Rob Hendriks

NOTA 1313 november 1981

Instituut voor Cultuurtechniek en Waterhuishouding
Wageningen



# EEN COMPUTERPROGRAMMA VOOR HET BEPALEN VAN DE OPTIMALE LIGGING VAN DRIE LIJNSTUKKEN DOOR EEN SERIE GETALLENPAREN

ir. J.G. Wesseling

I N H O U D

INLEIDING

Een vaak voorkomend probleem in het onderzoek is dat men uit een serie metingen een aantal paren waarnemingen heeft verkregen. In de meeste gevallen zullen deze paren door een of meerdere rechte lijnen kunnen worden benaderd. Het is natuurlijk mogelijk om 'op het oog' een rechte lijn door de paren punten te trekken, twee punten van deze lijn te nemen en hieruit de vergelijking van die lijn te bepalen. Dit blijkt in de praktijk nogal eens tot verschillende numerieke oplossingen te leiden, vooral indien de uitkomsten gespreid liggen.

Nu praktisch iedereen wel de beschikking heeft over een rekenmachine of computer, wordt de belangstelling voor het numeriek bepalen van de 'best fit' steeds groter. Hiermee komen we op het terrein van de optimalisatie van parameters.

Deze nota beschrijft een van de vele mogelijkheden voor het oplossen van zo'n probleem: de methode van Fletcher-Reeves. Er is voor deze methode gekozen vanwege zijn betrekkelijk eenvoudig rekenschema en relatief snelle convergentie.

Na een korte probleembeschrijving zal worden ingegaan op enkele oplossingsmethoden die algemeen worden gebruikt. Vervolgens zal in het kort de methode van Fletcher-Reeves worden besproken. Het computerprogramma FLETCH zal worden beschreven met enkele toepassingen van dit programma.

Het computerprogramma FLETCH is geschreven in UCSD PASCAL en is gebruikt op zowel een Heathkit H-11 als op een PDP-11/03 computer.

VERKLARING VAN VARIABELEN

| | |
|---|---|
| $(a_i, b_i)$ | waarden waarnemingspaar i |
| N | aantal waarnemingspaar |
| $c_j$ | richtingsafgeleide lijnstuk j |
| $(r_i, s_i)$ | te optimaliseren parameterpaar i |
| $\underline{x}$ | parametervector met te optimaliseren parameters |
| | $\underline{x}' = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) = (r_1, s_1, r_2, s_2, r_3, s_3, r_4, s_4)$ |
| $f(x)$ | te minimaliseren functie |
| $f'(x)$ | 1e afgeleide van te minimaliseren functie (1-dimensionaal) |
| $f''(x)$ | 2e afgeleide van te minimaliseren functie (1-dimensionaal) |
| $\underline{\nabla f}(\underline{x})$ | afgeleidenvector (meerdimensionaal) |
| | $\underline{\nabla . f}(\underline{x})' = (\frac{\partial f}{\partial x_1}, \ldots\ldots, \frac{\partial f}{\partial x_n})$ |
| $H(\underline{x})$ | Hessiaanmatrix |

$$H(\underline{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix}$$

| | |
|---|---|
| $q(\underline{x})$ | benadering van functie f in $\underline{x}$ |
| $\underline{x}_k$ | benadering van optimum in stap k |
| $\underline{d}$ | Vector wijzend in richting van steilste helling |
| n | aantal te optimaliseren parameters |
| $\underline{\varepsilon}$ | afbreekkriterium optimalisatie procedure |
| | $\underline{\varepsilon}' = (\varepsilon_1, \ldots, \varepsilon_n)$ |
| O | restterm bij Taylor ontwikkeling |
| $\underline{e}_j$ | $j^e$ eenheidsvector |

## 1. PROBLEEMBESCHRIJVING

Stel men heeft een aantal waarnemingsparen $(a_1,b_1),\ldots\ldots,(a_N,b_N)$. Deze waarnemingen (N in totaal) kunnen van laboratoriumproeven of van veldmetingen afkomstig zijn. Het is nu gewenst om door deze punten aan aantal (al of niet rechte) lijnstukken te trekken. In deze nota wordt ervan uitgegaan dat men een serie punten heeft waardoor drie rechte lijnstukken getrokken moeten worden. De vergelijking van lijnstukken kan men als volgt opgeven:

$$s(r) = \begin{cases} c_1.(r-r_1) + s_1 & r_1 \leq r < r_2 \\ c_2.(r-r_2) + s_2 & r_2 \leq r < r_3 \\ c_3.(r-r_3) + s_3 & r_3 \leq r < r_4 \end{cases} \qquad (1)$$

Hierin zijn $(r_i,s_i)$, i=1(1)4 de coördinaten van de begin- en eindpunten van de lijnstukken. Zie fig. 1.

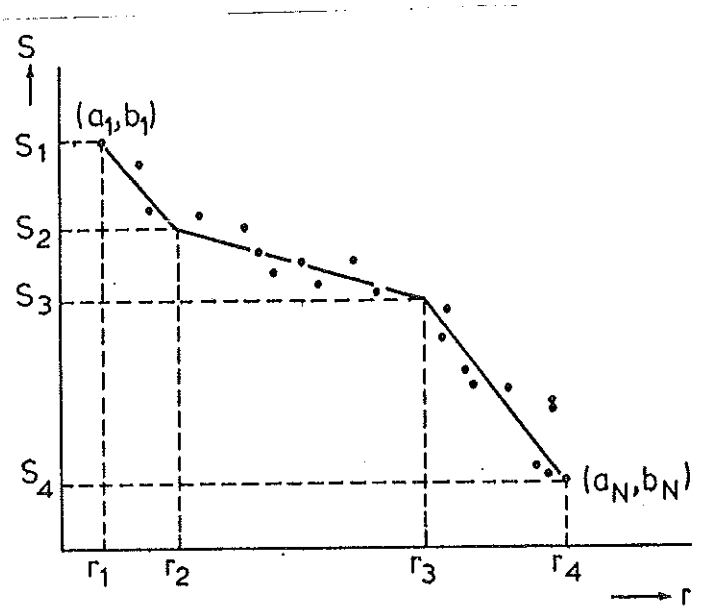


Fig. 1. Drie lijnstukken getrokken door een serie punten

De variabele $c_i$ kan nu worden bepaald uit

$$c_i = \frac{s_{i+1}-s_i}{r_{i+1}-r_i} \qquad (2)$$

Op deze wijze wordt ervoor gezorgd dat de lijnstukken aansluiten tot een continue lijn. De vraag is nu waar de paren $(r_i,s_i)$ gekozen moeten worden om de beste aanpassing te verkrijgen. Als aanpassingskriterium wordt het minimaal zijn van de som van de kwadraten van de afwijkingen van de funktiewaarden ten opzichte van de meetwaarden gebruikt:

$$f(\underline{r},\underline{s})=\sum_{i=1}^{N}(b_i-s(a_i))^2 \qquad (3)$$

De waarden van $r_1$ en $r_4$ liggen meestal vast (beneden- en bovengrens van $b_i$), zodat wanneer we aannemen dat dat voor ons onderzoek ook geldt, er 6 parameters overblijven die bepaald moeten worden, namelijk $s_1,r_2,s_2,r_3,s_3$ en $s_4$.

## 2. OPTIMALISERINGSMETHODEN

In de optimaliseringstechniek kent men twee hoofdgroepen problemen, namelijk die met en die zonder randvoorwaarden. Met randvoorwaarden wil zeggen dat de waarde die een parameter kan hebben wordt begrensd door er bijvoorbeeld een maximum of minimumwaarde aan te stellen. Ook kan het gebeuren dat er een soort relatie tussen twee parameters bestaat waarbij men met de oplossing van het probleem rekening moet houden. In dit hoofdstuk zullen enkele methoden worden genoemd voor het oplossen van optimalisatieproblemen zonder nevenvoorwaarde. De algemene vorm van het optimalisatieprobleem is: zoek een zodanig vector $\underline{x}$ dat de functiewaarde $f(\underline{x})$ minimaal is.

De oplossingsschema's kunnen worden onderverdeeld in twee groepen:

a. lijnzoekmethoden
b. direct search methoden (direkte zoekmethoden)

### 2.1. L i j n z o e k m e t h o d e n

De lijnzoekmethoden bestaan in het algemeen uit twee fasen:

I. genereer een richting waarin men moet zoeken om zo dicht mogelijk bij het minimum te komen;
II. zoek in deze richting.

Een van de eenvoudigste methoden die onder de lijnzoekmethode vallen is de Gulden Snede methode. Bij deze methode wordt aan beide zijden van een gebied waarvan men verwacht dat daar het minimum binnen ligt, een punt genomen, zodat men een interval (U,V) heeft waarin de functie bekend is die geminimaliseerd moet worden in dit interval. Door nu volgens een bepaald schema het interval systematisch te verkleinen, wordt het minimum als het ware van beide zijden benaderd door de intervalgrenzen. Zie fig. 2. Voor een uitgebreide beschrijving van dit schema kan worden verwezen naar VAN BEEK en HENDRIKS (1978).

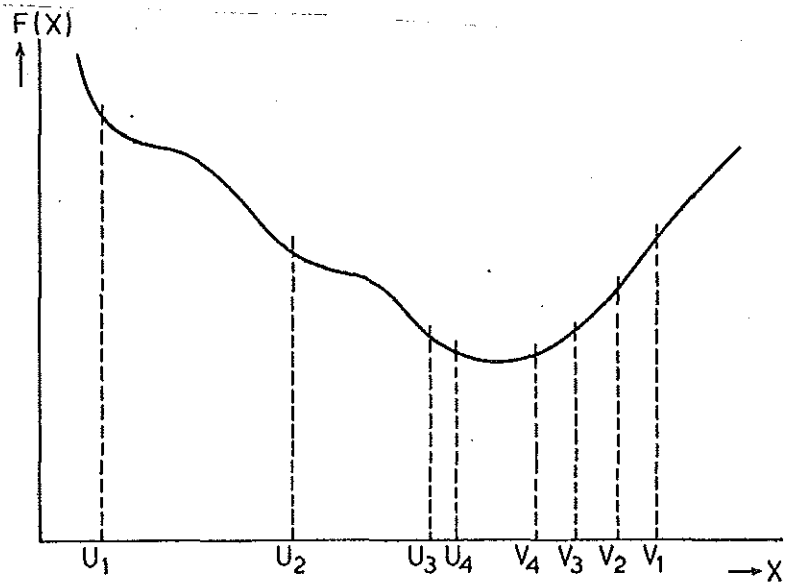


Fig. 2. Benadering van het minimum van een functie door toepassing van het Gulden Snede algoritme (VAN BEEK en HENDRIKS, 1978)

Een andere, meestal snel werkende lijnzoekmethode, is die van Newton. Ook deze is, zoals hij hier zal worden besproken, geschikt voor eendimensionale problemen.

Stel we hebben de te minimaliseren functie f(x), waarvan de afgeleiden f'(x) en f"(x) te bepalen zijn. In het punt $x_k$ kunnen we de functie f(x) door toepassing van een Taylor-ontwikkeling benaderen door de kwadratische functie q(x):

$$q(x) = f(x_k) + f'(x_k).(x-x_k) + \tfrac{1}{2}.f''(x_k).(x-x_k)^2 \qquad (4)$$

Voor het minimum van een functie geldt dat de afgeleide van de functie nul is in dit minimum. Differentiëren van (4) levert nu

$$q'(x) = f'(x_k) + f''(x_k).(x-x_k) = 0 \qquad (5)$$

Stel $x = x_{k+1}$ voldoet aan (5), dan volgt hieruit de volgende formule:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)} \qquad (f''(x_k) \neq 0) \qquad (6)$$

Deze methode blijkt vrij snel naar een oplossing te convergeren.

Een laatste veelgebruikte methode voor het eendimensionale geval is de zogenaamde kwadratische interpolatie. Men zoekt hierbij drie punten van de functie waarbij $x_1 < x_2 < x_3$ en $f(x_2) < f(x_1)$ en $f(x_2) < f(x_3)$. Door deze punten trekt men nu een parabool. Deze parabool is volledig bepaald door de drie punten. Het minimum van de parabool is nu op eenvoudige wijze te bepalen. Stel dit punt is $x_4$ en de bijbehorende functiewaarde $f(x_4)$ (zie fig. 3).

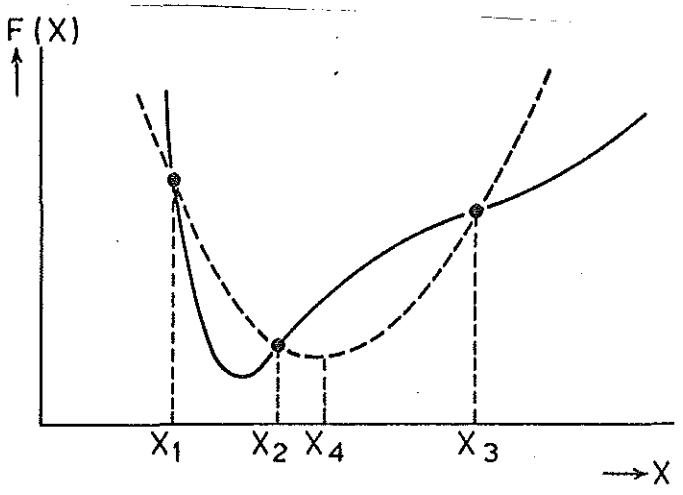


Fig. 3. Kwadratische interpolatie

Afhankelijk van $f(x_4)$ wordt dan een nieuwe parabool berekend door de punten:

I. als $f(x_4) \leq f(x_2)$ dan $\{x_1,x_2,x_3\} \leftarrow \{x_2,x_4,x_3\}$

II. als $f(x_2) < f(x_4) \leq f(x_3)$ dan $\{x_1,x_2,x_3\} \leftarrow \{x_1,x_2,x_4\}$

Bij deze methode zal het punt $x_4$ naar het minimum van de functie convergeren. Dit gebeurt sneller dan bij de Gulden Snede methode, doch langzamer dan bij de methode van Newton (VAN BEEK en HENDRIKS, 1978).

De afleiding van een methode voor het minimaliseren van een functie met meer dan één parameter verloopt identiek aan die voor het 1-dimensionale geval, wat in de vectorvoorstelling duidelijk tot uiting komt.

Stel we hebben een functie f, met n parameters, dus

$f: E_n \rightarrow R$ met continue partiele afgeleiden.

Taylor ontwikkeling van deze functie in de omgeving van het punt $\underline{x}_1$ geeft:

$$f(\underline{x}_1 + \underline{d}) = f(\underline{x}_1) + \underline{d}'.\ \underline{\nabla f}(\underline{x}_1) + 0(\underline{d}) \qquad (7)$$

De vraag is nu welke $\underline{d}$ men moet kiezen opdat $\underline{x}_1 + \underline{d}$ de steilste helling in $\underline{x}_1$ levert en dus het snelst naar het minimum zal leiden. Daar het inprodukt van 2 loodrechte vectoren 0 is, is het duidelijk dat als men voor $\underline{d}$ een vector loodrecht op $\underline{\nabla f}(\underline{x}_1)$ neemt, $f(\underline{x}_1+\underline{d})$ slechts weinig zal veranderen. De grootste verandering zal optreden als men $\underline{d} = -\underline{\nabla f}(\underline{x}_i)$ neemt.

De procedure in deze zogenaamde 'steilste helling methode' is nu:

i. ga uit van een punt $\underline{x}_o$, dus stel i=0

ii. bereken in dit punt $f(\underline{x}_i)$ en $\underline{\nabla f}(\underline{x}_i)$
Als $\underline{\nabla f}(\underline{x}_i) = \underline{0}$ dan is het proces klaar

iii. $\underline{d}_i = -\underline{\nabla f}(\underline{x}_i)$
Bepaal nu het minimum van de functie f langs de lijn door $\underline{x}_i$ in de richting $\underline{d}_i$, ofwel bepaal de $\alpha_i$ uit

$$\min_{\alpha_i} \{f(\underline{x}_i + \alpha_i.\underline{d}_i)\},\ \alpha_i \geq 0 \qquad (8)$$

iv. $\underline{x}_{i+1} = \underline{x}_i + \alpha_i.\underline{d}_i$

ga naar ii waarin $\underline{x}_i$ vervangen wordt door $\underline{x}_{i+1}$

Bij deze methode zal het minimum volgens een 'zig-zag-patroon' worden benaderd, zie fig. 4. STOL (1975) bewees dat de steilste helling en dus het snelste convergeren naar een minimum alleen opgaat voor lineaire functies (STOL, 1975, pag. 110). In alle andere gevallen wordt een sub-minimum gevonden, en zeker niet 'het snelst' (STOL, 1975, fig. 5 en 38). Hiermee moet goed rekening worden gehouden bij het kiezen van een oplossingsmethode voor een bepaald probleem.

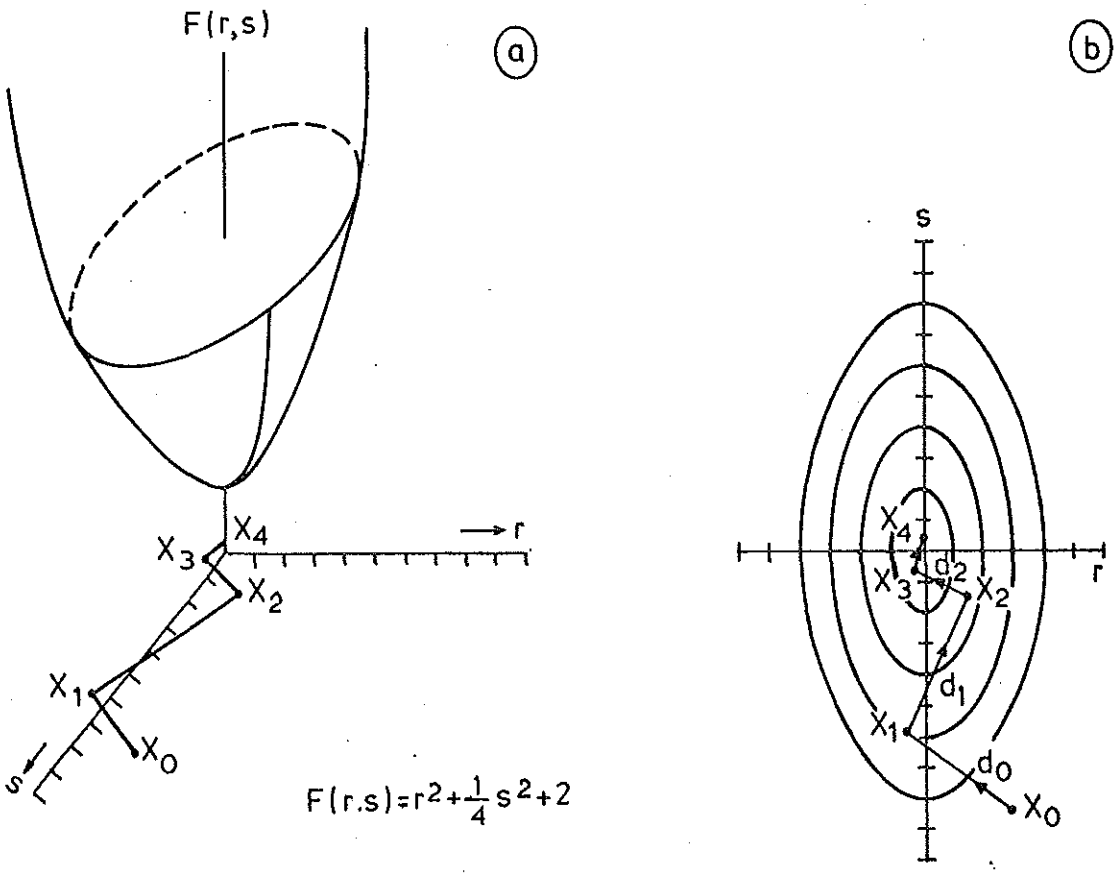


Fig. 4. Benadering van het minimum van een functie $f(r,s) = r^2+\frac{1}{4}s^2+2$ met behulp van de steilste helling methode

i. Driedimensionaal

ii. Tweedimensionaal beeld met lijnen van gelijke functiewaarde

Bij de methode van Newton neemt men nog een extra term mee in de Taylorreeks die men van de benadering van de functie $r : E_n \rightarrow R$ ontwikkelt in punt $\underline{\hat{x}} \in E_n$:

$$q(\underline{x}) = f(\underline{\hat{x}}) + \underline{\nabla f}(\underline{\hat{x}}).(x-\underline{\hat{x}}).(\underline{x}-\underline{\hat{x}}) + \tfrac{1}{2}.(\underline{x}-\underline{\hat{x}})'.H(\underline{\hat{x}}).(x-\underline{\hat{x}}) \qquad (9)$$

Voor een minimum moet nu gelden dat $\underline{\nabla q}(\underline{x}) = 0$ ofwel voor $\underline{x} = \underline{x}_{k+1}$

$$\underline{\nabla f}(\underline{x}_k) + H(\underline{x}_k)'.(\underline{x}_{k+1}-\underline{x}_k) = 0 \qquad (10)$$

Dit levert nu

$$\underline{x}_{k+1} = \underline{x}_k - H^{-1}(\underline{x}_k).\underline{\nabla f}(\underline{x}_k) \qquad (11)$$

Daar het inverteren van een matrix een tijdrovend en arbeidsintensief werk is werkt men meestal volgens het volgende schema waarbij ervan wordt uitgegaan dat over de per stap geringe voortgang over het vereffeningsoppervlak de matrix H niet veel verandert:

i. bereken zoekrichting $\underline{d}_k = -H^{-1}(\underline{x}_j).\underline{\nabla f}(\underline{x}_k)$ (12)

met j de stap waarin H voor de j-de maal verandert ($j < k$), ofwel zoek de oplossing van het stelsel vergelijkingen

$$H(\underline{x}_j).\underline{d}_k = -\underline{\nabla f}(\underline{x}_k) \qquad (13)$$

Dit kan op een van de vele bekende manieren gebeuren, bijvoorbeeld door middel van Cholesky-decompositie of Gauss eliminatie

ii. $\underline{x}_{k+1} = \underline{x}_k + \alpha_i.\underline{d}_k$ (14)

Hier geeft het berekenen van $\alpha_i$ als verhouding tussen schaalfaktoren een belangrijke versnelling van de procedure (STOL, 1975, pag. 67 e.v.).

Deze procedure wordt herhaald tot twee opeenvolgende waarden van $\underline{x}_k$ dicht genoeg bij elkaar liggen, dus totdat

$$|\underline{x}_{k+1} - \underline{x}_k| < \underline{\varepsilon} \tag{15}$$

waarin $\underline{\varepsilon}$ een vector is met tevoren vastgelegde waarden. Vaak gebruikt men ook als kriterium

$$|\underline{d}_k| < \underline{\varepsilon} \tag{16}$$

daar in het minimum de afgeleidenvector immers gelijk is aan de nulvector.

Bovenbeschreven methode wordt veelvuldig gebruikt als uitgangspunt voor gecompliceerdere methodes. Zo kan men bijvoorbeeld in stap ii weer in de richting $\underline{d}_k$ een lijnminimum gaan zoeken.

## 2.2. Direct Search methoden

Indien men de afgeleiden van een functie niet kan (of wil) bepalen, maakt men gebruik van een zogenaamde Direct Search methode. Deze methodes maken alleen gebruik van functiewaarden. Zij zijn uiteraard inferieur aan de methodes die met afgeleiden werken, daar laatstgenoemde methodes veel meer gebruik maken van de eigenschappen van de functies.

Een veelgebruikte methode, die ik hier niet zal bespreken omdat zij in praktisch elk boek wel voorkomt, is de zogenaamde Simplex methode. Een andere methode is die van Hooke en Jeeves. Deze kan als volgt worden beschreven (VAN BEEK EN HENDRIKS, 1978):

Stel $f : E_n \to R$ is overal continu in $E_n$

Het is de bedoeling f te minimaliseren door uitsluitend gebruik te maken van functiewaarden (het is immers mogelijk dat de gradient $\underline{\nabla} f(x)$ niet bestaat).

De methode van Hooke en Jeeves bestaat uit twee stappen:

i. een lokale stap

ii. een basisstap

Tijdens de lokale stap wordt het lokale gedrag van f onderzocht. De lokale stap heeft tevens tot doel een richting te bepalen waarlangs de functie afneemt. Tijdens de basisstap wordt een stap in die richting gezet. Er wordt uitgegaan van een startvector $\underline{x}_1$.

i. de lokale stap

Bij de lokale stap worden stappen ter grootte $\underline{e}_j\Delta$ gezet in de diverse coördinaatrichtingen, $j = 1,2,\ldots,n$. De lokale stap gaat nu als volgt:

(0) Bepaal de functiewaarde $f(\underline{x}_1)$ in het startpunt $\underline{x}_1$ en zet $j=1$

(1) Zet $\underline{x}_{j+1} = \underline{x}_j + \underline{e}_j\Delta$ (waarbij $\underline{e}_j$ de $j^e$ eenheidsvector is) en bereken $f(\underline{x}_{j+1})$

(2) Als $f(\underline{x}_{j+1}) > f(\underline{x}_j)$ dan zet $\underline{x}_{j+1} = \underline{x}_j - \underline{e}_j\Delta$ en bereken $f(\underline{x}_{j+1})$

(3) Als $f(\underline{x}_{j+1}) < f(\underline{x}_j)$ en $j < n$ dan zet $j = j+1$ en ga terug naar (1)

(4) Als $f(\underline{x}_{j+1}) \geq f(\underline{x}_j)$ en $j < n$ dan zet $\underline{x}_{j+1} = \underline{x}_j$ , $j=j+1$ en ga terug naar (1)

(5) Als $j=n$ dan zet $\underline{x}_B = \underline{x}_{j+1}$

Hiermee is de 'lokale stap' gereed.

De vector $\underline{x}_B$ wordt na deze lokale stap het lopende basispunt genoemd, terwijl de vector $\underline{x}_1$ waarmee begonnen is het vorige basispunt $\hat{\underline{x}}_B$ genoemd wordt.

ii. basisstap

De basisstap bestaat uit een stap uitgaande van $\underline{x}_B$ met richting $\underline{x}_B - \hat{\underline{x}}_B$ :

$$\underline{x}_N = \underline{x}_B + (\underline{x}_B - \hat{\underline{x}}_B) \qquad (17)$$

Na deze basisstap arriveren we in het punt $\underline{x}_N \in E_n$

Vergelijk nu niet $f(\underline{x}_N)$ met $f(\underline{x}_B)$ maar maak uitgaande van $\underline{x}_N$ wederom een lokale stap.
Laat het resultaat van de lokale stap het punt $\underline{x}_{LN} \in E_n$ zijn.

Indien geldt: $f(\underline{x}_{LN}) < f(\underline{x}_B)$ dan stellen we:

$$\hat{\underline{x}}_B := \underline{x}_B$$

en (18)

$$\underline{x}_B := \underline{x}_{LN},$$

met andere woorden het nieuwe lopende basispunt ($\underline{x}_B$) wordt $\underline{x}_{LN}$ en het vorige basispunt ($\hat{\underline{x}}_B$) wordt $\underline{x}_B$.

Vervolgens wordt een basisstap uitgevoerd zoals in (17) maar nu met de nieuwe $\hat{\underline{x}}_B$ en $\underline{x}_B$ zoals gedefinieerd in (18).

Indien echter geldt: $f(\underline{x}_{LN}) \geq f(\underline{x}_B)$ dan wordt teruggekeerd naar $\underline{x}_B$ en vanuit $\underline{x}_B$ wordt een lokale stap gedaan. Indien deze stap géén punt $\underline{x}_L$ oplevert met een lagere functiewaarde, dan wordt de stapgrootte $\underline{e}_j\Delta$ gereduceerd, $j=1,2,\ldots,n$ (bijvoorbeeld i.p.v. $\underline{e}_j\Delta$ nu $\rho\underline{e}_j\Delta$ met $0 < \rho < 1$) en wordt opnieuw een lokale stap gedaan vanuit $\underline{x}_B$. Indien de stapgrootte $\rho\underline{e}_j\Delta$ beneden bepaalde, tevoren gespecificeerde waarden $\varepsilon_j$, $j=1,2,\ldots,n$, gekomen zijn, wordt het proces gestopt.

De methode van Hooke en Jeeves is toegepast op hetzelfde probleem als zal worden beschreven voor de methode van Fletcher-Reeves. Voor het berekenen van maximaal 3 parameters is de rekentijd nog binnen de grenzen van het redelijke, maar voor een probleem met 5 parameters wordt, door de vele stappen die gedaan moeten worden, de rekentijd veel te groot.

Een laatste klasse van optimaliseringstechnieken die hier behandeld worden is de klasse die gebruik maakt van geconjugeerde richtingen (Conjugate Direction Methods). Het zou te ver voeren hiervan de theorie te behandelen. Daarom wordt daarvoor verwezen naar de literatuur (VAN BEEK en HENDRIKS, 1978, LUENBERGER, 1973). Een van deze methodes is de Fletcher-Reeves methode (LUENBERGER, 1973). Het voordeel van deze methode is dat er geen tweede afgeleiden van de functies bepaald hoeven te worden. Het complete algoritme is:

i. bereken bij een gegeven $\underline{x}_o$ de afgeleidevector $\underline{g}_o = \underline{\nabla f}(\underline{x}_o)'$ en stel $\underline{d}_o = \underline{g}_o$

ii. voor k=o,1,...n-1 :

a. stel $\underline{x}_{k+1} = \underline{x}_k + \alpha_k \cdot \underline{d}_k$ waarin $\alpha_k$ de functie

$f(\underline{x}_k + \alpha . \underline{d}_k)$ minimaliseert

b. bereken $\underline{g}_{k+1} = \underline{\nabla f}(\underline{x}_{k+1})'$

c. als $k \neq n-1$ stel dan

$$\underline{d}_{k+1} = - \underline{g}_{k+1} + \beta_k \cdot \underline{d}_k$$

waarin

$$\beta_k = \frac{\underline{g}'_{k+1} \cdot \underline{g}_{k+1}}{\underline{g}'_k \cdot \underline{g}_k}$$

iii. vervang $\underline{x}_o$ door $\underline{x}_k$ en ga naar stap i.

Deze methode maakt elke n stappen een stap volgens de steilste helling methode. Hierdoor is men verzekerd van convergentie. In de literatuur (LUENBERGER, 1973) wordt uitgebreid op de convergentie van deze methode ingegaan.

De waarde van de variabele n kan men nu zelf kiezen. Ik heb gekozen voor de waarde 10, waarbij echter wel wordt gekeken of $\underline{x}_{k+1}$ en $\underline{x}_k$ al naar een waarde convergeren. Indien deze waarden dicht genoeg bij elkaar liggen, wordt stap ii afgebroken en verdergegaan met stap iii. Het eerste wat men in het algemeen in stap iii zal doen is kijken of het hele proces al dicht genoeg naar een minimum is geconvergeerd. Als dit het geval is, zal de procedure worden afgebroken.

## 3. HET PROGRAMMA

Het computerprogramma FLETCH bestaat uit de volgende onderdelen:

- het hoofdprogramma
- de subprogramma's TIMER
  AFGEL
  COMPARE
  LIJNMIN
  DERIV
  OPT
  VARPRI
  PRPLOT
  XAXIS

Daar men in PASCAL een subroutine moet definiëren voordat hij aangeroepen kan worden (behalve bij gebruik van het FORWARD statement), zullen deze programma-onderdelen van onder naar boven besproken worden.

### a. Het hoofdprogramma

Er is in het programma een mogelijkheid ingebouwd om, tijdens de optimalisatie, de berekening af te breken zonder dat de resultaten verloren gaan. Dit komt omdat na iedere stap de resultaten worden weggeschreven. Het eerste wat men dus moet weten bij het starten van het programma is of men met nieuwe data wil beginnen, een oude afgebroken optimalisatie wil voortzetten of alleen de resultaten wil laten printplotten. Afhankelijk van de gekozen mogelijkheid roept het hoofdprogramma dan de betreffende subprogramma's aan.

Het aantal dataparen waardoor de lijnen moeten worden getrokken (NUMDAT) samen met deze dataparen (A en B) worden ingelezen van een datafile genaamd PASCAL:OPTIM.DAT.TEXT. Het maximaal toegestane aantal dataparen is 50.

Afhankelijk van bovengenoemde keuzemogelijkheid worden de beginwaarden (respectievelijk tot nu toe berekende waarden) ingelezen van de datafiles PASCAL:INIT.DAT.TEXT of PASCAL:TEMP.DATA.

De gebruiker van het programma moet de data geven als paren x- en y startwaarden. In het hoofdprogramma worden de laatste x- en y-waarde verwisseld, daar de x-waarde niet veranderd hoeft te worden, terwijl de y-waarde geoptimaliseerd moet worden, zodat de te optimaliseren parameters achter elkaar staan. Vervolgens worden de optimalisatie-, en uitvoersubprogramma's aangeroepen. Tenslotte wordt op het beeldscherm de tijd gegeven, die de computer nodig heeft gehad om het programma uit te voeren.

b. Procedure XAXIS

Deze procedure maakt de print-plot van de x-as op de printer. Daar in UCSD Pascal de lengte van een procedure beperkt moet blijven tot een gegeven aantal woorden, was het noodzakelijk deze handeling door een aparte procedure te laten uitvoeren.

c. Procedure PRPLOT

Door de procedure PRPLOT wordt een plot van de uitkomsten gemaakt op de printer. Als respectievelijk onder- en bovengrens voor de x-as worden de kleinste en grootste A-waarden gebruikt, voor de y-as de kleinste en grootste B-waarden.

d. Procedure VARPRI

Deze procedure geeft de x-waarden en y-waarden als dataparen.

e. Procedure OPT

Deze procedure zorgt voor het eigenlijke optimaliseren volgens de besproken methode van FLETCHER-REEVES. Hierbij is NTOT het maximaal aantal keren dat stap ii uitgevoerd mag worden zonder over te stappen naar stap iii. Er wordt wel getest of er nog een stap nodig is.

Elke keer wanneer van stap ii naar stap iii wordt overgegaan worden de tussenresultaten weggeschreven naar file PASCAL:TEMP.DATA. Hiervan kan men later weer gebruik maken als men het programma om de een of andere reden moet afbreken.

f. Procedure DERIV

In procedure DERIV wordt op numerieke wijze de afgeleidenvector $\underline{\nabla f}(\underline{x})$ bepaald.

g. Procedure LIJNMIN

In de procedure LIJNMIN wordt vanuit een punt gegeven door vector X in de richting die wordt gegeven door de vector DIR naar een lijnminimum gezocht. Het nieuwe minimum wordt XN genoemd. Na iedere keer dat dit subprogramma is aangeroepen verschijnt op het beeldscherm de nieuwe waarde van de som van de kwadraten van de afwijkingen.

h. Procedure COMPARE

Deze procedure vergelijkt de functiewaarde in een punt gegeven door de vector TEST met de functiewaarde in het minimum, gegeven door de vector MIN. Als de functiewaarde in TEST kleiner is dan in MIN wordt de vector TEST gecopiëerd naar MIN.

i. Function FUN

In deze function wordt de som van de kwadraten van de afwijkingen berekend. Dit geschiedt door bij elke A[I] een waarde te berekenen volgens (1). Deze wordt REK genoemd. Vervolgens wordt het verschil met B[I] berekend, waarna dit verschil gekwadrateerd wordt en al deze kwadraten worden gesommeerd.

j. Procedure AFGEL

De berekening van de hellingshoeken van de drie lijnstukken, zoals gegeven door verg. (2) wordt gedaan in de procedure AFGEL.

k. Procedure TIMER

TIMER geeft de tijd op het beeldscherm die is verlopen sinds het starten van het programma. Dit hoeft niet de rekentijd te zijn. De benodigde printtijd is hier ook bij inbegrepen, evenals de tijd benodigd voor het lezen en schrijven van data van en naar schijf.

## 4. DE INVOER

Het programma maakt gebruik van twee datafiles:

a. PASCAL:OPTIM.DAT,TEXT

Uit deze datafile worden de dataparen A en B gelezen, voorafgegaan door het aantal paren. De opbouw is als volgt:

| Variabele | Soort | Beschrijving |
|---|---|---|
| NUMDAT | integer | aantal dataparen |
| A[1] B[1] | real | datapaar 1 |
| A[2] B[2] | real | datapaar 2 |
| ⋮ | | |
| A[NUMDAT] B[NUMDAT] | real | datapaar NUMDAT |

De file bestaat dus uit NUMDAT+1 regels. De dataparen zijn real getallen gescheiden door een spatie.

b. PASCAL:INIT.DAT.TEXT

Deze file bevat de startwaarden voor de x- en y-variabelen. Deze moeten worden gegeven in de volgende volgorde:

| Variabele | Soort | Naam in fig. 1 |
|---|---|---|
| X[0] | real | r1 |
| X[1] | real | s1 |
| X[2] | real | r2 |
| X[3] | real | s2 |
| X[4] | real | r3 |
| X[5] | real | s3 |
| X[6] | real | r4 |
| X[7] | real | s4 |

De datafile bestaat dus uit 8 regels met elk 1 real getal.

## 5. PROGRAMMA EXECUTIE

Na het starten van het programma zal op het beeldscherm worden gevraagd welke aktie moet worden ondernomen. Er zijn drie keuzemogelijkheden:

a. Bij het intypen van een 1 neemt het programma aan dat het met de initiele waarden moet beginnen zoals die gegeven zijn in datafile PASCAL:INIT.DAT.TEXT.

b. Indien er al aan een serie gegevens is gerekend, maar het programma is afgebroken, kan men de berekening door laten gaan waar hij was afgebroken. De data zijn dan weggeschreven naar en worden weer ingelezen van datafile PASCAL:TEMP.DATA. Dit kan worden gedaan door een 2 in te typen.

c. Als er geen optimalisatie meer wordt gevraagd, maar alleen een nieuwe plot, moet een 3 worden ingetypt.

Bij zowel de akties a en b wordt op het beeld telkens als de lijnzoekprocedure wordt aangeroepen de waarde van de som van de kwadraten van de afwijkingen op het beeld gegeven. Dit is alleen bedoeld om de gebruiker inzicht te geven in de voortgang van het proces. Na afloop van de berekeningen worden de variabelen zowel op papier als op het beeldscherm afgedrukt. Ook de tijd die is verstreken tussen het starten van het programma en het eindigen van het programma wordt aangegeven. Deze tijd is uiteraard sterk afhankelijk van de initiele schatting van de variabelen.

## 6. VOORBEELDEN

In voorbeeld 1 is uitgegaan van een drietal lijnstukken. Van deze lijnstukken zijn 20 punten genomen. Vervolgens is hierop een ruis aangebracht, zodat de waarden niet meer exact op de lijnstukken liggen. Door deze waarden moeten nu de lijnstukken worden terugberekend. De waarden van de punten waardoor de lijnstukken moeten worden berekend zijn gegeven in tabel 1. De startwaarden van de variabelen zijn gegeven in tabel 2. Dit zijn tabellen zoals ze in de computer zijn

ingevoerd, en dienen dus tevens als voorbeeld voor de input. Het resultaat van de berekeningen is gegeven in fig. 5. Rekentijd voor dit probleem bedroeg circa 7 minuten inclusief uitprinten.

Tabel 1. a en b waarden van voorbeeld 1

```
 0.0
 7.0
 5.0
 6.0
 6.0
 6.0
10.0
 7.0
```

Tabel 2. Startwaarden voor de te optimaliseren parameters in voorbeeld 1

```
20
0.5  5.0
1.0  5.1
1.5  4.9
2.0  5.5
2.5  5.5
3.0  4.9
3.5  4.4
4.0  4.1
4.5  3.2
5.0  3.1
5.5  2.1
6.0  2.9
6.5  1.9
7.0  2.3
7.5  2.0
8.0  2.0
8.5  2.0
9.0  1.8
9.5  2.0
10.  2.2
```

```
      STARTWAARDEN                                          OPTIMALE WAARDEN
R:      0.00      5.00      6.00     10.00          R:      0.00      2.43      6.24     10.00
S:      7.00      6.00      6.00      7.00          S:      4.88      5.37      2.02      2.03

DE VERGELIJKINGEN VAN DE LIJNSTUKKEN ZIJN:
        S=    0.20 * R +    4.88        ALS    0.00 <= R <=   2.43
        S=   -0.88 * R +    7.51        ALS    2.43 <= R <=   6.24
        S=    0.00 * R +    2.01        ALS    6.24 <= R <=  10.00

SOM V.D. KWADRATEN V.D. AFWIJKINGEN=    1.248
```

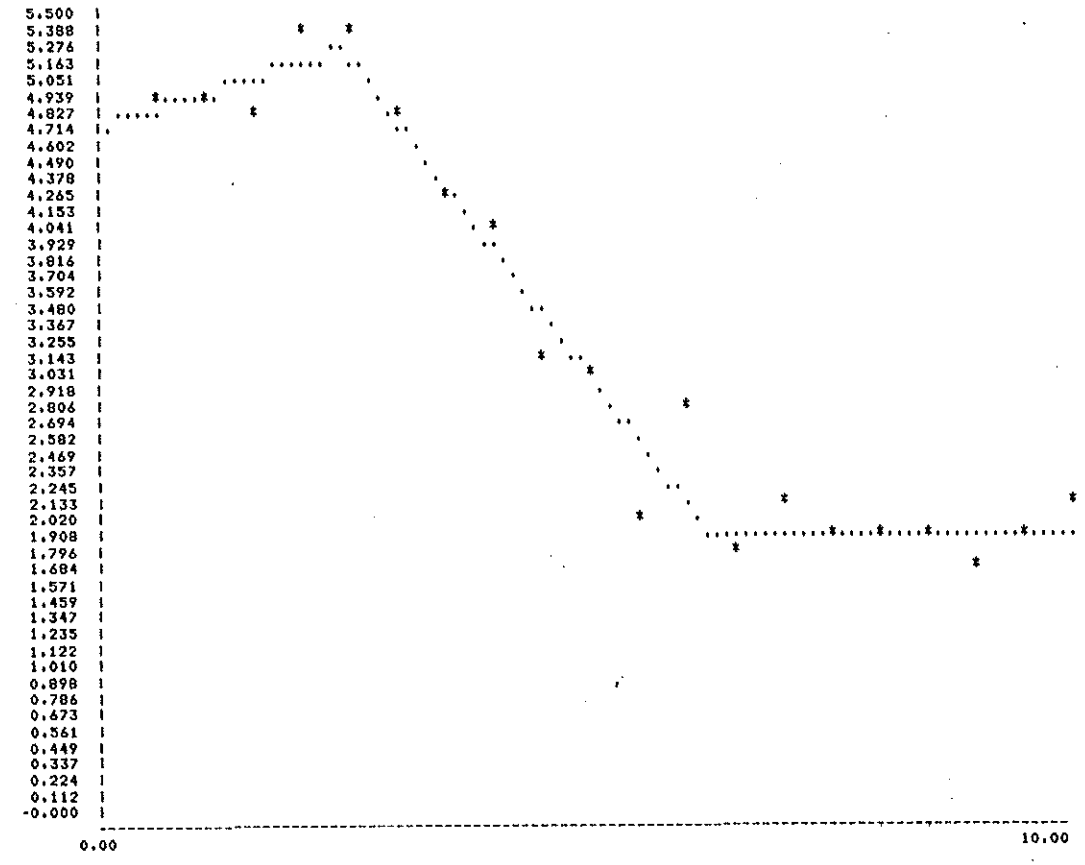


Fig. 5. Resultaat berekeningen voorbeeld 1

Voorbeeld 2 is afgeleid uit een eerdere publikatie (ALIVERTI and WESSELING, 1979) en berekent drie lijnstukken door 15 punten van een k-$\psi$ relatie. De waarden van de punten zijn overgenomen uit MUALEM, 1976, pag. 7 voor sticky clay. De 'conductivities' zijn gegeven als 'relative conductivities', zodat ze eerst moesten worden vermenigvuldigd met de 'saturated conductivity', die als eerste in de datafile voorkomt. Zie tabel 3. Daar deze relaties meestal logaritmisch de beste benadering van een rechte lijn geven, is in het programma direkt na het inlezen van de waarden de logaritme genomen van zowel k als $\psi$. De startwaarden voor de parameters zijn gegeven in tabel 4. Het resultaat is te zien in fig. 6. Bij dit resultaat moet men wel goed in de gaten houden dat de resultaten als logaritmes gegeven zijn, ook de vergelijkingen van de lijnen.

Tabel 3. Punten van k-$\psi$ relatie zoals gebruikt in voorbeeld 2 (naar

---

```
15
0.0219
1.00E+00 1.00E+00
2.50E+00 9.44E-01
4.60E+00 8.88E-01
1.00E+01 6.11E-01
2.12E+01 2.77E-01
3.90E+01 1.28E-01
5.40E+01 5.88E-02
8.90E+01 2.77E-02
1.00E+02 2.27E-02
1.70E+02 1.14E-02
2.80E+02 5.97E-03
4.90E+02 2.77E-03
1.00E+03 1.11E-03
2.80E+03 2.77E-04
1.00E+04 5.13E-05
```

Tabel 4. Startwaarden voor te optimaliseren s- en r-waarden

| |
|---|
| 0,0 |
| 0,0 |
| 1,0 |
| -3,0 |
| 3,0 |
| -5,0 |
| 4,0 |
| -7,0 |

Na het terugrekenen van de logaritmes ontstaan de volgende vergelijkingen:

$k = 0.03 \cdot \psi^{-0.03}$ voor $1 \leq \psi \leq 12,3$

$k = 0.45 \cdot \psi^{-1.44}$ voor $12.3 \leq \psi \leq 1350$

$k = 0,10 \cdot \psi^{-1.24}$ voor $1350 \leq \psi \leq 10000$

waarin $\psi$ in cm en k in cm/dag.

Met opzet is deze omrekening niet in het programma gedaan. Het programma gaat uit van lieaire schalen. In de procedure VARPRI is dit echter op zeer eenvoudige wijze in te brengen.

```
   STARTWAARDEN                                          OPTIMALE WAARDEN
R:      0.00      1.00      3.00      4.00          R:      0.00      1.09      3.13      4.00
S:      0.00     -3.00     -5.00     -7.00          S:     -1.59     -1.92     -4.86     -5.94

DE VERGELIJKINGEN VAN DE LIJNSTUKKEN ZIJN:
          S=   -0.30 * R +   -1.59          ALS    0.00 <= R <=   1.09
          S=   -1.44 * R +   -0.35          ALS    1.09 <= R <=   3.13
          S=   -1.24 * R +   -0.98          ALS    3.13 <= R <=   4.00

SOM V.D. KWADRATEN V.D. AFWIJKINGEN=    0.039
```

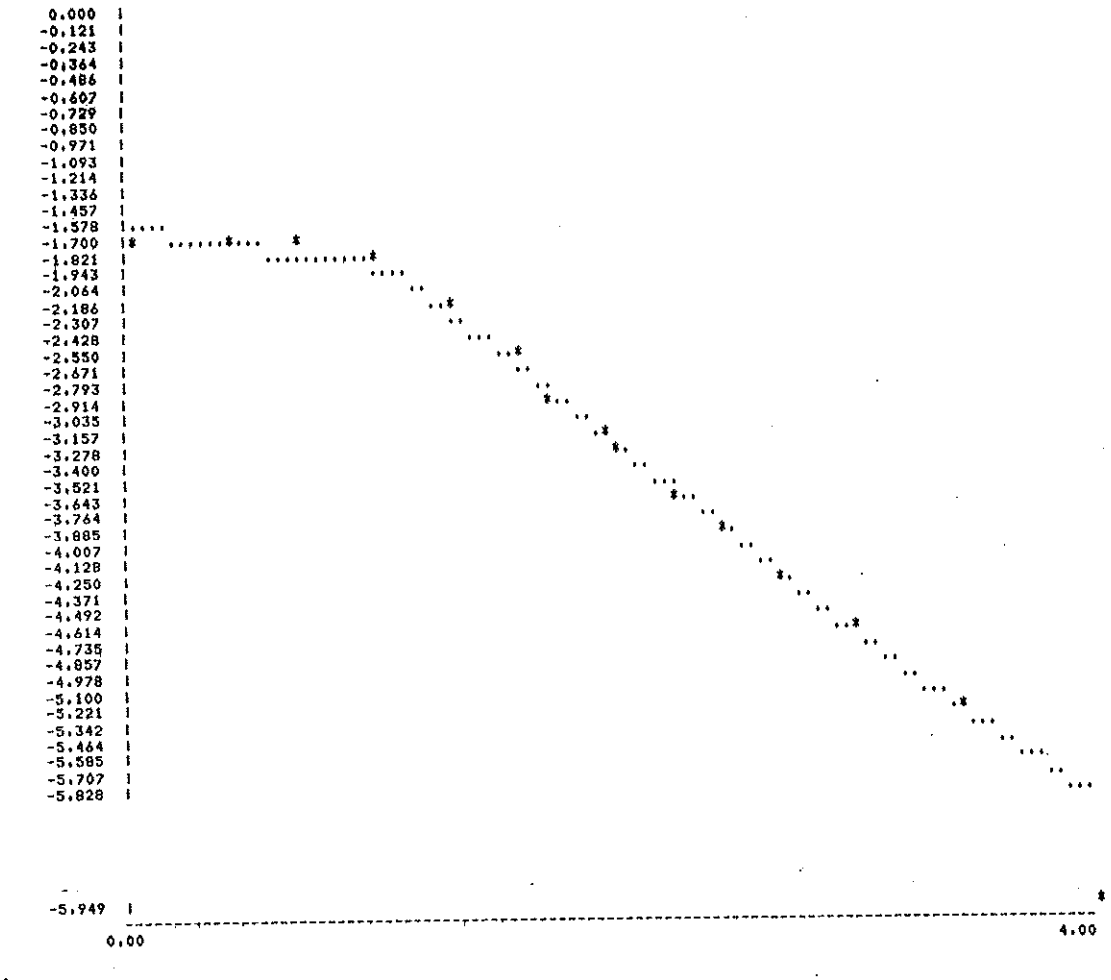


Fig. 6, Resultaat van de berekeningen uit voorbeeld 2

## 7. SLOTOPMERKINGEN

Natuurlijk is het ook mogelijk bovenbeschreven optimalisatie uit te voeren met het programmapakket OPTPAC van de Landbouwhogeschool (ALIVERTI and WESSELING, 1979). Het is mij echter gebleken dat dit pakket vrij duur in het gebruik is en bovendien de uitkomsten van de programma's sterk afhangen van de startwaarden van de variabelen. Naar mijn mening is dit bij de methode van Fletcher aanzienlijk minder het geval.

Ook heeft niet iedere computergebruiker de beschikking over OPTPAC, terwijl het programma FLETCH op iedere computer gedraaid kan worden die een Pascal-compiler heeft, zij het misschien met enkele kleine veranderingen.

Appendix A.

LISTING VAN HET COMPUTERPROGRAMMA FLETCH

```
1   1   1:0    1 (*$L PASCAL:PR.NOTA.TEXT*)
2   1   1:0    1 PROGRAM FLETCH;
3   1   1:0    3 (*****************************************************************)
4   1   1:0    3 (* PROGRAMMA VOOR HET BEREKENEN VAN DRIE LIJNSTUKKEN DOOR EEN  *)
5   1   1:0    3 (* SERIE PUNTEN. ER WORDT GEBRUIK GEMAAKT VAN DE METHODE VAN   *)
6   1   1:0    3 (* FLETCHER-REEVES.                                            *)
7   1   1:0    3 (* HET PROGRAMMA IS GESCHREVEN IN U.C.S.D. PASCAL VOOR ZOWEL   *)
8   1   1:0    3 (* DE HEATHKIT H-11 ALS DE PDP-11/03 COMPUTER.                 *)
9   1   1:0    3 (*=============================================================*)
10  1   1:0    3 (*      IR.J.G.WESSELING.                                      *)
11  1   1:0    3 (*      INSTITUUT VOOR CULTUURTECHNIEK EN WATERHUISHOUDING.    *)
12  1   1:0    3 (*      POSTBUS 35.                                            *)
13  1   1:0    3 (*      6700 AA WAGENINGEN.                                    *)
14  1   1:0    3 (*=============================================================*)
15  1   1:0    3 (* 30 NOV. 1981                                                *)
16  1   1:0    3 (*****************************************************************)
17  1   1:0    3 TYPE PARVEC=ARRAY[0..7] OF REAL;
18  1   1:0    3      DATAVEC=ARRAY[1..50] OF REAL;
19  1   1:0    3 VAR  XST,XB,X,XDB:PARVEC;
20  1   1:0   67      A,B: DATAVEC;
21  1   1:0  267      FUNCTIE,WIS:REAL;
22  1   1:0  271      I,ANTW,NUMPAR,NUMDAT,TIMST1,TIMST2: INTEGER;
23  1   1:0  277      LEES,PRINT:INTERACTIVE;
24  1   1:0  879
25  1   2:0    1 PROCEDURE TIMER(TIMST1,TIMST2:INTEGER);
26  1   2:0    3 (* PROCEDURE VOOR BEREKENEN EN UITSCHRYVEN VAN TIJDSDUUR *)
27  1   2:0    3 (* DIE GEREKEND IS. IN HET HOOFDPROGRAMMA MOETEN BIJ      *)
28  1   2:0    3 (* HET BEGIN VAN DE BEREKENINGEN TIMST1 EN TIMST2         *)
29  1   2:0    3 (* BEPAALD WORDEN MET BEHULP VAN  DE STANDAARDPROCEDURE   *)
30  1   2:0    3 (* TIME.                                                  *)
31  1   2:0    3 VAR  TIMEND1,TIMEND2,TIMDIF,TICKS,SECNDS,MINS:INTEGER;
32  1   2:0    0 BEGIN
33  1   2:1    0    TIME(TIMEND1,TIMEND2);
34  1   2:1    6    TIMDIF:=TIMEND2 - TIMST2;
35  1   2:1   11    TICKS := 2*(TIMDIF MOD 50);
36  1   2:1   18    TIMDIF := TIMDIF DIV 50;
37  1   2:1   23    SECNDS := TIMDIF MOD 60;
38  1   2:1   28    MINS := TIMDIF DIV 60;
39  1   2:1   33    WRITELN;
40  1   2:1   41    WRITELN('EXECUTION TIME: ',MINS:3,':',SECNDS:2,'.',TICKS:2);
41  1   2:1  127    WRITELN;
42  1   2:0  135 END;(*OF TIMER*)
43  1   2:0  148
44  1   3:0    1 PROCEDURE AFGEL(VAR AFG1,AFG2,AFG3:REAL; X:PARVEC);
45  1   3:0   21 (* PROCEDURE VOOR HET BEPALEN VAN DE HELLINGSHOEK VAN DE *)
46  1   3:0   21 (* LIJNSTUKKEN BEPAALD DOOR DE VARIABELEN X.             *)
47  1   3:0    0 BEGIN
48  1   3:1    0      AFG1:=(X[3]-X[1])/(X[2]-X[0]);
49  1   3:1   51      AFG2:=(X[5]-X[3])/(X[4]-X[2]);
50  1   3:1   97      AFG3:=(X[6]-X[5])/(X[7]-X[4]);
51  1   3:0  143 END;
52  1   3:0  156
53  1   4:0    3 FUNCTION FUN(VAR X:PARVEC):REAL;
54  1   4:0    4 (* FUNCTION VOOR HET BEREKENEN VAN DE SOM VAN DE KWADRATEN    *)
55  1   4:0    4 (* VAN DE AFWIJKINGEN. DE VARIABELEN X ZIJN DE TE             *)
56  1   4:0    4 (* OPTIMALISEREN PARAMETERS.                                  *)
57  1   4:0    4 VAR  I:INTEGER;
58  1   4:0    5      REK,AFG1,AFG2,AFG3,VERSCH,FUNTEM:REAL;
59  1   4:0    0 BEGIN
60  1   4:1    0       IF (X[2]-X[0]>0.01) AND (X[4]-X[2]>0.01) AND (X[7]-X[4]>0.01)
```

```
 61   1    4:1    84       THEN
 62   1    4:2    87      BEGIN (*TOEGESTANE WAARDEN*)
 63   1    4:3    87           AFGEL(AFG1,AFG2,AFG3,X);
 64   1    4:3    96           FUNTEM:=0.0;
 65   1    4:3   106           FOR I:=1 TO NUMDAT DO
 66   1    4:4   120          BEGIN
 67   1    4:5   120            IF A[I] < X[2]
 68   1    4:5   140            THEN
 69   1    4:6   146            REK:=X[1]+AFG1*(A[I]-X[0])
 70   1    4:5   184            ELSE
 71   1    4:6   190            IF (A[I]>=X[2]) AND (A[I]<X[4])
 72   1    4:6   238              THEN
 73   1    4:7   241              REK:=X[3]+AFG2*(A[I]-X[2])
 74   1    4:6   279              ELSE
 75   1    4:7   285              IF A[I]>= X[4]
 76   1    4:7   305                THEN
 77   1    4:8   311                REK:=X[5]+AFG3*(A[I]-X[4]);
 78   1    4:5   353                VERSCH:=REK-B[I];
 79   1    4:5   374                VERSCH:=VERSCH*VERSCH;
 80   1    4:5   387                FUNTEM:=FUNTEM+VERSCH;
 81   1    4:4   400          END;
 82   1    4:3   407        FUN:=FUNTEM;
 83   1    4:2   415       END
 84   1    4:1   415        ELSE (*NIET TOEGESTANE WAARDEN*)
 85   1    4:2   417        FUN:=100000.0;
 86   1    4:0   428 END;(*OF FUN*)
 87   1    4:0   448
 88   1    4:0   448
 89   1    5:0     1 PROCEDURE COMPARE(VAR MIN,TEST:PARVEC; VAR FO,FN:REAL);
 90   1    5:0     5 (* IN COMPARE WORDT VOOR EEN PARAMETERVECTOR TEST GEKEKEN OF DEZE *)
 91   1    5:0     5 (* BETERE RESULTATEN OPLEVERT DAN HET OUDE MINIMUMPUNT. INDIEN    *)
 92   1    5:0     5 (* DIT ZO IS, WORDT DE VECTOR TEST DE NIEUWE MINIMUMVECTOR.       *)
 93   1    5:0     5 VAR PAR:INTEGER;
 94   1    5:0     0 BEGIN
 95   1    5:1     0      FN:=FUN(TEST);
 96   1    5:1     8      IF FN < FO THEN
 97   1    5:2    18    BEGIN
 98   1    5:3    18         FO:=FN;
 99   1    5:3    24         FOR PAR:=0 TO 7 DO MIN[PAR]:=TEST[PAR];
100   1    5:2    60    END;
101   1    5:0    60 END;(*OF COMPARE*)
102   1    5:0    74
103   1    5:0    74
104   1    6:0     1 PROCEDURE LIJNMIN(VAR X,XN,DIR:PARVEC);
105   1    6:0     4 (* IN PROCEDURE BASSTAP WORDT IN DE RICHTING DIR              *)
106   1    6:0     4 (* EEN NIEUW MINIMUM VAN DE FUNCTIE GEZOCHT. XN   IS HET     *)
107   1    6:0     4 (* NIEUWE MINIMUM, DAT GEVONDEN IS.                          *)
108   1    6:0     4 VAR  I,J:INTEGER;
109   1    6:0     6     TEST:BOOLEAN;
110   1    6:0     7     FO,FN,FH,STEP,FAC:REAL;
111   1    6:0    17     XH,DIRMUL:PARVEC;
112   1    6:0     0 BEGIN
113   1    6:1     0   FOR I:=0 TO 7 DO XN[I]:=X[I];
114   1    6:1    37   FO:=FUN(XN);
115   1    6:1    46   STEP:=100.0;
116   1    6:1    56   FAC:=1.0;
117   1    6:1    66   WHILE STEP > 0.001 DO
118   1    6:2    80  BEGIN
119   1    6:3    80     STEP:=0.1*STEP;
120   1    6:3    95     FOR I:=0 TO 7 DO DIRMUL[I]:=STEP*DIR[I];
```

```
121   1    6:3   138     TEST:=FALSE;
122   1    6:3   141   REPEAT
123   1    6:4   141     BEGIN
124   1    6:5   141       FOR I:=0 TO 7 DO XH[I]:=XN[I]+FAC*DIRMUL[I];
125   1    6:5   195       FH:=FO;
126   1    6:5   203       COMPARE(XN,XH,FO,FN);
127   1    6:5   212       IF ABS(FH-FO)<0.0001 THEN
128   1    6:6   232      BEGIN
129   1    6:7   232          IF FAC < 0.0 THEN
130   1    6:8   246        BEGIN
131   1    6:9   246            FAC:=1.0;
132   1    6:9   256            TEST:=TRUE;
133   1    6:8   259        END
134   1    6:7   259       ELSE
135   1    6:8   261         FAC:=-1.0;
136   1    6:6   273      END;
137   1    6:4   273    END;
138   1    6:3   273    UNTIL TEST;
139   1    6:2   276  END;
140   1    6:1   278  WRITELN('AAN EINDE LIJNZOEKPROCEDURE F=',FO:10:4);
141   1    6:0   341 END;(*OF LIJNMIN*)
142   1    6:0   366
143   1    8:0     1 PROCEDURE DERIV(VAR X,AF:PARVEC);
144   1    8:0     3 (*NUMERIEKE BEREKENING VAN AFGELEIDENVECTOR*)
145   1    8:0     3 VAR  XP,XM:PARVEC;
146   1    8:0    35      F1,F2,DXL,DXH,XH,XL:REAL;
147   1    8:0    47      I,J:INTEGER;
148   1    8:0     0 BEGIN
149   1    8:1     0      XM[0]:=X[0];
150   1    8:1    19      XM[7]:=X[7];
151   1    8:1    38      XP[0]:=X[0];
152   1    8:1    57      XP[7]:=X[7];
153   1    8:1    76      AF[0]:=0.0;
154   1    8:1    92      AF[7]:=0.0;
155   1    8:1   108      FOR I:=1 TO 6 DO
156   1    8:2   121    BEGIN
157   1    8:3   121        FOR J:=1 TO 6 DO
158   1    8:4   134      BEGIN
159   1    8:5   134          IF J<>I THEN
160   1    8:6   141        BEGIN
161   1    8:7   141            XM[J]:=X[J];
162   1    8:7   162            XP[J]:=X[J];
163   1    8:6   183        END
164   1    8:5   183      ELSE
165   1    8:6   185        BEGIN
166   1    8:7   185            XM[J]:=X[J]-0.1;
167   1    8:7   213            XP[J]:=X[J]+0.1;
168   1    8:6   241        END;
169   1    8:4   241      END;
170   1    8:3   249        XL:=FUN(XM);
171   1    8:3   259        DXL:=0.1;
172   1    8:3   270        IF XL>9999.9 THEN
173   1    8:4   284      BEGIN
174   1    8:5   284          XM[I]:=X[I];
175   1    8:5   305          XL:=FUN(XM);
176   1    8:5   315          DXL:=0.0;
177   1    8:4   326      END;
178   1    8:3   326        XH:=FUN(XP);
179   1    8:3   336        DXH:=0.1;
180   1    8:3   346        IF XH>9999.9 THEN
```

```
181   1    8:1   360     BEGIN
182   1    8:5   360         XF[I]:=X[I];
183   1    8:5   381         XH:=FUN(X);
184   1    8:5   390         DXH:=0.0;
185   1    8:1   400     END;
186   1    8:3   400       IF ABS(DXL+DXH)>0.001 THEN AF[I]:=(XH-XL)/(DXH+DXL) ELSE
187   1    8:4   451       WRITELN(' KAN GEEN AFGELEIDEN NEMEN, DUS!');
188   1    8:2   503     END;
189   1    8:0   511 END;
190   1    8:0   530
191   1    9:0     1 PROCEDURE OPT(VAR X:FARVEC);
192   1    9:0     2 (*PROCEDURE VOOR HET REGELEN VAN DE OPTIMALISATIE*)
193   1    9:0     2 VAR  I,TEL,NTOT:INTEGER;
194   1    9:0     5      TEST,TESTKL:BOOLEAN;
195   1    9:0     7      XN,D,AF,AFN,XTEST,XTK:FARVEC;
196   1    9:0   103      SCHRYF:INTERACTIVE;
197   1    9:0   404      BETA,S,SN:REAL;
198   1    9:0     0 BEGIN
199   1    9:1     0      NTOT:=10;
200   1    9:1    12      TEST:=TRUE;
201   1    9:1    15      FOR I:=0 TO 7 DO
202   1    9:2    29    BEGIN
203   1    9:3    29        XTEST[I]:=X[I];
204   1    9:3    48        XTK[I]:=X[I];
205   1    9:2    67    END;
206   1    9:1    74      WHILE TEST DO
207   1    9:2    77   BEGIN
208   1    9:3    77       FOR I:=0 TO 7 DO XTK[I]:=XN[I];
209   1    9:3   118       DERIV(X,AF);
210   1    9:3   123       FOR I:=0 TO 7 DO D[I]:=-AF[I];
211   1    9:3   165       TEL:=0;
212   1    9:3   168       TESTKL:=TRUE;
213   1    9:3   171       WHILE (TESTKL) AND (TEL<NTOT) DO
214   1    9:4   178     BEGIN
215   1    9:5   178         TEL:=TEL+1;
216   1    9:5   183         LIJNMIN(X,XN,D);
217   1    9:5   190         IF TEL<NTOT THEN
218   1    9:6   195       BEGIN
219   1    9:7   195           DERIV(XN,AFN);
220   1    9:7   201           S:=0.0;
221   1    9:7   212           SN:=0.0;
222   1    9:7   224           TESTKL:=FALSE;
223   1    9:7   227           FOR I:=1 TO 6 DO
224   1    9:8   241         BEGIN
225   1    9:9   241             IF ABS(X[I]-XTK[I]) > 0.1 THEN TESTKL:=TRUE;
226   1    9:9   275             S:=S+AF[I]*AF[I];
227   1    9:9   307             SN:=SN+AFN[I]*AFN[I];
228   1    9:9   339             AF[I]:=AFN[I];
229   1    9:9   359             X[I]:=XN[I];
230   1    9:9   378             XTK[I]:=XN[I];
231   1    9:8   398         END;
232   1    9:7   405           BETA:=SN/S;
233   1    9:7   421           FOR I:=0 TO 7 DO D[I]:=-AFN[I]+BETA*D[I];
234   1    9:6   480       END;
235   1    9:4   480     END;
236   1    9:3   482       TEST:=FALSE;
237   1    9:3   485       FOR I:=0 TO 7 DO
238   1    9:4   499     BEGIN
239   1    9:5   499         IF ABS(XN[I]-XTEST[I])>0.01 THEN TEST:=TRUE;
240   1    9:5   535       XTEST[I]:=XN[I];
```

```
241   1     9:4    555    END;
242   1     9:3    562       FOR I:=0 TO 7 DO X[I]:=XN[I];
243   1     9:3    602       REWRITE(SCHRYF,'PASCAL:TEMP:DATA');
244   1     9:3    630       FOR I:=0 TO 7 DO WRITELN(SCHRYF,XN[I]);(*BEWAAR RESULTATEN*)
245   1     9:3    676       CLOSE(SCHRYF,LOCK);
246   1     9:2    684   END;
247   1     7:0    686 END;
248   1     9:0    732
249   1    10:D      1 PROCEDURE VARPRI(XST,X:PARVEC);
250   1    10:D     35 (*UITVOER VAN GEGEVENS*)
251   1    10:D     35 VAR H,AFG1,AFG2,AFG3:REAL;
252   1    10:D     43     TEL:INTEGER;
253   1    10:0      0 BEGIN
254   1    10:1      0     WRITE(PRINT,'     STARTWAARDEN');
255   1    10:1     39     FOR TEL:=1 TO 48 DO WRITE(PRINT,' ');
256   1    10:1     70     WRITELN(PRINT,'OPTIMALE WAARDEN');
257   1    10:1    106     WRITE(PRINT,'R:',XST[0]:10:2,XST[2]:10:2,XST[4]:10:2,XST[7]:10:2);
258   1    10:1    196     FOR TEL:=1 TO 18 DO WRITE(PRINT,' ');
259   1    10:1    227     WRITELN(PRINT,'R:',X[0]:10:2,X[2]:10:2,X[4]:10:2,X[7]:10:2);
260   1    10:1    325     WRITE(PRINT,'S:',XST[1]:10:2,XST[3]:10:2,XST[5]:10:2,XST[6]:10:2);
261   1    10:1    415     FOR TEL:=1 TO 18 DO WRITE(PRINT,' ');
262   1    10:1    446     WRITELN(PRINT,'S:',X[1]:10:2,X[3]:10:2,X[5]:10:2,X[6]:10:2);
263   1    10:1    544     WRITELN(PRINT);
264   1    10:1    552     AFGEL(AFG1,AFG2,AFG3,X);
265   1    10:1    562     H:=X[1]-AFG1*X[0];
266   1    10:1    592     WRITELN(PRINT,'DE VERGELIJKINGEN VAN DE LIJNSTUKKEN ZIJN:');
267   1    10:1    654     WRITE(PRINT,'           S=',AFG1:8:2,' * R +',H:8:2);
268   1    10:1    722     WRITELN(PRINT,'              ALS',X[0]:8:2,' <= R <=',X[2]:7:2);
269   1    10:1    815     H:=X[3]-AFG2*X[2];
270   1    10:1    845     WRITE(PRINT,'           S=',AFG2:8:2,' * R +',H:8:2);
271   1    10:1    913     WRITELN(PRINT,'              ALS',X[2]:8:2,' <= R <=',X[4]:7:2);
272   1    10:1   1006     H:=X[5]-AFG3*X[4];
273   1    10:1   1036     WRITE(PRINT,'           S=',AFG3:8:2,' * R +',H:8:2);
274   1    10:1   1104     WRITELN(PRINT,'              ALS',X[4]:8:2,' <= R <=',X[7]:7:2);
275   1    10:0   1197 END;
276   1    10:0   1216
277   1    11:D      1 PROCEDURE XAXIS(XMIN,XMAX:REAL);
278   1    11:D      5 (*TEKENEN VAN X-AS*)
279   1    11:D      5 VAR J:INTEGER;
280   1    11:0      0 BEGIN
281   1    11:1      0     FOR J:=1 TO 10 DO WRITE(PRINT,' ');
282   1    11:1     28     FOR J:=11 TO 111 DO WRITE(PRINT,'-');
283   1    11:1     56     WRITELN(PRINT);
284   1    11:1     64     WRITE(PRINT,XMIN:12:2);
285   1    11:1     77     FOR J:=13 TO 105 DO WRITE(PRINT,' ');
286   1    11:1    105     WRITELN(PRINT,XMAX:6:2);
287   1    11:0    126 END;(*OF XAXIS*)
288   1    11:0    144
289   1    12:D      1 PROCEDURE PRPLOT(VAR A,B:DATAVEC; X:PARVEC; NUMDAT:INTEGER);
290   1    12:D     21 (* PROCEDURE OM EEN PLOT VAN DE RESULTATEN TE MAKEN OP DE PRINTER*)
291   1    12:D     21 VAR PLOT:ARRAY[0..100] OF CHAR;
292   1    12:D    122     XPLD,YPLD:ARRAY[1..50] OF INTEGER;
293   1    12:D    222     YPLR:ARRAY[0..100]OF INTEGER;
294   1    12:D    323     XV,AFG1,AFG2,AFG3,XMIN,XMAX,YMIN,YMAX,STAPX,STAPY,Y:REAL;
295   1    12:D    345     DPL,I,J,PLAATS:INTEGER;
296   1    12:0      0 BEGIN
297   1    12:1      0      XMIN:=0.0;
298   1    12:1     16      XMAX:=X[7];
299   1    12:1     31      YMIN:=0.0;
300   1    12:1     42      YMAX:=0.0;
```

```
301   1   12:1    54      FOR I:=1 TO NUMDAT DO          (* BEPAAL MAX EN MIN*)
302   1   12:2    71    BEGIN
303   1   12:3    71        IF A[I]>XMAX THEN XMAX:=A[I];
304   1   12:3   111        IF A[I]<XMIN THEN XMIN:=A[I];
305   1   12:3   151        IF B[I]>YMAX THEN YMAX:=B[I];
306   1   12:3   191        IF B[I]<YMIN THEN YMIN:=B[I];
307   1   12:2   231    END;
308   1   12:1   241       STAPX:=(XMAX-XMIN)/100.0;       (* STAPGROOTTE X-AS*)
309   1   12:1   265       STAPY:=(YMAX-YMIN)/49.0;        (* STAPGROOTTE Y-AS*)
310   1   12:1   289       IF YMIN < 0.0 THEN DPL:=ROUND(ABS(YMIN)/STAPY)+1
311   1   12:1   318                     ELSE DPL:=0;
312   1   12:1   329       FOR I:=1 TO NUMDAT DO          (* POSITIES DATA *)
313   1   12:2   346    BEGIN
314   1   12:3   346        XPLD[I]:=ROUND(A[I]/STAPX);
315   1   12:3   381        YPLD[I]:=ROUND(B[I]/STAPY)+DPL;
316   1   12:2   419    END;
317   1   12:1   429      AFGEL(AFG1,AFG2,AFG3,X);          (* AFGELEIDEN LIJNEN *)
318   1   12:1   442       XV:=XMIN-STAPX;
319   1   12:1   458       FOR I:=0 TO 100 DO               (* BEREKEN LIJNPUNTEN *)
320   1   12:2   475     BEGIN
321   1   12:3   475         XV:=XV+STAPX;
322   1   12:3   491         IF XV < X[2]
323   1   12:3   504         THEN
324   1   12:4   510           YPLR[I]:=ROUND((X[1]+AFG1*(XV-X[0]))/STAPY)+DPL
325   1   12:3   562         ELSE
326   1   12:4   569           IF (XV>=X[2]) AND (XV<X[4])
327   1   12:4   603           THEN
328   1   12:5   606             YPLR[I]:=ROUND((X[3]+AFG2*(XV-X[2]))/STAPY)+DPL
329   1   12:4   658           ELSE
330   1   12:5   665             IF XV>= X[4]
331   1   12:5   678             THEN
332   1   12:6   684               YPLR[I]:=ROUND((X[5]+AFG3*(XV-X[4]))/STAPY)+DPL
333   1   12:2   736     END;
334   1   12:1   751     Y:=YMAX + STAPY;
335   1   12:1   767     FOR I:=50 DOWNTO 1 DO                    (* DE Y-WAARDEN*)
336   1   12:2   784   BEGIN
337   1   12:3   784       Y:=Y-STAPY;
338   1   12:3   800       FOR J:=0 TO 100 DO PLOT[J]:=' ';       (* MAAK REGEL SCHOON *)
339   1   12:3   839       WRITE(PRINT,Y:8:3,'  I');
340   1   12:3   868       FOR J:=0 TO 100 DO
341   1   12:4   885         IF YPLR[J]=I THEN PLOT[J]:=',';      (* LIJN *)
342   1   12:3   925       FOR J:=1 TO NUMDAT DO
343   1   12:4   942         IF YPLD[J]=I THEN
344   1   12:5   961         BEGIN
345   1   12:6   961           PLAATS:=XPLD[J];
346   1   12:6   978           PLOT[PLAATS]:='*';
347   1   12:5   990         END;
348   1   12:3  1000       FOR J:=0 TO 100 DO
349   1   12:4  1017      BEGIN
350   1   12:5  1017        WRITE(PRINT,PLOT[J]);       (* UITVOER *)
351   1   12:4  1037       END;
352   1   12:3  1047       WRITELN(PRINT);
353   1   12:2  1055     END;
354   1   12:2  1065
355   1   12:1  1065     XAXIS(XMIN,XMAX);               (* AS *)
356   1   12:0  1077 END;(*OF PRPLOT*)
357   1   12:0  1114
358   1    1:0     0 BEGIN (*OF MAIN PROGRAM*)
359   1    1:1     0   TIME(TIMST1,TIMST2);
360   1    1:1    31   REWRITE(PRINT,'PRINTER:');
```

```
361   1     1:1    52   NUMPAR:=6;
362   1     1:1    56   RESET(LEES,'PASCAL:OPTIM.DAT.TEXT');
363   1     1:1    90   READLN(LEES,NUMDAT);                          (*LEES AANTAL DATAPAREN*)
364   1     1:1   109   FOR I:=1 TO NUMDAT DO READLN(LEES,A[I],B[I]);   (*LEES DATA*)
365   1     1:1   185   CLOSE(LEES,LOCK);
366   1     1:1   194   WRITE('BEGINNEN(1), DOORGAAN(2) OF PLOTTEN(3)? '); (*WELKE AKTIE?*)
367   1     1:1   246   READLN(ANTW);
368   1     1:1   265   IF ANTW = 1 THEN RESET(LEES,'PASCAL:INIT.DAT.TEXT')
369   1     1:1   305                ELSE RESET(LEES,'PASCAL:TEMP.DATA');
370   1     1:1   336   FOR I:=0 TO 7 DO READLN(LEES,X[I]);
371   1     1:1   388   CLOSE(LEES,LOCK);
372   1     1:1   397   IF ANTW=1 THEN
373   1     1:2   404 BEGIN
374   1     1:3   404      WIS:=X[7];
375   1     1:3   419      X[7]:=X[6];
376   1     1:3   439      X[6]:=WIS;
377   1     1:2   454 END;
378   1     1:1   454   FOR I:=0 TO 7 DO XST[I]:=X[I];
379   1     1:1   505   IF ANTW <> 3 THEN
380   1     1:2   512 BEGIN
381   1     1:3   512      OPT(X);                                     (*OPTIMALISATIE*)
382   1     1:2   516 END;
383   1     1:1   516   VARPRI(XST,X);
384   1     1:1   522   WRITELN(PRINT);
385   1     1:1   530   FUNCTIE:=FUN(X);                             (*MINIMUM FUNCTIEWAARDE*)
386   1     1:1   541   WRITELN(PRINT,'SOM V.D. KWADRATEN V.D. AFWIJKINGEN=',FUNCTIE:9:3);
387   1     1:1   611   WRITELN(PRINT);
388   1     1:1   619   PRPLOT(A,B,X,NUMDAT);                        (*MAAK PLOT*)
389   1     1:1   631   WRITELN;
390   1     1:1   639   FOR I:=0 TO 7 DO WRITELN('X(',I:1,')=',X[I]:7:3);
391   1     1:1   735   WRITELN('F=',FUNCTIE);                       (*OUTPUT*)
392   1     1:1   771   TIMER(TIMST1,TIMST2);                        (*REKENTIJD*)
393   1     1:0   779 END.
```

Appendix B

WAAROM PASCAL?

Daar er, zover ik weet, op het ICW nog niet eerder een programma in de programmeertaal Pascal is verschenen (afgezien van enkele proefprogramma'tjes door verschillende mensen geschreven en vergeten), leek het mij nuttig even een korte toelichting te geven waarom ik Pascal heb gekozen voor dit programma.

Vanuit een zekere traditie, nog versterkt door het feit dat er pas enkele jaren andere programmeertalen te onzer beschikking staan, is het in Wageningen, en zeker op het ICW, gewoonte om computerprogramma's in FORTRAN te schrijven. Deze taal is ook bijzonder geschikt voor onze doeleinden. Maar al toen ik enkele jaren geleden voor het eerst kennis maakte met de Algol-achtige talen, ging mijn voorkeur uit naar deze laatste groep programmeertalen. Daar komt nog bij dat op Technische Hogescholen ongeveer 90% van het programmeerwerk in Algol wordt gedaan. Zodat we wel aan mogen nemen dat deze taal goed uitgetest is.

Toen ik op vrij goedkope wijze de hand kon leggen op een Pascalcompiler heb ik dan ook niet geaarzeld en ben met deze taal begonnen. Het grote voordeel van talen die tot de Algolachtigen behoren is het gebruik van een goed gestruktureerde opbouwmogelijkheid van de programma's. Het schijnt dat dit zelfs zo goed is bevallen dat de nieuwste FORTRAN compilers (FORTRAN V en FORTRAN 88) dit hebben overgenomen. Wat tot nu toe in FORTRAN-programma's altijd opviel, en waar ik me altijd zoveel mogelijk in heb beperkt, zijn de grote hoeveelheid GO TO statements. In de Algolachtige talen kan dit GO TO wel gebruikt worden, maar het wordt sterk afgeraden, en is ook niet nodig, indien een programma logisch wordt opgebouwd. Een volgend voordeel van bijvoorbeeld Pascal is de grote hoeveelheid commando's die ter beschikking staan voor het bouwen van lussen (WHILE-DO, REPEAT-UNTIL, FOR-UNTIL, etc.). In FORTRAN is men beperkt tot het gebruik van DO en GO TO.

Een volgend voordeel is, naar mijn mening, de blok opbouw van deze groep talen. Dit maakt het programmeren een stuk aantrekkelijker en de programma's overzichtelijker. Het is, mijns inziens,ook gemakkelijker te leren aan mensen die nog nooit met een computer hebben gewerkt.

In FORTRAN neemt de compiler automatisch aan dat variabelen waarvan de naam met een van de letters I t/m N begint integers zijn. Variabelen waarvan de naam met een andere letter begint zijn reals. In Pascal moet iedere te gebruiken variabele tevoren gedefinieerd worden. Dit lijkt in het begin lastig, maar in het gebruik is het gemakkelijk. Het gebeurt namelijk regelmatig dat men bijvoorbeeld in plaats van IN IM intypt, of wat nog moeilijker te onderscheiden is bij het even vlug doorkijken: IØ in plaats van IO. Bij dit soort dingen geeft de Pascal compiler een duidelijke foutmelding. Verder zal men op deze manier, als men eraan gewend is, zoveel mogelijk dezelfde variabele gebruiken. Dit spaart automatisch geheugenruimte en hoeft bij een juiste keuze van de naam het programma niet minder leesbaar te maken.

Een nadeel van Pascal is dat het geen COMMON-achtig statement kent. Alle variabelen die door moeten worden gegeven aan subroutines moeten òf in het hoofdprogramma al gedefiniëerd zijn, waarbij ze automatisch worden doorgegeven, òf ze moeten in de procedure aanhef worden gegeven. Een volgend nadeel van UCSD-Pascal versie II (hoe het met de andere versies zit weet ik niet) is dat het vrij lastig is om procedures extern van het hoofdprogramma te compileren. Hierbij komt vrij veel extra typewerk kijken dat nodig is om aan te geven dat de procedure extern is en dus variabelen ergens anders gedefinieerd kunnen zijn. Afhankelijk van de gebruiker kan men de in- en uitvoer methode van Pascal gemakkelijk of lastig vinden. Naar mijn mening is deze methode gemakkelijk. Het is iets omslachtiger dan in FORTRAN, maar biedt, naar mijn mening, een grotere vrijheid. Een laatste nadeel van Pascal is dat procedures niet onbeperkt lang mogen zijn. Dit in tegenstelling tot FORTRAN. De lengte van een procedure in Pascal is gebonden aan een maximum aantal woorden geheugenruimte. Hieruit volgt echter wel dat vaak veel kleine subprogramma'tjes zullen worden geschreven, wat de leesbaarheid van een programma wel bevordert.

Concluderend kan ik zeggen, dat het gebruik van Pascal zeker voordelen biedt. Uiteraard ook enkele nadelen, maar naar mijn mening wegen de voordelen ruim op tegen deze nadelen. Daarom zou ik dan ook willen aanbevelen dat men binnen het ICW meer tijd schenkt aan het gebruik van andere programmeertalen dan FORTRAN.

LITERATUUR

ALIVERTI-PIURI, E. and J.G. WESSELING, 1979. Calculation of height of capillary rise in non-homogeneous soil profiles. Nota ICW 1156, 46 pp.

BEEK, P. VAN en TH.H.B. HENDRIKS, 1978. Voortzetting Optimaliseringstechnieken II. Collegediktaat Landbouwhogeschool, 58 pp.

LUENBERGER, DAVID G., 1973. Introduction to linear and non-linear programming. Addison-Wesley Publishing Company, 305 pp.

MUALEM, Y., 1976. A catalogue of the hydraulic properties of unsaturated soils. Technion, Haifa. Development of methods, tools and solutions for unsaturated flow with application to watershed hydrology and other fields, 100 pp.

STOL, PH.TH., 1975. A contribution to theory and practice of nonlinear parameter optimization. Dissertatie L.H. 197 pp.

STUYT, L.C.P.M., 1980. Enige aspekten betreffende het optimaliseren van de parameters van een lineair 'conceptual model' voor simulatie van oppervlakte-afvoer. Doctoraalscriptie Landbouwhogeschool, Vakgroepen Hydraulica & Afvoerhydrologie en Wiskunde