EINDHOVEN UNIVERSITY OF TECHNOLOGY

Het gebruik van "Singular Value Decomposition" voor de analyse van de dynamica van mechanische systemen

Citation for published version (APA):
Starmans, E. M. (1987). *Het gebruik van "Singular Value Decomposition" voor de analyse van de dynamica van mechanische systemen*. (DCT rapporten; Vol. 1987.010). Technische Universiteit Eindhoven.

Document status and date:
Gepubliceerd: 01/01/1987

Document Version:
Uitgevers PDF, ook bekend als Version of Record

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 04. Oct. 2023

WFW 87.010

Het gebruik van "Singular Value Decomposition" voor de analyse van de dynamica van mechanische systemen

verslag van een stage-opdracht

door E.M. STARMANS
id.nr. 175605

15-01-1987

begeleider : Wil Koppens

Samenvatting

Het vastleggen van de positie van een multibody-systeem in termen van een set onafhankelijke parameters is vaak problematisch. In zo'n geval wordt vaak een set afhankelijke parameters gebruikt, terwijl de onderlinge relaties worden vastgelegd via een set algebraïsche vergelijkingen.
In dit verslag wordt de "Singular Value Decomposition"-methode bekeken, waarmee men uit deze set afhankelijke parameters een set onafhankelijke parameters kan samenstellen. Deze methode is gebruikt voor het bepalen van het dynamisch gedrag van een slinger. Dit is zowel numeriek als analytisch gedaan om een beter inzicht te krijgen in deze methode
Geconcludeerd kan worden dat de SVD-methode zeer bruikbaar kan zijn voor het analyseren van de dynamica van mechanische systemen.

Inhoudsopgave

Hoofdstuk 1: Inleiding

In de multibody-dynamica stuit men vaak op problemen bij het beschrijven van de configuratie van een systeem in termen van een set onafhankelijke parameters
Het is dan wel mogelijk een set afhankelijke parameters te nemen, waarbij men dan de onderlinge relaties verwerkt in een aantal algebraische vergelijkingen (de zgn. constraints). Vervolgens kan er met verschillende methoden een set onafhankelijke parameters worden bepaald uit deze afhankelijke parameters.
Eén zo'n methode is het onderwerp van deze stageopdracht, nl. de "Singular Value Decomposition"-methode.
Het doel van deze opdracht was vooral het begrijpelijk maken van deze methode. Het artikel "Application of Singular Value Decomposition for Analysis of Mechanical System Dynamics" in het "Journal of Mechanisms, Transmissions, and Automation in Design", mrt. 1985, Vol. 107, van de hand van N.K. Mani, E.J. Haug en K.E. Atkinson diende hierbij als leidraad.
De algemene theorie achter de SVD-methode wordt in hoofdstuk 2 gepresenteerd, en vervolgens in hoofdstuk 3 verduidelijkt aan de hand van de slinger.
Het volgende hoofdstuk bevat de beschrijving van een algorithme, dat op deze methode is gebaseerd.
Dit algorithme werd in een Fortran-programma verwerkt, zodat de werking van de methode ook numeriek gecontroleerd kon worden (zie hoofdstuk 5)
Enige toelichting bij dit programma wordt in het zesde hoofdstuk gegeven.

Hoofdstuk 2: Het gebruik van "Singular Value Decomposition" voor de analyse van de dynamica van mechanische systemen

Dit hoofdstuk bevat een samenvatting van het artikel "Application of Singular Value Decomposition for Analysis of Mechanical System Dynamics", dat door Mani, Haug en Atkinson is gepubliceerd in het "Journal of Mechanisms, Transmissions, and Automation in Design", mrt. 1985, Vol. 107

2.1 Inleiding

De kinematica van grootschalige systemen kan het eenvoudigst worden gedefinieerd met behulp van een maximale set Cartesische gegeneraliseerde coördinaten, die moeten voldoen aan kinematische randvoorwaarden. Zo'n coördinatensysteem kan echter minder geschikt zijn voor het oplossen van de bewegingsvergelijkingen.
In dit artikel wordt een "Singular Value Decomposition"-algorithme gepresenteerd, dat de Cartesische gegeneraliseerde coördinaten transformeert in een coördinatensysteem, dat beter geschikt is voor het oplossen van de bewegingsvergelijkingen.

De theorie wordt toegelicht aan de hand van vlakke dynamische systemen, maar is evenzeer toepasbaar op ruimtelijke dynamica.
Een typisch lichaam i is te zien in fig. 2.1; het Cartesische X-Y-coördinatensysteem ligt vast in de ruimte, het ξ_i-η_i-systeem is bevestigd aan lichaam i in het massamiddelpunt. De positie en oriëntatie van het lichaam in het X-Y-vlak worden gegeven door de coördinaten x_i en y_i van het

massamiddelpunt en de hoek φ_i die de ξ_i-as maakt met de X-as
Voor lichaam i kan nu een gegeneraliseerde coördinaatvector $\underset{\sim}{q}_i$ gedefinieerd worden als:

$$\underset{\sim}{q}_i = [x_i \;\; y_i \;\; \varphi_i]^T \qquad (2.1)$$

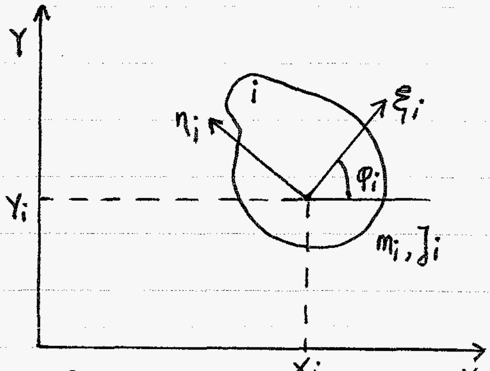

fig. 2.1

en de corcespondecende gegeneraliseerde krachtvector als:

$$\underset{\sim}{Q}_i = [Q_{ix} \;\; Q_{iy} \;\; Q_{i\varphi}]^T \qquad (2.2)$$

waarin Q_{ix} en Q_{iy} de X- en Y-componenten zijn van de kracht die aangrijpt in de oorsprong van het ξ_i-η_i-systeem, en $Q_{i\varphi}$ het moment dat op het lichaam wordt uitgeoefend.
Wanneer het systeem uit NB lichamen bestaat, kan de complete vector van gegeneraliseerde coördinaten gegeven worden door:

$$\underset{\sim}{q} = [\underset{\sim}{q}_1^T \;\; \underset{\sim}{q}_2^T \;\ldots\; \underset{\sim}{q}_{NB}^T]^T \quad , \underset{\sim}{q} \in R^n \qquad (2.3)$$

en de complete vector van gegeneraliseerde krachten door:

$$\underset{\sim}{Q} = [\underset{\sim}{Q}_1^T \;\; \underset{\sim}{Q}_2^T \;\ldots\; \underset{\sim}{Q}_{NB}^T]^T \quad , \underset{\sim}{Q} \in R^n \qquad (2.4)$$

met $n = 3 \cdot NB$ (2.5)

De kinematische randvoorwaarden tussen lichamen in het systeem kunnen worden geschreven als:

$$\underset{\sim}{\Phi}(t, \underset{\sim}{q}) = [\Phi_1 \;\; \Phi_2 \;\ldots\; \Phi_m]^T = \underset{\sim}{0} \quad , \underset{\sim}{\Phi} \in R^m \qquad (2.6)$$

waarin m het aantal randvoorwaarden is.
Om er zeker van te zijn dat deze voorwaarden onafhankelijk

zijn, is het noodzakelijk dat de $m \times n$ Jacobiaanmatrix $\underline{\Phi}_q$ rang m bezit. $\underline{\Phi}_q$ is als volgt gedefinieerd:

$$(\underline{\Phi}_q)_{ij} = \partial \Phi_i / \partial q_j \qquad (2.7)$$

De kinetische energie van het systeem kan worden geschreven als:

$$T = \frac{1}{2} \dot{\underset{\sim}{q}}^T \underline{M} \dot{\underset{\sim}{q}} \qquad (2.8)$$

waarin $\underline{M}$ de massamatrix van het systeem is.
De massamatrix van lichaam i wordt gedefinieerd als:

$$\underline{M}_i = \mathrm{Diag}[m_i, m_i, J_i] \qquad (2.9)$$

waarin m_i en J_i de massa en het traagheidsmoment van lichaam i zijn. Hieruit volgt de systeem-massamatrix:

$$\underline{M} = \mathrm{Diag}[\underline{M}_1, \underline{M}_2, \ldots, \underline{M}_{NB}] \qquad (2.10)$$

De Lagrange-bewegingsvergelijkingen voor systemen waarin geen arbeid wordt verricht in de verbindingen, luiden:

$$\underline{M} \ddot{\underset{\sim}{q}} + \underline{\Phi}_q^T \underset{\sim}{\lambda} = \underset{\sim}{Q} \qquad (2.11)$$

waarin $\underset{\sim}{\lambda}$ een vector van Lagrange-multiplicatoren is.
De beginvoorwaarden op het tijdstip $t = t_0$ zijn:

$$\left. \begin{aligned} \underset{\sim}{q}(t_0) &= \underset{\sim}{q}^0 \\ \dot{\underset{\sim}{q}}(t_0) &= \dot{\underset{\sim}{q}}^0 \end{aligned} \right\} \qquad (2.12)$$

waarbij de beginpositie $\underset{\sim}{q}^0$ en de beginsnelheid $\dot{\underset{\sim}{q}}^0$ aan de randvoorwaarden van het systeem moeten voldoen.

Differentiëren naar de tijd van de constraintvergelijking:

(2.6) geeft de snelheidsvergelijking

$$\underline{\Phi}_q \,\tilde{\dot{q}} + \tilde{\Phi}_t = \tilde{0} \qquad (2.13)$$

Nogmaals differentiëren geeft de versnellingsvergelijking

$$\underline{\Phi}_q \,\tilde{\ddot{q}} = -2\,\underline{\Phi}_{tq}\,\tilde{\dot{q}} - \underline{(\underline{\Phi}_q \tilde{\dot{q}})_q}\,\tilde{\dot{q}} - \tilde{\Phi}_{tt} \qquad (2.14)$$

Combineren van vergelijkingen (2.11) en (2.14) geeft een systeem van matrixvergelijkingen voor versnellingen en Lagrange-multiplicatoren:

$$\begin{bmatrix} \underline{M} & \underline{\Phi}_q^T \\ \underline{\Phi}_q & 0 \end{bmatrix} \begin{bmatrix} \tilde{\ddot{q}} \\ \tilde{\lambda} \end{bmatrix} = \begin{bmatrix} \tilde{Q} \\ -2\,\underline{\Phi}_{tq}\,\tilde{\dot{q}} - \underline{(\underline{\Phi}_q \tilde{\dot{q}})_q}\,\tilde{\dot{q}} - \tilde{\Phi}_{tt} \end{bmatrix} \qquad (2.15)$$

Dit systeem legt samen met de constraint vergelijking (2.6) en de beginvoorwaarden (2.12) de respons van het systeem volledig vast.
De vergelijkingen (2.15) en (2.6) vertegenwoordigen een systeem van gemengde "differential algebraic equations" (DAE).
De conventionele numerieke methoden voor het oplossen van gewone differentiaalvergelijkingen zijn over het algemeen niet toepasbaar op DAE-systemen
Singular Value Decomposition (SVD) is hiervoor wèl geschikt. Bij het gebruik van SVD worden niet alle n gegeneraliseerde coördinaten geïntegreerd, maar slechts de n-m onafhankelijke, waarna de afhankelijke coördinaten uit de constraintvergelijking volgen
Als onafhankelijke gegeneraliseerde coördinaten worden lineaire combinaties van de fysische gegeneraliseerde coördinaten $\tilde{q}$ geselecteerd volgens:

$$\tilde{zI} = \underline{VI}\,\tilde{q} \qquad (2.16)$$

waarin $\underline{VI}$ een $(n-m) \times n$ transformatiematrix is.
Doordat de rijen van VI onderling orthogonaal worden gekozen, zijn de resulterende lineaire combinaties onderling onafhankelijk.

2.2 Basiseigenschappen van SVD

De $m \times n$ Jacobiaan $\underline{\Phi_q}$, met $m < n$, kan worden gedecomponeerd in de vorm:

$$\underline{\Phi_q} = \underline{U}^T \underline{D}\, \underline{V} \tag{2.17}$$

waarin $\underline{U}$ en $\underline{V}$ orthonormale matrices van dimensie $m \times m$ resp. $n \times n$ zijn. De $m \times n$ matrix $\underline{D}$ heeft de vorm:

$$\underline{D} = \left[\begin{array}{cccc|c} \varepsilon_1 & & & & \\ & \varepsilon_2 & & 0 & \\ & & \ddots & & 0 \\ & 0 & & \varepsilon_m & \end{array}\right] \tag{2.18}$$

De laatste $n-m$ kolommen van $\underline{D}$ zijn nullen, en de ε's heten de singuliere waarden van matrix $\underline{\Phi_q}$, zo gerangschikt dat $\varepsilon_1 \geq \varepsilon_2 \geq \ldots \varepsilon_m \geq 0$
Uit (2.17) volgt

$$\underline{\Phi_q}\, \underline{\Phi_q}^T = \underline{U}^T \underline{D}\, \underline{V}\, \underline{V}^T \underline{D}^T \underline{U} = \underline{U}^T \underline{D}\, \underline{D}^T \underline{U} \equiv \underline{U}^T \underline{\Lambda}\, \underline{U} \tag{2.19}$$

waarin $\underline{\Lambda}$ de diagonaalmatrix $\underline{D}\,\underline{D}^T = \mathrm{Diag}\,[\varepsilon_1^2, \varepsilon_2^2, \ldots, \varepsilon_m^2]$ is.
Dus:

$$\underline{\Phi_q}\, \underline{\Phi_q}^T \underline{U}^T = \underline{U}^T \underline{\Lambda}\, \underline{U}\, \underline{U}^T = \underline{U}^T \underline{\Lambda} \tag{2.20}$$

Dit betekent dat kolommen van $\underline{U}^T$ (rijen van $\underline{U}$) orthonormale eigenvectoren van de symmetrische matrix $\underline{\Phi_q}\, \underline{\Phi_q}^T$ zijn, en de ε_i^2 de corresponderende eigenwaarden.

Analoog:

$$\underline{\Phi}_q^T \underline{\Phi}_q = \underline{V}^T \underline{D}^T \underline{U} \underline{U}^T \underline{D} \underline{V} = \underline{V}^T \underline{D}^T \underline{D} \underline{V} \equiv \underline{V}^T \underline{\Omega} \underline{V} \qquad (2.21)$$

waarin Ω de diagonaalmatrix $\underline{D}^T\underline{D} = \mathrm{Diag}[\varepsilon_1^2, \varepsilon_2^2, \ldots \varepsilon_m^2, 0, \ldots, 0]$ is : de laatste n-m elementen op de diagonaal zijn nullen

$$\underline{\Phi}_q^T \underline{\Phi}_q \underline{V}^T = \underline{V}^T \underline{\Omega} \underline{V} \underline{V}^T = \underline{V}^T \underline{\Omega} \qquad (2.22)$$

Dit betekent dat kolommen van $\underline{V}^T$ (rijen van $\underline{V}$) orthonormale eigenvectoren van de symmetrische matrix $\underline{\Phi}_q^T \underline{\Phi}_q$ zijn, en de ε_i^2, gevolgd door n-m nullen, de corresponderende eigenwaarden

2.3 Het partitioneren van gegeneraliseerde coördinaten met SVD

Een nieuwe variabele $\underset{\sim}{z}$ wordt gedefinieerd als

$$\underset{\sim}{z} = \underline{V} \underset{\sim}{q} \qquad (2.23)$$

Deze orthogonale transformatie levert een nieuwe vector $\underset{\sim}{z}$ met gegeneraliseerde coördinaten voor het systeem.
De eerste tijdsafgeleide van (2.23) geeft: ($\underline{V}$=constant)

$$\dot{\underset{\sim}{z}} = \underline{V} \dot{\underset{\sim}{q}} \qquad (2.24)$$

De afgeleide hiervan geeft:

$$\ddot{\underset{\sim}{z}} = \underline{V} \ddot{\underset{\sim}{q}} \qquad (2.25)$$

Beschouw nu een verstoring $\delta \underset{\sim}{z}$ van $\underset{\sim}{z}$ die voldoet aan de randvoorwaarden $\underset{\sim}{\Phi}(\underset{\sim}{z}) = \underset{\sim}{0}$

$$\underline{\Phi}_q \delta \underset{\sim}{q} = \underline{\Phi}_q \underline{V}^T \delta \underset{\sim}{z} = \underset{\sim}{0} \qquad (2.26)$$

waarin $\delta q = V^T \delta z$ uit (2.23). Met (2.17) en het feit dat V orthonormaal is, volgt

$$U^T D\, \delta z = 0 \qquad (2.27)$$

Aangezien U orthonormaal is, mag men (2.27) voorvermenigvuldigen met U en gebruik maken van de vorm van D om te krijgen:

$$[\varepsilon_1 \delta z_1 \quad \varepsilon_2 \delta z_2 \quad \cdots \quad \varepsilon_m \delta z_m]^T = 0 \qquad (2.28)$$

Dit toont aan dat $\delta z_{m+1} \cdots \delta z_n$ niet berekend kunnen worden uit (2.27), en dus alleen uit de differentiaalvergelijkingen van beweging. Daarom worden $z_{m+1} \cdots z_n$ geselecteerd als onafhankelijke gegeneraliseerde coördinaten voor het oplossen van de bewegingsvergelijkingen en $z_1 \cdots z_m$ als afhankelijke gegeneraliseerde coördinaten die berekend moeten worden uit de randvoorwaarden.

We beschouwen nu alleen randvoorwaarden waarvoor geldt $\Phi_t = 0$, zodat uit (2.13) volgt:

$$\dot{\Phi} \equiv \Phi_q \dot{q} = 0 \qquad (2.29)$$

Voorvermenigvuldigen met U en gebruiken van de definitie van $\dot{z}$ in (2.24) levert

$$D \dot{z} = 0 \qquad (2.30)$$

Vanwege de speciale vorm van D is dit

$$[\varepsilon_1 \dot{z}_1 \quad \varepsilon_2 \dot{z}_2 \quad \cdots \quad \varepsilon_m \dot{z}_m]^T = 0 \qquad (2.31)$$

Aangezien de ε's niet nul zijn voor een Jacobiaan met rang m, betekent dit

$$[\dot{z}_1 \ \dot{z}_2 \ \cdots \ \dot{z}_m]^T = \underset{\sim}{0} \qquad (2.32)$$

Omdat orthonormale transformaties norm behouden,

$$\|\dot{\underset{\sim}{z}}\| = \|\dot{\underset{\sim}{q}}\| \qquad (2.33)$$

en $\dot{z}_1 \cdots \dot{z}_m$ nul zijn, volgt nu

$$\sum_{i=m+1}^{n} \dot{z}_i^{\,2} = \sum_{i=1}^{n} \dot{q}_i^{\,2} \qquad (2.34)$$

Voor een systeem met eenheidsmassa's en -traagheidsmomenten ($\underline{M} = \underline{I}$) geeft (2.34) aan dat de kinetische energie van de onafhankelijke samengestelde coördinaten de totale kinetische energie omvat. Dit levert dus een criterium om vast te stellen wanneer een nieuwe set onafhankelijke samengestelde coördinaten moet worden gedefinieerd.

Uit (2.23) en (2.24) kunnen de vector van virtuële verplaatsingen $\delta \underset{\sim}{q}$ en de snelheidsvector $\dot{\underset{\sim}{q}}$ uitgedrukt worden in termen van rijen van $\underline{V}$

$$\delta \underset{\sim}{q} = \sum_{j=m+1}^{n} \delta z_j \, \underset{\sim}{V}_j^T \qquad (2.35)$$

$$\dot{\underset{\sim}{q}} = \sum_{j=m+1}^{n} \dot{z}_j \, \underset{\sim}{V}_j^T \qquad (2.36)$$

Dit betekent dat verplaatsingen beperkt worden tot een deelruimte van R^n die wordt opgespannen door $\underset{\sim}{V}_{m+1}^T \cdots \underset{\sim}{V}_n^T$, en snelheden langs de $\underset{\sim}{V}_1^T \cdots \underset{\sim}{V}_m^T$-assen nul zijn.
Dit betekent dat de onafhankelijke samengestelde coördinaten alle systeem-informatie bevatten.
Matrix $\underline{V}$ kan nu gepartitioneerd worden in twee sub-matrices $\underline{VI}$ en $\underline{VD}$, die de onafhankelijke en afhankelijke

gedeelten van matrix $\underline{V}$ voorstellen:

$$\underline{V} = \begin{array}{c} \left[\begin{array}{c} \underline{VD} \\ \hline \underline{VI} \end{array}\right] \end{array} \begin{array}{l} m \\ n-m \end{array} \tag{2.37}$$

Nadat de onafhankelijke samengestelde posities en snelheden zijn berekend, kunnen de fysische coördinaten en snelheden berekend worden met matrixvergelijkingen van de vorm:

$$\left[\begin{array}{c} \underline{\Phi_t} \\ \hline \underline{VI} \end{array}\right] \underset{\sim}{X} = \underset{\sim}{b} \tag{2.38}$$

Hoofdstuk 3: De slinger - theoretisch

Om de theorie uit hoofdstuk 2 duidelijk te maken, wordt in dit hoofdstuk een eenvoudig voorbeeld uitgewerkt: de slinger uit figuur 3.1

Deze slinger bestaat uit lichaam 1 met massa m_1 en massatraagheidsmoment J_1, dat d.m.v. een starre massaloze staaf (met lengte l) verbonden is met de oorsprong. De staaf kan wrijvingsloos roteren om O

N.B. In dit voorbeeld is de Y-as anders gekozen dan in hoofdstuk 2.

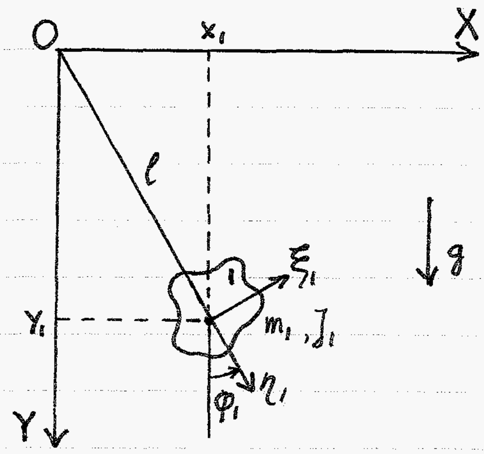

fig. 3.1

We gaan nu analoog aan hoofdstuk 2 de diverse vectoren en matrices bepalen en de vergelijkingen opstellen die het systeem vastleggen. Het aantal lichamen is 1, dus

$$NB = 1 \quad \Rightarrow n = 3 \tag{3.1}$$

Dus $$\underset{\sim}{q} = \underset{\sim}{q}' = [x_1\ y_1\ \varphi_1]^T \tag{3.2}$$

en $$\underset{\sim}{Q} = \underset{\sim}{Q}' = [Q_x'\ Q_y'\ Q_\varphi']^T = [0 \quad m_1 g \quad 0]^T \tag{3.3}$$

Aangezien lichaam 1 zich niet vrij door de ruimte kan bewegen, hebben we te maken met kinematische constraints

- de afstand van het massamiddelpunt van lichaam 1 tot O blijft steeds gelijk aan l $\Rightarrow x_1^2 + y_1^2 = l^2$
- de oriëntatie φ_1 van lichaam 1 blijft steeds gelijk aan de hoek die de staaf maakt met de Y-as $\Rightarrow \varphi_1 = \arctan\left(\frac{x_1}{y_1}\right)$ ofwel $\tan \varphi_1 = \frac{x_1}{y_1}$

We hebben dus te maken met twee kinematische randvoorwaarden

$$\Rightarrow \quad m = 2 \tag{3.4}$$

We kunnen deze voorwaarden vervangen door twee simpelere voorwaarden : $x_1 = l\sin\varphi_1$ en $y_1 = l\cos\varphi_1$, die we als volgt opbergen in de vector $\underset{\sim}{\Phi}$:

$$\underset{\sim}{\Phi} = \begin{bmatrix} x_1 - l\sin\varphi_1 \\ y_1 - l\cos\varphi_1 \end{bmatrix} \equiv \underset{\sim}{0} \tag{3.5}$$

$$\Rightarrow \quad \underline{\Phi_q} = \left[\partial\Phi_i / \partial q_j \right] = \begin{bmatrix} 1 & 0 & -l\cos\varphi_1 \\ 0 & 1 & l\sin\varphi_1 \end{bmatrix} \tag{3.6}$$

De systeemmassamatrix is:

$$\underline{M} = \begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_1 & 0 \\ 0 & 0 & J_1 \end{bmatrix} \tag{3.7}$$

Met (2.11) volgen de Lagrange-bewegingsvergelijkingen :

$$\begin{cases} m_1 \ddot{x}_1 + \lambda_1 = 0 \\ m_1 \ddot{y}_1 + \lambda_2 = m_1 g \\ J_1 \ddot{\varphi}_1 - \lambda_1 l\cos\varphi_1 + \lambda_2 l\sin\varphi_1 = 0 \end{cases} \tag{3.8}$$

Als we de slinger op tijdstip $t = t_0$ bslaten vanuit de positie $\varphi_1(t_0) = \varphi_1^\circ$ met beginsnelheid $\dot{\varphi}_1^\circ$ dan vinden we

$$\underset{\sim}{q}(t_0) = \underset{\sim}{q}^\circ = \left[l\sin\varphi_1^\circ \quad l\cos\varphi_1^\circ \quad \varphi_1^\circ \right]^T \tag{3.9}$$

$$\text{en} \quad \dot{\underset{\sim}{q}}(t_0) = \dot{\underset{\sim}{q}}^\circ = \left[l\dot{\varphi}_1^\circ\cos\varphi_1^\circ \quad -l\dot{\varphi}_1^\circ\sin\varphi_1^\circ \quad \dot{\varphi}_1^\circ \right]^T \tag{3.10}$$

Aangezien de tijd niet expliciet voorkomt in $\underset{\sim}{\Phi}$, geldt $\underset{\sim}{\Phi}_t = \underset{\sim}{0}$ en dus $\underline{\Phi_{tq}} = \underline{0}$ en $\Phi_{tt} = \underset{\sim}{0}$
Met (2.14) volgt nu de versnellingsvergelijking :

$$\begin{cases} \ddot{x}_1 - \ddot{\varphi}_1 l \cos\varphi_1 = -\dot{\varphi}_1^2 l \sin\varphi_1 \\ \ddot{y}_1 + \ddot{\varphi}_1 l \sin\varphi_1 = -\dot{\varphi}_1^2 l \cos\varphi_1 \end{cases} \tag{3.11}$$

waarbij gebruik is gemaakt van

$$\underset{\sim}{\dot{q}} = [\dot{x}_1 \;\; \dot{y}_1 \;\; \dot{\varphi}_1]^T \tag{3.12}$$

$$\underset{\sim}{\ddot{q}} = [\ddot{x}_1 \;\; \ddot{y}_1 \;\; \ddot{\varphi}_1]^T \tag{3.13}$$

$$\underline{\Phi}_q \underset{\sim}{\dot{q}} = \begin{bmatrix} \dot{x}_1 - \dot{\varphi}_1 l \cos\varphi_1 \\ \dot{y}_1 + \dot{\varphi}_1 l \sin\varphi_1 \end{bmatrix} \tag{3.14}$$

en $$\left(\underline{\Phi}_q \underset{\sim}{\dot{q}}\right)_q = \begin{bmatrix} 0 & 0 & \dot{\varphi}_1 l \sin\varphi_1 \\ 0 & 0 & \dot{\varphi}_1 l \cos\varphi_1 \end{bmatrix} \tag{3.15}$$

Volgens (2.15) kunnen we (3.8) en (3.11) combineren tot:

$$\begin{bmatrix} m_1 & 0 & 0 & 1 & 0 \\ 0 & m_1 & 0 & 0 & 1 \\ 0 & 0 & J_1 & -l\cos\varphi_1 & l\sin\varphi_1 \\ 1 & 0 & -l\cos\varphi_1 & 0 & 0 \\ 0 & 1 & l\sin\varphi_1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \ddot{x}_1 \\ \ddot{y}_1 \\ \ddot{\varphi}_1 \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} 0 \\ m_1 g \\ 0 \\ -\dot{\varphi}_1^2 l \sin\varphi_1 \\ -\dot{\varphi}_1^2 l \cos\varphi_1 \end{bmatrix} \tag{3.16}$$

Hieruit volgt één vergelijking in φ_1 :

$$(J_1 + m_1 l^2)\ddot{\varphi}_1 + m_1 g l \sin\varphi_1 = 0 \tag{3.17}$$

N.B.:
Dit resultaat kan ook gevonden worden door een eenvoudige evenwichtsbeschouwing om O : $I\ddot{\varphi} = \Sigma$ Momenten
waarbij het traagheidsmoment om O : $I = J_1 + m_1 l^2$

Vervolgens gaan we $\underline{\Phi}_q$ decomponeren volgens $\underline{\Phi}_q = \underline{U}^T \underline{D}\,\underline{V}$
Hiertoe bepalen we eerst de eigenwaarden van matrix $\underline{\Phi}_q \underline{\Phi}_q^T$

$$\underline{\Phi}_q \underline{\Phi}_q^T = \begin{bmatrix} 1 + l^2\cos^2\varphi_1 & -l^2\sin\varphi_1\cos\varphi_1 \\ -l^2\sin\varphi_1\cos\varphi_1 & 1+l^2\sin^2\varphi_1 \end{bmatrix} \qquad (3.18)$$

Eigenwaarden : $\det(\underline{\Phi}_q\underline{\Phi}_q^T - \lambda\underline{I}) = 0 \Rightarrow$

$$\lambda^2 - (2+l^2)\lambda + 1 + l^2 = 0 \Rightarrow$$

$$\begin{cases} \lambda_1 = 1+l^2 = \varepsilon_1^2 \Rightarrow \varepsilon_1 = \sqrt{1+l^2} \\ \lambda_2 = 1 = \varepsilon_2^2 \Rightarrow \varepsilon_2 = 1 \end{cases}$$

We kunnen nu de matrix $\underline{D}$ opschrijven :

$$\underline{D} = \begin{bmatrix} \sqrt{1+l^2} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad (3.19)$$

Eigenvectoren : $(\underline{\Phi}_q\underline{\Phi}_q^T - \lambda\underline{I})\,\underset{\sim}{a} = \underset{\sim}{0}$; $\|\underset{\sim}{a}\| = 1$

$$\lambda_1 = 1+l^2 \Rightarrow \underset{\sim}{a}_1 = [\cos\varphi_1 \;\; -\sin\varphi_1]^T$$
$$\lambda_2 = 1 \Rightarrow \underset{\sim}{a}_2 = [\sin\varphi_1 \;\; \cos\varphi_1]^T$$

We kunnen nu de matrix $\underline{U}$ opschrijven :

$$\underline{U} = \begin{bmatrix} \cos\varphi_1 & -\sin\varphi_1 \\ \sin\varphi_1 & \cos\varphi_1 \end{bmatrix} \qquad (3.20)$$

Nu gaan we de eigenwaarden van matrix $\underline{\Phi}_q^T\underline{\Phi}_q$ bepalen. Deze zijn echter dezelfde als die van $\underline{\Phi}_q\underline{\Phi}_q^T$, aangevuld met n-m nullen :

$$\begin{cases} \lambda_1 = 1+l^2 \\ \lambda_2 = 1 \\ \lambda_3 = 0 \end{cases}$$

Eigenvectoren : $(\underline{\Phi}_q^T\underline{\Phi}_q - \lambda\underline{I})\,\underset{\sim}{a} = \underset{\sim}{0}$; $\|\underset{\sim}{a}\| = 1$

$$\underline{\Phi}_q^T \underline{\Phi}_q = \begin{bmatrix} 1 & 0 & -l\cos\varphi_1 \\ 0 & 1 & l\sin\varphi_1 \\ -l\cos\varphi_1 & l\sin\varphi_1 & l^2 \end{bmatrix} \qquad (3.21)$$

$$\lambda_1 = 1 + l^2 \Rightarrow \underset{\sim}{a}_1 = \left[\frac{\cos\varphi_1}{\sqrt{1+l^2}} \;\; -\frac{\sin\varphi_1}{\sqrt{1+l^2}} \;\; -\frac{l}{\sqrt{1+l^2}} \right]^T$$

$$\lambda_2 = 1 \Rightarrow \underset{\sim}{a}_2 = \left[\sin\varphi_1 \;\; \cos\varphi_1 \;\; 0 \right]^T$$

$$\lambda_3 = 0 \Rightarrow \underset{\sim}{a}_3 = \left[\frac{l\cos\varphi_1}{\sqrt{1+l^2}} \;\; -\frac{l\sin\varphi_1}{\sqrt{1+l^2}} \;\; \frac{1}{\sqrt{1+l^2}} \right]^T$$

We kunnen nu de matrix $\underline{V}$ opschrijven :

$$\underline{V} = \begin{bmatrix} \frac{\cos\varphi_1}{\sqrt{1+l^2}} & -\frac{\sin\varphi_1}{\sqrt{1+l^2}} & -\frac{l}{\sqrt{1+l^2}} \\ \sin\varphi_1 & \cos\varphi_1 & 0 \\ \frac{l\cos\varphi_1}{\sqrt{1+l^2}} & -\frac{l\sin\varphi_1}{\sqrt{1+l^2}} & \frac{1}{\sqrt{1+l^2}} \end{bmatrix} \qquad (3.22)$$

We kunnen $\underline{V}$ als volgt partitioneren:

$$\underline{VD} = \begin{bmatrix} \frac{\cos\varphi_1}{\sqrt{1+l^2}} & -\frac{\sin\varphi_1}{\sqrt{1+l^2}} & -\frac{l}{\sqrt{1+l^2}} \\ \sin\varphi_1 & \cos\varphi_1 & 0 \end{bmatrix} \qquad (3.23)$$

$$\underline{VI} = \left[\frac{l\cos\varphi_1}{\sqrt{1+l^2}} \;\; -\frac{l\sin\varphi_1}{\sqrt{1+l^2}} \;\; \frac{1}{\sqrt{1+l^2}} \right] \qquad (3.24)$$

De nieuwe onafhankelijke samengestelde coördinaat ($n-m=1$, dus 1 vrijheidsgraad) volgt uit

$$\underset{\sim}{zI} = \underline{VI}\,\underset{\sim}{q} = \left[\frac{l\cos\varphi_1}{\sqrt{1+l^2}} x_1 - \frac{l\sin\varphi_1}{\sqrt{1+l^2}} y_1 + \frac{1}{\sqrt{1+l^2}} \varphi_1 \right] \qquad (3.25)$$

Deze nieuwe coördinaat is dus een lineaire combinatie van de drie fysische coördinaten x_1, y_1, en φ_1

De fysische betekenis van zI is eenvoudig in te zien wanneer we kijken naar zijn afgeleide, de onafhankelijke samengestelde snelheid $\dot{\underset{\sim}{zI}}$:

$$\dot{\underset{\sim}{z}}I = \left[\frac{l\cos\varphi_1}{\sqrt{1+l^2}} \cdot \dot{x}_1 - \frac{l\sin\varphi_1}{\sqrt{1+l^2}} \dot{y}_1 + \frac{1}{\sqrt{1+l^2}} \dot{\varphi}_1 \right] \qquad (3.26)$$

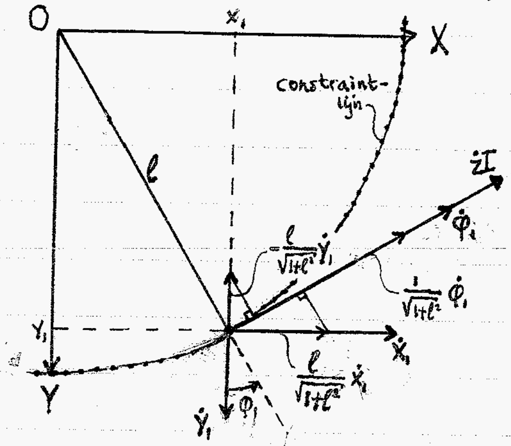

fig. 3.2

In de nevenstaande figuur is formule 3.26 grafisch uitgewerkt.
Hieruit blijkt dat de samengestelde snelheid $\dot{\underset{\sim}{z}}I$ gelijk is aan de som van de ontbondenen van de fysische snelheden langs de constraints, want

$$\dot{\underset{\sim}{z}}I = \left(\frac{l}{\sqrt{1+l^2}} \cdot \dot{x}_1\right) \cdot \cos\varphi_1 + \left(-\frac{l}{\sqrt{1+l^2}} \cdot \dot{y}_1\right) \cdot \sin\varphi_1 + \frac{1}{\sqrt{1+l^2}} \cdot \dot{\varphi}_1$$

M.a.w, bij het integreren van onafhankelijke samengestelde coördinaten beweegt het systeem zich langs een raaklijn aan de constraint-lijn.

Uit (3.17) bleek dat het systeem beschreven kon worden met alleen de variabele φ_1.
Dit volgt ook wanneer we in vergelijking 3.25 de constraints $x_1 = l\sin\varphi_1$ en $y_1 = l\cos\varphi_1$ betrekken:

$$\underset{\sim}{z}I = \frac{l^2\cos\varphi_1\sin\varphi_1}{\sqrt{1+l^2}} - \frac{l^2\sin\varphi_1\cos\varphi_1}{\sqrt{1+l^2}} + \frac{1}{\sqrt{1+l^2}} \varphi_1 \quad \Rightarrow$$

$$\underset{\sim}{z}I = \frac{1}{\sqrt{1+l^2}} \varphi_1 \qquad (3.27)$$

Analoog volgt nog voor de onafhankelijke samengestelde snelheid en versnelling:

$$\dot{\underset{\sim}{z}}I = \sqrt{1+l^2} \cdot \dot{\varphi}_1 \qquad (3.28)$$

$$\ddot{\underset{\sim}{z}}I = \sqrt{1+l^2} \cdot \ddot{\varphi}_1 \qquad (3.29)$$

Hoofdstuk 4 : Een algorithme voor het oplossen van de bewegingsvergelijkingen

Het algorithme dat in dit hoofdstuk wordt gepresenteerd, is grotendeels gebaseerd op het algorithme uit par. 7 van het artikel van Mani, Haug en Atkinson

Gegeneraliseerde coördinaten worden ~~samengesteld tot~~ onafhankelijke en afhankelijke gedeelten m.b.v. Singular Value Decomposition. Een subroutine, gebaseerd op een Adams-Bashforth predictor-corrector methode, wordt aangewend om de onafhankelijke samengestelde coördinaten te integreren. De fysische coördinaten worden hieruit berekend dmv. Newton-Raphson iteratie. Vervolgens worden de fysische snelheden bepaald uit de onafhankelijke samengestelde snelheden en tenslotte kan de vector van de fysische versnellingen opgelost worden

Het algorithme ziet er als volgt uit :

Het tijdsinterval dat wordt beschouwd is (t_0, t_{end}).
De index i geeft het huidige tijdstip aan, d.w.z.
$i=0$ betekent $t=t_0$

1. Lees de beginpositie, -snelheid en andere systeemdata. De gebruiker moet hierbij aangeven welke n-m posities $\underset{\sim}{v}$ en snelheden $\underset{\sim}{\dot{v}}$ volgens hem accuraat zijn. De Boolean matrices $\underline{B}_v$ en $\underline{B}_{\dot{v}}$ leggen de door de gebruiker aangegeven posities resp. snelheden vast op voorgeschreven waarden.

2. De positievector $\underset{\sim}{q}^{\circ}$ wordt nu gecorrigeerd m.b.v. Newton-Raphson iteratie :

$$\left[\begin{array}{c} \underline{\Phi}_q \; (\underset{\sim}{q}^{\circ}{}_{(k)}) \\ \hdashline \underline{B}_v \end{array}\right] \cdot \Delta \underset{\sim}{q}^{\circ}_{(k)} = \left[\begin{array}{c} -\underset{\sim}{\Phi}(\underset{\sim}{q}^{\circ}{}_{(k)}) \\ \hdashline \underset{\sim}{v} - \underline{B}_v \cdot \underset{\sim}{q}^{\circ}_{(k)} \end{array}\right] \tag{4.1}$$

$$\underset{\sim}{q}^{\circ}_{(k+1)} = \underset{\sim}{q}^{\circ}_{(k)} + \Delta \underset{\sim}{q}^{\circ}_{(k)} \qquad k = 1, \ldots \tag{4.2}$$

Hierin is k een iteratieteller. Bij iedere berekening van $\Delta \underset{\sim}{q}^{\circ}$ worden $\underline{\Phi}_q$ en $\underset{\sim}{\Phi}$ opnieuw berekend. Vergelijking (4.1) berekent achtereenvolgende correcties voor $\underset{\sim}{q}^{\circ}_{(k)}$ totdat aan alle kinematische constraints voldaan is met de gewenste nauwkeurigheid

3. Nu $\underset{\sim}{q}^{\circ}$ bekend is, kan $\underline{\Phi}_q^{\circ}$ berekend worden

4. Splits $\underline{\Phi}_q^{\circ}$ op m.b.v. Singular Value Decomposition:

$$\underline{\Phi}_q^{\circ} = \underline{U}^{\circ T} \underline{D}^{\circ} \underline{V}^{\circ} \tag{4.3}$$

Partitioneer $\underline{V}^{\circ}$ in onafhankelijke en afhankelijke delen:

$$\underline{V}^{\circ} = \left[\begin{array}{c} \underline{VD}^{\circ} \\ \hdashline \underline{VI}^{\circ} \end{array}\right] \tag{4.4}$$

5. De snelheidsvector $\dot{\underset{\sim}{q}}^{\circ}$ wordt nu berekend:

$$\left[\begin{array}{c} \underline{\Phi}_q^{\circ} \\ \hdashline \underline{B}_v \end{array}\right] \cdot \dot{\underset{\sim}{q}}^{\circ} = \left[\begin{array}{c} \underset{\sim}{0} \\ \hdashline \dot{\underset{\sim}{v}} \end{array}\right] \tag{4.5}$$

6. Bereken $\ddot{\underset{\sim}{q}}^{\circ}$ en $\underset{\sim}{\lambda}^{\circ}$ uit de versnellingsvergelijking:

$$\left[\begin{array}{c:c} \underline{M} & \underline{\Phi}_q^{\circ T} \\ \hdashline \underline{\Phi}_q^{\circ} & \underline{0} \end{array}\right] \left[\begin{array}{c} \ddot{\underset{\sim}{q}}^{\circ} \\ \hdashline \underset{\sim}{\lambda}^{\circ} \end{array}\right] = \left[\begin{array}{c} \underset{\sim}{Q} \\ -(\underline{\Phi}_q^{\circ} \cdot \dot{\underset{\sim}{q}}^{\circ})_q \cdot \dot{\underset{\sim}{q}}^{\circ} \end{array}\right] \tag{4.6}$$

7. Bereken de onafhankelijke samengestelde positie $\underset{\sim}{zI}^{\circ}$, snelheid $\dot{\underset{\sim}{zI}}^{\circ}$ en versnelling $\ddot{\underset{\sim}{zI}}$ volgens:

$$[\underset{\sim}{zI}^{\circ}\ \dot{\underset{\sim}{zI}}^{\circ}\ \ddot{\underset{\sim}{zI}}^{\circ}] = [\underline{VI}^{\circ}][\underset{\sim}{q}^{\circ}\ \dot{\underset{\sim}{q}}^{\circ}\ \ddot{\underset{\sim}{q}}^{\circ}] \tag{4.7}$$

8. Integreer $[\dot{\underset{\sim}{zI}}, \ddot{\underset{\sim}{zI}}]$ met $[\underset{\sim}{zI}^{\circ}, \dot{\underset{\sim}{zI}}^{\circ}]$ als begincondities om $[\underset{\sim}{zI}^{i+1}, \dot{\underset{\sim}{zI}}]$ op $t = t_{end}$ te krijgen, met behulp van Adams-Bashforth predictor-corrector integratie Hiertoe wordt een subroutine gestart die het interval (t_0, t_{end}) verdeelt in deelintervallen Δt_i zodat $t_{i+1} = t_i + \Delta t_i$.
Bij elke integratiestap worden de volgende onderdelen afgewerkt:

8a) Bereken d.m.v. integratie $[\underset{\sim}{zI}^{i+1}, \dot{\underset{\sim}{zI}}^{i+1}]$ met als begincondities $[\underset{\sim}{zI}^{i}, \dot{\underset{\sim}{zI}}^{i}]$.
De index i wordt nu met 1 opgehoogd zodat $t = t_{i+1} = t_i + \Delta t_i$

8b) Voorspel q^{i+1} en corrigeer dit iteratief m.b.v. Newton-Raphson totdat binnen de gewenste nauwkeurigheid wordt voldaan aan de constraints:

$$\left[\frac{\underline{\Phi}_q(\underset{\sim}{q}^{i+1}_{(k)})}{\underline{VI}}\right] \cdot \Delta\underset{\sim}{q}^{i+1}_{(k)} = \left[\frac{-\underset{\sim}{\Phi}(\underset{\sim}{q}^{i+1}_{(k)})}{\underset{\sim}{zI}^{i+1} - \underline{VI} \cdot \underset{\sim}{q}^{i+1}_{(k)}}\right] \tag{4.8}$$

$$\underset{\sim}{q}^{i+1}_{(k+1)} = \underset{\sim}{q}^{i+1}_{(k)} + \Delta\underset{\sim}{q}^{i+1}_{(k)} \qquad k = 1, \ldots \tag{4.9}$$

De positievector q^{i+1} kan voorspeld worden door $\underset{\sim}{q}^{i+1}_{(1)} = \underset{\sim}{q}^{i}$ te nemen. Bij iedere berekening van $\Delta\underset{\sim}{q}^{i+1}$ worden $\underline{\Phi}_q$ en $\underset{\sim}{\Phi}$ opnieuw berekend. De matrix $\underline{VI}$ blijft bij deze iteraties constant en dus gelijk aan $\underline{VI}^i$! Ook $\underset{\sim}{zI}^{i+1}$ blijft constant.

8c) Nu $\underset{\sim}{q}^{i+1}$ bekend is, kan $\underline{\Phi}_q^{i+1}$ berekend worden

8d) Splits $\underline{\Phi}_q^{i+1}$ op m.b.v. Singular Value Decomposition:

$$\underline{\Phi}_q^{\,i+1} = \underline{U}^{i+1^T}\,\underline{D}^{i+1}\,\underline{V}^{i+1} \qquad (4.10)$$

Partitioneer $\underline{V}^{i+1}$ in afhankelijke en onafhankelijke delen:

$$\underline{V}^{i+1} = \begin{bmatrix} \underline{VD}^{i+1} \\ \hline \underline{VI}^{i+1} \end{bmatrix} \qquad (4.11)$$

8e) Bereken $\dot{\underset{\sim}{q}}^{i+1}$ uit de snelheidsvergelijking:

$$\begin{bmatrix} \underline{\Phi}_q^{\,i+1} \\ \hline \underline{VI}^{i+1} \end{bmatrix} \cdot \dot{\underset{\sim}{q}}^{i+1} = \begin{bmatrix} \underset{\sim}{0} \\ \hline \dot{\underset{\sim}{zI}}^{i+1} \end{bmatrix} \qquad (4.12)$$

8f) Bereken de versnelling $\ddot{\underset{\sim}{q}}^{i+1}$ en de Lagrange-multiplicatoren $\underset{\sim}{\lambda}^{i+1}$ uit:

$$\begin{bmatrix} \underline{M} & \underline{\Phi}_q^{i+1^T} \\ \underline{\Phi}_q^{i+1} & \underline{0} \end{bmatrix} \begin{bmatrix} \ddot{\underset{\sim}{q}}^{i+1} \\ \underset{\sim}{\lambda}^{i+1} \end{bmatrix} = \begin{bmatrix} \underset{\sim}{Q} \\ -\underline{\left(\underline{\Phi}_q^{\,i+1} \cdot \dot{\underset{\sim}{q}}^{i+1}\right)_q}\, \dot{\underset{\sim}{q}}^{i+1} \end{bmatrix} \qquad (4.13)$$

8g) Bereken de onafhankelijke samengestelde positie, snelheid en versnelling volgens

$$\left[\underset{\sim}{zI}^{i+1} \;\; \dot{\underset{\sim}{zI}}^{i+1} \;\; \ddot{\underset{\sim}{zI}}^{i+1}\right] = \left[\underline{VI}^{i+1}\right]\left[\underset{\sim}{q}^{i+1} \;\; \dot{\underset{\sim}{q}}^{i+1} \;\; \ddot{\underset{\sim}{q}}^{i+1}\right] \qquad (4.14)$$

Bij deze stap worden $\underset{\sim}{zI}^{i+1}$ en $\dot{\underset{\sim}{zI}}$ dus opnieuw berekend. Indien $t_{i+1} = t_{end}$ wordt de subroutine beëindigd, zo niet, dan worden de stappen 8a) t/m 8g) herhaald.

Hoofdstuk 5 : De slinger – resultaten

Het algorithme uit hoofdstuk 4 wordt in dit hoofdstuk getoetst aan de hand van een eenvoudig voorbeeld. ~~Hiervoor is de slinger uit hoofdstuk 3 genomen.~~

Vergelijking (3.17) levert de bewegingsvergelijking:

$$(J_1 + m_1 l^2)\,\ddot{\varphi}_1 + m_1 g l \sin\varphi_1 = 0 \qquad (5.1)$$

Bij een kleine beginuitwijking φ_1^0, geldt de benadering $\sin\varphi_1 \approx \varphi_1$, wat leidt tot:

$$(J_1 + m_1 l^2)\,\ddot{\varphi}_1 + m_1 g l \varphi_1 = 0 \qquad (5.2)$$

Algemene oplossing:

$$\varphi_1 = A\cos\omega t + B\sin\omega t \qquad (5.3)$$

Ingevuld in (5.2) volgen de hoeksnelheid ω en de trillingstijd T:

$$\omega = \sqrt{\frac{m_1 g l}{J_1 + m_1 l^2}} \qquad (5.4)$$

$$T = 2\pi\sqrt{\frac{J_1 + m_1 l^2}{m_1 g l}} \qquad (5.5)$$

Beginvoorwaarden:

$$\varphi_1(t=0) = \varphi_1^0 \qquad (5.6)$$

$$\dot{\varphi}_1(t=0) = \dot{\varphi}_1^0 \qquad (5.7)$$

Ingevuld in (5.3) volgt voor de constanten A en B:

$$A = \varphi_1^\circ \quad , \quad B = \dot{\varphi}_1^\circ / \omega \tag{5.8}$$

Dus luidt de oplossing:

$$\varphi_1 = \varphi_1^\circ \sin \omega t + \frac{\dot{\varphi}_1^\circ}{\omega} \cos \omega t \tag{5.9}$$

met de hoeksnelheid ω als in (5.4)

De volgende waarden worden nu aangenomen:

$$\begin{aligned} m_1 &= 3.0 \text{ kg} \\ J_1 &= 2.0 \text{ kgm}^2 \\ l &= 0.75 \text{ m} \\ g &= 9.8 \text{ ms}^{-2} \\ \varphi_1^\circ &= \pi/6 \approx 0.5236 \text{ rad} \\ \dot{\varphi}_1^\circ &= 0.0 \text{ ms}^{-1} \end{aligned} \tag{5.10}$$

De beweging wordt nu beschreven door:

$$\varphi_1 = \pi/6 \sin \omega t \tag{5.11}$$

$$\omega = \sqrt{\frac{3*9.8*.75}{2+3*.75^2}} \approx 2.445 \text{ rad/s} \tag{5.12}$$

De slinger- of trillingstijd is:

$$T \approx 2.569 \text{ s} \tag{5.13}$$

Het algorithme kan nu getest worden door de uitwijking na één trilling te vergelijken met de beginuitwijking.
Beginpositie:

$$\underset{\sim}{q}(t=0) = \underset{\sim}{q}^\circ = \begin{bmatrix} x_1^\circ \\ y_1^\circ \\ \varphi_1^\circ \end{bmatrix} = \begin{bmatrix} 0.3750 \\ 0.6495 \\ 0.5236 \end{bmatrix} \tag{5.14}$$

Eindpositie:

$$\underset{\sim}{q}\,(t=2.57)=\begin{bmatrix} x_1 \\ y_1 \\ \varphi_1 \end{bmatrix}=\begin{bmatrix} 0.3737 \\ 0.6502 \\ 0.5217 \end{bmatrix} \qquad (5.15)$$

Deze uitwijking is dus kleiner dan de beginuitwijking.
We kunnen aan de snelheidsvector zien of er meer of minder dan één trilling is uitgevoerd.

$$\dot{\underset{\sim}{q}}\,(t=2.57)=\begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \\ \dot{\varphi}_1 \end{bmatrix}=\begin{bmatrix} 0.08939 \\ -0.05138 \\ 0.1375 \end{bmatrix} \qquad (5.16)$$

De hoeksnelheid $\dot{\varphi}_1$ is positief, d.w.z. dat er minder dan één trilling is uitgevoerd, m.a.w de berekende trillingstijd zal groter zijn dan de theoretische trillingstijd.
Dit verschil is grotendeels te wijten aan de benadering $\sin\varphi_1 \approx \varphi_1$. Overigens gaat het hier om een afwijking die kleiner is dan 1 %.

Bijlage I bevat de resultaten van deze berekening.
Contrôle van de tussentijdse berekening van $\underset{\sim}{q}$, $\dot{\underset{\sim}{q}}$ en $\ddot{q}$ toont aan dat er inderdaad één slingerbeweging wordt uitgevoerd.

Dit algorithme werd getest m.b.v. het programma SVD.FTN.
Toelichting bij dit programma wordt gegeven in hoofdstuk 6.

Hoofdstuk 6: Toelichting bij het programma

Het algorithme uit hoofdstuk 4 is verwerkt in het Fortran-programma SVD.FTN. De listing van dit programma is te vinden in bijlage II. In dit hoofdstuk wordt de werking ervan in het kort uitgelegd. Tevens wordt een voorbeeld-run (zie bijlage III) besproken en tenslotte wordt aangegeven welke wijzigingen moeten worden aangebracht als men een ander fysisch probleem wil aanpakken.

6.1 Het programma SVD.FTN

Allereerst worden de dimensies van het probleem vastgesteld: het aantal lichamen NB en het aantal constraints M wordt aangegeven. Vervolgens worden begin- en eindtijdstip ingelezen. Daarna wordt stap 1. uit hoofdstuk 4 uitgevoerd.
Vectoren worden in het programma voorgesteld door één-dimensionale arrays, matrices door tweedimensionale arrays. Zo wordt $\underset{\sim}{q}$ aangeduid met Q(I), $\underset{\sim}{v}$ met V(I), $\underline{B}_v$ met BV(I,J), $\underset{\sim}{\dot{q}}$ met QP(I), $\underset{\sim}{\dot{v}}$ met VP(I) en $\underline{B}_{\dot{v}}$ met BVP(I,J).
Om verwarring te voorkomen wordt de vector $\underset{\sim}{Q}$ aangeduid met QU(I) en de matrix $\underline{M}$ met MA(I,J)
Stap 2. uit hoofdstuk 4 wordt uitgevoerd m.b.v. de subroutine CALQ. Hierin zijn de vergelijkingen 4.1 en 4.2 verwerkt. Bij elke iteratie worden FI(I) $(= \underline{\Phi})$ en FQ(I,J) $(= \underline{\Phi}_q)$ opnieuw berekend door het aanroepen van de subroutine FIFQ. De berekening van Q(I) wordt stopgezet wanneer de "lengte" van DQ(I) $(= \Delta \underset{\sim}{q})$ kleiner is dan $0.1 \cdot 10^{-3}$
Vervolgens wordt de definitieve FQ berekend door nogmaals FIFQ aan te roepen (stap 3.)
Stap 4 is het uitvoeren van de Singular Value Decomposition in de subroutine SVD. Hierbij wordt VI(I,J) $(= \underline{VI})$ bepaald.

Het berekenen van QP(I) (stap 5.) geschiedt door de subroutine CALQP
Stap 6. wordt uitgevoerd door de subroutine CALQPP.
Hierbij wordt naast QPP(I) ook LAB(I) (= $\underset{\sim}{\lambda}$) uitgerekend.
Deze subroutine roept een andere subroutine aan, genaamd RL, waarin FQQP(I) (= de term "$-(\underline{\Phi}_q \cdot \underset{\sim}{\dot{q}})_q \cdot \underset{\sim}{\dot{q}}$" uit het rechterlid van vergelijking 4.6) wordt bepaald
Tenslotte wordt de subroutine CALZI aangeroepen (stap 7.)
Hierin worden ZI(I), ZPI(I) en ZPPI(I) (= resp. $\underset{\sim}{zI}$, $\underset{\sim}{\dot{z}I}$, $\underset{\sim}{\ddot{z}I}$) bepaald m.b.v. (4.7)
Het uitvoeren van stap 8. omvat het vaststellen van de begincondities Y(I), gevolgd door het aanroepen van de integratiesubroutine DØ2CAF*.
Na elke integratiestap roept deze subroutine op zijn beurt de zelfgeschreven subroutine FCN aan.
Deze subroutine werkt de volgende onderdelen af:

8a) De berekende waarden voor Y(I) worden aan ZI(I) en ZPI(I) toegekend
8b) Q(I) wordt berekend uit vgl (4.8) en (4.9). Aangezien deze vergelijkingen nagenoeg identiek zijn aan (4.1) en (4.2) kan hiervoor eveneens de subroutine CALQ gebruikt worden, waarbij enkel de argumenten BV en V zijn vervangen door VI en ZI.
8c) FQ wordt opnieuw berekend m.b.v. FIFQ
8d) VI wordt bepaald m.b.v. SVD
8e) QP(I) wordt berekend uit (4.12). Aangezien deze vergelijking nagenoeg identiek is aan (4.5) kan hiervoor eveneens de subroutine CALQP gebruikt worden, waarbij enkel de argumenten BVP en VP zijn vervangen door VI en ZPI

* De beschrijving van deze en enkele andere gebruikte subroutines uit de NAG-library is te vinden in bijlage IV

8f) QPP(I) wordt bepaald m.b.v CALQPP
8g) ZI(I), ZPI(I) en ZPPI(I) worden berekend m.b.v. CALZI
De stappen 8h) t/m 8j), niet vermeld in hoofdstuk 4, omvatten het opnieuw opstellen van de begincondities, alsmede het uitvoeren van de resultaten naar het array "RES" en naar het scherm.
Wanneer TEND is bereikt, is de subroutine DØ2CAF voltooid en volgt nog als 9e stap het opbergen van de resultaten in een uitvoerfile.

6.2 Een voorbeeld-run

Indien de SEG-file SVD.SEG aanwezig is, kan het programma gestart worden met het commando "SEG SVD". Op het scherm verschijnt: "GEEF TØ EN TEND". Deze twee waarden moeten nu worden ingegeven, gescheiden door een komma. Vervolgens worden de coördinaten van de beginpositie van alle lichamen opgevraagd (weer scheiden door komma's), waarna de complete vector van gegeneraliseerde coördinaten Q(I) op het scherm komt. Nu moet worden aangegeven, welke n-m elementen van deze vector accuraat zijn. De vector QP(I) wordt op dezelfde manier opgevraagd. Hierna wordt van alle lichamen de gegeneraliseerde krachtvector gevraagd, alsmede de massa en het traagheidsmoment.
Voordat de integratie begint, wordt de integratienauwkeurigheid TOL gevraagd. Het is aan te raden het programma meerdere keren te draaien, met toenemende nauwkeurigheid; bij $TOL = 10.0 \cdot 10^{-3}$ kan men een nauwkeurigheid van twee cijfers significant verwachten, bij $TOL = 10.0 \cdot 10^{-4}$ is dat drie cijfers, etc.
De gevraagde waarde STEP heeft alleen invloed op de uitvoerfile: twee opeenvolgende tijdstippen in deze file verschillen tenminste "STEP".

Vervolgens worden op elk tijdstip Q, QP, QPP, VI, ZI, ZPI en ZPPI afgedrukt.
Wanneer TEND is bereikt, wordt nog de naam van de uitvoerfile gevraagd, waarna het programma stopt.

6.3 Toepassing van het programma op andere fysische problemen

Wil men dit programma voor andere problemen gebruiken, dan moeten er enkele wijzigingen in worden aangebracht.
Wanneer de dimensies van zo'n probleem hetzelfde zijn, d.w.z. NB=1 en M=2, dan hoeven slechts de subroutines FIFQ en RL herschreven te worden, waarin $\underset{\sim}{\Phi}$, $\underline{\Phi_q}$ en $-(\underline{\Phi_q \cdot \dot{q}})_q \cdot \dot{q}$ worden berekend.
Zijn de dimensies anders, dan moeten tevens de dimensies van de diverse vectoren en matrices gewijzigd worden, alsmede de regels NB=1 en M=2.

- dimensies in de COMMON-statements:
  FI (M), Q (N), QP (N), VI (NMINM, N), Q (N), MA (N,N), LAB (M), FQQP (M), QPP (N), FQ (M,N), ZI (NMINM), ZPI (NMINM), ZPPI (NMINM), RES (100, (1+3*N+NMINM*N))
- dimensies in MAIN PROGRAM:
  NV (NMINM), NVP (NMINM), V (NMINM), BV (NMINM,N), VP (NMINM), BVP (NMINM,N), Y (NDV), W (NDV, 18)
- dimensies in subroutine CALQ
  A (N,N), BVVI (NMINM,N), C (N), DQ (N), AA (N,N), WS2 (N), WS3 (N), BVVIQ (NMINM), VZI (NMINM)
- dimensies in subroutine SVD
  MI (N,N), VT (N,N), S (N), UT (M,M), SV (M), V (N,N)
- dimensies in subroutine CALQP
  BVPVI (NMINM,N), VPZPI (NMINM), A (N,N), C (N), AA (N,N), WS2 (N), WS3 (N)
- dimensies in subroutine CALQPP
  E (NPM,NPM), H (NPM), QPPLAB (NPM), AA (N,N), WS4 (NPM), WS5 (NPM)

- dimensies in subroutine CALZI
  MQ(N,N), MZI(NMINM,N)

Hierbij geldt : N = 3*NB
NMINM = N - M
NPM = N+M
NDV = 2*NMINM

Het enige probleem dat kan ontstaan bij het werken met grotere dimensies, is de formattering van de diverse WRITE-statements. Deze zal in de meeste gevallen moeten worden aangepast.

Hoofdstuk 7 : Conclusies

De theorie, die aan de SVD-methode ten grondslag ligt, is vrij ingewikkeld. Hierdoor is de werking van deze methode niet onmiddellijk in te zien. Het voorbeeld van de slinger maakt echter in al zijn eenvoud een hoop zaken hieromtrent duidelijk. Men kan zich dan ook enigszins voorstellen wat er bij grootschalige systemen zal gebeuren wanneer men SVD toepast.
In het kort komt de SVD-methode hierop neer:
De m onafhankelijke samengestelde coördinaten $\underset{\sim}{zI}$ zijn lineaire combinaties van de n fysische coördinaten $\underset{\sim}{q}$ volgens $\underset{\sim}{zI} = \underline{VI}\,\underset{\sim}{q}$. Hierin is $\underline{VI}$ een mxn matrix die uit de m constraintvergelijkingen wordt bepaald.
Enkel de onafhankelijke samengestelde coördinaten hoeven geïntegreerd te worden. De fysische coördinaten kunnen weer hieruit worden berekend. Bij het integreren van $\underset{\sim}{zI}$ blijkt het systeem zich langs een raaklijn aan de constraints te bewegen.
De nauwkeurigheid die men kan bereiken met het algorithme uit hoofdstuk 4 is behoorlijk groot. Tevens is het aanpassen van het programma SVD.FTN vrij eenvoudig.
Al met al lijkt de "Singular Value Decomposition"-methode zeer goed bruikbaar in de multibody-dynamica
De vraag of deze methode ook toepasbaar is op ruimtelijke dynamica, zoals Mani, Haug en Atkinson claimen, viel buiten het kader van deze stage-opdracht; het lijkt echter aannemelijk, gezien de zeer algemene opbouw van de theorie.

Bijlage

T Q QP QPP

T	Q	QP	QPP
0.0000D 00			
	0.3750D 00	0.0000D 00	-0.1942D 01
	0.6495D 00	0.0000D 00	0.1121D 01
	0.5236D 00	0.0000D 00	-0.2990D 01
VI	0.5196D 00	-0.3000D 00	0.8000D 00
0.7149D-01			
	0.3705D 00	-0.1320D 00	-0.1941D 01
	0.6521D 00	0.7500D-01	0.1068D 01
	0.5167D 00	-0.2024D 00	-0.2954D 01
VI	0.5217D 00	-0.2964D 00	0.8000D 00
0.1360D 00			
	0.3580D 00	-0.2571D 00	-0.1935D 01
	0.6591D 00	0.1396D 00	0.9213D 00
	0.4976D 00	-0.3900D 00	-0.2854D 01
VI	0.5273D 00	-0.2864D 00	0.8000D 00
0.2005D 00			
	0.3374D 00	-0.3813D 00	-0.1911D 01
	0.6698D 00	0.1921D 00	0.6903D 00
	0.4666D 00	-0.5693D 00	-0.2690D 01
VI	0.5359D 00	-0.2699D 00	0.8000D 00
0.2650D 00			
	0.3089D 00	-0.5029D 00	-0.1850D 01
	0.6835D 00	0.2273D 00	0.3905D 00
	0.4244D 00	-0.7358D 00	-0.2462D 01
VI	0.5468D 00	-0.2471D 00	0.8000D 00
0.3296D 00			
	0.2726D 00	-0.6189D 00	-0.1732D 01
	0.6987D 00	0.2414D 00	0.4419D-01
	0.3719D 00	-0.8857D 00	-0.2173D 01
VI	0.5590D 00	-0.2181D 00	0.8000D 00
0.3941D 00			
	0.2292D 00	-0.7249D 00	-0.1541D 01
	0.7141D 00	0.2327D 00	-0.3169D 00
	0.3106D 00	-0.1015D 01	-0.1828D 01
VI	0.5713D 00	-0.1834D 00	0.8000D 00
0.4586D 00			
	0.1794D 00	-0.8159D 00	-0.1267D 01
	0.7282D 00	0.2011D 00	-0.6575D 00
	0.2416D 00	-0.1120D 01	-0.1431D 01
VI	0.5826D 00	-0.1436D 00	0.8000D 00
0.5649D 00			
	0.8662D-01	-0.9193D 00	-0.6464D 00

T	Q	QP	QPP
	0.7450D 00	0.1069D 00	-0.1075D 01
	0.1158D 00	-0.1234D 01	-0.6906D 00
VI	0.5960D 00	-0.6930D-01	0.8000D 00
0.6711D 00			
	-0.1340D-01	-0.9489D 00	0.1015D 00
	0.7499D 00	-0.1695D-01	-0.1199D 01
	-0.1786D-01	-0.1265D 01	0.1068D 00
VI	0.5999D 00	0.1072D-01	0.8000D 00
0.7774D 00			
	-0.1124D 00	-0.8985D 00	0.8299D 00
	0.7415D 00	-0.1362D 00	-0.9879D 00
	-0.1505D 00	-0.1212D 01	0.8965D 00
VI	0.5932D 00	0.8996D-01	0.8000D 00
0.8836D 00			
	-0.2022D 00	-0.7782D 00	0.1399D 01
	0.7222D 00	-0.2179D 00	-0.5123D 00
	-0.2730D 00	-0.1077D 01	0.1612D 01
VI	0.5778D 00	0.1618D 00	0.8000D 00
0.9899D 00			
	-0.2764D 00	-0.6089D 00	0.1747D 01
	0.6972D 00	-0.2414D 00	0.7733D-01
	-0.3774D 00	-0.8734D 00	0.2204D 01
VI	0.5578D 00	0.2211D 00	0.8000D 00
0.1096D 01			
	-0.3310D 00	-0.4136D 00	0.1901D 01
	0.6730D 00	-0.2034D 00	0.6192D 00
	-0.4570D 00	-0.6146D 00	0.2639D 01
VI	0.5384D 00	0.2648D 00	0.8000D 00
0.1202D 01			
	-0.3641D 00	-0.2089D 00	0.1940D 01
	0.6557D 00	-0.1160D 00	0.9906D 00
	-0.5069D 00	-0.3186D 00	0.2903D 01
VI	0.5245D 00	0.2913D 00	0.8000D 00
0.1309D 01			
	-0.3754D 00	-0.2440D-02	0.1943D 01
	0.6493D 00	-0.1411D-02	0.1123D 01
	-0.5242D 00	-0.3758D-02	0.2993D 01
VI	0.5194D 00	0.3003D 00	0.8000D 00
0.1415D 01			
	-0.3647D 00	0.2039D 00	0.1941D 01
	0.6554D 00	0.1135D 00	0.9966D 00
	-0.5078D 00	0.3111D 00	0.2907D 01
VI	0.5243D 00	0.2917D 00	0.8000D 00

T	Q	QP	QPP
0.1521D 01			
	-0.3321D 00	0.4088D 00	0.1903D 01
	0.6725D 00	0.2019D 00	0.6309D 00
	-0.4587D 00	0.6079D 00	0.2648D 01
VI	0.5380D 00	0.2657D 00	0.8000D 00
0.1627D 01			
	-0.2781D 00	0.6045D 00	0.1754D 01
	0.6965D 00	0.2413D 00	0.9205D-01
	-0.3799D 00	0.8678D 00	0.2217D 01
VI	0.5572D 00	0.2225D 00	0.8000D 00
0.1734D 01			
	-0.2045D 00	0.7747D 00	0.1412D 01
	0.7216D 00	0.2195D 00	-0.4983D 00
	-0.2761D 00	0.1074D 01	0.1630D 01
VI	0.5773D 00	0.1636D 00	0.8000D 00
0.1840D 01			
	-0.1151D 00	0.8967D 00	0.8483D 00
	0.7411D 00	0.1392D 00	-0.9794D 00
	-0.1540D 00	0.1210D 01	0.9174D 00
VI	0.5929D 00	0.9205D-01	0.8000D 00
0.1946D 01			
	-0.1619D-01	0.9493D 00	0.1227D 00
	0.7498D 00	0.2050D-01	-0.1200D 01
	-0.2159D-01	0.1266D 01	0.1291D 00
VI	0.5999D 00	0.1295D-01	0.8000D 00
0.2052D 01			
	0.8410D-01	0.9219D 00	-0.6284D 00
	0.7453D 00	-0.1040D 00	-0.1084D 01
	0.1124D 00	0.1237D 01	-0.6705D 00
VI	0.5962D 00	-0.6728D-01	0.8000D 00
0.2159D 01			
	0.1774D 00	0.8202D 00	-0.1256D 01
	0.7287D 00	-0.1997D 00	-0.6722D 00
	0.2388D 00	0.1126D 01	-0.1415D 01
VI	0.5830D 00	-0.1419D 00	0.8000D 00
0.2284D 01			
	0.2695D 00	0.6294D 00	-0.1722D 01
	0.6999D 00	-0.2423D 00	0.1310D-01
	0.3675D 00	0.8992D 00	-0.2149D 01
VI	0.5599D 00	-0.2156D 00	0.8000D 00
0.2388D 01			
	0.3252D 00	0.4414D 00	-0.1891D 01
	0.6758D 00	-0.2124D 00	0.5547D 00

T	Q	QP	QPP
	0.4485D 00	0.6531D 00	-0.2593D 01
VI	0.5407D 00	-0.2601D 00	0.8000D 00
0.2491D 01			
	0.3606D 00	0.2430D 00	-0.1940D 01
	0.6576D 00	-0.1332D 00	0.9470D 00
	0.5016D 00	0.3695D 00	-0.2875D 01
VI	0.5261D 00	-0.2885D 00	0.8000D 00
0.2570D 01			
	0.3737D 00	0.8939D-01	-0.1945D 01
	0.6502D 00	-0.5138D-01	0.1101D 01
	0.5217D 00	0.1375D 00	-0.2980D 01
VI	0.5202D 00	-0.2990D 00	0.8000D 00

Bijlage I

```
C       SVD.FTN


C       SVD.FTN
C       =======
C
C       MAIN PROGRAM
C       ------------
C
$INSERT SYSCOM>A$KEYS
C
        COMMON/BLOCK1/M,N,NMINM,NPM,NDV
        COMMON/BLOCK2/TO,TEND
        COMMON/BLOCK3/FI(2)
        COMMON/BLOCK4/TM
        COMMON/BLOCK5/Q(3)
        COMMON/BLOCK6/QP(3)
        COMMON/BLOCK7/VI(1,3)
        COMMON/BLOCK8/QU(3),MA(3,3),LAB(2),FQ3P(2)
        COMMON/BLOCK9/QPP(3)
        COMMON/BLOC10/FQ(2,3)
        COMMON/BLOC11/ZI(1),ZPI(1),ZPPI(1)
        COMMON/BLOC13/RES(100,13),II
C
        INTEGER*2 M,N,NMINM,NPM,NDV,II,NB,I,NV(1),NVP(1),IFAIL,
      % NOUT,INAME(10)
        REAL*8 TO,TEND,FI,TM,Q,QP,VI,QU,MA,LAB,FQQP,QPP,FQ,ZI,ZPI,ZPPI,
      % RES,V(1),BV(1,3),VP(1),BVP(1,3),Y(2),TOL,W(2,18)
        EXTERNAL FCN
C
        NB=1
        M=2
        N=3*NB
        NMINM=N-M
        NPM=N+M
        NDV=2*NMINM
C
C 1) INLEZEN VAN BEGINPOSITIE,-SNELHEID EN ANDERE SYSTEEMDATA
C
        WRITE(1,99999)
        READ(1,*) TO,TEND
        TM=TO
C
        DO 10 I=1,NB
           WRITE(1,99998) I,I,I
           READ(1,*) Q(3*I-2),Q(3*I-1),Q(3*I)
   10   CONTINUE
        WRITE(1,99997)
        DO 20 I=1,N
           WRITE(1,99996) Q(I)
   20   CONTINUE
C
        WRITE(1,99995) NMINM
        READ(1,*) (NV(I),I=1,NMINM)
        DO 30 I=1,NMINM
           V(I)=Q(NV(I))
           BV(I,NV(I))=1.0D0
   30   CONTINUE
C
        DO 40 I=1,NB
           WRITE(1,99994) I,I,I
           READ(1,*) QP(3*I-2),QP(3*I-1),QP(3*I)
   40   CONTINUE
```

```
C       SVD.FTN



        WRITE(1,99993)
        DO 50 I=1,N
           WRITE(1,99996) QP(I)
     50 CONTINUE
C
        WRITE(1,99992) NMINM
        READ(1,*) (NVP(I),I=1,NMINM)
        DO 60 I=1,NMINM
           VP(I)=QP(NVP(I))
           BVP(I,NVP(I))=1.0D0
     60 CONTINUE
C
        DO 70 I=1,NB
           WRITE(1,99991) I,I,I
           READ(1,*) QU(3*I-2),QU(3*I-1),QU(3*I)
     70 CONTINUE
C
        DO 80 I=1,NB
           WRITE(1,99990) I,I
           READ(1,*) MA((3*I-2),(3*I-2)),MA((3*I),(3*I))
           MA((3*I-1),(3*I-1))=MA((3*I-2),(3*I-2))
     80 CONTINUE
C
C 2) BEREKENING VAN Q
C
        CALL CALQ(BV,V)
C
C 3) BEREKENING VAN FQ
C
        CALL FIFQ
C
C 4) BEREKENING VAN VI
C
        CALL SVD
C
C 5) BEREKENING VAN QP
C
        CALL CALQP(BVP,VP)
C
C 6) BEREKENING VAN QPP
C
        CALL CALQPP
C
C 7) BEREKENING VAN ZI,ZPI,ZPPI
C
        CALL CALZI
C
C 8) STARTEN VAN DE INTEGRATIE-SUBROUTINE
C
        DO 90 I=1,NMINM
           Y(I*2-1)=ZI(I)
           Y(I*2)=ZPI(I)
     90 CONTINUE
C
        WRITE(1,99989)
        READ(1,*) TOL
        IFAIL=0
        CALL D02CAF(TM,TEND,NDV,Y,TOL,FCN,W,IFAIL)
        IF (IFAIL.GT.0) WRITE(1,99988)
        IF (TOL.LT.0.0D0) WRITE(1,99987)
```

```
C      SVD.FTN


C
C 9) RESULTATEN OPBERGEN IN UITVOERFILE
C
      NOUT=6
      WRITE(1,99986)
      READ(1,99985) (INAME(I),I=1,10)
      IF (EXST$A(INAME,INTS(20)))  CALL DELE$A(INAME,INTS(20))
      CALL OPEN$A(A$WRIT,INAME,INTS(20),INTS(2))
      WRITE(NOUT,99984)
      DO 100 I=1,II
         WRITE(NOUT,99983) RES(I,1)
         WRITE(NOUT,99982) RES(I,2),RES(I,5),RES(I,8)
         WRITE(NOUT,99982) RES(I,3),RES(I,6),RES(I,9)
         WRITE(NOUT,99982) RES(I,4),RES(I,7),RES(I,10)
         WRITE(NOUT,99981) RES(I,11),RES(I,12),RES(I,13)
  100 CONTINUE
      CALL CLOS$A(INTS(2))
      STOP
C
99999 FORMAT(1X/,'GEEF T0 EN TEND',/1X)
99998 FORMAT(1X/,'GEEF X(',I2,'),Y(',I2,'),PHI(',I2,')',/1X)
99997 FORMAT(1X/,'Q',/1X)
99996 FORMAT(6D12.4)
99995 FORMAT(1X/,'WELKE ',I2,' ELEMENTEN VAN Q ZIJN ACCURAAT?',/1X)
99994 FORMAT(1X/,'GEEF XP(',I2,'),YP(',I2,'),PHIP(',I2,')',/1X)
99993 FORMAT(1X/,'QP',/1X)
99992 FORMAT(1X/,'WELKE ',I2,' ELEMENTEN VAN QP ZIJN ACCURAAT?',/1X)
99991 FORMAT(1X/,'GEEF QUX(',I2,'),QUY(',I2,'),QUPHI(',I2,')',/1X)
99990 FORMAT(1X/,'GEEF M(',I2,'),J(',I2,')',/1X)
99989 FORMAT(1X/,'GEEF DE WAARDE VAN TOL',/1X)
99988 FORMAT('ERROR IN D02CAF')
99987 FORMAT('RANGE TOO SHORT FOR TOL')
99986 FORMAT('GEEF NAAM VAN UITVOERFILE',/1X)
99985 FORMAT(1X/,'GEEF NAAM VAN UITVOERFILE',/1X)
99985 FORMAT(10A2)
99984 FORMAT(' T',11X,'Q',11X,'QP',10X,'QPP')
99983 FORMAT(/,D12.4)
99982 FORMAT(12X,3D12.4)
99981 FORMAT(/,6X,'VI',4X,3D12.4)
C
      END
C
C*****************************************************************************
C
      SUBROUTINE FCN(T,Y,F)
      ---------------------
C
C
      COMMON/BLOCK1/M,N,NMINM,NPM,NDV
      COMMON/BLOCK2/T0,TEND
      COMMON/BLOCK3/FI(2)
      COMMON/BLOCK4/TM
      COMMON/BLOCK5/Q(3)
      COMMON/BLOCK6/QP(3)
      COMMON/BLOCK7/VI(1,3)
      COMMON/BLOCK9/QPP(3)
      COMMON/BLOC10/FQ(2,3)
      COMMON/BLOC11/ZI(1),ZPI(1),ZPPI(1)
      COMMON/BLOC13/RES(100,13),II
C
      INTEGER*2 M,N,NMINM,NPM,NDV,II,I,J
      REAL*8  T0,TEND,FI,TM,Q,QP,VI,QPP,FQ,ZI,ZPI,ZPPI,
```

```
C      SVD.FTN



       % RES,T,Y(NDV),F(NDV),STEP
C
        TM=T
        WRITE(1,99980) TM
C
C A) BEREKENING VAN ZI,ZPI
C
        WRITE(1,99979)
        DO 110 I=1,NMINM
           ZI(I)=Y(I*2-1)
           ZPI(I)=Y(I*2)
           WRITE(1,99978) ZI(I),ZPI(I)
  110 CONTINUE
        IF (II.EQ.0) GOTO 120
C
C B) BEREKENING VAN Q
C
        CALL CALQ(VI,ZI)
C
C C) BEREKENING VAN FQ
C
        CALL FIFQ
C
C D) BEREKENING VAN VI
C
        CALL SVD
C
C E) BEREKENING VAN QP
C
        CALL CALQP(VI,ZPI)
C
C F) BEREKENING VAN QPP
C
        CALL CALQPP
C
C G) BEREKENING VAN ZI,ZPI,ZPPI
C
        CALL CALZI
C
        GOTO 130
C
  120 WRITE(1,99977)
        READ(1,*) STEP
C
C H) RESULTATEN UITVOEREN NAAR SCHERM
C
  130 WRITE(1,99976)
        DO 140 I=1,N
           WRITE(1,99975) Q(I),QP(I),QPP(I)
  140 CONTINUE
        WRITE(1,99974)
        DO 150 I=1,NMINM
           WRITE(1,99973) (VI(I,J),J=1,N)
  150 CONTINUE
        WRITE(1,99972)
        DO 160 I=1,NMINM
           WRITE(1,99975) ZI(I),ZPI(I),ZPPI(I)
  160 CONTINUE
C
C I) BEGINCONDITIES VOOR INTEGRATIE
```

```
C     SVQ.FTN


C
      DO 170 I=1,NMINM
         Y(I*2-1)=ZI(I)
         Y(I*2)=ZPI(I)
         F(I*2-1)=Y(I*2)
         F(I*2)=ZPPI(I)
  170 CONTINUE
C
C J) RESULTATEN OPBERGEN IN ARRAY "RES"
C
  180 IF (II.EQ.0) GOTO 190
      IF (TM.GT.RES(II,1)+STEP) GOTO 190
      IF (TM.GT.RES(II,1)) GOTO 210
      II=II-1
      GOTO 180
  190 II=II+1
      RES(II,1)=TM
      DO 200 I=1,3
         RES(II,1+I)=Q(I)
         RES(II,4+I)=QP(I)
         RES(II,7+I)=QPP(I)
         RES(II,10+I)=VI(1,I)
  200 CONTINUE
      GOTO 220
  210 IF (TM.EQ.TEND) GOTO 190
  220 RETURN
C
99980 FORMAT(1X/,1X/,'T = ',D12.4,' SEC',/,'---------------------')
99979 FORMAT(1X/,' ZI',10X,'ZPI',/1X)
99978 FORMAT(2D12.4)
99977 FORMAT(1X/,'GEEF DE WAARDE VAN STEP',/1X)
99976 FORMAT(1X/,' Q',11X,'QP',10X,'QPP',/1X)
99975 FORMAT(3D12.4)
99974 FORMAT(1X/,' VI',/1X)
99973 FORMAT(6D12.4)
99972 FORMAT(1X/,' ZI',10X,'ZPI',9X,'ZPPI',/1X)
C
      END
C
C*****************************************************************************
C
      SUBROUTINE CALQ(BVVI,VZI)
      - ----------------------
C
C
      COMMON/BLOCK1/M,N,NMINM,NPM,NDV
      COMMON/BLOCK3/FI(2)
      COMMON/BLOCK5/Q(3)
      COMMON/BLOC10/FQ(2,3)
C
      INTEGER*2 M,N,NMINM,NPM,NDV,I,J,IFAIL,DIM,DIMWS1,OPT
      REAL*8 FI,Q,FQ,ACC,LENGDQ,A(3,3),BVVI(1,3),C(3),DQ(3),AA(3,3),
     % WS2(3),WS3(3),BVVIQ(1),VZI(1),WS1(1)
C
      ACC=0.1D-3
      LENGDQ=ACC
C
      DO 240 I=1,NMINM
         DO 230 J=1,N
            A(M+I,J)=BVVI(I,J)
  230    CONTINUE
```

```
  240 CONTINUE
C
  250 IF (LENGDQ.LT.ACC) GOTO 310
C
      CALL FIFQ
C
      DO 270 I=1,M
         DO 260 J=1,N
            A(I,J)=FQ(I,J)
  260    CONTINUE
  270 CONTINUE
C
      DO 280 I=1,M
         C(I)=-FI(I)
  280 CONTINUE
C
      DIM=1
      DIMWS1=1
      OPT=1
      IFAIL=0
      CALL F01CKF(BVVIQ,BVVI,Q,NMINM,DIM,N,WS1,DIMWS1,OPT,IFAIL)
      IF (IFAIL.GT.0) WRITE(1,99971)
C
      DO 290 I=1,NMINM
         C(M+I)=VZI(I)-BVVIQ(I)
  290 CONTINUE
C
      IFAIL=0
      CALL F04ATF(A,N,C,N,DQ,AA,N,WS2,WS3,IFAIL)
      IF (IFAIL.GT.0) WRITE(1,99970)
      LENGDQ=F05ABF(DQ,N)
C
      DO 300 I=1,N
         Q(I)=DQ(I)+Q(I)
  300 CONTINUE
C
      GOTO 250
C
  310 RETURN
C
99971 FORMAT('ERROR IN F01CKF')
99970 FORMAT('ERROR IN F04ATF')
C
      END
C
C******************************************************************************
C
      SUBROUTINE FIFQ
C     ---------------
C
      COMMON/BLOCK1/M,N,NMINM,NPM,NDV
      COMMON/BLOCK3/FI(2)
      COMMON/BLOCK4/TM
      COMMON/BLOCK5/Q(3)
      COMMON/BLOC10/FQ(2,3)
C
      INTEGER*2 M,N,NMINM,NPM,NDV
      REAL*8 FI,TM,Q,FQ,L
C
      L=0.75
```

```
C      SVD.FTN



C
       FI(1)=Q(1)-L*SIN(Q(3))
       FI(2)=Q(2)-L*COS(Q(3))
C
       FQ(1,1)=1.0D0
       FQ(1,2)=0.0D0
       FQ(1,3)=-L*COS(Q(3))
       FQ(2,1)=0.0D0
       FQ(2,2)=1.0D0
       FQ(2,3)=L*SIN(Q(3))
C
       RETURN
       END
C
C*******************************************************************************
C
       SUBROUTINE SVD
C      --------------
C
       COMMON/BLOCK1/M,N,NMINM,NPM,NDV
       COMMON/BLOCK7/VI(1,3)
       COMMON/BLOC10/FQ(2,3)
C
       INTEGER*2 M,N,NMINM,NPM,NDV,LWORK,IFAIL,I,N1,ICOL,J
       REAL*8 VI,FQ,
     % WORK(100),MI(3,3),VT(3,3),S(3),CC,GR,UT(2,2),SV(2),V(3,3)
C
       LWORK=3*M+M**2
       IFAIL=0
       CALL F02WCF(M,N,M,FQ,M,UT,M,SV,V,N,WORK,LWORK,IFAIL)
       IF (IFAIL.GT.0) WRITE(1,99969)
C
       DO 320 I=1,N
          MI(I,I)=1.0D0
  320  CONTINUE
C
       IFAIL=0
       CALL F01CLF(VT,MI,V,N,N,N,IFAIL)
       IF (IFAIL.GT.0) WRITE (1,99968)
C
       DO 330 I=1,NMINM
          VT(M+I,M+I)=1.0D0
  330  CONTINUE
C
       N1=1
       IFAIL=0
       CALL F05AAF(VT,N,N,N1,N,S,CC,ICOL,IFAIL)
       IF (IFAIL.GT.0) WRITE(1,99967)
       GR=0.99999
       IF (CC.GT.GR) WRITE (1,99966)
C
       DO 350 I=1,NMINM
          DO 340 J=1,N
             VI(I,J)=VT(J,M+I)
  340     CONTINUE
  350  CONTINUE
C
       RETURN
C
99969  FORMAT('ERROR IN F02WCF')
```

```
C       SVD.FTN



99968 FORMAT('ERROR IN F01CLF')
99967 FORMAT('ERROR IN F05AAF')
99966 FORMAT('AFH. KOL.  IN F05AAF')
C
      END
C
C*****************************************************************************
C
      SUBROUTINE CALQP(BVPVI,VPZPI)
C     ---------- ------------------
C
      COMMON/BLOCK1/M,N,NMINM,NPM,NDV
      COMMON/BLOCK6/QP(3)
      COMMON/BLOC10/FQ(2,3)
C
      INTEGER*2 M,N,NMINM,NPM,NDV,I,J,IFAIL
      REAL*8 QP,FQ,BVPVI(1,3),VPZPI(1),A(3,3),C(3),AA(3,3),WS2(3),
     % WS3(3)
C
      DO 370 I=1,NMINM
         DO 360 J=1,N
            A(M+I,J)=BVPVI(I,J)
  360    CONTINUE
  370 CONTINUE
C
      DO 390 I=1,M
         DO 380 J=1,N
            A(I,J)=FQ(I,J)
  380    CONTINUE
  390 CONTINUE
C
      DO 400 I=1,M
         C(I)=0.0D0
  400 CONTINUE
C
      DO 410 I=1,NMINM
         C(M+I)=VPZPI(I)
  410 CONTINUE
C
      IFAIL=0
      CALL F04ATF(A,N,C,N,QP,AA,N,WS2,WS3,IFAIL)
      IF (IFAIL.GT.0) WRITE(1,99965)
C
      RETURN
C
99965 FORMAT('ERROR IN F04ATF')
C
      END
C
C*****************************************************************************
C
      SUBROUTINE CALQPP
C     ---------- ------
C
      COMMON/BLOCK1/M,N,NMINM,NPM,NDV
      COMMON/BLOCK8/QU(3),MA(3,3),LAB(2),FQQP(2)
      COMMON/BLOCK9/QPP(3)
      COMMON/BLOC10/FQ(2,3)
C
      INTEGER*2 M,N,NMINM,NPM,NDV,I,J,IFAIL
```

```
C      SVD.FTN


       REAL*8 QU,MA,LAB,FQQP,QPP,FQ,
     % E(5,5),H(5),QPPLAB(5),AA(5,5),WS4(5),WS5(5)
C
       DO 430 I=1,N
       DO 420 J=1,N
             E(I,J)=MA(I,J)
  420     CONTINUE
  430 CONTINUE
C
       DO 450 I=1,M
          DO 440 J=1,N
             E(N+I,J)=FQ(I,J)
             E(J,N+I)=FQ(I,J)
  440     CONTINUE
  450 CONTINUE
C
       DO 460 I=1,N
          H(I)=QU(I)
  460 CONTINUE
C
       CALL RL
C
       DO 470 I=1,M
          H(N+I)=-FQQP(I)
  470 CONTINUE
C
       IFAIL=0
       CALL F04ATF(E,NPM,H,NPM,QPPLAB,AA,NPM,WS4,WS5,IFAIL)
       IF (IFAIL.GT.0) WRITE(1,99964)
C
       DO 480 I=1,N
          QPP(I)=QPPLAB(I)
  480 CONTINUE
C
       DO 490 I=1,M
          LAB(I)=QPPLAB(N+I)
  490 CONTINUE
C
       RETURN
C
99964 FORMAT('ERROR IN F04ATF')
C
       END
C
C*****************************************************************************
C
       SUBROUTINE RL
C      -------------
C
       COMMON/BLOCK1/M,N,NMINM,NPM,NDV
       COMMON/BLOCK4/TM
       COMMON/BLOCK5/Q(3)
       COMMON/BLOCK6/QP(3)
       COMMON/BLOCK8/QU(3),MA(3,3),LAB(2),FQQP(2)
C
       INTEGER*2 M,N,NMINM,NPM,NDV
       REAL*8 TM,Q,QP,QU,MA,LAB,FQQP,L
C
       L=0.75D0
       FQQP(1)=(QP(3)**2)*L*SIN(Q(3))
```

```
C       SVD.FIN



        FQQP(2)=L*COS(Q(3))*(QP(3)**2)
C
        RETURN
        END
C
C**************************************************************************************
C
        SUBROUTINE CALZI
C       ----------------
C
        COMMON/BLOCK1/M,N,NMINM,NPM,NDV
        COMMON/BLOCK5/Q(3)
        COMMON/BLOCK6/QP(3)
        COMMON/BLOCK7/VI(1,3)
        COMMON/BLOCK9/QPP(3)
        COMMON/BLOC11/ZI(1),ZPI(1),ZPPI(1)
C
        INTEGER*2 M,N,NMINM,NPM,NDV,I,IZ,OPT,IFAIL
        REAL*8 Q,QP,VI,QPP,ZI,ZPI,ZPPI,MQ(3,3),MZI(1,3)
C
        DO 500 I=1,N
           MQ(I,1)=Q(I)
           MQ(I,2)=QP(I)
           MQ(I,3)=QPP(I)
  500   CONTINUE
C
        IZ=1
        OPT=1
        IFAIL=0
        CALL FO1CKF(MZI,VI,MQ,NMINM,N,N,Z,IZ,OPT,IFAIL)
        IF (IFAIL.GT.0) WRITE(1,99963)
C
        DO 510 I=1,N
           ZI(I)=MZI(I,1)
           ZPI(I)=MZI(I,2)
           ZPPI(I)=MZI(I,3)
  510   CONTINUE
C
        RETURN
C
99963   FORMAT('ERROR IN FO1CKF')
C
        END
```

OK, SEC SVD

Bijlage I

OK, SEC SVD

GEEF TO EN TEND

0.0,2.57

GEEF X(1),Y(1),PHI(1)

0.4,0.6,0.5236

Q

0.4000D 00
0.6000D 00
0.5236D 00

WELKE 1 ELEMENTEN VAN Q ZIJN ACCURAAT?

3

GEEF XP(1),YP(1),PHIP(1)

0.0,0.0,0.0

QP

0.0000D 00
0.0000D 00
0.0000D 00

WELKE 1 ELEMENTEN VAN QP ZIJN ACCURAAT?

1

GEEF QUX(1),QUY(1),QUPHI(1)

0.0,29.4,0.0

GEEF M(1),J(1)

3.0,2.0

GEEF DE WAARDE VAN TOL

10.0D-04

T = 0.0000D 00 SEC

ZI ZPI

0.4189D 00 0.0000D 00

GEEF DE WAARDE VAN STEP

0.05

Q QP QPP

```
OK, SEC SVD



   0.3750D 00   0.0000D 00 -0.1942D 01
   0.6495D 00   0.0000D 00  0.1121D 01
   0.5236D 00   0.0000D 00 -0.2990D 01

  VI

   0.5196D 00 -0.3000D 00   0.8000D 00

  ZI             ZPI            ZPPI

   0.4189D 00   0.0000D 00 -0.3737D 01

 T =    0.0000D 00 SEC
 ------------------

  ZI             ZPI

   0.4189D 00   0.0000D 00

  Q              QP             QPP

   0.3750D 00   0.0000D 00 -0.1942D 01
   0.6495D 00   0.0000D 00  0.1121D 01
   0.5236D 00   0.0000D 00 -0.2990D 01

  VI

   0.5196D 00 -0.3000D 00   0.8000D 00

  ZI             ZPI            ZPPI

   0.4189D 00   0.0000D 00 -0.3737D 01

 T =    0.6868D-10 SEC
 ------------------

  ZI             ZPI

   0.4189D 00 -0.2567D-09

  Q              QP             QPP

   0.3750D 00 -0.1334D-09 -0.1942D 01
   0.6495D 00   0.7700D-10  0.1121D 01
   0.5236D 00 -0.2053D-09 -0.2990D 01

  VI

   0.5196D 00 -0.3000D 00   0.8000D 00

  ZI             ZPI            ZPPI

   0.4189D 00 -0.2567D-09 -0.3737D 01

 T =    0.1374D-09 SEC
 ------------------
```

! Nu wordt een sprong gemaakt naar het einde van de voorbeeld-run.

```
OK, SEC SVD


  0.3149D 00   0.4986D 00 -0.1878D 01
  0.6807D 00 -0.2306D 00   0.4253D 00
  0.4332D 00   0.7324D 00 -0.2510D 01

 VI

  0.5446D 00 -0.2519D 00   0.8000D 00

 ZI            ZPI           ZPPI

  0.3466D 00   0.9155D 00 -0.3138D 01

T =    0.2533D 01 SEC

 ZI            ZPI

  0.4394D 00   0.4434D 00

 Q             QP            QPP

  0.3645D 00   0.2325D 00 -0.1951D 01
  0.6355D 00 -0.1293D 00   0.9768D 00
  0.5075D 00   0.3547D 00 -0.2906D 01

 VI

  0.5244D 00 -0.2916D 00   0.8000D 00

 ZI            ZPI           ZPPI

  0.4060D 00   0.4434D 00 -0.3633D 01

T =    0.2533D 01 SEC

 ZI            ZPI

  0.4082D 00   0.4477D 00

 Q             QP            QPP

  0.3657D 00   0.2345D 00 -0.1956D 01
  0.6548D 00 -0.1310D 00   0.9820D 00
  0.5093D 00   0.3582D 00 -0.2915D 01

 VI

  0.5239D 00 -0.2925D 00   0.8000D 00

 ZI            ZPI           ZPPI

  0.4074D 00   0.4477D 00 -0.3644D 01

T =    0.2570D 01 SEC
```

```
OK, SEG SVD


 ZI             ZPI

  0.4223D 00   0.3125D 00

 Q              QP             QPP

  0.3734D 00   0.1626D 00 -0.1960D 01
  0.6504D 00 -0.9335D-01   0.1071D 01
  0.5212D 00   0.2500D 00 -0.2977D 01

 VI

  0.5203D 00 -0.2987D 00   0.8000D 00

 ZI             ZPI            ZPPI

  0.4169D 00   0.3125D 00 -0.3721D 01

GEEF NAAM VAN UITVOERFILE

S.R4

**** STOP
OK, SEG COMO -END
```

D02CAF – NAG FORTRAN Library Routine Document

NOTE: before using this routine, please read the appropriate implementation document to check the interpretation of ***bold italicised*** terms and other implementation–dependent details. The routine name may be precision–dependent.

1. Purpose

D02CAF integrates a system of first-order ordinary differential equations over a range with suitable initial conditions, using a variable-order variable-step Adams method.

2. Specification

```
      SUBROUTINE D02CAF (X, XEND, N, Y, TOL, FCN, W, IFAIL)
C     INTEGER N, IFAIL
C     real X, XEND, Y(N), TOL, W(N,18)
C     EXTERNAL FCN
```

3. Description

The routine integrates a system of ordinary differential equations

$$Y_i' = F_i(T, Y_1, Y_2, \ldots, Y_N) \qquad i = 1,2,\ldots,N$$

from T = X to T = XEND using a variable-order variable-step Adams method. The system is defined by a subroutine FCN supplied by the user, which evaluates F_i in terms of T and $Y_1, Y_2, \ldots, Y_N$ (see Section 5), and the values of $Y_1, Y_2, \ldots, Y_N$ must be given at T = X. The accuracy of the integration is controlled by the parameter TOL.

For a description of Adams methods and their practical implementation see [1]..

4. References

[1] HALL, G. and WATT, J.M. (eds.) Modern Numerical Methods for Ordinary Differential Equations. Clarendon Press, Oxford, 1976.

5. Parameters

X – ***real.***

Before entry, X must be set to the initial value of the independent variable T.

On exit, it contains XEND, unless an error has occurred, when it contains the value of T at the error.

XEND – ***real.***

On entry, XEND must specify the final value of the independent variable. If XEND < X on entry, integration will proceed in the negative direction.

Unchanged on exit.

N – INTEGER.

On entry, N must specify the number of differential equations.

Unchanged on exit.

Y – ***real*** array of DIMENSION at least (N).

Before entry, Y(1),Y(2),...,Y(N) must contain the initial values of the solution $Y_1, Y_2, \ldots, Y_N$.

On exit, Y(1),Y(2),...,Y(N) contain the computed values of the solution at the final value of T.

TOL – ***real.***

Before entry, TOL must be set to a **positive** tolerance for controlling the error in the integration.

The routine D02CAF has been designed so that for most problems a reduction in TOL leads to an approximately proportional reduction in the error in the solution at XEND. However, the actual relation between TOL and the accuracy achieved cannot be guaranteed. The user is strongly recommended to call D02CAF with more than one value for TOL and to compare the results obtained to estimate their accuracy. In the absence of any prior knowledge, the user might compare the results obtained by calling D02CAF with $TOL = 10.0^{-P}$ and $TOL = 10.0^{-P-1}$ where P correct decimal digits are required in the solution.

TOL is normally unchanged on exit. However if the range X to XEND is so short that a small change in TOL is unlikely to make any change in the computed solution, then, on return, TOL has its sign changed. This should be treated as a warning that the computed solution is likely to be more accurate than would be produced by using the same value of TOL on a larger range.

FCN – SUBROUTINE, supplied by the user.

FCN must evaluate the functions F_i (i.e. the derivatives Y_i') for given values of its arguments $T, Y_1, ..., Y_N$.

Its specification is:

```
SUBROUTINE FCN(T,Y,F)
real T,Y(n),F(n)
```

where n is the actual value of N in the call of D02CAF.

T – ***real.***

On entry, T specifies the value of the argument T.

Its value must not be changed.

Y – ***real*** array of DIMENSION (n).

On entry, Y(I) contains the value of the argument Y_I, for I = 1,...,n.

These values must not be changed.

F – ***real*** array of DIMENSION (n).

On exit, F(I) must contain the value of F_I, for I = 1,...,n.

FCN must be declared as EXTERNAL in the (sub)program from which D02CAF is called.

W – ***real*** array of DIMENSION (N,p), where $p \geq 18$.

Used as working space.

IFAIL – INTEGER.

On entry, IFAIL must be set to 0 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

Unless the routine detects an error (see next section), IFAIL contains 0 on exit.

6. Error Indicators and Warnings

Errors detected by the routine:–

IFAIL = 1

On entry, TOL $\leq$ 0.0
or N $\leq$ 0.

The latter error will cause a program breakdown with some compilers.

IFAIL = 2

With the given value of TOL, no further progress can be made across the integration range from the current point T = X, or the dependence of the error on TOL would be lost if further progress across the integration range were attempted (see Section 11 for a discussion of this error exit). The components Y(1),Y(2),...,Y(N) contain the computed values of the solution at the current point T = X.

IFAIL = 3

TOL is too small for the routine to start the integration (see Section 11). X and Y(1),Y(2),...,Y(N) retain their initial values.

IFAIL = 4

A serious error has occurred in an internal call to D02QAF . Check all subroutine calls and array dimensions. Seek expert help.

7. Auxiliary Routines

Details are distributed to sites in machine-readable form.

8. Timing

This depends on the complexity and mathematical properties of the system of differential equations defined by FCN, on the length of the range, and on the tolerance. There is also a small overhead of the form A+B×N, where A and B are machine-dependent computing terms.

9. Storage

The storage required by internally declared arrays is 32 ***real*** elements.

10. Accuracy

The accuracy depends on TOL, on the mathematical properties of the differential system, on the length of the range of integration and on the method. It can be controlled by varying TOL but the approximate proportionality of the error to TOL holds only for a restricted range of values of TOL. For TOL too large, the underlying theory may break down and the result of varying TOL may be unpredictable. For TOL too small, rounding errors may affect the solution significantly and an error exit with IFAIL = 2 or IFAIL = 3 is possible.

The user who requires a more reliable estimate of the accuracy achieved than can be obtained by varying TOL, is recommended to call the routine D02BDF where both the solution and a global error estimate are computed.

11. Further Comments

If the routine fails with IFAIL = 3, then it could be called again with a larger value of TOL if this has not already been tried. If the accuracy requested is really needed and cannot be obtained with this routine, then the system may be very stiff (see below) or so badly scaled that it cannot be solved to the required accuracy.

If the routine fails with IFAIL = 2, it is probable that it has been called with a value of TOL which is so small that a solution cannot be obtained on the range X to XEND. This can happen for well-behaved systems and very small values of TOL. The user should, however, consider whether there is a more fundamental difficulty. For example,

(i) in the region of a singularity (infinite value) of the solution, the routine will usually stop with IFAIL = 2, unless overflow occurs first. If overflow occurs using D02CAF , routine D02QAF can be used instead to trap the increasing solution before overflow occurs. In any case, numerical integration cannot be continued through a singularity, and analytical treatment should be considered;

(ii) for 'stiff' equations, where the solution contains rapidly decaying components, the routine will use very small steps in T (internally to D02CAF) to preserve stability. This will exhibit itself by making the computing time excessively long, or occasionally by an exit with IFAIL = 2. Adams methods are not efficient in such cases and the user should try the Gear method D02EAF .

Users with problems for which D02CAF is not sufficiently general should consider the routines D02CBF , D02CHF and D02QAF . Routine D02CBF can be used when output is required as points inside the range X to XEND (for example, for graph-plotting purposes) or more general error control is required. Use of D02CBF should be computationally more efficient than repeated calls to D02CAF to achieve the same result. Routine D02CHF can be used to calculate where a function of the components $Y_1, Y_2, \ldots, Y_N$ and their derivatives takes a specified value.

D02QAF is a more general Adams routine with many facilities including more general error control options and several criteria for interrupting the calculations.

12. Keywords

Adams Method,
Initial Value Problems,
Ordinary Differential Equations.

---

13. Example

To integrate the following equations (for a projectile)

$$y' = \tan(\phi)$$

$$v' = -0.032\frac{\tan(\phi)}{v} - 0.02 \times v \times \sec(\phi)$$

$$\phi' = -0.032/v^2$$

over an interval X = 0.0 to XEND = 8.0 starting with values y = 0.0, v = 0.5 and $\phi = \pi/5$. We write y = Y(1), v = Y(2) and ϕ = Y(3) and we set TOL = 1.0E–4 and TOL = 1.0E–5 in turn so that we may compare the solutions obtained. The value of π is obtained by using X01AAF .

13.1. Program Text

WARNING: This **single precision** example program may require amendment for certain implementations. The results produced may not be the same. If in doubt, please seek further advice (see **Essential Introduction** to the Library Manual).

```
C     D02CAF EXAMPLE PROGRAM TEXT
C     MARK 7 RELEASE. NAG COPYRIGHT 1978.
C     .. LOCAL SCALARS ..
      REAL PI, TOL, X, XEND
      INTEGER I, IFAIL, J, N, NOUT
C     .. LOCAL ARRAYS ..
      REAL W(3,18), Y(3)
```

```
C     .. FUNCTION REFERENCES ..
      REAL X01AAF
C     .. SUBROUTINE REFERENCES ..
C     D02CAF
C     ..
      EXTERNAL FCN
      DATA NOUT /6/
      WRITE (NOUT,99995)
      N = 3
      PI = X01AAF(PI)
      DO 20 J=4,5
         TOL = 10.**(-J)
         X = 0.E0
         XEND = 8.E0
         Y(1) = 0.E0
         Y(2) = 0.5E0
         Y(3) = 0.2E0*PI
         IFAIL = 1
         WRITE (NOUT,99998) TOL
         WRITE (NOUT,99999)
         WRITE (NOUT,99997) X, (Y(I),I=1,3)
         CALL D02CAF(X, XEND, N, Y, TOL, FCN, W, IFAIL)
         WRITE (NOUT,99997) X, (Y(I),I=1,3)
         IF (TOL.LT.0.) WRITE (NOUT,99994)
         WRITE (NOUT,99996) IFAIL
   20 CONTINUE
      STOP
99999 FORMAT (43H0  X          Y(1)          Y(2)          Y(3))
99998 FORMAT (5H0TOL=, E8.1)
99997 FORMAT (1H , F6.3, 3E13.5)
99996 FORMAT (8H  IFAIL=, I1)
99995 FORMAT (4(1X/), 31H D02CAF EXAMPLE PROGRAM RESULTS/1X)
99994 FORMAT (24H RANGE TOO SHORT FOR TOL)
      END
      SUBROUTINE FCN(T, Y, F)
C     .. SCALAR ARGUMENTS ..
      REAL T
C     .. ARRAY ARGUMENTS ..
      REAL F(3), Y(3)
C     ..
C     .. FUNCTION REFERENCES ..
      REAL COS, SIN
C     ..
      F(1) = SIN(Y(3))/COS(Y(3))
      F(2) = -0.032E0*F(1)/Y(2) - 0.02E0*Y(2)/COS(Y(3))
      F(3) = -0.032E0/(Y(2)*Y(2))
      RETURN
      END
```

13.2. Program Data

None.

13.3. Program Results

```
 D02CAF EXAMPLE PROGRAM RESULTS

 TOL= 0.1E-03
   X          Y(1)          Y(2)          Y(3)
  0.000  0.00000E+00  0.50000E+00  0.62832E+00
  8.000 -0.12455E+01  0.51293E+00 -0.85383E+00
  IFAIL=0
 TOL= 0.1E-04
   X          Y(1)          Y(2)          Y(3)
  0.000  0.00000E+00  0.50000E+00  0.62832E+00
  8.000 -0.12460E+01  0.51300E+00 -0.85371E+00
  IFAIL=0
```

1. Purpose

F01CKF returns with the result of the multiplication of two matrices B and C in the matrix A, with the option to overwrite B or C.

IMPORTANT: before using this routine, read the appropriate machine implementation document to check the interpretation of italicised terms and other implementation-dependent details.

2. Specification (FORTRAN IV)

```
      SUBROUTINE F01CKF(A,B,C,N,P,M,Z,IZ,OPT,IFAIL)
C     INTEGER N,P,M,IZ,OPT,IFAIL
C     real A,B,C,Z
C     DIMENSION A(N,P),B(N,M),C(M,P),Z(IZ)
```

3. Description

The $n \times m$ matrix B is post-multiplied by the $m \times p$ matrix C. If OPT=1 the result is formed in the $n \times p$ matrix A. If OPT=2, m must equal p, and the result is written back to B. If OPT=3, n must equal m, and the result is written back to C.

4. References None.

5. Parameters

A - *real* array of DIMENSION (N,P).
On exit, if OPT=1, A contains the result of the matrix multiplication.

B - *real* array of DIMENSION (N,M).
Before entry, all elements of B must be assigned a value.
On exit, if OPT=2, B contains the result of the multiplication.
Otherwise B is unchanged on exit.

C - *real* array of DIMENSION (M,P).
Before entry, all elements of C must be assigned a value.
On exit, if OPT=3, C contains the result of the multiplication.
Otherwise C is unchanged on exit.

N - INTEGER.
On entry, N specifies n, the first dimension of A and B as declared in the calling (sub)program. If OPT=3, n must equal m. Unchanged on exit.

P - INTEGER.
On entry, P specifies p, the second dimension of A and C as declared in the calling (sub)program. If OPT=2, p must equal m. Unchanged on exit.

F01CKF

5. Parameters (contd)

M - INTEGER.
On entry, M specifies m, the second dimension of B and first dimension of C as declared in the calling (sub)program. Unchanged on exit.

Z - *real* array of DIMENSION (IZ).
Used as working space.

IZ - INTEGER.
On entry, IZ specifies the dimension of Z. If OPT=1, IZ may be 1 otherwise IZ must be greater than or equal to m. Unchanged on exit.

OPT - INTEGER.
On entry, the value of OPT determines which array is to contain the final result.

a) OPT=1. A must be distinct from B and C and, on exit, contains the result. B and C need not be distinct in this case.

b) OPT=2. B must be distinct from C and on exit, contains the result. A is not used in this case and need not be distinct from B or C.

c) OPT=3. C must be distinct from B and on exit, contains the result. A is not used in this case and need not be distinct from B or C.

OPT is unchanged on exit.

IFAIL - INTEGER.
Before entry, IFAIL must be assigned a value. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0. Unless the routine detects an error (see Section 6), IFAIL contains 0 on exit.

6. Error Indicators

Errors detected by the routine:-

IFAIL = 1   On entry, M or P or N $\le$ 0.

IFAIL = 2   OPT=2 and M $\ne$ P.

IFAIL = 3   OPT=3 and N $\ne$ M.

IFAIL = 4   OPT$\ne$1 and IZ < M.

7. Auxiliary Routines

This routine calls the NAG Library routine P01AAF.

8. Timing

The timing increases with m,p and n.

9. Storage

There are no internally declared arrays.

10. Accuracy

Inner-products are accumulated using *additional precision*.

11. Further Comments

None.

12. Keywords

Matrix Multiplication.

13. Example

The example program multiplies the 2 × 3 matrix B and the 3 × 2 matrix C together and places the result in the 2 × 2 matrix A.

Program

This single precision example program may require amendment
 i) for use in a DOUBLE PRECISION implementation
ii) for use in either precision in certain implementations.
The results produced may differ slightly.

```
C     F01CKF EXAMPLE PROGRAM TEXT
C     NAG COPYRIGHT 1975
C     MARK 4.5 REVISED
      REAL A(2,2), B(2,3), C(3,2), Z(1)
      INTEGER I, IFAIL, NOUT
      DATA NOUT /6/
      WRITE (NOUT,99999)
      DO 20 I=1,3
         B(1,I) = FLOAT(I) - 1.
         C(I,1) = B(1,I)
         B(2,I) = FLOAT(I)
         C(I,2) = B(2,I)
   20 CONTINUE
      IFAIL = 0
      CALL F01CKF(A, B, C, 2, 2, 3, Z, 1, 1, IFAIL)
      IF (IFAIL.GT.0) GO TO 40
      WRITE (NOUT,99998)
      WRITE (NOUT,99997) A(1,1), A(1,2), A(2,1), A(2,2)
      STOP
```

13. Example

Program (contd)

```
   40 WRITE (NOUT,99996)
      STOP
99999 FORMAT (4(1X/), 31H F01CKF EXAMPLE PROGRAM RESULTS, 2(/1X))
99998 FORMAT (9HOMATRIX A/1X)
99997 FORMAT (1H , 2F7.1)
99996 FORMAT (16HOERROR IN F01CKF)
      END
```

Results

```
 F01CKF EXAMPLE PROGRAM RESULTS


 MATRIX A

     5.0    8.0
     8.0   14.0
```

F02WCF – NAG FORTRAN Library Routine Document

NOTE: before using this routine, please read the appropriate implementation document to check the interpretation of bold ***italicised*** terms and other implementation–dependent details. The routine name may be precision–dependent.

1. Purpose

F02WCF computes the singular values and left– and right-hand singular vectors of a real rectangular $m \times n$ matrix A, $A = QDP^T$, where $Q^TQ = P^TP = I_k$, $k = \min(m,n)$ and $D = \text{diag}(sv_1, sv_2, \ldots, sv_k)$ with $sv_1 \geq sv_2 \geq \ldots \geq sv_k \geq 0$.

2. Specification

```
      SUBROUTINE F02WCF (M, N, MINMN, A, NRA, Q, NRQ, SV, PT,
     1   NRPT, WORK, LWORK, IFAIL)
C     INTEGER M, N, MINMN, NRA, NRQ, NRPT, LWORK, IFAIL
C     real A(NRA,N), Q(NRQ,MINMN), SV(MINMN), PT(NRPT,N),
C    1   WORK(LWORK)
```

3. Description

A real $m \times n$ matrix A may be factorised by the singular value decomposition (SVD) as:

$$A = Q\begin{bmatrix} D \\ 0 \end{bmatrix}P^T \text{ if } m \geq n$$

Or

$$A = Q[D\ 0]P^T \text{ if } m \leq n.$$

Here Q is an $m \times m$ orthogonal matrix, P is an $n \times n$ orthogonal matrix, and D is a diagonal matrix of order $k = \min(m,n)$, whose non-negative diagonal elements are the singular values of A.

Let $\tilde{Q}$ be the $m \times k$ matrix consisting of the first k columns of Q – these are the left-hand singular vectors of A. Let $\tilde{P}$ be the $n \times k$ matrix consisting of the first k columns of P – these are the right-hand singular vectors of A. Then

$$A = \tilde{Q}D\tilde{P}^T$$

This routine returns $\tilde{Q}$ and $\tilde{P}^T$ as well as the diagonal elements of D, arranged in descending order.

If the matrix A is of rank r then in exact arithmetic $sv_{r+1} = sv_{r+2} = \ldots = sv_k = 0$, $k = \min(m,n)$.

The routine first reduces A to upper triangular form by Householder transformations when $m \geq n$ and by Givens plane rotations when $m < n$, the upper triangular form is then reduced to bidiagonal form by Givens plane rotations and finally the QR algorithm is used to obtain the SVD of the bidiagonal form.

4. References

[1] WILKINSON, J.H.
Singular – Value Decomposition – Basic Aspects.
In 'Numerical Software – Needs and Availability'. Ed. JACOBS D.A.H., Academic Press, London, 1978.

5. Parameters

M – INTEGER.

On entry, M must specify the number of rows of A. M ≥ 1.

Unchanged on exit.

N – INTEGER.

On entry, N must specify the number of columns of A. N ≥ 1.

Unchanged on exit.

MINMN – INTEGER.

On entry, MINMN must specify the minimum of M and N.

Unchanged on exit.

A – ***real*** array of DIMENSION (NRA,r)

where r ≥ N.
Before entry, the leading M × N part of A must contain the matrix to be factorised.

Unchanged on exit, **unless** the routine is called with the same array supplied for both A and Q or for both A and PT.

NRA – INTEGER.

On entry, NRA must specify the first dimension of A as declared in the calling (sub)program. NRA $\geq$ M.

Unchanged on exit.

Q – *real* array of DIMENSION (NRQ,s)

where s $\geq$ MINMN .

On successful exit, the leading M $\times$ MINMN part of Q contains the MINMN left-hand singular vectors of A, stored by **columns.**

The routine may be called with the same array supplied for both A and Q, provided that a different array is supplied for PT. In this case the leading M $\times$ MINMN part of A is overwritten by Q, and NRQ must be equal to NRA.

NRQ – INTEGER.

On entry, NRQ must specify the first dimension of Q as declared in the calling (sub)program. NRQ $\geq$ M.

Unchanged on exit.

SV – *real* array of DIMENSION at least (MINMN).

On successful exit, SV contains the MINMN singular values of A arranged in descending order.

PT – *real* array of DIMENSION (NRPT,t) where t $\geq$ N.

On successful exit, the leading MINMN $\times$ N part of PT contains the MINMN right-hand singular vectors of A, stored by **rows.**

The routine may be called with the same array supplied for both A and PT, provided that a different array is supplied for Q. In this case, the leading MINMN $\times$ N part of A is overwritten by PT, and NRPT must be equal to NRA.

NRPT – INTEGER.

On entry, NRPT must specify the first dimension of PT as declared in the calling (sub)program. NRPT $\geq$ MINMN.

Unchanged on exit.

WORK – *real* array of DIMENSION (LWORK).

On successful exit, WORK(1) contains the total number of iterations taken by the QR algorithm. Otherwise WORK is used as workspace.

LWORK – INTEGER.

On entry, LWORK must specify the length of the array WORK as declared in the calling (sub)program. LWORK must be at least 3$\times$MINMN, but unless M is close to N and provided that sufficient storage is available, then it is strongly recommended that LWORK be at least (3 $\times$MINMN+MINMN^2).

If M is not close to N then the routine is likely to be considerably faster with the larger value of LWORK. See Section 8.

Unchanged on exit.

IFAIL – INTEGER.

Before entry, IFAIL must be assigned a value. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

Unless the routine detects an error (see next section), IFAIL contains 0 on exit.

6. Error Indicators and Warnings

Errors detected by the routine:–

IFAIL = 1

On entry,

M < 1, or
N < 1, or
MINMN$\neq$min(M,N), or
NRA < M, or
NRQ < M, or
NRPT < MINMN, or
LWORK < 3 $\times$ MINMN.

IFAIL > 1

The QR algorithm has failed to converge to the singular values in 50 $\times$ MINMN iterations. In this case SV(1),SV(2),...,SV(IFAIL–1) may not have been correctly found and the remaining singular values may not be the smallest singular values. The matrix A has nevertheless been factorised as $A = QBP^T$ where B is an upper bidiagonal matrix with SV(1),SV(2),...,SV(MINMN) as its diagonal elements and WORK(2),WORK(3),...,WORK(MINMN) as its super-diagonal elements.

This failure is not likely to occur.

7. Auxiliary Routines

This routine calls the NAG Library routines F01LZF, F01QAF, F01QBF, F02SZF, F02WAY, F02WBY, F02WBZ, F02WCW, F02WCX, F02WCY, F02WCZ, P01AAF, X02AAF and X02AGF.

8. Timing

The following figures are intended only to be an approximate guide. They are based upon the assumption that the QR algorithm takes an average of two iterations per singular value.

If the routine is called with LWORK $\geq$ (3 $\times$ MINMN + MINMN^2), then the time taken is approximately proportional to $3n^2(m+4n)$ when $m \geq n$, and $5m^2(n+2m)$ when $m < n$.

If the routine is called with LWORK $\leq$ (3 $\times$ MINMN + $MINMN^2$), then the time taken is approximately proportional to $8n^2(m+\frac{1}{2}n)$ when $m \geq n$, and $10m^2(n+\frac{1}{2}m)$ when $m < n$.

The same approximate proportionality factor applies in each case.

9. Storage

There are no internally declared arrays.

10. Accuracy

The computed factors $\tilde{Q}$, D and $\tilde{P}^T$ satisfy the relation

$$\tilde{Q} D \tilde{P}^T = A + E,$$

where

$$||E||_2 \leq c \times eps \times ||A||_2,$$

eps being the machine accuracy (see NAG Library routine X02AAF) and c a modest function of M and N. Note that $||A_2|| = sv_1$.

11. Further Comments

Singular vectors associated with a zero or multiple singular value, are not uniquely determined, even in exact arithmetic, and very different results may be obtained if they are computed on different machines.

This routine is column-biased and so is suitable for use in paged environments.

12. Keywords

M $\times$ N Real Matrix, Rank, Singular Value Decomposition.

13. Example

To obtain the singular value decomposition of the matrix A given by

$$A = \begin{bmatrix} 22.25 & 31.75 & -38.25 & 65.50 \\ 20.00 & 26.75 & 28.50 & -26.50 \\ -15.25 & 24.25 & 27.75 & 18.50 \\ 27.25 & 10.00 & 3.00 & 2.00 \\ -17.25 & -30.75 & 11.25 & 7.50 \\ 17.25 & 30.75 & -11.25 & -7.50 \end{bmatrix}$$

WARNING: This single precision example program may require amendment for certain implementations. The results produced may not be the same. If in doubt, please seek further advice (see **Essential Introduction** to the Library Manual).

13.1. Program Text

```
C     F02WCF EXAMPLE PROGRAM TEXT
C     MARK 8 RELEASE. NAG COPYRIGHT 1979.
C     .. LOCAL SCALARS ..
      INTEGER I, IFAIL, J, LWORK, M, N, NIN, NOUT, NRA, NRPT, NRQ
C     .. LOCAL ARRAYS ..
      REAL A(10,6), PT(6,6), Q(10,6), SV(6), TITLE(7), WORK(28)
C     .. SUBROUTINE REFERENCES ..
C     F02WCF
C     ..
      DATA NIN /5/, NOUT /6/
      READ (NIN,99999) TITLE
      WRITE (NOUT,99998) (TITLE(I),I=1,6)
      NRA = 10
      NRQ = 10
      NRPT = 6
      LWORK = 28
      IFAIL = 0
      M = 6
      N = 4
      DO 20 I=1,N
         READ (NIN,99997) (A(J,I),J=1,M)
   20 CONTINUE
      CALL F02WCF(M, N, N, A, NRA, Q, NRQ, SV, PT, NRPT, WORK,
```

```
     * LWORK, IFAIL)
      WRITE (NOUT,99996)
      WRITE (NOUT,99992) ((A(I,J),J=1,N),I=1,M)
      WRITE (NOUT,99995)
      WRITE (NOUT,99992) ((Q(I,J),J=1,N),I=1,M)
      WRITE (NOUT,99994)
      WRITE (NOUT,99992) (SV(I),I=1,N)
      WRITE (NOUT,99993)
      WRITE (NOUT,99992) ((PT(I,J),J=1,N),I=1,N)
      STOP
99999 FORMAT (6A4, 1A3)
99998 FORMAT (4(1X/), 1H , 5A4, 1A3, 7HRESULTS/1X)
99997 FORMAT (6F7.2)
99996 FORMAT (9H MATRIX A)
99995 FORMAT (9H0MATRIX Q)
99994 FORMAT (16H0SINGULAR VALUES)
99993 FORMAT (12H0MATRIX P**T)
99992 FORMAT (1X, 4F9.3)
      END
```

13.2. Program Data

```
F02WCF EXAMPLE PROGRAM DATA
  22.25  20.00 -15.25  27.25 -17.25  17.25
  31.75  26.75  24.25  10.00 -30.75  30.75
 -38.25  28.50  27.75   3.00  11.25 -11.25
  65.50 -26.50  18.50   2.00   7.50  -7.50
```

13.3. Program Results

```
 F02WCF EXAMPLE PROGRAM RESULTS

 MATRIX A
   22.250   31.750  -38.250   65.500
   20.000   26.750   28.500  -26.500
  -15.250   24.250   27.750   18.500
   27.250   10.000    3.000    2.000
  -17.250  -30.750   11.250    7.500
   17.250   30.750  -11.250   -7.500
 MATRIX Q

    0.929    0.143    0.071   -0.143
   -0.143   -0.714    0.143   -0.286
    0.071   -0.143    0.929    0.143
    0.143   -0.286   -0.143   -0.714
   -0.214    0.429    0.214   -0.429
    0.214   -0.429   -0.214    0.429
 SINGULAR VALUES

   91.000   68.250   45.500   22.750
 MATRIX P**T

    0.308    0.462   -0.462    0.692
   -0.462   -0.692   -0.308    0.462
   -0.462    0.308    0.692    0.462
   -0.692    0.462   -0.462   -0.308
```