deelontwerp vloeistofmechanika

C. H. Marks

deelontwerp vloeistofmechanika

Het bepalen van opslingerfaktoren in vertakte kanaalsystemen

C.H. Marks

1. Doelstelling.

De oorspronkelijke doelstelling was na te gaan, hoe in IJmuiden de gunstigste situering voor een nieuwe sluis gevonden kon worden. De meest gunstige situering is die, waarbij het optreden van seiches, dus van golfverschijnselen, minimaal is.
Deze doelstelling is gewijzigd tot het maken van een systeem ter bepaling van de opslingerfaktoren in een willekeurig vertakt stelsel van open leidingen, waarbij het mechanisme op één plaats wordt opgewekt.
In verband met de zeer grote hoeveelheid rekenwerk, die nodig is om fysische verschijnselen als deze analytisch te beschrijven, is gepoogd het systeem voor gebruik op de TR-4 computer van de Wiskundige Dienst geschikt te maken.

2. Beschrijving van de rekenwijze.

Voor een open leiding geldt de vierpoolvergelijking, welke uit de vloeistofmechanika bekend is:

$$h(x) = \left\{h(0).\cosh rx - \frac{r}{bj\omega}.Q(0).\sinh rx\right\}.e^{j\omega t} \text{ en}$$

$$Q(x) = \left\{Q(0).\cosh rx - \frac{bj\omega}{r}.h(0).\sinh rx\right\}.e^{j\omega t}$$

In onze berekeningen is gesteld $t = 0$, dan is $e^{j\omega t} = 1$.
Elk vertakt stelsel van kanalen bestaat uit een aantal (n5) leidingen, waarvoor de vierpoolvergelijking geldt.
In de knooppunten (n4) van de leidingen geldt voor de naar het knooppunt gerichte debieten $\Sigma Q = 0$.
In het punt 0, waar het mechanisme wordt gegenereerd is de gegeven knooppuntsvoorwaarde $\hat{h}(0) = 1$.
Als in alle knooppunten de golfamplitude bekend is, dan zijn deze bekend in verhouding tot de gegeven amplitude. Dan hebben we dus de opslingerfactoren gevonden.
Door toepassing van de vierpoolvergelijking wordt geen inzicht verkregen in de tijd, die nodig is voor het aanslingeren van het systeem. Er is aangenomen, dat de aanslingering reeds oneindig lang gaande is, zodat de uiteindelijke evenwichtstoestand is verkregen.

3. Programma's.

Het computerprogramma bestaat uit 3 delen:

a. het programma, waarin de vergelijkingen worden opgesteld.

b. een procedure, volgens welke de vergelijkingen worden opgelost.

c. een getalband, waarop voor het onderhavige geval de specifieke gegevens zijn aangegeven.

Deze drie delen worden door middel van ponsbanden in de computer ingevoerd. Op de banden is de tekst van de programma's door middel van gaatjes, die in verschillende combinaties de letter- en cijfersymbolen representeren, aangebracht.

Bij het programmeren moet aan twee voorwaarden worden voldaan:

a. het programma moet syntactisch juist zijn. Dan kan het programma door de computer verwerkt worden.

b. het programma moet een juiste voorstelling geven van de gewenste wiskundige behandelingen.

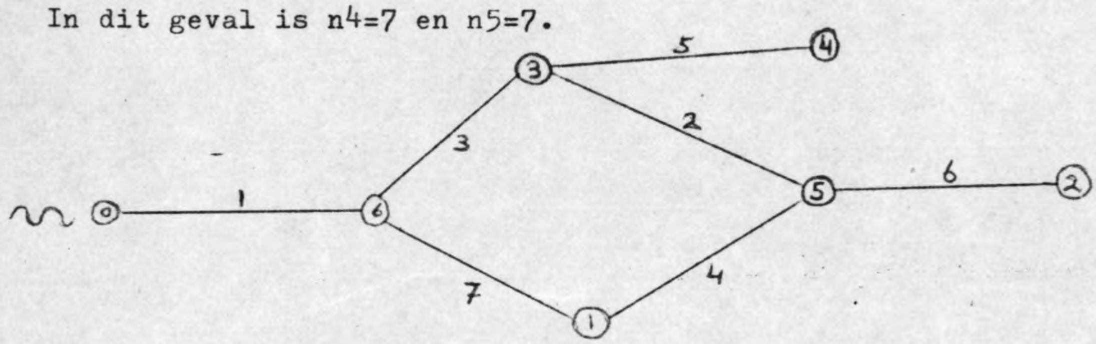
Aan de eerste voorwaarde kan vrij eenvoudig worden voldaan, omdat, indien er niet aan voldaan is, een uitvoer verkregen wordt, die aangeeft waar een fout zit en van welke aard die fout is. Men kan de fout dus snel corrigeren. In vrijwel elk programma van enige omvang treden syntactische fouten op, die successivelijk verwijderd worden.

Als niet aan de tweede voorwaarde wordt voldaan, verkrijgt men foutieve uitkomsten. Men kan aan de uitkomsten vaak niet zien of zij al dan niet juist zijn. Voordat tot toepassing van het programma wordt overgegaan moeten dus de uitkomsten worden geverifiëerd aan de uitkomsten van een bekend probleem. Het programma moet a.h.w. geijkt worden.

Als een fout aanwezig is, kan zij vaak worden gelokaliseerd m.b.v. testopdrachten. Deze testopdrachten geven een uitvoer. Als die uitvoer juist is, is het programma tot daar meestal juist. In ons probleem is lange tijd gepoogd met behulp van testopdrachten en door programmawijzigingen een programma te krijgen, dat juiste uitkomsten levert, Bij problemen, als het onze, is de topologie (en daarmee het aantal onbekenden) van het stelsel kanalen niet vooraf bekend. Voor alles moet deze topologie bekend worden gemaakt aan de rekenautomaat. In het programma worden daarom in 1 ingelezen de getallen $n4$ en $n5$, die op de getalband staan en per geval verschillen. Zij stellen voor het aantal knooppunten en het aantal takken. Deze twee getallen bepalen de grootte van het aantal vergelijkingen.

Alle knooppunten zijn genummerd en wel van 0 tot $n4-1$. De nummers van het begin- en eindpunt van alle $n5$ takken worden vastgelegd in 11. Ook worden hier ingelezen voor elke tak de lengte, breedte, diepte en de geschatte, gemiddelde stroomamplitude.

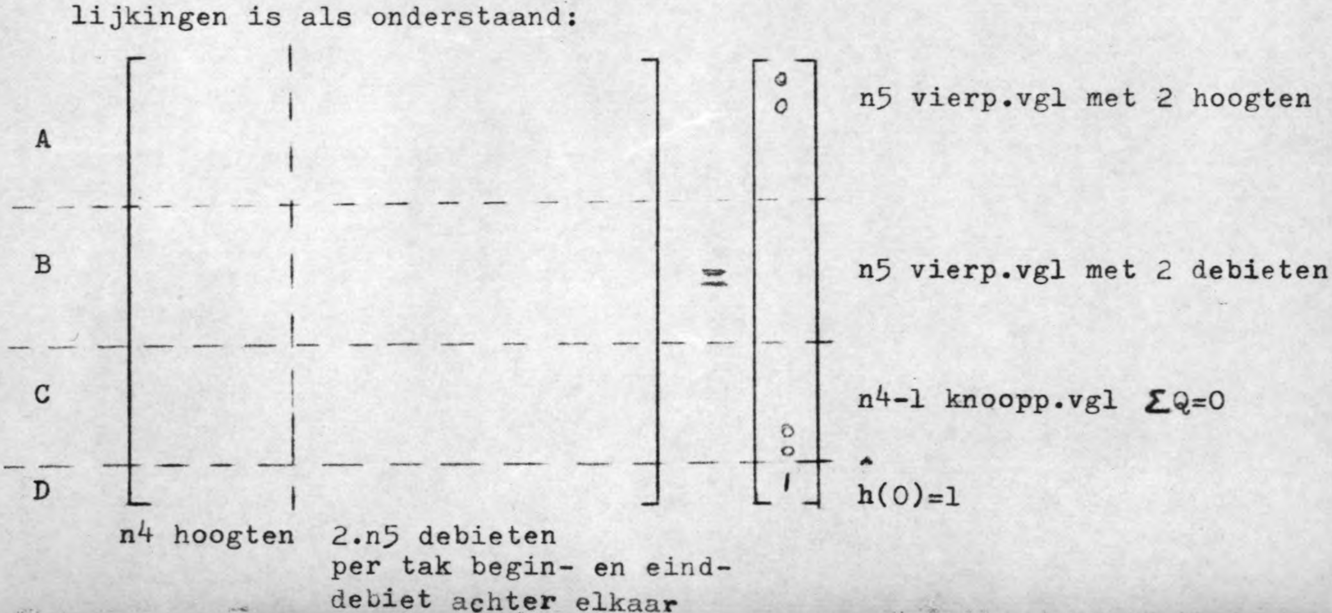
Het punt, waar het mechanisme wordt gegenereerd heeft altijd het nummer 0 en is het begin van tak 1. De verdere nummering is willekeurig. Onderstaand voorbeeld is een voorbeeld van nummering. In dit geval is $n4=7$ en $n5=7$.



Nu is de topologie bekend, want van elke tak weet de computer tussen welke knooppunten zij loopt.

Het aantal onbekenden is nu $n4+2.n5$. Immers voor alle $n5$ takken zijn de debieten onbekend en in alle knooppunten de opslingerfaktor. In punt 0 is de golfamplitude 1.

De struktuur van de coëfficiëntenmatrix van het stelsel vergelijkingen is als onderstaand:



Voor elke tak is aangenomen, dat de positieve richting loopt van het eerste naar het tweede knooppunt.
Bij positieve richting van de stroom en daar rekenen we mee, gaat het debiet aan het begin van een tak het knooppunt uit en aan het eind het knooppunt in. In een knooppunt geldt voor de som van de inkomende en uitgaande debieten $\Sigma Q=0$. In de vergelijking voor $\Sigma Q=0$ hebben begin- en einddebieten verschillende tekens.
Deze vergelijkingen worden in 111 opgesteld en komen in deel C van de matrix. Aangezien alle knooppunten in de lusopdracht zijn betrokken, worden de juiste vergelijkingen opgesteld. Als een punt van slechts één tak deel uitmaakt en dus een eindpunt is, wordt de voorwaarde Q=0. Dit klopt, want op de regel voor de vergelijking van dat punt komt dan slechts één coëfficiënt.
In deel 1V van het programma wordt de golfhoogte in het punt 0, waar de golf wordt opgewekt, 1 gesteld. In de matrix gebeurt dit in deel D.
In V worden de coëfficiënten van de vierpoolvergelijking per tak bepaald, waarna de vergelijkingen in V1 worden ingevuld.
De vergelijkingen, die het verband geven tussen h(x) en h(0) en Q(0) komen in deel A en de vergelijkingen met Q(x), h(0) en Q(0) komen in deel B van de matrix.
De coëfficiënten van deze vergelijkingen zijn complexe getallen.
Een complex getal bestaat uit een reële en een imaginaire component. Een complex getal moet daarom in een 2 dimensionaal rooster gedeclareerd worden, anders kan er met de bestaande procedures niet mee gerekend worden.
Oorspronkelijk is de matrix opgezet in 3 dimensies (rijen, kolommen en complexe dimensie); de matrixelementen heten hier z. Voor de oplossing worden de elementen 2 dimensionaal gemaakt door de rijelementen door te nummeren; de elementen heten nu y.
Alle coëfficiënten komen op de juiste plaats in de z-matrix, doordat voor de eerste index gebruik wordt gemaakt van het taknummer en voor de tweede index van het nummer van het knooppunt.
In deel C van de matrix is juist het omgekeerde toegepast, omdat hier op de rijen knooppuntsvergelijkingen komen.
Als nu de vergelijkingen zijn opgesteld en de matrix is ingevuld, moeten zij worden opgelost.

In Vll wordt de procedure linvgl aangeroepen, die het stelsel vergelijkingen oplost.
De oplossingsprocedure is niet gebonden aan deze specifieke toepassing, doch geniet algemene geldigheid voor een stelsel lineaire, niet homogene, complexe vergelijkingen.
De oplossing wordt verkregen door een eliminatieproces.
Te beginnen bij de eerste rij wordt op elke rij het eerste element gezocht, waarvan de modulus ongelijk nul is. De hele rij met bijbehorende bekende term wordt door dit element gedeeld. Het element zelf, het spilelement dus, heeft de waarde één gekregen.
Vervolgens wordt het spilelement geëlimineerd uit de overige rijen door de oorspronkelijke rij, na vermenigvuldiging met het in de kolom van het spilelement voorkomende element van die overige rijen, af te trekken van de andere rijen. In de kolom van het spilelement komen alleen nullen. De hele matrix wordt dus schoon geveegd. Er blijft per rij en per kolom één 1 over.
Als het stelsel vergelijkingen niet homogeen is, staan de énen niet op de hoofddiagonaal en daardoor staan de elementen, die de oplossing van het stelsel vormen, niet in de juiste volgorde.
Door de rijen van de matrix te verwisselen worden de oplossingselementen in de juiste volgorde gezet. Daarmee is de oplossing volledig en kan zij in Vlll uitgevoerd worden.
De uitvoer is van onderstaande gedaante:

knoop 1	reële-	imag.-comp.	modulus v/d opslingerfaktor
2	"	"	"
.			
n4	"	"	"

tak 1	reëel	imag.	modulus	reëel	imag.	modulus
2	"	"	"	"	"	"
.						
n5	"	"	"	"	"	"
	v/h debiet a/h begin van de tak			v/h debiet a/h einde van de tak		

In lX wordt het gemiddelde genomen van de moduli van het debiet aan het **begin** en aan het eind van elke tak. De gevonden waarde wordt opnieuw ingevoerd bij "over", zodat een iteratie optreedt, waardoor de juiste waarde voor de wrijving wordt benaderd. Deze iteratie wordt vier maal uitgevoerd. Bovenstaande uitvoer wordt voor elke ω dus vijf maal gegeven en wordt aangeduid door iteratie 1, 2, 3, 4, 5.

In X wordt een nieuwe waarde voor ω bepaald, waarvoor het proces opnieuw wordt verricht. De nieuwe omega = oude omega + stap.

De getalband is van de volgende vorm:

n4,n5;

0, ,l(tak),b(tak),d(tak),debg(tak); } dit voor alle n5 takken

" " " " " "

omega,stap,stop;

Als voor alle takken debg(tak) = 0, dan is in de eerste iteratie de wrijving verwaarloosd.
Met een reeds bekend programma (bijlage 2) kunnen de opslingerfaktoren worden bepaald zonder wrijving. De uitkomsten moeten in dit geval overeenstemmen met de uitkomsten van onze methode in de eerste iteratie.
We hebben na draaien van verschillende voorbeelden slechts 1 geval gevonden, waarin onze eerste iteratie en de reeds bekende methode overeenkomstige resultaten opleverden (bijlagen 1 en 2). De bij ons in eerste iteratie verkregen uitkomsten keren terug in de tweede en derde iteratie; kennelijk is de limietwaarde reeds in eerste instantie bereikt. In iteratie 4 echter krijgen we geheel nieuwe uitkomsten en in iteratie 5 komen de oude getallen weer terug. Bij gelijke invoer kunnen kennelijk verschillende uitkomsten optreden. Dit is zeer onwaarschijnlijk. Er is dus een fout in het programma aanwezig.
In bijlage 3 (deel 2) blijkt, dat een topologisch gelijke situatie als in bijlage 1 foute uitkomsten levert ($\Sigma Q \neq 0$).
In bijlage 4 blijkt, dat zelfs als alleen een aantal takken aan elkaar geschakeld zijn in de knooppunten niet aan $\Sigma Q=0$ wordt voldaan. Uit de testopdrachten blijkt in dit voorbeeld, dat de vergelijkingen goed zijn opgesteld. De fout zit dus waarschijnlijk in de oplossingsprocedure. Hierin is echter tot nu toe geen fout gevonden.

4. Benodigde rekentijd.

De rekentijd per omega (5 iteraties) op de computer is afhankelijk van het aantal onbekenden n4+2.n5.

In onderstaande tabel is de rekentijd voor verschillende gevallen aangegeven.

n4	n5	n4+2.n5	tijd in sec.	tijd in min.
4	3	10	57	1
6	5	16	223	4
10	10	30	1275	21

Als n = n4+2.n5, dan is de benodigde rekentijd in minuten per omega voor te stellen door tijd = $n^3/1200$.

Voor grotere stelsels is het aantal vergelijkingen en daarmee de benodigde rekentijd zeer groot. Dat de rekentijd zo groot is, is een gevolg van het rekenen met e-machten ter bepaling van sinh en cosh en van het rekenen met complexe getallen. De e-machten worden via een reeksontwikkeling bepaald. Elke complexe handeling duurt ca. 4 maal zo lang als de overeenkomstige lineaire handeling. Aangezien rekentijd vrij duur, verdienen andere rekenwijzen zeker de voorkeur boven de hier toegepaste methode. Gedacht kan worden aan een rekenwijze, die met differenties werkt. In dat geval kan ook de tijdsinvloed in rekening worden gebracht.

5. Conclusie.

De opgestelde programma's vertonen een afwijking, die leidt tot onjuiste uitkomsten. In bepaalde gevallen, als de topologie van het stelsel gunstig is, treedt de fout niet op. Dit is toeval.

De voor de methode benodigde rekentijd is zeer groot, waardoor toepassing van deze methode geen aanbeveling verdient.

Het deelontwerp heeft niet tot resultaten geleid.

Procedure voor de oplossing van n lineaire, niet homogene, complexe vergelijkingen.

```
==a3,
          'procedure' linvgl(x,bek,bok,n);
          'array' x,bek,bok;
          'integer' n;
'begin'   'integer' g,i,mk,f,a;
          'real' ab,ac;
          'boolean' nul,regel;


'begin'    'integer' k;
           'array' h[1:4,1:2];
           'integer' 'procedure' p(i,j);
           'value' i,j;
           'integer' i,j;
   'begin'   'real' a;
             k:=k-1;
             a:=h[i,1]*h[j,1]-h[i,2]*h[j,2];
             h[k,2]:=h[i,1]*h[j,2]+h[i,2]*h[j,1];
             h[k,1]:=a;
             p:=k
   'end';
             'integer' 'procedure' q(i,j);
             'value' i,j;
             'integer' i,j;
   'begin'   'real' a,b;
             k:=k-1;
             b:=h[j,1]*h[j,1]+h[j,2]*h[j,2];
             a:=(h[i,1]*h[j,1]+h[i,2]*h[j,2])/b;
             h[k,2]:=(h[i,2]*h[j,1]-h[i,1]*h[j,2])/b;
             h[k,1]:=a;
             q:=k
   'end';
             'integer' 'procedure' s(i,j);
             'value' i,j;
             'integer' i,j;
   'begin'   k:=k-1;
             h[k,1]:=h[i,1]+h[j,1];
             h[k,2]:=h[i,2]+h[j,2];
             s:=k
   'end';
             'integer' 'procedure' t(i,a);
             'value' i;
             'integer' i;
             'array' a;
   'begin'   k:=k+1;
             h[k,1]:=a[i,1];
             h[k,2]:=a[i,2];
             t:=k
   'end';
             'integer' 'procedure' j(c,d);
             'real' c,d;
   'begin'   k:=k+1;
             h[k,1]:=c;
             h[k,2]:=d; j:=k
   'end';
             'procedure' u(i,j,r);
             'value' i,j;
             'integer' i,j;
             'array' r;
   'begin'   r[j,1]:=h[i,1];
             r[j,2]:=h[i,2];
             k:=0
   'end';

             'integer' 'procedure' v(i,j);
             'value' i,j;
             'integer' i,j;
   'begin'   k:=k-1;
             h[k,1]:=h[i,1]-h[j,1];
             h[k,2]:=h[i,2]-h[j,2];
             v:=k
   'end';
             k:=0;

             'for' g:=1 'step' 1 'until' n 'do'
   'begin'   mk:=1;
opnieuw:
       'begin'  ac:=abs(x[(g-1)*n+mk,1])+abs(x[(g-1)*n+mk,2]);
                nul:=ac 'not equal' 0;
                'if' nul 'then'
           'begin'  'for' i:=1 'step' 1 'until' n 'do'
                     'if' i 'not equal' mk 'then'
                'begin'  u(q(t((g-1)*n+i,x),t((g-1)*n+mk,x)),(g-1)*n+i,x);
                          'for' f:=1 'step' 1 'until' n 'do'
                            'if' f 'not equal' g 'then'
                         u(v(t((f-1)*n+i,x),p(t((f-1)*n+mk,x),t((g-1)*n+i,x
                        ))),(f-1)*n+i,x);
      'end';
              u(q(t(g,bek),t((g-1)*n+mk,x)),g,bek);
                       u(j(1,0),(g-1)*n+mk,x);
                       'for' f:=1 'step' 1 'until' n 'do' .
                       'if' f 'not equal' g 'then'
       'begin'         u(v(t(f,bek),p(t(g,bek),t((f-1)*n+mk,x))),f,bek);
                       u(j(0,0),(f-1)*n+mk,x);
       'end'
           'end'
                'else'
   'begin'   mk:=mk+1; test(''mk'',mk); 'go to' opnieuw;
   'end';
'end';
'end';
         'for' g:=1 'step' 1 'until' n 'do'
'begin'  'for' i:=1 'step' 1 'until' n 'do'
   'begin'  vasko(1,0,x[(g-1)*n+i,1]); space(1);
   'end';
            vasko(5,3,bek[g,1]); nlcr(1);
'end';
         'comment' nu is de complexe matrix opgelost, maar de
              eentjes staan niet op de hoofddiagonaal, zij moeten
              nog geordend worden;
              'for' g:=1 'step' 1 'until' n 'do'
              'for'a:=1,2 'do' bok[g,a]:=0;
              'for' g:=1 'step' 1'until' n 'do' 'for' a:=1,2 'do'
'begin'  'for' i:=1 'step' 1 'until' n 'do'
   'begin'  ab:=x[(g-1)*n+i,1];
            regel:=ab 'greater' 0.5; 'if' regel'then'
            bok[i,a]:=bek[g,a];
      'end'
 'end'
'end';   'for' g:=1 'step' 1 'until' n 'do'
'begin'  vasko(5,3,bok[g,1]); nlcr(1);
'end';
'end'
```

Programma voor de opstelling van de vierpoolvergelijkingen voor een willekeurig vertakt stelsel van open waterleidingen, alsmede voor de opstelling van de knooppuntsvoorwaarden.

```
==a1,

'begin'  'real' chezy,stap,omega;
        'procedure' linvgl; 'code';
        'integer' n1,n2,n3,n4,n5,tak,tok,a,c,a1,m,n,s1,s2,tijd,
        stop,es,ev;
I. . . . . . .read(n4,n5);
        n1:=n4+2*n5;


'begin'   'integer' k;
          'array' h[1:4,1:2];
          'integer' 'procedure' p(i,j);
          'value' i,j;
          'integer' i,j;
  'begin'   'real' a;
            k:=k-1;
            a:=h[i,1]*h[j,1]-h[i,2]*h[j,2];
            h[k,2]:=h[i,1]*h[j,2]+h[i,2]*h[j,1];
            h[k,1]:=a;
            p:=k
  'end';
            'integer' 'procedure' q(i,j);
            'value' i,j;
            'integer' i,j;
  'begin'   'real' a,b;
       *    k:=k-1;
            b:=h[j,1]*h[j,1]+h[j,2]*h[j,2];
            a:=(h[i,1]*h[j,1]+h[i,2]*h[j,2])/b;
            h[k,2]:=(h[i,2]*h[j,1]-h[i,1]*h[j,2])/b;
            h[k,1]:=a;
            q:=k
  'end';
            'integer' 'procedure' s(i,j);
            'value' i,j;
            'integer' i,j;
  'begin'   k:=k-1;
            h[k,1]:=h[i,1]+h[j,1];
            h[k,2]:=h[i,2]+h[j,2];
            s:=k
  'end';
            'integer' 'procedure' t(i,a);
            'value' i;
            'integer' i;
            'array' a;
  'begin'   k:=k+1;
            h[k,1]:=a[i,1];
            h[k,2]:=a[i,2];
            t:=k
  'end';
            'integer' 'procedure' j(c,d);
            'real' c,d;
  'begin'   k:=k+1;
            h[k,1]:=c;
            h[k,2]:=d; j:=k
  'end';
            'procedure' u(i,j,r);
            'value' i,j;
            'integer' i,j;
            'array' r;
  'begin'   r[j,1]:=h[i,1];
            r[j,2]:=h[i,2];
            k:=0
  'end';
            'integer' 'procedure' e(i,a);
            'value' i;
            'integer' i;
            'array' a;
  'begin'   'real' d;
            k:=k+1;
            d:=exp(a[i,1]);
            h[k,2]:=d*sin(a[i,2]);
            h[k,1]:=d*cos(a[i,2]);
            e:=k
  'end';

            'integer' 'procedure' v(i,j);
            'value' i,j;
            'integer' i,j;
  'begin'   k:=k-1;
            h[k,1]:=h[i,1]-h[j,1];
            h[k,2]:=h[i,2]-h[j,2];
            v:=k
  'end';
            k:=0;
  'begin'   'array' y[1:(2*n5+n4) 'power' 2,1:2],z[1:2*n5+n4,1:2*n5+
            n4,1:2],phi,r,ch,sh,rx,deb0,deb1[1:n5,1:2],
            debg[1:n5],ga,b,l,d,ka,ar[1:n5],
            h1[1:2*n5+n4,1:2],hg[1:2*n5+n4,1:2];
            'boolean' leeg,goed; 'integer' 'array' knoop[1:n5,1:2];

            chezy:=50; tijd:=klok;
            test(''tijd='',tijd);
      [     'for' tak:=1 'step' 1 'until' n5 'do'
II. . .     read(knoop[tak,1],knoop[tak,2],l[tak],b[tak],d[tak],
            debg[tak]);
      [     read(omega,stap,stop);
            es:=1; ev:=1;
  over:       'for' m:=1 'step' 1 'until' n4+2*n5 'do'
            'for' n:=1 'step' 1 'until' n4+2*n5 'do'
            'for' a:=1,2 'do' z[m,n,a]:=y[(m-1)*n1+n,a]:=0;
            'for' tak:=2 'step' 1 'until' n5 'do'
      [  'begin' z[2*n5+knoop[tak,1],n4+2*tak-1,1]:=1;
III. .          z[2*n5+knoop[tak,2],n4+2*tak,1]:=-1;
      [  'end';
                z[2*n5+knoop[1,2],n4+2,1]:=-1;
IV. . . . . . . . . .z[2*n5+n4,1,1]:=1;
         'for' m:=1 'step' 1 'until' n4+2*n5-1 'do'
            'for' a:=1,2 'do'
            h1[m,a]:=0; h1[n4+2*n5,1]:=1; h1[n4+2*n5,2]:=0;
            test(''6''); print(omega);
            'for' tak:=1 'step' 1 'until' n5 'do'
  'begin'   ga[tak]:=1/(9.81*b[tak]*d[tak]);
             ar[tak]:=b[tak]*d[tak]/(b[tak]+2*d[tak]);
             ka[tak]:=(8/(3*3.14159))*debg[tak]/((chezy*b[tak]
             *d[tak]) 'power' 2*ar[tak]);
             r[tak,2]:=sqrt((ga[tak]*b[tak]*omega*omega+sqrt
             ((ga[tak]*b[tak]*omega 'power' 2) 'power' 2+
              (ka[tak]*b[tak]*omega) 'power' 2))/2);
V. . .        r[tak,1]:=ka[tak]*b[tak]*omega/(2*r[tak,2]);
              test(''7'');
             phi[tak,1]:=+r[tak,2]/(b[tak]*omega);
             phi[tak,2]:=-r[tak,1]/(b[tak]*omega);
             rx[tak,1]:=r[tak,1]*l[tak];
             rx[tak,2]:=r[tak,2]*l[tak]; test(''a'',rx[tak,2]);

         u(q(s(e(tak,rx),q(j(1,0),e(tak,rx))),j(2,0)),tak,ch);
         u(q(v(e(tak,rx),q(j(1,0),e(tak,rx))),j(2,0)),tak,sh);
         'for' a:=1,2 'do'
  'begin'   z[tak,knoop[tak,1]+1,a]:=+
            ch[tak,a];
            z[tak,knoop[tak,2]+1,1]:=-1;
            z[tak+n5,n4+2*tak-1,a]:=+
            ch[tak,a];
VI. . .     z[tak+n5,n4+2*tak,1]:=-1;
  'end';
            u(p(p(j(-1,0),t(tak,sh)),t(tak,phi)),
            (tak-1)*n1+n4+2*tak-1,y);
            u(q(p(j(-1,0),t(tak,sh)),t(tak,phi)),
            (tak+n5-1)*n1+knoop[tak,1]+1,y);
 'end';
            'for' m:=1 'step' 1 'until' n4+2*n5 'do'
            'for' n:=1 'step' 1 'until' n4+2*n5 'do'
            'for' a:=1,2 'do'
            y[(m-1)*(n4+2*n5)+n,a]:=z[m,n,a]+y[(m-1)*n1+n,a];
VII. . . . . . . . .linvgl(y,h1,hg,n4+2*n5); test(''linvgl'');
            print(''iteratie''); vasko(3,0,es);
            'for' tak:=1 'step' 1 'until' n5 'do'
  'begin'   u(t(n4+2*tak-1,hg),tak,deb0);
            u(t(n4+2*tak,hg),tak,deb1);
  'end';
            'for' m:=1 'step' 1 'until' n4 'do'
    'begin'    print(''knoop''); vasko(3,0,m-1);
            vasko(7,3,hg[m,1],hg[m,2],sqrt(hg[m,1] 'power' 2+
            hg[m,2] 'power' 2));
  'end';    nlcr(1);
            'for' tak:=1 'step' 1 'until' n5 'do'
VIII. . 'begin' print(''tak''); vasko(2,0,tak);
            vasko(7,3,deb0[tak,1],deb0[tak,2],sqrt(deb0[tak,1]
            'power' 2+deb0[tak,2] 'power' 2),deb1[tak,1],
            deb1[tak,2],sqrt(deb1[tak,1] 'power' 2+deb1[tak,2]
            'power' 2));
  'end';    nlcr(2);
            print(''tijd=''); vasko(5,0,+klok-tijd);
            write(''sec.'');

    nlcr(2);
          es:=es+1; 'for' tak:=1 'step' 1 'until' n5 'do'
IX. .     debg[tak]:=(sqrt(deb0[tak,1] 'power' 2+deb0[tak,2] 'power'
          2)+sqrt(deb1[tak,1] 'power' 2+deb1[tak,2] 'power' 2))/2;
          'if' es 'less' 6 'then' 'go to' over; npag;
          ev:=ev+1; 'for' tak:=1 'step' 1 'until' n5 'do'
X. . .    debg[tak]:=0; omega:=omega+stap; es:=1;
          'if' ev 'less' stop 'then' 'go to' over;
 'end'
 'end'
 'end'
```