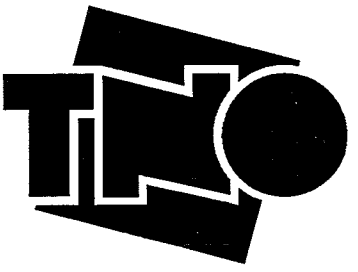
Implementatie van complexe gesloten ketens in MADYMO



Stageverslag

Implementatie van Complexe Gesloten Ketens in MADYMO.

P.G.M. Kruijt

Rapportno.: WFW 93.089

Stageverslag

Implementatie van

Complexe Gesloten Kinematische Ketens

in

MADYMO

Auteur: P.G.M. Kruijt
I.D.nummer: 260117

Opdrachtgevers: TNO
Instituut voor Wegtransportmiddelen,
Delft
&
Technische Universiteit Eindhoven
Faculteit Werktuigbouwkunde
Vakgroep Fundamentele Werktuigbouwkunde (WFW)
Eindhoven

Mentoren: dr ir W.P. Koppens (TNO-IW)
dr ir A.A.H.J. Sauren (TUE-WFW)

Rapportnummer: WFW 93.089

Datum: juni 1993

Samenvatting

In grote multibody-systemen komen vaak ketens voor die kinematisch gesloten zijn. In MADYMO is het modelleren van dergelijke multibody-systemen niet zonder meer mogelijk omdat MADYMO in principe geschreven is voor open ketens.
Er bestaat echter de mogelijkheid voor de gebruiker om zelf een kinematische verbinding te definiëren.
Het doel van deze stageopdracht is te bekijken of het mogelijk is gesloten ketens door een dergelijke kinematisch verbinding te programmeren. Dit wordt bekeken aan de hand van een vijfpunts voorwielophanging. Een extra moeilijkheid in dit geval is dat de posities, snelheden en versnellingen niet expliciet uit te drukken zijn in de graden van vrijheid (en hun eerste en tweede tijdsafgeleiden).
Het zal mogelijk blijken om in MADYMO een dergelijke gesloten kinematische keten te vervangen door een door de gebruiker te definiëren kinematische verbinding. Voorwaarde hierbij is wel dat de massa van de gemodelleerde verbinding te verwaarlozen is ten opzichte van de massa's van de lichamen die door deze verbinding verbonden worden.

Voorwoord

Dit rapport is geschreven in opdracht van het Instituut voor Wegtransportmiddelen van TNO Delft in samenwerking met de vakgroep WFW van de faculteit Werktuigbouwkunde, Technische Universiteit Eindhoven.

Het rapport is een weergave van de werkzaamheden en resultaten die in het kader van een stageopdracht hebben plaatsgevonden. De stage is uitgevoerd onder begeleiding van dr ir W.P. Koppens bij TNO Delft. Contactpersoon op de TU Eindhoven was dr ir A.A.H.J. Sauren.
Vanaf deze plaats wil ik hen bedanken voor de van hun ontvangen ondersteuning en adviezen.

Inhoudsopgave

Hoofdstuk 1

Inleiding

Deze stage is uitgevoerd in opdracht van TNO (Instituut voor Wegtransportmiddelen) naar aanleiding van een nieuwe versie van het multibody-pakket MADYMO dat door TNO ontwikkeld is. MADYMO is ontwikkeld voor de simulatie van het gedrag van het menselijk lichaam in geval van botsingen. MADYMO is mede daarom een multibody-pakket waarin in principe alleen open ketens gemodelleerd kunnen worden. Dit houdt in dat binnen een dergelijk open multibody-systeem geen ketens gevormd kunnen worden die een ononderbroken kring vormen (zie fig. 1 en 2). Hierdoor is het modelleren van gesloten ketens zoals bijvoorbeeld een kruk-drijfstangmechanisme niet zonder meer mogelijk.

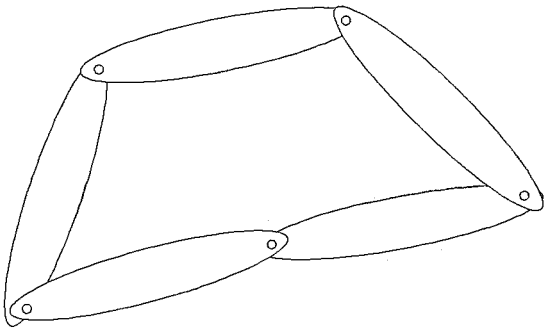


fig. 1 Voorbeeld van een gesloten kinematische keten.

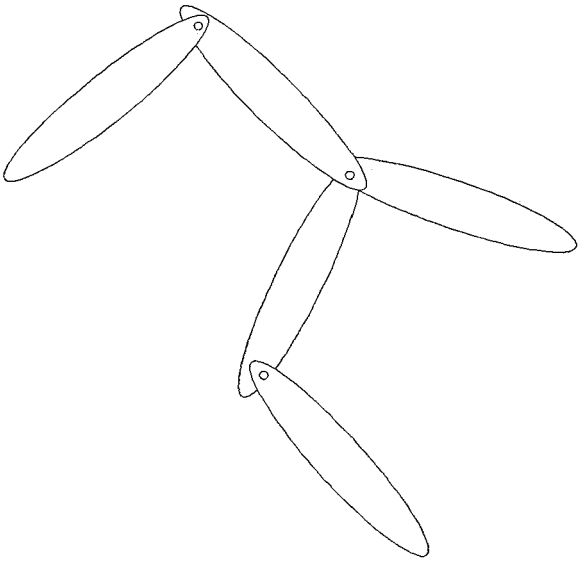


fig. 2 Voorbeeld van een open kinematische keten.

De gebruiker kan echter zelf een kinematische verbinding in MADYMO definiëren. Een aantal lichamen en verbindingen van een systeem waarin gesloten ketens voorkomen wordt vervangen door een nieuwe kinematische verbinding. Hierbij wordt de massa van de lichamen in deze verbinding verwaarloosd. Deze verbinding kan door de gebruiker met behulp van een subroutine gemodelleerd worden. Hij

krijgt daartoe een beperkt aantal (door hem zelf opgegeven) positionele vrijheidsgraden en hun eerste tijdsafgeleiden. Met behulp daarvan moet in een door hem te leveren subroutine de verdere kinematica (overige posities, snelheden en versnellingen) beschreven worden op een wijze die MADYMO kan verwerken.
Tot nu toe zijn alleen nog eenvoudige mechanismen geprogrammeerd waarvan de posities, snelheden en versnellingen expliciet uit te drukken waren in de vrijheidsgraden en hun eerste tijdsafgeleiden. Doel van deze stage is te bekijken in hoeverre het mogelijk is een systeem dat een gesloten keten bevat in MADYMO te modelleren, als van deze gesloten keten de posities, snelheden en versnellingen niet expliciet uit de vrijheidsgraden volgen. De gesloten keten wordt beschreven door middel van een zogenaamde user-joint. Aan de hand van een vijfpunts voorwielophanging zal dit worden onderzocht.
In hoofdstuk 2 komt de noodzakelijke theorie aan bod. In dit hoofdstuk wordt tevens een model afgeleid waaruit de betrekkingen afgeleid kunnen worden om de posities, snelheden en versnellingen uit te rekenen op een zodanige wijze dat ze in de USRJN3-routine van MADYMO gebruikt kunnen worden.
Hoofdstuk 3 handelt over de implementatie van de gevonden betrekkingen in de subroutine en behandelt de meest belangrijke onderdelen hieruit.
De verificatie van het geprogrammeerde model wordt behandeld in hoofdstuk 4.
In hoofdstuk 5 volgen uiteindelijk de conclusies en aanbevelingen met betrekking tot het gebruik van een user-joint en een korte behandeling van wat eventueel nog onderzocht kan worden.

Hoofdstuk 2

Kinematica

2.1 Inleiding

Om de kinematica van een lichaam of verbindingselement te kunnen beschrijven moet een methode gezocht worden om de positie en oriëntatie van een lichaam ten opzichte van een ander lichaam te beschrijven. Met de uitdrukkingen voor positie en oriëntatie zijn dan de eerste en tweede tijdsafgeleiden (snelheden en versnellingen) te bepalen waardoor het gehele kinematische gedrag vastligt en het dynamisch gedrag bepaald kan worden.
In deze paragraaf wordt de noodzakelijke theorie behandeld om de kinematica van een multibodysysteem te bepalen.

2.1.1 Kinematica van een verbindingselement

De kinematica van een verbindingselement tussen twee lichamen wordt beschreven door de positie en oriëntatie van lichaam j ten opzichte van lichaam i. Op de lichamen i en j bevinden zich resp. oorsprong O^i met assenstelsel $\underline{\xi}^i$ en oorsprong O^j met assenstelsel $\underline{\xi}^j$. Deze assenstelsels behoren toe aan het verbindingselement en hoeven niet samen te vallen met de assenstelsels van de lichamen i en j. De oorsprong van O^i wordt door de vector $\vec{d}^{ij}$ met de oorsprong O^j verbonden. Door de vector $\vec{d}^{ij}$ ligt de positie van O^j ten opzichte van O^i vast. Oorsprong O^j kan ook verdraaid zijn ten opzichte van de oorsprong O^i of $O^{i'}$ (de oorsprong $O^{i'}$ heeft de $\xi^{i'}$-, $\eta^{i'}$- en $\zeta^{i'}$-as

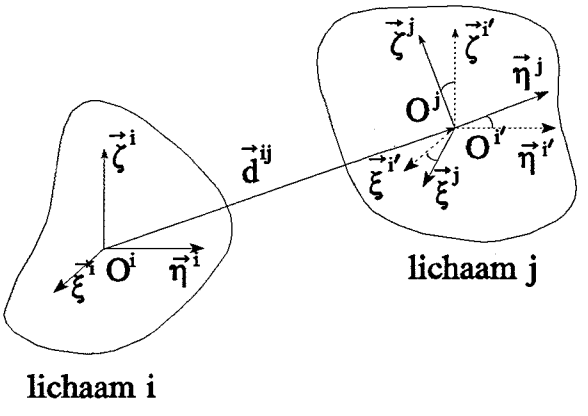


fig. 3 Beschrijving van de positie van lichaam j t.o.v. lichaam i.

evenwijdig liggen aan de ξ^i-, η^i- en ζ^i-as van oorsprong O^i). Assenstelsel $\underline{\xi}^j$ is dus geroteerd ten opzichte van $\underline{\xi}^i$ (of $\underline{\xi}^{i'}$). Deze rotatie is orthogonaal, dat wil zeggen dat het een zuivere rotatie is, zonder dat de lengte van de assen of de oriëntatie van de assen van het assenstelsel ten opzichte van elkaar gewijzigd wordt. De rotatie van assenstelsel $\underline{\xi}^j$ ten opzichte van assenstelsel $\underline{\xi}^i$ wordt als volgt beschreven met behulp van een rotatietensor $\boldsymbol{Q}^{ij}$:

$$\underline{\xi}^{jT} = \boldsymbol{Q}^{ij} \cdot \underline{\xi}^{iT} \tag{1}$$

Omdat $\boldsymbol{Q}^{ij}$ orthogonaal is geldt:

$$\boldsymbol{Q}^{ij} \cdot \boldsymbol{Q}^{ijc} = \boldsymbol{Q}^{ijc} \cdot \boldsymbol{Q}^{ij} = \boldsymbol{I} \tag{2}$$

Differentiatie van (1) naar de tijd levert:

$$\frac{\mathrm{d}}{\mathrm{dt}} \underline{\xi}^{jT} = \dot{\boldsymbol{Q}} \cdot \underline{\xi}^{iT} \tag{3}$$

Met gebruikmaking van (2) is dit te schrijven als:

$$\frac{\mathrm{d}}{\mathrm{dt}} \underline{\xi}^{jT} = \dot{\boldsymbol{Q}}^{ij} \cdot \boldsymbol{Q}^{ijc} \cdot \boldsymbol{Q}^{ij} \cdot \underline{\xi}^{iT} = \dot{\boldsymbol{Q}}^{ij} \cdot \boldsymbol{Q}^{ijc} \cdot \underline{\xi}^{jT} \tag{4}$$

De tensor $\boldsymbol{Q}^{ij}$ moet op elk tijdstip t voldoen aan de orthogonaliteitsrelatie (2). Door (2) naar de tijd te differentiëren wordt de volgende uitdrukking gevonden:

$$\dot{\boldsymbol{Q}}^{ij} \cdot \boldsymbol{Q}^{ijc} + \boldsymbol{Q}^{ij} \cdot \dot{\boldsymbol{Q}}^{ijc} = 0 \tag{5}$$

Hieruit volgt dat de tensor $\dot{\boldsymbol{Q}}^{ij} \cdot (\boldsymbol{Q}^{ij})^c$ scheefsymmetrisch is. Van elke scheefsymmetrische tensor $\boldsymbol{A}$ is het inwendig product van de tensor met een vector $\vec{a}$ te schrijven als een uitwendig product van de axiaalvector $\vec{w}$ van de tensor met de vector $\vec{a}$:

$$\boldsymbol{A} \cdot \vec{a} = \vec{w} \times \vec{a} \tag{6}$$

Hiermee is (4) te schrijven als:

$$\underline{\xi}^{jT} = \vec{\omega}^{ij} \times \underline{\xi}^{jT} \tag{7}$$

Hierin is $\vec{\omega}^{ij}$ de relatieve hoeksnelheidsvector van $\underline{\xi}^j$ ten opzichte van $\underline{\xi}^{i'}$. Omdat (7) voor de basis $\underline{\xi}^j$ geldt, geldt dit ook voor alle vectoren die opgespannen worden door

de basis $\underline{\xi}^j$.

Voor een willekeurige vector $\vec{b}^j$ blijkt dus te gelden:

$$\frac{d}{dt}\vec{b}^j = \vec{\omega}^{ij} \times \vec{b}^j \tag{8}$$

Voor de tweede tijdsafgeleide van een vector $\vec{b}^j$ blijkt op soortgelijke wijze de volgende betrekking af te leiden:

$$\frac{d^2}{dt^2}\vec{b}^j = \left(\frac{d}{dt}\vec{\omega}^{ij}\right) \times \vec{b}^j + \vec{\omega}^{ij} \times \left(\vec{\omega}^{ij} \times \vec{b}^j\right) \tag{9}$$

Van deze betrekkingen zal later dankbaar gebruik gemaakt worden.

2.1.2 Eulerparameters

Om de oriëntatie van een assenstelsel O^j ten opzichte van een assenstelsel O^i vast te leggen wordt de zogenaamde relatieve rotatietensor $\underline{Q}^{ij}$ gebruikt. De matrixrepresentatie hiervan bestaat uit negen elementen die onderling afhankelijk zijn. Om deze elementen te bepalen kan gebruik gemaakt worden van verschillende methoden. Een aantal hiervan zijn bijvoorbeeld de beschrijvingswijze met behulp van Eulerparameters, Eulerhoeken en Bryant- of Cardanhoeken (zie bijv. [2]). In dit verslag zal gebruik gemaakt worden van Eulerparameters.

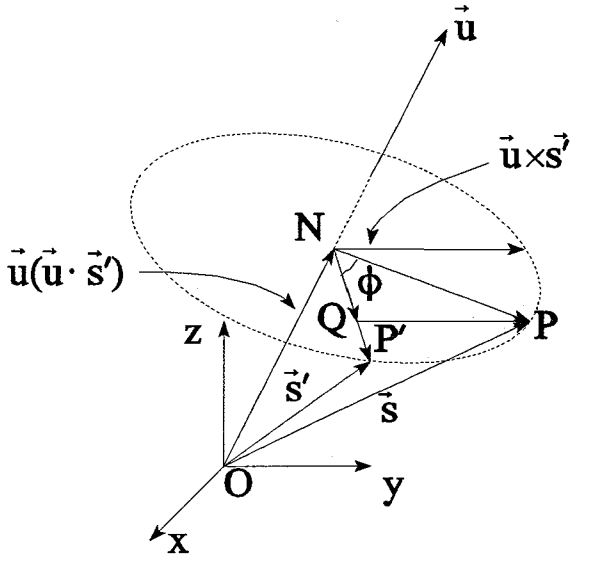


fig. 4 Vectordiagram voor de afleiding van Eulerparameters

Eulerparameters zijn gebaseerd op het theorema van Euler dat een coördinatentransformatie verkregen kan worden door een rotatie rond een geschikt gekozen as. Het spreekt voor zich dat in dat geval gezocht zal worden naar parameters die verband houden met de rotatie en de rotatie-as. Deze parameters zijn de rotatiehoek en de richtingscosinussen van de rotatie-as.

In fig. 4 wordt de initiële positie van een vector $\vec{s}$ gegeven door *OP* en de uiteindelij-

ke positie $\vec{s}''$ door OP'. De eenheidsvector langs de rotatie-as wordt weergegeven door $\vec{u}$. Vector $\vec{s}$ kan samengesteld worden uit drie andere vectoren:

$$\vec{s}=ON+NQ+QP \tag{10}$$

De afstand tussen de punten O en N is gelijk aan $\vec{u}\cdot\vec{s}''$, zodat vector ON geschreven kan worden als:

$$ON=\vec{u}(\vec{u}\cdot\vec{s}'') \tag{11}$$

Op een dergelijke wijze kan vector NP' als volgt geschreven worden:

$$NP'=\vec{s}''-ON=\vec{s}''-\vec{u}(\vec{u}\cdot\vec{s}'') \tag{12}$$

Hieruit volgt:

$$NQ=\{\vec{s}''-\vec{u}(\vec{u}\cdot\vec{s}'')\}\cos\phi \tag{13}$$

De grootte van vector NP' is gelijk aan die van de vectoren NP en $\vec{u}\times\vec{s}''$. Hierdoor kan vector QP uitgedrukt worden als:

$$QP=\vec{u}\times\vec{s}''\sin\phi \tag{14}$$

Substitutie van (11), (13) en (14) in (10) en samenvoegen van termen, leidt tot de rotatieformule:

$$\vec{s}=\vec{s}''\cos\phi+\vec{u}(\vec{u}\cdot\vec{s}'')(1-\cos\phi)+\vec{u}\times\vec{s}''\sin\phi \tag{15}$$

Met behulp van de volgende trigoniometrische identiteiten

$$\begin{aligned}\cos\phi &= 2\cos^2(\phi/2)-1\\ \sin\phi &= 2\sin(\phi/2)\cos(\phi/2)\\ 1-\cos\phi &= 2\sin^2(\phi/2)\end{aligned} \tag{16}$$

en de nieuwe parameters

$$\begin{aligned}e_0 &= \cos(\phi/2)\\ \vec{e} &= \vec{u}\sin(\phi/2)\end{aligned} \tag{17}$$

kan de rotatieformule (15) in de volgende, meer bruikbare, vorm gebracht worden:

$$\vec{s}=(2e_0^2-1)\vec{s}'+2\vec{e}(\vec{e}\cdot\vec{s}')+2e_0\vec{e}\times\vec{s}' \tag{18}$$

Herschrijven in kolomrepresentatie van $\vec{e}$ in $\underline{e}=[e_1,e_2,e_3]^T$ geeft:

$$\underline{s}=\left\{(2e_0^2-1)\underline{I}\right)+2\underline{e}\underline{e}^T+2e_0\tilde{\underline{e}}\Big\}\underline{s}' \tag{19}$$

Waarin $\underline{I}$ de eenheidsmatrix is en $\tilde{\underline{e}}$ gedefinieerd is volgens:

$$\tilde{\underline{e}} = \begin{pmatrix} 0 & -e_3 & e_2 \\ e_3 & 0 & -e_1 \\ -e_2 & e_1 & 0 \end{pmatrix} \tag{20}$$

De termen tussen vierkante haken vormen de gezochte matrixrepresentatie van de relatieve rotatietensor $\underline{Q}^{ij}$ uit (1). Uitschrijven van deze termen levert de volgende matrix $\underline{Q}^{ij}$:

$$\underline{Q}^{ij} = 2\begin{pmatrix} e_0^2+e_1^2-\frac{1}{2} & e_1e_2-e_0e_3 & e_1e_3+e_0e_2 \\ e_1e_2+e_0e_3 & e_0^2+e_2^2-\frac{1}{2} & e_2e_3-e_0e_1 \\ e_1e_3-e_0e_2 & e_2e_3-e_0e_1 & e_0^2+e_3^2-\frac{1}{2} \end{pmatrix} \tag{21}$$

De vier parameters e_0, e_1, e_2 en e_3 zijn de zogenaamde Eulerparameters.
Uit (16) volgt dat de vier Eulerparameters afhankelijk zijn. Aangezien geldt dat

$$\cos^2(\phi/2)+\underline{u}\underline{u}^T\sin^2(\phi/2)=1 \tag{22}$$

volgt met behulp van (17):

$$e_0^2+e_1^2+e_2^2+e_3^2=1 \tag{23}$$

Hiervan gebruik makend is $\underline{Q}^{ij}$ ook te schrijven als:

$$Q^{ij} = \begin{pmatrix} e_0^2+e_1^2-e_2^2-e_3^2 & 2(e_1e_2-e_0e_3) & 2(e_0e_2+e_1e_3) \\ 2(e_1e_2+e_0e_3) & e_0^2-e_1^2+e_2^2-e_3^2 & 2(e_2e_3-e_0e_1) \\ 2(e_1e_3-e_0e_2) & 2(e_2e_3+e_0e_1) & e_0^2-e_1^2-e_2^2+e_3^2 \end{pmatrix} \tag{24}$$

Het voordeel van Eulerparameters boven de andere genoemde methoden om de rotatietensor te beschrijven is dat vanwege de halve hoek ($\phi/2$) de sinus en cosinus samen de oriëntatie van een assenstelsel ten opzichte van een ander assenstelsel eenduidig vastleggen tussen $-\pi \leq \phi \leq \pi$.

2.2 Beschrijving van het probleem

In MADYMO moet de voorwielophanging van een auto gemodelleerd worden. De te modelleren voorwielophanging bestaat uit de volgende onderdelen:

- het chassis van de auto,
- de wieldrager,
- vijf stangen die het chassis met de wieldrager verbinden.

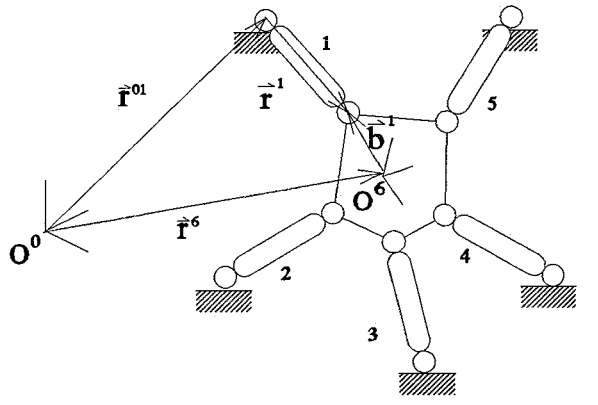


fig. 5 Schematische weergave van een voorwielophanging.

Daar MADYMO slechts geschikt is voor open ketens, geschiedde het modelleren van een dergelijk multibodysysteem in het verleden door het plaatsen van een zeer grote massa aan het 'vrije' uiteinde van een lichaam. Ook werden wel veerelementen gebruikt waarvan de stijfheid relatief zeer hoog was. Dit is in nieuwere versies niet meer noodzakelijk omdat de gebruiker met behulp van de subroutine USRJN3 zelf een eigen kinematische verbinding ter beschikking kan stellen.

Het schematisch model van de voorwielophanging wordt getoond in fig.5. Aan de wieldrager kan dan door MADYMO een 'wiel met band' toegevoegd worden. De gebruiker dient daartoe te beschrijven wat de posities, snelheden en versnellingen van de wieldrager ten opzichte van het chassis zijn.

2.3 Modellering

Voor de modellering van de stangen is gekozen voor bol-bol-scharnieren. Dit in plaats van stangen die door middel van bolscharnieren en kruisscharnieren met chassis en wieldrager verbonden zijn. Het voordeel hiervan is dat de kinematische verbindingsbetrekkingen, Φ^i, voor alle stangen dezelfde vorm hebben en dat de oriëntatie van de kruisscharnieren niet voor extra problemen zorgt.

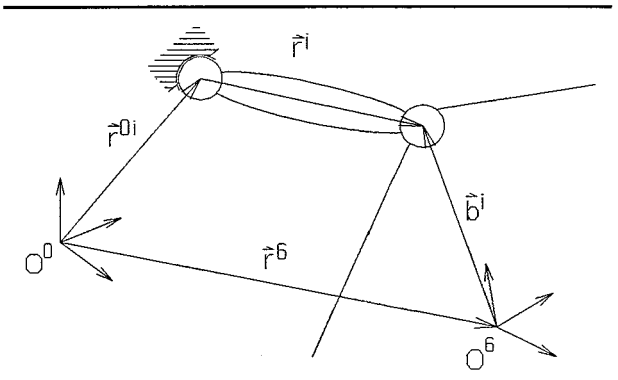


fig. 6 Eén bol-bol-scharnier uit de voorwielophanging.

Het gebruik van bol-bolscharnieren leidt ertoe dat er in principe slechts twee lichamen zijn (het chassis en de wieldrager) waarvan er één als vaste wereld wordt beschouwd (het chassis). Dit leidt tot zes graden van vrijheid tussen chassis en wieldrager. Aangezien elk bol-bol-scharnier één vrijheidsgraad vastlegt blijft er met vijf van deze scharnieren dus slechts één vrijheidsgraad over. Voor deze extra voor te schrijven graad van vrijheid is de verplaatsing in z-richting (r_3^6) gekozen.

2.4 Positieanalyse

Om de positie van een punt op de wieldrager vast te leggen moeten de kinematische verbindingsbetrekkingen opgesteld worden waaraan de gekozen scharnieren voldoen. De kinematische verbindingsbetrekking voor een bol-bol-scharnier luidt in het bovenstaande geval (zie [1]):

$$\Phi^i \equiv \vec{r}^{iT} \cdot \vec{r}^i - l^{i2} = 0 \tag{25}$$

Hierin is i het nummer van het betreffende bol-bolscharnier (i=1,...,5), $\vec{r}^i$ de vector die van begin- naar eindpunt van het bol-bolscharnier wijst en l^i de lengte van dat scharnier. De vector $\vec{r}^i$ wordt gegeven door (zie fig.6):

$$\vec{r}^i = \vec{r}^6 + \vec{b}^i - \vec{r}^{0i} \tag{26}$$

De vector $\vec{b}^i$ wordt in dit geval beschreven ten opzichte van het assenstelsel op het

chassis (O^0). Daar de oriëntatie en lengte van deze vector ongewijzigd blijven ten opzichte van de lokale basis van de wieldrager is de vector $\vec{b}^{\,i}$ te schrijven als een rotatie van de bijbehorende lokale vector $\vec{b}^{\,i\prime}$ (in het O^6-stelsel) ten opzichte van het O^0-stelsel door middel van een rotatietensor $\boldsymbol{Q}$ volgens (vgl. (1)):

$$\vec{b}^{\,i}=\boldsymbol{Q}\cdot\vec{b}^{\,i\prime} \tag{27}$$

Door substitutie van (26) en (27) in (25) wordt de volgende betrekking verkregen:

$$\Phi^{\,i}\equiv\vec{r}^{\,i}\cdot\vec{r}^{\,i}+\vec{b}^{\,i}\cdot\vec{b}^{\,i}+\vec{r}^{\,0i}\cdot\vec{r}^{\,0i}-\left(l^i\right)^2+2\left(\vec{r}^{\,6}\cdot\boldsymbol{Q}\cdot\vec{b}^{\,i\prime}-\vec{r}^{\,0i}\cdot\boldsymbol{Q}\cdot\vec{b}^{\,i\prime}-\vec{r}^{\,6}\cdot\vec{r}^{\,0i}\right)=0 \tag{28}$$

Hierbij is gebruik gemaakt van de betrekking $\boldsymbol{Q}^{\mathrm{T}}\cdot\boldsymbol{Q}=\boldsymbol{I}$ (zie o.a.[2]). Vergelijking (28) bevat 11 onbekenden: $r_1^{\,6}$, $r_2^{\,6}$ en de 9 elementen van de tensor $\boldsymbol{Q}$ ($r_3^{\,6}$ wordt voorgeschreven). Er zijn echter slechts vijf (onafhankelijke) vergelijkingen waardoor dit stelsel niet zonder meer opgelost kan worden. Het aantal onbekenden in de tensor $\boldsymbol{Q}$ kan tot vier worden teruggebracht indien de matrixrepresentatie $\boldsymbol{Q}$ van de rotatietensor $\boldsymbol{Q}$ beschreven wordt door gebruik te maken van de Eulerparameters e_0, e_1, e_2 en e_3 volgens (24).

Nu resteren nog slechts 6 onbekenden voor 5 vergelijkingen. De zesde vergelijking wordt geleverd door de onderlinge afhankelijkheid van de Eulerparameters (23):

$$\Phi^{\,6}\equiv e_0^2+e_1^2+e_2^2+e_3^2-1=0 \tag{29}$$

Op deze wijze ontstaat een stelsel van zes vergelijkingen (5 kinematische verbindingsbetrekkingen en de onderlinge afhankelijkheid van de Eulerparameters) met zes onbekenden ($r_1^{\,6}$, $r_2^{\,6}$ en de vier Eulerparameters). Dit stelsel vergelijkingen, $\Phi^i=0$ ($i=1,...,6$), is in principe op te lossen.

Uit (25) blijkt echter direct dat er geen stelsel vergelijkingen ontstaat dat lineair is in de onbekenden. Als voor de vijf scharnieren de verbindingsbetrekkingen ingevuld worden blijkt immers dat er sprake is van een inwendig product van een vector met zichzelf. Ook de afhankelijkheid van de Eulerparameters (vgl. (29)) is niet lineair.

2.4.1 Oplossing niet-lineaire vergelijkingen

Om dit stelsel niet-lineaire algebraïsche vergelijkingen op te lossen kan gebruik gemaakt worden van een iteratief algoritme. In dit geval is gekozen voor de oplos-

singsmethode volgens Newton-Raphson. Voor deze methode worden de onbekende parameters (r_1^6, r_2^6 en e_0 t/m e_3) in een kolomvector $\underline{q}$ gezet en wordt een matrix $\underline{\underline{A}}$ met partiële afgeleiden van de functies Φ^i naar deze onbekenden $\underline{q}$ berekend:

$$\underline{\underline{A}} \equiv \frac{\partial \underline{\Phi}}{\partial \underline{q}} = \underline{\Phi}_{,\underline{q}} \tag{30.a}$$

De methode komt in het kort hier op neer:

1. Neem een beginoplossing $\underline{q}_i$ (i=0) aan voor de onbekenden (30.b)
2. Bereken met deze oplossing een foutkolom $\underline{\varepsilon}_i = \underline{\Phi}(\underline{q}_i)$ (30.c)
3. Los de kolom $\Delta\underline{q}_i$ met aanpassingen voor de kolom $\underline{q}_i$ op uit $\underline{\underline{A}} \cdot \Delta\underline{q}_i = \underline{\varepsilon}_i$ (30.d)
4. Bereken een nieuwe kolom $\underline{q}_{i+1} = \underline{q}_i + \Delta\underline{q}_i$ en ga terug naar stap 2 (30.e)

Gestopt wordt als het aantal iteraties te hoog wordt (de oplossing convergeert dan niet of te langzaam) of als de norm van de foutkolom $\underline{\varepsilon}_i$ kleiner wordt dan een tevoren gekozen waarde (de oplossing is dan voldoende nauwkeurig).

In de eerste kolom van de bovenste vijf rijen van de matrix $\underline{\underline{A}}$ komen de partiële afgeleiden van de kinematische verbindingsbetrekkingen Φ^i (i=1,...5) naar q_1 (r_1^6):

$$\frac{\partial \Phi^i(\underline{q})}{\partial r_1^6} = 2r_1^6 + 2 \cdot \left\{ (1\ 0\ 0)\, \underline{\underline{Q}}\, \underline{b}^{i\prime} - r_1^{0i} \right\} \tag{31}$$

In de tweede kolom van de bovenste vijf rijen van de matrix $\underline{\underline{A}}$ komen de partiële afgeleiden van de kinematische verbindingsbetrekkingen Φ^i (i=1,...5) naar q_2 (r_2^6):

$$\frac{\partial \Phi^i(\underline{q})}{\partial r_2^6} = 2r_2^6 + 2 \cdot \left\{ (0\ 1\ 0)\, \underline{\underline{Q}}\, \underline{b}^{i\prime} - r_2^{0i} \right\} \tag{32}$$

In de derde kolom van de bovenste vijf rijen van de matrix $\underline{\underline{A}}$ komen de partiële afgeleiden van de kinematische verbindingsbetrekkingen Φ^i (i=1,...5) naar q_3 (e_0):

$$\frac{\partial \Phi^i(\underline{q})}{\partial e_0} = 4(\underline{r}^6 - \underline{r}^{0i}) \begin{pmatrix} e_0 & -e_3 & e_2 \\ e_3 & e_0 & -e_1 \\ -e_2 & e_1 & e_0 \end{pmatrix} \underline{b}^{i\prime} \tag{33}$$

In de vierde kolom van de bovenste vijf rijen van de matrix $\underline{\underline{A}}$ komen de partiële

afgeleiden van de kinematische verbindingsbetrekkingen Φ^i (i=1,...5) naar q_4 (e_1):

$$\frac{\partial \Phi^{i}(\underline{q})}{\partial e_1} = 4(\underline{r}^6 - \underline{r}^{0i}) \begin{pmatrix} e_1 & e_2 & e_3 \\ e_2 & -e_1 & -e_0 \\ e_3 & e_0 & -e_1 \end{pmatrix} \underline{b}^{i\prime} \tag{34}$$

In de vijfde kolom van de bovenste vijf rijen van de matrix $\underline{\underline{A}}$ komen de partiële afgeleiden van de kinematische verbindingsbetrekkingen Φ^i (i=1,...5) naar q_5 (e_2):

$$\frac{\partial \Phi^{i}(\underline{q})}{\partial e_2} = 4(\underline{r}^6 - \underline{r}^{0i}) \begin{pmatrix} -e_2 & e_1 & e_0 \\ e_1 & e_2 & e_3 \\ -e_0 & e_3 & -e_2 \end{pmatrix} \underline{b}^{i\prime} \tag{35}$$

In de zesde kolom van de bovenste vijf rijen van de matrix $\underline{\underline{A}}$ komen de partiële afgeleiden van de kinematische verbindingsbetrekkingen Φ^i (i=1,...5) naar q_6 (e_3):

$$\frac{\partial \Phi^{i}(\underline{q})}{\partial e_3} = 4(\underline{r}^6 - \underline{r}^{0i}) \begin{pmatrix} -e_3 & -e_0 & e_1 \\ e_0 & -e_3 & e_2 \\ e_1 & e_2 & e_3 \end{pmatrix} \underline{b}^{i\prime} \tag{36}$$

In de zesde rij komen de partiële afgeleiden van (29) naar de onbekende parameters te staan. Deze vergelijkingen zijn zeer eenvoudig; voor de eerste en tweede kolom luiden zij:

$$\frac{\partial \Phi^{6}}{\partial r_1^{6}} = \frac{\partial \Phi^{6}}{\partial r_2^{6}} = 0 \tag{37}$$

en voor de derde tot en met de zesde kolom (j=0,...,3):

$$\frac{\partial \Phi^{6}}{\partial e_j} = 2 \cdot e_j \tag{38}$$

Met behulp van deze afgeleiden kan de matrix $\underline{\underline{A}}$ gevuld worden zodat de onbekende parameters $\underline{q}$ bepaald kunnen worden. De rotatiematrix $\underline{\underline{Q}}$ kan vervolgens uit de Eulerparameters berekend worden met (24). Hiermee is de positie van de wieldrager ten opzichte van de oorsprong op het chassis beschreven ($\underline{r}^6$ en $\underline{\underline{Q}}$ zijn bekend).

2.5 Snelheidsanalyse

De snelheid van de wieldrager ten opzichte van het chassis kan beschreven worden door 2 snelheidsvectoren met ieder 3 componenten: de relatieve lineaire snelheidsvector ($\dot{\underline{r}}^6=(\dot{r}_1^6,\dot{r}_2^6,\dot{r}_3^6)^T$) en de relatieve hoeksnelheidsvector ($\underline{\omega}=(\omega_1,\omega_2,\omega_3)^T$). De derde component van de relatieve lineaire snelheidsvector ($\dot{r}_3^6$) wordt door MADYMO aan de subroutine doorgegeven.
Voor het berekenen van de snelheden wordt uitgegaan van de eerste tijdsafgeleide van de vijf kinematische verbindingsbetrekkingen volgens (25) (om typografische redenen wordt voortaan gebruik gemaakt van kolom- en matrixrepresentaties van resp. vectoren en tensoren ten opzichte van de basis O^0):

$$\dot{\Phi}^i = 2\underline{r}^{iT}\dot{\underline{r}}^i = 0 \tag{39}$$

waarin $\dot{\underline{r}}^i$ beschreven wordt door:

$$\dot{\underline{r}}^i = \dot{\underline{r}}^6 + \dot{\underline{b}}^i - \dot{\underline{r}}^{0i} = \dot{\underline{r}}^6 + \underline{\omega}\times\underline{b}^i \tag{40}$$

Met deze vijf (in de onbekenden lineaire) vergelijkingen zijn de onbekende snelheden vrij eenvoudig te bepalen. Door (40) in (39) te substitueren resulteert de volgende uitdrukking:

$$2\underline{r}^{iT}\dot{\underline{r}}^6 + 2\left(\underline{b}^i\times\underline{r}^i\right)^T\underline{\omega} = 0 \tag{41}$$

Hierbij is gebruik gemaakt van de volgende identiteit:

$$\underline{r}^i\cdot(\underline{\omega}\times\underline{b}^i) = (\underline{b}^i\times\underline{r}^i)\cdot\underline{\omega} \tag{42}$$

Vervolgens wordt vergelijking (41) uitgeschreven en opgesplitst in bekende en onbekende termen. Uiteindelijk volgt dan de volgende matrixvergelijking:

$$2\begin{pmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ r_1^i & r_2^i & b_2^i r_3^i - b_3^i r_2^i & b_3^i r_1^i - b_1^i r_3^i & b_1^i r_2^i - b_2^i r_1^i \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \cdot \begin{pmatrix} \dot{r}_1^6 \\ \dot{r}_2^6 \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix} = -2 \begin{pmatrix} \vdots \\ \vdots \\ r_3^i \\ \vdots \\ \vdots \end{pmatrix} \dot{r}_3^6 \tag{43}$$

Uit deze betrekking zijn door middel van matrixinversie of LU-decompositie in combinatie met terugsubstitutie de nog onbekende snelheden te bepalen.

2.6 Versnellingsanalyse

De relatieve lineaire- en hoekversnellingen van de wieldrager ten opzichte van het chassis kunnen bepaald worden uit de tweede tijdsafgeleide van de kinematische verbindingsbetrekkingen (25). MADYMO verwacht echter dat deze versnellingen worden opgesplitst in twee aparte stukken: een stuk dat direct afhankelijk is van de tweede tijdsafgeleide van de gekozen vrijheidsgraad (hier $\ddot{r}_3^6$) en een stuk dat daar onafhankelijk van is.

De enige data die vanuit MADYMO verkregen worden zijn de gekozen graad van vrijheid $\underline{q}$ en de daarbij behorende eerste tijdsafgeleide $\dot{\underline{q}}$ (hier resp. r_3^6 en $\dot{r}_3^6$). Hierdoor is het niet mogelijk de versnellingen te bepalen via LU-decompositie omdat het rechterlid bestaat uit een deel dat niet afhangt van de tweede tijdsafgeleide van de gekozen vrijheidsgraad en een deel dat daar wel van afhangt. Dit is op te lossen door hetzelfde te werk te gaan als bij de snelheidsanalyse, maar nu het rechterlid op te delen in twee stukken: een stuk onafhankelijk van de tweede tijdsafgeleide van de gekozen vrijheidsgraad en een stuk dat daar wel van afhangt.

De tweede tijdsafgeleide van de verbindingsbetrekkingen luidt:

$$\ddot{\Phi}^{\,i} = 2\left(\dot{\underline{r}}^{iT}\dot{\underline{r}}^{\,i} + \underline{r}^{iT}\ddot{\underline{r}}^{\,i}\right) = 0 \tag{44}$$

Waarbij gebruik is gemaakt van:

$$\underline{\ddot{r}}^i = \underline{\ddot{r}}^6 + \underline{\omega}\times\underline{\dot{b}}^i + \underline{\dot{\omega}}\times\underline{b}^i = \underline{\ddot{r}}^i + \underline{\omega}\times(\underline{\omega}\times\underline{b}^i) + \underline{\dot{\omega}}\times\underline{b}^i \tag{45}$$

Substitutie van (45) in (44) leidt tot de volgende uitdrukking:

$$2\underline{r}^{i\,T}\underline{\ddot{r}}^6 + 2\left(\underline{b}^i\times\underline{r}^i\right)\underline{\dot{\omega}} = -2\left(\underline{\dot{r}}^{i\,T}\underline{\dot{r}}^i + \underline{r}^i\left(\underline{\omega}\times\left(\underline{\omega}\times\underline{b}^i\right)\right)\right) \tag{46}$$

Als het linkerlid van deze vergelijking wordt vergeleken met het linkerlid van vergelijking (41) dan valt op dat dit nagenoeg dezelfde structuur heeft: slechts de onbekenden zijn van een hogere afgeleide, de factoren zijn identiek. Het zal daarom duidelijk zijn dat na herschrijven van (46) dezelfde uitdrukking volgt als bij (43) maar dat het rechterlid een term meer bevat:

$$2\begin{pmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ r_1^i & r_2^i & b_2^i r_3^i - b_3^i r_2^i & b_3^i r_1^i - b_1^i r_3^i & b_1^i r_2^i - b_2^i r_1^i \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \cdot \begin{pmatrix} \ddot{r}_1^6 \\ \ddot{r}_2^6 \\ \dot{\omega}_1 \\ \dot{\omega}_2 \\ \dot{\omega}_3 \end{pmatrix} = -2\begin{pmatrix} \vdots \\ \vdots \\ r_3^i \\ \vdots \\ \vdots \end{pmatrix}\ddot{r}_3^6 - 2\begin{pmatrix} \vdots \\ \vdots \\ v^i \\ \vdots \\ \vdots \end{pmatrix} \tag{47}$$

waarin v^i gegeven wordt door:

$$v^i = \underline{\dot{r}}^{i\,T}\underline{\dot{r}}^i + \underline{r}^i\left(\underline{\omega}\times\left(\underline{\omega}\times\underline{b}^i\right)\right) \tag{48}$$

Door nu de matrix uit (47) te inverteren en deze geïnverteerde matrix na te vermenigvuldigen met de twee termen uit het rechterlid ontstaan de vergelijkingen voor de onbekende versnellingen in termen afhankelijk en onafhankelijk van de tweede tijdsafgeleide van de gekozen graad van vrijheid ($\ddot{r}_3^6$). Als deze matrix gelijkgesteld wordt aan $\underline{\underline{T}}$ (met inverse $\underline{\underline{T}}^{-1}$) dan resulteert het volgende stelsel vergelijkingen:

$$\begin{pmatrix} \ddot{r}_1^6 \\ \ddot{r}_2^6 \\ \dot{\omega}_1 \\ \dot{\omega}_2 \\ \dot{\omega}_3 \end{pmatrix} = -2\underline{T}^{-1} \cdot \begin{pmatrix} r_3^1 \\ r_3^2 \\ r_3^3 \\ r_3^4 \\ r_3^5 \end{pmatrix} \cdot \ddot{r}_3^6 - 2\underline{T}^{-1} \cdot \begin{pmatrix} v^1 \\ v^2 \\ v^3 \\ v^4 \\ v^5 \end{pmatrix} \tag{49}$$

In deze serie vergelijkingen moet nu nog een zesde (zeer triviale) vergelijking toegevoegd worden om de zes vergelijkingen voor de twee maal drie componenten van de lineaire- en hoekversnellingen te completeren, nl.:

$$\ddot{r}_3^6 = 1 \cdot \ddot{r}_3^6 + 0 \tag{50}$$

Hiermee zijn uiteindelijk alle onbekenden berekend die MADYMO nodig heeft, wanneer de gebruiker zelf een verbindingselement wil definiëren dat niet onder de standaardverbindingselementen (joints) van MADYMO valt. Op deze wijze zijn dan ook gesloten ketens te definiëren wat anders niet of slechts in beperkte mate mogelijk is.

Hoofdstuk 3

Implementatie

3.1 Inleiding

MADYMO is geschreven in Fortran 77. Dit houdt bijna vanzelfsprekend in dat de toe te voegen subroutine ook in deze taal geschreven moet worden. Om compatibiliteit tussen verschillende compilers te waarborgen, is gebruik gemaakt van de ANSI-standaard (ANSI-Fortran X3.9-1978). Bij het schrijven van de subroutine is geen gebruik gemaakt van standaard-bibliotheken waarvan de bronteksten niet voor de gebruiker beschikbaar zijn (zoals de NAG Workstation Library). Dit is onder andere gedaan om ervoor te zorgen dat de routine gebruikt kan worden op elke computer waar MADYMO geïnstalleerd is. Doordat gebruik gemaakt is van bibliotheken waarvan de bronteksten wel voor de gebruiker beschikbaar zijn, is de routine ook eenvoudiger te begrijpen: alles wat noodzakelijk is voor de berekeningen in de routine bevindt zich in dezelfde brontekst.

3.2 Variabelen in USRJN3

Vanuit MADYMO 3D worden aan de subroutine USRJN3 de gekozen vrijheidsgraden en hun eerste tijdsafgeleides doorgegeven. De gebruiker dient daaruit verschillende kolommen en matrices te berekenen die MADYMO nodig heeft om resulterende snelheden en versnellingen (en daarmee krachten en momenten) uit te rekenen op de lichamen die USRJN3 met elkaar verbindt. De coördinaatsystemen van de elementen mogen willekeurig gekozen worden, waarbij er echter wel rekening mee gehouden moet worden dat positievector, rotatiematrix etc. gegeven worden vanuit het lager genummerde lichaam naar het hoger genummerde lichaam.
De data die de gebruiker vanuit MADYMO 3D tot zijn beschikking krijgt, bevinden zich in de volgende twee variabelen:

Q	n×1 kolom met de actuele waarden van de n gekozen graden van vrijheid $\underline{q}$ (in dit geval r_3^6),
QT	n×1 kolom met de actuele waarden van de eerste tijdsafgeleide van de n gekozen graden van vrijheid $\underline{q}$ ($\dot{\underline{q}}=\dot{r}_3^6$).

De data die de gebruiker zelf moet laten berekenen door de subroutine, moet in de volgende matrices en kolommen gezet worden:

SD	3×1 kolom waarin de componenten van de relatieve positievector $\underline{d}^{ij}$ (in dit geval $\underline{r}^6$) moeten worden opgeslagen,
SDT	3×1 kolom waarin de componenten van de relatieve snelheidsvector $\dot{\underline{d}}^{ij}$ (hier $\underline{\dot{r}}^6$) moeten worden opgeslagen,
SDTT	3×1 kolom waarin dat gedeelte van de relatieve versnellingsvector $\ddot{\underline{d}}^{ij}$ opgeslagen moet worden, dat niet afhangt van de tweede tijdsafgeleide van de gekozen vrijheidsgraden $\underline{q}$ ($\ddot{\underline{q}}$),
SDQ	3×n matrix die gevuld moet worden met de coëfficiëntmatrix van de tweede tijdsafgeleide van de n vrijheidsgraden $\underline{q}$ ($\ddot{\underline{q}}$) in de uitdrukking voor de relatieve versnelling,
CD	3×3 matrix die gevuld moet worden met de componenten van de relatieve rotatiematrix Q^{ij} (in dit geval de matrix Q),
CDT	3×1 kolom waarin de componenten van de relatieve hoeksnelheidsvector $\underline{\omega}^{ij}$ (in dit geval ω) ten opzichte van element i moeten worden opgeslagen,
CDTT	3×1 kolom waarin het gedeelte van de relatieve versnellingsvector $\dot{\underline{\omega}}^{ij}$ moet worden opgeslagen, dat niet afhangt van de tweede tijdsafgeleide van de vrijheidsgraden,
CDQ	3×n matrix waarin de coëfficiëntmatrix van de tweede tijdsafgeleide van de vrijheidsgraden $\ddot{\underline{q}}$ in de uitdrukking voor de relatieve hoekversnelling moet worden opgeslagen.

In dit geval is er voor de verbinding slechts één graad van vrijheid, namelijk de z-verplaatsing van het hoger genummerde lichaam. Daarom zijn Q en QT kolommen met lengte 1 en hebben de matrices SDQ en CDQ dimensie 3×1.

3.3 Initialisatie nulstand

Om voor elke willekeurige z-waarde (r_3^6) van de wieldrager de volledige positie te berekenen, wordt uitgegaan van de nulstand. Dit is de stand waarin de x- en z-coördinaat van het wielmidden gelijk zijn aan nul met een referentie-assenstelsel op het chassis volgens DIN 70000. In de nulstand zijn alleen de coördinaten van de diverse punten op het chassis en de wieldrager gegeven. Hieruit worden eerst de vectoren berekend die de van belang zijnde punten ten opzichte van elkaar beschrijven. Dit gebeurt in de regels 52-125 (zie 'Subroutine USRJN3' in Appendix A.2). De vectoren $\vec{r}^{0i}$ (i=0,...,5) leggen de punten P_iX vast waar de bol-bol-scharnieren aan het chassis gekoppeld zijn. De vectoren $\vec{r}^{i}$ leggen de positie van de andere zijde van het scharnier in een punt X_i op de wieldrager vast ten opzichte van het corresponderende punt P_iX op het chassis. De vector $\vec{r}^{6}$ beschrijft de positie van het wielmidden ten opzichte van het referentieframe en de vectoren $\vec{b}^{i}$ beschrijven de posities van de punten X_i ten opzichte van het wielmidden in een locaal coördinaatstelsel.
De punten P_iX en X_i worden weergegeven in tabel 1.

Tabel 1: Coördinaten van de bevestigingspunten op het chassis (P_iX) en de wieldrager (X_i)

Punt	x	y	z	Punt	x	y	z
P_1X	-0,0640	0,4130	0,3270	X_1	-0,0640	0,6360	0,3450
P_2X	-0,3030	0,4320	0,2950	X_2	-0,1170	0,6360	0,3380
P_3X	-0,0930	0,3660	0,0040	X_3	-0,1880	0,6470	-0,0230
P_4X	-0,2360	0,3880	-0,1090	X_4	-0,0050	0,7370	-0,1300
P_5X	0,2115	0,3845	-0,1000	X_5	-0,0025	0,7370	-0,1340
B_1	0,0000	0,7680	0,0000				

In principe hoeft het berekenen van de verschillende positievectoren niet in de USRJN3-routine te gebeuren. Deze berekeningen kunnen eenmalig gedaan worden in de subroutine USRIN3 en met behulp van een zogenaamd 'COMMON-block' (zie o.a. [3] en [4]) overgedragen worden aan de subroutine USRJN3. Het berekenen van de positievectoren in USRIN3 heeft nog een voordeel waarop later in dit verslag teruggekomen zal worden. Toch zijn deze berekeningen uitgevoerd binnen de subroutine USRJN3 om de volledige brontekst bij elkaar te houden.

3.4 Berekening positievector en rotatiematrix

De onbekenden in de posities zijn de eerste twee componenten van de vector $\underline{r}^6$ en de relatieve rotatiematrix $\underline{Q}$. Zoals volgt uit (25) en (29), zijn de vergelijkingen waaruit deze onbekenden op te lossen zijn, niet lineair in de onbekenden. Dit leidt tot het toepassen van een iteratieve methode om numerieke waarden voor de onbekenden te vinden. Gekozen is voor de methode volgens Newton-Raphson (regel 128-180). Voor deze methode zijn de kinematische verbindingsbetrekkingen en de afhankelijkheid van de Eulerparameters en hun partiële afgeleiden naar de onbekenden nodig.

Omdat de vijf kinematische verbindingsbetrekkingen dezelfde vorm hebben, kunnen zij met dezelfde subroutine berekend worden waaraan telkens de juiste parameters worden meegeven. De i^e kinematische verbindingsbetrekking ($i=1,...,5$) wordt berekend volgens (25) en (26) in de functie CALCFI (regel 393-408). In de subroutine CALCF (regel 409-427) wordt CALCFI voor elke verbindingsbetrekking aangeroepen om de fout te berekenen. De afhankelijkheid van de Eulerparameters wordt in regel 425 toegevoegd. Voor het berekenen van de coëfficiënten uit de matrix $\underline{A}$ (vgl. (30.a)) zijn de partiële afgeleiden van de verbindingsbetrekkingen naar de onbekenden nodig. Ook deze hebben dezelfde vorm per partiële afgeleide en kunnen dus met eenzelfde functie-aanroep berekend worden. De partiële afgeleiden worden volgens vergelijking (31) tot en met (37) berekend in de functies DFIDQ1 tot en met DFIDQ6 (regel 313-392). De subroutine CDFDQ (regel 428-500) roept deze functies beurtelings aan en vult de matrix met de juiste waarden. CDFDQ vult zelf de partiële afgeleiden van (29) (vgl^n (37) en (38)) naar de onbekenden in (in de regels 448, 458, 468, 478, 488 en 498). De aldus berekende foutkolom $\underline{\varepsilon}$ en matrix $\underline{A}$ worden gebruikt om een kolom met aanpassingen $\Delta\underline{q}$ op de geschatte waarden voor de onbekenden te berekenen. Hiervoor dient het stelsel vergelijkingen $\underline{A}\cdot\Delta\underline{q}=\underline{\varepsilon}$ (vgl (30.d)) opgelost te worden. Dit wordt gedaan met behulp van LU-decompositie en terugsubstitutie in de subroutines LUDCMP en LUBKSB (resp. regel 635-721 en 722-762) afkomstig uit [5]. Na het aanpassen van de onbekenden (regel 164-189) wordt de Euclidische norm van $\Delta\underline{q}$ bepaald om na te gaan of de oplossing reeds voldoende is geconvergeerd. Verder wordt nagegaan of het maximaal aantal iteraties verstreken is en eventueel een foutmelding gegeven (regel 170-177).

3.5 Berekening snelheden en versnellingen

Een groot deel van de snelheden en versnellingen wordt beschreven door dezelfde matrix: de matrix T uit vgl. (49). De enige verschillen zijn de rechterleden en de kolom met onbekenden. Als de rechterleden nader beschouwd worden blijkt dat het rechterlid uit (43) op een factor $\ddot{r}_3^6/\dot{r}_3^6$ na gelijk is aan dat deel van het rechterlid van (47) dat afhankelijk is van de tweede tijdsafgeleide van de vrijheidsgraad.
Omdat de matrix in beide gevallen hetzelfde is en het rechterlid op een factor na gelijk, hoeven deze maar één keer berekend te worden (regel 193-238).
Normaliter is (43) het eenvoudigst op te lossen door middel van LU-decompositie, maar omdat voor het oplossen van (47) matrixinversie noodzakelijk is en het dezelfde matrix betreft, wordt de oplossing van (26) berekend door de matrix te inverteren. De matrixinversie vindt plaats in de routine INVMAT (regel 591-621). Hierbij wordt vijf maal gebruik gemaakt van LU-decompositie, waarbij het rechterlid slechts gevuld is met een 1 die telkens een plaats opschuift.
Door de inverse matrix na te vermenigvuldigen met het rechterlid wordt de oplossing bepaald. Hierbij dient men er echter rekening mee te houden dat de factor $\ddot{r}_3^6/\dot{r}_3^6$ nog niet verdisconteerd is. Men heeft dus het afhankelijke deel uitgerekend van (47). Om de onbekenden uit (43) te bepalen dient nog vermenigvuldigd te worden met $\dot{r}_3^6$.
Hiermee zijn in een keer zowel de snelheden als de, van de tweede tijdsafgeleide van de vrijheidsgraad afhangende versnellingen, berekend. De enige onbekenden die nog berekend moeten worden zijn de versnellingen die niet afhangen van de tweede tijdsafgeleide van de vrijheidsgraad. Daartoe wordt het tweede gedeelte van het rechterlid van (47) uitgerekend en voorvermenigvuldigd met de inverse van de matrix ($\underline{T}^{-1}$). Omdat de coëfficiënten van het rechterlid berekend worden uit dezelfde vergelijking (48), is ook hiervoor een functie geschreven (CRHSI; regel 512-544).
Hiermee zijn alle onbekenden berekend die MADYMO verlangt om een door de gebruiker gedefinieerd verbindingselement toe te passen in het programma.

Hoofdstuk 4

Verificatie

4.1 Inleiding

Geen programma (of programmaonderdeel) wordt geschreven zonder dat er fouten gemaakt worden. Programmeerfouten komen al snel boven water als de brontekst gecompileerd wordt; hele reeksen foutmeldingen zijn het gevolg. Als deze fouten uit het programma verwijderd zijn, blijft er nog een hele reeks fouten over: intentiefouten. Intentiefouten zijn fouten in het implementeren van formules in het programma: er wordt een minteken vergeten of er worden twee verkeerde variabelen met elkaar vermenigvuldigd. Deze fouten leveren geen foutmeldingen van de compiler en zijn dus een stuk moeilijker te vinden.
Om na te gaan of de subroutine USRJN3 ook daadwerkelijk de vijfpuntswielophanging beschrijft is gebruik gemaakt van een aantal methoden. Eén daarvan kan plaatsvinden zonder hulp van MADYMO. MADYMO zelf biedt ook twee eenvoudige mogelijkheden om na te gaan of er bij het programmeren van de subroutine USRJN3 intentiefouten gemaakt zijn.

4.2 Verificatie zonder MADYMO

De subroutine USRJN3 is allereerst geverifieerd zonder gebruik te maken van MADYMO. In plaats daarvan is gebruik gemaakt van een tweede Fortran-programma (PVAGEN; zie appendix A.1) in combinatie met PC-Matlab. Het programma PVAGEN voorziet USRJN3 van de noodzakelijke waarden voor plaats en snelheid in z-richting. USRJN3 berekent de daarbij horende matrices en kolommen zoals hiervoor beschreven. PVAGEN stelt uit de afhankelijke en onafhankelijke delen van de versnelling de versnellingscomponenten samen en voert alle posities, snelheden en versnellingen uit naar het beeldscherm of een data-file.

Deze data-file kan worden ingelezen in PC-Matlab. In Matlab is deze datafile opgedeeld in kolommen waarin zich de afzonderlijke positie-, snelheids- en versnellingscomponenten bevinden.
De verificatie van de posities heeft visueel plaatsgevonden: van de geprogrammeerde wielophanging waren grafieken beschikbaar waarin de x- en y- verplaatsing uitgezet waren tegen de z-verplaatsing van het wiel. Deze bleken zeer goed overeen te komen met hetgeen door de subroutine berekend was.
De verificatie van de lineaire snelheden en versnellingen is numeriek gebeurd. In Matlab zijn met behulp van de 'gradient' functie de numerieke afgeleiden van de plaats en de snelheid naar de tijd bepaald. De functie 'gradient' maakt hierbij gebruik van een centraal differentieschema (behalve in begin- en eindpunt). De aldus door Matlab berekende waarden voor de snelheid en de versnelling zijn vergeleken met de waarden die door USRJN3 berekend waren. Afwijkingen tussen Matlab en USRJN3 bleken in de orde van 1% te liggen. Deze afwijkingen zijn te wijten aan Matlab omdat in de grafieken een aantal steile hellingen zitten. Bij numerieke bepaling van de afgeleiden zijn deze van grote invloed op de gemaakte fout.
De verificatie van de hoekversnellingen is op dezelfde wijze gedaan; deze zijn door Matlab uit de hoeksnelheden berekend en vergeleken met de hoekversnellingen die PVAGEN uitgevoerd had. De hoeksnelheden zelf zijn niet geverifieerd. Dit komt doordat de hoeksnelheden bepaald zouden moeten worden uit de rotatiematrix $\underline{Q}$. Daar Matlab een reeks scalars opslaat als een vector en een reeks vectoren als een matrix zou een reeks matrices opgeslagen moeten worden als een '3D'-matrix. PC-Matlab biedt hiertoe geen eenvoudige mogelijkheid.

4.3 Verificatie met behulp van MADYMO

Madymo biedt de gebruiker de mogelijkheid om extra informatie met betrekking tot de verschillende deelsystemen van een multibody-systeem uit te voeren in een zogenaamde debug-file. In deze debug-file staan onder andere de posities en snelheden van een lichaam dat aan een verbindingselement verbonden is. Deze zijn echter door USRJN3 al berekend.
Wat van groter belang is, is de kinetische energie van het systeem. In het geval dat er zich in het systeem slechts lichamen en verbindindingselementen bevinden, wordt er

geen energie gedissipeerd. Als de zwaartekracht nu 'uitgezet' wordt en er een beginsnelheid aan het systeem opgelegd wordt, dan moet de totale kinetische energie van het systeem constant blijven. Aangezien in de debug-file op een aantal discrete tijdstippen de kinetische energie vastgelegd is, is dit eenvoudig te controleren. Controle van de debug-file levert inderdaad dat de kinetische energie voor het systeem constant is (1,92 J).

Een andere mogelijkheid is het aanbrengen van veren (in MADYMO Kelvin-elementen) tussen de twee lichamen die door het verbindingselement verbonden worden. Als dit gebeurt op de plaatsen waar de stangen het ene lichaam met het andere verbinden, dan zou de rek van de veren nul moeten zijn, aangezien de stangen in het model star zijn verondersteld. De verlenging van de verschillende veren kan afgelezen worden in de file RELONG. Uitvoer van deze file leert dat de grootste rek van de ordegrootte 10^{-8} is. Aangezien dit ook de nauwkeurigheid is waarmee de positievector, $\underline{r}^6$, en de relatieve rotatiematrix, Q, berekend worden, mag dit verwacht worden.

Hoofdstuk 5

Conclusies en aanbevelingen

5.1 Modellering van gesloten ketens

Met behulp van de routine USRJN3 blijkt het binnen MADYMO inderdaad mogelijk om gesloten kinematische ketens te modelleren waarvan de posities, snelheden en versnellingen niet expliciet zijn uit te drukken in de gekozen vrijheidsgraden en eerste en tweede tijdsafgeleiden daarvan. Omdat de verbindingsvergelijkingen in het algemeen niet meer lineair zullen zijn, dient gebruik gemaakt te worden van een iteratieve methode om de posities te berekenen.

5.2 Veren en dempers

De userjoint beschrijft in de huidige situatie slechts de kinematica van een vijfpunts voorwielophanging (op voorwaarde dat de massa's van de stangen en de wieldrager verwaarloosd kunnen worden). In de werkelijke wielophanging bevinden zich behalve de stangen en de wieldrager ook nog een veer en een demper. Deze zijn met het ene uiteinde verbonden aan het chassis en met het andere aan stang 5. De veer en demper kunnen niet in de userjoint-routine gemodelleerd worden omdat deze routine slechts de kinematica van een verbindindingselement beschrijft.
De veer en demper kunnen ook niet met een uiteinde aan het chassis gehangen worden en met het andere in de userjoint omdat MADYMO de stangen in de userjoint niet 'ziet' (de plaats van de stangen is alleen in de userjoint-routine bekend). Alleen de positie en oriëntatie van het hoger genummerde lichaam ten opzichte van het lager genummerde lichaam wordt door de routine beschreven.
De veer- en dempingsparameters van de originele veer en demper moeten dus omgerekend worden naar veer- en dempingsparameters voor een veer en demper die aan het uiteinde van het verbindingselement bevestigd ziten. De resulterende krachten en momenten op de wieldrager zullen in dat geval niet exact kloppen met

de originele situatie. Dit komt doordat de beweging van het uiteinde van het verbindingselement niet precies gelijk is aan de beweging van stang 5.
Dit probleem kan misschien verholpen worden door gebruik te maken van de userforce-routine in combimatie met een COMMON-blok. In het COMMON-blok worden dan alle van belang zijnde waarden, zoals plaats en snelheid van stang 5 ten opzichte van het bevestigingspunt op de carrosserie, overgedragen. In de userforce-routine kunnen dan aan de hand van de resulterende krachten en momenten op stang 5 de krachtene en momenten op wiel en carrosserie berekend worden.

5.3 Massaloosheid

De, door middel van de routine USRJN3 gemodelleerde, gesloten ketens hebben het nadeel dat zij massaloos zijn. Als de massa's van de stangen in verhouding tot chassis en wieldrager met wiel niet klein genoeg zijn, dan zal de op deze wijze berekende dynamica van het systeem sterk afwijken van de realiteit. Zijn de massa's van de stangen echter klein genoeg dan mag ervan worden uitgegaan dat de simulatie redelijk overeenkomt met de werkelijkheid.
Een mogelijkheid om de dynamica voor relatief grote massa's toch in orde te krijgen wordt misschien geboden door de subroutine USRFO3. In USRFO3 kunnen krachten en momenten voorgeschreven worden die op een lichaam of verbindingselement werken. Door nu vanuit deze routine de routine USRJN3 aan te roepen kan de massatraagheid van het systeem en eventueel aanwezige veren en dempers in rekening gebracht worden. Een fout hierin wordt veroorzaakt doordat MADYMO de versnellingen (en daarmee snelheden en posities) berekend aan de hand van de krachten en momenten die op een lichaam werken. Op de geschetste wijze gebeurt het tegengestelde: de krachten en momenten worden berekend aan de hand van de versnellingen, snelheden en posities. De routine USRFO3 loopt dus een tijdstap achter op het systeem. Als de tijdstappen klein genoeg gekozen worden zou dit echter een redelijke benadering kunnen geven.
Een andere mogelijkheid wordt geboden door massa's te lumpen aan aan de uiteinden van de userjoint. Dit heeft echter tot gevolg dat de massatraagheid ten gevolge van de traagheidsmomenten van de stangen verwaarloosd wordt, waardoor de gevonden traagheidskrachten altijd lager zullen zijn dan de werkelijk optredende traagheidskrachten.

5.4 Hoeveelheid user-joints

Het aantal kinematische verbindingen dat de gebruiker in MADYMO in kan geven is beperkt tot 1. Dit houdt in dat als men, uitgaande van dit geval, zowel de voor- als achterwielophanging van een auto wil modelleren, en deze wezenlijk verschillen, dit niet zonder meer mogelijk is. Een truc om dit toch mogelijk te maken, is door binnen de USRJN3-routine een teller te gebruiken die bijhoudt hoe vaak de routine wordt aangesproken. Alle verbindingen worden namelijk in volgorde doorlopen en door een teller bij te houden weet men dan welke verbinding bedoeld wordt. Nadelen hiervan zijn dat de verbindingen dan wel evenveel vrijheidsgraden moeten hebben en dat de programmeur van de routine precies moet weten in welke volgorde de verbindingen door MADYMO aangesproken worden; hierin mag dan ook later geen verandering komen.
Een andere mogelijkheid kan misschien in een nieuwe versie van MADYMO gebracht worden. Door de gebruiker in de invoerfile onder het keyword JOINTS en na de naam USER een nummer mee te laten geven, en vervolgens te zorgen dat MADYMO dit nummer meegeeft aan de USRJN3-routine, kan deze routine bepalen om welk type verbinding het gaat. Een nadeel is dan echter nog steeds dat het aantal vrijheidsgraden van alle verbindingen in de routine hetzelfde moet zijn omdat in de routine USRSY3 het aantal graden van vrijheid van de user-joint vastligt in de array JNTDOF. Dit is wellicht te verhelpen door de user-joint van entry 1 naar het einde van deze array te verplaatsen en deze array te verlengen zodat van meer user-joints de vrijheidsgraden in te vullen zijn.

5.5 Numerieke efficiëntie

De numerieke efficiëntie van de subroutine USRJN3 kan duidelijk nog verbeterd worden. In de routine bevinden zich bijvoorbeeld twee subroutines om een matrix na te vermenigvuldigen met een vector: één voor de vermenigvuldiging van een 3×3 matrix met een 3×1 kolom en één voor de vermenigvuldiging van een 5×5 matrix met een 5×1 kolom. Een meer van belang zijnde verbetering kan verkregen worden door de berekening van de verbindindingsvectoren van de stangen uit de posities van de scharnierpunten en de beginschatting voor de Eulerparameters plaats te laten vinden in de subroutine USRIN3 en over te dragen door een COMMON-block. De bereke-

ning van de positievectoren van de stangen vindt dan slechts eenmalig plaats bij het starten van MADYMO en niet iedere keer dat de routine USRJN3 aangeroepen wordt. Tevens worden dan de berekende waarden voor de Eulerparameters en de verbindingsvector tussen de twee lichamen bewaard, hetgeen een betere beginschatting levert voor het Newton-Raphson proces dan in het geval dat iedere keer uitgegaan wordt van de nulstand.

Literatuur

[1] P. E. NIKRAVESH: Computer Aided Analysis of Mechanical Systems; Prentice Hall International Editions, Englewood Cliffs, New Jersey, 1988, pp. 196.

[2] A. SAUREN: Multibody Dynamica; Dictaat 4659, Technische Universiteit Eindhoven, 1990, pp. 1.1.

[3] J. VALENTINO: Fortran for Technologists and Engineers; Holt, Rinehart and Winston, 1985.

[4] A.W.M. DE JONG: BasishandleidingFortran 77; RC-Informatie AG-117, Technische Universiteit Eindhoven, 1988.

[5] Numerical RecipesFortran; Cambridge University Press, 1985.

[6] MADYMO5.0 UsersManual 3D; TNO (IW), 1992.

[7] MADYMO5.0 ProgrammersManual; TNO (IW), 1992.

[8] D. HEUVELMANS & R. NEIJSSEN: Een verkenning van de mogelijkheden met MADYMO 5.0 + bijlagen; WFW-rapport 92.133, Technische Universiteit Eindhoven, Faculteit Werktuigbouwkunde, Vakgroep WFW, 1992

Appendix A

Fortran broncodes

In deze appendix zijn de Fortran broncodes opgenomen die gebruikt zijn voor het programmeren van de kinematische verbindinding (subroutine USRJN3; appendix A.2) en het testen daarvan (programma PVAGEN; appendix A.1).
PVAGEN genereert posities en snelheden voor USRJN3 en stelt de lineaire resp. hoekversnellingen samen uit SDTT en SDQ resp. CDTT en CDQ en de versnelling van de vrijheidsgraad QTT. Uitvoer van posities, snelheden en versnellingen vindt plaats naar het scherm maar kan met behulp van een directive (>) naar een file geschreven worden (bijv. PVAGEN > PVA.DAT).
Wellicht ten overvloede zij opgemerkt dat de regelnummers (nnn:) niet tot de broncode behoren, doch slechts als referentie dienst doen.

A.1 Programma PVAGEN

```
 1:       PROGRAM PVAGEN
 2: C     ************************************************************************
 3: C     * PROGRAM TO EVALUATE SUBROUTINE USRJN3 BY SUPPLYING VALUES FOR Q *
 4: C     * AND QT SO ALL THE OTHER MATRICES CAN BE COMPUTED.  DATA IS      *
 5: C     * WRITTEN TO SCREEN OR ASCII-FILE TO BE EVALUATED IN E.G. MATLAB. *
 6: C     ************************************************************************
 7:       IMPLICIT NONE
 8:       DOUBLE PRECISION Q(1),QT(1),SD(3),SDT(3),SDTT(3),SDQ(3,1),
 9:      1                 CD(3,3),CDT(3),CDTT(3),CDQ(3,1)
10:       DOUBLE PRECISION Z,PI,QTT(1),LACC(3),AACC(3)
11:       INTEGER I
12:       EXTERNAL USRJN3
13: C    ************************************************************************
14: C    * INITIALIZE ALL MATRICES AND VECTORS TO ZERO; EXCEPT THE RELATIVE
15: C    * ROTATIONMATRIX WHICH IS INITIALIZED TO THE UNITY-MATRIX
16: C    *
17:       PI=3.14159265259D0
18:       DO 10 I=1,3
19:         SD(I)=0.0D0
20:         SDT(I)=0.0D0
```

```
21:         SDTT(I)=0.0D0
22:         SDQ(I,1)=0.0D0
23:         CD(I,1)=0.0D0
24:         CD(I,2)=0.0D0
25:         CD(I,3)=0.0D0
26:         CDT(I)=0.0D0
27:         CDTT(I)=0.0D0
28:         CDQ(I,1)=0.0D0
29:         CD(I,I)=1.0D0
30:    10 CONTINUE
31: C   ************************************************************************
32: C   * SET UP EXCITATION FOR USRJN3 IN Q(1) AND QT(1)
33: C   *
34:       DO 11 Z=0.0D0,PI*2.0D0,PI/3.0D1
35:         Q(1)=0.2D0*DSIN(Z)
36:         QT(1)=0.4D0*PI*DCOS(Z)
37:         QTT(1)=-0.8D0*PI**2*DSIN(Z)
38:         CALL USRJN3(Q,QT,SD,SDT,SDTT,SDQ,CD,CDT,CDTT,CDQ)
39: C   ************************************************************************
40: C   * LACC CONTAINS ABSOLUTE LINEAR ACCELERATIONS
41: C   * COMPUTED FROM COEFFICIENTS DEPENDENT AND INDEPENDENT ON QTT
42: C   *
43:         LACC(1)=SDTT(1)+SDQ(1,1)*QTT(1)
44:         LACC(2)=SDTT(2)+SDQ(2,1)*QTT(1)
45:         LACC(3)=SDTT(3)+SDQ(3,1)*QTT(1)
46: C   ************************************************************************
47: C   * AACC CONTAINS ABSOLUTE ANGULAR ACCELERATIONS
48: C   * COMPUTED FROM COEFFICIENTS DEPENDENT AND INDEPENDENT ON QTT
49: C   *
50:         AACC(1)=CDTT(1)+CDQ(1,1)*QTT(1)
51:         AACC(2)=CDTT(2)+CDQ(2,1)*QTT(1)
52:         AACC(3)=CDTT(3)+CDQ(3,1)*QTT(1)
53: C
54: C     WRITE X,Y,Z,XT,YT,ZT,XTT,YTT,ZTT,W1,W2,W3,WT1,WT2,WT3
55: C
56:         WRITE(*,9999)SD(1),SD(2),SD(3),SDT(1),SDT(2),SDT(3),LACC(1),
57:      &               LACC(2),LACC(3),CDT(1),CDT(2),CDT(3),AACC(1),
58:      &               AACC(2),AACC(3)
59:    11 CONTINUE
60:       STOP 'Task Completed.'
61:  9999 FORMAT(3X,E18.12,3X,E18.12,3X,E18.12,3X,E18.12,3X,E18.12,3X,
62:      &       E18.12,3X,E18.12,3X,E18.12,3X,E18.12,3X,E18.12,3X,E18.12,
63:      &       3X,E18.12,3X,E18.12,3X,E18.12,3X,E18.12)
64:       END
```

A.2 Subroutine USRJN3

```
      SUBROUTINE USRJN3(Q,QT,SD,SDT,SDTT,SDQ,D,CDT,CDTT,CDQ)
      IMPLICIT NONE
C
C     IN MADYMO-SUBROUTINE USRSY3 FOLLOWING DEGREES OF FREEDOM OF THIS
C     NEW JOINT HAVE TO BE SPECIFIED:
C     JNTDOF(1,1) = 1
C     JNTDOF(2,1) = 1
C
C     THIS USRJN3-ROUTINE SIMULATES A 5-POINT FRONT WHEEL SUSPENSION
C     RESEMBLING THAT OF SOME MERCEDESES.
C     CHASSIS IS BODY I; WHEELCARRIER IS BODY J
C     THE WHEELCARRIER IS CONNECTED TO THE CHASSIS BY 5 RODS BETWEEN
C     POINTS PIX AND XI (I=1..5). SD IS THE VECTOR CONNECTING
C     ORIGIN OF THE CHASSIS TO THE CENTER OF THE WHEEL (ORIGIN OF THE
C     WHEELCARRIER). COORDINATES OF ALL POINTS ARE IN METERS.
C     ALL EQUATIONS MENTIONED ARE DESCRIBED IN THE REPORT
C     'COMPLEX CLOSED CHAINS IN MADYMO'
C
C     VARIABLES PASSED BY SUBROUTINE
C
      DOUBLE PRECISION Q(*),QT(*),SD(3),SDT(3),SDTT(3),SDQ(3,*),
     1                 D(3,3),CDT(3),CDTT(3),CDQ(3,*)
C
C     LOCAL VARIABLES (POINTS ON WHEELCARRIER AND CHASSIS)
C
      DOUBLE PRECISION P1X(3),P2X(3),P3X(3),P4X(3),P5X(3),X1(3),X2(3),
     1                 X3(3),X4(3),X5(3),B1A(3)
C
C     VARIABLES TO CALCULATE LENGTH OF BARS AND INTERNAL COORDINATES
C     WITHIN WHEELCARRIER
C
      DOUBLE PRECISION L1(3),L2(3),L3(3),L4(3),L5(3),R01(3),R02(3),
     1                 R03(3),R04(3),R05(3),B1(3),B2(3),B3(3),B4(3),
     2                 B5(3),E0,E1,E2,E3,EPS,DFDQ(6,6),F(6),NORM,
     3                 NORM2,C(5),RHS(5),RHSQ(5),VI(3),ZMIN,ZMAX,
     4                 AVMAT(5,5),INVAV(5,5),CRHSI,TVEC5(5),
     5                 DB1(3),DB2(3),DB3(3),DB4(3),DB5(3)
      INTEGER I,ITER,ITERM,INDX(6)
C
      PARAMETER (ITERM=15 , ZMIN=-0.21D0 , ZMAX=0.28D0 , EPS = 1.0D-8)
C
C     TEST IF Q(1) IS WITHIN RANGE OF LOCAL Z-COORDINATES
C
      IF (Q(1).GT.ZMAX) THEN
        PRINT *,' Local Z coordinate of userjoint = ',Q(1),'.'
        PAUSE ' Z cannot become larger than 0.28 m !!'
      ENDIF
      IF (Q(1).LT.ZMIN) THEN
        PRINT *,' Local Z coordinate of userjoint = ',Q(1),'.'
```

```
50:         PAUSE ' Z cannot become smaller than -0.21 m !!'
51:       ENDIF
52: C
53: C     INITIALIZE COORDINATES OF LINKS ON THE CHASSIS
54: C     P1X UPTO P5X FROM BARS 1 UPTO 5.
55: C     (ALL MEASURES IN METERS!!!)
56: C
57:       P1X(1)=-0.0640D0
58:       P1X(2)= 0.4130D0
59:       P1X(3)= 0.3270D0
60: C
61:       P2X(1)=-0.3030D0
62:       P2X(2)= 0.4320D0
63:       P2X(3)= 0.2950D0
64: C
65:       P3X(1)=-0.0930D0
66:       P3X(2)= 0.3660D0
67:       P3X(3)= 0.0040D0
68: C
69:       P4X(1)=-0.2360D0
70:       P4X(2)= 0.3880D0
71:       P4X(3)=-0.1090D0
72: C
73:       P5X(1)= 0.2115D0
74:       P5X(2)= 0.3845D0
75:       P5X(3)=-0.1000D0
76: C
77: C     INITIALIZE COORDINATES OF ATTACHMENTS OF BARS 1 UPTO 5
78: C     ON WHEELCARRIER (X1 UPTO X5)
79: C
80:       X1(1)=-0.0640D0
81:       X1(2)= 0.6360D0
82:       X1(3)= 0.3450D0
83: C
84:       X2(1)=-0.1170D0
85:       X2(2)= 0.6360D0
86:       X2(3)= 0.3380D0
87: C
88:       X3(1)=-0.1880D0
89:       X3(2)= 0.6470D0
90:       X3(3)=-0.0230D0
91: C
92:       X4(1)=-0.0050D0
93:       X4(2)= 0.7370D0
94:       X4(3)=-0.1300D0
95: C
96:       X5(1)= 0.0025D0
97:       X5(2)= 0.7370D0
98:       X5(3)=-0.1340D0
99: C
100: C    INITIALIZE COORDINATES OF CENTER OF WHEEL: B1A
101: C
```

```
      B1A(1)= 0.0000D0
      B1A(2)= 0.7680D0
      B1A(3)= 0.0000D0
C
C     CALCULATE LENGTH OF BARS AND LOCAL WHEELCARRIER COORDINATES
C
      DO 10 I=1,3
        SD(I)=B1A(I)
        L1(I)=X1(I)-P1X(I)
        L2(I)=X2(I)-P2X(I)
        L3(I)=X3(I)-P3X(I)
        L4(I)=X4(I)-P4X(I)
        L5(I)=X5(I)-P5X(I)
        B1(I)=X1(I)-SD(I)
        B2(I)=X2(I)-SD(I)
        B3(I)=X3(I)-SD(I)
        B4(I)=X4(I)-SD(I)
        B5(I)=X5(I)-SD(I)
        R01(I)=P1X(I)
        R02(I)=P2X(I)
        R03(I)=P3X(I)
        R04(I)=P4X(I)
        R05(I)=P5X(I)
   10 CONTINUE
      SD(3)=Q(1)
      SDT(3)=QT(1)
C
C     NEWTON RAPHSON STEP 1
C     ASSUME INITIAL SOLUTION
C
      E0=1.0D0
      E1=0.0D0
      E2=0.0D0
      E3=0.0D0
      ITER=0
C
C     NEWTON-RAPHSON ITERATIONS
C
   12 ITER=ITER+1
      CALL CALCD(D,E0,E1,E2,E3)
C
C     CALCULATE ERRORCOLUMN F: NEWTON-RAPHSON STEP 2
C
      CALL CALCF(F,R01,R02,R03,R04,R05,SD,B1,B2,B3,B4,B5,
     1           L1,L2,L3,L4,L5,D,E0,E1,E2,E3)
C
C     EQUATION (30)
C
      CALL CDFDQ(DFDQ,R01,R02,R03,R04,R05,SD,B1,B2,B3,B4,B5,
     1             D,E0,E1,E2,E3)
C
C     NEWTON-RAPHSON STEP 3
```

```
154: C     CALCULATE COLUMN WITH ADJUSTMENTS TO COLUMN Q
155: C     LU DECOMPOSITION OF MATRIX DFDQ
156: C
157:       CALL LUDCMP(DFDQ,6,6,INDX)
158:       CALL LUBKSB(DFDQ,6,6,INDX,F)
159: C
160: C     F CONTAINS SOLUTION
161: C     NEWTON-RAPHSON STEP 4
162: C     ADJUST VARIABLES
163: C
164:       SD(1)=SD(1)+F(1)
165:       SD(2)=SD(2)+F(2)
166:       E0=E0+F(3)
167:       E1=E1+F(4)
168:       E2=E2+F(5)
169:       E3=E3+F(6)
170:       NORM2=NORM(F)
171:       IF (NORM2.GE.EPS) THEN
172:         IF (ITER.LE.ITERM) THEN
173:           GOTO 12
174:         ELSE
175:           PAUSE 'Solution failed to converge.'
176:         END IF
177:       END IF
178: C
179: C     **********     POSITION-ANALYSIS ENDED     **********
180: C
181: C     CALCULATE RELATIVE ROTATION MATRIX OUT OF EULERPARAMATERS E0..E3
182: C     AND USE IT TO COMPUTE NEW GLOBAL VECTORS B1..B5
183: C
184:       CALL CALCD(D,E0,E1,E2,E3)
185: C
186: C     CONVERT LOCAL VECTORS BI TO GLOBAL VECTORS DBI (EQUATION (27))
187: C
188:       CALL MXV(D,B1,DB1)
189:       CALL MXV(D,B2,DB2)
190:       CALL MXV(D,B3,DB3)
191:       CALL MXV(D,B4,DB4)
192:       CALL MXV(D,B5,DB5)
193: C
194: C     ***************** LINEAR AND ANGULAR VELOCITIES *****************
195: C     AVMAT IS MATRIX IN EQUATIONS (43) AND (47)
196: C     RHSQ IS RIGHT-HAND-SIDE-VECTOR OF EQUATION (43) WITHOUT THE TERM
197: C     DR6(3)/DT (=QT(1)). THIS IS CORRECTED WHEN SDT AND CDT ARE FILLED.
198: C
199:       CALL CALCVI(SD,R01,DB1,VI)
200:       RHSQ(1)=-2.0D0*VI(3)
201:       AVMAT(1,1)=2.0D0*VI(1)
202:       AVMAT(1,2)=2.0D0*VI(2)
203:       AVMAT(1,3)=2.0D0*(VI(3)*DB1(2)-VI(2)*DB1(3))
204:       AVMAT(1,4)=2.0D0*(VI(1)*DB1(3)-VI(3)*DB1(1))
205:       AVMAT(1,5)=2.0D0*(VI(2)*DB1(1)-VI(1)*DB1(2))
```

```
206: C
207:       CALL CALCVI(SD,R02,DB2,VI)
208:       RHSQ(2)=-2.0D0*VI(3)
209:       AVMAT(2,1)=2.0D0*VI(1)
210:       AVMAT(2,2)=2.0D0*VI(2)
211:       AVMAT(2,3)=2.0D0*(VI(3)*DB2(2)-VI(2)*DB2(3))
212:       AVMAT(2,4)=2.0D0*(VI(1)*DB2(3)-VI(3)*DB2(1))
213:       AVMAT(2,5)=2.0D0*(VI(2)*DB2(1)-VI(1)*DB2(2))
214: C
215:       CALL CALCVI(SD,R03,DB3,VI)
216:       RHSQ(3)=-2.0D0*VI(3)
217:       AVMAT(3,1)=2.0D0*VI(1)
218:       AVMAT(3,2)=2.0D0*VI(2)
219:       AVMAT(3,3)=2.0D0*(VI(3)*DB3(2)-VI(2)*DB3(3))
220:       AVMAT(3,4)=2.0D0*(VI(1)*DB3(3)-VI(3)*DB3(1))
221:       AVMAT(3,5)=2.0D0*(VI(2)*DB3(1)-VI(1)*DB3(2))
222: C
223:       CALL CALCVI(SD,R04,DB4,VI)
224:       RHSQ(4)=-2.0D0*VI(3)
225:       AVMAT(4,1)=2.0D0*VI(1)
226:       AVMAT(4,2)=2.0D0*VI(2)
227:       AVMAT(4,3)=2.0D0*(VI(3)*DB4(2)-VI(2)*DB4(3))
228:       AVMAT(4,4)=2.0D0*(VI(1)*DB4(3)-VI(3)*DB4(1))
229:       AVMAT(4,5)=2.0D0*(VI(2)*DB4(1)-VI(1)*DB4(2))
230: C
231:       CALL CALCVI(SD,R05,DB5,VI)
232:       RHSQ(5)=-2.0D0*VI(3)
233:       AVMAT(5,1)=2.0D0*VI(1)
234:       AVMAT(5,2)=2.0D0*VI(2)
235:       AVMAT(5,3)=2.0D0*(VI(3)*DB5(2)-VI(2)*DB5(3))
236:       AVMAT(5,4)=2.0D0*(VI(1)*DB5(3)-VI(3)*DB5(1))
237:       AVMAT(5,5)=2.0D0*(VI(2)*DB5(1)-VI(1)*DB5(2))
238: C
239:       CALL INVMAT(AVMAT,INVAV)
240:       CALL MXV5(INVAV,RHSQ,C)
241: C
242: C     C CONTAINS SOLUTION: C(1)=SDT(1)/QT(1); C(2)=SDT(2)/QT(1);
243: C     C(3)=CDT(1)/QT(1); C(4)=CDT(2)/QT(1); C(5)=CDT(3)/QT(1)
244: C
245:       CDT(1)=C(3)*QT(1)
246:       CDT(2)=C(4)*QT(1)
247:       CDT(3)=C(5)*QT(1)
248:       SDT(1)=C(1)*QT(1)
249:       SDT(2)=C(2)*QT(1)
250:       SDT(3)=QT(1)
251: C
252: C     **********     VELOCITY-ANALYSIS ENDED     ***********
253: C
254: C     ****** LINEAR AND ANGULAR ACCELERATION ANALYSIS ******
255: C
256: C     C IS STILL FILLED WITH SDQ AND CDQ FROM VELOCITY ANALYSIS (MATRIX
257: C     AND PART OF THE RIGHT-HAND-SIDE ARE THE SAME). COMPARE EQUATIONS
```

```
258: C     (43) AND (47)
259: C
260:       SDQ(1,1)=C(1)
261:       SDQ(2,1)=C(2)
262:       SDQ(3,1)=1.0D0
263:       CDQ(1,1)=C(3)
264:       CDQ(2,1)=C(4)
265:       CDQ(3,1)=C(5)
266: C
267: C     RHS CONTAINS VALUES INDEPENDENT OF QTT OF THE RIGHT-HAND-SIDE OF
268: C     EQUATION (47). RHS IS FILLED ACCORDING TO EQUATIONS (47) AND (48)
269: C
270:       RHS(1)=CRHSI(SD,SDT,CDT,DB1,R01)
271:       RHS(2)=CRHSI(SD,SDT,CDT,DB2,R02)
272:       RHS(3)=CRHSI(SD,SDT,CDT,DB3,R03)
273:       RHS(4)=CRHSI(SD,SDT,CDT,DB4,R04)
274:       RHS(5)=CRHSI(SD,SDT,CDT,DB5,R05)
275: C
276:       CALL MXV5(INVAV,RHS,TVEC5)
277:       SDTT(1)=TVEC5(1)
278:       SDTT(2)=TVEC5(2)
279:       SDTT(3)=0.0D0
280:       CDTT(1)=TVEC5(3)
281:       CDTT(2)=TVEC5(4)
282:       CDTT(3)=TVEC5(5)
283: C
284: C     **********   ACCELERATION ANALYSIS ENDED   **********
285: C
286:       RETURN
287:       END
288: C
289: C   ************************************************************************
290: C   *                          MAIN PROGRAM END                            *
291: C   ************************************************************************
292: C
293: C   ************************************************************************
294: C    SUBROUTINES TO CALCULATE MATRIX WITH PARTIAL DERIVATIVES (JACOBIAN)
295: C
296: C   ************************************************************************
297: C   * CALCD CALCULATES RELATIVE ROTATION-MATRIX D OUT OF E0,E1,E2 AND E3
298: C   * ACCORDING TO EQUATION (24)
299: C   *
300:       SUBROUTINE CALCD(D,E0,E1,E2,E3)
301:         DOUBLE PRECISION D(3,3),E0,E1,E2,E3
302:         D(1,1)=E0*E0+E1*E1-E2*E2-E3*E3
303:         D(1,2)=2.0D0*(E1*E2-E0*E3)
304:         D(1,3)=2.0D0*(E1*E3+E0*E2)
305:         D(2,1)=2.0D0*(E1*E2+E0*E3)
306:         D(2,2)=E0*E0+E2*E2-E1*E1-E3*E3
307:         D(2,3)=2.0D0*(E2*E3-E0*E1)
308:         D(3,1)=2.0D0*(E1*E3-E0*E2)
309:         D(3,2)=2.0D0*(E0*E1+E2*E3)
```

```
310:         D(3,3)=E0*E0+E3*E3-E1*E1-E2*E2
311:       RETURN
312:       END
313: C   ***********************************************************************
314: C   * DFIDQ1 CALCULATES DERIVATIVE OF I-TH CONSTRAINTEQUATION TO Q1 (X)
315: C   * ACCORDING TO EQUATION (31)
316: C   *
317:       DOUBLE PRECISION FUNCTION DFIDQ1(R6,Q,BI,ROI)
318:         DOUBLE PRECISION R6(3),Q(3,3),BI(3),ROI(3),TVEC(3)
319:         CALL MXV(Q,BI,TVEC)
320:         DFIDQ1=2.0D0*(TVEC(1)+R6(1)-ROI(1))
321:       RETURN
322:       END
323: C   ***********************************************************************
324: C   * DFIDQ2 CALCULATES DERIVATIVE OF I-TH CONSTRAINTEQUATION TO Q2 (Y)
325: C   * ACCORDING TO EQUATION (30)
326: C   *
327:       DOUBLE PRECISION FUNCTION DFIDQ2(R6,Q,BI,ROI)
328:         DOUBLE PRECISION R6(3),Q(3,3),BI(3),ROI(3),TVEC(3)
329:         CALL MXV(Q,BI,TVEC)
330:         DFIDQ2=2.0D0*(TVEC(2)+R6(2)-ROI(2))
331:       RETURN
332:       END
333: C   ***********************************************************************
334: C   * DFIDQ3 CALCULATES DERIVATIVE OF I-TH CONSTRAINTEQUATION TO Q3 (E0)
335: C   * ACCORDING TO EQUATION (33)
336: C   *
337:       DOUBLE PRECISION FUNCTION DFIDQ3(ROI,R6,E0,E1,E2,E3,BI)
338:         DOUBLE PRECISION ROI(3),R6(3),E0,E1,E2,E3,BI(3),TVEC(3),TEMP
339:         TVEC(1)= 2.0D0*E0*BI(1)-E3*BI(2)+E2*BI(3)
340:         TVEC(2)= E3*BI(1)+2.0D0*E0*BI(2)-E1*BI(3)
341:         TVEC(3)=-E2*BI(1)+E1*BI(2)+2.0D0*E0*BI(3)
342:         TEMP=(ROI(1)-R6(1))*TVEC(1)
343:         TEMP=TEMP+(ROI(2)-R6(2))*TVEC(2)
344:         TEMP=TEMP+(ROI(3)-R6(3))*TVEC(3)
345:         DFIDQ3=-4.0D0*TEMP
346:       RETURN
347:       END
348: C   ***********************************************************************
349: C   * DFIDQ4 CALCULATES DERIVATIVE OF I-TH CONSTRAINTEQUATION TO Q4 (E1)
350: C   * ACCORDING TO EQUATION (34)
351: C   *
352:       DOUBLE PRECISION FUNCTION DFIDQ4(ROI,R6,E0,E1,E2,E3,BI)
353:         DOUBLE PRECISION ROI(3),R6(3),E0,E1,E2,E3,BI(3),TVEC(3),TEMP
354:         TVEC(1)= 2.0D0*E1*BI(1)+E2*BI(2)+E3*BI(3)
355:         TVEC(2)= E2*BI(1)-E0*BI(3)
356:         TVEC(3)= E3*BI(1)+E0*BI(2)
357:         TEMP=(ROI(1)-R6(1))*TVEC(1)
358:         TEMP=TEMP+(ROI(2)-R6(2))*TVEC(2)
359:         TEMP=TEMP+(ROI(3)-R6(3))*TVEC(3)
360:         DFIDQ4=-4.0D0*TEMP
361:       RETURN
```

```
362:       END
363: C   ***********************************************************************
364: C   * DFIDQ5 CALCULATES DERIVATIVE OF I-TH CONSTRAINTEQUATION TO Q5 (E2)
365: C   * ACCORDING TO EQUATION (35)
366: C   *
367:       DOUBLE PRECISION FUNCTION DFIDQ5(ROI,R6,E0,E1,E2,E3,BI)
368:         DOUBLE PRECISION ROI(3),R6(3),E0,E1,E2,E3,BI(3),TVEC(3),TEMP
369:         TVEC(1)= E1*BI(2)+E0*BI(3)
370:         TVEC(2)= E1*BI(1)+2.0D0*E2*BI(2)+E3*BI(3)
371:         TVEC(3)=-E0*BI(1)+E3*BI(2)
372:         TEMP=(ROI(1)-R6(1))*TVEC(1)
373:         TEMP=TEMP+(ROI(2)-R6(2))*TVEC(2)
374:         TEMP=TEMP+(ROI(3)-R6(3))*TVEC(3)
375:         DFIDQ5=-4.0D0*TEMP
376:       RETURN
377:       END
378: C   ***********************************************************************
379: C   * DFIDQ6 CALCULATES DERIVATIVE OF I-TH CONSTRAINTEQUATION TO Q6 (E3)
380: C   * ACCORDING TO EQUATION (36)
381: C   *
382:       DOUBLE PRECISION FUNCTION DFIDQ6(ROI,R6,E0,E1,E2,E3,BI)
383:         DOUBLE PRECISION ROI(3),R6(3),E0,E1,E2,E3,BI(3),TVEC(3),TEMP
384:         TVEC(1)=-E0*BI(2)+E1*BI(3)
385:         TVEC(2)= E0*BI(1)+E2*BI(3)
386:         TVEC(3)= E1*BI(1)+E2*BI(2)+2.0D0*E3*BI(3)
387:         TEMP=(ROI(1)-R6(1))*TVEC(1)
388:         TEMP=TEMP+(ROI(2)-R6(2))*TVEC(2)
389:         TEMP=TEMP+(ROI(3)-R6(3))*TVEC(3)
390:         DFIDQ6=-4.0D0*TEMP
391:       RETURN
392:       END
393: C   ***********************************************************************
394: C   * CALCFI CALCULATES THE RESIDUAL F(I) OF THE CONSTRAINTEQUATIONS
395: C   * (25) AND (26) ACCORDING TO:  F(I)=||ROI-Q*BI-R6||^2-||LI||^2.
396: C   * (STEP 2 OF NEWTON-RAPHSON PROCESS)
397: C   *
398:       DOUBLE PRECISION FUNCTION CALCFI(ROI,BI,R6,LI,Q)
399:         DOUBLE PRECISION ROI(3),BI(3),R6(3),LI(3),Q(3,3),TVEC(3),TEMP
400:         INTEGER I
401:         CALCFI=0.0D0
402:         CALL MXV(Q,BI,TVEC)
403:         DO 10 I=1,3
404:           TEMP=ROI(I)-TVEC(I)-R6(I)
405:           CALCFI=CALCFI-(TEMP*TEMP-LI(I)*LI(I))
406:    10   CONTINUE
407:       RETURN
408:       END
409: C   ***********************************************************************
410: C   * CALCF CALCULATES COLUMN F WITH DEVIATIONS FROM THE CONSTRAINT-
411: C   * EQUATIONS (25) AND (26) WITH NULLVECTOR FOR USE IN NEWTON-RAPHSON
412: C   * PROCESS STEP 2
413: C   *
```

```
414:       SUBROUTINE CALCF(F,R01,R02,R03,R04,R05,R6,B1,B2,B3,B4,B5,
415:      1                 L1,L2,L3,L4,L5,Q,E0,E1,E2,E3)
416:       DOUBLE PRECISION F(6),L1(3),L2(3),L3(3),L4(3),L5(3),R01(3),R02(3),
417:      1                 R03(3),R04(3),R05(3),B1(3),B2(3),B3(3),
418:      2                 B4(3),B5(3),R6(3),E0,E1,E2,E3,Q(3,3),CALCFI
419:         F(1)=CALCFI(R01,B1,R6,L1,Q)
420:         F(2)=CALCFI(R02,B2,R6,L2,Q)
421:         F(3)=CALCFI(R03,B3,R6,L3,Q)
422:         F(4)=CALCFI(R04,B4,R6,L4,Q)
423:         F(5)=CALCFI(R05,B5,R6,L5,Q)
424: C     EQUATION (29)
425:         F(6)=1.0D0-(E0*E0+E1*E1+E2*E2+E3*E3)
426:       RETURN
427:       END
428: C   ***********************************************************************
429: C   * CDFDQ CALCULATES MATRIX DFDQ WITH PARTIAL DERIVATIVES OF THE
430: C   * CONSTRAINT-EQUATION WITH RESPECT TO THE UNKNOWN VARIABLES Q
431: C   * SEE EQUATION (30.A)
432: C   *
433:       SUBROUTINE CDFDQ(DFDQ,R01,R02,R03,R04,R05,R6,B1,B2,B3,B4,B5,
434:      1                 Q,E0,E1,E2,E3)
435:       DOUBLE PRECISION DFDQ(6,6),R01(3),R02(3),R03(3),R04(3),R05(3),
436:      1                 B1(3),B2(3),B3(3),B4(3),B5(3),R6(3),
437:      2                 E0,E1,E2,E3,Q(3,3),DFIDQ1,DFIDQ2,DFIDQ3,
438:      3                 DFIDQ4,DFIDQ5,DFIDQ6
439: C
440: C     FIRST COLUMN
441: C
442:         DFDQ(1,1)=DFIDQ1(R6,Q,B1,R01)
443:         DFDQ(2,1)=DFIDQ1(R6,Q,B2,R02)
444:         DFDQ(3,1)=DFIDQ1(R6,Q,B3,R03)
445:         DFDQ(4,1)=DFIDQ1(R6,Q,B4,R04)
446:         DFDQ(5,1)=DFIDQ1(R6,Q,B5,R05)
447: C     EQUATION (37)
448:         DFDQ(6,1)=0.0D0
449: C
450: C     SECOND COLUMN
451: C
452:         DFDQ(1,2)=DFIDQ2(R6,Q,B1,R01)
453:         DFDQ(2,2)=DFIDQ2(R6,Q,B2,R02)
454:         DFDQ(3,2)=DFIDQ2(R6,Q,B3,R03)
455:         DFDQ(4,2)=DFIDQ2(R6,Q,B4,R04)
456:         DFDQ(5,2)=DFIDQ2(R6,Q,B5,R05)
457: C     EQUATION (37)
458:         DFDQ(6,2)=0.0D0
459: C
460: C     THIRD COLUMN
461: C
462:         DFDQ(1,3)=DFIDQ3(R01,R6,E0,E1,E2,E3,B1)
463:         DFDQ(2,3)=DFIDQ3(R02,R6,E0,E1,E2,E3,B2)
464:         DFDQ(3,3)=DFIDQ3(R03,R6,E0,E1,E2,E3,B3)
465:         DFDQ(4,3)=DFIDQ3(R04,R6,E0,E1,E2,E3,B4)
```

```
466:         DFDQ(5,3)=DFIDQ3(R05,R6,E0,E1,E2,E3,B5)
467: C     EQUATION (38)
468:         DFDQ(6,3)=2.0D0*E0
469: C
470: C     FOURTH COLUMN
471: C
472:         DFDQ(1,4)=DFIDQ4(R01,R6,E0,E1,E2,E3,B1)
473:         DFDQ(2,4)=DFIDQ4(R02,R6,E0,E1,E2,E3,B2)
474:         DFDQ(3,4)=DFIDQ4(R03,R6,E0,E1,E2,E3,B3)
475:         DFDQ(4,4)=DFIDQ4(R04,R6,E0,E1,E2,E3,B4)
476:         DFDQ(5,4)=DFIDQ4(R05,R6,E0,E1,E2,E3,B5)
477: C     EQUATION (38)
478:         DFDQ(6,4)=2.0D0*E1
479: C
480: C     FIFTH COLUMN
481: C
482:         DFDQ(1,5)=DFIDQ5(R01,R6,E0,E1,E2,E3,B1)
483:         DFDQ(2,5)=DFIDQ5(R02,R6,E0,E1,E2,E3,B2)
484:         DFDQ(3,5)=DFIDQ5(R03,R6,E0,E1,E2,E3,B3)
485:         DFDQ(4,5)=DFIDQ5(R04,R6,E0,E1,E2,E3,B4)
486:         DFDQ(5,5)=DFIDQ5(R05,R6,E0,E1,E2,E3,B5)
487: C     EQUATION (38)
488:         DFDQ(6,5)=2.0D0*E2
489: C
490: C     SIXTH COLUMN
491: C
492:         DFDQ(1,6)=DFIDQ6(R01,R6,E0,E1,E2,E3,B1)
493:         DFDQ(2,6)=DFIDQ6(R02,R6,E0,E1,E2,E3,B2)
494:         DFDQ(3,6)=DFIDQ6(R03,R6,E0,E1,E2,E3,B3)
495:         DFDQ(4,6)=DFIDQ6(R04,R6,E0,E1,E2,E3,B4)
496:         DFDQ(5,6)=DFIDQ6(R05,R6,E0,E1,E2,E3,B5)
497: C     EQUATION (38)
498:         DFDQ(6,6)=2.0D0*E3
499:       RETURN
500:       END
501: C   ************************************************************************
502: C   * CALCVI CALCULATES THE VECTOR (R6+DBI-R0I) THIS EQUALS VECTOR RI
503: C   * IN EQUATIONS (25) AND (26)
504: C   *
505:       SUBROUTINE CALCVI(R,RI,BI,VI)
506:         DOUBLE PRECISION R(3),RI(3),BI(3),VI(3)
507:         VI(1)=R(1)-RI(1)+BI(1)
508:         VI(2)=R(2)-RI(2)+BI(2)
509:         VI(3)=R(3)-RI(3)+BI(3)
510:       RETURN
511:       END
512: C   ************************************************************************
513: C   * CRHSI CALCULATES THE I-TH ENTRY OF THE RIGHT-HAND-SIDE-VECTOR
514: C   * IN THE EXPRESSION FOR THE ACCELERATIONS (COLUMN -2*VI IN
515: C   * EQUATION (47) ACCORDING TO EQUATION (48))
516: C   *
517:       DOUBLE PRECISION FUNCTION CRHSI(V,VDOT,W,D,R0)
```

```
518:         DOUBLE PRECISION V(3),VDOT(3),W(3),D(3),R0(3),TV(3),
519:      1                   TV1(3),VI(3)
520: C
521: C     TEMPORARY VECTOR TV CONTAINS THE RESULTS OF THE EXPRESSION
522: C     W CROSS W CROSS B
523: C
524:       TV(1)=W(1)*(W(2)*D(2)+W(3)*D(3))-(W(2)**2+W(3)**2)*D(1)
525:       TV(2)=W(2)*(W(3)*D(3)+W(1)*D(1))-(W(3)**2+W(1)**2)*D(2)
526:       TV(3)=W(3)*(W(1)*D(1)+W(2)*D(2))-(W(1)**2+W(2)**2)*D(3)
527: C
528:       TV1(1)=VDOT(1)+W(2)*D(3)-W(3)*D(2)
529:       TV1(2)=VDOT(2)+W(3)*D(1)-W(1)*D(3)
530:       TV1(3)=VDOT(3)+W(1)*D(2)-W(2)*D(1)
531: C
532:       TV1(1)=TV1(1)**2
533:       TV1(2)=TV1(2)**2
534:       TV1(3)=TV1(3)**2
535: C
536:       CALL CALCVI(V,R0,D,VI)
537: C
538:       TV(1)=TV(1)*VI(1)
539:       TV(2)=TV(2)*VI(2)
540:       TV(3)=TV(3)*VI(3)
541: C
542:       CRHSI=-2.0D0*(TV(1)+TV(2)+TV(3)+TV1(1)+TV1(2)+TV1(3))
543:       RETURN
544:       END
545:       SUBROUTINE MXV(M,V,R)
546: C   ***********************************************************************
547: C   * MXV CALCULATES THE PRODUCT OF A 3X3 MATRIX WITH A 3X1 VECTOR
548: C   *
549: C   * M : 3X3 MATRIX
550: C   * V : 3X1 VECTOR
551: C   * R : 3X1 VECTOR WITH RESULTS OF M * V
552: C   *
553:         DOUBLE PRECISION M(3,3),V(3),R(3)
554:         INTEGER I
555:         DO 10 I=1,3
556:           R(I)=M(I,1)*V(1)+M(I,2)*V(2)+M(I,3)*V(3)
557:    10   CONTINUE
558:       RETURN
559:       END
560: C   ***********************************************************************
561: C   * MXV5 CALCULATES THE PRODUCT OF A 5X5 MATRIX WITH A 5X1 VECTOR
562: C   *
563: C   *
564: C   * M : 5X5 MATRIX
565: C   * V : 5X1 VECTOR
566: C   * R : 5X1 VECTOR WITH RESULTS OF M * V
567: C   *
568:       SUBROUTINE MXV5(M,V,R)
569:         DOUBLE PRECISION M(5,5),V(5),R(5)
```

```
570:         INTEGER I
571:         DO 10 I=1,5
572:           R(I)=M(I,1)*V(1)+M(I,2)*V(2)+M(I,3)*V(3)
573:           R(I)=R(I)+M(I,4)*V(4)+M(I,5)*V(5)
574:     10   CONTINUE
575:        RETURN
576:        END
577: C    ************************************************************************
578: C    * NORM(V) CALCULATES THE 2-NORM (EUCLIDIAN NORM) OF A VECTOR V
579: C    * WITH LENGTH 6
580: C    *
581:        DOUBLE PRECISION FUNCTION NORM(V)
582:          DOUBLE PRECISION V(6),TEMP
583:          INTEGER I
584:          TEMP=0.0D0
585:          DO 10 I=1,6
586:            TEMP=TEMP+V(I)*V(I)
587:     10   CONTINUE
588:          NORM=DSQRT(TEMP)
589:        RETURN
590:        END
591: C    ************************************************************************
592: C    * INVMAT INVERTS A 5X5 MATRIX USING LU-DECOMPOSITION
593: C    * AND STORES RESULTS IN INVMAT
594: C    *
595:        SUBROUTINE INVMAT(MAT,INV)
596:          IMPLICIT NONE
597:          DOUBLE PRECISION MAT(5,5),RESMAT(5,5),INV(5,5),B(5)
598:          INTEGER I,J,INDX(5)
599:          CALL COPYM(MAT,RESMAT)
600:          CALL LUDCMP(RESMAT,5,5,INDX)
601:          DO 20 I=1,5
602:            DO 19 J=I,5
603:              IF (I.EQ.J) THEN
604:                INV(I,J)=1.0D0
605:              ELSE
606:                INV(I,J)=0.0D0
607:                INV(J,I)=0.0D0
608:              ENDIF
609:     19     CONTINUE
610:     20   CONTINUE
611:          DO 30 I=1,5
612:            DO 21 J=1,5
613:              B(J)=INV(J,I)
614:     21     CONTINUE
615:            CALL LUBKSB(RESMAT,5,5,INDX,B)
616:            DO 22 J=1,5
617:              INV(J,I)=B(J)
618:     22     CONTINUE
619:     30   CONTINUE
620:        RETURN
621:        END
```

```
C   ***********************************************************************
C   * COPYM COPIES A MATRIX FOR SAFEKEEPING WHEN USING LU-DECOMPOSITION
C   *
      SUBROUTINE COPYM(M1,M2)
      DOUBLE PRECISION M1(5,5),M2(5,5)
      INTEGER I,J
      DO 100 I=1,5
        DO 99 J=1,5
          M2(I,J)=M1(I,J)
   99   CONTINUE
  100 CONTINUE
      RETURN
      END
C   ***********************************************************************
C   * SUBROUTINE LUDCMP
C   *
C   * GIVEN A NXN MATRIX A, WITH PHYSICAL DIMENSION NP, THIS ROUTINE
C   * REPLACES IT BY THE LU-DECOMPOSITION OF A ROWWISE PERMUTATION OF
C   * ITSELF. A AND N ARE INPUT; A IS OUTPUT. INDX IS AN OUTPUT VECTOR
C   * WHICH RECORDS THE ROW PERMUTATION EFFECTED BY THE PARTIAL PIVOTTING.
C   * THIS ROUTINE IS USED IN COMBINATION WITH THE SUBROUTINE LUBKSB TO
C   * SOLVE LINEAR EQUATIONS.
C   *
      SUBROUTINE LUDCMP(A,N,NP,INDX)
      IMPLICIT NONE
      INTEGER NMAX, N, NP
      DOUBLE PRECISION TINY
      PARAMETER (NMAX=100,TINY=1.0E-20)
C
C     VV STORES THE IMPLICIT SCALING OF EACH ROW
C
      DOUBLE PRECISION A(NP,NP),VV(NMAX), AAMAX, SUM, K, DUM
      INTEGER INDX(N),I,J,IMAX
C
C     LOOP OVER ROWS TO GET IMPLICIT SCALING INFORMATION
C
      DO 12 I=1,N
        AAMAX=0.
        DO 11 J=1,N
          IF (ABS(A(I,J)).GT.AAMAX) AAMAX=ABS(A(I,J))
   11   CONTINUE
C
C       NO NONZERO LARGEST ELEMENT
C
        IF (AAMAX.EQ.0.) PAUSE 'T-matrix singular.'
C
C       SAVE THE SCALING
C
        VV(I)=1.0D0/AAMAX
   12 CONTINUE
C
C     HERE BEGINS THE LOOP OVER COLUMNS OF CROUT'S METHOD
```

```
674: C
675:       DO 19 J=1,N
676:         IF (J.GT.1) THEN
677:           DO 14 I=1,J-1
678:             SUM=A(I,J)
679:             IF (I.GT.1)THEN
680:               DO 13 K=1,I-1
681:                 SUM=SUM-A(I,K)*A(K,J)
682:    13           CONTINUE
683:               A(I,J)=SUM
684:             ENDIF
685:    14       CONTINUE
686:         ENDIF
687:         AAMAX=0.0D0
688:         DO 16 I=J,N
689:           SUM=A(I,J)
690:           IF (J.GT.1)THEN
691:             DO 15 K=1,J-1
692:               SUM=SUM-A(I,K)*A(K,J)
693:    15         CONTINUE
694:             A(I,J)=SUM
695:           ENDIF
696:           DUM=VV(I)*ABS(SUM)
697:           IF (DUM.GE.AAMAX) THEN
698:             IMAX=I
699:             AAMAX=DUM
700:           ENDIF
701:    16   CONTINUE
702:         IF (J.NE.IMAX)THEN
703:           DO 17 K=1,N
704:             DUM=A(IMAX,K)
705:             A(IMAX,K)=A(J,K)
706:             A(J,K)=DUM
707:    17     CONTINUE
708:           VV(IMAX)=VV(J)
709:         ENDIF
710:         INDX(J)=IMAX
711:         IF(J.NE.N)THEN
712:           IF(A(J,J).EQ.0.)A(J,J)=TINY
713:           DUM=1./A(J,J)
714:           DO 18 I=J+1,N
715:             A(I,J)=A(I,J)*DUM
716:    18     CONTINUE
717:         ENDIF
718:    19 CONTINUE
719:       IF(A(N,N).EQ.0.)A(N,N)=TINY
720:       RETURN
721:       END
722: C    ***********************************************************************
723: C    * SUBROUTINE LUBKSB
724: C    *
725: C    * SOLVES THE SET OF N LINEAR EQUATIONS A.X=B. HERE A IS INPUT, NOT
```

```
C   * AS THE MATRIX A BUT RATHER AS ITS LU-DECOMPOSITION, OBTAINED BY
C   * ROUTINE LUDCMP. INDX IS INPUT AS THE PERMUTATION VECTOR RETURNED
C   * BY LUDCMP. B IS INPUT  AS THE RIGHT-HAND SIDE VECTOR B, AND
C   * RETURNS WITH THE SOLUTION VECTOR X. A, N, NP AND INDX ARE NOT
C   * MODIFIED BY THIS ROUTINE AND CAN BE LEFT IN PLACE FOR SUCCESSIVE
C   * CALLS WITH DIFFERENT RIGHT-HAND SIDES B. THE SOLUTION VECTOR X
C   * WILL BE RETURNED IN THE ARRAY B.
C   *
      SUBROUTINE LUBKSB(A,N,NP,INDX,B)
      INTEGER I, J, N, NP
      DOUBLE PRECISION A(NP,NP),B(N),SUM
      INTEGER INDX(N),II,LL
      II=0
      DO 12 I=1,N
        LL=INDX(I)
        SUM=B(LL)
        B(LL)=B(I)
        IF (II.NE.0)THEN
          DO 11 J=II,I-1
            SUM=SUM-A(I,J)*B(J)
11        CONTINUE
        ELSE IF (SUM.NE.0.) THEN
          II=I
        ENDIF
        B(I)=SUM
12    CONTINUE
      DO 14 I=N,1,-1
        SUM=B(I)
        IF(I.LT.N)THEN
          DO 13 J=I+1,N
            SUM=SUM-A(I,J)*B(J)
13        CONTINUE
        ENDIF
        B(I)=SUM/A(I,I)
14    CONTINUE
      RETURN
      END
```

Appendix B

MADYMO invoer-files

In deze appendix zijn de invoerfiles weergegeven zoals deze gebruikt zijn om de subroutine USRJN3 op haar werking te testen. Appendix B.1 geeft de invoerfile weer waar de z-positie door middel van een driver wordt voorgeschreven zodat een regelmatig verloop van deze coördinaat in de tijd verkregen wordt. Appendix B.2 geeft de invoer-file weer waarmee op tijdstip t=0 een snelheid wordt voorgeschreven. De debug-file van deze invoerfile geeft aan of de kinetische energie van het systeem constant blijft.
Beide systemen bestaan uit twee lichamen (de vaste wereld en een wiel) en een kinematische verbinding (de wielophanging).

B.1 Invoer-file voor voorgeschreven plaats

```
1
Test van USRJN3: Voorwielophanging
Mei 1993, P.G.M. Kruijt
0.0 1.0
0 0.025
0 0.5  0.01   0.1
INERTIAL SPACE
NULPUNT
PLANES
  0  1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0 0 0 NUL
-999
END INERTIAL SPACE
SYSTEM 1
WIEL
CONFIGURATION
 1
-999
GEOMETRY
  0.0 0.0 0.0 0.0 0.0 0.0  Wielmidden
-999
INERTIA
```

```
  12.0  0.74 1.46 0.74
-999
JOINTS
  1 USER
-999
ORIENTATIONS
  1  1  1  2   0.0
-999
ELLIPSOIDS
  1 .6 .15 .6 .0 .0 .0 2
-999
DRIVER
  1  1
-999
FUNCTIONS
 2
  0.0  -0.2  1.0  0.2
-999
END SYSTEM 1
FORCE MODEL
ACCELERATION FIELDS
  1  1  0  0  0
-999
KELVIN ELEMENTS
  -1 0 -.064  .413   .327 1 1  -.064  -.132  .345
  -1 0 -.303  .432   .295 1 1  -.117  -.132  .338
  -1 0 -.093  .366   .004 1 1  -.188  -.121 -.023
  -1 0 -.236  .388  -.109 1 1  -.005  -.031 -.130
  -1 0  .2115 .3845 -.100 1 1   .0025 -.031 -.134
-999
END FORCE MODELS
OUTPUT CONTROL PARAMETERS
  1  0  0.025  2
LINDIS
  1  1  0.0  0.0  0.0  -1 0       Wielverpl.
-999
CONSTRAINT LOADS
 1 1
-999
FORCES
 3  1  1
 3  2  1
 3  3  1
 3  4  1
 3  5  1
-999
END OUTPUT
END INPUT
```

B.2 Invoer-file voor voorgeschreven beginsnelheid

```
1
Test van USRJN3: Voorwielophanging
Mei 1993, P.G.M. Kruijt
0.0 1.0
0 0.025
0 0.5  0.01   0.1
INERTIAL SPACE
NULPUNT
PLANES
  0  1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0 0 0 NUL
-999
END INERTIAL SPACE
SYSTEM 1
WIEL
CONFIGURATION
 1
-999
GEOMETRY
  0.0 0.0 0.0 0.0 0.0 0.0  Wielmidden
-999
INERTIA
  12.0  0.74 1.46 0.74
-999
JOINTS
  1 USER
-999
ORIENTATIONS
  1  1  1  2   0.0
-999
ELLIPSOIDS
  1 .6 .15 .6 .0 .0 .0 2
-999
INITIAL CONDITIONS

JOINT DOF
  1  FREE   -0.2  0.3
-999
END SYSTEM 1
FORCE MODEL
ACCELERATION FIELDS
  1  1  0  0  0
-999
KELVIN ELEMENTS
  -1 0 -.064  .413   .327 1 1  -.064  -.132  .345
  -1 0 -.303  .432   .295 1 1  -.117  -.132  .338
  -1 0 -.093  .366   .004 1 1  -.188  -.121 -.023
  -1 0 -.236  .388  -.109 1 1  -.005  -.031 -.130
  -1 0  .2115 .3845 -.100 1 1   .0025 -.031 -.134
-999
```

```
END FORCE MODELS
OUTPUT CONTROL PARAMETERS
  1  0  0.025  2
LINDIS
  1  1  0.0  0.0  0.0  -1 0        Wielverpl.
-999
CONSTRAINT LOADS
 1 1
-999
FORCES
 3  1  1
 3  2  1
 3  3  1
 3  4  1
 3  5  1
-999
END OUTPUT
END INPUT
```