERS      := ( 0, -0.1382, 0, 0 )
  P.ON.BODY.1              := ( 0.0, 0.0 )
  P.ON.BODY.2              := ( 0.0, 0.0 )
  Q.ON.BODY.1              := ( 1.0, 0.0 )
  Q.ON.BODY.2              := ( 1.0, 0.0 )
  CURVE.DRIVER             := 'NONE'
  JOINT.NAME               := 'NONE'
  ANGULAR.UNITS            := 'RADIANS'
UP
CREATE DRIVER
  NAME                     := 'DR2'
  BODY.1.NAME              := 'ZOOL'
  BODY.2.NAME              := 'NONE'
  TYPE.DRIVER              := 'X'
  DRIVING.FUNCTION         := 'HARMONIC'
  FUNCTION.PARAMETERS      := ( 0, -0.00637, 21.71, 0 )
  P.ON.BODY.1              := ( 0.0, 0.0 )
  P.ON.BODY.2              := ( 0.0, 0.0 )
  Q.ON.BODY.1              := ( 1.0, 0.0 )
  Q.ON.BODY.2              := ( 1.0, 0.0 )
  CURVE.DRIVER             := 'NONE'
  JOINT.NAME               := 'NONE'
```

```
  ANGULAR.UNITS                := 'RADIANS'
UP
```

DSL.VB2

```
ANALYSIS
CREATE SYSTEM.DATA
  UNITS                      := 'SI'
  ANALYSIS.TYPE                := 'DYNAMIC'
  STARTING.TIME                := '0.0'
  ENDING.TIME                  := '.2'
  PRINT.INTERVAL               := '0.005'
  GRAVITY.SEA.LEVEL              := '9.80665'
  X.GRAVITY                  := '0.0'
  Y.GRAVITY                  := '-1.0'
  SCALE.GRAVITY.COEF             := '1.0'
  MATRIX.OPERATIONS              := 'SPARSE'
  REDUNDANCY.CHECK               := 'TRUE'
  LU.TOL                     := '1.0D-12'
  ASSEMBLY.TOL                 := '1.0D-3'
  BYPASS.ASSEMBLY              := 'FALSE'
  OUTPUT.FILE                := 'BINARY'
  REFERENCE.FRAME                := 'LOCAL'
  DEBUG.FLAG                 := 'TRUE'
UP
CREATE DYNAMIC.DATA
  REACTION.FORCES              := 'FALSE'
  FORCE.COORDINATES              := 'GLOBAL'
  PRINT.METHOD               := 'INTERPOLATED'
  MAX.INT.STEP               := '0.001'
  SOLUTION.TOL                 := '0.002'
  INTEGRATION.TOL              := '0.0001'
UP
UP
FORCE
CREATE RSDA
  NAME                       := 'VEER1'
  JOINT.NAME                 := 'ENKEL'
  ORIENTATION.ANGLE              := '0.0'
  SPRING.CONSTANT              := '500'
  DAMPING.COEFFICIENT            := '100'
  ACTUATOR.TORQUE              := '0.0'
  CURVE.SPRING               := 'NONE'
  CURVE.DAMPER                 := 'NONE'
  CURVE.ACTUATOR               := 'NONE'
  ANGULAR.UNITS                := 'RADIANS'
UP
CREATE RSDA
  NAME                       := 'VEER2'
  JOINT.NAME                 := 'NEK'
  ORIENTATION.ANGLE              := '0.0'
  SPRING.CONSTANT              := '150'
  DAMPING.COEFFICIENT            := '25'
  ACTUATOR.TORQUE              := '0.0'
  CURVE.SPRING               := 'NONE'
```

```
  CURVE.DAMPER              := 'NONE'
  CURVE.ACTUATOR            := 'NONE'
  ANGULAR.UNITS             := 'RADIANS'
UP
UP
JOINTS
CREATE REVOLUTEJOINT
  NAME                      := 'ENKEL'
  BODY.1.NAME               := 'LIJF'
  BODY.2.NAME               := 'VOET'
  P.ON.BODY.1               := ( 0, -0.445 )
  P.ON.BODY.2               := ( 0.0, 0.0 )
  Q.ON.BODY.1               := ( 1, -0.445 )
  Q.ON.BODY.2               := ( 1.0, 0.0 )
  NODE.1                    := '0'
  NODE.2                    := '0'
UP
CREATE REVOLUTEJOINT
  NAME                      := 'NEK'
  BODY.1.NAME               := 'HOOFD'
  BODY.2.NAME               := 'LIJF'
  P.ON.BODY.1               := ( 0, -0.395 )
  P.ON.BODY.2               := ( 0, 0.445 )
  Q.ON.BODY.1               := ( 1, -0.395 )
  Q.ON.BODY.2               := ( 1, 0.445 )
  NODE.1                    := '0'
  NODE.2                    := '0'
UP
CREATE TRANSLATIONALJOINT
  NAME                      := 'TR1'
  BODY.1.NAME               := 'ZOOL'
  BODY.2.NAME               := 'GROND'
  P.ON.BODY.1               := ( 0.0, 0.0 )
  P.ON.BODY.2               := ( 0.0, 0.0 )
  Q.ON.BODY.1               := ( 1.0, 0.0 )
  Q.ON.BODY.2               := ( 1.0, 0.0 )
  NODE.1                    := '0'
  NODE.2                    := '0'
UP
CREATE TRANSLATIONALJOINT
  NAME                      := 'TR2'
  BODY.1.NAME               := 'VOET'
  BODY.2.NAME               := 'ZOOL'
  P.ON.BODY.1               := ( 0.0, 0.0 )
  P.ON.BODY.2               := ( 0.0, 0.0 )
  Q.ON.BODY.1               := ( 1.0, 0.0 )
  Q.ON.BODY.2               := ( 1.0, 0.0 )
  NODE.1                    := '0'
  NODE.2                    := '0'
UP
UP
CREATE BODY
  NAME                      := 'GROND'
  CENTER.OF.GRAVITY         := ( 0.0, 0.0 )
  PHI                       := '0.0'
  FIXED.TO.GROUND           := 'TRUE'
```

```
   MASS                    := '1.0'
   INERTIA                 := '1.0'
   XG.FORCE                := '0.0'
   YG.FORCE                := '0.0'
   TORQUE.CONSTANT         := '0.0'
   CURVE.XGF               := 'NONE'
   CURVE.YGF               := 'NONE'
   CURVE.TORQUE            := 'NONE'
   OUTLINE.SHAPE           := 'NONE'
   SHAPE.CENTER            := ( 0.0, 0.0 )
   ANGULAR.UNITS           := 'RADIANS'
   FLEXIBLE                := 'FALSE'
   SUPERELEMENT            := 'FALSE'
UP
CREATE BODY
   NAME                    := 'LIJF'
   CENTER.OF.GRAVITY       := ( 0.0, 0.0 )
   PHI                     := '0.0'
   FIXED.TO.GROUND         := 'FALSE'
   MASS                    := '31.6'
   INERTIA                 := '2.0859'
   XG.FORCE                := '0.0'
   YG.FORCE                := '0.0'
   TORQUE.CONSTANT         := '0.0'
   CURVE.XGF               := 'NONE'
   CURVE.YGF               := 'NONE'
   CURVE.TORQUE            := 'NONE'
   OUTLINE.SHAPE           := 'NONE'
   SHAPE.CENTER            := ( 0.0, 0.0 )
   ANGULAR.UNITS           := 'RADIANS'
   FLEXIBLE                := 'FALSE'
   SUPERELEMENT            := 'FALSE'
UP
CREATE BODY
   NAME                    := 'VOET'
   CENTER.OF.GRAVITY       := ( 0.0, 0.0 )
   PHI                     := '0.0'
   FIXED.TO.GROUND         := 'FALSE'
   MASS                    := '0.001'
   INERTIA                 := '0.0001'
   XG.FORCE                := '0.0'
   YG.FORCE                := '0.0'
   TORQUE.CONSTANT         := '0.0'
   CURVE.XGF               := 'NONE'
   CURVE.YGF               := 'NONE'
   CURVE.TORQUE            := 'NONE'
   OUTLINE.SHAPE           := 'NONE'
   SHAPE.CENTER            := ( 0.0, 0.0 )
   ANGULAR.UNITS           := 'RADIANS'
   FLEXIBLE                := 'FALSE'
   SUPERELEMENT            := 'FALSE'
UP
CREATE BODY
   NAME                    := 'ZOOL'
   CENTER.OF.GRAVITY       := ( 0.0, 0.0 )
   PHI                     := '0.0'
```

```
  FIXED.TO.GROUND          := 'FALSE'
  MASS                   := '0.001'
  INERTIA                := '0.0001'
  XG.FORCE               := '0.0'
  YG.FORCE               := '0.0'
  TORQUE.CONSTANT          := '0.0'
  CURVE.XGF              := 'NONE'
  CURVE.YGF              := 'NONE'
  CURVE.TORQUE             := 'NONE'
  OUTLINE.SHAPE            := 'NONE'
  SHAPE.CENTER             := ( 0.0, 0.0 )
  ANGULAR.UNITS            := 'RADIANS'
  FLEXIBLE               := 'FALSE'
  SUPERELEMENT             := 'FALSE'
UP
CREATE BODY
  NAME                   := 'HOOFD'
  CENTER.OF.GRAVITY        := ( 0.0, 0.0 )
  PHI                  := '0.0'
  FIXED.TO.GROUND          := 'FALSE'
  MASS                   := '31.4'
  INERTIA                := '1.6331'
  XG.FORCE               := '0.0'
  YG.FORCE               := '0.0'
  TORQUE.CONSTANT          := '0.0'
  CURVE.XGF              := 'NONE'
  CURVE.YGF              := 'NONE'
  CURVE.TORQUE             := 'NONE'
  OUTLINE.SHAPE            := 'NONE'
  SHAPE.CENTER             := ( 0.0, 0.0 )
  ANGULAR.UNITS            := 'RADIANS'
  FLEXIBLE               := 'FALSE'
  SUPERELEMENT             := 'FALSE'
UP
CREATE INITIAL.CONDITION
  NAME                   := 'IN1'
  BODY.1.NAME              := 'LIJF'
  BODY.2.NAME              := 'NONE'
  TYPE.INITIAL.COND        := 'PHI'
  INITIAL.VALUE            := '0.0'
  TIME.DERIVATIVE          := '0.0'
  P.ON.BODY.1            := ( 0.0, 0.0 )
  P.ON.BODY.2            := ( 0.0, 0.0 )
  EXTRA.COORD              := '0'
  ANGULAR.UNITS            := 'RADIANS'
UP
CREATE INITIAL.CONDITION
  NAME                   := 'IN2'
  BODY.1.NAME              := 'HOOFD'
  BODY.2.NAME              := 'NONE'
  TYPE.INITIAL.COND        := 'PHI'
  INITIAL.VALUE            := '0.0'
  TIME.DERIVATIVE          := '0.0'
  P.ON.BODY.1            := ( 0.0, 0.0 )
  P.ON.BODY.2            := ( 0.0, 0.0 )
  EXTRA.COORD              := '0'
```

```
  ANGULAR.UNITS                 := 'RADIANS'
UP
CREATE POINT.OF.INTEREST
  NAME                        := 'MARKER'
  BODY.NAME                     := 'HOOFD'
  P.ON.BODY                    := ( 0, 0.395 )
  NODE                        := '0'
UP
CREATE POINT.OF.INTEREST
  NAME                        := 'MARKNEK'
  BODY.NAME                     := 'LIJF'
  P.ON.BODY                    := ( 0, 0.445 )
  NODE                        := '0'
UP
CREATE POINT.OF.INTEREST
  NAME                        := 'MARKENK'
  BODY.NAME                     := 'LIJF'
  P.ON.BODY                    := ( 0, -0.444 )
  NODE                        := '0'
UP
CREATE DRIVER
  NAME                        := 'DR1'
  BODY.1.NAME                   := 'VOET'
  BODY.2.NAME                   := 'ZOOL'
  TYPE.DRIVER                  := 'X.DIFF'
  DRIVING.FUNCTION                := 'POLYNOMIAL'
  FUNCTION.PARAMETERS               := ( 0, -0.1382, 0, 0 )
  P.ON.BODY.1                  := ( 0.0, 0.0 )
  P.ON.BODY.2                  := ( 0.0, 0.0 )
  Q.ON.BODY.1                  := ( 1.0, 0.0 )
  Q.ON.BODY.2                  := ( 1.0, 0.0 )
  CURVE.DRIVER                   := 'NONE'
  JOINT.NAME                   := 'NONE'
  ANGULAR.UNITS                  := 'RADIANS'
UP
CREATE DRIVER
  NAME                        := 'DR2'
  BODY.1.NAME                   := 'ZOOL'
  BODY.2.NAME                   := 'NONE'
  TYPE.DRIVER                  := 'X'
  DRIVING.FUNCTION                := 'HARMONIC'
  FUNCTION.PARAMETERS               := ( 0, -0.00637, 21.71, 0 )
  P.ON.BODY.1                  := ( 0.0, 0.0 )
  P.ON.BODY.2                  := ( 0.0, 0.0 )
  Q.ON.BODY.1                  := ( 1.0, 0.0 )
  Q.ON.BODY.2                  := ( 1.0, 0.0 )
  CURVE.DRIVER                   := 'NONE'
  JOINT.NAME                   := 'NONE'
  ANGULAR.UNITS                  := 'RADIANS'
UP
```

TRISL.VB2

```
ANALYSIS
CREATE SYSTEM.DATA
```

```
  UNITS                    := 'SI'
  ANALYSIS.TYPE              := 'DYNAMIC'
  STARTING.TIME              := '0.0'
  ENDING.TIME                := '.2'
  PRINT.INTERVAL             := '0.005'
  GRAVITY.SEA.LEVEL          := '9.80665'
  X.GRAVITY                  := '0.0'
  Y.GRAVITY                  := '-1.0'
  SCALE.GRAVITY.COEF         := '1.0'
  MATRIX.OPERATIONS          := 'SPARSE'
  REDUNDANCY.CHECK           := 'TRUE'
  LU.TOL                   := '1.0D-12'
  ASSEMBLY.TOL               := '1.0D-3'
  BYPASS.ASSEMBLY            := 'FALSE'
  OUTPUT.FILE              := 'BINARY'
  REFERENCE.FRAME            := 'LOCAL'
  DEBUG.FLAG               := 'TRUE'
UP
CREATE DYNAMIC.DATA
  REACTION.FORCES            := 'FALSE'
  FORCE.COORDINATES          := 'GLOBAL'
  PRINT.METHOD             := 'INTERPOLATED'
  MAX.INT.STEP             := '0.001'
  SOLUTION.TOL               := '0.002'
  INTEGRATION.TOL            := '0.0001'
UP
UP
FORCE
CREATE RSDA
  NAME                     := 'VEER1'
  JOINT.NAME               := 'ENKEL'
  ORIENTATION.ANGLE          := '0.0'
  SPRING.CONSTANT            := '0'
  DAMPING.COEFFICIENT        := '0'
  ACTUATOR.TORQUE            := '0.0'
  CURVE.SPRING             := 'NONE'
  CURVE.DAMPER               := 'NONE'
  CURVE.ACTUATOR             := 'NONE'
  ANGULAR.UNITS              := 'RADIANS'
UP
CREATE RSDA
  NAME                     := 'VEER2'
  JOINT.NAME               := 'NEK'
  ORIENTATION.ANGLE          := '0.0'
  SPRING.CONSTANT            := '0'
  DAMPING.COEFFICIENT        := '0'
  ACTUATOR.TORQUE            := '0.0'
  CURVE.SPRING             := 'NONE'
  CURVE.DAMPER               := 'NONE'
  CURVE.ACTUATOR             := 'NONE'
  ANGULAR.UNITS              := 'RADIANS'
UP
CREATE RSDA
  NAME                     := 'VEER3'
  JOINT.NAME               := 'HEUP'
  ORIENTATION.ANGLE          := '0.0'
```

```
  SPRING.CONSTANT              := '0.0'
  DAMPING.COEFFICIENT          := '0.0'
  ACTUATOR.TORQUE              := '0.0'
  CURVE.SPRING                 := 'NONE'
  CURVE.DAMPER                 := 'NONE'
  CURVE.ACTUATOR               := 'NONE'
  ANGULAR.UNITS                := 'RADIANS'
UP
UP
JOINTS
CREATE REVOLUTE.JOINT
  NAME                         := 'ENKEL'
  BODY.1.NAME                  := 'BEEN'
  BODY.2.NAME                  := 'VOET'
  P.ON.BODY.1                  := ( 0, -.45 )
  P.ON.BODY.2                  := ( 0.0, 0.0 )
  Q.ON.BODY.1                  := ( 1, -.45 )
  Q.ON.BODY.2                  := ( 1.0, 0.0 )
  NODE.1                       := '0'
  NODE.2                       := '0'
UP
CREATE REVOLUTE.JOINT
  NAME                         := 'NEK'
  BODY.1.NAME                  := 'HOOFD'
  BODY.2.NAME                  := 'LIJF'
  P.ON.BODY.1                  := ( 0, -0.15 )
  P.ON.BODY.2                  := ( 0, 0.25 )
  Q.ON.BODY.1                  := ( 1, -0.15 )
  Q.ON.BODY.2                  := ( 1, 0.25 )
  NODE.1                       := '0'
  NODE.2                       := '0'
UP
CREATE REVOLUTE.JOINT
  NAME                         := 'HEUP'
  BODY.1.NAME                  := 'BEEN'
  BODY.2.NAME                  := 'LIJF'
  P.ON.BODY.1                  := ( 0, 0.45 )
  P.ON.BODY.2                  := ( 0, -0.25 )
  Q.ON.BODY.1                  := ( 1, 0.45 )
  Q.ON.BODY.2                  := ( 1, -0.25 )
  NODE.1                       := '0'
  NODE.2                       := '0'
UP
CREATE TRANSLATIONAL.JOINT
  NAME                         := 'TR1'
  BODY.1.NAME                  := 'ZOOL'
  BODY.2.NAME                  := 'GROND'
  P.ON.BODY.1                  := ( 0.0, 0.0 )
  P.ON.BODY.2                  := ( 0.0, 0.0 )
  Q.ON.BODY.1                  := ( 1.0, 0.0 )
  Q.ON.BODY.2                  := ( 1.0, 0.0 )
  NODE.1                       := '0'
  NODE.2                       := '0'
UP
CREATE TRANSLATIONAL.JOINT
  NAME                         := 'TR2'
```

```
  BODY.1.NAME                := 'VOET'
  BODY.2.NAME                := 'ZOOL'
  P.ON.BODY.1               := ( 0.0, 0.0 )
  P.ON.BODY.2               := ( 0.0, 0.0 )
  Q.ON.BODY.1               := ( 1.0, 0.0 )
  Q.ON.BODY.2               := ( 1.0, 0.0 )
  NODE.1                    := '0'
  NODE.2                    := '0'
UP
UP
CREATE BODY
  NAME                      := 'GROND'
  CENTER.OF.GRAVITY              := ( 0.0, 0.0 )
  PHI                       := '0.0'
  FIXED.TO.GROUND                := 'TRUE'
  MASS                      := '1.0'
  INERTIA                   := '1.0'
  XG.FORCE                  := '0.0'
  YG.FORCE                  := '0.0'
  TORQUE.CONSTANT                := '0.0'
  CURVE.XGF                 := 'NONE'
  CURVE.YGF                 := 'NONE'
  CURVE.TORQUE              := 'NONE'
  OUTLINE.SHAPE             := 'NONE'
  SHAPE.CENTER              := ( 0.0, 0.0 )
  ANGULAR.UNITS             := 'RADIANS'
  FLEXIBLE                  := 'FALSE'
  SUPERELEMENT              := 'FALSE'
UP
CREATE BODY
  NAME                      := 'LIJF'
  CENTER.OF.GRAVITY              := ( 0.0, 0.0 )
  PHI                       := '0.0'
  FIXED.TO.GROUND                := 'FALSE'
  MASS                      := '27.4'
  INERTIA                   := '1.8495'
  XG.FORCE                  := '0.0'
  YG.FORCE                  := '0.0'
  TORQUE.CONSTANT                := '0.0'
  CURVE.XGF                 := 'NONE'
  CURVE.YGF                 := 'NONE'
  CURVE.TORQUE              := 'NONE'
  OUTLINE.SHAPE             := 'NONE'
  SHAPE.CENTER              := ( 0.0, 0.0 )
  ANGULAR.UNITS             := 'RADIANS'
  FLEXIBLE                  := 'FALSE'
  SUPERELEMENT              := 'FALSE'
UP
CREATE BODY
  NAME                      := 'VOET'
  CENTER.OF.GRAVITY              := ( 0.0, 0.0 )
  PHI                       := '0.0'
  FIXED.TO.GROUND                := 'FALSE'
  MASS                      := '0.001'
  INERTIA                   := '0.0001'
  XG.FORCE                  := '0.0'
```

```
  YG.FORCE                  := '0.0'
  TORQUE.CONSTANT             := '0.0'
  CURVE.XGF                 := 'NONE'
  CURVE.YGF                 := 'NONE'
  CURVE.TORQUE              := 'NONE'
  OUTLINE.SHAPE             := 'NONE'
  SHAPE.CENTER              := ( 0.0, 0.0 )
  ANGULAR.UNITS             := 'RADIANS'
  FLEXIBLE                  := 'FALSE'
  SUPERELEMENT              := 'FALSE'
UP
CREATE BODY
  NAME                      := 'ZOOL'
  CENTER.OF.GRAVITY           := ( 0.0, 0.0 )
  PHI                     := '0.0'
  FIXED.TO.GROUND           := 'FALSE'
  MASS                    := '0.001'
  INERTIA                 := '0.0001'
  XG.FORCE                  := '0.0'
  YG.FORCE                  := '0.0'
  TORQUE.CONSTANT             := '0.0'
  CURVE.XGF                 := 'NONE'
  CURVE.YGF                 := 'NONE'
  CURVE.TORQUE              := 'NONE'
  OUTLINE.SHAPE             := 'NONE'
  SHAPE.CENTER              := ( 0.0, 0.0 )
  ANGULAR.UNITS             := 'RADIANS'
  FLEXIBLE                  := 'FALSE'
  SUPERELEMENT              := 'FALSE'
UP
CREATE BODY
  NAME                      := 'HOOFD'
  CENTER.OF.GRAVITY           := ( 0.0, 0.0 )
  PHI                     := '0.0'
  FIXED.TO.GROUND           := 'FALSE'
  MASS                    := '4'
  INERTIA                 := '0.03'
  XG.FORCE                  := '0.0'
  YG.FORCE                  := '0.0'
  TORQUE.CONSTANT             := '0.0'
  CURVE.XGF                 := 'NONE'
  CURVE.YGF                 := 'NONE'
  CURVE.TORQUE              := 'NONE'
  OUTLINE.SHAPE             := 'NONE'
  SHAPE.CENTER              := ( 0.0, 0.0 )
  ANGULAR.UNITS             := 'RADIANS'
  FLEXIBLE                  := 'FALSE'
  SUPERELEMENT              := 'FALSE'
UP
CREATE BODY
  NAME                      := 'BEEN'
  CENTER.OF.GRAVITY           := ( 0.0, 0.0 )
  PHI                     := '0.0'
  FIXED.TO.GROUND           := 'FALSE'
  MASS                    := '31.6'
  INERTIA                 := '2.0859'
```

```
  XG.FORCE                    := '0.0'
  YG.FORCE                    := '0.0'
  TORQUE.CONSTANT                := '0.0'
  CURVE.XGF                   := 'NONE'
  CURVE.YGF                   := 'NONE'
  CURVE.TORQUE                   := 'NONE'
  OUTLINE.SHAPE                  := 'NONE'
  SHAPE.CENTER                   := ( 0.0, 0.0 )
  ANGULAR.UNITS                  := 'RADIANS'
  FLEXIBLE                    := 'FALSE'
  SUPERELEMENT                   := 'FALSE'
UP
CREATE INITIAL.CONDITION
  NAME                        := 'IN1'
  BODY.1.NAME                    := 'LIJF'
  BODY.2.NAME                    := 'NONE'
  TYPE.INITIAL.COND              := 'PHI'
  INITIAL.VALUE                  := '0.0'
  TIME.DERIVATIVE                := '0.0'
  P.ON.BODY.1                 := ( 0.0, 0.0 )
  P.ON.BODY.2                 := ( 0.0, 0.0 )
  EXTRA.COORD                    := '0'
  ANGULAR.UNITS                  := 'RADIANS'
UP
CREATE INITIAL.CONDITION
  NAME                        := 'IN2'
  BODY.1.NAME                    := 'HOOFD'
  BODY.2.NAME                    := 'NONE'
  TYPE.INITIAL.COND              := 'PHI'
  INITIAL.VALUE                  := '0.0'
  TIME.DERIVATIVE                := '0.0'
  P.ON.BODY.1                 := ( 0.0, 0.0 )
  P.ON.BODY.2                 := ( 0.0, 0.0 )
  EXTRA.COORD                    := '0'
  ANGULAR.UNITS                  := 'RADIANS'
UP
CREATE INITIAL.CONDITION
  NAME                        := 'IN3'
  BODY.1.NAME                    := 'BEEN'
  BODY.2.NAME                    := 'NONE'
  TYPE.INITIAL.COND              := 'PHI'
  INITIAL.VALUE                  := '0.0'
  TIME.DERIVATIVE                := '0.0'
  P.ON.BODY.1                 := ( 0.0, 0.0 )
  P.ON.BODY.2                 := ( 0.0, 0.0 )
  EXTRA.COORD                    := '0'
  ANGULAR.UNITS                  := 'RADIANS'
UP
CREATE POINT.OF.INTEREST
  NAME                        := 'MARKER'
  BODY.NAME                      := 'HOOFD'
  P.ON.BODY                   := ( 0, 0.15 )
  NODE                        := '0'
UP
CREATE POINT.OF.INTEREST
  NAME                        := 'MARKNEK'
```

```
  BODY.NAME                    := 'LIJF'
  P.ON.BODY                    := ( 0.0, 0.25 )
  NODE                         := '0'
UP
CREATE POINT.OF.INTEREST
  NAME                         := 'MARKENK'
  BODY.NAME                    := 'LIJF'
  P.ON.BODY                    := ( 0, -0.444 )
  NODE                         := '0'
UP
CREATE POINT.OF.INTEREST
  NAME                         := 'MARKHEUP'
  BODY.NAME                    := 'BEEN'
  P.ON.BODY                    := ( 0, 0.45 )
  NODE                         := '0'
UP
CREATE DRIVER
  NAME                         := 'DR1'
  BODY.1.NAME                  := 'VOET'
  BODY.2.NAME                  := 'ZOOL'
  TYPE.DRIVER                  := 'X.DIFF'
  DRIVING.FUNCTION             := 'POLYNOMIAL'
  FUNCTION.PARAMETERS          := ( 0, -0.1382, 0, 0 )
  P.ON.BODY.1                  := ( 0.0, 0.0 )
  P.ON.BODY.2                  := ( 0.0, 0.0 )
  Q.ON.BODY.1                  := ( 1.0, 0.0 )
  Q.ON.BODY.2                  := ( 1.0, 0.0 )
  CURVE.DRIVER                 := 'NONE'
  JOINT.NAME                   := 'NONE'
  ANGULAR.UNITS                := 'RADIANS'
UP
CREATE DRIVER
  NAME                         := 'DR2'
  BODY.1.NAME                  := 'ZOOL'
  BODY.2.NAME                  := 'NONE'
  TYPE.DRIVER                  := 'X'
  DRIVING.FUNCTION             := 'HARMONIC'
  FUNCTION.PARAMETERS          := ( 0, -0.00637, 21.71, 0 )
  P.ON.BODY.1                  := ( 0.0, 0.0 )
  P.ON.BODY.2                  := ( 0.0, 0.0 )
  Q.ON.BODY.1                  := ( 1.0, 0.0 )
  Q.ON.BODY.2                  := ( 1.0, 0.0 )
  CURVE.DRIVER                 := 'NONE'
  JOINT.NAME                   := 'NONE'
  ANGULAR.UNITS                := 'RADIANS'
UP
```

BIJLAGE 6

PC-Matlab programma's en routines voor het oplossen van 1e-orde differentiaalvergelijkingen

NLSYST.M

```
% Bepalen van de coëfficienten voor de toestandsbeschrijving van een niet lineaire bewegings-
% vergelijking in de vorm: A*x" +B*x' +C*x + D*x*x^2 = F
% Waarin: A = [a11, a12; a21, a22] ; B = [b11, b12; b21, b22] ; C = [c11, c12; c21, c22] en
% D = [d1 ; d2]

% Ingeven van de coëfficientenmatrices

A=input('massamatrix =');
B=input('dempingsmatrix =');
C=input('stijfheidsmatrix =');
D=input('niet-lineaire-coefficienten-vector =');

% Berekenen van de coëfficienten

d=det(A);
g1=-A(1,2)/d;
g2=A(2,2)/d;
g3=A(1,2)*D(2)/A(2,2)/d;
g4=-A(2,1)*g3;
g5=-D(1)/A(2,2)/A(2,2)/d;
g6=-D(1)*(1-A(2,1)/A(2,2))*(1-A(2,1)/A(2,2))/d;
g7=-2*D(1)*(1-A(2,1)/A(2,2))/A(2,2)/d;
g8=-A(2,1)*g5;
g9=-A(2,1)*g6;
g10=-A(2,1)*g7;
g11=(A(1,2)*C(2,2)-A(2,2)*C(1,2))/A(2,2)/d;
g12=-A(2,1)*g11-(A(2,2)*C(1,1)-A(1,2)*C(2,1))/d;
g13=(A(1,2)*B(2,2)-A(2,2)*B(1,2))/A(2,2)/d;
g14=-A(2,1)*g13-A(2,2)*B(1,1)/d;
g15=-D(2)/A(2,2);
g16=-A(2,1)*g15;
g17=-C(2,2)/A(2,2);
g18=-A(2,1)*g17-C(2,1);
g19=-B(2,2)/A(2,2);
g20=-A(2,1)*g19;

% globaal definiëren van de toestandscoëfficienten

global g1 g2 g3 g4 g5 g6 g7 g8 g9 g10 g11 g12 g13 g14 g15 g16 g17 g18 g19 g20

end
```

NLTOEST.M

```
function yprime = nltoest(t,y)

% Invoeren van de voorgeschreven rechterzijde van de vergelijkingen i.d.v.v de krachtvector F

F=[126.024*sin(21.71*t) ; 37.209*sin(21.71*t)];

% Invoeren van de toestandsbeschrijving m.b.v. de coëfficienten uit NLSYST.M

yprime = [y(4) ;

          y(3) ;

          g1*F(2)+g2*F(1)+(g3+g6)*y(1)*y(3)*y(3)+(g4+g9)*y(2)*y(3)*y(3)+...
          +g5*y(1)*y(4)*y(4)+g7*y(1)*y(3)*y(4)+g8*y(2)*y(4)*y(4)+g10*y(2)*y(3)*y(4)+...
          g11*y(1)+g12*y(2)+g13*y(4)+g14*y(3) ;

          F(2)+g15*y(1)*y(3)*y(3)+g16*y(2)*y(3)*y(3)+g17*y(1)+g18*y(2)+g19*y(4)+g20*y(3) ] ;

end
```

ODE45PA2.M

```
% Programma met toestemming overgenomen van Pascal Etman

function  yout = ode45pa2(FunFcn,time, y0, tol, trace)

%ODE45pa2 Aangepaste versie van ode45.m Tevens worden meer punten bepaald.
%       In PAR zitten de parameters van FunFcn.
%       Integrate a system of ordinary differential equations using
%       4th and 5th order Runge-Kutta formulas.  See also ODE23 and
%       ODEDEMO.M.
%       [T,Y] = ODE45('yprime', time, Y0) integrates the system
%       of ordinary differential equations described by the M-file
%       YPRIME.M over the interval T0 to Tfinal and using initial
%       conditions Y0. T0 = eerste element van TIME, tfinal is laatste
%       element in TIME.
%       [T, Y] = ODE45(F, time, Y0, TOL, 1) uses tolerance TOL
%       and displays status while the integration proceeds.
%
% INPUT:
% F     - String containing name of user-supplied problem description.
%         Call: yprime = fun(t,y) where F = 'fun'.
%         t      - Time (scalar).
%         y      - Solution column-vector.
%         yprime - Returned derivative column-vector; yprime(i) = dy(i)/dt.
% time    Tijdstippen waarop uitkomst moet worden berekend. Including:
%         - Initial value of t.
%         - Final value of t.
% y0    - Initial value column-vector for time T0.
% tol   - The desired accuracy. (Default: tol = 1.e-6).
% trace - If nonzero, each step is printed. (Default: trace = 0).
%
```

```
% OUTPUT:
% Y  - Returned solution, one solution column-vector per tout-value.
%
% The result can be displayed by: plot(tout, yout).

%   C.B. Moler, 3-25-87.
%   Copyright (c) 1987 by the MathWorks, Inc.
%   All rights reserved.

% The Fehlberg coefficients:

alpha = [1/4  3/8  12/13  1  1/2]';
beta  = [ [    1      0      0     0      0    0]/4
          [    3      9      0     0      0    0]/32
          [ 1932  -7200   7296     0      0    0]/2197
          [ 8341 -32832  29440  -845      0    0]/4104
          [-6080  41040 -28352  9295  -5643    0]/20520 ]';
gamma = [ [902880  0  3953664  3855735  -1371249  277020]/7618050
          [ -2090  0    22528    21970    -15048  -27360]/752400 ]';
pow = 1/5;

if nargin < 6, trace = 0; end
if nargin < 5, tol = 1.e-6; end

% Initialization

[r,c]=size(time);
if r < c
    time=time';    % Gebruikersvriendelijker gemaakt door Henrie de Haas
end

[r,c]=size(time);  % time =  kolom
t0=time(1);
tfinal=time(r);
t = t0;
hmax = (tfinal - t)/20;
hmin = (tfinal - t)/20000;
h = (tfinal-t)/100;
y = y0(:);
f = y*zeros(1,6);
tout = t;
yout = y.';
tau = tol * max(norm(y, 'inf'), 1);

if trace
    clc, t, h, y
end


% The main loop

k=1;
    while (k <=(r-1))
            if h>=(time(k+1)-time(k))
                h=(time(k+1)-time(k));
```

```
        % Compute the slopes

        f(:,1) = feval(FunFcn,t,y);
        for j = 1:5
            f(:,j+1) = feval(FunFcn, t+alpha(j)*h, y+h*f*beta(:,j));
        end

        % Estimate the error and the acceptable error

        delta = norm(h*f*gamma(:,2),'inf');
        tau = tol*max(norm(y,'inf'),1.0);

        % Update the solution only if the error is acceptable

        if delta <= tau
            t = time(k+1); %=t+h
            y = y + h*f*gamma(:,1);
            yout = [yout; y.'];
            k=k+1;
        end

        if trace
            home, t, h, y
        end

        % Update the step size

        if delta ~= 0.0
            h = min(hmax, 0.8*h*(tau/delta)^pow);
        end

    else  % h<time(k+1)-time(k)
            % From time(k) to time(k+1) integration with smaller steps than hmax
            % Computed points between h en hmax are not put in the ouput

        while ((t<time(k+1))&(h>=hmin))
            okay=0;
            hulp=h;
            if ((t+h)>=time(k+1))
                h=time(k+1)-t;
                okay=1;
            end

            % Compute the slopes

            f(:,1) = feval(FunFcn,t,y);
            for j = 1:5
                f(:,j+1) = feval(FunFcn, t+alpha(j)*h, y+h*f*beta(:,j));

            end

            % Estimate the error and the acceptable error

            delta = norm(h*f*gamma(:,2),'inf');
            tau = tol*max(norm(y,'inf'),1.0);
```

```
                % Update the solution only if the error is acceptable

                if delta <= tau
                    t = t + h;
                    y = y + h*f*gamma(:,1);
                    if okay==1
                        yout = [yout; y.'];
                    end
                end

                if trace
                    home, t, h, y
                end

                % Update the step size

                if delta ~= 0.0
                    if okay==0
                        h = min(hmax, 0.8*h*(tau/delta)^pow);
                    else
                        h=hulp;
                        h = min(hmax, 0.8*h*(tau/delta)^pow);
                    end
                end

            end     % while



            if (t < time(k+1))
                disp('SINGULARITY LIKELY.')
                t
                k=r;    % The main loop is cancelled
            end

            k=k+1;

        end     % if

    end         % while

end
```

BIJLAGE 7

PC-Matlab programma's voor parametervariatie

GEVH.M

```
% Gevoeligheidsbepaling van het dubbelslingermodel, berekend de responsie gegeven de veer en
% dempingsconstanten

% Ingeven van modelgegevens

m1=input('m1 = ');
m2=input('m2 = ');
h1=input('h1 = ');
h2=input('h2 = ');
l1=2*h1;
l2=2*h2;
g=input('g = ');
J1=m1*h1*h1/3;
J2=m2*h2*h2/3;

% Ingeven van beginschatting

pause(disp ('%x=[b1,b2,k1,k2]'))
x=input('x = ');

% Berekenen van coëfficientenmatrices

AA=[J1+m1*h1*h1+m2*l1*l1+m2*l1*h2 m2*l1*h2 ; J2+m2*h2*h2+m2*l1*h2 J2+m2*h2*h2]
BB=[0 , 0 ; 0 , 0]
CC=[-(m2*l1+m1*h1)*g , 0 ; -m2*h2*g , -m2*h2*g]
DD=[-m2*l1*h2 , m2*l1*h2]

% Ingeven van beginwaarden en globaal definiëren

t0 = 0
tfinal=0.2
y0=[0;0;0;0]
toll=1.e-8

global t0 tfinal y0 toll  AA BB CC DD
global g1 g2 g3 g4 g5 g6 g7 g8 g9 g10 g11 g12 g13 g14 g15 g16 g17 g18 g19 g20

% Reserveer ruimte om het resultaat in op te slaan

P=zeros(40,30);

% Bereken de responsie voor variërende dempingsconstanten en veerstijfheden

j = 1

xo=x;
    for i = 1:14
        [g1,g2,g3,g4,g5,g6,g7,g8,g9,g10,g11,g12,g13,g14,g15,g16,g17,g18,g19,g20]=coef(x);
```

```
        [T,Y]=ode45('nltoest',t0,tfinal,y0,toll);  % Responsiebepaling


        [n,m]=size(Y);          % Selectie en opslaan van resulaten
        k=1+(i-1)*m/2;
        P(1,k:i*m/2)=[Y(1:n,2) , (Y(1:n,1)/AA(2,2)-AA(2,1)/AA(2,2)*Y(1:n,2))];

% Informatie over het gevoeligheidsproces

        t=ones(1,2);
        T=T*t;
        plot(T,P(1:j*n,k:i*m/2))
        text(0.2,0.6,['b1 = 'num2str(x(1))],'sc')
        text(0.2,0.55,['b2 = 'num2str(x(2))],'sc')
        text(0.2,0.5,['k1 = 'num2str(x(3))],'sc')
        text(0.2,0.45,['k2 = 'num2str(x(4))],'sc')

        x(j)=x(j)+10
    end
    x=xo;

    P11=P(:,1:14);

% Bewaren van de resultaten in een ascii-file

    save GP1.m P11 /ascii
    save GT1.m T /ascii
end

j=2

xo=x;
    for i = 1:14
        [g1,g2,g3,g4,g5,g6,g7,g8,g9,g10,g11,g12,g13,g14,g15,g16,g17,..
        g18,g19,g20]=coef(x);

        [T,Y]=ode23('nltoest',t0,tfinal,y0);  % Responsiebepaling

        [n,m]=size(Y);          % Selectie en opslaan resultaten
        k=1+(i-1)*m/2;
        P(1,k:i*m/2)=[Y(1:n,2) , (Y(1:n,1)/AA(2,2)-AA(2,1)/AA(2,2)*Y(1:n,2))];

% Informatie over het gevoeligheidsproces

        t=ones(1,2);
        T=T*t;
        plot(T,P(1:j*n,k:i*m/2))
        text(0.2,0.6,['b1 = 'num2str(x(1))],'sc')
        text(0.2,0.55,['b2 = 'num2str(x(2))],'sc')
        text(0.2,0.5,['k1 = 'num2str(x(3))],'sc')
        text(0.2,0.45,['k2 = 'num2str(x(4))],'sc')

        x(j)=x(j)+10
    end
    x=xo
```

```
P21=P(:,1:14);

% Bewaren van de resultaten in een ascii-file

save gp2.m P21 /ascii
save GT2.m T /ascii

j=3

xo=x;
for i = 1:14
[g1,g2,g3,g4,g5,g6,g7,g8,g9,g10,g11,g12,g13,g14,g15,g16,g17,g18,g19,g20]=coef(x);

[T,Y]=ode23('nltoest',t0,tfinal,y0); % Responsiebepaling

[n,m]=size(Y);          % Selectie en opslaan resultaten
k=1+(i-1)*m/2;
P(1:n,k:i*m/2)=[Y(1:n,2) , (Y(1:n,1)/AA(2,2)-AA(2,1)/AA(2,2)*Y(1:n,2))];

% Informatie over het gevoeligheidsproces

t=ones(1,2);
T=T*t;
plot(T,P(1:n,k:i*m/2))
text(0.2,0.6,['b1 = 'num2str(x(1))],'sc')
text(0.2,0.55,['b2 = 'num2str(x(2))],'sc')
text(0.2,0.5,['k1 = 'num2str(x(3))],'sc')
text(0.2,0.45,['k2 = 'num2str(x(4))],'sc')

x(j)=x(j)+25
end
x=xo

P31=P(:,1:14);

% Bewaren van de resultaten in een ascii-file

save gp3.m P31 /ascii
save GT3.m T /ascii

j=4

xo=x;
for i = 1:14
[g1,g2,g3,g4,g5,g6,g7,g8,g9,g10,g11,g12,g13,g14,g15,g16,g17,g18,g19,g20]=coef(x);

[T,Y]=ode23('nltoest',t0,tfinal,y0); % Responsiebepaling

[n,m]=size(Y);          % Selectie en opslaan resultaten
k=1+(i-1)*m/2;
P(1:n,k:i*m/2)=[Y(1:n,2) , (Y(1:n,1)/AA(2,2)-AA(2,1)/AA(2,2)*Y(1:n,2))];

% Informatie over het gevoeligheidsproces

t=ones(1,2);
T=T*t;
```

```
        plot(T,P(1:n,k:i*m/2))
        text(0.2,0.6,['b1 = 'num2str(x(1))],'sc')
        text(0.2,0.55,['b2 = 'num2str(x(2))],'sc')
        text(0.2,0.5,['k1 = 'num2str(x(3))],'sc')
        text(0.2,0.45,['k2 = 'num2str(x(4))],'sc')

        x(j)=x(j)+25
    end
    x=xo

    P41=P(:,1:14);

% Bewaren van de resultaten in een ascii-file

    save gp4.m P41 /ascii
    save GT4.m T /ascii

end
```

COEF.M

```
function [g1,g2,g3,g4,g5,g6,g7,g8,g9,g10,g11,g12,g13,g14,g15,g16,g17,g18,g19,g20]=coef(x)

% Berekenen van de coëfficienten van de bewegingsvergelijkingen in de toestandsbeschrijving

d=det(AA);
g1=-AA(1,2)/d;
g2=AA(2,2)/d;
g3=AA(1,2)*DD(2)/AA(2,2)/d;
g4=-AA(2,1)*g3;
g5=-DD(1)/AA(2,2)/AA(2,2)/d;
g6=-DD(1)*(1-AA(2,1)/AA(2,2))*(1-AA(2,1)/AA(2,2))/d;
g7=-2*DD(1)*(1-AA(2,1)/AA(2,2))/AA(2,2)/d;
g8=-AA(2,1)*g5;
g9=-AA(2,1)*g6;
g10=-AA(2,1)*g7;
g11=(AA(1,2)*CC(2,2)+AA(1,2)*x(4)-AA(2,2)*CC(1,2)+AA(2,2)*x(4))/AA(2,2)/d;
g12=-AA(2,1)*g11-(AA(2,2)*CC(1,1)+AA(2,2)*x(3)-AA(1,2)*CC(2,1))/d;
g13=(AA(1,2)+AA(2,2))*x(2)/AA(2,2)/d;
g14=-AA(2,1)*g13-AA(2,2)*x(1)/d;
g15=-DD(2)/AA(2,2);
g16=-AA(2,1)*g15;
g17=-CC(2,2)/AA(2,2)-x(4)/AA(2,2);
g18=-AA(2,1)*g17-CC(2,1);
g19=-x(2)/AA(2,2);
g20=-AA(2,1)*g19;

end
```

GEVHJ.M

```
% gevoeligheidsbepaling van het dubbelslingermodel, berekend de responsie gegeven de massa's
% en hoogteligging van het massamiddelpunt als functie van de schatting x

% Ingeven van de modelparameters

b1=input('b1 = ');
b2=input('b2 = ');
k1=input('k1 = ');
k2=input('k2 = ');
g=input('g = ');
pause(disp ('% x = [m1, m2, h1, h2] '));
x=input('x = ');

m1=x(1);
m2=x(2);
h1=x(3);
h2=x(4);
l1=2*h1;
l2=2*h2;
J1=m1*h1*h1/3;
J2=m2*h2*h2/3;

% Berekenen van de coëfficientenmatrices A, B, C en D

AA=[J1+m1*h1*h1+m2*l1*l1+m2*l1*h2 m2*l1*h2 ; J2+m2*h2*h2+m2*l1*h2 J2+m2*h2*h2]
BB=[b1,-b2;0,b2]
CC=[k1-(m2*l1+m1*h1)*g , -k2 ; -m2*h2*g , k2-m2*h2*g]
DD=[-m2*l1*h2 , m2*l1*h2]

% Ingeven van de beginwaarden en het globaal definiëren

t0 = 0
tfinal=0.2
y0=[0;0;0;0]
toll=1.e-6
global t0 tfinal y0 toll b1 b2 k1 k2 g
global g1 g2 g3 g4 g5 g6 g7 g8 g9 g10 g11 g12 g13 g14 g15 g16 g17 g18 g19 g20

% Ruimte reserveren om het resultaat op te bergen

P=zeros(4*52,14*2);

% Responsiebepaling bij variërende m1, m2, h1 en h2

j=1;

xo=x;
    for i = 1:14
        [g1,g2,g3,g4,g5,g6,g7,g8,g9,g10,g11,g12,g13,g14,g15,g16,g17,g18,g19,g20]=coefj(x);

        [T1,Y]=ode45('nltoest',t0,tfinal,y0,toll);    % Responsiebepaling

        [n,m]=size(Y);          % Selectie en opslaan van de resultaten
        k=1+(i-1)*m/2;
```

```
        P1(1:n,k:i*m/2)=[Y(1:n,2) , (Y(1:n,1)/AA(2,2)-AA(2,1)/AA(2,2)*Y(1:n,2))];

% Informatie over het gevoeligheidsproces
        t=ones(1,2);
        T=T1*t;
        plot(T,P1(1:n,k:i*m/2))

        x(j)=x(j)+1.5
    end

    P11=P1(:,1:14);
    P12=P1(:,15:28);
x=xo;

% Bewaren van de resultaten in een ascii-file

save GP1.m P11 P12 /ascii
save GT1.m T1 /ascii


j=2;

xo=x;
    for i = 1:14
        [g1,g2,g3,g4,g5,g6,g7,g8,g9,g10,g11,g12,g13,g14,g15,g16,g17,g18,g19,g20]=coefj(x);;

        [T2,Y]=ode45('nltoest',t0,tfinal,y0,toll);  % Responsiebepaling

        [n,m]=size(Y);        % Selectie en opslaan van de resultaten
        k=1+(i-1)*m/2;
        P2(1:n,k:i*m/2)=[Y(1:n,2) , (Y(1:n,1)/AA(2,2)-AA(2,1)/AA(2,2)*Y(1:n,2))];

% Informatie over het gevoeligheidsproces
        t=ones(1,2);
        T=T2*t;
        plot(T,P2(1:n,k:i*m/2))

        x(j)=x(j)+1.5
    end

    P21=P2(:,1:14);
    P22=P2(:,15:28);
x=xo;

% Bewaren van de resultaten in een ascii-file

save GP2.m P21 P22 /ascii
save GT2.m T2 /ascii


j=3;

xo=x;
    for i = 1:14
        [g1,g2,g3,g4,g5,g6,g7,g8,g9,g10,g11,g12,g13,g14,g15,g16,g17,g18,g19,g20]=coefj(x);
```

```
        [T3,Y]=ode45('nltoest',t0,tfinal,y0,toll); % Responsiebepaling

        [n,m]=size(Y);          % Selectie en opslaan van de resultaten
        k=1+(i-1)*m/2;
        P3(1:n,k:i*m/2)=[Y(1:n,2) , (Y(1:n,1)/AA(2,2)-AA(2,1)/AA(2,2)*Y(1:n,2))];

% Informatie over het gevoeligheidsproces

        t=ones(1,2);
        T=T3*t;
        plot(T,P3(1:n,k:i*m/2))

        x(j)=x(j)+0.015
    end

    P31=P3(:,1:14);
    P32=P3(:,15:28);
x=xo;

% Bewaren van de resultaten in een ascii-file

save GP3.m P31 P32 /ascii
save GT3.m T3 /ascii


j=4;

xo=x;
    for i = 1:14
        [g1,g2,g3,g4,g5,g6,g7,g8,g9,g10,g11,g12,g13,g14,g15,g16,g17,g18,g19,g20]=coefj(x);

        [T4,Y]=ode45('nltoest',t0,tfinal,y0,toll);  % Responsiebepaling

        [n,m]=size(Y);          % Selectie en opslaan van resultaten
        k=1+(i-1)*m/2;
        P4(1:n,k:i*m/2)=[Y(1:n,2) , (Y(1:n,1)/AA(2,2)-AA(2,1)/AA(2,2)*Y(1:n,2))];

% Informatie over het gevoeligheidsproces

        t=ones(1,2);
        T=T4*t;
        plot(T,P4(1:n,k:i*m/2))

        x(j)=x(j)+0.015
    end

    P41=P4(:,1:14);
    P42=P4(:,15:28);
x=xo;

% Bewaren van de resultaten in een ascii-file

save GP4.m P41 P42 /ascii
save GT4.m T4 /ascii

end
```

COEFJ.M

```
function [g1,g2,g3,g4,g5,g6,g7,g8,g9,g10,g11,g12,g13,g14,g15,g16,g17,g18,g19,g20]=coef(x)

% Ingeven van de modelparameters, als functie van de geschatte waarde x

m1=x(1);
m2=x(2);
h1=x(3);
h2=x(4);
l1=2*h1;
l2=2*h2;
J1=m1*h1*h1/3;
J2=m2*h2*h2/3;

% Berekenen van de coëfficientenmatrices A, B, C en D

AA=[J1+m1*h1*h1+m2*l1*l1+m2*l1*h2 m2*l1*h2 ; J2+m2*h2*h2+m2*l1*h2 J2+m2*h2*h2];
BB=[b1,-b2;0,b2];
CC=[k1-(m2*l1+m1*h1)*g , -k2 ; -m2*h2*g , k2-m2*h2*g];
DD=[-m2*l1*h2 , m2*l1*h2];

% Berekenen van de coëfficienten voor de toestandsbeschrijving

d=det(AA);
g1=-AA(1,2)/d;
g2=AA(2,2)/d;
g3=AA(1,2)*DD(2)/AA(2,2)/d;
g4=-AA(2,1)*g3;
g5=-DD(1)/AA(2,2)/AA(2,2)/d;
g6=-DD(1)*(1-AA(2,1)/AA(2,2))*(1-AA(2,1)/AA(2,2))/d;
g7=-2*DD(1)*(1-AA(2,1)/AA(2,2))/AA(2,2)/d;
g8=-AA(2,1)*g5;
g9=-AA(2,1)*g6;
g10=-AA(2,1)*g7;
g11=(AA(1,2)*CC(2,2)-AA(2,2)*CC(1,2))/AA(2,2)/d;
g12=-AA(2,1)*g11-(AA(2,2)*CC(1,1)-AA(1,2)*CC(2,1))/d;
g13=(AA(1,2)+AA(2,2))*b2/AA(2,2)/d;
g14=-AA(2,1)*g13-AA(2,2)*b1/d;
g15=-DD(2)/AA(2,2);
g16=-AA(2,1)*g15;
g17=-CC(2,2)/AA(2,2);
g18=-AA(2,1)*g17-CC(2,1);
g19=-b2/AA(2,2);
g20=-AA(2,1)*g19;

end
```

GEVHMTM.M

```
% Gevoeligheidsbepaling van het dubbelslingermodel, berekend de responsie gegeven de
% massatraagheidsmomenten

%Ingeven van modelgegevens

m1=input('m1 = ');
m2=input('m2 = ');
h1=input('h1 = ');
h2=input('h2 = ');
b1=input('b1 = ');
b2=input('b2 = ');
k1=input('k1 = ');
k2=input('k2 = ');
g=input('g = ');
l1=2*h1;
l2=2*h2;
xdda=3*sin(21.71*t);

% Ingeven van een beginschatting voor J1 en J2

pause(disp ('% x=[J1,J2]'))
x=input('x = ');

% Berekenen van de coëfficientenmatrices

AA=[m1*h1*h1+m2*l1*l1+m2*l1*h2 m2*l1*h2 ; m2*h2*h2+m2*l1*h2 m2*h2*h2];
BB=[b1 , -b2 ; 0 , b2];
CC=[k1-(m2*l1+m1*h1)*g , -k2 ; -m2*h2*g , k2-m2*h2*g];
DD=[-m2*l1*h2 , m2*l1*h2];

% Ingeven van de beginwaarden

t0 = 0;
tfinal=0.2;
y0=[0;0;0;0];
toll=1.e-6;
F=[(m1*h1+m2*l1)*xdda , m2*h2*xdda];

% Globaal definiëren van coëfficienten

global t0 tfinal y0 toll DA AA BB CC DD
global g1 g2 g3 g4 g5 g6 g7 g8 g9 g10 g11 g12 g13 g14 g15 g16 g17 g18 g19 g20

% Ruimte reserveren voor de resultaten

P=zeros(50,30)

% responsiebepaling bij variërende massatraagheidsmomenten J1 en J2

j=1

    for i = 1:14
        [g1,g2,g3,g4,g5,g6,g7,g8,g9,g10,g11,g12,g13,g14,g15,g16,g17,g18,g19,g20]=coefmtm(x);
```

```
        [T1,Y]=ode45('nltoest',t0,tfinal,y0,toll);  % Responsiebepaling

        [n,m]=size(Y);          % Selectie en opslaan van resultaten
        k=1+(i-1)*m/2;
        P1(1:n,k:i*m/2) = [Y(1:n,2) , (Y(1:n,1)/(AA(2,2)+x(2))+
                                -(AA(2,1)+x(2))/(AA(2,2)+x(2))*Y(1:n,2))];

% Informatie over het gevoeligheidsproces

        t=ones(1,2);
        T=T*t;
        plot(T1,P1(1:n,k:i*m/2))

        x(j)=x(j)+0.2
    end

    P11=P1(:,1:14);
    P12=P1(:,15:28);

% Bewaren van de resultaten in een ascii-file

    save gpm1.m P11 P12 /ascii


j=2

    for i = 1:14
        [g1,g2,g3,g4,g5,g6,g7,g8,g9,g10,g11,g12,g13,g14,g15,g16,g17,g18,g19,g20]=coefmtm(x);

        [T2,Y]=ode45('nltoest',t0,tfinal,y0,toll);  % Responsiebepaling

        [n,m]=size(Y);              % Selectie en opslaan van resultaten
        k=1+(i-1)*m/2;
        P2(1:n,k:i*m/2)=[Y(1:n,2) , (Y(1:n,1)/(AA(2,2)+x(2))-(AA(2,1)+
                                            x(2))/(AA(2,2)+x(2))*Y(1:n,2))];

% Informatie over het gevoeligheidsproces

        t=ones(1,2);
        T=T*t;
        plot(T2,P2(1:n,k:i*m/2))

        x(j)=x(j)+0.2
    end

    P21=P2(:,1:14);
    P22=P2(:,15:28);

% Bewaren van de resultaten in een ascii-file

    save gpm2.m P21 P22 /ascii

end
```

COEFMTM.M

```
% Programma voor het opstellen van de coëfficienten bij variatie van het massatraagheidsmoment

function [g1,g2,g3,g4,g5,g6,g7,g8,g9,g10,g11,g12,g13,g14,g15,g16,g17,g18,g19,g20]=coefmtm(x)

A=AA+[x(1),0;x(2),x(2)];
d=det(A);
g1=-A(1,2)/d;
g2=A(2,2)/d;
g3=A(1,2)*DD(2)/A(2,2)/d;
g4=-A(2,1)*g3;
g5=-DD(1)/A(2,2)/A(2,2)/d;
g6=-DD(1)*(1-A(2,1)/A(2,2))*(1-A(2,1)/A(2,2))/d;
g7=-2*DD(1)*(1-A(2,1)/A(2,2))/A(2,2)/d;
g8=-A(2,1)*g5;
g9=-A(2,1)*g6;
g10=-A(2,1)*g7;
g11=(A(1,2)*CC(2,2)-A(2,2)*CC(1,2))/A(2,2)/d;
g12=-A(2,1)*g11-(A(2,2)*CC(1,1)-A(1,2)*CC(2,1))/d;
g13=(A(1,2)+A(2,2))*BB(2,2)/A(2,2)/d;
g14=-A(2,1)*g13-A(2,2)*BB(1,1)/d;
g15=-DD(2)/A(2,2);
g16=-A(2,1)*g15;
g17=-CC(2,2)/A(2,2);
g18=-A(2,1)*g17-CC(2,1);
g19=-BB(2,2)/A(2,2);
g20=-A(2,1)*g19;

end
```

BIJLAGE 8

PC-Matlab programma voor een kleinste-kwadraten-fit op het enkelslingermodel

DV2FITTT.M

```
function [C]=dv2fit(PHI,t,F)
%functie om een 2e orde differentiaalvergelijking te fitten
%
[n,m]=size(PHI);
%Maak PHI en t van hetzelfde formaat
T=ones(m,1)*t';
T=T';
%
PHID=diff(PHI)./diff(T);
%
%Maak een kolom met tijdstippen t + 1/2*delta t
td=(t(1:(n-1))+t(2:n))/2;
%Maak PHID en td van hetzelfde formaat
TD=ones(m,1)*td';
TD=TD';
%
PHIDD = diff(PHID)./diff(TD);
%
%Maak PHI, PHID en PHIDD van hetzelfde formaat
PHIS=PHI(2:(n-1),:);
%
PHIDS=(PHID(1:(n-2),:)+PHID(2:(n-1),:))/2;
%
%Berg PHI en zijn afgeleides onder in matrix A
A=[PHIDD,PHIDS,PHIS];
%
%Maak de krachtenmatrix van hetzelfde formaat
FS=F(2:(n-1),:);
%
%Selecteer bekende matrices en voeg ze aan de krachtenmatrix toe
M=input('massamatrix {hoog naar laag}, onbekend = 0 ');
%
B=input('dempingsmatrix {hoog naar laag}, onbekend = 0 ');
%
K=input('stijfheidsmatrix {hoog naar laag}, onbekend = 0 ');
%
if M==0
   RHS=FS;
else V=A(:,1:m)*M';
      RHS=FS-V;
      A=[PHIDS,PHIS];
end
%
if B==0
   RHS=RHS;
elseif M==0
       W=A(:,(m+1):(2*m))*B';
       RHS=RHS-W;
       A=[PHIDD,PHIS];
```

```
else   W=A(:,1:m)*B';
       A=PHIS;
end
%
if K==0
   RHS=RHS;
elseif M==0 & B==0
       Y=A(:,((2*m)+1):(3*m))*K';
       RHS=RHS-Y;
       A=[PHIDD,PHIDS];
elseif M==0 & B~=0
       Y=A(:,(m+1):(2*m))*K';
       RHS=RHS-Y;
       A=PHIDD;
elseif M~=0 & B==0
       Y=A(:,1:m)*K';
       RHS=RHS-Y;
       A=PHIDS;
end
%
%Los het stelsel vergelijkingen in een kleinste-kwadraten manier op
C=A\RHS
end
```

BIJLAGE 9

PC-Matlab programma voor een kleinste-kwadraten-fit op het dubbelslingermodel

DV2FIT.M

```
function [C]=dv2fit(PHI,t,F)
%functie om een 2e orde differentiaalvergelijking te fitten
%
[n,m]=size(PHI);
%Maak PHI en t van hetzelfde formaat
T=ones(m,1)*t';
T=T';
%
PHID=diff(PHI)./diff(T);
%
%Maak een kolom met tijdstippen t + 1/2*delta t
td=(t(1:(n-1))+t(2:n))/2;
%Maak PHID en td van hetzelfde formaat
TD=ones(m,1)*td';
TD=TD';
%
PHIDD = diff(PHID)./diff(TD);
%
%Maak PHI, PHID en PHIDD van hetzelfde formaat
PHIS=PHI(2:(n-1),:);
%
PHIDS=(PHID(1:(n-2),:)+PHID(2:(n-1),:))/2;
%
%Berg PHI en zijn afgeleides onder in matrix A
A=[PHIDD,PHIDS,PHIS];
%
%Maak de krachtenmatrix van hetzelfde formaat
FS=F(2:(n-1),:);
%
%Selecteer bekende matrices en voeg ze aan de krachtenmatrix toe
M=input('massamatrix {hoog naar laag}, onbekend = 0 ');
%
B=input('dempingsmatrix {hoog naar laag}, onbekend = 0 ');
%
K=input('stijfheidsmatrix {hoog naar laag}, onbekend = 0 ');
%
%Splits de matrices A en FS op in matrices per lichaam (diagonaalvorm van
%de massamatrix, dempingsmatrix en stijfheidsmatrix vormen het resultaat)
%en splits de bekende matrices op in waarden per lichaam
for I=1:m;
    i=(m+1)-I;

    Ai=[PHIDD(:,I),PHIDS(:,I),PHIS(:,I)];
    Fi=FS(:,I);
%Verwerk de bekende matrices
    if M==0
       RHSi=Fi;
    else M(i)=M(I,I);
         V=Ai(i,1)*M(i);
```

```
        RHSi=Fi-V;
        Ai=[PHIDS(:,I),PHIS(:,I)];
    end
%
    if B==0
       RHSi=RHSi;
    elseif M==0
            B(i)=B(I,I);
            W=Ai(i,2)*B(i);
            RHSi=RHSi-W;
            Ai=[PHIDD(:,I),PHIS(:,I)];
    else   B(i)=B(I,I);
         W=Ai(i,1)*B(i);
         Ai=PHIS(:,I);
    end
%
    if K==0
       RHSi=RHSi;
    elseif M==0 & B==0
            K(i)=K(I,I);
            Y=Ai(i,3)*K(i);
            RHSi=RHSi-Y;
            Ai=[PHIDD(:,I),PHIDS(:,I)];
    elseif M==0 & B~=0
            K(i)=K(I,I);
            Y=Ai(i,2)*K(i);
            RHSi=RHSi-Y;
            Ai=PHIDD(:,I);
    elseif M~=0 & B==0
            K(i)=K(I,I);
            Y=Ai(i,2)*K(i);
            RHSi=RHSi-Y;
            Ai=PHIDS(:,I);
    end
%
%Los het stelsel vergelijkingen in een kleinste-kwadraten manier op
    Ci=Ai\RHSi;
    [y,z]=size(Ci);
    for k=1:z
        for j=1:y
            C(i,j)=Ci(j,k);
        end
    end
end
C
end
```

BIJLAGE 10

PC-Matlab programma voor stapsgewijze kleinste-kwadraten-fit op het dubbelslingermodel

CF.M

```
% Routine om de onbekende grootheden te fitten op niet lineaire
%2e orde d.v.'s
%haal de data binnen en selecteer de gewenste gegevens
be;
t=XE(:,1);
X1=XE(:,2);
X2=XHE(:,2);
X3=XS(:,2);
X=[X1 X2 X3];          % samenstellen verplaatsingsmatrix
%
l = input('rijvector met slingerlengtes, l=  ')
F = input('matrix met bekende externe krachten, F=  ')
%Zet de verplaatsingen om im hoekverdraaiingen
%
[a,b] = size(X);
for j = 1:(b-1);
   PHI(:,j) = asin((X(:,j)-X(:,(j+1)))/l(j));    % [PHI1,PHI2]
end
%Bereken de onbekenden op een kleinste-kwadraten manier
%
[n,m]=size(PHI);
%Bereken de afgeleiden (1e en 2e) van PHI
T=ones(m,1)*t';
T=T';
%
PHID=diff(PHI)./diff(T);                    % [PHID1,PHID2]
%
%Maak een kolom met tijdstippen t + 1/2*delta t
td=(t(1:(n-1))+t(2:n))/2;
%
TD=ones(m,1)*td';
TD=TD';
%
PHIDD = diff(PHID)./diff(TD);               % [PHIDD1,PHIDD2]
%
%Maak PHI, PHID en PHIDD van hetzelfde formaat
PHIS=PHI(2:(n-1),:);
%
PHIDS=(PHID(1:(n-2),:)+PHID(2:(n-1),:))/2;
%
NLPHI1=PHIS(:,2).*(PHIDS(:,1)+PHIDS(:,2)).*(PHIDS(:,1)+PHIDS(:,2));
NLPHI2=PHIS(:,2).*PHIDS(:,1).*PHIDS(:,1);
NLPHI=[NLPHI1,NLPHI2];
%Berg PHI en zijn afgeleides onder in matrix A
A=[PHIDD,PHIDS,PHIS,NLPHI];    % [PHIDD1,PHIDD2,PHID1,PHID2,PHI1,PHI2,NLPHI1,NLPHI2]
%
%Maak de krachtenmatrix van hetzelfde formaat
FS=F(2:(n-1),:);
%
```

```
pause %2e bewegingsvergelijking, let op B(2)1=0!
%Splits alle matrices op per bewegingsvergelijking
FI=FS(:,2);
M=input('massa"s, onbekend = -1, [M1,M2] =   ');          % A(2,:)
B=input('dempingsfactoren, onbekend = -1, [B1,B2] =   '); % [0,-1]
K=input('veerstijfheden, onbekend = -1, [K1,K2] =   ');   % [-1,-1]
D=input('niet lineaire term, onbekend = -1, D2 = ');      % DD2
%
RHS2=FI;
A2=A;

if M(1,1) ~= -1
   Y12=M(1,1).*A2(:,1);
   RHS2=RHS2-Y12;
   A2=A2(:,2:8);
end

if M(1,2) ~= -1
   if M(1,1) ~= -1
      Y22=M(1,2).*A2(:,1);
      A2=A2(:,2:7);
   else Y22=M(1,2).*A2(:,2);
        A2=[A2(:,1),A2(:,3:8)];
   end
   RHS2=RHS2-Y22;
end

if B(1,1) ~= -1
   if min(size(A2))==8
      V12=B(1,1).*A2(:,3);
      A2=[A2(:,1:2),A2(:,4:8)];
   elseif min(size(A2))==7
         V12=B(1,1).*A2(:,2);
         A2=[A2(:,1),A2(:,3:7)];
   else V12=B(1,1).*A2(:,1);
        A2=A2(:,2:6);
   end
   RHS2=RHS2-V12;
end

if B(1,2) ~= -1
   if min(size(A2))==8
      V22=B(1,2).*A2(:,4);
      A2=[A2(:,1:3),A2(:,5:8)];
   elseif min(size(A2))==7
         V22=B(1,2).*A2(:,3);
         A2=[A2(:,1:2),A2(:,4:7)];
   elseif min(size(A2))==6
         V22=B(1,2).*A2(:,2);
         A2=[A2(:,1),A2(:,3:6)];
   else V22=B(1,2).*A2(:,1);
        A2=A2(:,2:5);
   end
   RHS2=RHS2-V22;
end
```

```
if K(1,1) ~= -1
   if min(size(A2))==8
      W12=K(1,1).*A2(:,5);
      A2=[A2(:,1:4),A2(:,6:8)];
   elseif min(size(A2))==7
         W12=K(1,1).*A2(:,4);
         A2=[A2(:,1:3),A2(:,5:7)];
   elseif min(size(A2))==6
         W12=K(1,1).*A2(:,3);
         A2=[A2(:,1:2),A2(:,4:6)];
   elseif min(size(A2))==5
         W12=K(1,1).*A2(:,2);
         A2=[A2(:,1),A2(:,3:5)];
   else W12=K(1,1).*A2(:,1);
        A2=A2(:,2:4);
   end
   RHS2=RHS2-W12;
end

if K(1,2) ~= -1
   if min(size(A2))==8
      W22=K(1,2).*A2(:,6);
      A2=[A2(:,1:5),A2(:,7:8)];
   elseif min(size(A2))==7
         W22=K(1,2).*A2(:,5);
         A2=[A2(:,1:4),A2(:,6:7)];
   elseif min(size(A2))==6
         W22=K(1,2).*A2(:,4);
         A2=[A2(:,1:3),A2(:,5:6)];
   elseif min(size(A2))==5
         W22=K(1,2).*A2(:,3);
         A2=[A2(:,1:2),A2(:,4:5)];
   elseif min(size(A2))==4
         W22=K(1,2).*A2(:,2);
         A2=[A2(:,1),A2(:,3:4)];
   else W22=K(1,2).*A2(:,1);
        A2=A(:,2:3);
   end
   RHS2=RHS2-W22;
end

if D~=-1
   if min(size(A2))==8
      Z2=D.*A2(:,8);
      A2=A2(:,1:6);
   elseif min(size(A2))==7
         Z2=D.*A2(:,7);
         A2=A2(:,1:5);
   elseif min(size(A2))==6
         Z2=D.*A2(:,6);
         A2=A2(:,1:4);
   elseif min(size(A2))==5
         Z2=D.*A2(:,5);
         A2=A2(:,1:3);
   elseif min(size(A2))==4
         Z2=D.*A2(:,4);
```

```
        A2=A2(:,1:2);
   else min(size(A2))==3
        Z2=D.*A2(:,3);
        A2=A2(:,1);
   end
   RHS2=RHS2-Z2;
end

C2=A2\RHS2

pause % 1e bewegingsvergelijking, let op koppeling ! ! !
%
M=input('massa"s, onbekend = -1, [M1,M2] =  ');
pause % B2 = -B2(2)
B=input('dempingsfactoren, onbekend = -1, [B1,B2] =  ');
pause % K2 = K1(2) - K2(2)
K=input('veerstijfheden, onbekend = -1, [K1,K2] =  ');
D=input('niet lineaire term, onbekend = -1, D1 =  ');
%
FI=FS(:,1);
RHS1=FI;
A1=A;

if M(1,1) ~= -1
   Y11=M(1,1).*A1(:,1);
   RHS1=RHS1-Y11;
   A1=A1(:,2:8);
end

if M(1,2) ~= -1
   if M(1,1) ~= -1
      Y21=M(1,2).*A1(:,1);
      A1=A1(:,2:7);
   else Y21=M(1,2).*A1(:,2);
        A1=[A1(:,1),A1(:,3:8)];
   RHS1=RHS1-Y21;
   end
end

if B(1,1) ~= -1
   if min(size(A1))==8
      V11=B(1,1).*A1(:,3);
      A1=[A1(:,1:2),A1(:,4:8)];
   elseif min(size(A1))==7
        V11=B(1,1).*A1(:,2);
        A1=[A1(:,1),A1(:,3:7)];
   else V11=B(1,1).*A1(:,1);
       A1=A1(:,2:6);
   end
   RHS1=RHS1-V11;
end

if B(1,2) ~= -1
   if min(size(A1))==8
      V21=B(1,2).*A1(:,4);
      A1=[A1(:,1:3),A1(:,5:8)];
```

```
   elseif min(size(A1)) = =7
          V21=B(1,2).*A1(:,3);
          A1=[A1(:,1:2),A1(:,4:7)];
   elseif min(size(A1)) = =6
          V21=B(1,2).*A1(:,2);
          A1=[A1(:,1),A1(:,3:6)];
   else V21=B(1,2).*A1(:,1);
        A1=A1(:,2:5);
   end
   RHS1=RHS1-V21;
end

if K(1,1) ~= -1
   if min(size(A1)) = =8
      W11=K(1,1).*A1(:,5);
      A1=[A1(:,1:4),A1(:,6:8)];
   elseif min(size(A1)) = =7
          W11=K(1,1).*A1(:,4);
          A1=[A1(:,1:3),A1(:,5:7)];
   elseif min(size(A1)) = =6
          W11=K(1,1).*A1(:,3);
          A1=[A1(:,1:2),A1(:,4:6)];
   elseif min(size(A1)) = =5
          W11=K(1,1).*A1(:,2);
          A1=[A1(:,1),A1(:,3:5)];
   else W11=K(1,1).*A1(:,1);
        A1=A1(:,2:4);
   end
   RHS1=RHS1-W11;
end

if K(1,2) ~= -1
   if min(size(A1)) = =8
      W21=K(1,2).*A1(:,6);
      A1=[A1(:,1:5),A1(:,7:8)];
   elseif min(size(A1)) = =7
          W21=K(1,2).*A1(:,5);
          A1=[A1(:,1:4),A1(:,6:7)];
   elseif min(size(A1)) = =6
          W21=K(1,2).*A1(:,4);
          A1=[A1(:,1:3),A1(:,5:6)];
   elseif min(size(A1)) = =5
          W21=K(1,2).*A1(:,3);
          A1=[A1(:,1:2),A1(:,4:5)];
   else W21=K(1,2).*A1(:,2);
        A1=[A1(:,1),A1(:,3:4)];
   end
   RHS1=RHS1-W21;
end

if D ~= -1
   if min(size(A1)) = =8
      Z1=D.*A1(:,7);
      A1=A1(:,1:6);
   elseif min(size(A1)) = =7
          Z1=D.*A1(:,6);
```

```
        A1=A1(:,1:5);
   elseif min(size(A1))==6
        Z1=D.*A1(:,5);
        A1=A1(:,1:4);
   elseif min(size(A1))==5
        Z1=D.*A1(:,4);
        A1=A1(:,1:3);
   elseif min(size(A1))==4
        Z1=D.*A1(:,3);
        A1=A1(:,1:2);
   else Z1=D.*A1(:,2);
       A1=A(:,1);
   end
   RHS1=RHS1-Z1;
end

C1=A1\RHS1
%
C1,C2

end
```

BE.M

```
dsl;                    % file met "meetgegevens" gegenereerd via DADS
dsl2;                   % file met "meetgegevens" gegenereerd via DADS
t=XE(:,1);              % tijdvector

% Ingeven van de modelparameters

m1=input('m1 = ');
m2=input('m2 = ');
h1=input('h1 = ');
h2=input('h2 = ');
g=input('g = ');
l1=2*h1;
l2=2*h2;
J1=m1*l1*l1/12;
J2=m2*l2*l2/12;
xdda=3*sin(21.71*t);
F=[(m1*h1+m2*l1)*xdda , m2*h2*xdda]

pause(disp ('% x = [b1, b2, k1, k2] '))  % De in te geven beginschatting

% De matrices A, B, C en D uit de bewegingsvergelijkingen

AA=[J1+m1*h1*h1+m2*l1*l1+m2*l1*h2 m2*l1*h2 ; J2+m2*h2*h2+m2*l1*h2 J2+m2*h2*h2]
BB=[0 , 0 ; 0 , 0]
CC=[-(m2*l1+m1*h1)*g , 0 ; -m2*h2*g , -m2*h2*g]
DD=[-m2*l1*h2 , m2*l1*h2]
```

```
% De beginwaarden

t0 = 0
tfinal=0.2
y0=[0;0;0;0]
toll=1.e-8

% Berg alle modelgegevens op in een matrix en maak de coëfficienten globaal

DA=[t,DIS1(:,2),DIS2(:,2)];
global t0 tfinal y0 toll DA AA BB CC DD
global g1 g2 g3 g4 g5 g6 g7 g8 g9 g10 g11 g12 g13 g14 g15 g16 g17 g18 g19 g20

end
```

BIJLAGE 11

De Nelder-Mead simplex methode

Simplex methode

Dit is een iteratieve methode voor het vinden van een minimum van een niet-lineaire functie, waarbij vanuit een beginschatting op zodanige wijze een volgende schatting wordt bepaald dat de functiewaarde van de doelfunctie F altijd afneemt. De Simplex methode is een direkte methode, die geen gebruik maakt van de afgeleiden. Het minimum wordt dan ook gevonden door het steeds maar weer vergelijken van de doelfunctie in verschillende hoekpunten van de simplex (een simplex is een veelhoek met een aantal hoekpunten, waarbij het aantal één groter is dan de dimensie van de parameterruimte. Vb. Bij een doelfunctie die 2 parameters bezit zal de simplex bestaan uit een driehoek.)

De ***Nelder-Mead Simplex methode****houdt in:*

Stel dat bj iteratie k de hoekpunten van de simplex zijn: x_0, x_1, ..., x_{n-1}, x_n met bijbehorende functiewaarden F_0, F_1, ..., F_{n-1}, F_n zodanig gerangschikt dat: $F_n > F_{n-1} > ..... > F_1 > F_0$

x_0 Is het beste hoekpunt van de simplex en x_n het slechtste, dize moet dan ook vervangen worden. We definiëren daarom een middelpunt c voor de hoekpunten x_0, x_1, ..., x_{n-1}.

Dit middelpunt wordt als volgt berekend:

Het punt x_n wordt vervangen door het punt x_r dat als volgt gevonden wordt:

waarbij $\alpha > 0$ de reflectiecoëfficiënt is.

Er zijn 3 verschillende situaties mogelijk voor het punt x_r. De functiewaarde van x_r, F_r is zodanig dat:

1. *$F_0 < F_r < F_{n-1}$*
2. *$F_r < F_0$ → x_r is het nieuwe beste punt*
3. *$F_r > F_{n-1}$ → x_r is het nieuwe slechtste punt*

ad.1) x_r neemt de plaats in voor x_n en de iteratieslag is ten einde.

ad.2) Blijkbaar is de reflectie in deze richting zo goed, dat het zin heeft om de simplex in deze richting te expanderen, waarvoor het punt x_e wordt bepaald.

met $\beta > 1$ de expansie coëfficiënt.

Indien $F_e < F_0$ dan neemt x_e de plaats in van x_n, anders wordt aangenomen dat de expansie gefaald heeft en wordt x_n vervangen door x_r.

In beide gevallen is de iteratie ten einde.

ad.3) Als x_r het nieuwe slechtste punt is dan wordt aangenomen dat de simplex te groot is om enige voortgang te boeken in deze richting. Daarom wordt het punt x_c gedefinieerd.

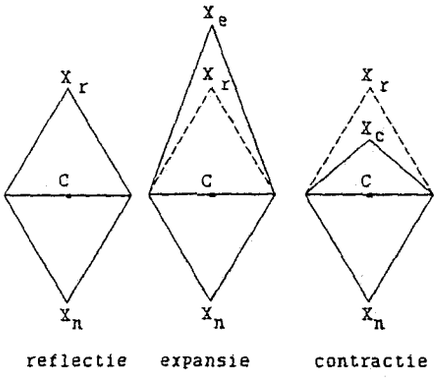
Waarbij γ de contractiecoëfficiënt is; $0 < \gamma < 1$.

Als $F_c < min(F_n F_r)$ dan is de contractie geslaagd en wordt x_n vervangen door x_c. Echter wanneer $F_c < min(F_n F_r)$ niet geldt, dan wordt een iteratiestap uitgevoerd waarbij alle afstanden tussen het beste hoekpunt x_0 en de rest van de hoekpunten van de simplex gehalveerd worden. Deze iteratiestap wordt ook wel een uitgebreide contractie genoemd.

De reflectie-, expansie- en contractiecoëfficiënten staan vast ingesteld binnen de PC-Matlab routine en zijn als volgt:

$\alpha = 1$
$\beta = 2$
$\gamma = 0{,}5$

De reflectie, expansie en contractie stappen zijn in figuur A1 weergegeven voor een functie F met 2 variabelen.



figuur A1. De verschillende stappen voor het Nelder-Mead algorithme.

BIJLAGE 12

PC-Matlab programma voor een fit op het dubbelslingermodel m.b.v. FMINS

MEETFIT.M

```
% Programma om te fitten op de meetgegevens van Mariëlle en Pauline

% Het ophalen van de meetgegevens

hv2;
hv8;
hv5;

% Selectie van het meetgebied via een plaatje van de gegevens

HVM2=hv2(56,2);
HVM8=hv8(56,2);
HVM5=hv5(56,2);
HV2=hv2(56:71,2)-HVM2;
HV8=hv8(56:71,2)-HVM8;
HV5=hv5(56:71,2)-HVM5;

HV=[hv2(56:71,2),HV8,HV5]; % De te gebruiken meetgegevens
t=[0;hv2(1:15,1)];         % De bijbehorende tijdvector

% Invoeren van de modelgegevens

m1 = input('m1 = ');
m2 = input('m2 = ');
h1 = input('h1 = ');
h2 = input('h2 = ');
g = input('g = ');
J1=m1*h1*h1/3;
J2=m2*h2*h2/3;
l1=2*h1;
l2=2*h2;
xdda=3*sin(21.71*t);

pause(disp ('% x = [b1, b2, k1, k2] '));
x=input('x = ')

% opstellen van de coëfficientenmatrices van de bewegingsvergelijkingen

AA=[J1+m1*h1*h1+m2*l1*l1+m2*l1*h2 m2*l1*h2 ; J2+m2*h2*h2+m2*l1*h2 J2+m2*h2*h2]
BB=[0 , 0 ; 0 , 0]
CC=[-(m2*l1+m1*h1)*g , 0 ; -m2*h2*g , -m2*h2*g]
DD=[-m2*l1*h2 , m2*l1*h2]

% ingeven van meettijd en begincondities

t0 = 0
tfinal=0.2
y0=[0;0;0;0]
toll=1.e-8
```

```
l=[l1 , l2]
F=[(m1*h1+m2*l1)*xdda , m2*h2*xdda]

% omzetten van de horizontale verplaatsingen in hoekverdraaiingen

PHI1=-asin((HV(:,2)-HV(:,1))/l1);
PHI2=-asin((HV(:,3)-HV(:,2))/l2);

% Opbergen van de meetgegevens in een matrix en globaal definiëren van coëfficienten

DA=[t,PHI1,PHI2];
global t0 tfinal y0 toll DA AA BB CC DD

% Responsiebepaling via de toestandsbeschrijving en globaal definiëren van coëfficienten

[g1,g2,g3,g4,g5,g6,g7,g8,g9,g10,g11,g12,g13,g14,g15,g16,g17,g18,g19,g20] = coef(x);
global g1 g2 g3 g4 g5 g6 g7 g8 g9 g10 g11 g12 g13 g14 g15 g16 g17 g18 g19 g20

% Berekenen van de optimale parameters b1, b2, k1 en k2 via PC-Matlab routine FMINS

X=fmins('fit',x)      % FMINS is te vervangen door FMINU, bij CONSTR dient ook 'fit' te worden
                      % vervangen door 'fitg'

end
```

FITG.M

```
% Programma ter definitie van een fitfunctie met beginvoorwaarden, berekend de responsie
% gegeven de veer en dempingsconstanten

function [f,G]=fit(x,DATA)

% Inlezen van meetgegevens

DATA=DA;
time=DATA(:,1);

% Berekenen van toestandscoëfficienten

[g1,g2,g3,g4,g5,g6,g7,g8,g9,g10,g11,g12,g13,g14,g15,g16,g17,g18,g19,g20]=coef(x);

% Berekenen van de responsie

Y=ode45pa2('nltoest',time,y0);

% Bewaren van de resultaten van de berekening en meting

P=[Y(:,2) , (Y(:,1)/AA(2,2)-AA(2,1)/AA(2,2)*Y(:,2))];
Q=[DATA(:,2) , DATA(:,3)];

% Definitie van de te optimaliseren fout

f=norm(P-Q);
```

```
% Ingeven van de randvoorwaarden

G=[-x(1),-x(2),-x(3),-x(4)];

% Informatie over het fitproces

plot(time,P(:,1),time,P(:,2),time,Q(:,1),'+',time,Q(:,2),'o')
text(0.2,0.6,['b1 = 'num2str(x(1))],'sc')
text(0.2,0.55,['b2 = 'num2str(x(2))],'sc')
text(0.2,0.5,['k1 = 'num2str(x(3))],'sc')
text(0.2,0.45,['k2 = 'num2str(x(4))],'sc')
text(0.2,0.4,['fout = 'num2str(f)],'sc')

end
```

fminu, fmins

Purpose:

Find the minimum of an unconstrained multivariable function.

Synopsis:

```
x = fminu('fun', x0)
x = fminu('fun', x0, options)
x = fminu('fun', x0, options, 'grad')
x = fminu('fun', x0, options, 'grad', p1, p2, ... )
[x, options] = fminu('fun', x0, ... )

[ ... ] = fmins('fun', x0, ... )
```

Description:

fminu and fmins find the minimum of a scalar function of several variables, starting from an initial estimate. This is generally referred to as *unconstrained non-linear optimization* and is mathematically stated as:

$$\underset{X}{\text{minimize}}\ f(X)$$

where bold uppercase characters indicate that X is a matrix or vector, and lowercase characters indicate that f is a scalar function.

x = fminu('fun', x0) starts at x0 and finds a minimum using the function fun described in the M-file fun.m. The function fun should return a scalar function value, f = fun(x).

Alternatively, an expression can be substituted for the function name, with x representing the independent variables. For example:

```
x = fminu('fun(x.*x)', x0)
```

x = fminu('fun', x0, options) performs the minimization according to the options vector. In particular:

- options(1) controls display. Setting this to a value of 1 produces a tabular display of intermediate results.
- options(2) controls the accuracy of x at the solution.
- options(3) controls the accuracy of f at the solution.

For more information on options, including default settings, see the foptions reference page.

x = fminu('fun', x0, options, 'grad') calls the function grad to obtain the partial derivatives of the function (the Jacobian), df/dX, at the point X:

```
df = grad(x).
```

x = fminu('fun', x0, options, 'grad', p1, p2, ...) passes parameters directly to the function fun and grad. For example:

```
f = fun(x, p1, p2, ... )
df = grad(x, p1, p2, ... )
```

This is useful so that the same M-file can solve a number of similar problems with different parameters, avoiding the need to use global variables. Both options and grad can be empty matrices, in which case default options are used and finite differences are calculated.

A second lefthand argument returns optimization parameters. For example,

```
[x, options] = fminu('fun', x0)
```

returns the parameters used in the optimization. options(10) contains the number of function evaluations used.

The default algorithm for fminu is the BFGS [2-5] quasi-Newton method using a mixed quadratic and cubic line search procedure. Setting options(7) to 1 sets the line search algorithm to a cubic polynomial method. fmins uses the simplex search algorithm of Nelder and Mead[1] and has the identical calling syntax as fminu. Note that since fmins does not use gradient information grad is always ignored.

Examples:

Find values that minimize:

$$f(x) = 100\,(x_2 - x_1^2)^2 + (1 - x_1)^2$$

starting at the point:

$$x = [-1.2\ \ 1]$$

Step 1: Write M-file:

```
function f = fun(x)
f = 100*(x(2)-x(1)^2)^2+(1-x(1))^2;          % Cost function
```

Step 2: Invoke Optimization Routine:

```
x = [-1,1];              % Make a starting guess at the solution
x = fminu('fun', x)
```

After 132 iterations, this example generates the solution:

```
x =
   1.0000 1.0000
fun(x) =
   8.8348e-11
```

Limitations:

The function to be minimized must be continuous. fminu may only give local solutions.

fminu only handles real variables. When *X* has complex variables, they must be split into real and imaginary parts.

Algorithm:

The default algorithm for fminu is a quasi-Newton method. fmins uses the simplex search method of Nelder and Mead[1]. fmins is generally less efficient than fminu for problems of order greater than two. However, when the problem is highly discontinuous, fmins may be more robust.

The quasi-Newton method (i.e., fminu) uses the BFGS [2-5] formula for updating the approximation of the Hessian matrix. The DFP [7, 8] formula, that avoids direct calculation of the inverse Hessian, is selected by setting options(6) = 1. A steepest descent is selected by setting options(6) = 2, although this is not recommended.

The default line search algorithm, options(7) = 0, is a safeguarded mixed quadratic and cubic polynomial interpolation and extrapolation method. A safeguarded cubic polynomial method may be selected by setting options(7) = 1. This method generally requires fewer function evaluations but more gradient evaluations. Thus, if gradients are being supplied and can be calculated inexpensively, the cubic polynomial line search method is preferable.

See also:

foptions

References:

[1] J.A. Nelder and R. Mead, *A Simplex Method for Function Minimization,* Computer Journal Vol. 7, p. 308-313.

[2] C.G. Broydon, *The Convergence of a Class of Double-rank Minimization Algorithms*, J. of the Inst. of Mathematics and its Applic., Vol. 6, p. 76-90, 1970.

[3] R. Fletcher, *A New Approach to Variable Metric Algorithms*, Computer Journal, Vol. 13, p. 317-322, 1970.

[4] D. Golfarb, *A Family of Variable Metric Updates Derived by Variational Means*, Mathematics of Comput., Vol. 24, p. 23-26, 1970.

[5] D.F. Shanno, *Conditioning of Quasi-Newton Methods for Function Minimization*, Mathematics of Comput., Vol. 24, p. 647-656, 1970.

[6] W.C. Davidon, *Variable Metric Method for Minimization*, A.E.C. Research and Development Report, ANL - 5990, 1959.

[7] R. Fletcher and M.J.D. Powell, *A Rapidly Convergent Descent Method for Minimization*, Computer Journal, Vol. 6, p. 163-168, 1963.

[8] R. Fletcher, *Practical Methods of Optimization*, Vol. 1, Unconstrained Optimization, John Wiley and Sons.

constr

Purpose:

Find the minimum of a constrained multivariable function.

Synopsis:

```
x = constr('fun', x0)
x = constr('fun', x0, options)
x = constr('fun', x0, options, vlb, vub, 'grad')
x = constr('fun', x0, options, vlb, vub, 'grad', p1, p2, ... )
[x, options] = constr('fun', x0, ... )
```

Description:

constr finds the constrained minimum of a scalar function of several variables, starting at an initial estimate. This is generally referred to as *constrained non-linear optimization*, and is mathematically stated as:

$$\underset{X}{\text{minimize}}\ f(X) \qquad \text{subject to: } G(X) \le \mathbf{0}$$

where bold uppercase characters indicate that X and $G(X)$ are matrices, and plain lowercase characters indicate that $f(X)$ is a scalar function.

x = constr('fun', x0) starts at x0 and finds a minimum of the function fun, defined in an M-file named fun.m. The function fun should return two arguments: a scalar value, f, of the function to be minimized, and a matrix of constraints, g:

```
[f,g] = fun(x)
```

Alternatively, an expression can be used with x representing the independent variables and with f and g representing the function and constraints. For example:

```
x = constr('f=fun(x); g=cstr(x);', x0)
```

The objective function, f, is minimized such that g ≤ zeros(g).

x = constr('fun', x0, options) defines a vector of optional parameters. In particular:

- options(1) controls display. Setting this to a value of 1 produces a tabular display of intermediate results.
- options(2) controls the accuracy of x at the solution.
- options(3) controls the accuracy of f at the solution.
- options(4) sets the maximum constraint violation that is acceptable.

For more information on the options vector, including default settings, see the foptions reference page.

x = constr('fun', x, options, vlb, vub) defines a set of lower and upper bounds on x through the matrices vlb and vub. This restricts the solution to the range vlb ≤ x ≤ vub. The variables, vlb and vub, are normally the same size as x. However, if vlb has *n* elements and less elements than x then only the first *n* elements in x are lower bounded; upper bounds in vub are defined in the same manner.

x = constr('fun', x0, options, vlb, vub, 'grad') calls the function grad, that returns the partial derivatives of the function and the constraints, *df/dX* and *dg/dX*, at the point *X*:

```
[df, dg] = grad(x)
```

The columns of dg should contain the partial derivatives for each of the constraints respectively, (i.e., the *i*th column of dg corresponds to the partial derivative of the *i*th constraint with respect to each of the elements in x).

x = constr('fun', x0, options, vlb, vub, 'grad', p1, p2, ...) passes parameters (i.e., p1, p2, etc.), directly to the function fun and grad:

```
[f, g] = fun(x, p1, p2, ... )
[df, dg] = grad(x, p1, p2, ... )
```

This is useful so that the same M-file can solve a number of similar problems with different parameters, avoiding the need to use global variables. Both options and grad may be set to empty matrices, in which case default options are used and finite differences are calculated.

A second lefthand argument returns optimization parameters. For example,

```
[x, options] = constr('fun', x0)
```

returns the parameters used in the optimization. options(10) contains the number of function evaluations used.

Equality constraints are placed in the first elements of g. When using equality constraints, options(13) specifies the number of equality constraints (see Tutorial, Section 2.1).

Examples:

Find values of X that minimize $f(X) = -x_1x_2x_3$, starting at the point X = [10,10,10], and subject to the constraints:

$$-x_1 - 2x_2 - 2x_3 \le 0$$
$$x_1 + 2x_2 + 2x_3 \le 72$$

Step 1: Write M-file:

```
function [f, g] = fun(x)
f = -x(1) * x(2) * x(3);
g(1) = -x(1) - 2 * x(2) - 2 * x(3);  % Evaluate Constraints
g(2) = x(1) + 2 * x(2) + 2 * x(3) - 72;
```

Step 2: Invoke Optimization Routine:

```
x0 = [10, 10, 10];      % Starting guess at the solution
x = constr('fun', x0)   % Invoke optimizer
```

After 49 iterations, the solution is:

```
x =
  24.0000   12.0000   12.0000

[f, g] = fun(x)

f =

 -3.4560e+03

g =

 -72    0
```

Algorithm:

constr uses a Sequential Quadratic Programming (SQP) method. In this method, a Quadratic Programming (QP) subproblem is solved at each iteration. An estimate of the Hessian of the Lagrangian is updated at each iteration using the BFGS formula (see fminu, references [3-6]).

A line search is performed using a merit function similar to that proposed by Han[1] and Powell[2,3]. The QP subproblem is solved using an active set strategy similar to that described in Gill, Murray, and Wright[4].

Limitations:

The function to be minimized and the constraints must be continuous. constr may only give local solutions.

When the problem is infeasible, constr attempt to minimize the maximum constraint value.

See also:

fminu, foptions

References:

[1] S.P. Han, A *Globally Convergent Method for Nonlinear Programming,* Journal of Optimization Theory and Applications, Vol. 22, 1977, p. 297.

[2] M.J.D. Powell, *The Convergence of Variable Metric Methods For Nonlinearly Constrained Optimization Calculations,* Nonlinear Programming 3, ed. O.L. Mangasarian, R.R. Meyer, and S.M. Robinson Academic Press, 1978.

[3] M.J.D. Powell, *A Fast Algorithm for Nonlineary Constrained Optimization Calculations*, Numerical Analysis, ed. G.A. Watson, Lecture Notes in Mathematics, Springer Varleg, Vol. 630, 1978.

[4]P.E. Gill, W. Murray, and M.H. Wright, *Practical Optimization,* Academic Press, London, 1981, p. 176-180.