Verdeling van een convex gebied in vierhoekige elementen t.b.v. 2D-meshgenerator

Verdeling van een convex gebied in vierhoekige elementen t.b.v. 2D-meshgenerator

WE-74-4 jac wismans

Inhoudsopgave

Voorwoord.

Dit rapport geeft een beschrijving v.d procedure divconvexvierhoek. Het is in eerste instantie als discussie-stuk bedoeld voor de werkgroep I/O van Progel die zich bezig houdt met de ontwikkeling v.e. 2D- Meshgenerator. Bij de samenstelling v.h. programma is nog nauwelijks rekening gehouden met zaken als : efficiënt gebruik van geheugenruimte, effectief aanroepen van getallen e.d. : zaken die de rekentijd nog aanmerkelijk zullen kunnen bekorten.
Het doel was in eerste instantie een programma te krijgen dat werkte. In een later stadium kan m.b.v. dit rapport een meer efficiënt programma worden opgezet.

Jac Wismans
feb. 1974

1 Inleiding

1.1 Een belangrijk blok bij de 2D-meshgenerator is het genereren van een elementverdeling in een gebied waarvan de knooppunten op de rand gegeven zijn. Vaak zal zo'n gebied een concave vorm hebben. Is dit het geval dan zal zo'n gebied eerst in een aantal convexe gebieden verdeeld moeten worden.
In dit rapport wordt een beschrijving gegeven v.d. procedure Divconvex-vierhoek welke een convex gebied in vierhoekige elementen verdeeld.
Procedures die bij Divconvexvierhoek gebruikt worden, zullen afzonderlijk besproken worden.
In het nog te verschijnen afstudeerverslag zullen literatuur, theoretische achtergronden e.d. aan de orde komen.

1.2 Doel van dit rapport is:

a) De leden v.d. werkgroep, die zich bezig houdt met de ontwikkeling van de 2D-meshgenerator, een duidelijk beeld te geven v.d opzet van dit programma.

b) De plaatsen in het programma aan te duiden waar op min of meer willekeurige gronden criteria zijn gekozen die een belangrijke invloed op de elementvorm kunnen hebben (b.v. toelaatbare hoeken).

c) De plaatsen in het programma aan te geven, waar, door eenvoudige veranderingen in het programma aan te brengen, een vermindering v.d. benodigde rekentijd kan worden verkregen; vaak echter gepaard gaande met een slechtere elementvorm.

d) Overzetten van dit programma in andere talen te vergemakkelijken.

1.3 De beschrijving v.d verschillende procedures is meestal als volgt:

a) een inleiding : Hierin wordt in het kort de functie v.d procedure beschreven

b) gebruiksaanwijzing : beschrijving van globale en formele parameters

c) toelichting : beschrijving v.d werking v.d procedure.

d) blokschema : bij een aantal procedures wordt de werking d.m.v. een blokschema verduidelijkt.

e) discutabel : in deze paragraaf worden gehanteerde en eventueel te veranderen criteria besproken. Tevens zijn hierin soms suggesties opgenomen om een lagere reken tijd te verkrijgen.

1.4 Vanaf blz 57. worden voorbeelden gegeven v.d. met behulp v.d. procedure Div convex vierhoek gemaakte verdelingen.
Tevens zijn opgenomen een aantal voorbeelden van een uit de verdeling in vierhoeken gemaakte verdeling in driehoeken m.b.v. procedure Makedriehoek. Beschrijving van deze procedure + commentaar is in dit rapport opgenomen.

1.5 Achter elke procedure is de programma tekst opgenomen in de taal Beathe.
Beathe is de implementatie van Algol 60 op de B 6700 v.d Technische Hogeschool Eindhoven (Burroughs Extended Algol Technische Hogeschool Eindhoven)

Procedure Divconvexvierhoek

1. Inleiding

De procedure Divconvexvierhoek verdeelt een convexgebied, waarvan de rand met knooppunten is gegeven in vierhoekige elementen.

2. Gebruiksaanwijzing

Divconvexvierhoek(co, ct, kct, af, hk, td, ae, ak).

2.1 Globale parameters.

procedure DIVA
procedure DIV B
procedure DIV C
procedure DIV D
procedure DIV F
procedure Renumct
procedure Divline A
real procedure Hkcos
boolean procedure Contreline

Deze procedures worden allen afzonderlijk in dit rapport besproken.

2.2 Formele parameters.

integer kct — geeft het aantal knooppunten op de rand(contour) aan. kct moet, omdat we vierhoekige elementen wensen, even zijn.

integer ae — elementnummer van het eerstvolgende element dat gemaakt wordt.

integer ak — knooppuntsnummer van het eerstvolgende knooppunt dat gemaakt wordt.

array co — array met de grenzen [1:n, 1:2]. Bevat de coördinaten v.d. knooppunten op de contour en wordt gevuld met de coördinaten v.d gemaakte knooppunten.
n is een ruime schatting van het totaal aantal te verwachten knooppunten.

vb: co[i,1] heeft of krijgt de waarde v.d x-coördinaat van het knooppunt met knooppuntsnummer: i

array ct — array met de grenzen [0: kct+1]

Dit array bevat de knooppuntsnummers v.d knooppunten op de contour. De contour wordt met de klok mee doorlopen.

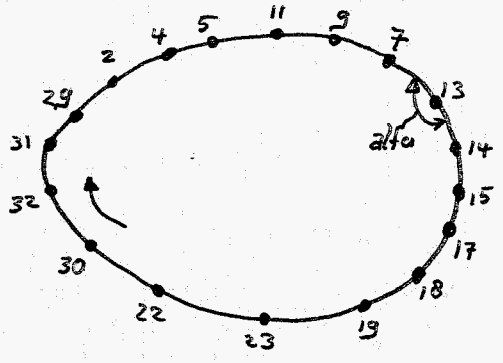


fig. 1

v.b: kct = 18 — ct[1] zou b.v. 13 kunnen zijn.

dan geldt: ct[2] = 14, ct[3] = 15 etc...

ct[0] = 7, ct[kct] = 7,

ct[kct+1] = 13.

array af — array met de grenzen [0: kct+1].

Dit array bevat de afstanden tussen de opeenvolgende knooppunten v.d contour.

v.b. zie fig 1. Omdat ct[1] is 13, is af[1] de afstand tussen 13 en 14. af[0] is de afstand tussen 7 en 13. etc....

array hk — array met de grenzen [0: kct+1].

Dit array bevat de cosinus v.d binnenhoeken van de verschillende knooppunten op de contour

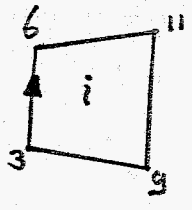
v.b zie fig 1. hk[1] = cosinus(alfa) etc...

array td — array met de grenzen [1:l, 1:4].

Dit array bevat na afloop v.d procedure de topologische beschrijving v.d elementen.

l is een ruime schatting van het aantal te verwachten elementen.

v.b: elementnummer: i



td[i,1] = 3 — td[i,3] = 11

td[i,2] = 6 — td[i,4] = 9

3.1 Werking v.d. procedure

De getallen in de tekst verwijzen naar de nummers v.d. verschillende blokken in het blokschema (4.1)

Gegeven: Een convex gebied met een rand (verder contour genoemd), bestaande uit knooppunten, verbonden door rechte lijnen. Deze knooppunten worden knooppunten van de aan de contour grenzende elementen. Op de contour worden verder geen knooppunten gegenereerd. (zie fig 1)

We wensen in dit gebied een elementverdeling aan te brengen, bestaande uit vierhoekige elementen, zodanig dat het grofheidsverloop v.d elementen zo gelijkmatig mogelijk is. Onder grofheid verstaan we in dit verhaal: $\frac{\text{elementribbe}}{\text{standaardlengte}}$.

Uit de grootte v.d elementribben op de contour wordt de grootte v.d elementribben in het gebied bepaalt (procedure Bpg).

De rand wordt steeds met de klok mee doorlopen.

We gaan nu als volgt te werk:

We zoeken op de contour de meest "scherpe" binnenhoek, die echter niet groter dan een bepaalde waarde (b.v. 90°) mag zijn ②. Vinden we zo'n hoek niet dan wordt het gebied in 2 stukken verdeeld (fig 2)

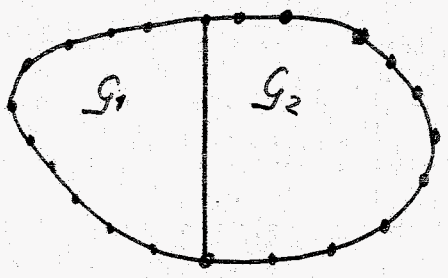


fig 2.

Op de lijn die het gebied verdeeld worden een aantal knooppunten gegenereerd, rekening houdende met de verdeling op de contour v/h. gebied.

We hebben nu 2 gebieden: G_1 en G_2. Elk gebied wordt verder afzonderlijk behandeld.

In zo'n subgebied zal een hoek die aan de criteria voldoet, waarschijnlijk wel gevonden worden, zoniet, dan zal verdere verdeling in subgebieden noodzakelijk zijn.

Uitgaande van deze scherpe hoek kappen we van het gebied een strook af zoals is aangegeven in fig 3.

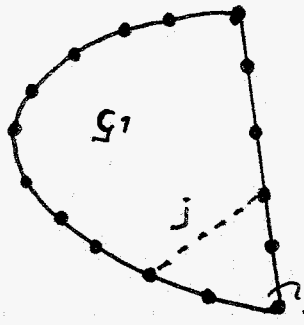


fig 3

Indien de hoeken die de lijn j met de contour maakt niet te scherp zijn (11), worden op de lijn klz punten gegenereerd (15). klz mag één of drie zijn. Indien er méér dan 3 knooppunten op de lijn gegenereerd worden, wordt lijn j niet geaccepteerd (16)

De afgekapte strook wordt als volgt in elementen verdeeld:

Indien klz = één (19):

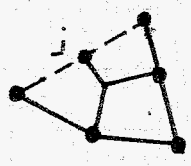


fig 4

Indien klz = drie (20):

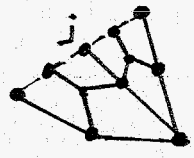


fig 5

Indien de strook niet acceptabel is omdat er teveel punten op de lijn j gegenereerd zijn of omdat de hoeken van lijn j met de contour te scherp zijn proberen we vanuit een andere scherpe hoek te beginnen of gaan we het gebied in verdere subgebieden onderverdelen. (12)

De volgende stap is: We trekken een lijn j zoals is aangegeven in fig 6:

fig 6

We controleren weer de hoeken die deze lijn met de contour maakt.

Indien deze hoeken acceptabel zijn (25) en bovendien α en β (24) niet te klein zijn, worden op de lijn klz punten gegenereerd (26).

Wijkt het aantal punten op de lijn j meer dan 4 af van het aantal dat op de vorige lijn is gegenereerd dan is de lijn niet acceptabel (27)
Wanneer nu de lijn wel acceptabel is gaan we de strook tussen de lijn j en en de vorige lijn verdelen op de in 2.3.2 aangegeven wijze (28).
Op deze manier hebben we dus een 2e strook v/h gebied afgekapt.
We kunnen nu deze stap een aantal malen herhalen.
Dit stroken afkappen kan op de volgende manieren tot een einde komen:

a) Zoals we reeds gezien hebben: als de lijn niet acceptabel is omdat de hoeken met de contour te scherp worden of omdat er te veel of te weinig knooppunten op de lijn gegenereerd zijn.
b) Indien er te weinig knooppunten overblijven om een "zinnige" lijn te trekken (23):

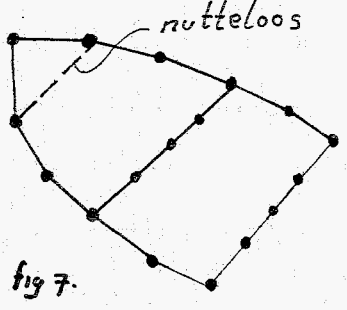


fig 7.

of:

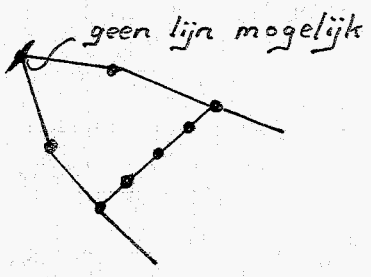


fig 8

In geval (a) wordt hetgeen wat overblijft van het gebied beschouwd als een nieuw subgebied (29) en wordt het gehele verhaal, te beginnen bij het zoeken v.e. scherpe hoek herhaald.
In geval (b) wordt indien het overblijvend subgebied slechts 6 knooppunten bevat een verdeling toegepast afhankelijk v.d vorm van dit gebiedje (DIVC).
Indien er meer dan 6 knooppunten zijn doen we hetzelfde als in geval (a).
We herhalen dit hele proces totdat alle subgebieden verdeeld zijn.
De knooppuntsnummering is op deze manier natuurlijk vrij willekeurig tot stand gekomen, evenals de positionering v.e. aantal knooppunten.
Repositionering en hernummering zal dan ook nog zeker noodzakelijk zijn.
De belangrijkste fase bij het aanbrengen van de elementverdeling is het verdelen van een lijn in een aantal stukken.

De plaatsing van de knooppunten op een lijn zal namelijk voor het grootste gedeelte bepalen hoe het grofheidsverloop van de elementen zal worden. Bij de bepaling van de plaats v.d. knooppunten zal steeds gekeken worden naar de contour van het oorspronkelijk gebied G, dus niet naar de contour van de eventueel ontstane subgebiedjes. We bereiken hiermee dat de elementverdeling plaatselijk niet kan ontaarden in te grote of te kleine elementen.

3.2. Een strook in elementen verdelen. (28)

De toelichting zal geschieden aan de hand v.e. voorbeeld.
Zie eventueel ook Blokschema 4.2.

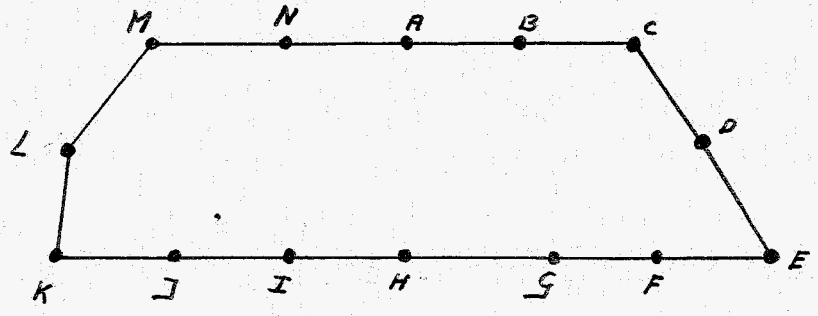


fig. 9

We kunnen 3 fasen onderscheiden bij het verdelen v.e. strook:

a) Verdeel het gebied ABCDEFG. tmax = 2.
tmax geeft aan het aantal malen dat een standaardverdeling gemaakt moet worden.

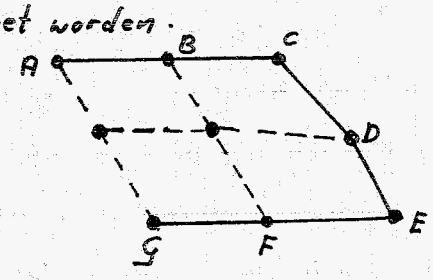


fig. 10

Voor het verdelen gebruiken we procedure DIVA

b) Verdeel het gebied JIKLMNA. Dit geschiedt op de zelfde manier als onder a

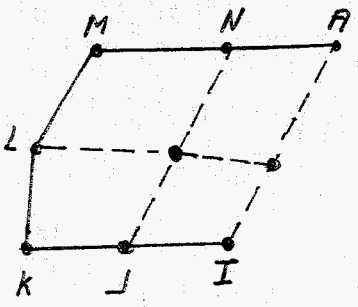


fig. 11

c) Indien $I = G$ hebben we een knooppunt te veel gegenereerd. We verwijderen dan het laatst gegenereerde knooppunt en moeten de topologische beschrijving v.d. middelste 4 elementen nog in orde maken.

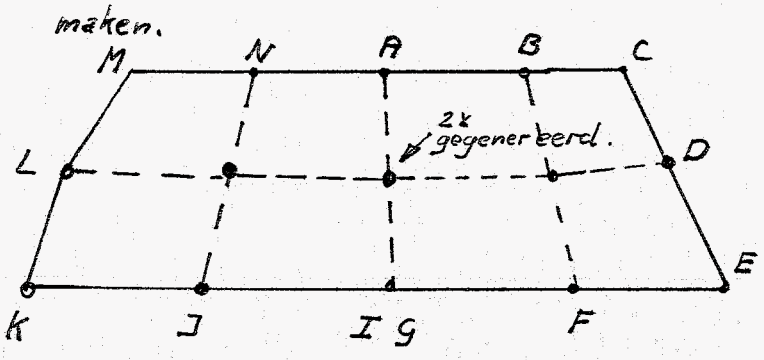


fig 12.

Indien tussen I en G één punt ligt, wordt het gebied AGHI verdeeld m.b.v. DIVB en indien tussen G en I 3 knooppunten liggen m.b.v. DIVD. (fig 13)

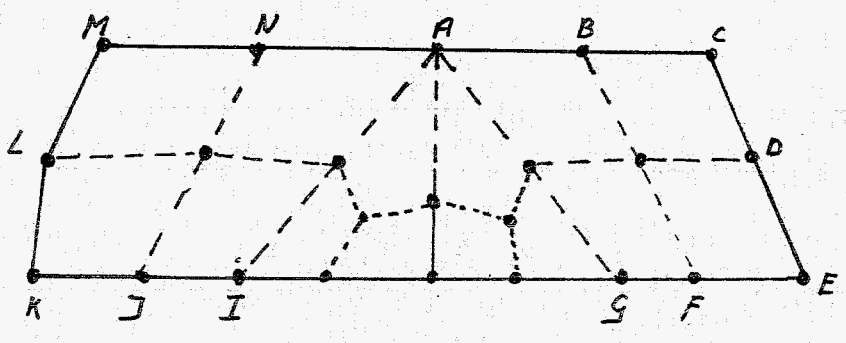
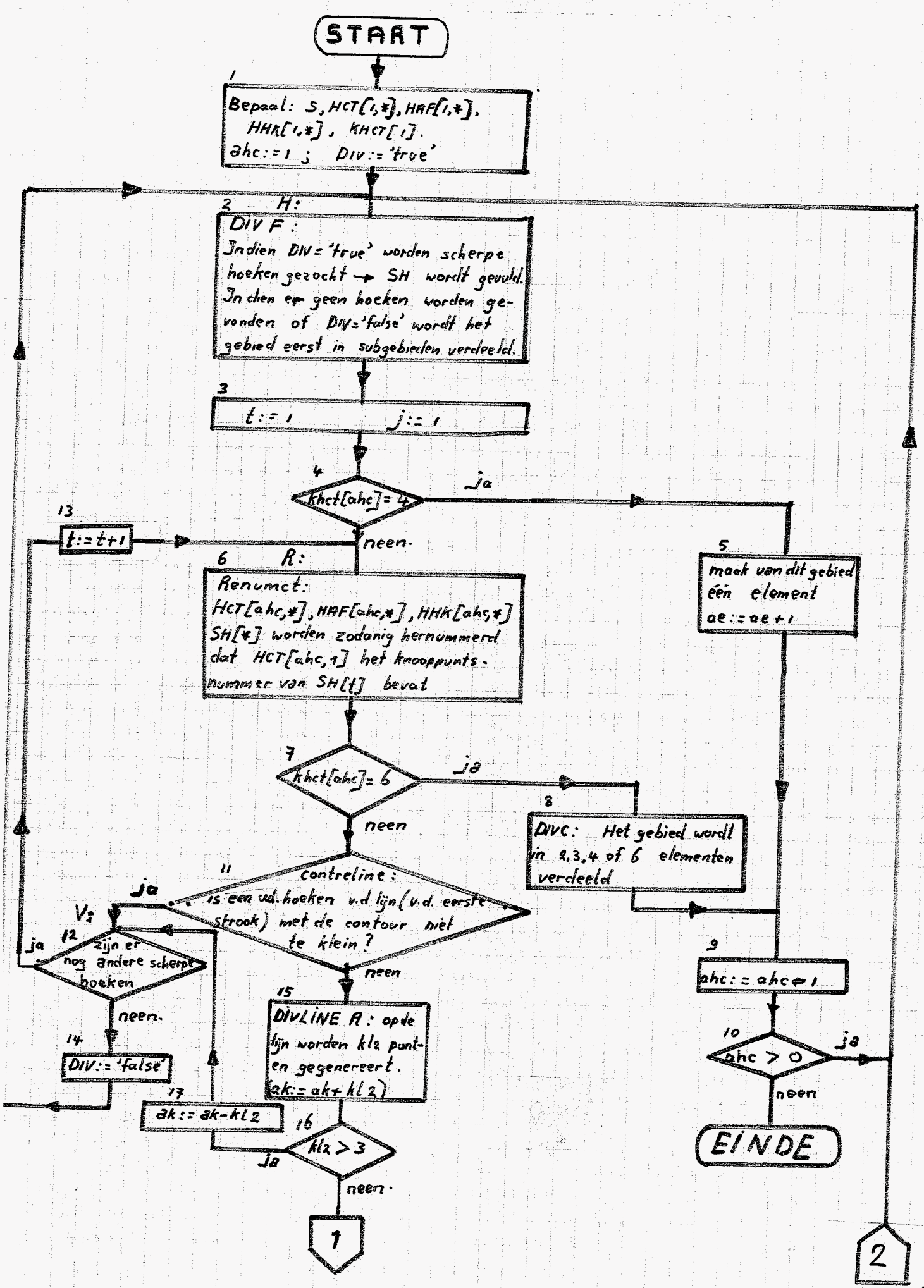
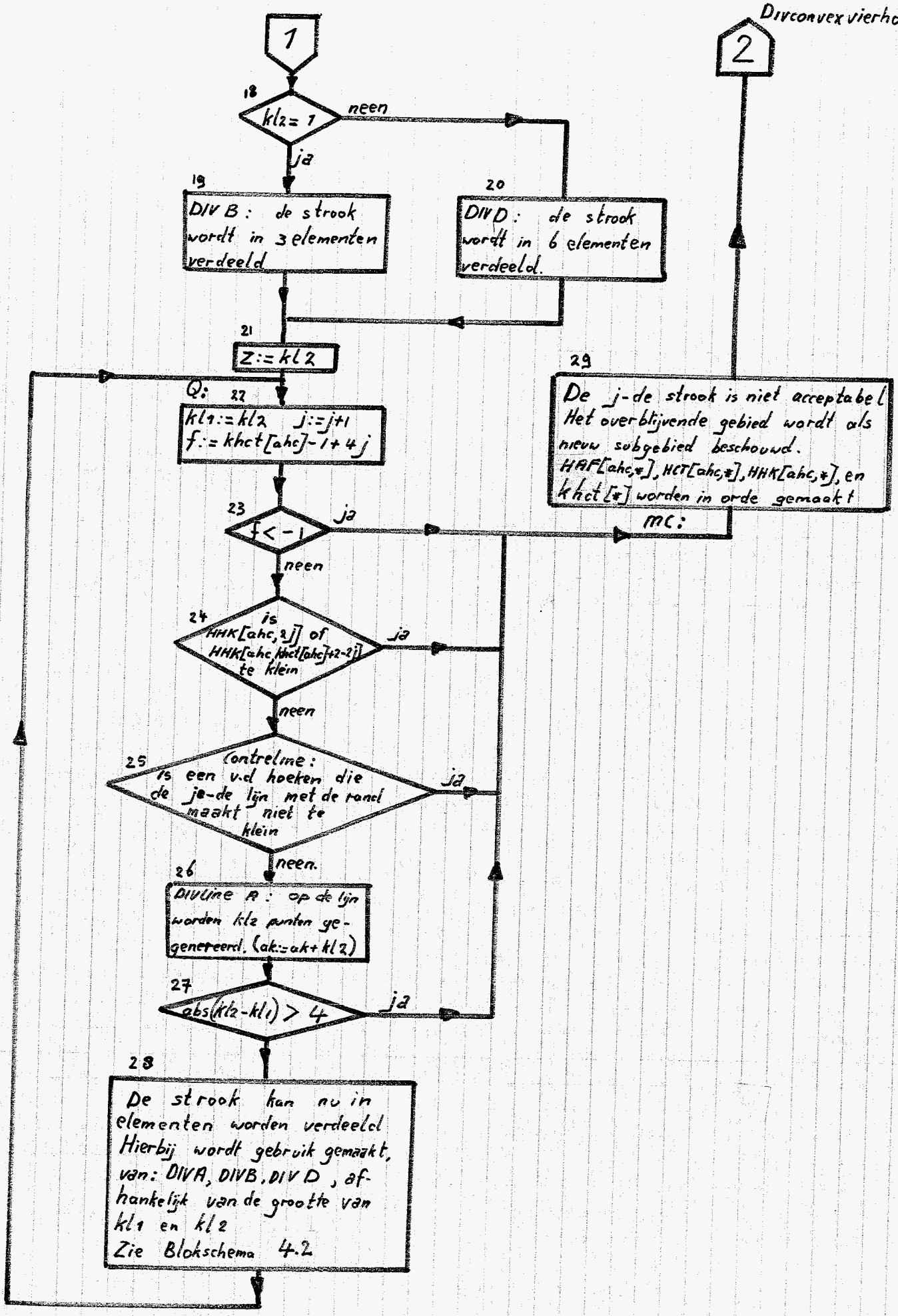


fig 13.

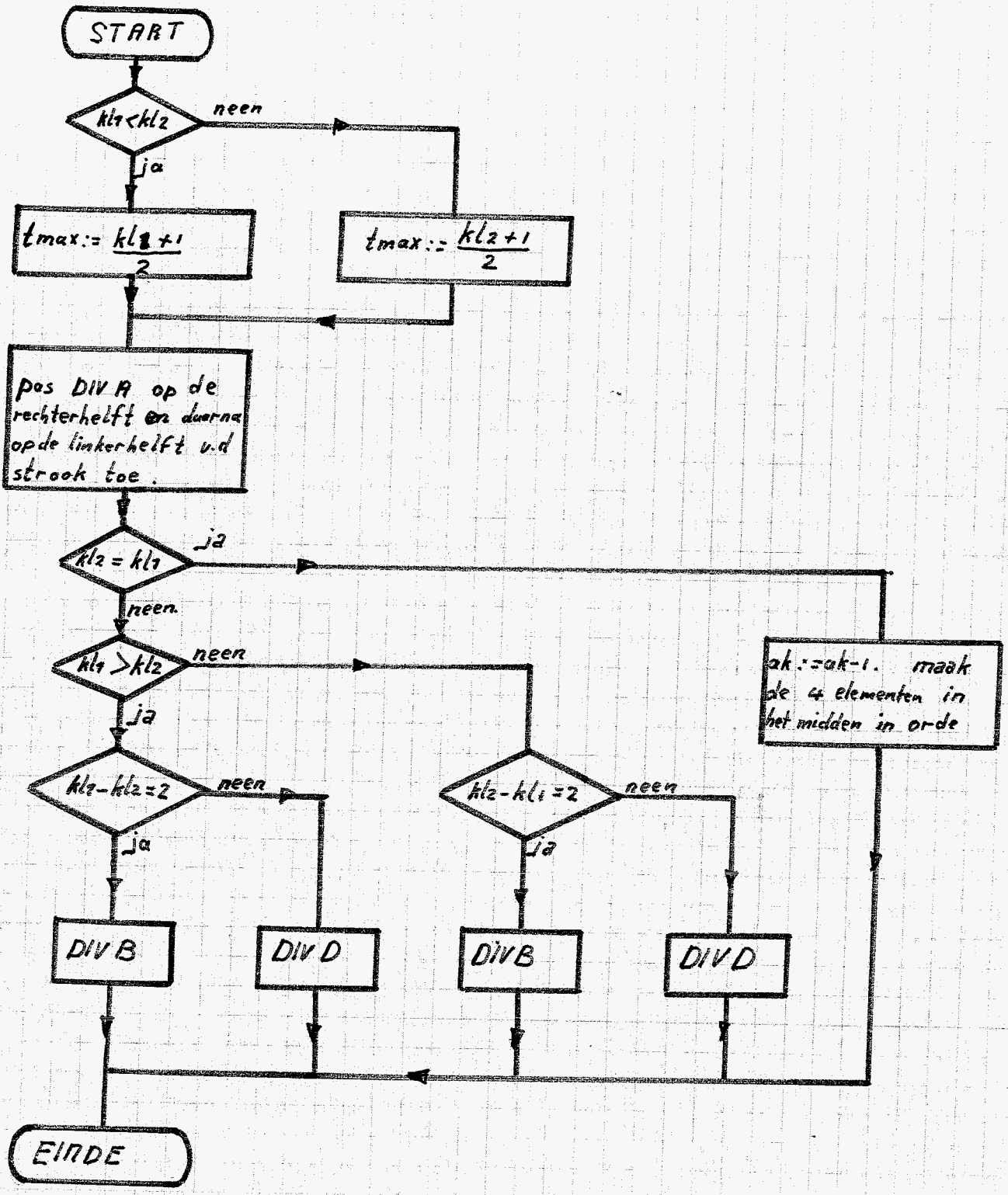
4.7

Blokschema




1
2
18
kl2 = 1
neen
ja
19
DIV B : de strook wordt in 3 elementen verdeeld
20
DIV D : de strook wordt in 6 elementen verdeeld.
21
z := kl2
Q:
22
kl1 := kl2 j := j+1
f := khct[ahc] - 1 + 4j
23
f < -1
ja
neen
24
is HHK[ahc,2j] of HHK[ahc,khct[ahc]+2-2j] te klein
ja
neen
25
contreline : is een v.d hoeken die de je-de lijn met de rand maakt niet te klein
ja
neen.
26
DIVline A : op de lijn worden kl2 punten ge-genereerd. (ak := ak + kl2)
27
abs(kl2 - kl1) > 4
ja
28
De strook kan nu in elementen worden verdeeld Hierbij wordt gebruik gemaakt, van: DIVA, DIVB, DIV D, afhankelijk van de grootte van kl1 en kl2
Zie Blokschema 4.2
29
De j-de strook is niet acceptabel Het overblijvende gebied wordt als nieuw subgebied beschouwd.
HAF[ahc,*], HCT[ahc,*], HHK[ahc,*], en khct[*] worden in orde gemaakt
mc:

4.2 Blokschema:
een strook in elementen verdelen (blok 28 van Hoofdblok)



5 Lokale parameters.

5.1 Inleiding

In het volgende worden een aantal lokale parameters besproken
Een aantal van deze parameters zijn gehanteerd in het Blokschema : 4
Een gedeelte van de parameters wordt tevens gebruikt als globale parameters bij de gehanteerde procedures. Bij de beschrijving van deze procedures zal vaak naar dit hoofdstuk worden verwezen.

5.2 **array** HCT — Array met de grenzen [1:p, 0 : kct+5]
In dit array worden de knooppuntsnummers van de contouren van de verschillende gemaakte subgebiedjes opgeslagen.
p is een ruime schatting van het max. te verwachten aantal subgebiedjes (p = 3 x ln kct)
kct+5 is een ruime schatting van het max aantal knooppunten dat op een hulpcontour kan voorkomen.

array HAF, HHK. — Zijn array's met de zelfde grenzen als HCT. en bevatten resp. de afstanden en de cosinus v.d hoeken van de verschillende hulpcontouren.

array KHCT — Array met de grenzen [1:p]
Dit array bevat het aantal knooppunten op de verschillende hulpcontouren.

integer ahc — geeft het laatst gegenereerde subgebied aan.

array SH — array met de grenzen [1:6].
In 3.1 hebben we gezien dat we beginnen met het zoeken v.d. meest "scherpe" hoek, die echter kleiner moet zijn dan een waarde α. Wanneer α b.v 90° is zullen we omdat we een convex gebied hebben maximaal 4 "scherpe" hoeken aantreffen. Wanneer HHK[ahc,i] de meest "scherpe" hoek is die kleiner is dan α dan krijgt SH[1] de waarde i
Wanneer HHK[ahc,j] de op een na scherpste hoek is, kleiner dan α dan krijgt SH[2] de waarde j etc...

integer	ash	geeft aan het aantal hoeken dat op de hierboven beschreven wijze is gevonden.
array	omt	array met de grenzen [o:kct], wordt gebruikt om de contour te beschrijven van gebiedjes waarin we een bepaalde standaardverdeling wensen aan te brengen (DIVA, DIVB, DIVC, DIVD)
integer	kl1, kl2	geven aan het aantal knooppunten die op een lijn komen te liggen. kl2: laatste lijn kl1: voorlaatste lijn
integer	f	geeft aan wanneer er te weinig knooppunten overblijven om een zinnige lijn te trekken
integer	tmax	geeft aan hoe vaak bij DIVA een bepaalde standaardverdeling moet worden gemaakt.
integer	j	een maat voor het aantal stroken dat afgehakt is.
real	s	de kortste afstand uit het array AF, wordt o.a. gebruikt als standaardlengte bij de grofheidsbepaling.
boolean	DIV	de procedure DIVF bestaat uit 2 fasen: het zoeken van "scherpe" hoeken en het verdelen in subgebieden Wanneer DIV de waarde <u>false</u> heeft wordt het zoeken van "scherpe" hoeken overgeslagen.

6 Discutabel.

De gevolgde methode in de procedure zal op een aantal plaatsen voor verbetering vatbaar zijn. Belangrijke maatstaven hiervoor zijn de verkregen elementvorm en de benodigde rekentijd.
Een criteria dat op beide maatstaven invloed heeft, is de toelaatbare hoek waaronder een lijn de contour mag snijden.
Is de toelaatbare hoek klein dan kunnen de elementen langs de rand een onaanvaarbare vorm krijgen. De benodigde rekentijd zal dan verminderen omdat dan minder subgebiedjes hoeven te ontstaan.
Het grootste deel v.d rekentijd is nodig voor het bepalen v.d. elementgrootte in het gebied. Dit gebeurt in de procedures: DIVA, DIVD en Divline A. m.b.v. de procedure BPG. Bij de betreffende procedures zal op dit belangrijk aspekt worden teruggekomen.

7. Programmatekst

```
'PROCEDURE' DIVCONVEXVIERHOEK(CO,CT,KCT,AF,HK,TD,AE,AK);                    DIVG
  'ARRAY'CO[1:1],CT[0],AF[0],HK[0],TD[1:1];'INTEGER'KCT,AE,AK;               DIVG
 'BEGIN' 'BOOLEAN'DIV;'INTEGER'ASH,J,AHC,P,KL1,KL2,F,K,TMAX,T,Z,L;           DIVG
         P:=3*LN(KCT);                                                      DIVG
  'BEGIN''ARRAY'SH[1:6],HCT[1:P,0:KCT+5],OMT[0:KCT],HAF[1:P,0:KCT+5],       DIVG
            HHK[1:P,0:KCT+5],KHCT[1:P];                                     DIVG
       'REAL'S,A,B,C,D,E,G;'LABEL'H,O T,R,G,MC,V;                           DIVG
    S:=1000;DIV:='TRUE';                KHCT[1]:=KCT;AHC:=1;                DIVG
    'FOR'J:=0'STEP'1'UNTIL'KCT'DO'                                          DIVG
    'BEGIN''IF'S'GTR'AF[J]'THEN'S:=AF[J];                                   DIVG
      HCT[1,J]:=CT[J];HAF[1,J]:=AF[J];HHK[1,J]:=HK[J];'END';                DIVG
      HCT[1,KCT+1]:=HCT[1,1];HAF[1,KCT+1]:=HAF[1,1];                        DIVG
      HHK[1,KCT+1]:=HHK[1,1];                                               DIVG
H:  DIVF(CO,KHCT,HCT,AHC,AF,SH,ASH,AK,DIV,S,CT,KCT,HAF,HHK);                DIVG
    T:=1;J:=1;'IF'KHCT[AHC]'EQL'4'THEN'                                     DIVG
                'BEGIN'TD[AE,1]:=HCT[AHC,1];TD[AE,2]:=HCT[AHC,2];           DIVG
                       TD[AE,3]:=HCT[AHC,3];TD[AE,4]:=HCT[AHC,4];           DIVG
                       AE:=AE+1;'GOTO'O T;'END';                            DIVG
R:  RENUMCT(HCT[AHC,*],KHCT[AHC],SH  ,HHK[AHC,*],HAF[AHC,*],ASH,T);         DIVG
    'IF'KHCT[AHC]'EQL'6'THEN'                                               DIVG
```

```
        'BEGIN'DIVC(CO,TD,AK,AE,HHK[AHC,*],HCT[AHC,*]);'GOTO'O T;            DIVG 18
        'END';                                                              DIVG 18
    'IF'CONTRLINE(CO,KHCT[AHC],HCT[AHC,*],HAF[AHC,*],0.8,J)'THEN'           DIVG 19
I:      'BEGIN'T:=T+1;'IF'T'LEQ'ASH'THEN''GOTO'R'ELSE'                      DIVG 20
            'BEGIN'DIV:='FALSE';'GOTO'H;'END';'END';                        DIVG 21
    DIVLINEA(KC,CT,CO,AF,HCT[AHC,*],KHCT[AHC],AK,2*J+1,KHCT[AHC]            DIVG 22
      =2*J+1,'FALSE','TRUE',S,KL2);                                         DIVG 23
    'IF'KL2'GTR'3'THEN''BEGIN'AK:=AK-KL2;'GOTO'V;'END';                     DIVG 24
    OMT[1]:=HCT[AHC,1];OMT[4]:=AK-1;                                        DIVG 25
    OMT[2]:=HCT[AHC,2];OMT[5]:=HCT[AHC,KHCT[AHC]-1];                        DIVG 26
MT[3]:=HCT[AHC,3];OMT[6]:=HCT[AHC,KHCT[AHC]];                               DIVG 27
    'IF'KL2'EQL'1'THEN'DIVB(CO,TD,AK,AE,OMT)                                DIVG 28
    'ELSE''BEGIN'OMT[7]:=OMT[5];OMT[8]:=OMT[6];OMT[4]:=AK-1;                DIVG 29
              OMT[5]:=AK-2  ;OMT[6]:=AK-3;                                  DIVG 30
          DIVD(KCT,CT,CO,AF,TD,AK,AE,OMT);'END';                            DIVG 31
    Z:=KL2;                                                                 DIVG 32
I:  KL1:=KL2;J:=J+1;F:=KHCT[AHC]-1-4*J;                                     DIVG 33
    'IF'F'LEQ'1'THEN''GOTO'MC;                                              DIVG 34
    'IF'HHK[AHC,2*J]'GTR'-0.1'OR'HHK[AHC,KHCT[AHC]+2-2*J]                   DIVG 34
          'GTR'-0.1'THEN''GOTO'MC;                                          DIVG 34
              'IF'CONTRLINE(CO,KHCT[AHC],HCT[AHC,*],HAF[AHC,*],0.8,J)DIVG 34
      'THEN''GOTO'MC;                                                       DIVG 35
    DIVLINEA(KCT,CT,CO,AF,HCT[AHC,*],KHCT[AHC],AK,2*J+1,KHCT[AHC]           DIVG 36
      =2*J+1,'FALSE','TRUE',S,KL2);                                         DIVG 37
    'IF'ABS(KL1-KL2)'GTR'4'THEN''BEGIN'AK:=AK-KL2;'GOTO'MC'END';            DIVG 38
  'IF'KL1'LSS'KL2'THEN'TMAX:=(KL1+1)/2'ELSE'TMAX:=(KL2+1)/2;                DIVG 39
    'FOR'K:=1'STEP'1'UNTIL'TMAX'DO'                                         DIVG 40
      'BEGIN'OMT[K]:=AK-  1-KL2-TMAX+K-Z;OMT[TMAX+3+K]:=AK-K;               DIVG 41
      'END'; OMT[TMAX+1]:=HCT[AHC,2*J-1];                                   DIVG 42
             OMT[TMAX+2]:=HCT[AHC,2*J  ];                                   DIVG 43
             OMT[TMAX+3]:=HCT[AHC,2*J+1];                                   DIVG 44
             DIVA(KCT,CT,CO,AF,TD,AK,AE,TMAX,OMT);                          DIVG 45
```

```
    'FOR'K:=1'STEP'1'UNTIL'TMAX'DO'                                          DIVG 460
      'BEGIN'OMT[K]:=AK-KL2-KL;                                              DIVG 470
             OMT[TMAX+3+K]:=AK-KL2-KL1-Z-1+K-TMAX;'END';                     DIVG 480
             OMT[TMAX+1]:=HCT[AHC,KHCT[AHC]-2*J+1];                          DIVG 490
             OMT[TMAX+2]:=HCT[AHC,KHCT[AHC]-2*J+2];                          DIVG 500
             OMT[TMAX+3]:=HCT[AHC,KHCT[AHC]-2*J+3];                          DIVG 510
             DIVA(KCT,CT,CO,AF,TD,AK,AE,TMAX,OMT);                           DIVG 520
    'IF'KL2'EQL'KL1'THEN''BEGIN'AK:=AK-1;TD[AE-1,4]:=AK-TMAX;                DIVG 530
                                          TD[AE-2,4]:=AK-TMAX;'END'          DIVG 540
    'ELSE''BEGIN'                                                            DIVG 541
    'IF'KL1'GTR'KL2'THEN'                                                    DIVG 550
      'BEGIN'OMT[1]:=AK-3*TMAX;OMT[2]:=AK-1;                                 DIVG 560
        OMT[3]:=AK-  TMAX-KL2-Z-KL1- 1;OMT[4]:=OMT[3]+1;                     DIVG 570
        OMT[5]:=OMT[4]+1;OMT[6]:=AK-TMAX-1;                                  DIVG 580
      'IF'KL1 -KL2'EQL'2'THEN'DIVB(CO,TD,AK,AE,OMT)                          DIVG 590
      'ELSE''BEGIN'OMT[6]:=OMT[5]+1;OMT[7]:=OMT[6]+1;OMT[8]:=AK-             DIVG 600
             TMAX-1;DIVD(KCT,CT,CO,AF,TD,AK,AE,OMT);                         DIVG 610
          'END';                                                             DIVG 620
      'END'                                                                  DIVG 630
    'ELSE''BEGIN'OMT[1]:=AK-3*TMAX-KL2-Z;OMT[2]:=AK-TMAX-1;                  DIVG 640
        OMT[3]:=AK-3*TMAX    ;OMT[4]:=OMT[3]-1;                              DIVG 650
        OMT[5]:=OMT[4]-1;OMT[6]:=AK-1;                                       DIVG 660
        'IF'KL2-KL1'EQL'2'THEN'DIVB(CO,TD,AK,AE,OMT)                         DIVG 670
        'ELSE''BEGIN'OMT[6]:=OMT[5]-1;OMT[7]:=OMT[6]-1;                      DIVG 680
             OMT[8]:=AK-1;DIVD(KCT,CT,CO,AF,TD,AK,AE,OMT);                   DIVG 690
          'END';                                                             DIVG 700
      'END';                                                                 DIVG 710
```

```
    'END';                                                                   DIVG 701
    Z:=2*TMAX+ABS(KL2-KL1)-1;'GOTO'C;                                        DIVG 720
4C: L:=KHCT[AHC]-4*J+5;                                                      DIVG 730
    'FOR'K:=1'STEP'1'UNTIL'L'DO'                                             DIVG 740
    'BEGIN'HCT[AHC,K]:=HCT[AHC,2*J-2+K];HHK[AHC,K]:=HHK[AHC,2*J-2+K];        DIVG 750
           HAF[AHC,K]:=HAF[AHC,2*J-2+K];'END';                               DIVG 760
    'FOR'K:=1'STEP'1'UNTIL'KL1'DO'                                           DIVG 770
    'BEGIN'HCT[AHC,L+K]:=AK-Z-KL1-1+K;HHK[AHC,L+K]:=-1;'END';                DIVG 780
    A:=CO[HCT[AHC,L-1],1];C:=CO[HCT[AHC,L],1];E:=CO[HCT[AHC,L+1],1];         DIVG 790
    B:=CO[HCT[AHC,L-1],2];D:=CO[HCT[AHC,L],2];G:=CO[HCT[AHC,L+1],2];         DIVG 800
    HHK[AHC,L]:=HKCOS(A,B,C,D,E,G);HAF[AHC,L]:=SQRT((E-C)**2+                DIVG 810
    (G-D)**2);HCT[AHC,L+KL1+1]:=HCT[AHC,1];                                  DIVG 820
    A:=E;B:=G;'FOR'K:=1'STEP'1'UNTIL'KL1'DO'                                 DIVG 830
    'BEGIN'C:=CO[HCT[AHC,L+1+K],1];                                          DIVG 840
           D:=CO[HCT[AHC,L+1+K],2];                                          DIVG 850
           HAF[AHC,L+K]:=SQRT((A-C)**2+(B-D)**2);A:=C;B:=D;'END';            DIVG 860
    C:=CO[HCT[AHC,L+KL1],1];E:=CO[HCT[AHC,2],1];                             DIVG 870
    D:=CO[HCT[AHC,L+KL1],2];G:=CO[HCT[AHC,2],2];                             DIVG 880
    HHK[AHC,1]:=HKCOS(C,D,A,B,E,G);                                          DIVG 890
    HHK[AHC,0]:=HHK[AHC,L+KL1];HHK[AHC,L+KL1+1]:=HHK[AHC,1];                 DIVG 900
    HCT[AHC,0]:=HCT[AHC,L+KL1];                                              DIVG 910
    HAF[AHC,0]:=HAF[AHC,L+KL1];HAF[AHC,L+KL1+1]:=HAF[AHC,1];                 DIVG 920
    KHCT[AHC]:=L+KL1;                                                        DIVG 930
    'GOTO'H;                                                                 DIVG 940
T :  AHC:=AHC-1;'IF'AHC'GTR'0'THEN''GOTO'H;                                  DIVG 950
  'END';                                                                     DIVG 951
END'DIVCONVEXVIERHOEK;                                                       DIVG 960
```

Procedure DIV A

1. Inleiding.

De procedure DIV A verdeelt een gebied zoals aangegeven in fig 1 in elementen.

fig 1

2. Gebruiksaanwijzing.

DIVA (kct, ct, co, af, td, ak, ae, tmax, omt, s)

2.1 Globale parameters

procedure BPG

2.2 Formele parameters

integer	kct	} Zie	Divconvexvierhoek 2.2.
array	ct		
array	co		
array	af		
array	td		
array	ak		
array	ae		
integer	tmax	} Zie	Divconvexvierhoek 5.2
array	omt		

3 Werking v.d procedure

voorbeeld

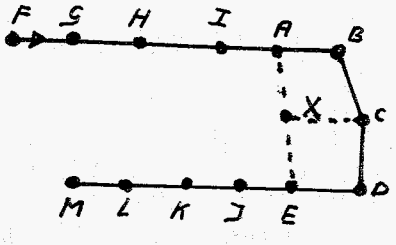


fig 2

omt[1] = F (knooppuntsnr.)
omt[2] = G
⋮
omt[2xtmax+3] = M

t max = 5

We trekken lijn AE. In de punten A en E bepalen we de grofheid en we verdelen daarna de lijn AE in 2 stukken die zich verhouden als de grofheid in A en E → knooppunt X.
We hebben nu 2 elementen: ABCX en XCDE.
We herhalen dit door IJ te trekken, daarna HK ... etc dus in totaal voeren we tmax maal een standaardverdeling uit. We genereren tmax knooppunten en 2 x tmax elementen.

4 Discutabel

Voor elke 2 elementen die gegenereerd worden, moet 2 maal de grofheid worden bepaald. Deze grofheidsbepaling kost vrij veel rekentijd. We zouden deze achterwege kunnen laten door b.v.:

a) X in het midden van AE te leggen of:

b) AE te verdelen in de verhouding (AB + AI) : (ED + EI).

We hebben in deze procedure dan de formele parameters: fct, ct en af niet nodig.
Voor deze procedure kan de benodigde rekentijd dan een factor 100 lager worden

5 Programmatekst

```
'PROCEDURE'DIVA(KCT,CT,CO,AF,TD,AK,AE,TMAX,OMT);                    DIVA  10
  'INTEGER'KCT,AK,AE,TMAX;                                          DIVA  20
  'ARRAY'CT[0],CO[1,1],AF[0],TD[1,1],OMT[0];                        DIVA  40
  'BEGIN''REAL'GA,GB,A,B,C,D;                                       DIVA  50
        'INTEGER'T;                                                 DIVA  60
    'FOR'T:=0'STEP'1'UNTIL'TMAX-1'DO'                               DIVA  70
    'BEGIN'A:=CO[OMT[TMAX-T],1];C:=CO[OMT[TMAX+4+T],1];            DIVA  80
           B:=CO[OMT[TMAX-T],2];D:=CO[OMT[TMAX+4+T],2];             DIVA  90
           BPG(CO,CT,A,B,KCT,GA,1,AF);                              DIVA 100
           BPG(CO,CT,C,D,KCT,GB,1,AF);                              DIVA 110
           CO[AK,1]:=A+GA/(GA+GB)*(C-A);                            DIVA 120
           CO[AK,2]:=B+GA/(GA+GB)*(D-B);                            DIVA 130
           TD[AE,1]:=OMT[TMAX-T];TD[AE,2]:=OMT[TMAX+1-T];           DIVA 140
           TD[AE,3]:=OMT[TMAX+2-T];TD[AE,4]:=AK;AE:=AE+1;           DIVA 150
           TD[AE,1]:=OMT[TMAX+2+T];TD[AE,2]:=OMT[TMAX+3+T];         DIVA 160
           TD[AE,3]:=OMT[TMAX+4+T];TD[AE,4]:=AK;                    DIVA 170
           OMT[TMAX-T+1]:=AK;OMT[TMAX+T+3]:=AK;                     DIVA 180
           AK:=AK+1;AE:=AE+1;                                       DIVA 190
    'END';                                                          DIVA 200
  'END'DIVA;                                                        DIVA 210
```

Procedure DIV B

1 Inleiding

De procedure verdeeld een gebied zoals is aangegeven in fig 1 in 3 elementen door in het zwaartepunt v.d knooppunten: omt[2], omt[4] en omt[6] een knooppunt te leggen.

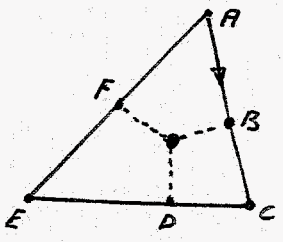


fig 1

omt[1] = A
⋮
omt[6] = F

2 Gebruiksaanwijzing

DIVB (co, td, ak, ae, omt)

2.1 Formele parameters

array	co	Zie Divconvexvierhoek 2.2
array	td	
integer	ak	
integer	ae	
array	omt →	Zie Divconvexvierhoek 5.2

Discutabel

De motivering van de situering van het knooppunt is:

a) We krijgen nooit elementen met hoeken > 180°

b) procedure vraagt nauwelijks rekentijd.

We zouden evenals bij DIV A het knooppunt kunnen genereren, door een aantal malen de grofheid te bepalen. We zullen de hoeken v.d ontstane elementen dan echter moeten controleren.

Ook zouden we het punt zodanig kunnen plaatsen dat de 3 bijbehorende hoeken elk zoveel mogelijk 120° zijn. Beide methodes zullen echter veel meer rekentijd vergen.

4 Programmatekst

```
'PROCEDURE'DIVB(CO,TD,AK,AE,OMT);                                          DIVB   1
  'INTEGER' AK,AE;'ARRAY'CO[1,1],TD[1,1],OMT[0];                           DIVB   2
  'BEGIN'                                                                  DIVB   3
  CO[AK,1]:=(CO[OMT[2],1]+CO[OMT[4],1]                                     DIVB   4
                                  +CO[OMT[6],1])/3;                        DIVB   5
  TD[AE,1]:=OMT[1];TD[AE,2]:=OMT[2];TD[AE,3]:=AK;TD[AE,4]:=OMT[6];         DIVB   6
  CO[AK,2]:=(CO[OMT[2],2]+CO[OMT[4],2]                                     DIVB   7
                                  +CO[OMT[6],2])/3;                        DIVB   8
  AE:=AE+1;                                                                DIVB   9
  TD[AE,1]:=OMT[3];TD[AE,2]:=OMT[4];TD[AE,3]:=AK;TD[AE,4]:=OMT[2];         DIVB  10
  AE:=AE+1;                                                                DIVB  11
  TD[AE,1]:=OMT[5];TD[AE,2]:=OMT[6];TD[AE,3]:=AK;TD[AE,4]:=OMT[4];         DIVB  12
  AE:=AE+1;AK:=AK+1;                                                       DIVB  13
  'END'DIVB;                                                               DIVB  14
```

Procedure DIVC

1 Inleiding

De procedure verdeelt een gebied zoals is aangegeven in fig 1 in 2, 3, 4 of 6 elementen. Welke verdeling gekozen wordt is afhankelijk v.d. vorm van het gebied.

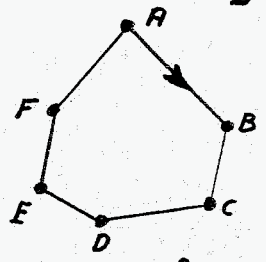


fig. 1

omt[1] = A (scherpste hoek) hk[1] = cosinus A
⋮ ⋮
omt[6] = F hk[6] = cosinus F

2 Gebruiksaanwijzing

DIVC(co, td, ak, ae, hk, omt)

.1 Globale parameters

procedure DIVB

.2 Formele parameters

array co
array td
array ak
array ae

} zie Divconvexvierhoek 2.2

array hk — array met de grenzen [0:6], bevat de cosinus v.d. hoeken v.h. gebied.

array omt — zie Divconvexvierhoek s. 2.

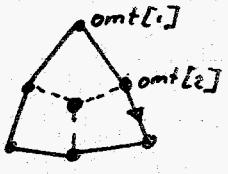
3 Werking v.d procedure

De cosinus v.d hoeken van het gebied is voor ons het criterium wat bepaalt welke verdeling ontstaat Voor de gehanteerde criteria wordt verwezen naar 5: Blokschema.

De volgende 5 verdelingen zijn mogelijk :

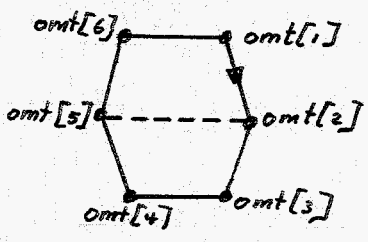
3.1 verdeling 1

Dit is de reeds behandelde procedure DIV B :



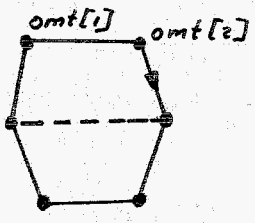
Er worden een knooppunt en drie elementen gegenereerd.

3.2 verdeling 2



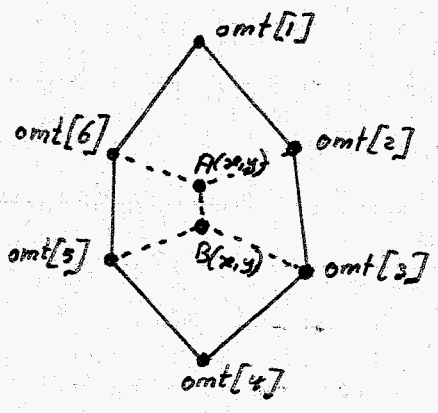
Er worden geen knooppunten twee elementen gegenereerd.

3.3 verdeling 3



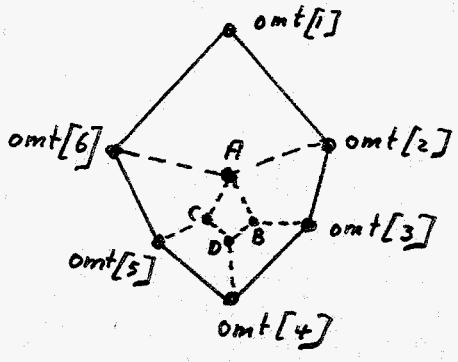
Er worden geen knooppunt en twee elementen gegenereerd.

3.4 verdeling 4



Er worden 2 knooppunten gegenereerd. In de procedure gebeurt dit door A_x te leggen op het gemiddelde van 2 maal de x-coördinaat van omt[2] en omt[6] en 1 maal de x-coördinaat van omt[3] en omt[5]
A_y komt te liggen op het gemiddelde van de overeenkomstige y-coördinaten.
B_x komt te liggen op het gemiddelde van 2 maal de x-coördinaat van omt[3] en omt[5] en 1 maal de x-coördinaat van omt[2] en omt[6]. B_y komt te liggen op het gemiddelde v.d overeenkomstige y-coördinaten. Er worden in totaal 4 elementen gegenereerd.

3.5 verdeling 5

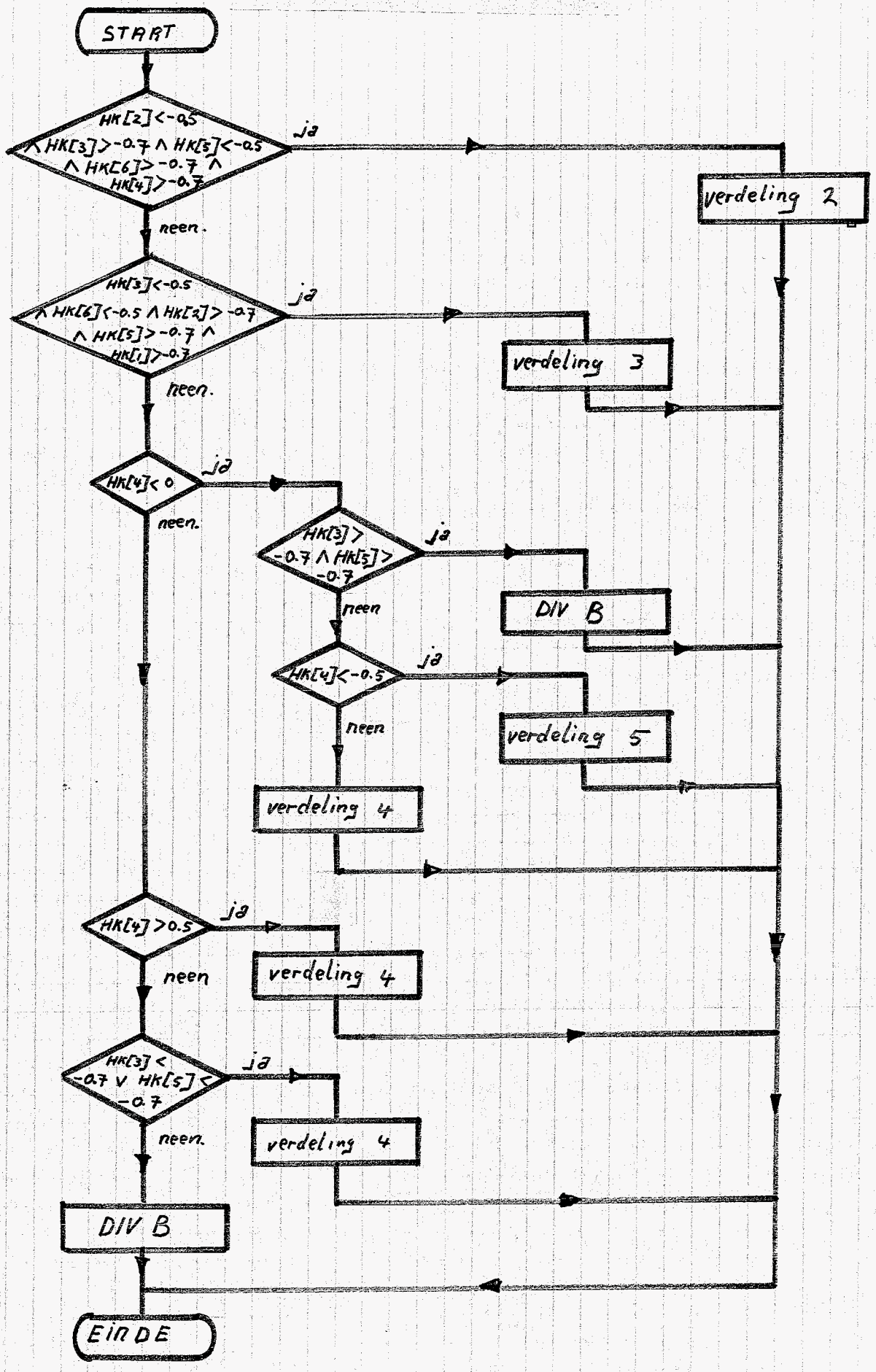


Er worden 4 knooppunten gegenereerd.
A wordt op de zelfde manier gevonden als A van 3.4
B komt op het zwaartepunt van A, omt[3] en omt[4].
C komt op het zwaartepunt van A, omt[4] en omt[5].
D komt op het zwaartepunt van B, omt[4] en C.
De verkregen verdeling is niet idiaal, ze zal echter niet vaak hoeven voor te komen.

4 Discutabel

De verschillende criteria (zie e.v. blokschema) zijn voor verandering vatbaar.
Anderen dan de gehanteerde verdelingen zijn mogelijk.
De positionering van de knooppunten kan ook m.b.v. grofheidsbepaling tot stand komen, de rekentijd zal dan echter toenemen.
Gezien het feit dat deze procedure in het totale programma niet al te vaak gehanteerd wordt, alleen wanneer gebiedjes met 6 knooppunten overblijven, kan verbetering van deze procedure in eerste instantie beter achterwegen blijven.

5 Blokschema



6 Programmatekst

```
'PROCEDURE'DIVC(CO,TD,AK,AE,HK,OMT);                                    DIVC  10
  'INTEGER'AK,AE;'ARRAY'CO[1,1],TD[1,1],HK[0],OMT[0];                   DIVC  20
  'BEGIN''LABEL'P,Q,OUT;'INTEGER'J;                                     DIVC  30
  'IF'HK[2]'LSS'=0.5'AND'HK[3]'GTR'=0.7'AND'HK[5]'LSS'=0.5'AND'         DIVC  31
  HK[6]     'GTR'=0.7'AND'HK[4]'GTR'=0.7'THEN'                          DIVC  32
  'BEGIN''FOR'J:=1'STEP'1'UNTIL'4'DO'TD[AE,J]:=OMT[J+1];AE:=AE+1;       DIVC  33
        TD[AE,1]:=OMT[5];TD[AE,2]:=OMT[6];TD[AE,3]:=OMT[1];TD[AE,4]:=   DIVC  35
        OMT[2];AE:=AE+1;'GOTO'OUT;                                      DIVC  36
  'END';                                                                DIVC 36A
  'IF'HK[3]'LSS'=0.5'AND'HK[6]'LSS'=0.5'AND'                            DIVC  37
   HK[2]'GTR'=0.7'AND'HK[5]'GTR'=0.7'AND'HK[4]'GTR'=0.7'THEN'           DIVC  38
   'BEGIN''FOR'J:=1'STEP'1'UNTIL'3'DO'TD[AE,J]:=OMT[J];                DIVC  39
        TD[AE,4]:=OMT[6];AE:=AE+1;                                      DIVC 39B
        'FOR'J:=1'STEP'1'UNTIL'4'DO'TD[AE,J]:=OMT[J+2];AE:=AE+1;        DIVC 39C
  'GOTO'OUT;'END';                                                      DIVC 39D
'IF'HK[4]'LEQ'0'THEN'                                                   DIVC  40
    'BEGIN''IF'HK[3]'GTR'=0.7'AND'HK[5]'GTR'=0.7'THEN'                  DIVC  50
      'BEGIN'DIVB(CO,TD,AK,AE,OMT);'GOTO'OUT;'END'                      DIVC  60
        'ELSE''BEGIN''IF'HK[4]'LEQ'=0.5'THEN''GOTO'P                    DIVC  70
                  'ELSE''GOTO'Q;                                        DIVC  80
              'END'                                                     DIVC  90
    'END'                                                               DIVC 100
'ELSE''BEGIN''IF'HK[4]'GTR'0.5'THEN''GOTO'Q                            DIVC 110
          'ELSE''BEGIN''IF'HK[3]'LSS'=0.7'OR'HK[5]'LSS'=0.7             DIVC 120
                  'THEN''GOTO'Q                                         DIVC 130
                  'ELSE''BEGIN'DIVB(CO,TD,AK,AE,OMT);                   DIVC 140
                        'GOTO'OUT;'END';                                DIVC 150
              'END';                                                    DIVC 160
    'END';                                                              DIVC 170
P: TD[AE,1]:=OMT[1];TD[AE,2]:=OMT[2];TD[AE,3]:=AK    ;TD[AE,4]:=OMT[6]; DIVC 180
  AE:=AE+1;                                                             DIVC 190
  TD[AE,1]:=OMT[2];TD[AE,2]:=OMT[3];TD[AE,3]:=AK+1  ;TD[AE,4]:=AK    ;  DIVC 200
  AE:=AE+1;                                                             DIVC 210
  TD[AE,1]:=OMT[3];TD[AE,2]:=OMT[4];TD[AE,3]:=OMT[5];TD[AE,4]:=AK+1  ;  DIVC 220
  AE:=AE+1;                                                             DIVC 230
  TD[AE,1]:=OMT[5];TD[AE,2]:=OMT[6];TD[AE,3]:=AK    ;TD[AE,4]:=AK+1  ;  DIVC 240
  AE:=AE+1;                                                             DIVC 250
  CO[AK,1]:=(CO[OMT[6],1]*2+CO[OMT[2],1]*2+CO[OMT[3],1]+                DIVC 260
                                        CO[OMT[5],1])/6;                DIVC 270
  CO[AK,2]:=(CO[OMT[6],2]*2+CO[OMT[2],2]*2+CO[OMT[3],2]+                DIVC 280
                                        CO[OMT[5],2])/6;AK:=AK+1;       DIVC 290
  CO[AK,1]:=(CO[OMT[3],1]*2+CO[OMT[5],1]*2+CO[OMT[6],1]+                DIVC 300
                                        CO[OMT[2],1])/6;                DIVC 310
  CO[AK,2]:=(CO[OMT[3],2]*2+CO[OMT[5],2]*2+CO[OMT[6],2]+                DIVC 320
                                        CO[OMT[2],2])/6;AK:=AK+1;       DIVC 330
  'GOTO'OUT;                                                            DIVC 340
Q: TD[AE,1]:=OMT[1];TD[AE,2]:=OMT[2];TD[AE,3]:=AK    ;TD[AE,4]:=OMT[6];DIVC 350
  AE:=AE+1;                                                             DIVC 360
  TD[AE,1]:=OMT[2];TD[AE,2]:=OMT[3];TD[AE,3]:=AK+1  ;TD[AE,4]:=AK    ;  DIVC 370
  AE:=AE+1;                                                             DIVC 380
  TD[AE,1]:=OMT[3];TD[AE,2]:=OMT[4];TD[AE,3]:=AK+3  ;TD[AE,4]:=AK+1  ;  DIVC 390
  AE:=AE+1;                                                             DIVC 400
  TD[AE,1]:=OMT[4];TD[AE,2]:=OMT[5];TD[AE,3]:=AK+2  ;TD[AE,4]:=AK+3  ;  DIVC 410
  AE:=AE+1;                                                             DIVC 420
  TD[AE,1]:=OMT[5];TD[AE,2]:=OMT[6];TD[AE,3]:=AK    ;TD[AE,4]:=AK+2  ;  DIVC 430
  AE:=AE+1;                                                             DIVC 440
  TD[AE,1]:=AK     ;TD[AE,2]:=AK+1  ;TD[AE,3]:=AK+3  ;TD[AE,4]:=AK+2  ;  DIVC 450
  AE:=AE+1;                                                             DIVC 460
  CO[AK,1]:=(CO[OMT[2],1]+CO[OMT[3],1]+CO[OMT[5],1]+CO[OMT[6],1])/4;    DIVC 470
```

```
    CO[AK,2]:=(CO[OMT[2],2]+CO[OMT[3],2]+CO[OMT[5],2]+CO[OMT[6],2])/4;    DIVC 48(
    AK:=AK+1;                                                            DIVC 49(
    CO[AK,1]:=(CO[OMT[3],1]+CO[OMT[4],1]+CO[AK-1,1])/3;                  DIVC 50(
    CO[AK,2]:=(CO[OMT[3],2]+CO[OMT[4],2]+CO[AK-1,2])/3;AK:=AK+1;         DIVC 51(
    CO[AK,1]:=(CO[OMT[4],1]+CO[OMT[5],1]+CO[AK-2,1])/3;                  DIVC 52(
    CO[AK,2]:=(CO[OMT[4],2]+CO[OMT[5],2]+CO[AK-2,2])/3;AK:=AK+1;         DIVC 53(
    CO[AK,1]:=(CO[OMT[4],1]+CO[AK-1  ,1]+CO[AK-2,1])/3;                  DIVC 54(
    CO[AK,2]:=(CO[OMT[4],2]+CO[AK-1  ,2]+CO[AK-2,2])/3;AK:=AK+1;         DIVC 55(
OUT: 'END'DIVC;                                                          DIVC 56(
```

Procedure DIVD

1 Inleiding

De procedure verdeelt een gebied zoals aangegeven in fig 1 in 6 elementen, door procedure DIVB 2 maal toe te passen.

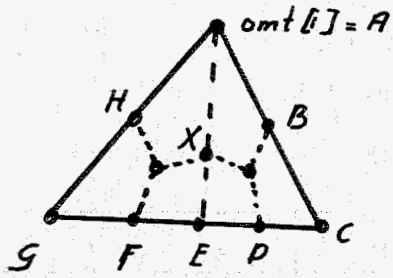


fig 1

omt [1] = A
omt [2] = B
⋮
omt [8] = H

2 Gebruiksaanwijzing

DIVD(kct, ct, co, af, td, ak, ae, omt)

2.1 Globale parameters

procedure	DIVB
procedure	BPG

2.2 Formele parameters

integer	kct	Zie Divconvexvierhoek 2.2
array	ct	
array	co	
array	af	
array	td	
integer	ak	
integer	ae	
array	omt	Zie Divconvexvierhoek 5.2

3 Werking v.d. procedure

Het gebied wordt in 2 stukken verdeeld door de lijn AE te trekken. In A en E bepalen we de grofheid (m.b.v. Bpg) en we verdelen de lijn AE in stukken die zich verhouden als de grofheden in A en E. Op de gebieden C, D, E, X, A, B, en G, H, A, X, E, F passen we DIVB toe. Als omt[i] wordt bij deze gebieden resp C en G genomen.

4 Discutabel

Positionering van X kan ook gebeuren door X te leggen op het zwaartepunt van H, B en E. → minder rekentijd.

5 Programmatekst

```
'PROCEDURE'DIVD(KCT,CT,CO,AF,TD,AK,AE,OMT);                          DIVD  10
  'INTEGER'KCT,AK,AE;                                                DIVD  20
  'ARRAY'CT[0],CO[1,1],AF[0],TD[1,1],OMT[0];                         DIVD  30
  'BEGIN''REAL'GA,GB,A,B,C,D;                                        DIVD  40
        'ARRAY'HOMT[0:6];                                            DIVD  50
    A:=CO[OMT[1],1];B:=CO[OMT[1],2];                                 DIVD  60
    C:=CO[OMT[5],1];D:=CO[OMT[5],2];                                 DIVD  70
    BPG(CO,CT,A,B,KCT,GA,1,AF);                                      DIVD  80
    BPG(CO,CT,C,D,KCT,GB,1,AF);                                      DIVD  90
    CO[AK,1]:=A+GA/(GA+GB)*(C-A);                                    DIVD 100
    CO[AK,2]:=B+GA/(GA+GB)*(D-B);                                    DIVD 110
    HOMT[1]:=OMT[3];HOMT[2]:=OMT[4];HOMT[3]:=OMT[5];HOMT[4]:=AK;     DIVD 120
    HOMT[5]:=OMT[1];HOMT[6]:=OMT[2];AK:=AK+1;                        DIVD 130
    DIVB(CO,TD,AK,AE,HOMT);                                          DIVD 140
    HOMT[1]:=OMT[7];HOMT[2]:=OMT[8];HOMT[3]:=OMT[1];                 DIVD 150
    HOMT[5]:=OMT[5];HOMT[6]:=OMT[6];                                 DIVD 160
    DIVB(CO,TD,  AK,AE,HOMT);                                        DIVD 170
  'END'DIVD;                                                         DIVD 180
```

Procedure DIVF

1. Inleiding

De procedure F heeft 2 functies:

a) Zij zoekt "scherpe" hoeken kleiner dan een waarde alfa en slaat ze in volgorde van grootte op in array SH

b) Indien Div = false , of indien er geen "scherpe" hoeken gevonden worden verdeelt ze het gebied in 2 subgebieden

2 Gebruiksaanwijzing

DIVF(co, khct, hct, ahc, af, sh, ash, ak, DIV, s, ct, kct, haf, hhk).

2.1 Globale parameters

procedure	Divline A
procedure	Makeline A
procedure	Hkcos

2.2 Formele parameters

array	co	Zie Divconvexvierhoek 2.2
array	af	
integer	ak	
real	s	
array	ct	
integer	kct	
array	khct	Zie Divconvexvierhoek 5.2
array	hct	
integer	ahc	
array	SH	
integer	ash	
boolean	DIV	
array	haf	
array	hhk	

3. Toelichting

Zie 5 : Blokschema.

Belangrijke lokale parameters zijn :

integer van
integer tot
} geven aan hoe de lijn die bij Makeline A wordt gemaakt, komt te lopen.
De lijn loopt van HCT[ahc, van] → Hct[ahc, tot]

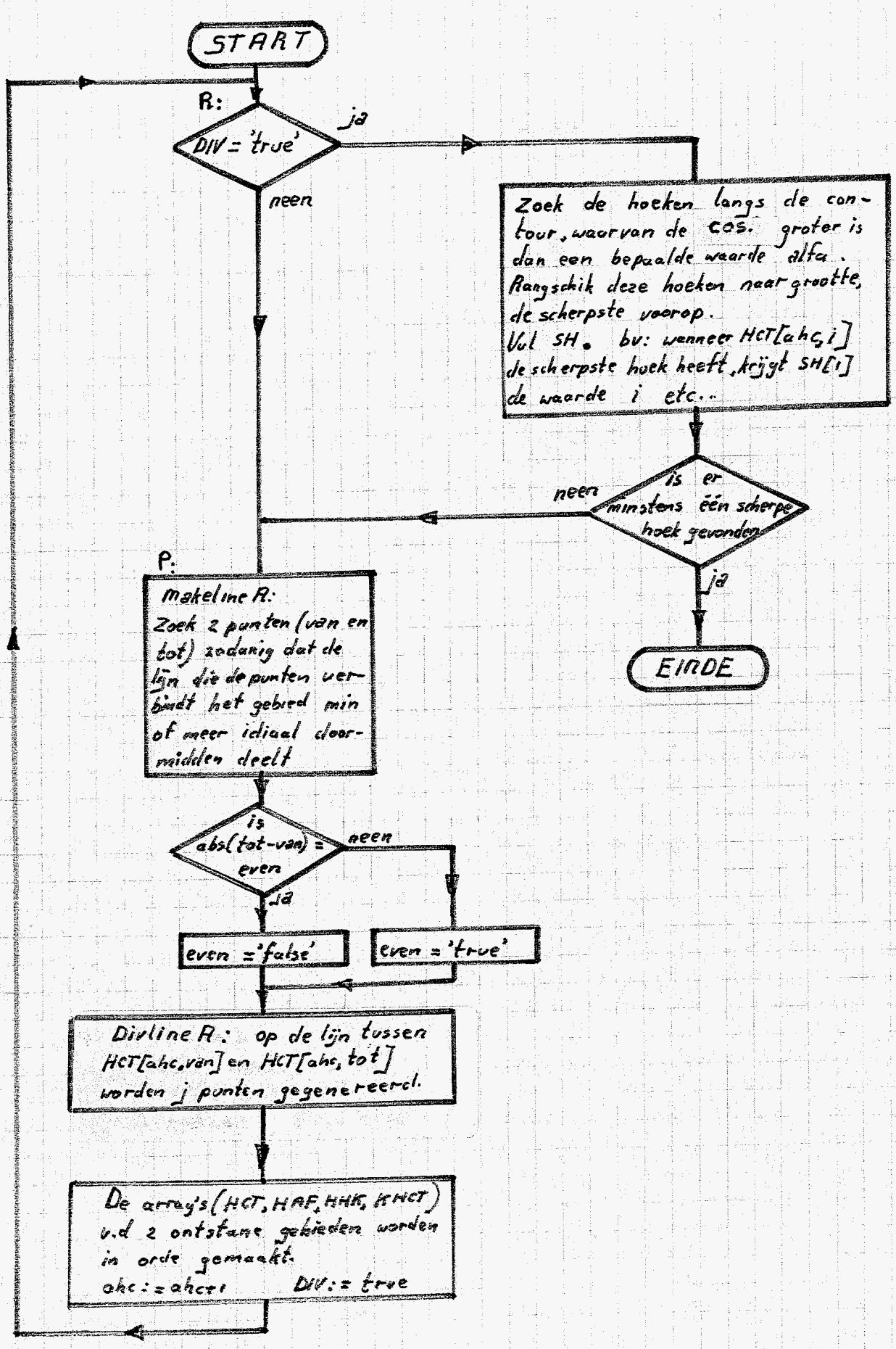
boolean even geeft aan of op de lijn een even of oneven aantal punten moet worden gegenereerd.
als even = true wordt er een even aantal punten gegenereerd.

4. Discutabel

De "scherpe" hoeken op de contour moeten kleiner zijn dan een bepaalde waarde alfa. In de gehanteerde testvoorbeelden is alfa ≈ 100° (cos(alfa) = -0.2)
Wanneer we alfa te scherp maken zal het gebied in meer subgebieden worden verdeeld → onregelmatigere meshverdeling en meer rekentijd.

5

Blokschema



6 Programmatekst

```
'PROCEDURE'DIVF(CO,KHCT,HCT,AHC,AF,SH,ASH,AK,DIV,S,CT,KCT,HAF,HHK);                DIFF   1
  'REAL'S;'INTEGER'AHC,ASH,AK,KCT;'BOOLEAN'DIV;                                    DIFF   2
  'ARRAY'CO[1,1],KHCT[1],HCT[1,0],AF[0],      CT[0],SH[1],HAF[1,0],              DIFF   3
         HHK[1,0];                                                                 DIFF   4
  'BEGIN''INTEGER'L,T,VAN,TOT,J;'LABEL'P,R;'ARRAY'HA[0:KHCT[AHC]],                 DIFF   5
                                          HOEK[1:5],  H[1:4];                      DIFF   5
       'REAL'A,B,C,D,E,F;'BOOLEAN'EVEN;                                           DIFF   6
R:     EVEN:='TRUE';                                                             DIFF   7
       'IF'DIV'THEN''BEGIN'T:=1;'FOR'L:=1'STEP'1'UNTIL'KHCT[AHC]'DO'             DIFF   8
        'BEGIN''IF'HHK[AHC,L]'GTR'=0.2'THEN''BEGIN'SH[T]:=L;ASH:=T;              DIFF   9
                  HOEK[T]:=HHK[AHC,L];      T:=T+1;'END';'END';                   DIFF  10
      'FOR'L:=1'STEP'1'UNTIL'ASH-1'DO''BEGIN'                                    DIVF  10
      'FOR'J:=L+1'STEP'1'UNTIL'ASH'DO''BEGIN'                                    DIVF  10
        'IF'HOEK[L]'LSS'HOEK[J]'THEN''BEGIN'                                     DIVF  10
          A:=SH[L];SH[L]:=SH[J];SH[J]:=A;                                        DIVF  10
          A:=HOEK[J];HOEK[J]:=HOEK[L];HOEK[L]:=A;'END';'END';'END';              DIVF  10
            'IF'T'EQL'1'THEN''GOTO'P;                                            DIFF  11
              'END''ELSE'                                                        DIFF  12
':     'BEGIN'MAKELINEA(KHCT[AHC],HCT[AHC,*],CO,VAN,TOT,HAF[AHC,*],              DIFF  13
                HHK[AHC,*]);                                                     DIFF  13
      'IF'(TOT-VAN)/2'EQL'ENTIER((TOT-VAN)/2)'THEN'EVEN:='FALSE';                DIFF  14
      DIVLINEA(KCT,CT,CO,AF,HCT[AHC,*],KHCT[AHC],AK,VAN,TOT,EVEN,                DIFF  15
                'TRUE',S,J);                                                     DIFF  16
      A:=CO[HCT[AHC,VAN],1];                                                     DIFF  17
      B:=CO[HCT[AHC,VAN],2];                                                     DIFF  18
      'FOR'L:=0'STEP'1'UNTIL'J'DO'                                               DIFF  19
      'BEGIN''IF'L'EQL'J'THEN''BEGIN'C:=CO[HCT[AHC,TOT],1];                     DIFF  20
                                      D:=CO[HCT[AHC,TOT],2];'END'                DIFF  21
                          'ELSE''BEGIN'C:=CO[AK-1-L,1];                          DIFF  22
                                      D:=CO[AK-1-L,2];'END';                     DIFF  23
        HA[L]:=SQRT((A-C)**2+(B-D)**2);A:=C;B:=D;'END';                          DIFF  24
    A:=CO[HCT[AHC,VAN-1],1];E:=CO[AK-J,1];                                       DIFF  25
    B:=CO[HCT[AHC,VAN-1],2];F:=CO[AK-J,2];                                       DIFF  26
    C:=CO[HCT[AHC,VAN  ],1];                                                     DIFF  27
    D:=CO[HCT[AHC,VAN  ],2];                                                     DIFF  28
    H[1]:=HKCOS(A,B,C,D,E,F);A:=E;B:=F;                                          DIFF  29
    E:=CO[HCT[AHC,VAN+1],1];                                                     DIFF  30
    F:=CO[HCT[AHC,VAN+1],2];                                                     DIFF  31
    H[2]:=HKCOS(A,B,C,D,E,F);                                                    DIFF  32
    A:=CO[HCT[AHC,TOT-1],1];E:=CO[AK-1,1];                                       DIFF  33
    B:=CO[HCT[AHC,TOT-1],2];F:=CO[AK-1,2];                                       DIFF  34
    C:=CO[HCT[AHC,TOT  ],1];                                                     DIFF  35
    D:=CO[HCT[AHC,TOT  ],2];                                                     DIFF  36
    H[3]:=HKCOS(A,B,C,D,E,F);A:=E;B:=F;                                          DIFF  37
    E:=CO[HCT[AHC,TOT+1],1];                                                     DIFF  38
    F:=CO[HCT[AHC,TOT+1],2];                                                     DIFF  39
    H[4]:=HKCOS(A,B,C,D,E,F);                                                    DIFF  40
    'FOR'L:=0'STEP'1'UNTIL'VAN-1'DO'                                             DIFF  41
    'BEGIN'HCT[AHC+1,L]:=HCT[AHC,L];HAF[AHC+1,L]:=HAF[AHC,L];                    DIFF  42
```

```
            HHK[AHC+1,L]:=HHK[AHC,L];'END';                                    DIFF 430
    HAF[AHC+1,VAN]:=HA[0];HHK[AHC+1,VAN]:=H[1];                                DIFF 440
    HCT[AHC+1,VAN]:=HCT[AHC,VAN];                                              DIFF 450
    'FOR'L:=1'STEP'1'UNTIL'J'DO'                                               DIFF 460
    'BEGIN'HCT[AHC+1,VAN+L]:=AK=     L;HAF[AHC+1,VAN+L]:=HA[L];                DIFF 470
       HHK[AHC+1,VAN+L]:=-1;'END';                                             DIFF 480
    HCT[AHC+1,VAN+J+1]:=HCT[AHC,TOT];                                          DIFF 490
    HAF[AHC+1,VAN+J+1]:=HAF[AHC,TOT];HHK[AHC+1,VAN+J+1]:=H[4];                 DIFF 500
    'FOR'L:=1'STEP'1'UNTIL'KHCT[AHC]-TOT'DO'                                   DIFF 510
    'BEGIN'HCT[AHC+1,VAN+J+1+L]:=HCT[AHC,TOT+L];                               DIFF 520
            HAF[AHC+1,VAN+J+1+L]:=HAF[AHC,TOT+L];                              DIFF 530
            HHK[AHC+1,VAN+J+1+L]:=HHK[AHC,TOT+L];'END';                        DIFF 540
    'FOR'L:=1'STEP'1'UNTIL'TOT-VAN'DO'                                         DIFF 550
    'BEGIN'HCT[AHC,L]:=HCT[AHC,VAN-1+L];                                       DIFF 560
            HAF[AHC,L]:=HAF[AHC,VAN-1+L];                                      DIFF 570
            HHK[AHC,L]:=HHK[AHC,VAN-1+L];'END';                                DIFF 580
    HHK[AHC,1]:=H[2];HCT[AHC,TOT-VAN+1]:=HCT[AHC,TOT];                         DIFF 590
    HHK[AHC,TOT-VAN+1]:=H[3];HAF[AHC,TOT-VAN+1]:=HA[J];                        DIFF 600
    'FOR'L:=1'STEP'1'UNTIL'J'DO'                                               DIFF 610
    'BEGIN'HCT[AHC,TOT-VAN+1+L]:=AK-J-1+L;                                     DIFF 620
            HAF[AHC,TOT-VAN+1+L]:=HA[J-L];                                     DIFF 630
            HHK[AHC,TOT-VAN+1+L]:=-1;'END';                                    DIFF 640
    HAF[AHC,0]:=HA[0];                                                         DIVF 650
    KHCT[AHC]:=VAN+J+KHCT[AHC]-TOT+1;                                          DIFF 660
    KHCT[AHC+1]:=TOT-VAN+1+J;                                                  DIFF 670
    HCT[AHC,0]:=HCT[AHC,KHCT[AHC]];                                            DIVF670A
    HHK[AHC,0]:=HHK[AHC,KHCT[AHC]];                                            DIVF670B
    HHK[AHC+1,0]:=HHK[AHC+1,KHCT[AHC+1]];                                      DIVF670B
    HCT[AHC,KHCT[AHC]+1]:=HCT[AHC,1];                                          DIVF 671
    HHK[AHC,KHCT[AHC]+1]:=HHK[AHC,1];                                          DIVF 672
    HAF[AHC,KHCT[AHC]+1]:=HAF[AHC,1];                                          DIVF 673
    HCT[AHC+1,KHCT[AHC+1]+1]:=HCT[AHC+1,1];                                    DIFF 680
    HHK[AHC+1,KHCT[AHC+1]+1]:=HHK[AHC+1,1];                                    DIFF 690
    HAF[AHC+1,KHCT[AHC+1]+1]:=HAF[AHC+1,1];                                    DIFF 700
    AHC:=AHC+1;DIV:='TRUE';'GOTO'R;                                            DIFF 710
  'END';                                                                       DIFF 720
'END'DIFF;                                                                     DIFF 720
```

Procedure renumct

1. Inleiding

Een gebied wordt zodanig hernummerd dat ct[i]. het knooppuntsnummer v.d "scherpe" hoek SH[t] bevat, van waaruit we het gebied in stroken gaan verdelen.

2. Gebruiksaanwijzing

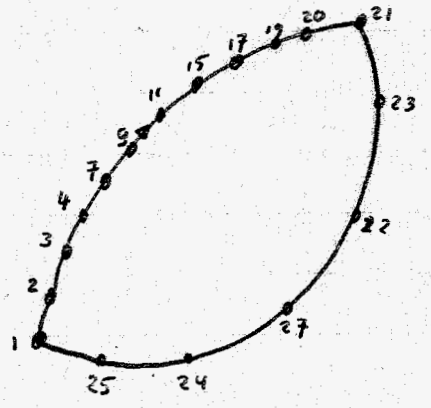
Renumct (ct, kct, SH, HK, AF, ASH, t)

2.1 Formele parameters

array	ct	arrays met de grenzen [0:kct+1], bevatten de
array	hk	knooppuntsnummers, cosinus v.d hoeken en afstanden v.d con-
array	af	tour die hernummerd moet worden.
integer	kct	aantal knooppunten op de contour
array	SH	zie Divconvexvierhoek 5.2
integer	ASH	
integer	t	geeft aan met de hoeveelste "scherpe" hoek v.d. contour we bezig zijn

3 Toelichting

voorbeeld:



Stel: CT[1] = 11. en SH[1] = 6 (dus het 6e knooppunt op de contour, dit is 21, heeft een "scherpe" hoek)
We moeten nu de array's ct, hk en af 5 plaatsen opschuiven om de gewenste situatie te bereiken
dus ct[1] := ct[1+5] = 21 etc.....
Natuurlijk moet ook het array SH indien ash > t aangepast worden. b.v. Stel SH[2] = 12 (dus het 12e knooppunt op de contour, dit is 1, heeft een "scherpe" hoek). Dan krijgen we:
SH[2] := SH[2] - 5 = 7.

4 Programmatekst.

```
'PROCEDURE'RENUMCT(CT,KCT,SH,HK,AF,ASH,T);                                    RNCT   1
  'INTEGER'KCT,ASH,T;'ARRAY'CT[0],SH[1],HK[0],AF[0];                          RNCT   2
  'BEGIN''ARRAY'A[0:KCT],B[0:KCT],C[0:KCT];'INTEGER'J,STAP;                   RNCT   3
    STAP:=SH[T]-1;'FOR'J:=0'STEP'1'UNTIL'STAP'DO'                             RNCT   4
      'BEGIN'A[J]:=CT[J];B[J]:=HK[J];C[J]:=AF[J];'END';                       RNCT   5
    'FOR'J:=0'STEP'1'UNTIL'KCT      -STAP'DO'                                 RNCT   6
      'BEGIN'CT[J]:=CT[J+STAP];HK[J]:=HK[J+STAP];AF[J]:=AF[J+STAP];           RNCT   7
      'END';                                                                  RNCT   8
    'FOR'J:=0'STEP'1'UNTIL'STAP'DO''BEGIN'CT[KCT-STAP+J]:=A[J];               RNCT   9
          HK[KCT-STAP+J]:=B[J];AF[KCT-STAP+J]:=C[J];'END';                    RNCT  10
      'FOR'J:=T'STEP'1'UNTIL'ASH'DO''BEGIN'SH[J]:=(SH[J]-STAP+KCT)           RNCT  10
          'MOD'KCT;'IF'SH[J]'EQL'0'THEN'SH[J]:=KCT;'END';                     RNCT  11
    CT[KCT+1]:=CT[1];HK[KCT+1]:=HK[1];AF[KCT+1]:=AF[1];                       RNCT  12
  'END'RENUMCT;                                                               RNCT  13
```

Procedure Divline A

1 Inleiding

De procedure legt op een lijn tussen de knooppunten HCT[van] en HCT[tot] een aantal knooppunten rekeninghoudende met de knooppuntsverdeling op de oorspronkelijke contour (ct).

2 Gebruiksaanwijzing

Divline A(kct, ct, co, af, hct, khct, ak, van, tot, even, vrk, s, j).

2.1 Globale parameters

procedure BPG

2.2 Formele parameters

integer	kct	Zie Divconvexvierhoek 2.2
array	ct	
array	co	
array	af	
integer	ak	
array	hct	Zie Divconvexvierhoek 5.2
array	khct	
real	s	
integer	van	Zie DIVF 3
integer	tot	
boolean	even	
boolean	vrk.	krijgt de waarde true indien we een vierhoekige elementverdeling wensen.
integer	j	na afloop v.d procedure bevat j het aantal gegenereerde knooppunten.

3 Werking v.d procedure

De getallen in de beschrijving verwijzen naar het Blokschema.
We gaan als volgt te werk:

① Bepaal de grofheid in HCT[van] en HCT[tot]. (fig 1)
We beginnen op de plaats waar de grofheid het kleinst is: ga (gmin), dit punt heeft de coördinaten: (A,B). Het ander punt heeft coördinaten (C,D).

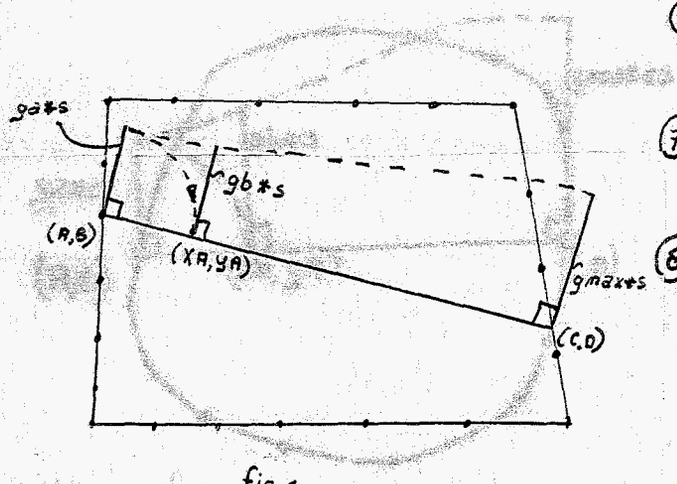


fig 1

⑤ Vanaf (A,B) passen we een lijnstuk af met lengte: ga*s → (XA, YA)

⑦ Na controle v.d ligging van dit punt (XA, YA), bepalen we de grofheid in (XA, YA): gb.

⑧ Vanaf (A,B) passen we een lijnstuk af met lengte: $\frac{(ga+gb)}{2} \times s \rightarrow (X, Y)$

Na controle van de ligging van punt (X,Y), kunnen we in (X,Y) de grofheid bepalen en met dit gegeven een nieuw lijnstuk vanuit (A,B) afzetten. We kunnen dit proces een aantal malen herhalen, totdat bv. de afstand tussen (X,Y) en voorlaatste punt (X,Y) kleiner is dan een bepaalde waarde. In de procedure herhalen we deze slag maximaal 4 maal omdat anders te veel rekentijd nodig is.
Het laatste punt (X,Y) slaan we op in (HCO[1,1], HCO[1,2]).
Vanuit (HCO[1,1], HCO[1,2]) gaan we het gehele proces herhalen: (XA, YA) afzetten, daarna (X,Y) etc.... We vinden dan: (HCO[2,1], HCO[2,2]).
Natuurlijk moeten we steeds de ligging v.d gevonden punten controleren.
We doen dit door te controleren of ze niet te dicht bij (C,D) liggen of buiten de lijn (A,B - C,B) vallen. ⑨
Als op deze manier tenslotte de lijn verdeeld is hebben we j punten gevonden.
We hebben dan een situatie zoals aangegeven in fig 2.

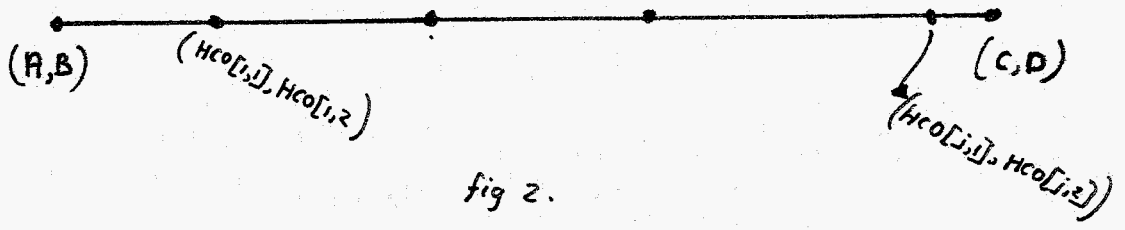


fig 2.

Als we vierhoekige elementen willen genereren zullen aan het aantal knooppunten op de lijn eisen gesteld moeten worden, b.v: even of oneven. Wanneer nu het aantal knooppunten op de lijn niet klopt, wordt het laatst gegenereerde punt weggehaald.
Indien er o punten gegenereerd zijn en we wensen een oneven aantal knooppunten (22), dan moet er een extra punt gegenereerd worden.
Het zal duidelijk zijn dat meestal de afstand tussen het laatst gegenereerde punt en (C,D) niet voldoet.
Door nu vanuit C,D in 2 stappen een lijnstuk te genereren (17) kunnen we een redelijke benadering v.d. elementsribbe grenzend aan (C,D) verkrijgen (fig 3)

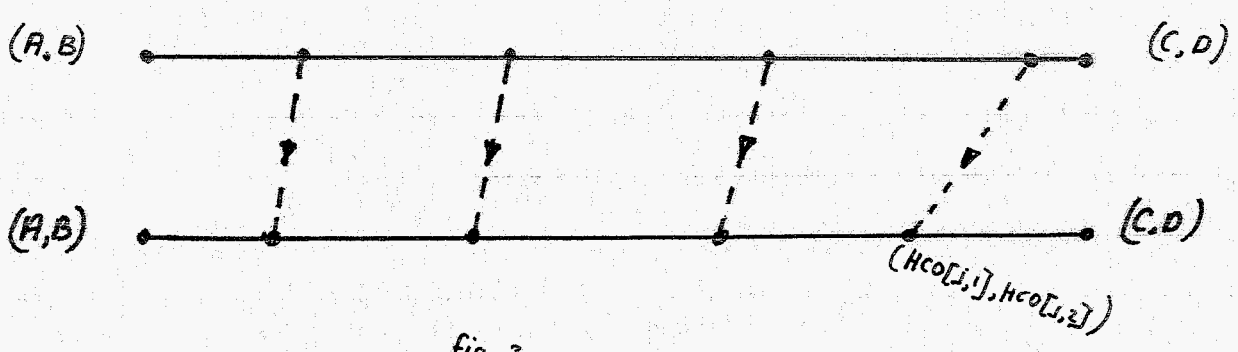


fig. 3.

De overige j-1 knooppunten kunnen nu gevonden worden door het lijnstuk (A,B − HCO[j,1], HCO[j,2]) in stukken te verdelen die zich verhouden als de afstanden v.d. oorspronkelijke punten.

opm. Bij de bepaling v.d. x en y- coördinaten wordt gebruik gemaakt van cos a en sin a. $\cos a = \frac{(C-A)_{x}}{\text{afstand}\{(A,B)-(C,D)\}}$ s.

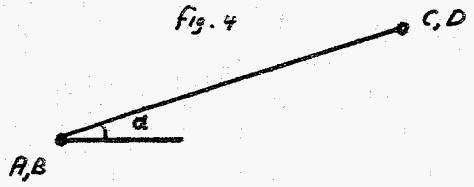
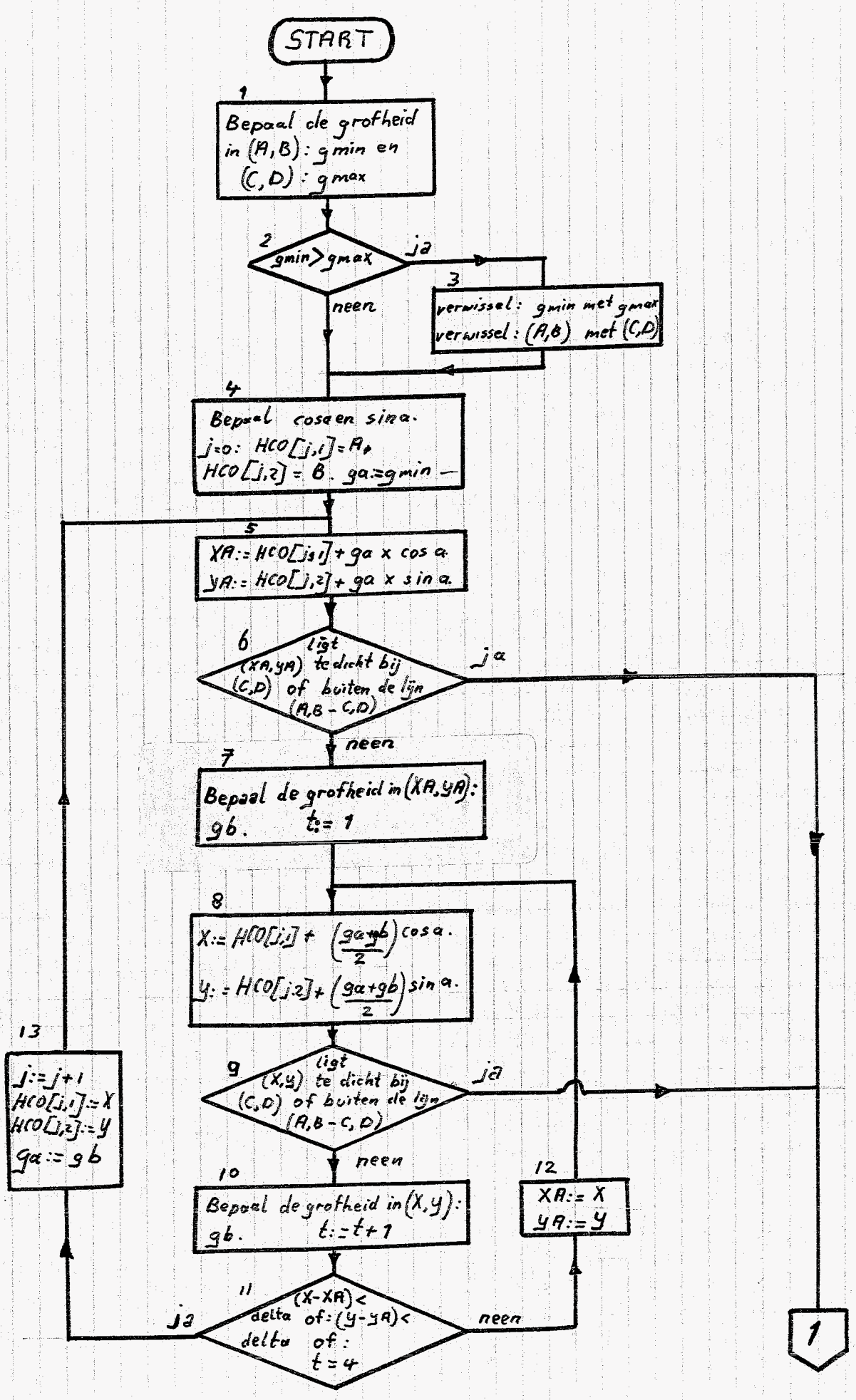
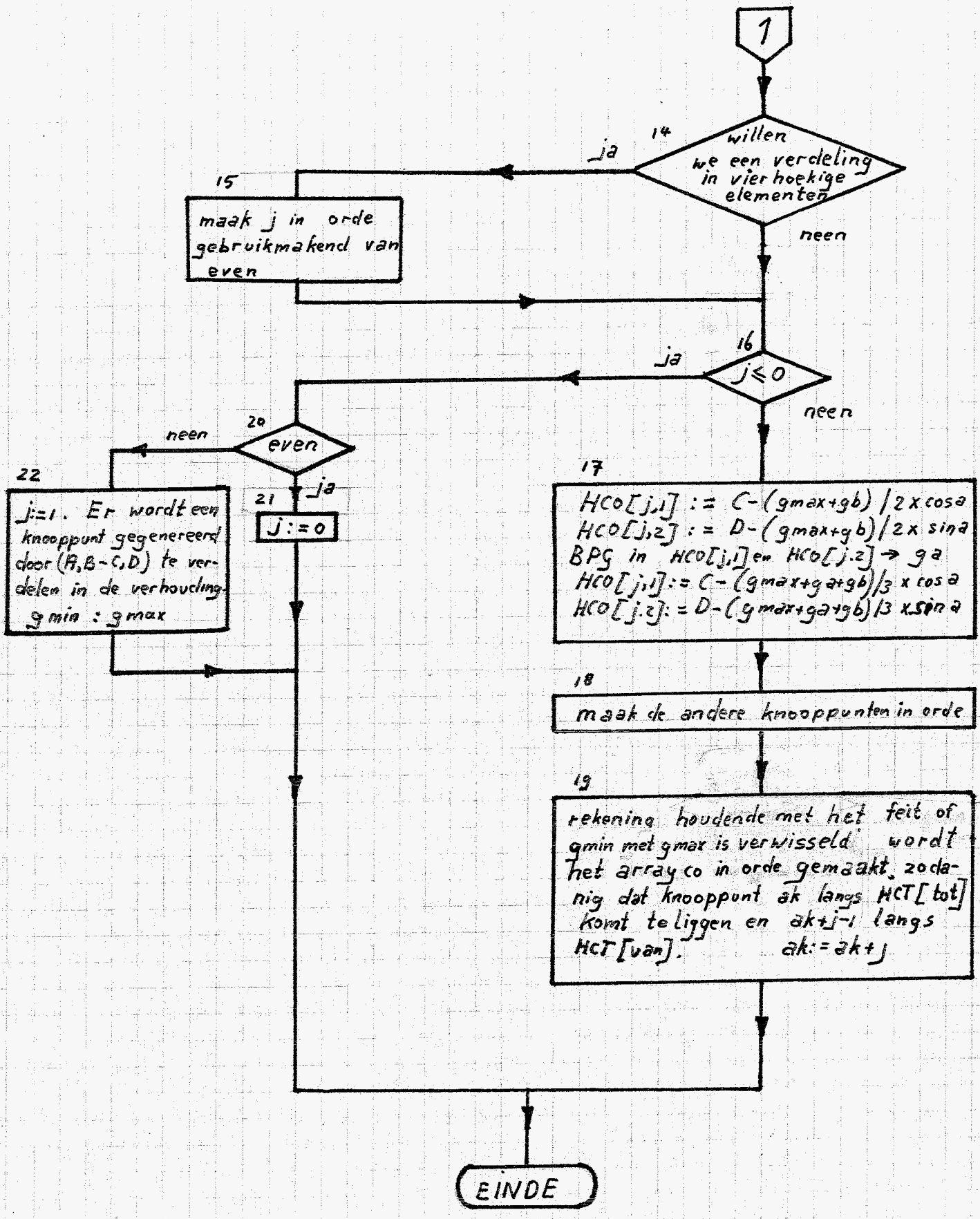


fig. 4

4 Blokschema




1
14
willen we een verdeling in vierhoekige elementen
ja
neen
15
maak j in orde gebruikmakend van even
16
j≤0
ja
neen
20
even
neen
ja
22
j:=1. Er wordt een knooppunt gegenereerd door (A,B-C,D) te verdelen in de verhouding gmin : gmax
21
j:=0
17
HCO[j,1] := C-(gmax+gb)/2 x cosa
HCO[j,2] := D-(gmax+gb)/2 x sina
BPS in HCO[j,1] en HCO[j,2] → ga
HCO[j,1] := C-(gmax+ga+gb)/3 x cosa
HCO[j,2] := D-(gmax+ga+gb)/3 x sina
18
maak de andere knooppunten in orde
19
rekening houdende met het feit of gmin met gmax is verwisseld wordt het array co in orde gemaakt, zodanig dat knooppunt ak langs HCT[tot] komt te liggen en ak+j-1 langs HCT[van]. ak:=ak+j
EINDE

5 Discotabel

Het iteratief proces bij het bepalen v.d. knooppunten kost, omdat bij elke stap de grofheid moet worden bepaald veel rekentijd.
Sterk te overwegen valt daarom deze stappen weg te laten.
Voor elk knooppunt wordt dan slechts éénmaal de grofheid bepaald.
De onnauwkeurigheid die we hierdoor krijgen zal, omdat de knooppunten toch nog herplaatst moeten worden, erg meevallen.

```
PROCEDURE'DIVLINEA(KCT,CT,CO,AF,HCT,KHCT,AK,VAN,TOT,EVEN,VRK,S,J);                DLA   10
  'INTEGER'KCT,AK,VAN,TOT,J,KHCT;'REAL'S;'BOOLEAN'VRK,EVEN;                       DLA   20
  'ARRAY'CT[0],CO[1,1],AF[0],HCT[0];                                              DLA   30
  'BEGIN''REAL'A,B,C,D,GMAX,GMIN,GA,GB,AFST,DELTA,ETHA,COSA,SINA,XA,YA,           DLA   40
            Y,X,K,L;                                                              DLA   50
      'INTEGER'T;'LABEL'V,W,R,Q;'ARRAY'HCO[0:KHCT,1:2];                           DLA   60
    A:=CO[HCT[VAN],1];B:=CO[HCT[VAN],2];C:=CO[HCT[TOT],1];                        DLA   70
    D:=CO[HCT[TOT],2];                                                            DLA   80
    BPG(CO,CT,A,B,KCT,GMIN,S,AF);                                                 DLA   90
    BPG(CO,CT,C,D,KCT,GMAX,S,AF);L:=0;                                            DLA  100
    'IF'GMIN'GTR'GMAX'THEN''BEGIN'L:=GMAX;GMAX:=GMIN;GMIN:=L;K:=A;                DLA  110
                                  A:=C;C:=K;K:=B;B:=D;D:=K;'END';                 DLA  120
    AFST:=SQRT((A-C)**2+(B-D)**2);DELTA:=(0.1*S*GMIN)**2;ETHA:=0.2*S;             DLA  130
    COSA:=(C-A)*S/AFST;SINA:=(D-B)*S/AFST;                                        DLA  140
    J:=0;HCO[J,1]:=A;HCO[J,2]:=B;GA:=GMIN;                                        DLA  150
V:  XA:=HCO[J,1]+GA*COSA;YA:=HCO[J,2]+GA*SINA;                                    DLA  160
    'IF'(ABS(XA-C)'LSS'ETHA'AND'ABS(YA-D)'LSS'ETHA)'OR'                           DLA  170
      (SIGN(C-XA)'EQL'SIGN(A-XA)'AND'SIGN(D-YA)'EQL'SIGN(B-YA))                   DLA  180
      'THEN''GOTO'R;                                                            DLA  190
    T:=1;BPG(CO,CT,XA,YA,KCT,GB,S,AF);                                            DLA  200
N:  X:=HCO[J,1]+(GA+GB)/2*COSA;Y:=HCO[J,2]+(GA+GB)/2*SINA;                        DLA  210
    'IF'(ABS(X-C)'LSS'ETHA'AND'ABS(Y-D)'LSS'ETHA)'OR'                             DLA  220
      (SIGN(C-X)'EQL'SIGN(A-X)'AND'SIGN(D-Y)'EQL'SIGN(B-Y))                       DLA  230
      'THEN''GOTO'R;                                                            DLA  240
    T:=T+1;BPG(CO,CT,X,Y,KCT,GB,S,AF);                                            DLA  250
    'IF'((X-XA)**2+(Y-YA)**2)'LSS'DELTA 'OR'T'EQL'4'THEN'                         DLA  260
    'BEGIN'J:=J+1;HCO[J,1]:=X;HCO[J,2]:=Y;GA:=GB;'GOTO'V'END'                     DLA  270
    'ELSE''BEGIN'XA:=X;YA:=Y;'GOTO'N;'END';                                       DLA  280
R:  'IF'VRK'THEN'                                                                 DLA  290
    'BEGIN''IF'EVEN'THEN''BEGIN''IF'ENTIER(J/2)'NEQ'J/2'THEN'                     DLA  300
                                  J:=J-1'END'                                     DLA  310
                'ELSE''BEGIN''IF'ENTIER(J/2)'EQL'J/2'THEN'                        DLA  320
                                  J:=J-1'END';                                    DLA  330
    'END';                                                                        DLA  340
    'IF'J'LEQ'1'THEN'                                                             DLA  350
    'BEGIN''IF'EVEN'THEN'J:=0                                                     DLA  360
          'ELSE''BEGIN'CO[AK,1]:=A+GMIN/(GMIN+GMAX)*(C-A);                        DLA  370
                      CO[AK,2]:=B+GMIN/(GMIN+GMAX)*(D-B);                         DLA  380
                      J:=1;AK:=AK+1;                                              DLA  390
               'END';'GOTO'Q;                                                     DLA  400
    'END';                                                                        DLA  410
    X:=HCO[J,1];Y:=HCO[J,2];                                                      DLA  420
    HCO[J,1] :=C-(GMAX+GB)/2*COSA;                                                DLA  440
    HCO[J,2] :=D-(GMAX+GB)/2*SINA;                                                DLA  450
    BPG(CO,CT,HCO[J,1],HCO[J,2],KCT,GA,S,AF);                                     DLA  451
    HCO[J,1]:=C-(GMAX+GB+GA)/3*COSA;                                              DLA  452
    HCO[J,2]:=D-(GMAX+GB+GA)/3*SINA;                                              DLA  452
  'IF'ABS(COSA)'GTR'ABS(SINA)'THEN'K:=(X-A)/(HCO[J,1]-A)
    'ELSE'K:=(Y-B)/(HCO[J,2]-B);                                                  DLA  490
    'FOR'T:=J-1'STEP'-1'UNTIL'1'DO'                                               DLA  500
    'BEGIN'XA:=HCO[T,1];YA:=HCO[T,2];                                             DLA  510
      HCO[T,1]:=HCO[T+1,1]-(X-XA)/K;                                              DLA  520
      HCO[T,2]:=HCO[T+1,2]-(Y-YA)/K;X:=XA;Y:=YA;                                  DLA  530
    'END';                                                                        DLA  540
  'IF'L'EQL'0'THEN'
    'BEGIN''FOR'T:=J'STEP'-1'UNTIL'1'DO'                                         DLA  560
          'BEGIN'CO[AK+J-T,1]:=HCO[T,1];CO[AK+J-T,2]:=HCO[T,2];                   DLA  570
          'END';                                                                  DLA  580
    'END''ELSE'                                                                   DLA  590
    'BEGIN''FOR'T:=1'STEP'1'UNTIL'J'DO'                                           DLA  600
          'BEGIN'CO[AK+T-1,1]:=HCO[T,1];CO[AK+T-1,2]:=HCO[T,2];                   DLA  610
          'END';                                                                  DLA  620
    'END';                                                                        DLA  630
    AK:=AK+J;                                                                     DLA  640
Q:'END'DIVLINEA;                                                                  DLA  650
```

Procedure Hkcos

1 Inleiding

De procedure bepaalt de cosinus van een hoek alfa.
A, B, C, D, E, F zijn knooppuntscoördinaten v.d. 3 bij de hoek behorende knooppunten.

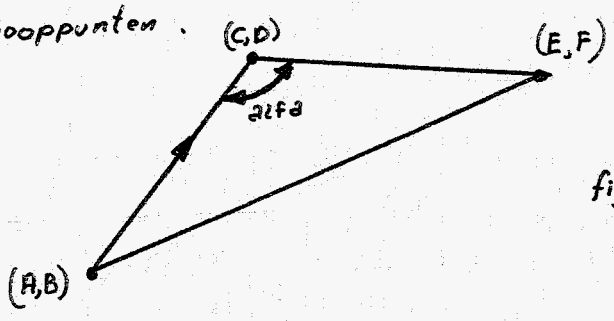


fig. 1

De bepaling van de cosinus gebeurt m.b.v. de cos-regel.

2 Gebruiksaanwijzing

real procedure Hkcos (A, B, C, D, E, F)

2.1 Formele parameters

real	A
real	B
real	C
real	D
real	E
real	F

Zie fig 1

3 Programmatekst

```
'REAL''PROCEDURE'HKCOS(A,B,C,D,E,F);                    HKCOS 1
    'REAL'A,B,C,D,E,F;                                  HKCOS 2
    'BEGIN''REAL'K,M,KM;                                HKCOS
      K:=(A-C)**2+(D-B)**2;                             HKCOS
      M:=(E-C)**2+(F-D)**2;                             HKCOS
      KM:=2*SQRT(K)*SQRT(M);                            HKCOS
      HKCOS:=(K+M-(E-A)**2-(F-B)**2)/KM;                HKCOS
'END'HKCOS;
```

Procedure Contreline

1 Inleiding

De procedure controleert of de cosinus v.d hoeken die de lijn j maakt met de contour niet groter is dan een bepaalde waarde alfa. Indien de lijn niet voldoet krijgt contreline de waarde true

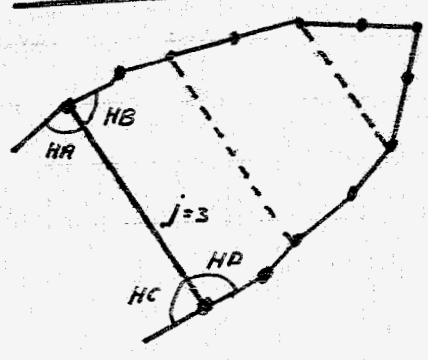
2 Gebruiksaanwijzing

Boolean procedure Contreline(co, kct, ct, af, alfa, j)

2.1 Formele parameters

array	co	Zie Divconvexvierhoek 2.2
integer	kct	aantal knooppunten op de contour van het gebied, waarin we de lijn controleren.
array	ct	array met de grenzen [0:kct+1] bevat de knooppuntsnummers v.d contour.
array	af	array met de grenzen [0:kct+1] bevat de afstanden tussen de knooppunten v.d. contour.
real	alfa	max. toelaatbare cosinus van de hoek die de lijn met de contour maakt.
integer	j	geeft aan welke lijn gecontroleerd moet worden. j-1 is het aantal reeds afgesneden stroken.

3. Toelichting



De procedure berekent m.b.v. de cos-regel de 4 cosinussen: HA, HB, HC, HD.
Indien een of meer der cosinussen groter is dan alfa, wordt de lijn niet geaccepteerd.

4. Programmatekst

```
'BOOLEAN''PROCEDURE'CONTRLINE(CO,KCT,CT,AF,ALFA,J);                    CLINE 10
  'REAL'ALFA;'INTEGER'KCT,J;'ARRAY'CO[1,1],CT[0],AF[0];                CLINE 20
  'BEGIN''REAL'A,B,C,D,E,F,G,H,I,K,L,M,P,Q,R,S,T,U,V,W,X,WU,WV,WW,WX,  CLINE 30
            WP,HA,HB,HC,HD;                                            CLINE 40
    A:=CO[CT[2*J],1];C:=CO[CT[2*J+1],1];E:=CO[CT[2*J+2],1];            CLINE 50
    B:=CO[CT[2*J],2];D:=CO[CT[2*J+1],2];F:=CO[CT[2*J+2],2];            CLINE 60
    G:=CO[CT[KCT+2-2*J],1];I:=CO[CT[KCT+1-2*J],1];L:=CO[CT[KCT-2*J],1];CLINE 70
    H:=CO[CT[KCT+2-2*J],2];K:=CO[CT[KCT+1-2*J],2];M:=CO[CT[KCT-2*J],2];CLINE 80
    P:=(C-I)**2+(K-D)**2;Q:=(C-G)**2+(H-D)**2;R:=(A-I)**2+(B-K)**2;     CLINE 90
    S:=(C-L)**2+(M-D)**2;T:=(E-I)**2+(F-K)**2;                         CLINE100
    WU:=AF[   2*J ];WV:=AF[   2*J+1 ];                                 CLINE110
    WW:=AF[   KCT-2*J+1 ];WX:=AF[   KCT-2*J ];                         CLINE120
    WP:=SQRT(P);U:=WU**2;V:=WV**2;W:=WW**2;X:=WX**2;                   CLINE130
    HA:=(P+U-R)/(2*WU*WP);                                             CLINE140
    HB:=(P+V-T)/(2*WV*WP);                                             CLINE150
    HC:=(P+X-S)/(2*WX*WP);                                             CLINE160
    HD:=(P+W-Q)/(2*WW*WP);                                             CLINE170
    'IF'(HA'GTR'ALFA)'OR'(HB'GTR'ALFA)'OR'(HC'GTR'ALFA)'OR'            CLINE180
       (HD'GTR'ALFA)                                                   CLINE181
    'THEN'CONTRLINE:='TRUE ''ELSE'CONTRLINE:='FALSE';                  CLINE190
  'END'CONTRLINE;                                                      CLINE200
```

Procedure Makeline A

1. Inleiding

De procedure zoekt een lijn van HCT[van] naar HCT[tot], zodanig dat het gebied min of meer ideaal in tweeën wordt gedeeld.

2. Gebruiksaanwijzing

Procedure MakelineA(kct, ct, co, van, tot, af, hk)

2.1 Formele parameters

integer kct — aantal knooppunten op de contour v/h gebied waarin we een lijn willen trekken.

array ct, array af, array hk — array's met de grenzen [0:kct+1], bevatten resp. de knooppuntsnummers, afstanden en cosinus v.d. hoeken v.d. knooppunten op de contour.

array co — zie Div convexvierhoek 2.2

integer van, integer tot — zie DIVF 3

3 Toelichting

We verdelen het gebied zodanig dat we 2 gebieden krijgen met elk het zelfde aantal knooppunten. In totaal kunnen er dan $\frac{kct}{2}$ lijnen worden getrokken.

Van elke mogelijke lijn bepalen we de cosinus v.d hoeken met de contour: HA, HB, HC en HD en controleren tevens de hoeken : A, B, C, D .. (fig 1)

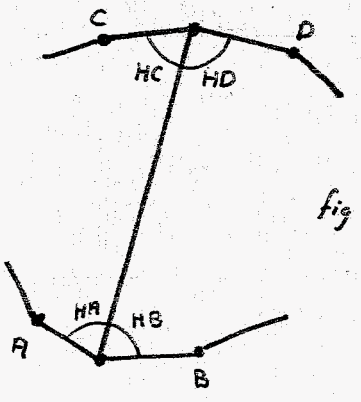


fig 1

Wanneer deze hoeken A, B, C, D te scherp zijn kunnen we de lijn beter niet accepteren, we voorkomen dan een situatie zoals aangegeven in fig. 2. Het "smalste" gebied zou dan ongunstige elementen opleveren bij verdere verdeling.

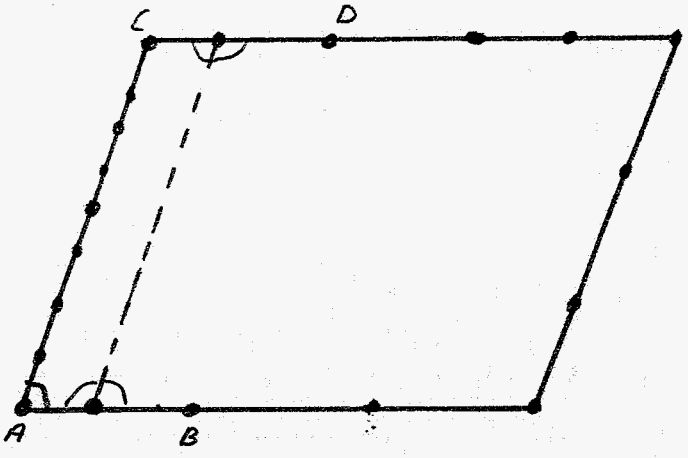


fig 2.

De meest acceptabele lijn voor ons is diegene die met de contour zoveel mogelijk, hoeken van 60° of 120° maakt en waarvan de hoeken A, B, C en D stomper zijn dan een bepaalde waarde. In de procedure hebben we geeist dat de cosinus v.d hoeken kleiner moet zijn dan −0,5.
Indien geen lijn kan worden gevonden waarvan de cosinussen v.d hoeken A, B, C D kleiner zijn dan −0.5 kijken we alleen naar de eerste voorwaarde.

4 Discutabel

We hadden ook kunnen eisen dat de lijn met de contour zoveel mogelijk hoeken van 90° moet maken. De verwachting is dan dat de elementen een iets minder ideale vorm krijgen.
We hebben geeist dat beide gebieden evenveel knooppunten krijgen. Laten we deze eis vallen dan stijgt het aantal mogelijke lijnen natuurlijk sterk.
Er zullen dan best een aantal lijnen zijn die het gebied idealer verdelen.
De procedure zal daardoor echter veel te gecompliceerd worden en veel te veel rekentijd vergen. Bovendien is niet goed duidelijk is wat we onder ideaal moeten verstaan.

5. Programmatekst

```
'PROCEDURE'MAKELINEA(KCT,CT,CO,VAN,TOT,AF,HK);                                  MKLA  10
        'INTEGER'KCT,VAN,TOT;                                                   MKLA  20
        'ARRAY'CT[0],CO[1,1],AF[0],HK[0];                                       MKLA  30
  'BEGIN''REAL'A,B,C,D,E,F,G,H,I,K,L,M,AA,AB,AC,AD,AE,AL,AG,AH,AI,HMIN,MKLA  40
            HM,HA;'INTEGER'J,T,V;                                               MKLA  50
    T:=KCT/2;HMIN:=100;HM:=100;VAN:=0;                                          MKLA  60
    E:=CO[CT[0],1];F:=CO[CT[0],2];A:=CO[CT[1],1];B:=CO[CT[1],2];                MKLA  70
    I:=  CO[CT[T],1];K:=CO[CT[T],2];G:=CO[CT[T+1],1];H:=CO[CT[T+1],2];MKLA  80
    AL:=AF[0]**2;AH:=AF[T]**2;                                                  MKLA  90
    'FOR'J:=1'STEP'1'UNTIL'T'DO'                                                MKLA 100
    'BEGIN'C:=CO[CT[J+1],1];D:=CO[CT[J+1],2];                                   MKLA 110
          L:=CO[CT[T+J+1],1];M:=CO[CT[T+J+1],2];                                MKLA 120
          AG:=AF[J];AI:=AF[T+J];                                                MKLA 130
          AA:=(G-A)**2+(H-B)**2;AB:=(G-E)**2+(H-F)**2;                          MKLA 140
          AC:=(G-C)**2+(H-D)**2;AD:=(L-A)**2+(M-B)**2;                          MKLA 150
          AE:=(I-A)**2+(K-B)**2;                                                MKLA 160
          HA:=ABS((AA+AL-AB)**2/(AL*AA)-1)+                                     MKLA 170
             ABS((AA+AG-AC)**2/(AG*AA)-1)+                                      MKLA 180
             ABS((AA+AH-AE)**2/(AH*AA)-1)+                                      MKLA 190
             ABS((AA+AI-AD)**2/(AI*AA)-1);                                      MKLA 200
          'IF'HA'LSS'HMIN'AND'HK[J-1]'LSS'-0.5'AND'HK[J+1]'LSS'-0.5  MKLA 210
             'AND'HK[T+J-1]'LSS'-0.5'AND'HK[T+J+1]'LSS'-0.5'THEN'       MKLA 211
                          'BEGIN'HMIN:=HA;VAN:=J;TOT:=T+J;'END';        MKLA 212
                     E:=A;F:=B;A:=C;B:=D;I:=G;K:=H;G:=L;H:=M;           MKLA 220
                     AL:=AG;AH:=AI;                                             MKLA 230
          'IF'HA'LSS'HM'THEN''BEGIN'V:=J;HM:=HA'END';                           MKLA 231
    'END';                                                                      MKLA 240
    'IF'VAN'EQL'0'THEN''BEGIN'VAN:=V;TOT:=T+V'END';                             MKLA 241
  'END'MAKELINE A;                                                              MKLA 250
```

Procedure Bpg

1 Inleiding

Op basis v.d integraal formule van Gauchy uit de complexe functie theori, bepaalt deze procedure de grofheid in een punt (X,Y), als de knooppunten-verdeling op de contour gegeven is. De procedure is ook van toepassing op een concaaf gebied, ze is dan echter veel minder nauwkeurig.

2 Gebruiksaanwijzing.

BPG(knooppunt, contour, x, y, knpc, g, s, p);

2.1 Formele parameters

array	knooppunt	array met de grenzen (1:n, 1:2) bevat o.a. de coördinaten v.d knooppunten op de contour.
array	contour	array met de grenzen (0:knpc+1), bevat de knooppunts-nummers v.d. knooppunten op de contour.
integer	knpc	geeft het aantal knooppunten op de contour aan.
real	x	de x-coördinaat v.h. punt waarin we de grofheid willen bepalen
real	y	de y-coördinaat v.h. punt waarin we de grofheid willen bepalen
array	p	array met de grenzen [0:knpc+1], bevat de afstanden tussen de knooppunten op de contour.
real	g	de gevonden grofheid
real	s	de kortste afstand uit het array P.

3 Werking v.d procedure

Zie ook 4 : Blokschema.

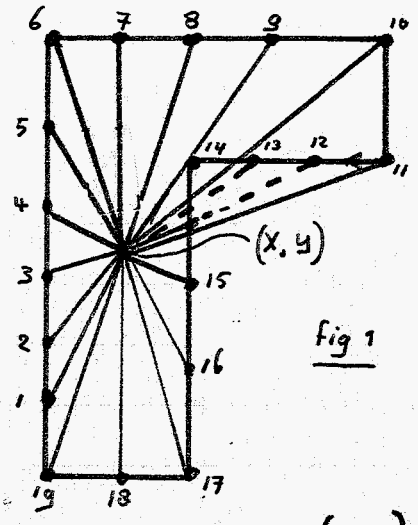


fig 1

Geg: de contour met de knooppunten. Elk driehoekje door (X,y) met de contour gemaakt wordt in de procedure bekeken. (zie fig 1).

Als voorbeeld bekijken we nader het driehoekje: (X,y - 2=3) zie fig 2

De letters in de tekening corresponderen met de in de procedure gehanteerde lokale parameters.

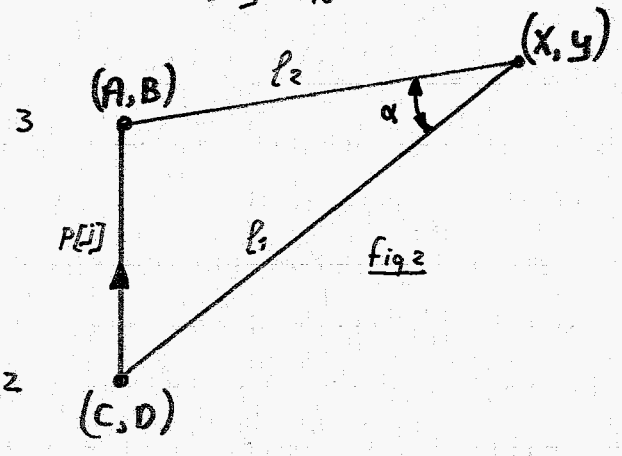


fig 2

(C,D) is het j^e knooppunt v.d contour.

Na bepaling v.d zijden v.d driehoeken en kwadraten van deze zijden bepalen we m.b.v. de determinantregel het opp. v.d driehoek

$$Opp = (A \times y) + (X \times D) + (C \times B) - (X \times B) - (C \times y) - (A \times D).$$

Indien dit opp > 0 is (3), betekent dit dat we de zijde (C,D - A,B) in feite vanuit (X,y) niet "zien". Een voorbeeld hiervan is de zijde (12 - 13). De invloed van de grofheid van zo'n lijnstuk op de grofheid in (X,y) is voor ons niet interessant. We laten zo'n driehoek verder buiten beschouwing. Zijde 10 - 11 die we ook niet "zien" maakt met (X,y) een driehoek met een negatief opp. Controles om een dergelijke zijden niet mee te laten doen zijn niet ingebouwd. Meestal ligt zo'n zijde echter ver weg zodat de invloed op de grofheid in (X,y) niet al te groot zal zijn.

Indien het opp ≈ 0 is (4), kunnen zich de volgende 3 situaties voordoen : zie fig 3, 4 en 5

a) (X,y) ligt in het verlengde van (A,B - C,D). Dit is het geval als $p^2[j] < abs(l_1^2 - l_2^2)$ (5). In dit geval laten we de zijde geen invloed op de grofheid in (X,y) hebben. (fig 3)

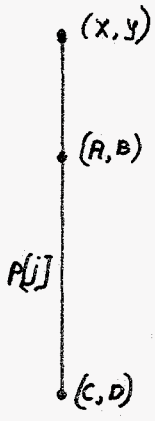


fig. 3

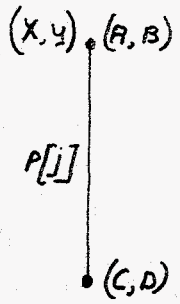


fig. 4

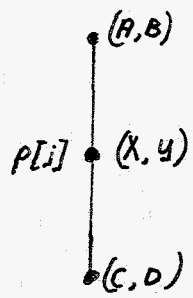


fig. 5

b) (X,y) ligt op of vlakbij (A,B). (6). : (fig 4)
We nemen dan als grofheid in (X,y) : $\dfrac{p[j] + p[j+1]}{2 \times s}$

c) (X,y) ligt tussen de punten (A,B) en (C,D) (fig 5).
Als grofheid nemen we de grofheid v.h. lijnstuk $(A,B-C,D)$: $\dfrac{p[j]}{s}$

In het geval dat $Opp < 0$ heeft lijnstuk $(A,B)-(C,D)$ invloed op de grofheid in (X,y), die we tot uiting brengen door $p[j]$ te vermenigvuldigen met een factor alfa = $\dfrac{\alpha}{\text{afstand } (X,y) \text{ tot lijn stuk: } (A,B)-(C,D)}$

α wordt bepaald met de cos. regel.

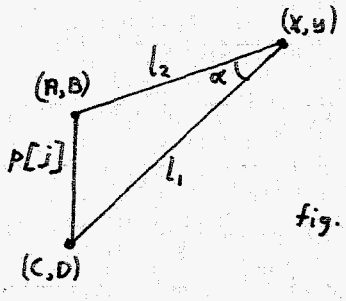


fig. 6

Indien $p[j]^2 < abs(l_1^2 - l_2^2)$ (10) (fig 6),
brengen we de afstand van (X,y) tot $(A,B)-(C,D)$
in rekening door : $(l_1 + l_2)/2$.
En anders (fig 7) door de werkelijke afstand
tot het lijn stuk te nemen : $(2 \times Opp)/p[j]$ (11).

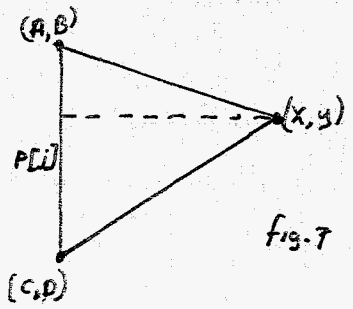
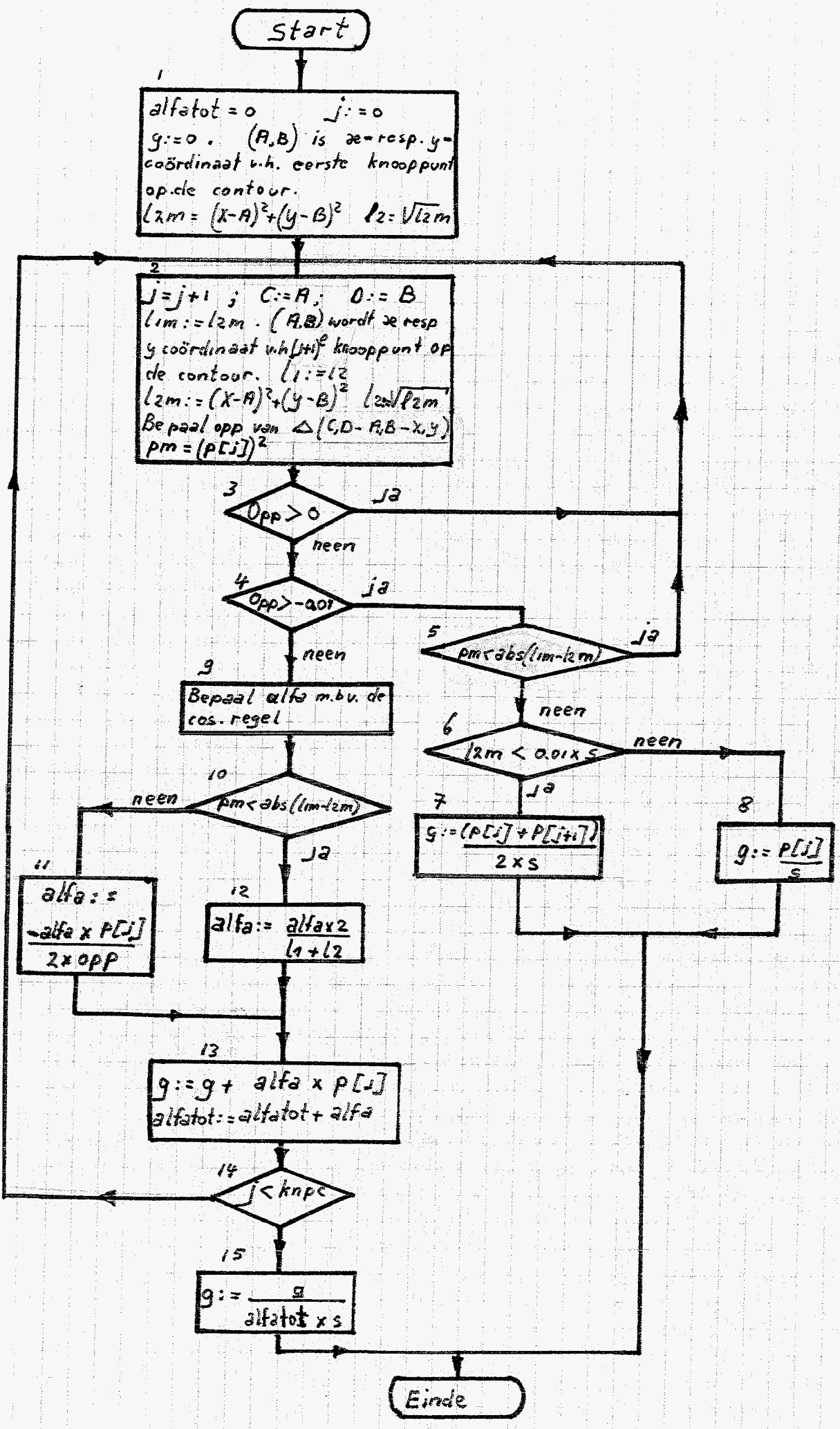


fig. 7

De uiteindelijke grofheid in (X,y) vinden we door sommatie v.d grofheden die we in elk driehoekje gevonden hebben en deze som te delen door de som v.d. factoren alfa.

4 Blokschema



5. Programmatekst

```
'PROCEDURE'BPG(KNOOPPUNT,CONTOUR,X,Y,KNPC,G,S,P);                    BPG  1
'REAL'G,X,Y,S ;'INTEGER'KNPC;'ARRAY'KNOOPPUNT[1,1],CONTOUR[0],P[0];  BPG  2
'BEGIN''REAL'A,B,ALFA,L1,L2,L1M,L2M,PM,OPP,ALFATOT,C,D;              BPG  3
'INTEGER'J;'LABEL'OUT,VERVOLG;                                       BPG  4
    ALFATOT:=0;G:=0;A:=KNOOPPUNT[CONTOUR[1],1];B:=KNOOPPUNT[CONTOUR   BPG  5
    [1],2];L2M:=(X-A)**2+(Y-B)**2;L2:=SQRT(L2M);                     BPG  6
    'FOR'J:=1'STEP'1'UNTIL'KNPC'DO'                                  BPG  7
VERVOLG:'BEGIN'C:=A;D:=B;                                            BPG  8
        L1M:=L2M;A:=KNOOPPUNT[CONTOUR[J+1],1];L1:=L2;                BPG  9
               B:=KNOOPPUNT[CONTOUR[J+1],2];                         BPG 10
        L2M:=(X-A)**2+(Y-B)**2;L2:=SQRT(L2M);PM:=P[J]**2;            BPG 11
        OPP:=A*Y+X*D+C*B-X*B-C*Y-A*D;                                BPG 12
        'IF'OPP'GTR'0'THEN''BEGIN'J:=J+1;'GOTO'VERVOLG'END';         BPG 13
        'IF'OPP'GTR'-0.01'THEN''BEGIN'                               BPG 14
        'IF'PM'LSS'ABS(L1M-L2M)'THEN''BEGIN'J:=J+1;'GOTO'VERVOLG'END' BPG 15
        'ELSE''BEGIN''IF'L2M'LSS'0.01*S'THEN'G:=(P[J]+P[J+1])/(2*S)  BPG 16
                  'ELSE'G:=P[J]/S;'GOTO'OUT;'END';'END';             BPG 16A
        ALFA:=ARCCOS((L1M+L2M-PM)/(2*L1*L2));                        BPG 17
        'IF'PM'LSS'ABS(L1M-L2M)' THEN'                               BPG 18
        ALFA:=ALFA*2/(L1+L2)                                         BPG 19
             'ELSE'ALFA:=-ALFA*P[J]/(2*OPP);                         BPG 20
        G:=G+ALFA*P[J];ALFATOT:=ALFATOT+ALFA;                        BPG 21
    'END';                                                           BPG 22
    G:=G/(ALFATOT*S );                                               BPG 23
OUT:'END'BEPAALGROOTHEID;                                            BPG 24
```

Procedure Make driehoek

1 Inleiding

Uitgaande van de verdeling in vierhoeken kunnen we een verdeling in driehoeken verkrijgen, door in de vierhoeken één diagonaal te trekken, zodanig dat de 2 ontstane driehoeken zoveel mogelijk gelijkzijdig zijn.

2 Gebruiksaanwijzing

Procedure Makedriehoek (TD, TDD, co, ae);

2.1 Formele parameters:

array TD, array co	} Zie Divconvexvierhoek.(2.2)
array TDD	array met de grenzen [1:p, 1:3] bevat na afloop v.d procedure de knooppuntsnummers v.d ontstane driehoekige elementen
integer ae	Bevat bij het begin v.d procedure het aantal vierhoekige elementen en na afloop het aantal driehoekige elementen.

3 Discutabel

In de Bijlagen zijn enkele voorbeelden opgenomen van een verdeling in driehoeken.
Een procedure die direct driehoeken genereereert dus niet via de weg van vierkanten zal op een snellere wijze betere resultaten kunnen opleveren. Een dergelijke procedure zal in het nog te verschijnen rapport ter sprake komen.

aantal knooppunten op de omtrek: 32
aantal gegenereerde knooppunten: 63
aantal gegenereerde elementen : 70
benodigde rekentijd : 33 sec

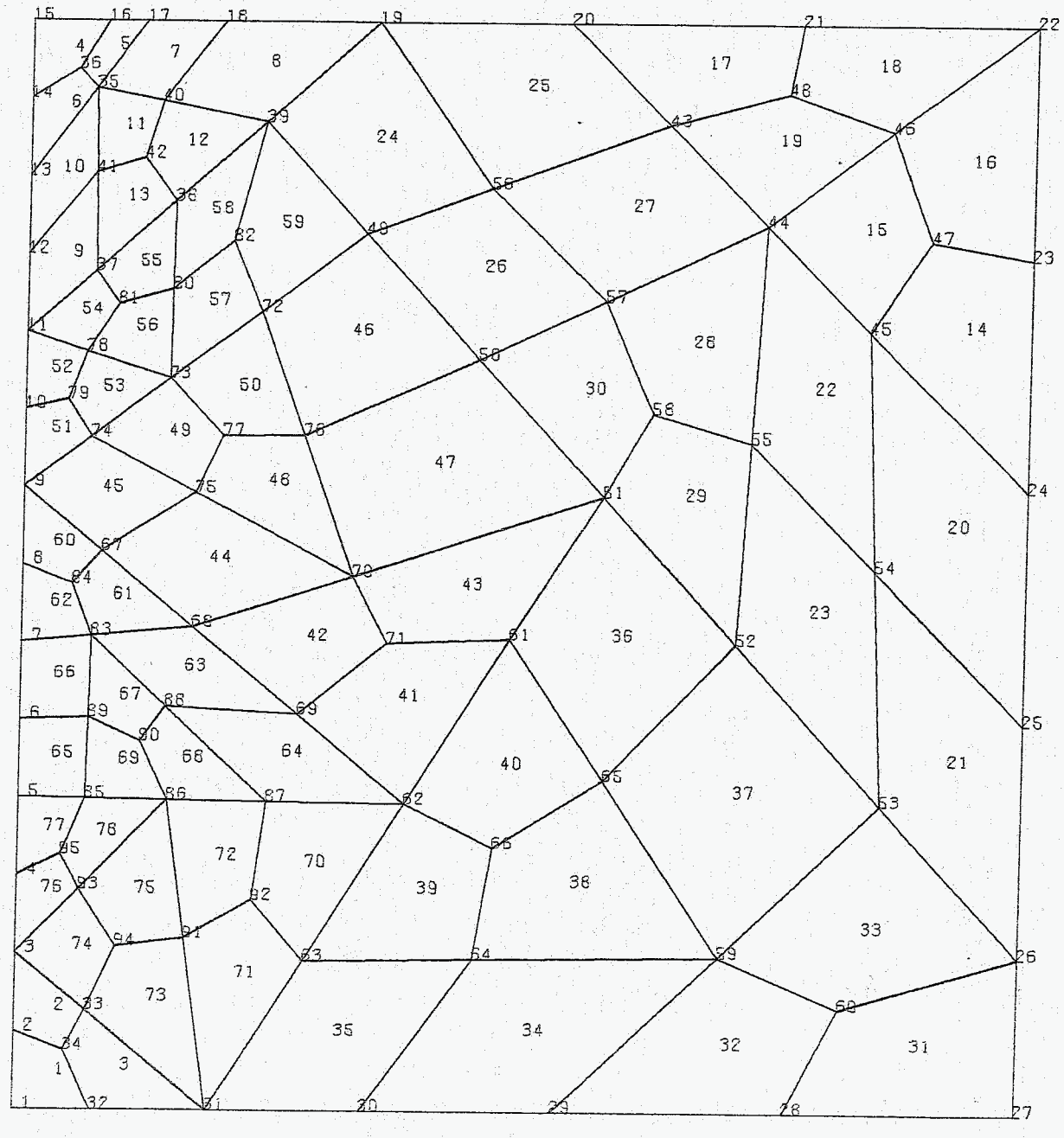


fig 1

aantal knooppunten op de omtrek: 32

aantal gegenereerde knooppunten : 63

aantal gegenereerde elementen : 156

benodigde rekentijd : 33 sec.

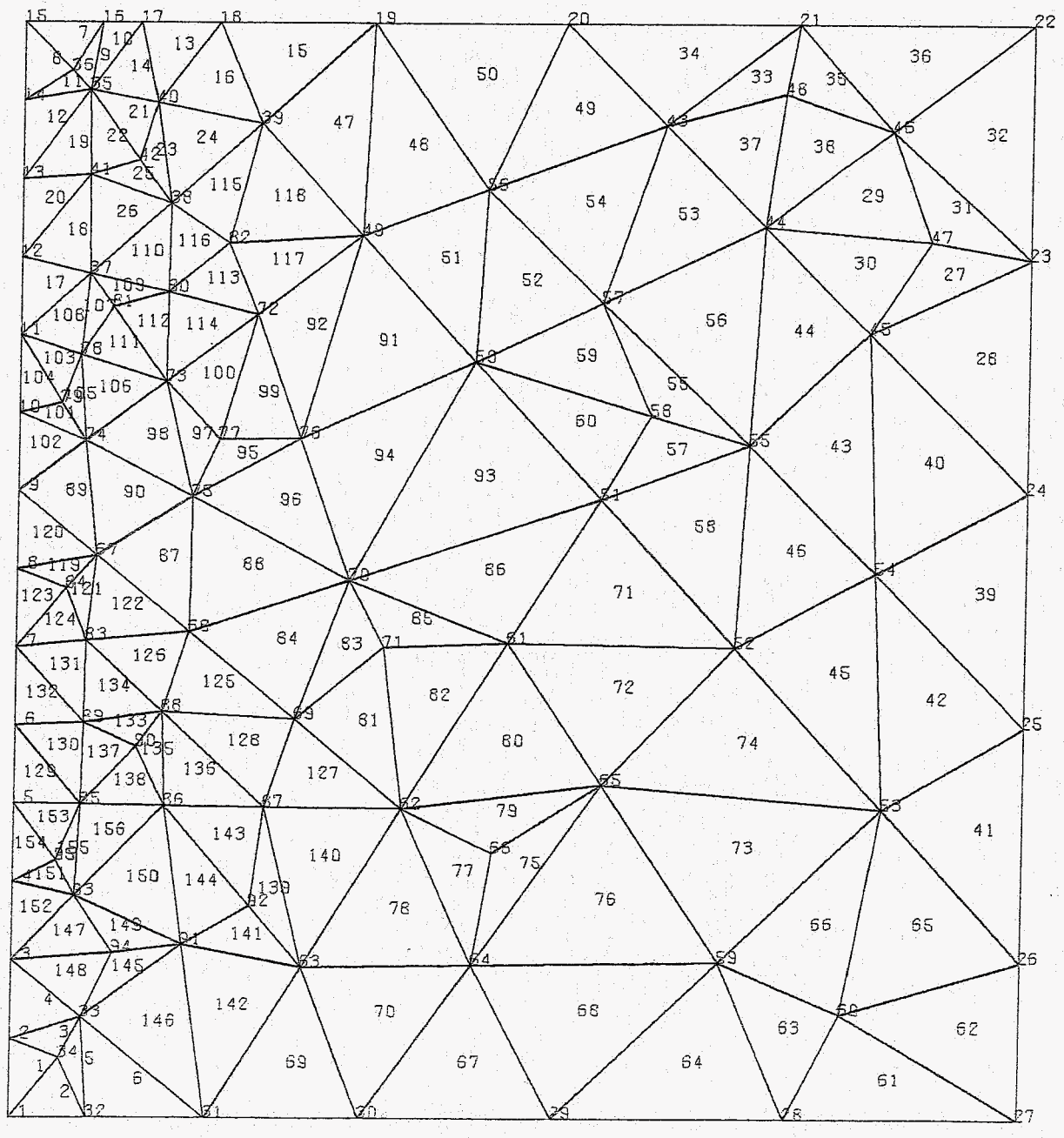
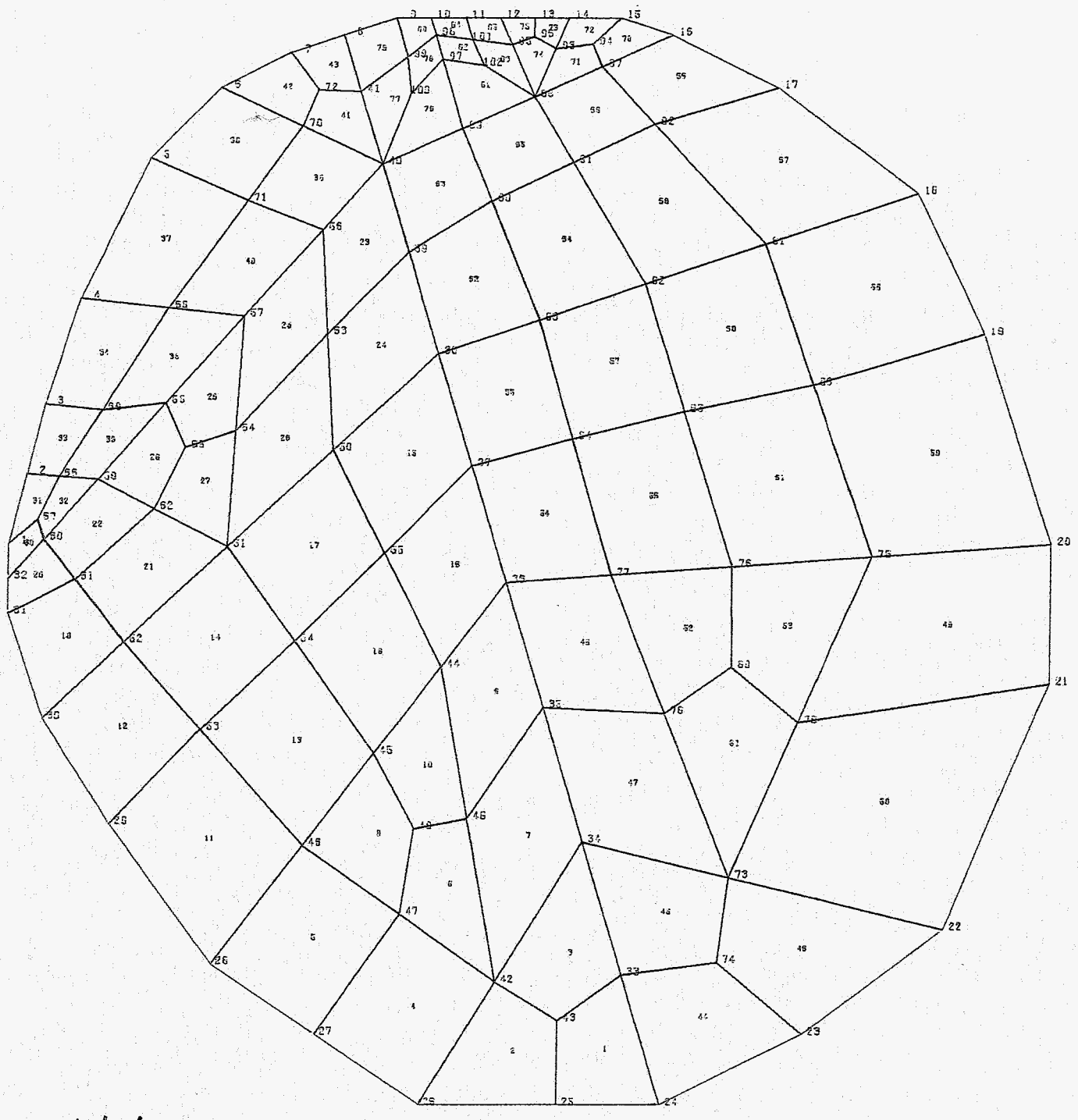
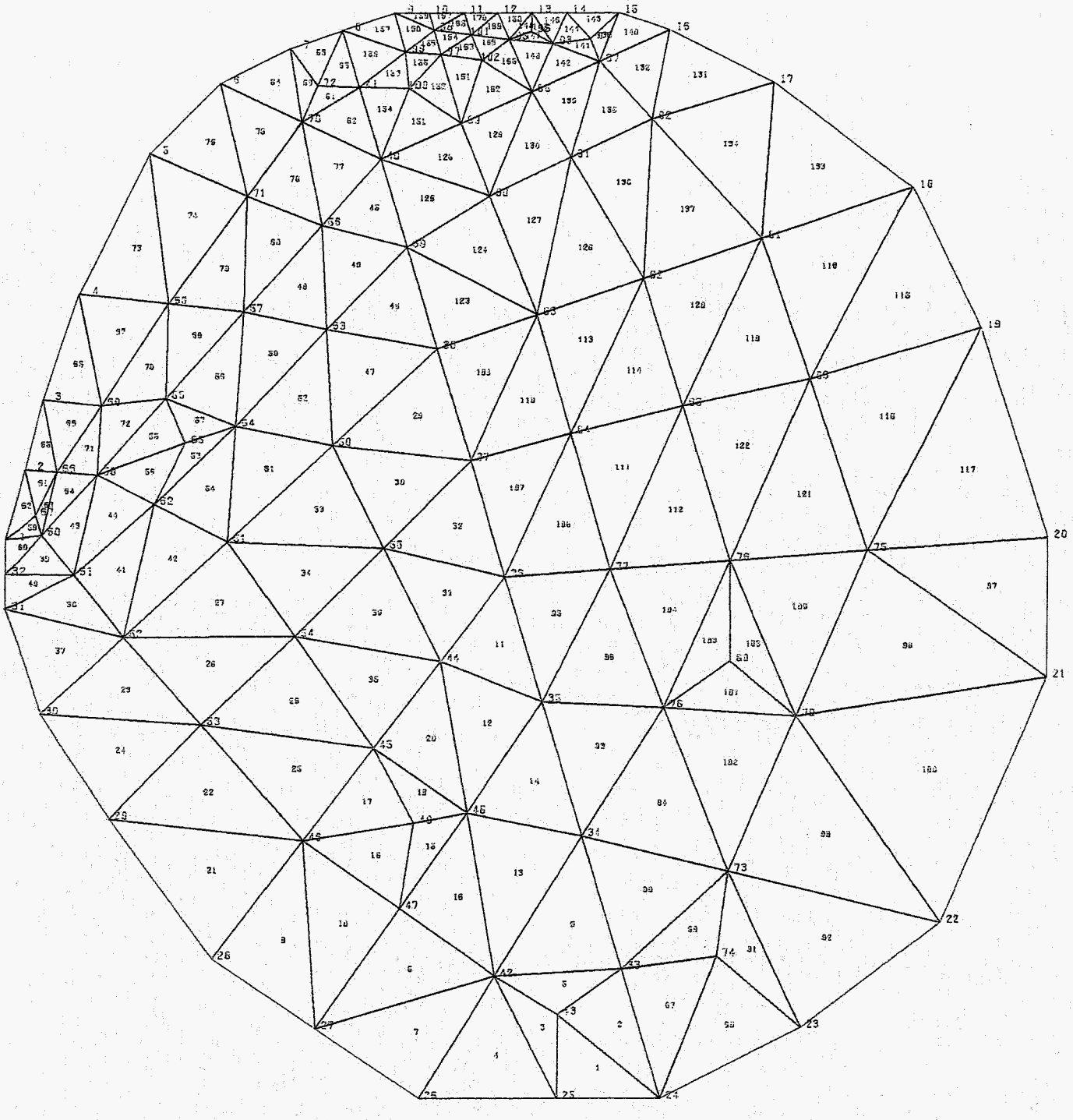


fig 2



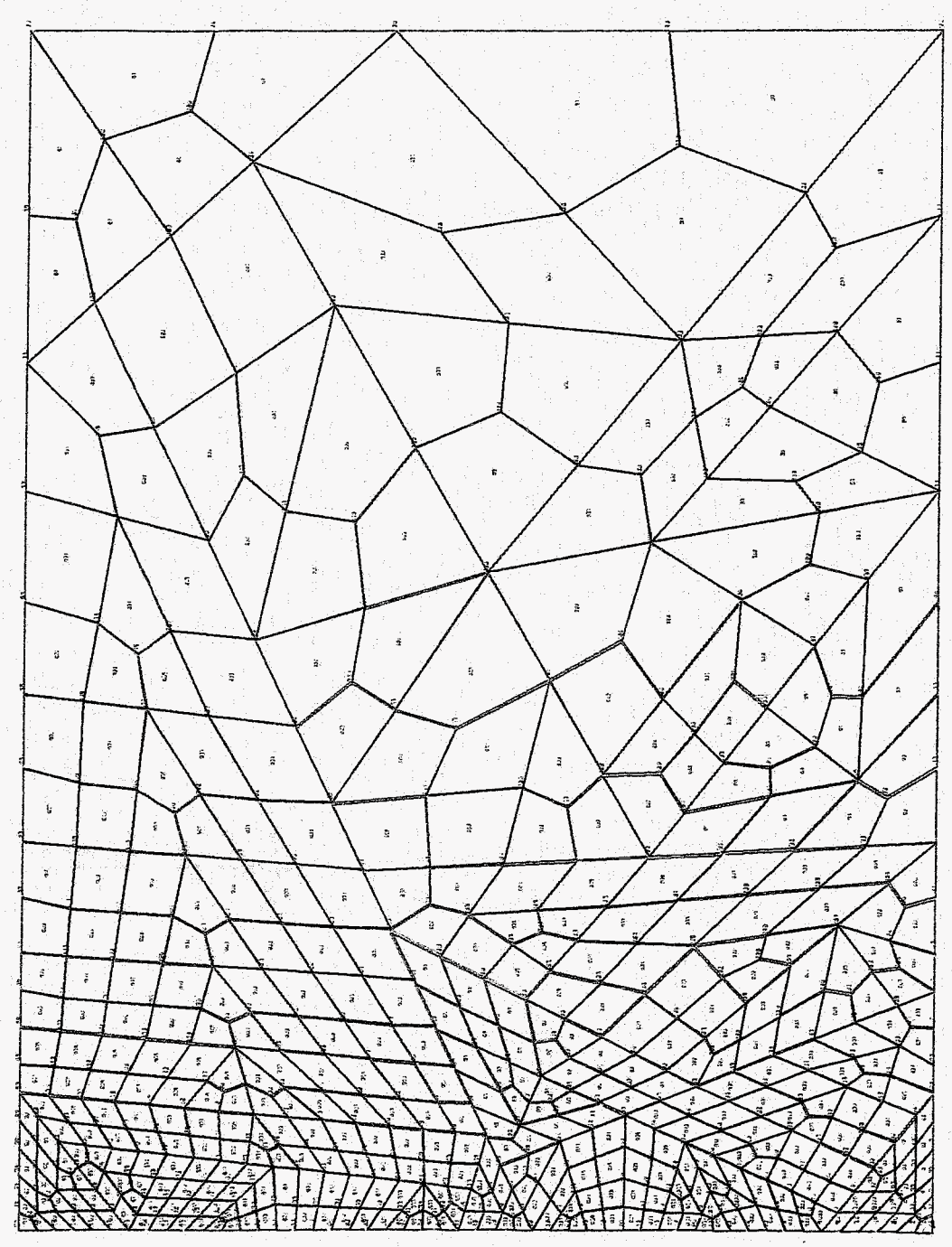
aantal knooppunten op de omtrek: 32
aantal gegenereerde knooppunten : 70
aantal gegenereerde elementen ; 85
benodigde rekentijd ; 28 sec.

fig 3



aantal knooppunten op de omtrek : 32
aantal gegenereerde knooppunten : 70
aantal gegenereerde elementen : 170
benodigde rekentijd : 20 sec

fig 4



aantal knooppunten op de omtrek: 92
aantal gegenereerde knooppunten: 406
aantal gegenereerde elementen : 451
benodigde rekentijd : 200 sec

fig 5