Compensatie van niveauverstoringen van satelliet-bakensignalen

Rozendaal, J.

Award date:
1989

5535

FACULTEIT ELECTROTECHNIEK
TECHNISCHE UNIVERSITEIT EINDHOVEN
VAKGROEP TELECOMMUNICATIE

COMPENSATIE VAN NIVEAUVERSTORINGEN VAN SATELLIET-BAKENSIGNALEN

door

J. Rozendaal

Verslag van het afstudeerwerk
uitgevoerd van 1-11-1988 tot 8-8-1989
Afstudeerhoogleraar: Prof. dr. ir. G. Brussaard
Begeleider: Ir A. Mawira

SAMENVATTING

Bij het propagatie onderzoek met behulp van Olympus in de 20-30 GHz band wil men te weten komen wat de invloed is van de atmosfeer op de uitgezonden bakensignalen. Er zijn echter ook andere verschijnselen die invloed hebben op de signalen van deze satelliet. Om nu de invloed van de atmosfeer te kunnen meten, moeten de effecten van deze andere, verstorende, verschijnselen op de meetresultaten te niet worden gedaan. Dit verslag beschrijft een methode om de effecten van dit soort verstoringen te compenseren.

Eerst wordt er een inventarisatie gemaakt van alle verschijnselen die van invloed zijn op de metingen van de satellietsignalen. Vervolgens worden deze verschijnselen gemodelleerd en wordt er aangegeven op welke manier de effecten kunnen worden gecompenseerd. Dit wordt apart voor hoofd- en kruispolarisatiesignaal gedaan, omdat de effecten op beide signalen verschillend zijn. Uitgaande van deze modellen wordt de opzet geschetst voor een computerprogramma, dat de aangegeven bewerkingen uitvoert op de metingen aan Olympus. Het programma wordt vervolgens gedetailleerd beschreven. Vanuit deze gedetailleerde beschrijving is tenslotte, in Turbo C, een programma geschreven, dat op een PC-AT kan draaien. De tekst van het programma is bijgevoegd in de appendices van het verslag.

INHOUDSOPGAVE

INLEIDING.

Na lancering van de Olympus satelliet in voorjaar 1989 zullen op grote aantallen grondstations in Europa metingen worden verricht aan de door deze satelliet uit te zenden bakensignalen. Olympus zal bakensignalen uitzenden op 12.5, 20 en 30 GHz. Op 12.5 en 30 GHz zal alleen een verticaal gepolariseerd signaal worden uitgezonden, terwijl op 20 GHz zowel een horizontaal als een verticaal gepolariseerd signaal wordt uitgezonden. Van deze signalen zal onder meer de ontvangst in zowel de hoofdpolarisatierichting als in de kruispolarisatierichting gemeten worden.

Doel van deze metingen is de invloed vast te stellen die de atmosfeer, voornamelijk in de vorm van regen, heeft op de voortplanting van signalen in het gebied tussen 20 en 30 GHz, omdat dit frequentiegebied in de zeer nabije toekomst voor communicatiedoeleinden gebruikt zal gaan worden.

Ook in geval er geen verschijnselen zijn waarvan men de invloed op het verzonden signaal zou willen meten, dan nog is het ontvangen signaal niet constant. Om nu de gewenste metingen te kunnen verrichten moet bekend zijn welke verschijnselen het gemeten signaal mede beïnvloeden en wat hun invloed op het ontvangen signaal is.

Het doel van dit onderzoek is nu om een inventarisatie te maken van alle verschijnselen die een satellietbakensignaal beïnvloeden, deze invloeden te modelleren en uitgaande van dit model software te schrijven om de ongewenste invloeden van deze verschijnselen in propagatiemetingen te compenseren. Tevens zal geprobeerd worden aan te geven met welke nauwkeurigheid dit compenseren kan gebeuren.

De invloeden op de hoofd- en kruispolarisatiecomponent zijn niet gelijk en zullen om deze reden ieder afzonderlijk behandeld worden. Voor ieder van beide zal worden bekeken welke compensatiemechanismen kunnen worden gebruikt.

1 HET HOOFDPOLARISATIESIGNAAL.

1.1 Het algemene model

Om kennis te verkrijgen over de propagatie van signalen langs een satellietpad, zijn metingen over langere tijd nodig. Olympus, een communicatiesatelliet die door de ESA in voorjaar 1989 in een geo-stationaire baan om aarde gebracht zal worden, biedt onderzoekers de mogelijkheid propagatie metingen te verrichten. Voor dit doel zal Olympus continu bakensignalen uitzenden op de frequenties 12.5, 20 en 30 GHz. De signalen op 12.5 en 30 GHz zullen lineair gepolariseerd zijn in één richting en wel verticaal. Het signaal op 20 GHz zal in feite bestaan uit twee lineair gepolariseerde signalen loodrecht op elkaar, de een horizontaal de ander verticaal. De richting waarin een signaal is gepolariseerd noemt men de hoofdpolarisatie richting van dat signaal. Wordt een signaal van een satelliet naar aarde gezonden, dan zal het door allerlei oorzaken worden gedempt. Bij propagatie metingen is men voor het signaal in de hoofdpolarisatie richting voornamelijk geïnteresseerd in de demping die optreedt onder invloed van regen. Om deze meting goed uit te kunnen voeren is het noodzakelijk te weten welke andere verschijnselen ook een bijdrage leveren aan de demping van dit signaal. Voor het signaal in de hoofdpolarisatierichting geldt de volgende formule (zie ook fig. 1.1)

$$P_r = EIRP - L_{fs} + G_r - A_v - A_l - V_s - A_w - A_r \qquad (1.1)$$

waarin

P_r het gemeten vermogen in de ontvanger (dBW)

G_r winstfactor van de ontvangantenne (dB)

EIRP het schijnbaar door de satelliet uitgezonden isotrope vermogen $P_t + G_t$ (dBW)

L_{fs} de vrije-ruimte demping (dB)

A_v demping door gassen in de atmosfeer (dB)

A_l demping door waterdruppels in wolken (dB)

V_s variaties veroorzaakt door instabiliteiten in het zend- of ontvangsysteem (dB)

A_w demping veroorzaakt doordat regen, sneeuw of ijs achterblijft op de ontvangantenne (dB)

A_r demping veroorzaakt door regen (dB)

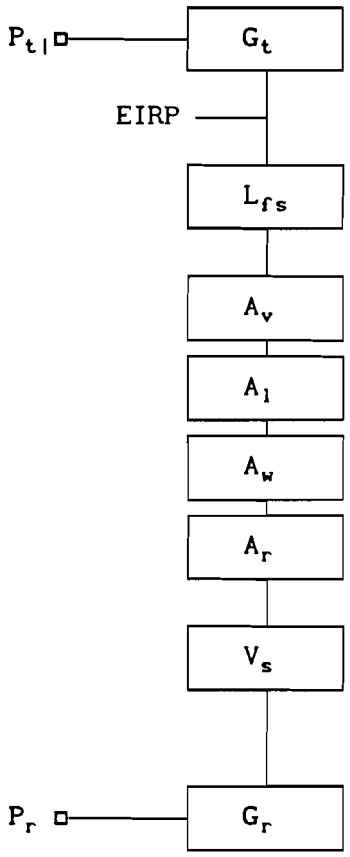


fig. 1.1 Model van de verstoringen van het co-polar signaal.

Om nu de demping door regen te kunnen bepalen uit de meting van P_r moeten alle andere componenten van (1.1) bekend zijn. Deze zullen nu stuk voor stuk nader bekeken worden en geprobeerd zal worden voor ieder van deze gevallen een adequaat model te bepalen.

1.2 De deelmodellen

1.2.1 EIRP het satellietvermogen

De EIRP (Equivalent Isotropically Radiated Power) is het vermogen dat de satelliet zou uitstralen als hij een isotrope straler was, waarbij de grootte zo is gekozen dat de flux die hierdoor op de aarde zou invallen gelijk is aan de daadwerkelijk invallende flux. De EIRP vervangt dus de factoren P_t en G_t uit fig.1.1. De EIRP van de satelliet hoeft niet gemodelleerd te worden. Dit is een door de fabrikant verstrekt gegeven en wordt geacht constant te zijn. Eventuele variaties in de EIRP worden meegenomen onder het punt variaties veroorzaakt door het systeem.
Voor Olympus zijn de volgende waarden gegeven.
EIRP: 10 dBW bij 12.5 GHZ en 24 dBW bij 20 en 30 GHz. [1]

1.2.2 L_{fs} de vrije-ruimte demping

Voor de vrije-ruimte demping is sprake van het model van een isotrope straler, die met een vermogen eirp(W) straalt. De flux die hierdoor op aarde invalt is [2]

$$\Phi = \frac{eirp}{4\pi d^2} \tag{1.2}$$

waarin

d de afstand satelliet ontvanger (m)

Het door een antenne opgevangen vermogen is dan

$$p_r = \Phi \cdot A_e \tag{1.3}$$

met [2]

$$A_e = \frac{\lambda^2}{4\pi} \cdot g_r \tag{1.4}$$

waarin

A_e de effectieve antenneapertuur (m^2)

λ de golflengte (m)

g_r de antennewinst van de ontvangantenne

ofwel

$$p_r = eirp \cdot g_r \cdot \frac{\lambda^2}{(4\pi d)^2} \qquad (1.5)$$

en met $P_r = 10 \cdot^{10}\log(p_r)$, $EIRP = 10 \cdot^{10}\log(eirp)$ en $G_r = 10 \cdot^{10}\log(g_r)$

$$P_r = EIRP + G_r - 10 \cdot^{10}\log\left(\frac{16\ \pi^2\ d^2}{\lambda^2}\right) \qquad (1.6)$$

waaruit voor de vrije-ruimte demping volgt [3]

$$L_{fs} = 10 \cdot^{10}\log\left(\frac{16\ \pi^2\ d^2}{\lambda^2}\right) \qquad (1.7)$$

1.2.3 A_v demping door gassen in de atmosfeer

In het frequentiegebied tussen 10 en 30 GHz zullen gassen in de atmosfeer een bijdrage leveren aan de demping van het satellietsignaal doordat absorptie optreedt aan de moleculen. De gassen die hieraan bijdragen zijn waterstof en zuurstof. De totale demping die deze absorptie veroorzaakt over een padlengte r wordt gegeven door [4,5]

$$A_v = \int_0^r \gamma(r)dr \quad dB \qquad (1.8)$$

waarin

$$\gamma(r) = \gamma_o(r) + \gamma_v(r) \quad dB/km$$

met

$\gamma(r)$ de specifieke demping door absorptie en

$\gamma_o(r)$ en $\gamma_v(r)$ de respectievelijke bijdragen van zuurstof en waterdamp hieraan.

Exacte berekening van deze specifieke demping is zeer complex. Voor practische toepassingen kan echter gebruik gemaakt worden van de volgende benaderingsformules voor de specifieke dempingen door zuurstof en waterdamp op grondniveau (1013 mb, 15 °C) [5]

$$\gamma_o = \left[7.19 \cdot 10^{-3} + \frac{6.09}{f^2+0.227} + \frac{4.81}{(f-57)^2+150}\right] f^2 \cdot 10^{-3} \ dB/km \qquad (1.9)$$

$$\gamma_v = \left[0.050 + 0.0021\rho_v + \frac{3.6}{(f-22.2)^2+8.6}\right] f^2 \cdot \rho_v \cdot 10^{-4} \quad \text{dB/km} \qquad (1.10)$$

voor 10 GHz < f < 30 GHz

waarin

f frequentie (GHz)

ρ_v waterdampdichtheid (g/m^3)

Voor een satellietpad moet (1.8) geïntegreerd worden over de atmosfeer om de totale demping te verkrijgen. Aangenomen wordt dat de dichtheid van beide gassen exponentiëel afneemt met de hoogte en een aparte equivalente hoogte voor zuurstof en waterdamp wordt ingevoerd, zodat geldt (voor ϑ>0.175)

$$A_v = \frac{\gamma_o h_o + \gamma_v h_v}{\sin\vartheta} \qquad (1.11)$$

waarin

h_o de equivalente zuurstofhoogte (km)

h_v de equivalente waterdamphoogte (km)

ϑ de elevatiehoek (rad)

Voor h_o en h_v gelden de volgende benaderingen [5]

h_o = 6 km

$$h_v = h_{v0}\left(1 + \frac{3.0}{(f-22.2)^2+5}\right) \text{ km} \qquad (1.12)$$

met

h_{v0} = 1.6 km bij helder weer

h_{v0} = 2.1 km bij regen

1.2.4 A_l demping door waterdruppels in wolken

Als een microgolfsignaal een gebied passeert waarin zich waterdruppels bevinden zal demping optreden ten gevolge van twee mechanismen:
1. scattering; 2. absorptie.
Zijn de druppels zeer klein van afmeting, zoals in wolken en mist, dan kan de bijdrage van scattering worden verwaarloosd ten opzichte van die van absorptie. [6]

De demping kan nu worden geschreven als

$$A_l = \gamma_l h_l \qquad (1.13)$$

waarin

γ_l de specifieke demping door absorptie (dB/km)

h_l de lengte van het pad door de wolken of mist (km)

In het frequentiegebied van 2 GHz tot 60 Ghz is de specifieke demping voor het geval van zeer kleine druppels slechts afhankelijk van de totale hoeveelheid water per volume eenheid en van de frequentie. Dit verband wordt gegeven door [6]

$$\gamma_l = 4.87 \cdot 10^{-4} \cdot f^2 \cdot \rho_l \quad (dB/km) \qquad (1.14)$$

waarin

ρ_l de waterdichtheid (g/m^3)

f de frequentie (GHz)

Deze formule (1.14) geldt bij een temperatuur van 18 °C. De temperatuur van wolken is echter ongeveer 0 °C, zodat een correctiefactor moet worden ingevoerd. Voor het gebied van 10 GHz tot 30 GHz kan worden volstaan met een constante factor van 1.95 [6], zodat voor wolken in het beoogde frequentiegebied geldt

$$\gamma_l = 9.5 \cdot 10^{-4} \cdot f^2 \cdot \rho_l \qquad (1.15)$$

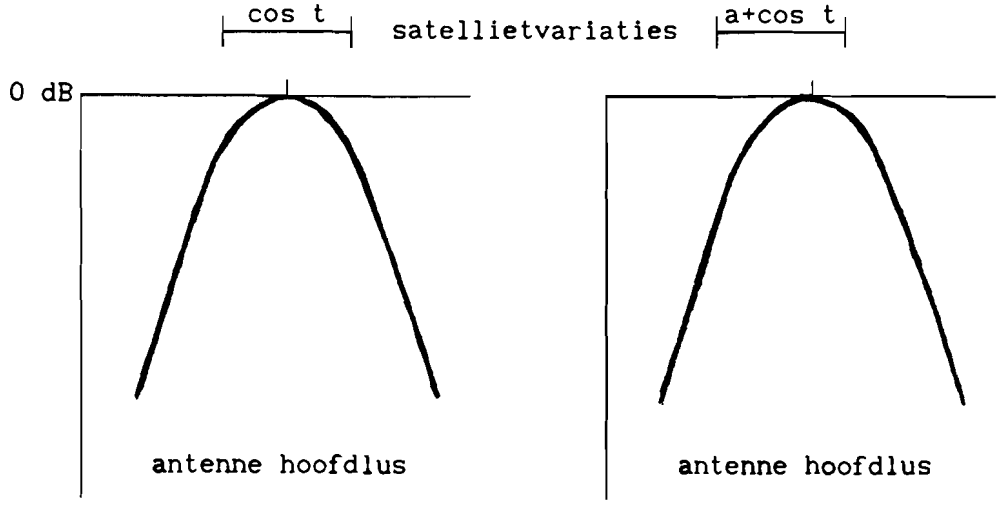
1.2.5 V_s variaties veroorzaakt door instabiliteiten in het zend- of ontvangsysteem

De systeemvariaties kunnen worden onderverdeeld in drie subgroepen.

1. variaties door onnauwkeurigheden in het meetsysteem.
Deze variaties zijn afhankelijk van de gebruikte apparatuur en de staat van onderhoud. Er mag van worden uitgegaan dat elk meetstation voldoende zorg heeft voor deze zaken, zodat dit niet hoeft te worden meegenomen.

2. variaties door het trackingsysteem.
Een geo-stationaire satelliet zal nooit geheel stil staan ten opzichte van een punt op aarde, maar zijn positie zal een harmonische trilling

uitvoeren rondom een gemiddelde waarde. Tevens zal dit gemiddelde zich langzaam langs de hemel verplaatsen, maar deze verplaatsing mag over een aantal dagen constant worden verondersteld.
De meeste antennesystemen beschikken tegenwoordig over een automatisch tracking systeem, zodat de antenne de satelliet zal kunnen volgen. [7] Mocht de antenne echter niet over zo'n systeem beschikken of mocht de automatische tracking voor korte tijd uitvallen, dan zal als gevolg hiervan de sterkte van het signaal variëren.
Nemen we aan dat het antennepatroon in de hoofdlus een kwadratisch verloop heeft en dat de antenne staat gericht op het gemiddelde punt van de satelliet (zie fig. 1.2a), dan zal het signaal variëren met $\cos^2 t = 0.5 + 0.5\cos 2t$ en dus een harmonische component hebben.
Compensatie kan dan eenvoudigweg geschieden door de parameters van de harmonische functie te schatten uit de data van voorafgaande en opvolgende dagen. Staat de antenne uit het midden (zie fig. 1.2b), dan zal het signaal variëren met $(a+\cos t)^2 = a^2 + 0.5 + 2a\cos t + 0.5\cos 2t$, met a maximaal 0.5 als de antenne in het uiterste punt staat. Voor compensatie zijn nu gegevens nodig over de exacte baan van de satelliet.



a. Antenne in gemiddelde satelliet punt

b. Antenne naast gemiddelde satelliet punt

fig. 1.2 Satellietbeweging ten opzichte van stilstaande antenne

Een andere mogelijkheid is gebruik maken van programtracking. Hierbij wordt de antenne gericht met behulp van een programma dat gebruik maakt van gegevens over de te verwachten satellietbaan. [7]

3. variaties in de sterkte van het door de satelliet uitgezonden signaal.
Onder invloed van opwarming door de zon en daarna weer afkoelen van de satelliet kan het satellietsignaal dagelijkse variaties vertonen. Deze variaties zullen onder meer afhangen van de temperatuurstabiliteit van de satelliet. Welke invloed de zon op Olympus zal hebben kan alleen proefondervindelijk vastgesteld worden.

1.2.6 A_w demping door water e.d. op de antenne

Water, sneeuw of ijs dat achterblijft op het antennesysteem zal een extra bijdrage leveren aan de demping van het te meten signaal.
Er kan een onderscheid worden gemaakt tussen twee systemen.

1. Antennesystemen die zonder bescherming in de open lucht zijn geplaatst. Hier kan zich water e.d. ophopen op het reflectoroppervlak of op de feed. Water op de feed zal voorkomen in de vorm van druppels en deze zullen korte maar hevige pieken kunnen veroorzaken in de demping, welke uit de meetresultaten verwijderd moeten worden omdat ze geen propagatie event vormen. Er bestaan systemen, zoals een snel roterend venster voor de feed of een blower die de druppels van de feed afblaast, die druppelvorming tegengaan, waardoor dit probleem niet op zal treden. Het verdient sterke aanbeveling van dit soort systemen gebruik te maken.
Op het reflector oppervlak kan zich zowel een waterfilm vormen als een sneeuwlaag of een ijsafzetting, die elk een extra bijdrage aan de demping kunnen geven. Droge sneeuw op de antenne heeft weliswaar vrijwel geen invloed op de demping, maar bij het smelten van de sneeuw kan demping optreden van meerdere dB's voor langere tijd. Tevens kan, doordat sneeuw achterblijft op de schotel, beam-squinting ontstaan. Dit wil zeggen dat de antenne een schijnbaar andere hoofdrichting zal hebben doordat het oppervlak van de schotel een andere vorm lijkt te hebben gekregen. Het effect van sneeuw en ijs kan worden

voorkomen door het reflector oppervlak te verwarmen, zodat ijs- en sneeuwvorming niet optreedt.
De invloed van water zal sterk afhangen van de kwaliteit van het oppervlak. Kan het water snel wegvloeien en krijgt het niet de mogelijkheid een waterfilm te vormen, dan kan de invloed op de demping verwaarloosd worden. Vormt zich echter een waterfilm, dan zal de demping afhangen van de dikte van de waterfilm, de frequentie en de vorm van de antenne. [8,9]. Het is dus van het grootste belang dat het reflector oppervlak waterafstotend wordt gemaakt. Dit kan bereikt worden door het oppervlak te behandelen met speciale coatings. [10]

2. Antennesystemen met een omhullende bescherming (radome).
De radome zal nu zorgen voor een bijdrage aan de demping, maar deze is gering en blijft constant bij een bepaalde frequentie.
De neerslag zal zich nu afzetten op het radome oppervlak. Hiervoor geldt, net als voor het reflector oppervlak bij onbeschermde antennes, dat ijs- en sneeuwafzetting tegengegaan kunnen worden door verwarming van het oppervlak en dat de demping door regen sterk afhangt van de waterafstotende werking van het oppervlak. Kan het water een film vormen op het oppervlak, dan zal de invloed hiervan op de demping aanzienlijk zijn [9,12,13] en kan zelfs oplopen tot ongeveer 10 dB bij 30 GHz voor een waterfilm met een dikte van 0.3 mm. [11]
Uit verschillende onderzoeken is gebleken dat het gebruik van waterafstotende lagen een zeer gunstige invloed heeft, omdat het water hierdoor geen film meer zal vormen maar snel van het oppervlak zal afstromen. [10,14,15] Door inwerking van het zonlicht en door vuilafzetting door luchtverontreiniging kunnen de waterafstotende eigenschappen echter sterk teruglopen, zodat de demping tijdens regen perioden weer sterk zal toenemen. [11]

Het is niet goed mogelijk algemene berekeningsmethoden te geven om de effecten van neerslag op antennesystemen te niet te doen, omdat de specifieke omstandigheden van te veel invloed zijn. Elk individueel meetstation zal zijn eigen afweging moeten maken of en op welke wijze deze effecten geëlimineerd moeten worden. Het zal echter duidelijk zijn dat deze effecten niet zonder meer mogen worden genegeerd.

1.3 De radiometer

Een belangrijk hulpinstrument bij propagatiemetingen is de radiometer. Een radiometer meet in feite de electromagnetische ruisemissie van de hemel(sky-noise). Omdat een absorberend medium microgolven uitstraalt, waarvan de intensiteit afhangt van de fysieke temperatuur van het medium en van zijn absorptie, kan uit dit soort metingen de demping worden afgeleid bij de frequentie waarop de radiometer meet. De factoren die absorptie veroorzaken in de atmosfeer zijn de reeds in **1.2.3** en **1.2.4** besproken water, waterdamp en zuurstof. Bij metingen met een radiometer moet er echter rekening mee worden gehouden dat niet alleen straling uit de atmosfeer wordt gemeten, maar dat er ook een gedeelte wordt gemeten dat vanuit de omgeving en dan vooral de van grond afkomstig is. Tevens zal de radiometer voor grote dempingen (>7dB) naar een verzadigingswaarde toegaan.

1.3.1 Sky-noise

De metingen van een radiometer worden niet gegeven in vermogen, maar de eenheid van temperatuur wordt gebruikt (K). De temperatuur, die gemeten wordt is niet direct de gezochte sky-noise temperature, maar de antenne temperatuur. De sky-noise temperature is hieruit als volgt af te leiden [16]

$$T_s = \frac{A_f}{h} T_a + \left(1 - \frac{A_f}{h}\right) T_e \qquad (1.16)$$

waarin

- T_s sky-noise temperature (K)
- T_a antenne temperatuur (K)
- T_e omgevingstemperatuur (K)
- A_f feed verliesfactor
- h gedeelte van de antennebundel dat naar de hemel is gericht

1.3.2 Demping

Uit de sky-noise temperature is de demping als volgt te berekenen [16]

$$T_s = T_m\left(1 - \frac{1}{a_t}\right) + \frac{1}{a_t} T_c \qquad (1.17)$$

waarin

T_m effectieve medium temperatuur (K)
T_c kosmische ruistemperatuur (K)
a_t atmosferische verzwakking

Voor de totale demping volgt dan

$$A_t = 10\cdot^{10}\log(a_t) = 10\cdot^{10}\log\left(\frac{T_m-T_c}{T_m-T_s}\right) \qquad (1.18)$$

1.3.3 Verband tussen demping en atmosfeer

Omdat de demping van het radiometer signaal alleen afhangt van de water en waterdamp hoeveelheid en de zuurstof in de atmosfeer kan dit verband als volgt worden gegeven (1.11), (1.13)

$$A_t = \frac{h_v\gamma_v}{\sin\vartheta} + \frac{h_o\gamma_o}{\sin\vartheta} + \gamma_l h_l \qquad (1.19)$$

Dit kan door het invullen van (1.9), (1.10), (1.12) en (1.14) in (1.19) worden herleid tot

$$A_t = a(f)\cdot V^2 + b(f)\cdot V + c(f)\cdot L + d(f) \qquad (1.20)$$

met

$V = \rho_v\cdot h_{v0}$ de totale hoeveelheid waterdamp per kolom van een vierkante meter in het satellietpad
$L = \rho_l\cdot h_l$ de hoeveelheid water per kolom van een vierkante meter wolk in het satellietpad

De factoren a(f), b(f), c(f) en d(f) zijn slechts van de frequentie afhankelijk, zodat bij een bekende frequentie een vergelijking

overblijft met twee onbekenden. Wordt gebruik gemaakt van twee radiometers op verschillende frequenties, dan kunnen V en L uit de radiometer metingen berekend worden. Met behulp van deze resultaten kan dan weer de demping bij een willekeurige andere frequentie onder gelijke omstandigheden berekend worden. Dit maakt het mogelijk radiometer metingen te gebruiken om de bijdrage van de atmosfeer aan de demping van een satellietsignaal te schatten.
Is maar één radiometer beschikbaar dan kan ρ_v bij de grond bepaald worden uit metingen van de omgevingstemperatuur en de relatieve vochtigheid om een schatting voor V te geven, met behulp van de volgende formules [17]

$$V = h_{v0} \cdot \rho_v \qquad (kg/m^2) \tag{1.21}$$

en

$$\rho_v = 216.7 \cdot e/T_e \qquad (g/m^3) \tag{1.22}$$

waarin

$$e = e_s \cdot rH/100$$

$$e_s = 5854 \cdot 10^{(20-2950/T_e)} \cdot T_e^{-5}$$

met

T_e de omgevingstemperatuur
e de waterdampdruk
e_s de verzadigingsdampdruk bij T_e
rH de relatieve vochtigheid in %

zodat geldt

$$V = 20296.988 \cdot 10^{(20-2950/T_e)} \cdot T_e^{-6} \cdot rH \qquad (kg/m^2)$$

Zijn er meer dan twee radiometers, dan zijn er meer dan twee vergelijkingen met twee onbekenden. Dit zal in principe een strijdig stelsel opleveren. Er moeten dan twee radiometers geselecteerd worden of er moet een procedure zijn die aan de verschillende metingen een verschillend gewicht toekent en de resultaten al naar gelang hun gewicht laat meewegen in de berekening van de demping. Omdat de demping bij hogere frequenties grotere variaties zal vertonen ligt het voor de hand aan de metingen bij hogere frequenties een groter gewicht toe te kennen.

2 HET KRUISPOLARISATIESIGNAAL

2.1 Algemeen

Wordt een signaal in één polarisatierichting -hetzij lineair, hetzij circulair- uitgezonden, dan zal, zeker in het hier beschouwde frequentiegebied van 10 GHz tot 30 GHz, het ontvangen signaal niet alleen een component in de polarisatierichting van de zender, de hoofdpolarisatierichting, hebben, maar ook een orthogonale component. Deze orthogonale component noemt men de kruispolarisatie. Kruispolarisatie ontstaat door een aantal verschillende oorzaken. De totale kruispolarisatie wordt gegeven door (zie ook fig. 2.1)

$$K_t = K_s + K_a + K_o + K_w \qquad (2.1)$$

waarin

K_t de totale gemeten kruispolaristie
K_s de kruispolarisatie door de satelliet
K_a de kruispolarisatie door de atmosfeer
K_o de kruispolarisatie door de ontvanger
K_w de kruispolarisatie doordat water e.d. achterblijft op het ontvangantennesysteem

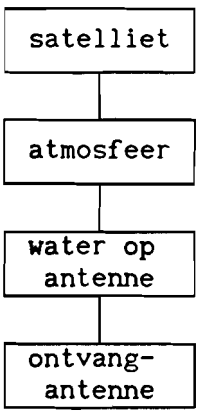


fig. 2.1 Factoren die kruispolarisatie veroorzaken

Bij propagatie metingen is men geïnteresseerd in de kruispolarisatie die veroorzaakt wordt door de atmosfeer. Om deze te kunnen onderscheiden van de overige effecten is het noodzakelijk te weten hoe elk van deze kruispolarisatie effecten ontstaan en te kijken welke methoden kunnen worden gebruikt om de ongewenste effecten uit de resultaten te verwijderen.

Vaak zal niet de sterkte van het kruispolarisatiesignaal worden gebruikt als maat voor de kruispolarisatie, maar de XPD (cross polarization discrimination), waardoor de sterkte van het kruispolarisatiesignaal gerelateerd wordt aan de sterkte van het hoofdpolarisatiesignaal. De XPD wordt gedefinieerd als

$$XPD = 20 \cdot {}^{10}\log \left| \frac{E_h}{E_k} \right| \qquad (2.2)$$

waarin

E_h de sterkte van het hoofdpolarisatiesignaal
E_k de sterkte van het kruispolarisatiesignaal

2.2 De verschillende kruispolarisatiebijdragen

2.2.1 Kruispolarisatie door de atmosfeer

De kruispolarisatie door de atmosfeer bestaat uit twee componenten

1. Kruispolarisatie door regen.
Doordat regendruppels niet rond zijn, maar enigzins afgeplat, en doordat ze gekanteld zijn ten gevolge van de windgradient, zal een gedeelte van het signaal dat wordt uitgezonden in één polarisatie richting terug te vinden zijn in de richting hier loodrecht op. [2] Kruispolarisatie van deze soort zal altijd gepaard gaan met een demping van het hoofdpolarisatiesignaal.

2. Kruispolarisatie door ijskristallen.
IJskristallen, die hoog in de lucht gevormd kunnen worden, zullen meestal een niet symmetrische vorm hebben en kunnen gericht worden door electrische velden. De kruispolarisatie die hierdoor wordt

veroorzaakt zal over het algemeen niet gepaard gaan met noemenswaardige demping. Men spreekt in dit geval ook wel van anomale kruispolarisatie. [18,19,20,21]

2.2.2 Kruispolarisatie door de ontvanger

De ontvanger zal op twee manieren bijdragen aan de kruispolarisatie.

1. Door thermische ruis in de ontvanger. Door goede specificaties van het grondstation is deze bijdrage onder controle te houden. [1]

2. Doordat de ontvangantenne niet ideaal is. In de praktijk zal een antenne nooit helemaal ideaal zijn, waardoor de antenne een gedeelte van het ontvangen signaal naar de andere polarisatierichting zal kunnen verschuiven. [2] Een goed antenne ontwerp zal de kruispolarisatie eigenschappen van een antenne verbeteren, waarbij een XPD van 40 dB haalbaar moet zijn voor zeer geavanceerde systemen. Wil men echter de atmosferische kruispolarisatie meten met een nauwkeurigheid van 1dB, dan moet de XPD van de antennes minstens 20 dB beter zijn dan die van de atmosfeer. [22,23] Daar men de atmosferische XPD vanaf 35 dB wil meten is het dus noodzakelijk de effecten die door de antennes worden veroorzaakt uit de resultaten te verwijderen.

2.2.3 Kruispolarisatie door de satelliet

Voor de satellietantenne geldt, net als voor de ontvangantenne, dat het niet perfect zijn van de antenne kruispolarisatie zal veroorzaken. De sterkte van de kruispolarisatie component die door de satelliet wordt veroorzaakt ligt in de ordegrootte van 30-35 dB, maar zal aanmerkelijk worden beïnvloed door het feit dat de antenne dagelijks door de zon wordt opgewarmd. Dit kan sterke variaties gedurende de dag veroorzaken. Over periodes van enkele weken zullen deze dagelijkse variaties een redelijk vast patroon vertonen. Dit patroon zal zich langzaam met de seizoenen wijzigen. Hierdoor kunnen metingen van voorafgaande en opvolgende dagen gebruikt worden om een schatting te geven van het verloop van de kruispolarisatie op een bepaalde dag. Er

ontstaat echter een probleem in geval van een event waarbij demping optreedt, omdat de kruispolarisatie component van de satelliet op zijn pad door de atmosfeer ook gedempt zal worden. Daar deze demping niet optreedt voor kruispolarisatie door de ontvanger kan in een dergelijk geval alleen gecorrigeerd worden als de beide componenten van elkaar onderscheiden kunnen worden.

2.2.4 Kruispolarisatie door water e.d. op de antenne

Water, sneeuw of ijs dat achterblijft op de antenne zal een extra kruispolarisatiecomponent opleveren. [1,8] De grootte van deze component is afhankelijk van de vorm en afmetingen van de antenne, van dikte van de neerslaglaag op de antenne, de polarisatierichting en de frequentie. Het tegengaan van deze verschijnselen kan het best geschieden door ze te proberen te voorkomen. Maatregelen om dit te bereiken zijn reeds beschreven in **1.2.6**. Uit recent onderzoek is echter gebleken, dat het gebruik van waterafstotende materialen voor de bouw van radomes mogelijk een licht verslechterend effect op de kruispolarisatie heeft. [24] Het betreft hier een eerste onderzoek en er kunnen dan ook geen vergaande conclusies aan verbonden worden. Gewacht moet worden op verdere onderzoeksresultaten, die deze vermoedens al of niet bevestigen. Is preventie niet mogelijk, dan zal per geval bekeken moeten worden wat er gedaan kan worden om deze effecten te compenseren.

2.3 Fasedraaiing bij kruispolarisatie

Bovenbeschreven verschijnselen dragen niet alleen bij tot het ontstaan van kruispolarisatie, maar zorgen er ook voor dat er faseverschillen ontstaan tussen de signalen, wat het noodzakelijk maakt ook de fase van het kruispolarisatiesignaal te meten t.o.v. het hoofdpolarisatie-signaal. [21]

3 OPZET VAN HET SOFTWARE SYSTEEM VOOR HET HOOFDPOLARISATIESIGNAAL.

3.1 Algemeen

Binnenkomende propagatiegegevens moeten gecorrigeerd kunnen worden door het software systeem voor de ongewenste verschijnselen zoals geschetst in hoofdstuk 1. Het softwaresysteem moet draaien op een ms-dos pc-at en de software moet geschreven worden in de programmeertaal C (gebruikt is Borland Turbo-C 2.0).

Hiertoe zijn beschikbaar, voor iedere dag, metingen van het hoofdpolarisatiesignaal, metingen van één of meer radiometers en eventueel een aantal meteorologische gegevens zoals omgevingstemperatuur, relatieve vochtigheid, regenintensiteit.

Om correcties toe te passen wordt voor iedere dag die men wil corrigeren een referentieniveau bepaald. Het referentieniveau geeft een schatting van het signaalniveau veroorzaakt door alle optredende ongewenste verschijnselen gezamenlijk, behalve atmosferische demping, gedurende de betreffende dag.

Bij clear-sky dagen bestaat de verstoring alleen uit variaties door het systeem en een kleine atmosferische bijdrage. Hier kan de invloed van de atmosfeer worden afgeschat uit de radiometergegevens van die dag. De waarden die overblijven na aftrekken van deze schatting voor de atmosferische invloed vormen de systeemcurve voor de betreffende dag. De systeemcurve voor een bepaalde periode geeft een schatting van de invloed van variaties in het systeem over die periode en bepaalt het referentieniveau. Het verschil tusssen gemeten signaalwaarde en referentieniveau geeft de totale atmosferische demping.

Voor dagen waarop een of meer events optreden, dat wil zeggen dagen waarbij de demping een van te voren vastgestelde drempel overschrijdt, kan de atmosfeerdemping worden bepaald voor die tijden waarop de demping minder is dan de gegeven drempel. De systeemcurve kan dan voor dezelfde tijdstippen worden bepaald.

Omdat variaties in het systeem een dagelijks terugkerend patroon hebben dat in de tijd slechts langzaam verandert, kan de gebruiker een aantal aansluitende dagen kiezen waaruit een gemiddelde curve wordt bepaald om de systeemcurve aan te vullen voor de eventperiodes.

Eventueel kan de totale atmosfeerdemping met behulp van de radiometers worden bepaald tot een niveau dat hoger ligt dan de eventdrempel, zelfs tot het niveau waarop de radiometer in verzadiging raakt. De nauwkeurigheid van de atmosfeerdemping wordt dan weliswaar geringer, maar ook de periode waarover de systeeminvloed moet worden geschat wordt geringer. (zie fig. 3.1)

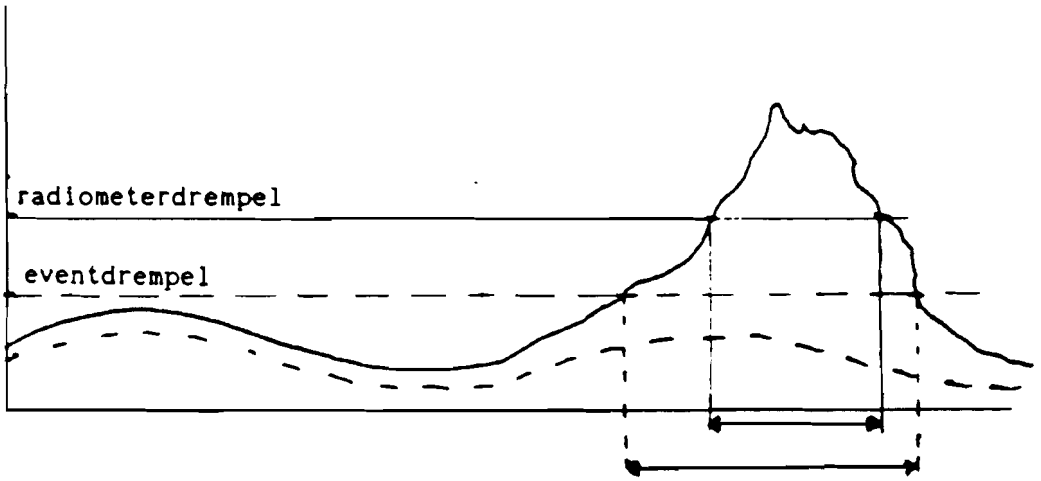


fig. 3.1 Verschillende correctieintervallen van de systeemcurve bij eventdrempel en radiometerdrempel

De gebruiker kan zelf bepalen welke methode hij prefereert, omdat hij de intervallen waar de radiometers niet gebruikt mogen worden zelf moet specificeren. Het systeem zal echter als default waarde steeds de eventdrempel kiezen.

Afgezien van deze correcties die iedere dag moeten worden uitgevoerd, kunnen ook nog fouten optreden door het trackingsysteem. De gebruiker moet deze zelf signaleren en het patroon hiervan vaststellen, zodat het programma een curve kan maken om de data te corrigeren.

De gebruiker krijgt ook de mogelijkheid een correctie uit te voeren voor het dempingseffect van water dat achterblijft op de antenne. De gebruiker moet zelf een verband opgeven tussen de demping en de regenintensiteit alsmede de periode waarvoor hij de correctie wil toepassen.

De invoerdata, de correctiewaarden, de gecorrigeerde waarden en de uitvoerwaarden moeten steeds grafisch op het scherm weergegeven kunnen worden voor inspectie door de gebruiker. Hiertoe wordt een grafische module vervaardigd.

Het totale systeem ziet er nu uit zoals geschetst in figuur 3.2

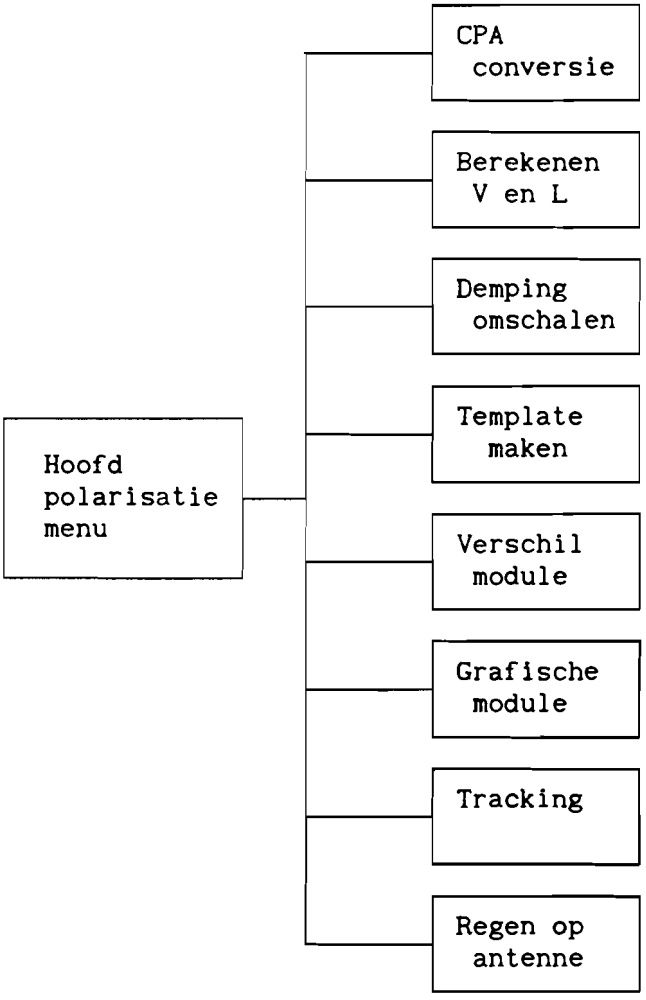


fig. 3.2 Opbouw van het softwaresysteem

Het systeem is zo opgezet dat voor iedere afgeronde bewerking een module vanuit het hoofdmenu kan worden aangeroepen. De eerste drie modules moeten steeds worden uitgevoerd. Het maken van templates hoeft alleen te gebeuren als er een event is opgetreden. De verschil module is een stuk gereedschap om het verschil tussen twee files uit te rekenen en wordt steeds als het nodig is aangeroepen, bijvoorbeeld om systeemcurves te maken voor de templatebepaling. Ook de grafische module is een hulpmodule. Deze kan steeds worden gebruikt om files te inspecteren voorafgaand aan en na afloop van een bewerking. De laatste twee modules zijn optioneel.

De gebruiker moet zelf, afhankelijk van de beschikbare invoer en de reeds verwerkte data, bepalen welke bewerking wordt uitgevoerd. De procedures checken alleen op de aanwezigheid van de benodigde files. Omdat de eerste drie modules altijd na elkaar moeten worden uitgevoerd zal in het keuzemenu na afloop van een van deze bewerkingen de hierop volgende keuze worden benadrukt.

Iedere module beschikt over een file met parameters. Na aanroepen van de betreffende module worden de parameters zichtbaar gemaakt en kan de gebruiker ze, waar nodig, wijzigen. Deze nieuwe waarden worden dan in de parameterfile opgeslagen.

Hieronder zal voor iedere module beschreven worden welke handelingen door de betreffende module verricht moeten worden, welke parameters nodig zijn en wat de in- en uitvoer zal zijn.

3.2 De modules

3.2.1 CPA conversie

Deze module converteert de meetwaarden van de hoofdpolarisatiesignalen naar CPA waarden. Hiertoe moet de module het volgende doen.
-lezen van een file met hoofdpolarisatie meetwaarden
-de meetwaarden omrekenen naar CPA waarden
-de CPA waarden opbergen in een file

Parameters
-de conversie gegevens voor de omrekening
-de namen van de in- en uitvoerfile
-begin- en eindtijdstip

Invoer
-file met hoofdpolarisatiemeetwaarden

Uitvoer
-file met CPA waarden

3.2.2 Berekenen V en L waarden

Deze module berekent uit de radiometer metingen de totale hoeveelheid water en waterdamp per vierkante meter op het satellietpad, V en L, zoals aangegeven in 1.3.3.
Hiertoe moet hij achtereenvolgens de volgende bewerkingen uitvoeren.
-lezen van de radiometerdata
-conversie naar antenne temperatuur (K)
-berekenen sky-noise temperatuur uit de antenne temperatuur
-berekenen van de demping uit de sky-noise temperatuur
-berekenen van V en L uit de demping
-wegbergen van V en L in een file

Parameters
-de radiometerfrequentie(s)
-de conversiegegevens voor de berekening van de antenne temperatuur voor iedere radiometer
-de antenne parameters voor de berekenig van de sky-noise temperatuur
-de namen van de in- en uitvoerfile
-begin- en eindtijdstip

Invoer
-de file met radiometerdata

Uitvoer
-de file met V en L per tijdblok

3.2.3 Demping omschalen

Deze module berekent met behulp van de waarden van V en L de atmosfeer demping bij de gegeven bakenfrequentie. Hiertoe voert de module de volgende bewerkingen uit.
-lezen van de file met V en L
-berekenen van de demping bij de bakenfrequentie
-wegbergen van de dempingsgegevens in een file

Parameters
-de bakenfrequentie
-de namen van de in- en uitvoerfile
-begin- en eindtijdstip

Invoer
-de file met V en L waarden

Uitvoer
-de file met berekende dempingswaarden bij de bakenfrequentie

3.2.4 Template maken

Deze module berekent een schatting voor de dagelijkse variatie die door het systeem wordt veroorzaakt ten tijde van een event. De berekening gebeurt door middeling van de systeemcurves van een aantal voorafgaande of opvolgende dagen gedurende hetzelfde tijdinterval als waarvoor geschat moet worden. De gebruiker bepaalt het aantal te gebruiken dagen en welke dagen geschikt zijn. De gemiddelde waarden worden gebruikt als schatting voor de systeemcurve ten tijde van het event.
De module voert de volgende bewerkingen uit.
-lezen benodigde systeemcurves van de andere dagen
-middelen van de gelezen waarden
-opslaan van de gemiddelden in een file

Parameters
-aantal invoerfiles
-de namen van de invoerfiles
-begin- en eindtijdstip van de gewenste periode

Invoer
-systeemcurves voor clear-sky periodes

UItvoer
-Template voor systeemvariaties tijdens event

3.2.5 Verschil module

Deze module moet het verschil kunnen bepalen voor iedere combinatie van twee tijdreeksen. De gebruiker moet er zelf zorg voor dragen dat de juiste files gebruikt worden en dat begin- en eindpunten en periodes overeen komen.
De tijdreeksen kunnen bijvoorbeeld files met CPA waarden, files met berekende atmosferische demping of files met templates zijn.
Het verschil kan alleen worden uitgerekend als beide files de opgegeven periode bevatten.
De module voert de volgende bewerkingen uit.
-lezen beide invoerfiles
-berekenen van het verschil tussen de invoerfiles
-wegbergen in een file van het berekende verschil

Parameters
-de namen van de twee invoerfiles
-de naam van de uitvoerfile
-begin- en eindtijdstip

Invoer
-twee files met een tijdreeks

Uitvoer
-file met berekende verschilwaarden

3.2.6 Grafische module

Deze module moet iedere gewenste tijdreeks grafisch op het beeldscherm kunnen weergeven. De gebruiker geeft de gewenste periode op en het bereik en de filenaam die hij op het scherm wil hebben. De module roept de grafische mode aan en tekent met behulp van de ingelezen waarden het tijdverloop.
De module voert de volgende bewerkingen uit.
-lezen opgegeven file
-schalen van het scherm
-tekenen van de tijdreeks

Parameters
-naam invoerfile
-begin- en eindtijd
-vertikale schaalwaarde

Invoer
-file met tijdreeks

Uitvoer
-grafiek naar beeldscherm

3.2.7 Tracking

Deze module berekent een correctiefiguur in het geval dat er verstoringen zijn doordat de tracking niet functioneert. De gebruiker moet zelf bepalen of gebruik van deze module noodzakelijk is.
Er wordt van uitgegaan dat het ontbreken van tracking een sinusvormige variatie oplevert. De module bepaalt nu de amplitude en de periode van de sinus en de plaats van een maximum of geeft de gebruiker de mogelijkheid deze op te geven.

Hiertoe voert hij de volgende bewerkingen uit.
-lezen van een of meerdere CPA files
-berekenen van amplitude, periode en plaats van de sinus of inlezen van door de gebruiker hiervoor gegeven waarden
-vaststellen van de sinuskromme voor het gegeven tijdinterval
-wegbergen van de sinuswaarden in een file

Parameters
-namen van de in- en uitvoerfile(s)
-begin- en eindtijdstip van de te corrigeren periode

Invoer
-CPA file(s) na correctie voor de atmosfeerdemping
-gebruikersvariabelen voor de sinus

Uitvoer
-file met sinuswaarden voor de gewenste periode
deze file geeft een schatting van de trackingfout en kan met de verschilmodule van de CPA file worden afgetrokken

3.2.8 Regen op antenne

Deze module moet de additionele demping berekenen die wordt veroorzaakt doordat regen achterblijft op het onvangantennesysteem. De gebruiker moet bepalen of gebruik van deze module nodig is aan de hand van de resultaten van de template bepaling. Omdat de invloed van regen op de antenne voor ieder meetstation verschillend zal zijn moet de module de gebruiker de mogelijkheid geven zelf te bepalen hoe deze extra demping wordt berekend. Voor deze berekening moet gebruik gemaakt kunnen worden van meteorologische gegevens zoals: omgevingstemperatuur, regenintensiteit, windsnelheid en windrichting.
De module voert de volgende bewerkingen uit.
-berekenen van de demping uit de opgegeven formule
-wegbergen van de resultaten in een file

Parameters

-naam van de uitvoerfile

-gebruikersformule

-begin- en eindtijdstip

Invoer

-meteorologische gegevens

Uitvoer

-file met waarden die met behulp van de verschilmodule van de signaalwaarden moeten worden afgetrokken

4 GEDETAILLEERDE BESCHRIJVING VAN DE SOFTWARE VOOR HET HOOFDPOLARISATIESIGNAAL.

Bij de gedetailleerde beschrijving van de software wordt uitgegaan van de opbouw zoals geschetst in fig. 3.2. Ieder van de genoemde modules wordt zo beschreven dat hiermee overgegaan kan worden tot het daadwerkelijk schrijven van het programma onderdeel. Omdat de opzet van het geheel modulair is kan voor ieder onderdeel afzonderlijk vanuit de gedetailleerde beschrijving de software geschreven worden. Van alle belangrijke programmadelen wordt een flowschema gegeven. Voor deze beschrijving is gebruik gemaakt van [25,26].

4.1 Het Hoofdprogramma

Het hoofdprogramma bevat de mainmodule van het gehele programma. Het geeft de keuze tussen het hoofdpolarisatiemenu en het kruispolarisatiemenu. Na het verlaten van elk van deze menu's keert men terug naar het hoofdprogramma. Het programma wordt beëindigd door de keuze "beëindig het programma". De opbouw van het programma is hetzelfde als hieronder beschreven voor het hoofdpolarisatiemenu.

4.1.1 Het Hoofdpolarisatiemenu

Dit onderdeel bevat het keuzemenu voor het gehele hoofdpolarisatie gedeelte van het programma. Van hieruit moet gekozen worden welk onderdeel wordt uitgevoerd, waarna teruggekeerd wordt naar het keuzemenu. Hiertoe wordt de functie menu aangeroepen. Deze zet, grafisch, de keuze mogelijkheden op het scherm. Met behulp van de up- en downpijlen kan de keuze worden gemaakt en daarna met de entertoets geactiveerd. Er zijn negen keuze mogelijkheden: de acht programma onderdelen en de mogelijkheid "beëindig het programma". Menu kent aan iedere keuze een getalwaarde toe die wordt teruggeven. Een case-statement selecteert met dit getal het juiste programma onderdeel. Om te zorgen dat steeds wordt teruggekeerd naar het

keuzemenu wordt de functie menu binnen een do-loop uitgevoerd. De keuze "beëindig het programma" levert de voorwaarde om de do-loop te verlaten, waarna vanzelf wordt teruggekeerd naar het hoofdprogramma. Het flowschema voor dit programma is weergegeven in fig. 4.1.

De functie menu is als volgt opgebouwd. Allereerst worden strings gemaakt met de teksten voor het keuzemenu. Vervolgens wordt de achtergrond getekend en worden de teksten in de figuur geplaatst. Nu wordt de keuzeparameter b gelijk aan een startwaarde gemaakt. De functie begint nu een loop waarin achtereenvolgens de achtergrond veranderd wordt voor de keuze die hoort bij de waarde van b, gewacht wordt op het indrukken van een toets, de achtergrond weer normaal wordt gemaakt, als de ingedrukte toets een up- of downpijl is de waarde van b wordt verlaagd of verhoogd waarbij er voor gezorgd wordt dat de waarde van b niet te laag of te hoog wordt, en als de entertoets wordt ingedrukt de voorwaarde wordt gegeven om de loop te verlaten. Alle andere toetsen worden genegeerd. Is de loop verlaten dan wordt de waarde van b, die de gemaakte keuze vertegenwoordigt, teruggegeven aan het hoofdprogramma. Het flowschema van de functie menu is weergegeven in fig. 4.2.
De programmadelen die behoren bij de acht keuzes van het hoofdmenu worden hieronder apart beschreven. Dit zijn ieder aanzienlijk grote delen van het programma, die ieder weer bestaan uit verschillende onderdelen.

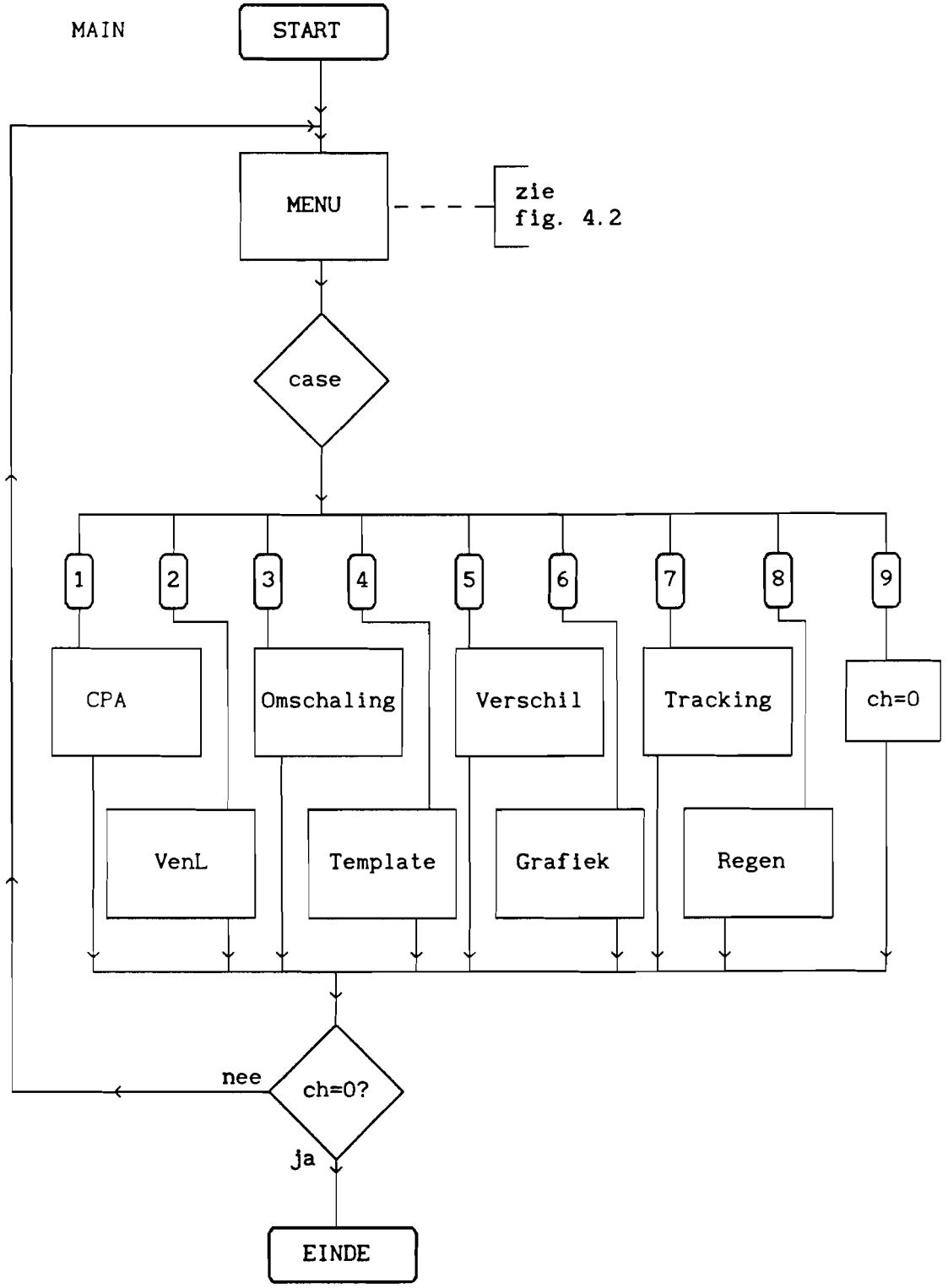


fig. 4.1 Flowschema van het hoofdprogramma

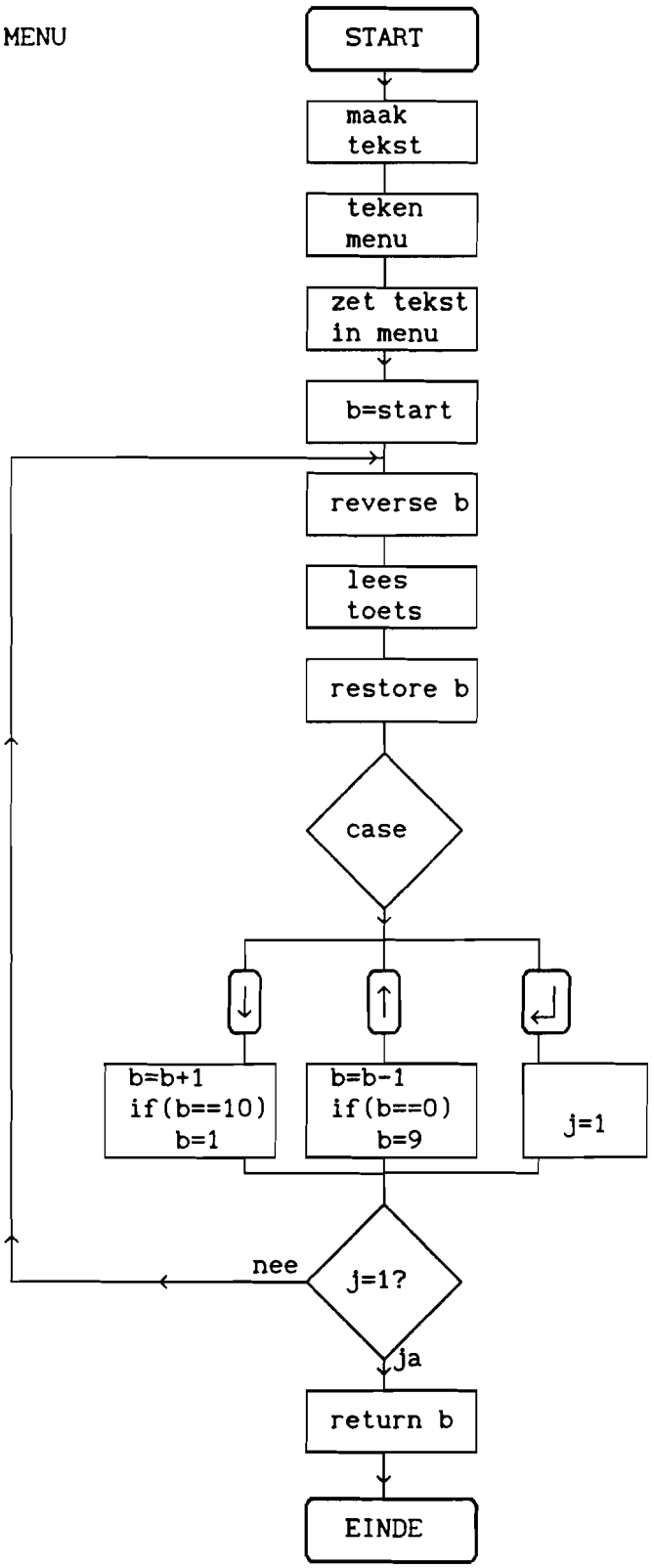


fig. 4.2 Flowschema van de functie menu

4.2 CPA

De functie CPA converteert de meetwaarden van een hoofdpolarisatie signaal naar CPA waarden. Hiertoe roept de functie eerst de functie parameters_cpa aan. Deze zet de parameters die nodig zijn op het beeldscherm en geeft de mogelijkheid deze te wijzigen. Hieronder vallen ook begin- en eindtijd van de te bewerken periode en de namen van de in - en uitvoerfile. De parameterlijst biedt ook de keuzemogelijkheid terug te keren naar het hoofdprogramma. De functie geeft dan de waarde 1 aan de variabele 'terug'. Is men tevreden dan worden de parameters opgeslagen en geeft de functie de waarde 0 aan 'terug' en gaat het programma verder. Vervolgens wordt de functie checkfile_cpa aangeroepen. Deze checkt of de invoerfile bestaat en of de opgegeven periode hierin valt. Zoniet dan wordt teruggegaan naar de parameterlijst, zodat een andere naam of periode gegeven kan worden. Ook wordt gekeken of de uitvoerfile bestaat. Bestaat de file niet dan wordt aan de variabele 'uit' de waarde 0 toegekend. Bestaat de file wel dan wordt gekeken of wordt overschreven, 'uit' wordt 2, of wordt toegevoegd aan het eind van de file, 'uit' wordt 1. Wil men niet overschrijven of toevoegen dan wordt teruggegaan naar de parameter lijst. Is alles in orde dan geeft de functie de waarde 1 terug aan de variabele 'check', zoniet dan wordt 'check' 0 en wordt teruggegaan naar de parameterlijst. Als laatste wordt dan de functie converteer aangeroepen. Deze opent de in- en uitvoerfile voert de conversie uit en schrijft de CPA waarden in de uitvoerfile. De functie krijgt de parameterlijst en de waarde van 'uit' mee. Is de bewerking voltooid dan geeft de functie CPA hiervan een melding. Flowschema in fig. 4.3.

De functie parameters_cpa leest de parameters uit de file cpa.par, zet deze op het scherm en geeft de mogelijkheid ze te veranderen of om terug te keren naar het hoofdprogramma. Hiertoe opent de functie eerst de file cpa.par. Als de file nog niet bestaat wordt hij gemaakt en gevuld met nullen. Daarna wordt de file gelezen en worden de waarden in een structure gezet. Vervolgens worden de parameternamen op het scherm gezet en dan daarachter de bijbehorende waarden. Met de variabele y wordt de regel bijgehouden waar de cursor zich bevindt. Als beginpositie wordt gekozen de regel met de keuze 'tevreden'. De functie gaat nu een loop uitvoeren waarin eerst de cursor op de regel

wordt gezet die hoort bij de waarde van y en dan wordt gewacht tot er een toets wordt ingedrukt. Bij een down pijl wordt de waarde van y met één verhoogd, bij een up pijl met één verlaagd, waarbij als y te groot of te klein wordt naar boven- of onderaan wordt gesprongen. Bij indrukken van de entertoets wordt gekeken of de cursor op de regel met 'tevreden' staat, dan wordt de voorwaarde gezet om de loop te verlaten, of op de regel met 'terug naar hoofdmenu', dan wordt de functie beëindigd en de waarde 1 aan 'terug' gegeven. Op alle overige plaatsen wordt de entertoets genegeerd. Wordt een andere toets ingedrukt dan wordt gekeken of hij alphanumeriek is. Zo ja dan wordt alles wat op die regel wordt getypt gelezen en aan de betreffende parameter toegekend. Zo nee dan wordt de toets genegeerd. De gebruiker moet er zelf voor zorgen dat de opgegeven waarden overeenstemmen met het formaat dat geldt voor de betreffende parameter. Wordt de loop verlaten dan worden de nieuwe waarden voor de parameters in de .par file geschreven en wordt aan 'terug' de waarde 0 gegeven. Het flowschema wordt gegeven in fig. 4.4.

De functie checkfile_cpa controleert of de opgegeven files bestaan en of de opgegeven periode hierin valt. De functie opent hiertoe eerst de invoerfile. Bestaat deze niet dan wordt de functie beëindigd en de variabele 'check' wordt 0. Bestaat de file wel dan wordt gekeken of de opgegeven periode hierin valt. Is dit niet zo dan wordt de file weer gesloten en 'check' wordt 0. Is dit wel zo dan wordt de file gesloten en de uitvoerfile wordt geopend. Bestaat deze niet dan wordt de variabele'uit' die de status van de uitvoerfile weergeeft 0 gemaakt en 'check' krijgt de waarde 1. Bestaat de file wel dan wordt gekeken of de opgegeven periode binnen de file valt. Is dit niet zo dan wordt gevraagd of men achteraan wil toevoegen. Bij niet toevoegen wordt de file gesloten en 'check' wordt 0. Bij wel toevoegen wordt 'uit' 1. Is dit wel zo dan wordt gevraagd of men wil overschrijven. Bij niet overschrijven wordt de file gesloten en 'check' wordt 0. Bij wel overschrijven wordt 'uit' 2. De file wordt nu gesloten en 'check' krijgt de waarde 1. Als de waarde van 'check' 0 is wordt teruggegaan naar de functie parameters_cpa, is 'check' 1 dan gaat het programma verder. De waarde van 'uit' wordt via een pointer teruggegeven aan het programma.

De functie converteer leest de meetwaarden uit de invoerfile, voert de conversie uit en schrijft de CPA waarden in de uitvoerfile. hiertoe opent de functie eerst de invoerfile en dan afhankelijk van de waarde van 'uit' de uitvoerfile. Is 'uit' 0, dan bestaat de uitvoerfile nog niet en kan deze worden geopend om in te schrijven, de file wordt dan vanzelf gemaakt. Is 'uit' 1, dan bestaat de file wel en moet worden geopend om toe te voegen. Is uit 2, dan bestaat de file en moet worden geopend om in te schrijven en het beginadres moet worden opgezocht. Vervolgens wordt in een loop de informatie van één tijdblok gelezen, met behulp van de conversieformule en de parameters de bijbehorende CPA waarde berekend en de uitkomst in de uitvoerfile geschreven. De loop wordt beëindigd als het laatste tijdblok van de opgegeven periode is behandeld. Hierna worden de files gesloten en keert het programma vanzelf terug naar het hoofdprogramma.

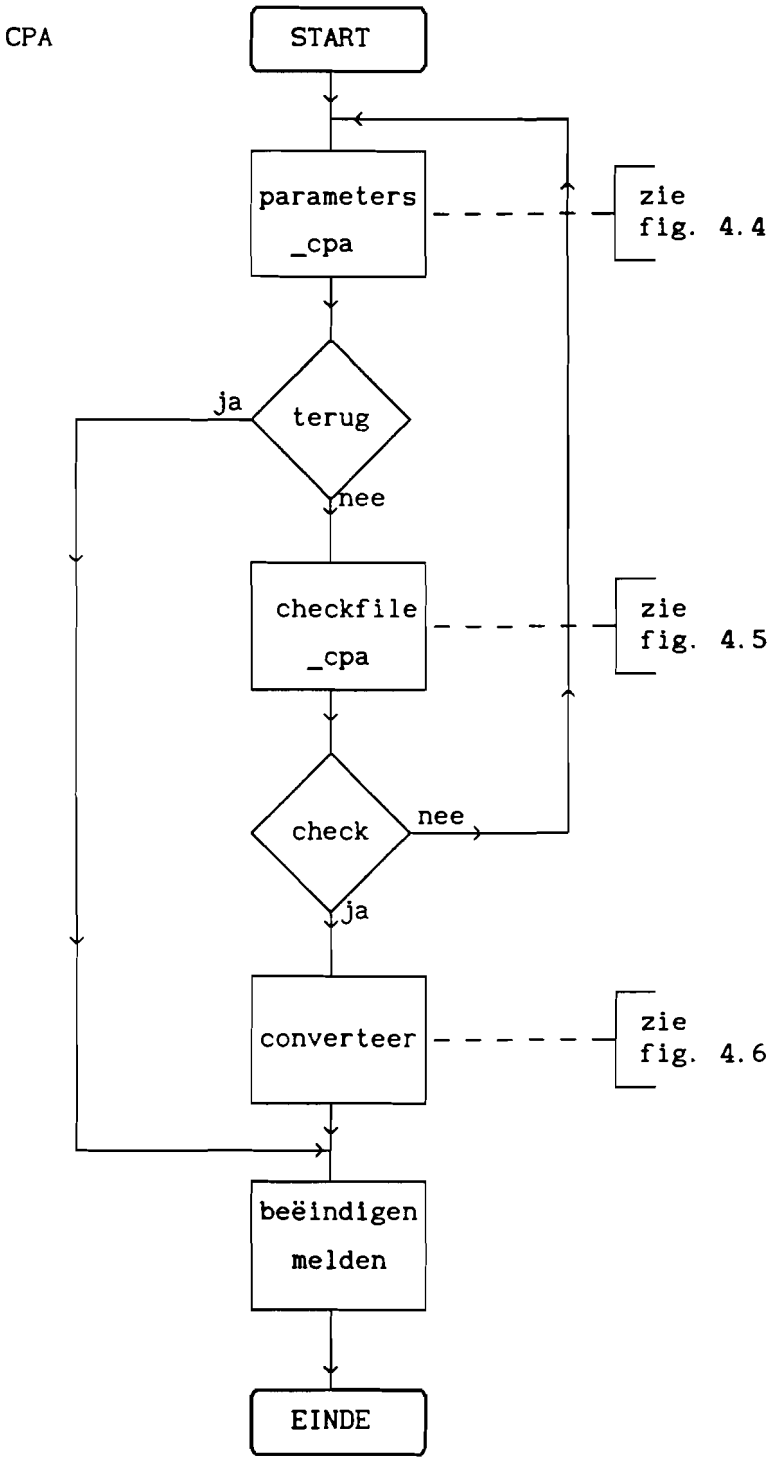


fig. 4.3 Flowschema van de functie CPA

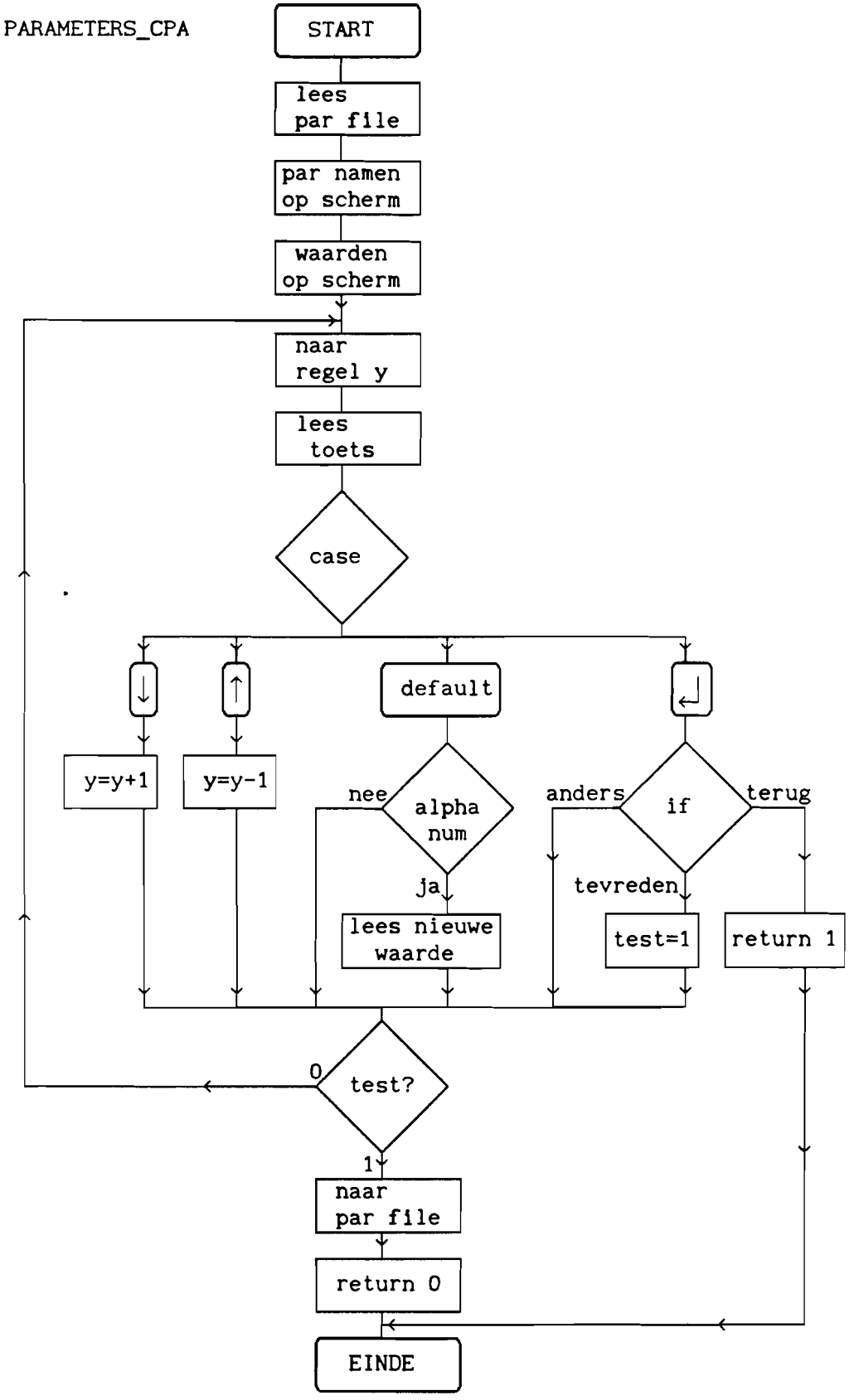


fig. 4.4 Flowschema van de functie parameters_cpa

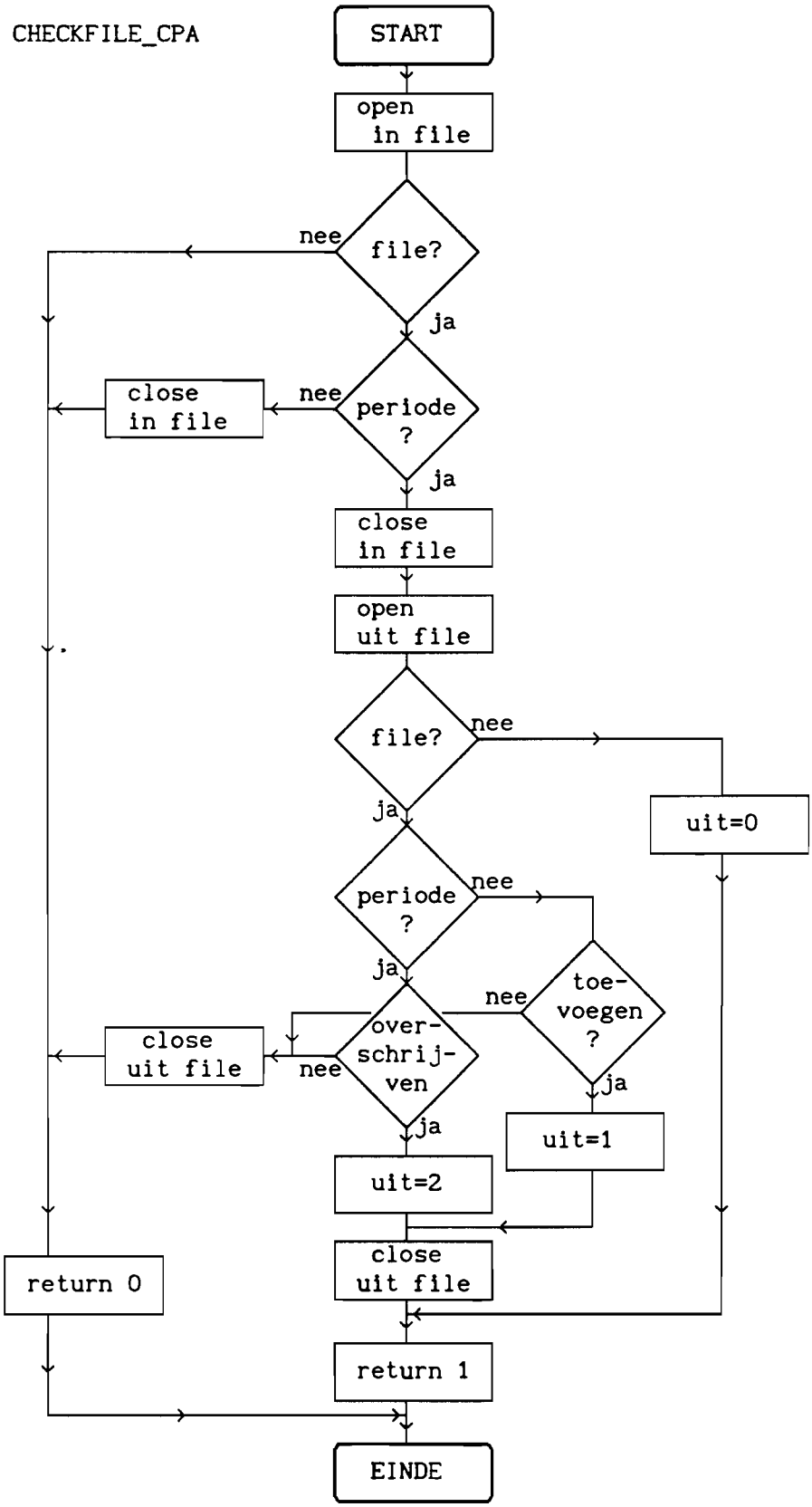


fig. 4.5 Flowschema van de functie checkfile_cpa

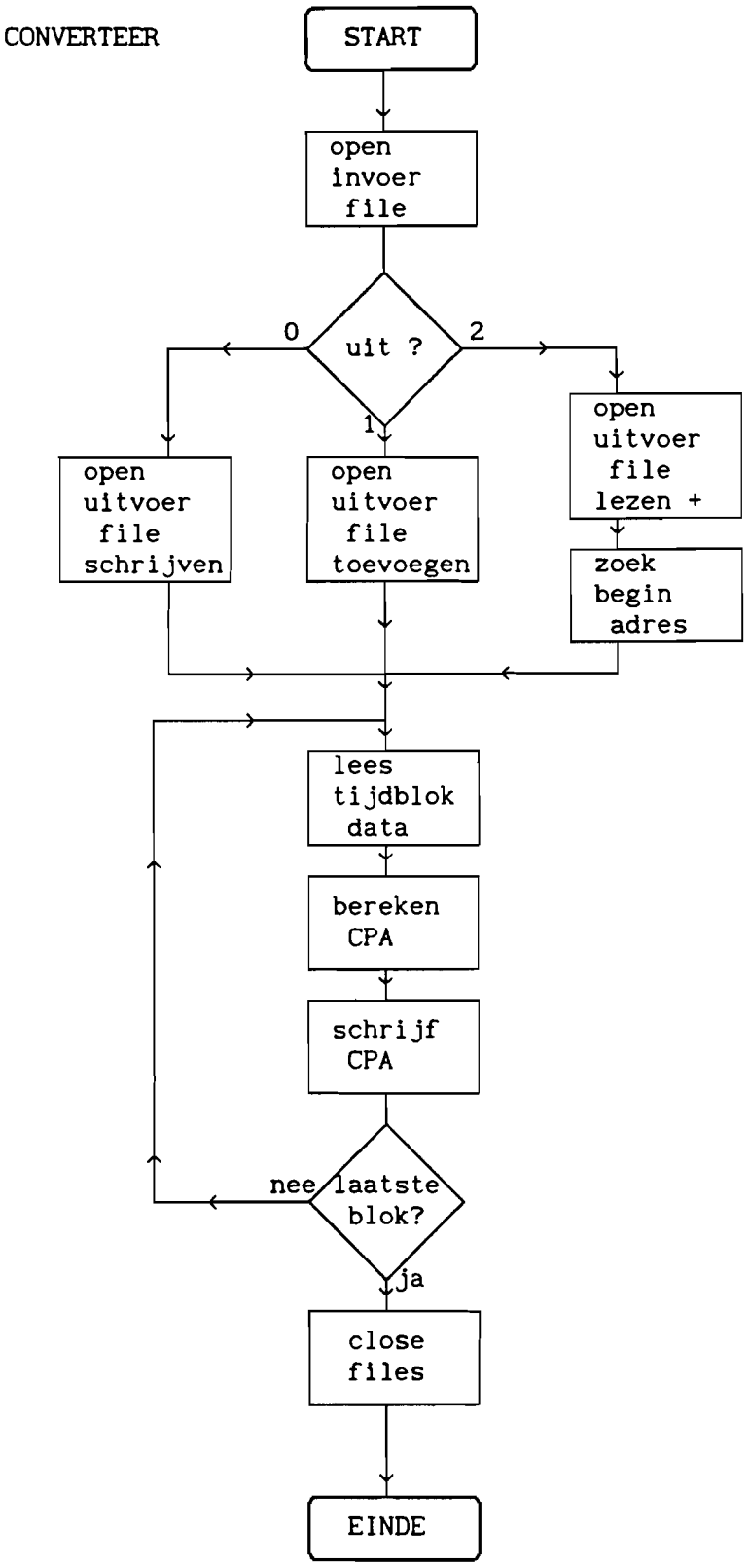


fig. 4.6 Flowschema van de functie converteer

4.3 VenL

De functie VenL berekent uit radiometermetingen en meteorologische gegevens de hoeveelheid waterdamp per vierkante meter, V, en de hoeveelheid water per vierkante meter, L, op het satellietpad. Hiertoe moet eerst een eventdrempel worden vastgesteld. Deze geeft een grenswaarde in dBs. Komt de, uit metingen van een van de radiometers berekende, demping boven deze grens, dan wordt dit resultaat als niet betrouwbaar meer beschouwd. De functie zal de default waarde op het scherm zetten. De gebruiker kan, als hij dat nodig acht, de waarde veranderen. Bij opnieuw aanroepen van de functie zal steeds weer de default waarde worden aangenomen. Daarna vraagt de functie het aantal radiometers dat gebruikt wordt. Afhankelijk van dit aantal worden verschillende functies aangeroepen. Is maar één radiometer aanwezig dan wordt VenL1 aangeroepen. Deze berekent V en L uit de metingen van één radiometer, de relatieve vochtigheid en de omgevingstemperatuur. Bij twee radiometers wordt VenL2 aangeroepen. Deze berekent V en L direct uit de radiometer gegevens. Zijn er meer dan twee radiometers dan wordt VenLmeer aangeroepen. Deze geeft de mogelijkheid twee radiometers te selecteren waarna, met behulp van VenL2, V en L worden berekend. Na afloop wordt het beëindigen van de functie gemeld. Het flowschema van VenL wordt gegeven in fig. 4.7.

4.3.1 VenL1

De functie VenL1 berekent de waarden van V en L uit de metingen van één radiometer, de omgevingstemperatuur en de relatieve vochtigheid. Hiertoe roept de functie eerst de functies parameters_VenL1 en checkfile_VenL1 aan. Deze hebben dezelfde taak en opbouw als de functies parameters_CPA en checkfile_CPA. De gebruikte parameters verschillen echter en staan in de file VenL.par en ook de afmetingen van de in- en uitvoerfile zullen verschillen. Voor flowschema's zie fig. 4.4 en fig. 4.5. Daarna wordt berekenL uitgevoerd. Deze berekent eerst de benodigde parameters, vervolgens V en daarmee en met de metingen L en bergt V en L op in de uitvoerfile.Als laatste wordt het beëindigen van de functie gemeld. Het flowschema van VenL1 wordt gegeven in fig. 4.8.

De functie berekenL berekent voor ieder tijdblok de benodigde parameters, de waarde van V en daarmee de waarde van L en schrijft de waarden van V en L naar de uitvoerfile. Hiertoe opent de functie eerst de invoerfile en dan de uitvoerfile zoals wordt bepaald door de waarde van 'uit' (zie ook de functie converteer) en voert vervolgens een do loop uit waarin eerst de meetwaarden worden gelezen en geconverteerd naar demping en dan de frequentieafhankelijke parameters, A, B, C en D uit formule 1.20, door aanroepen van de functie radiometervariabelen, en V uit de formules 1.21 en 1.22, door aanroepen van de functie berekenV, worden berekend. Hiermee wordt dan de waarde van L berekend uit formule 1.20 en de waarden van V en L worden in de uitvoerfile gezet. Blijkt de, uit de radiometermeting berekende, waarde voor de demping boven de eventdrempel te liggen, dan krijgen V en L niet realistische, i.c. negatieve, waarden. De do loop wordt beëindigd als het laatste tijdblok is afgehandeld. De functies radiometervariabelen en berekenV voeren eenvoudige berekeningen uit. Hiervan zijn dan ook geen flowschema's gegeven. Het flowschema van berekenL wordt gegeven in fig. 4.9.

4.3.2 VenL2

De functie VenL2 berekent de waarden van V en L uit de metingen van twee radiometers. Hiertoe worden allereerst de functies parameters_VenL2 en checkfile_VenL2 aangeroepen. Opbouw en werking van deze functies zijn hetzelfde als bij CPA (zie fig. 4.4 en fig. 4.5), maar ze zijn afgestemd op de door VenL2 gebruikte parameters en files. Daarna wordt de functie berekenVenL aangeroepen. Deze berekent voor ieder tijdblok eerst de benodigde parameters en dan de waarden van V en L en zet deze in de uitvoerfile. Als laatste wordt het beëindigen van de functie gemeld. Het flowschema van de functie VenL2 wordt gegeven in fig. 4.10.

De functie berekenVenL berekent per tijdblok eerst de benodigde parameters en daarmee de waarden van V en L en zet deze in de uitvoerfile. Hiertoe opent de functie eerst de invoerfile en dan afhankelijk van de waarde van 'uit' de uitvoerfile (zie de functie converteer). Dan begint de functie een loop waarin eerst de

meetwaarden van één tijdblok worden gelezen, dan voor beide radiometerfrequenties de frequentieafhankelijke parameters A, B, C en D uit formule 1.20 worden berekend, door aanroepen van de functie radiometervariabelen2, en uit de meetwaarden met de parameters via de conversieformules de demping wordt berekend. V en L worden hieruit berekend door het oplossen van twee vergelijkingen van het type 1.20 en deze resultaten worden in de uitvoerfile geschreven. Komt de berekende demping voor een van beide radiometers boven de eventdrempel dan krijgen V en L een negatieve waarde. De loop wordt beëindigd als het laatste opgegeven tijdblok is afgehandeld. Het flowschema van de functie berekenVenL wordt gegeven in fig. 4.11.

4.3.3 VenLmeer

De functie VenLmeer berekent de waarden van V en L als er meer dan twee radiometers aanwezig zijn. Hiertoe zal de functie eerst vragen twee radiometers te selecteren en meedelen dat hiermee dezelfde berekeningen zullen worden uitgevoerd als waren er slechts twee radiometers. Hierna zal de functie dan ook daadwerkelijk hetzelfde doen als de functie VenL2. Voor een beschrijving zie 4.3.2 en de figuren 4.10 en 4.11.

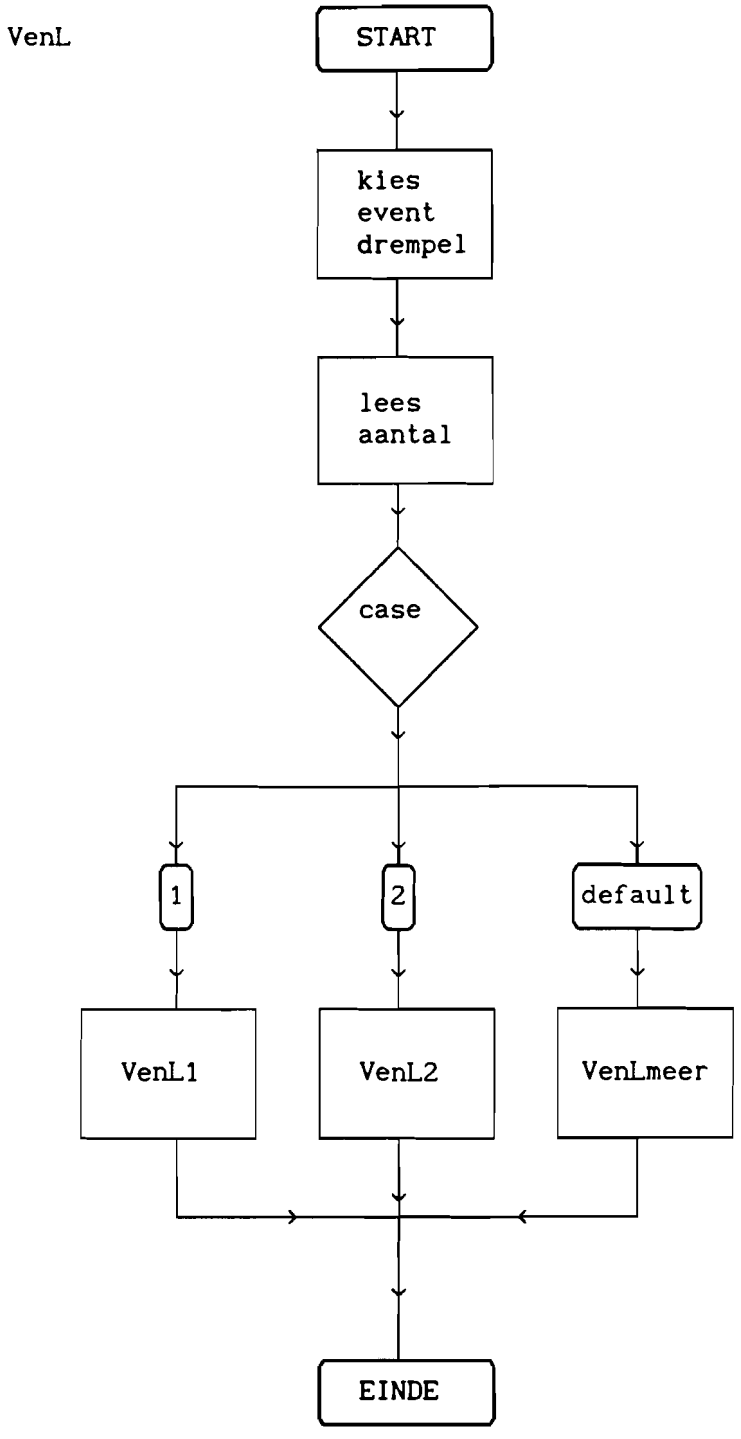


fig. 4.7 Flowschema van de functie VenL

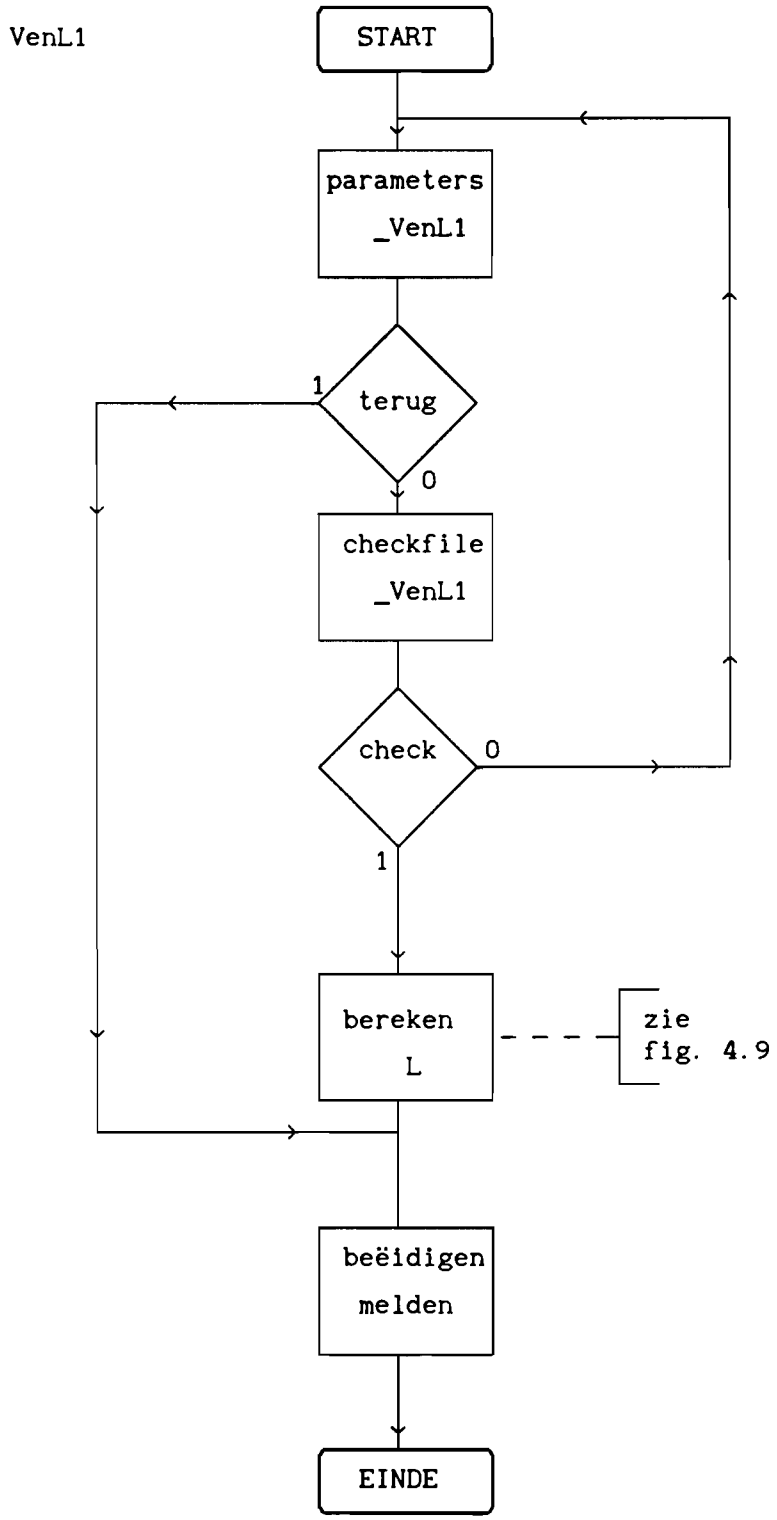


fig. 4.8 Flowschema van de functie VenL1

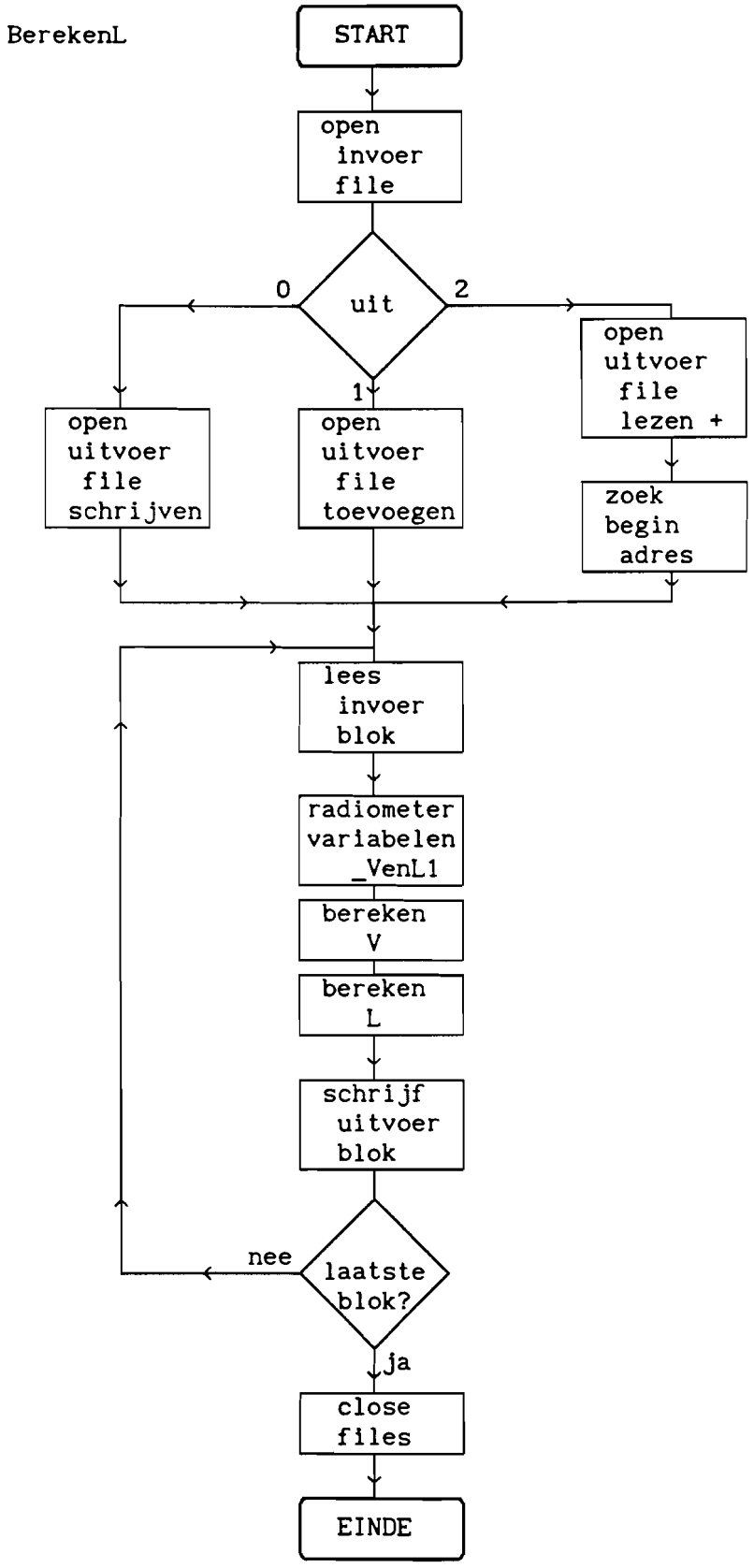


fig. 4.9 Flowschema van de functie berekenL

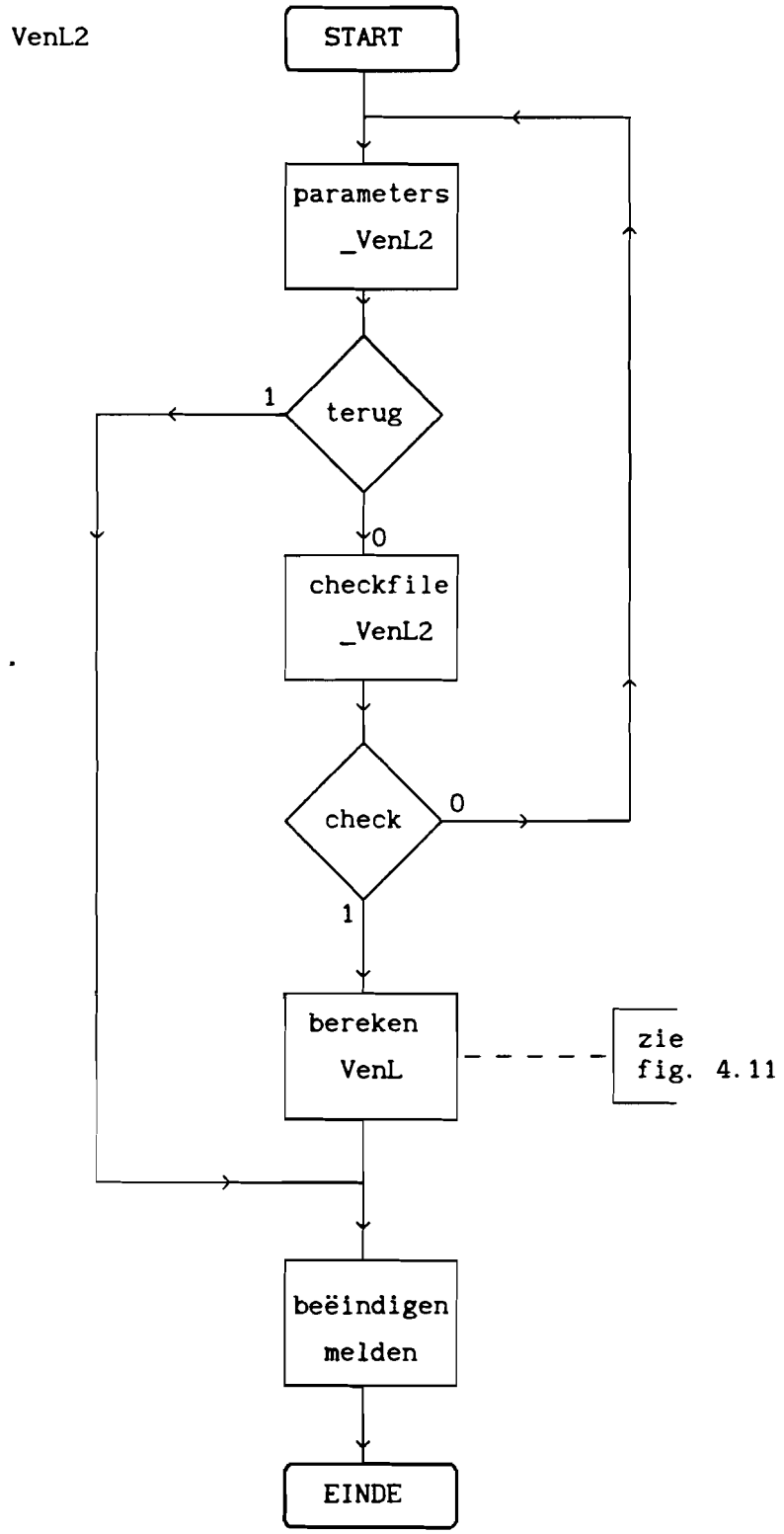


fig. 4.10 Flowschema van de functie VenL2

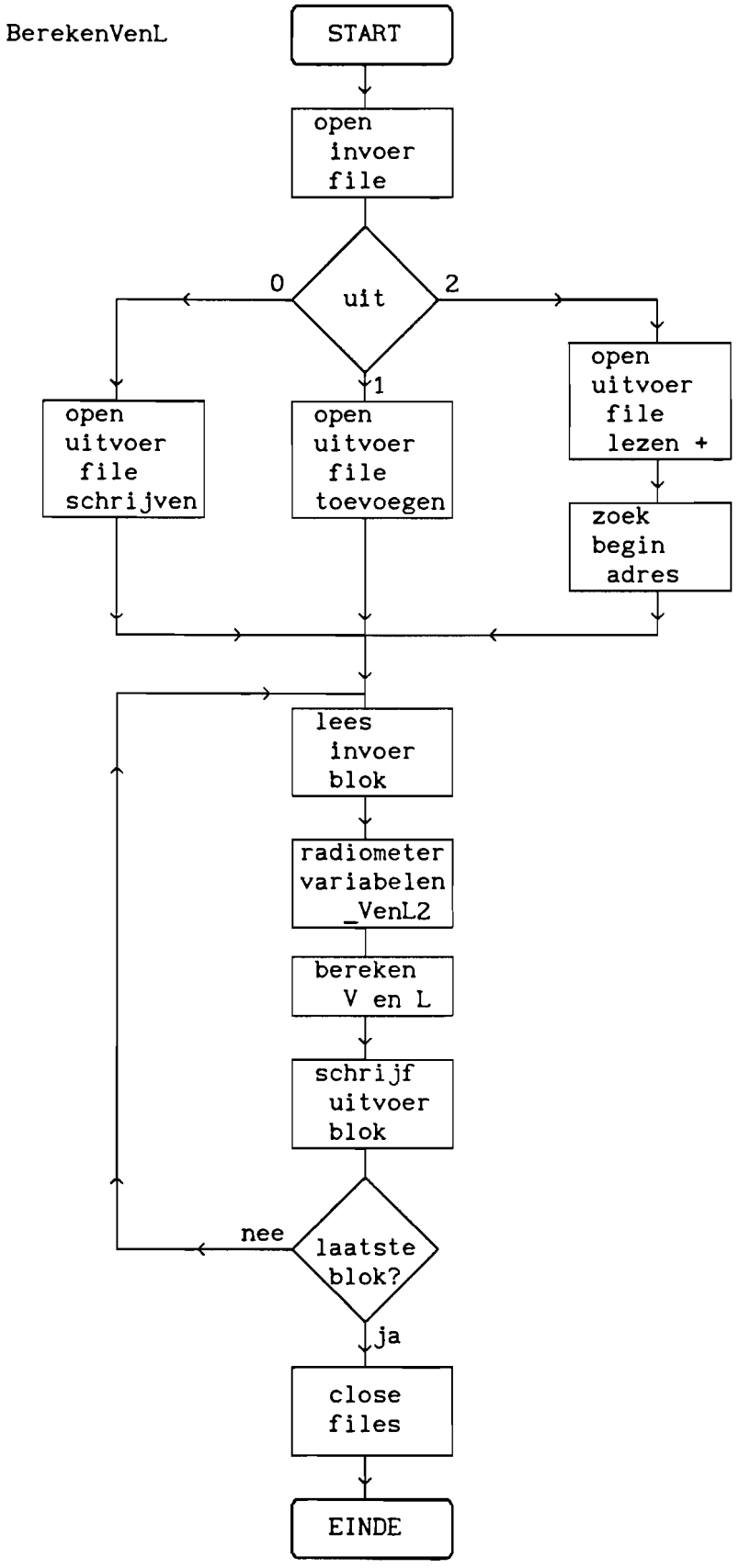


fig. 4.11 Flowschema van de functie berekenVenL

4.4 Omschaling

De functie Omschaling berekent uit de waarden van V en L de demping die door de atmosfeer wordt veroorzaakt bij een op te geven frequentie. Hiertoe worden eerst de functies parameters_omschaling en checkfile_omschaling aangeroepen. Deze roepen de bijbehorende parameters, waaronder de frequentie waarnaar de demping moet worden omgeschaald, op ter inspectie en wijziging en checken de in- en uitvoerfile zoals beschreven bij de gelijknamige functies bij CPA (zie ook fig. 4.4 en fig.4.5). Dan wordt de functie demping aangeroepen. Deze berekent de benodigde parameters en daarmee en met de ingelezen waarden van V en L de demping bij de opgegeven frequentie, via formule 1.20, en zet deze waarden in de uitvoerfile. Tenslotte meldt de functie aan de gebruiker dat de bewerking beëindigd is. Het flowschema van de functie Omschaling wordt gegeven in fig. 4.12.

De functie demping berekent de benodigde parameters en daarmee en met de waarden van V en L de atmosferische demping bij de opgegeven frequentie. Hiertoe opent de functie eerst de invoerfile en dan afhankelijk van de waarde van 'uit' de uitvoerfile, zoals beschreven bij de functie converteer. Dan start de functie een loop waarin telkens een tijdblok wordt gelezen, de frequentieafhankelijke parameters uit formule 1.20, door aanroepen van de functie dempingvariabelen, worden berekend, hiermee en met de waarden van V en L, via formule 1.20, de demping wordt berekend en deze dempingwaarde in de uitvoerfile wordt gezet. Hebben V en L een negatieve waarde, dan lag de, uit de radiometer(s) berekende, demping boven de eventdrempel en kan ook voor de omgeschaalde demping geen zinvolle waarde worden bepaald. De demping krijgt nu een zeer grote negatieve waarde om aan te geven dat er geen betekenisvolle waarde kon worden afgeleid. De loop wordt beëindigd als het laatste opgegeven blok is behandeld. Als laatste sluit dan de functie de files. Het flowschema van de functie demping wordt gegeven in fig. 4.13.

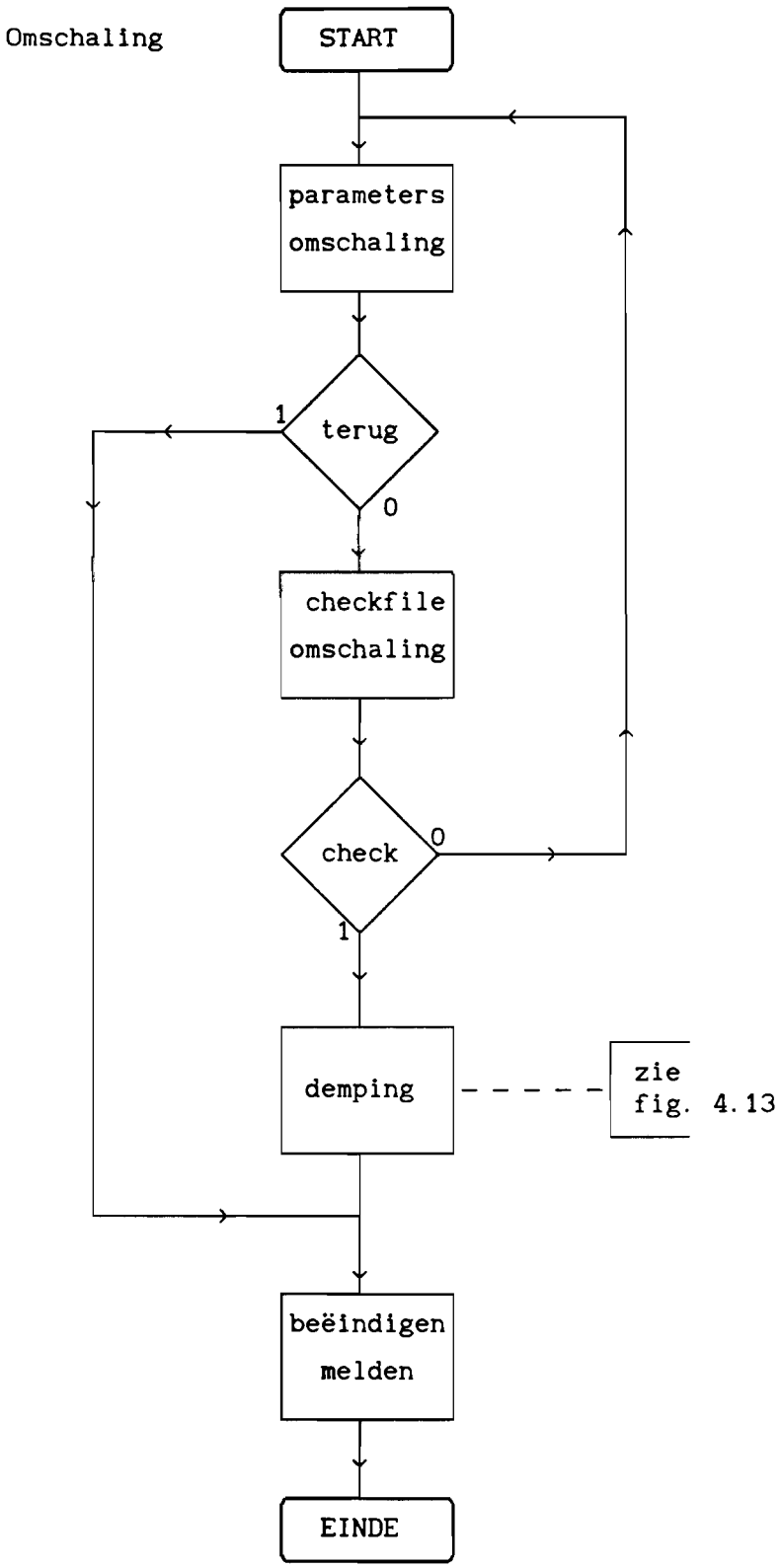


fig. 4.12 Flowschema van de functie Omschaling

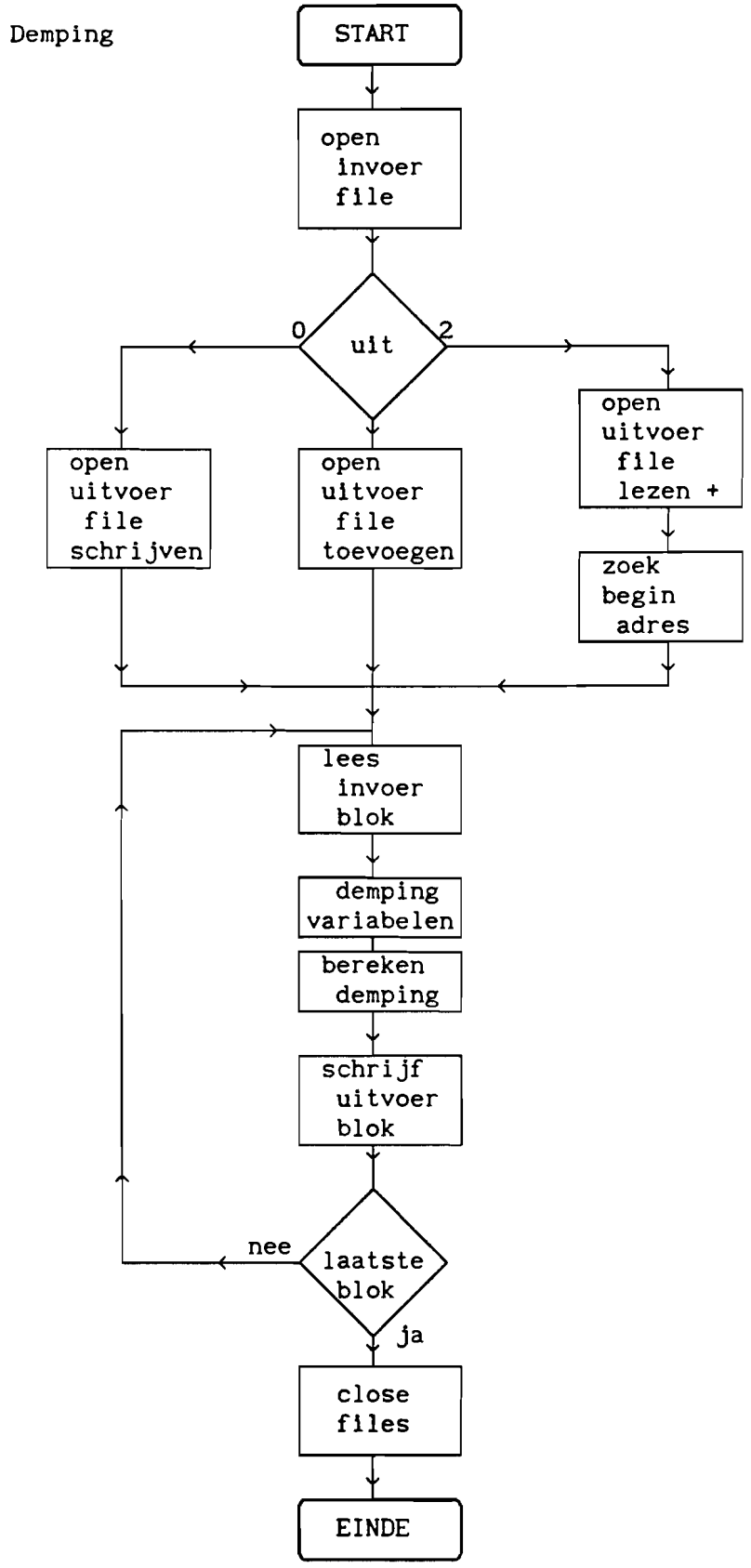


fig. 4.13 Flowschema van de functie demping

4.5 Template

De functie Template maakt een schatting voor de systeemcurve tijdens een event door middeling van de systeemcurves van voorafgaande dagen voor dezelfde periode. Hiertoe vraagt de functie allereerst het aantal files dat men wil gebruiken voor de middeling. Afhankelijk van dit aantal worden de parameters op het scherm gezet ter inspectie en wijziging door de functie parameters_template en worden de invoerfiles en de uitvoerfile gecheckt door checkfile_template, waarbij opbouw en functioneren van deze functies gaat zoals beschreven bij de functie CPA (zie ook de figuren 4.4 en 4.5). Daarna wordt de functie gemiddelde aangeroepen. Deze berekent voor ieder tijdblok het gemiddelde van de waarden van de systeemcurven van de voorafgaande dagen en zet het resultaat in de uitvoerfile. Als laatste wordt dan het gereedkomen van de functie gemeld. Een flowschema van de functie Template wordt gegeven in fig. 4.14.

De functie gemiddelde berekent voor ieder punt van de te schatten systeemcurve het gemiddelde van de systeemcurves van de voorgaande dagen en zet dit resultaat in de uitvoerfile. De functie opent eerst, afhankelijk van het aantal invoerfiles, de invoerfiles en, afhankelijk van de waarde van 'uit', de uitvoerfile (zie ook de functie converteer). Dan wordt er in een loop per tijdblok van iedere invoerfile de waarde gelezen, het gemiddelde hiervan berekend en deze gemiddelde waarde in de uitvoerfile gezet. Is een van de dempingwaarden een groot negatief getal, doordat geen zinvolle waarde kon worden berekend (zie **4.4**), dan wordt deze waarde niet meegenomen in de middeling en wordt automatisch gemiddeld over de overige wel zinvolle waarden. Deze loop wordt beëindigd als deze handelingen voor het laatste opgegeven tijdblok zijn verricht. Ten slotte worden dan alle invoerfiles en de uitvoerfile gesloten. Een flowschema van de functie gemiddelde wordt gegeven in fig. 4.15.

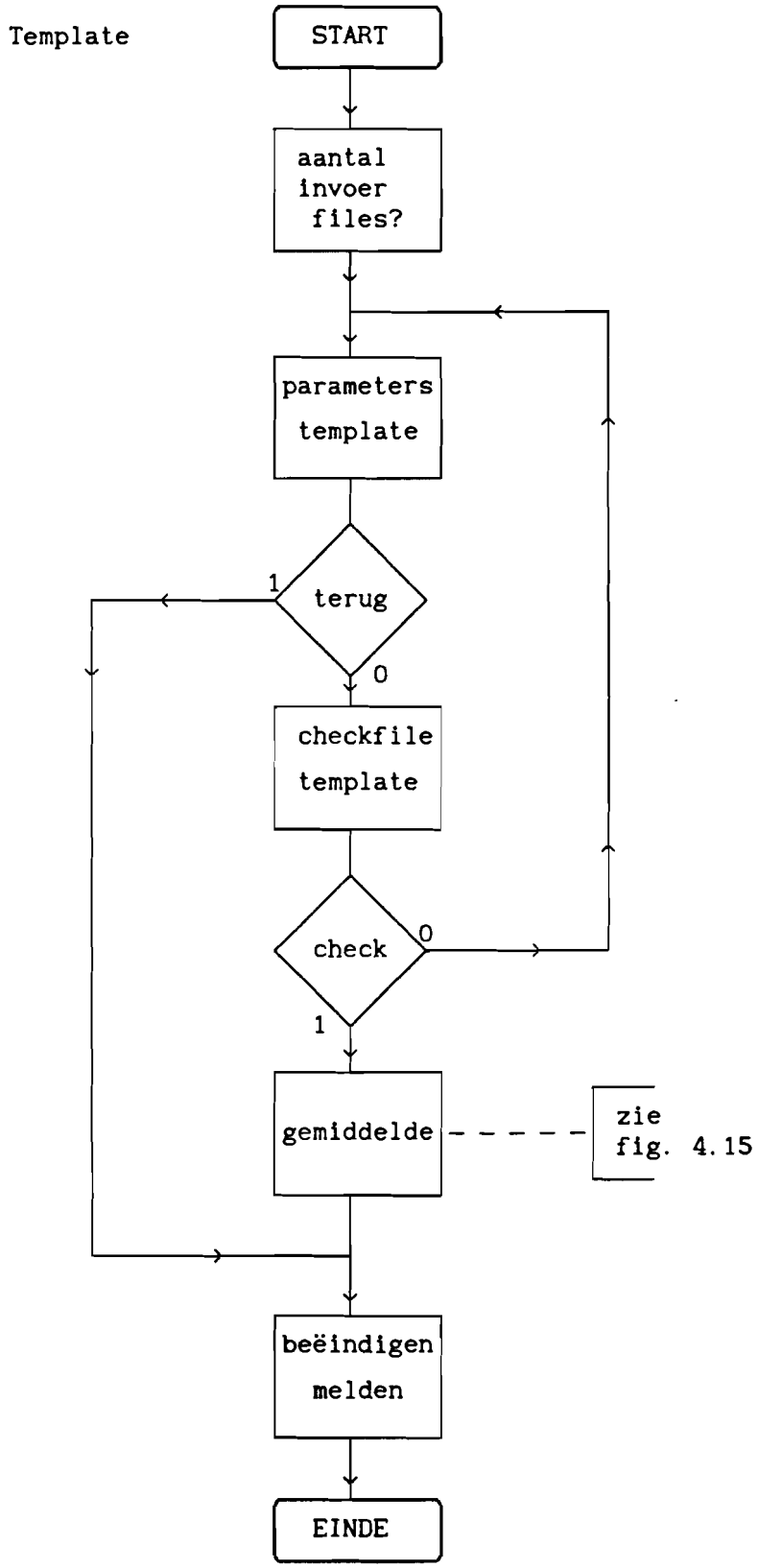


fig. 4.14 Flowschema van de functie Template

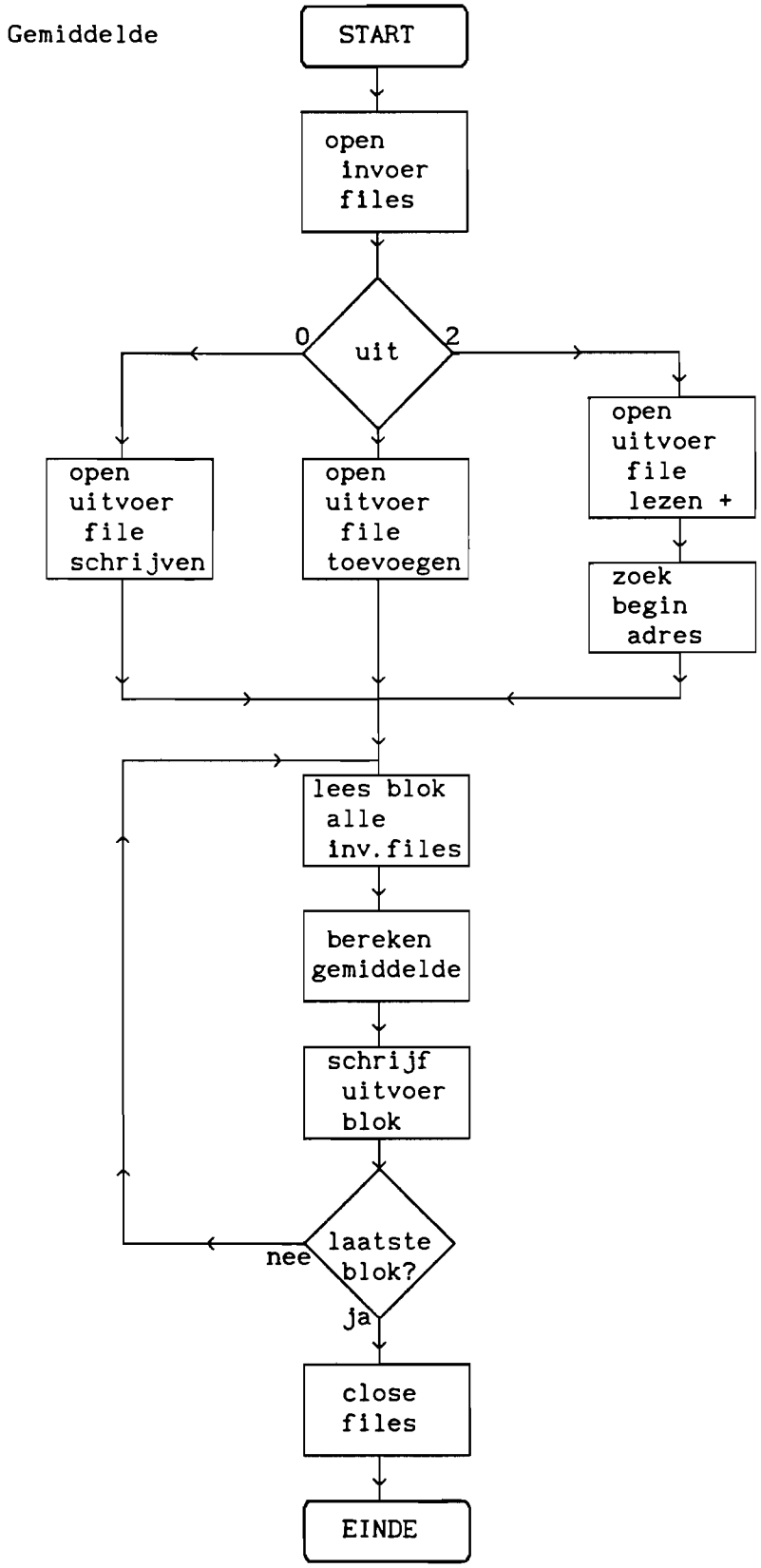


fig. 4.15 Flowschema van de functie gemiddelde

4.6 Verschil

De functie Verschil is een functie die op meerdere plaatsen in het programma te hulp geroepen kan worden. Zij berekent het verschil tussen de waarden in twee files over een zelfde periode in beide files. Bijvoorbeeld tussen de CPA waarden en de uit de radiometer metingen berekende waarden voor de atmosfeerdemping, wat dan een eerste schatting oplevert voor de systeemcurve. De functie zet eerst met behulp van de functie parameter_verschil de benodigde parameters op het scherm, ter inspectie en wijziging, en checkt met behulp van de functie checkfile_verschil de invoerfiles en de uitvoerfile. De werking van deze functies is zoals beschreven bij de functie CPA (zie ook de figuren 4.4 en 4.5). Dan wordt de functie aftrek aangeroepen. Deze trekt punt voor punt de waarden van de tweede invoerfile af van die van de eerste en bergt het resultaat in de uitvoerfile. Als laatste meldt de functie dat de bewerking is beëindigd. Een flowschema van de functie Verschil wordt gegeven in fig. 4.16.

De functie aftrek berekent punt voor punt het verschil tussen de waarden in de twee invoerfiles en zet het resultaat in de uitvoerfile. Hiertoe opent ze eerst de invoerfiles en, afhankelijk van de waarde van 'uit', de uitvoerfile (zie ook de functie converteer). Dan wordt, in een loop, steeds van beide invoerfiles hetzelfde tijdblok gelezen, wordt de waarde van de tweede file afgetrokken van die van de eerste en wordt het resultaat in de uitvoerfile gezet. De loop wordt beëindigd als het laatste opgegeven tijdblok is afgehandeld. Als laatste sluit de functie dan de invoerfiles en de uitvoerfile. Een flowschema van de functie aftrek wordt gegeven in fig. 4.17.

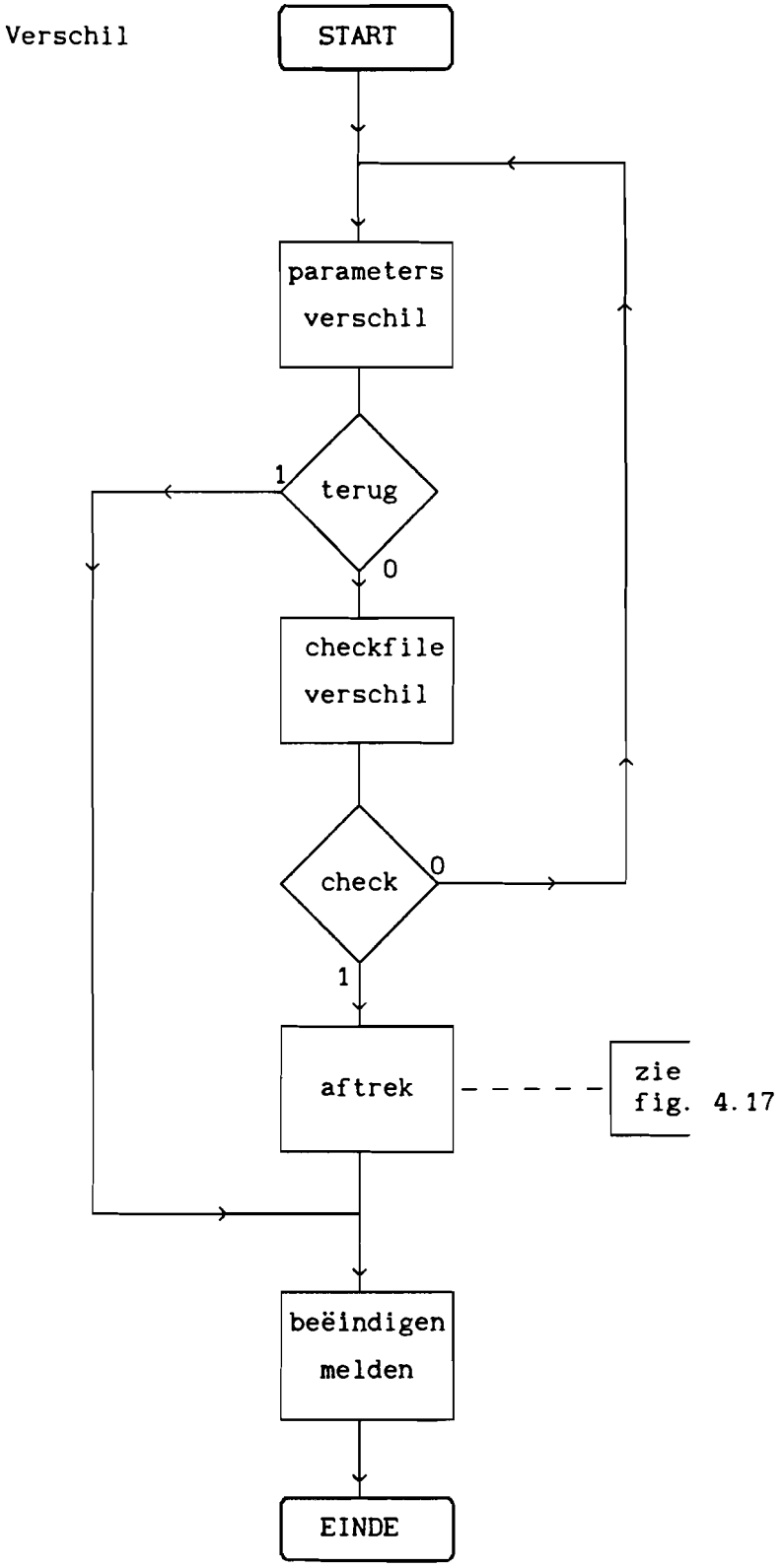


fig. 4.16 Flowschema van de functie Verschil

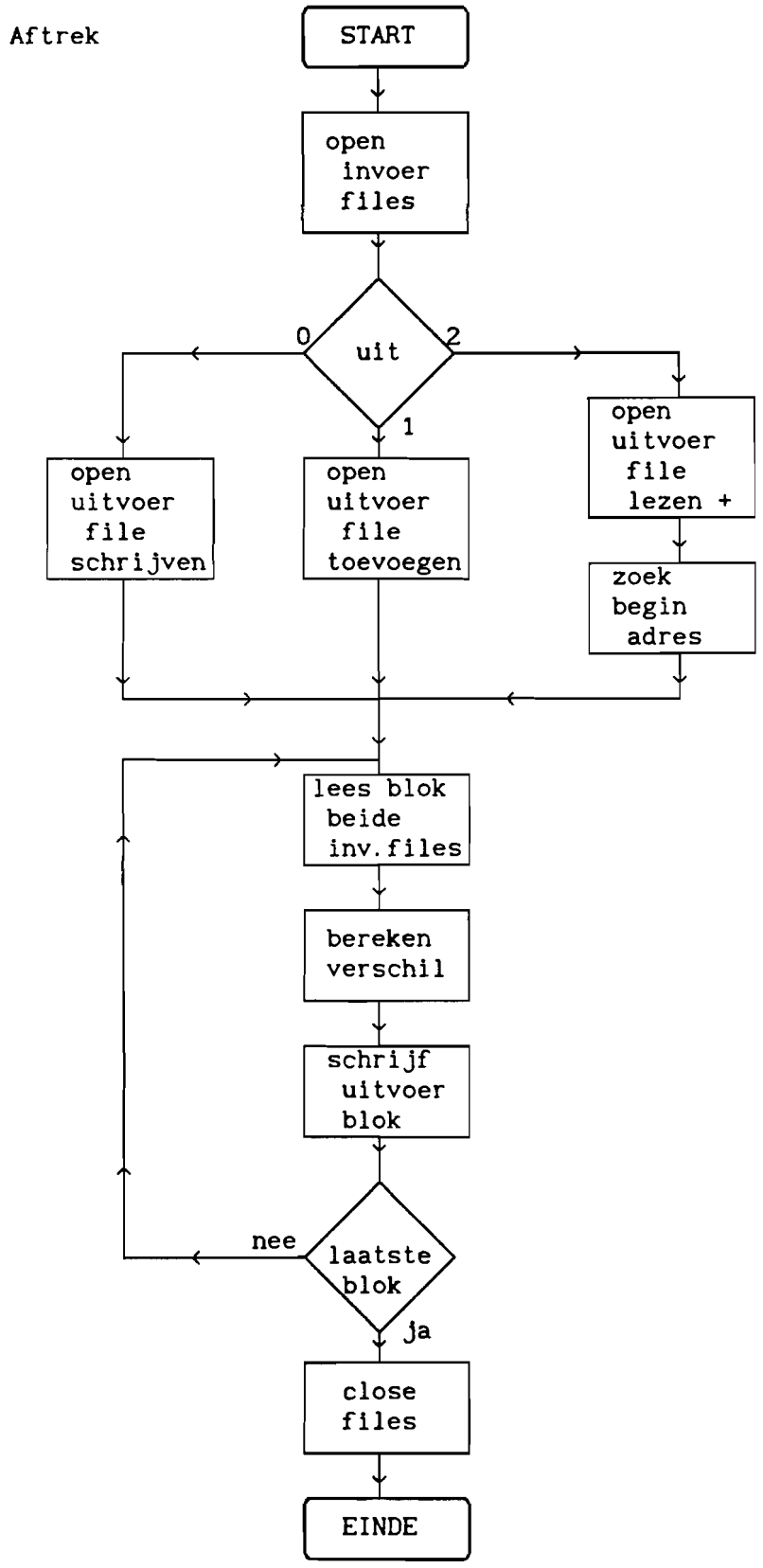


fig. 4.17 Flowschema van de functie aftrek

4.7 Grafiek

De functie grafiek kan naar keuze van een, twee of drie files een grafische weergave op het beeldscherm geven. Hiertoe vraagt de functie eerst hoeveel grafieken men wil en vervolgens van welke soort de grafieken zijn. Dan worden de parameters op het scherm gezet m.b.v. de functie parameters_grafiek. Deze werkt op eenzelfde manier als is beschreven bij de functie parameters_cpa (zie ook fig. 4.4). Daarna wordt de functie checkfile_grafiek aangeroepen. De werking hiervan is nagenoeg gelijk aan die van de functie checkfile_cpa, alleen hoeft de functie geen uitvoerfile te checken, maar wordt er wel gecheckt of de namen van de files overeenkomen met de opgegeven soort. Hiertoe is het nodig dat de namen van de files die door een bepaalde module als uitvoer worden geleverd steeds met dezelfde letters beginnen. Dit wordt gedaan omdat bij het tekenen van de grafieken ook vermeld wordt van welke soort een grafiek is. De functie kijkt of de eerste letters van de naam van de opgegeven file gelijk zijn aan een string die volgens hem bij dat soort files hoort. De functie checkfile_grafiek wordt zoveel maal uitgevoerd als er invoerfiles zijn. Tenslotte wordt de functie teken aangeroepen. Deze opent de invoerfiles tekent de assenkruisen berekent de functiewaarden en tekent deze op het beeldscherm en sluit na afloop de geopende files weer. Het flowschema van de functie grafiek wordt gegeven in fig. 4.18.

De functie teken verzorgt het daadwerkelijk op het scherm brengen van de grafieken en bijbehorende assenkruisen. Hiertoe verdeelt de functie eerst de beschikbare beeldruimte over het aantal te tekenen grafieken. Vervolgens wordt grafiek voor grafiek de invoerfile geopend, het assenkruis getekend en van de bijbehorende tekst voorzien. Het aantal waarden uit de file wordt verdeeld over het aantal beeldpunten van de grafiek en de grafiekwaarden worden berekend en punt voor punt getekend. Is de ene grafiek klaar dan wordt gekeken of er nog meer volgen. Zoniet dan worden de files gesloten en is de functie klaar. De grafieken blijven op het scherm staan totdat een toets wordt ingedrukt. Het flowschema van de functie teken wordt gegeven in fig. 4.19.

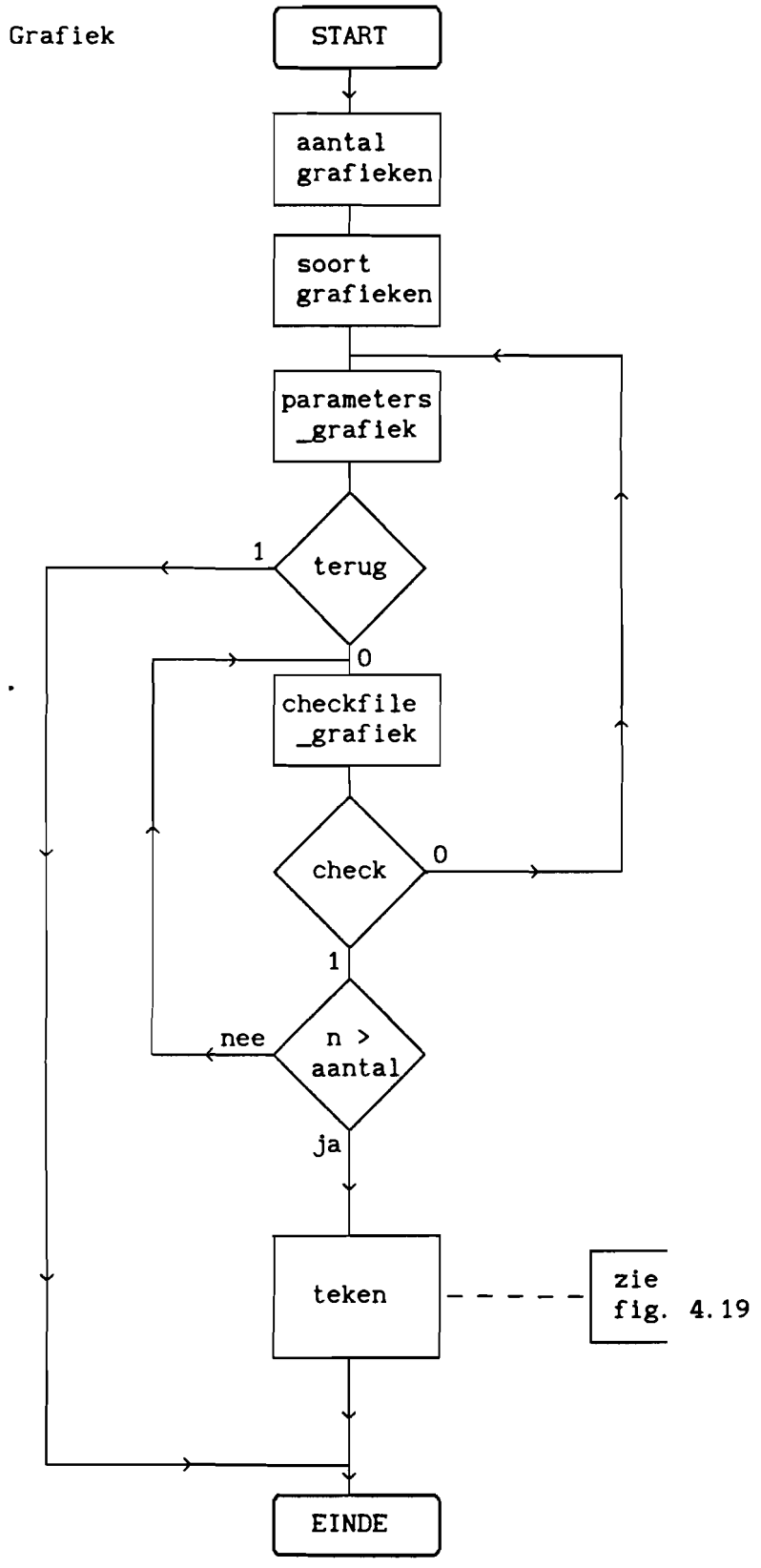


fig 4.18 Flowschema van de functie grafiek

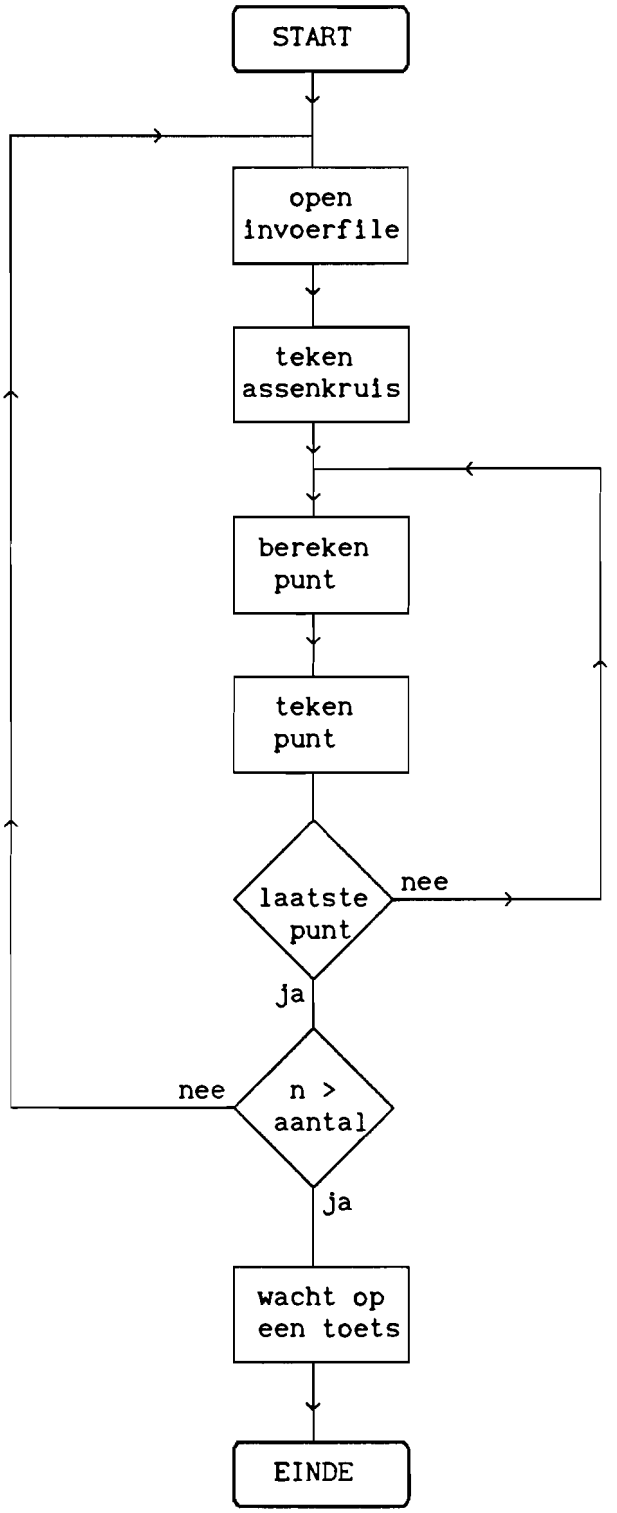


fig 4.19 Flowschema van de functie teken

5 OPZET VAN HET SOFTWARESYSTEEM VOOR HET KRUISPOLARISATIESIGNAAL

5.1 Algemeen

Het softwaresysteem moet niet alleen voor het hoofdpolarisatiesignaal maar ook voor het kruispolarisatiesignaal schattingen kunnen maken van de fouten om correcties mogelijk te maken. In hoofdstuk 2 is geschetst welke verstoringen er optreden voor het kruispolarisatiesignaal. Het maken van templates voor de kruispolarisatie berust nu op het gegeven dat de kruispolarisatie die gemeten wordt in afwezigheid van een event een maat is voor de door het systeem veroorzaakte kruispolarisatie.

Het systeem gaat nu als volgt te werk. Eerst wordt uit de meetgegevens de waarde bepaald van het hoofdpolarisatiesignaal (niet CPA), het kruispolarisatiesignaal en de fase. Hiermee wordt de waarde berekend van de XPD, de infase component van de XPD en de quadratuur component van de XPD (alle in vermogenseenheden en niet in dBs). Vervolgens wordt van de infase en quadratuur componenten van de XPD een template gemaakt op dezelfde wijze als bij het hoofdpolarisatiesignaal namelijk door middeling van een aantal nabij gelegen dagen. Is het resultaat dat zo wordt verkregen niet bevredigend dan kan door interpolatie een benadering worden gemaakt voor de template. De verkregen templates worden vervolgens van de infase en quadratuur componenten afgetrokken en uit de zo verkregen waarden worden de gecorrigeerde waarden voor de XPD en voor de fase berekend.

Om in iedere fase van de bewerking inspecties te kunnen uitvoeren zal er een grafische module zijn om de tijdreeksen weer te geven, maar tevens zal er een module zijn om de waarden van de infase component tegen die van de quadratuur component uit te zetten.

De opbouw van het systeem is zo dat voor iedere afgeronde bewerking een functie vanuit een keuzemenu kan worden aangeroepen. Het systeem ziet er dan uit zoals geschetst in figuur 5.1.

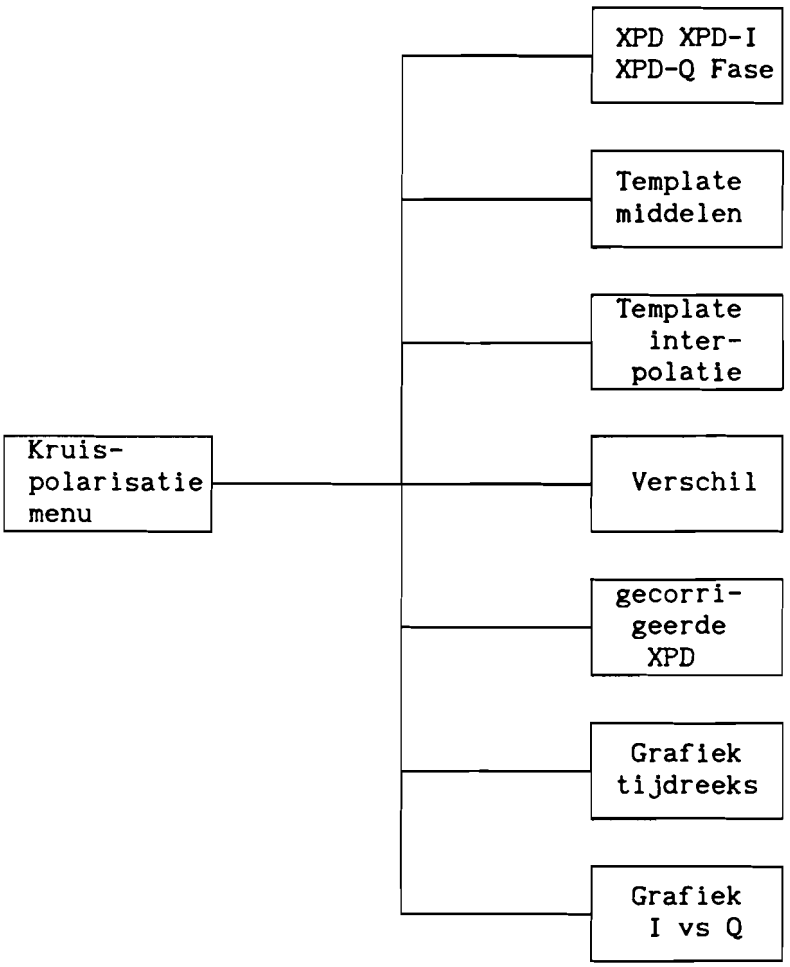


fig. 5.2 Opbouw van de kruispolarisatiesoftware

Ieder van de functies beschikt over een file met parameters. Na aanroep van de betreffende functie zal deze eerst zijn parameters op het scherm zetten en de mogelijkheid geven ze te wijzigen. De gewijzigde waarden worden opgeslagen in de parameterfile. Ieder van de functies checkt ook steeds of de opgegeven in- en uitvoerfiles wel bestaan en de juiste periodes bevatten.

Hieronder wordt voor ieder van de functies beschreven welke handelingen deze moet verrichten, welke parameters nodig zijn en wat de in- en uitvoer zal zijn.

5.2 De modules

5.2.1 XPD, XPD-I, XPD-Q en Fase

Deze module berekent uit de meetwaarden de XPD, de fase, de infase component van de XPD, XPD-I, en de quadratuur component van de XPD, XPD-Q. Ieder van deze grootheden wordt in een aparte file geborgen van een formaat dat door de module grafiek gebruikt kan worden. Voor de bepaling van de XPD wordt eerst de sterkte van het hoofdpolarisatiesignaal en van het kruispolarisatiesignaal bepaald en vervolgens met formule (2.2) de XPD berekend. Hiertoe doet de module het volgende.

-lezen van de file met meetwaarden
-berekenen hoofdpolarisatiesignaal
-berekenen kruispolarisatiesignaal
-berekenen XPD
-berekenen verschilfase
-berekenen infase component van XPD
-berekenen quadratuur component van XPD
-opbergen XPD XPD-I XPD-Q en fase ieder in een file

Parameters
-conversiegegevens voor berekening hoofdpolarisatiesignaal
-conversiegegevens voor berekening kruispolarisatiesignaal
-conversiegegevens voor berekening fase
-de namen van de in- en uitvoerfiles
-begin- en eindtijdstip

Invoer
-file met meetwaarden

Uitvoer
-file met XPD waarden
-file met XPD-I waarden
-file met XPD-Q waarden
-file met fase waarden

5.2.2 Template door middeling

Hiervoor wordt de template module van het hoofdpolarisatie gedeelte aangeroepen. De gebruiker dient er zelf zorg voor te dragen dat de juiste files als invoer worden aangeboden (zie **3.2.4**).

5.2.3 Template door interpolatie

Blijkt bij grafische inspectie dat de template in een bepaald gebied niet voldoet, dan kan voor dat gebied een lineaire interpolatie gemaakt worden. De gebruiker geeft begin- en eindtijdstip op voor het gebied waarover hij een interpolatie wil hebben. De module neemt de waarde bij het beginpunt en de waarde bij het eindpunt en berekent voor de tussenliggende punten zodanige waarden dat een rechte lijn ontstaat.

Parameters
-Invoerfilenaam
-begin- en eindtijd

Invoer
-Te corrigeren templatefile

Uitvoer
-Gecorrigeerde templatefile

5.2.4 Verschil

Om tijdreeksen van elkaar af te trekken wordt gebruik gemaakt van de module verschil uit het hoofdpolarisatie gedeelte (zie **3.2.5**)

5.2.5 Gecorrigeerde XPD

Door aftrekken van de gemaakte templates van de infase en quadratuur componenten van de XPD ontstaan gecorrigeerde waarden voor deze

grootheden. Uit deze gecorrigeerde waarden is nu de gecorrigeerde XPD af te leiden met de formule $XPD=\sqrt{(XPD-I)^2+(XPD-Q)^2}$. Deze methode is voor de enkelvoudig gepolariseerde bakens de enige mogelijkheid. Voor het, zowel horizontaal als verticaal gepolariseerde, 20 GHz baken staat echter nog een andere mogelijkheid open. Deze methode, die wordt beschreven in [1], berust op het gebruik van een zogenaamde cancellation matrix. De theorie hiervoor wordt verder uitgewerkt in appendix-L. Tevens berekent de module de nieuwe waarde voor de fase, die wordt gegeven door Fase=arctan(XPD-Q/XPD-I).
De module doet nu achtereenvolgens.
-inlezen file met gecorrigeerde XPD-I waarden
-inlezen file met gecorrigeerde XPD-Q waarden
-berekenen gecorrigeerde XPD waarden
-berekenen gecorrigeerde Fase waarden
-gecorrigeerde XPD waarden en Fase waarden
in de uitvoerfiles schrijven

Parameters
-namen van de in- en uitvoerfiles
-begin- en eindtijdstip

Invoer
-file met gecorrigeerde XPD-I waarden
-file met gecorrigeerde XPD-Q waarden

Uitvoer
-file met gecorrigeerde XPD waarden
-file met gecorrigeerde Fase waarden

5.2.6 Grafiek van een tijdreeks

Voor het grafisch weergeven van tijdreeksen wordt gebruik gemaakt van de module grafiek van het hoofdpolarisatie gedeelte.

5.2.7 Grafiek van I- versus Q-component

Deze module moet de mogelijkheid geven de I- tegen de Q-component van de XPD uit te zetten. De gebruiker geeft de periode op waarvoor hij de weergave wil hebben. De module gaat naar de grafische mode en tekent, punt voor punt, het verloop van de Q-component als functie van de I-component. Hiertoe doet de module achtereenvolgens.
-lezen XPD-I file
-lezen XPD-Q file
-berekenen plaats van de punten op het beeldscherm
-weergeven van de punten op het beeldscherm

Parameters
-de namen van de invoerfiles
-begin- en eindtijdstip
-de maximaal weer te geven waarde

Invoer
-file met XPD-I waarden
-file met XPD-Q waarden

Uitvoer
-grafiek naar het beeldscherm

6 GEDETAILLEERDE BESCHRIJVING VAN DE SOFTWARE VOOR HET KRUISPOLARISATIESIGNAAL

Bij de gedetailleerde beschrijving van de software wordt uitgegaan van de opbouw zoals geschetst in fig. 5.1. Ieder van de genoemde modules wordt zo beschreven dat hiermee overgegaan kan worden tot het daadwerkelijk schrijven van het programma onderdeel. Omdat de opzet van het geheel modulair is kan voor ieder onderdeel afzonderlijk vanuit de gedetailleerde beschrijving de software geschreven worden. Van alle belangrijke programmadelen wordt een flowschema gegeven. Voor deze beschrijving is gebruik gemaakt van [25,26].

6.1 Het Kruispolarisatiemenu

Dit onderdeel bevat de keuze module voor het gehele kruispolarisatie gedeelte van het programma. Het wordt aangeroepen vanuit het hoofdprogramma. Van hieruit moet gekozen worden welk onderdeel wordt uitgevoerd, waarna teruggekeerd wordt naar dit keuzemenu. Hiertoe wordt de functie xpdmenu aangeroepen. Deze zet, grafisch, de keuze mogelijkheden op het scherm. Met behulp van de up- en downpijlen kan de keuze worden gemaakt en daarna met de entertoets geactiveerd. Er zijn acht keuze mogelijkheden: de zeven programma onderdelen en de mogelijkheid "beëindig het programma". Xpdmenu kent aan iedere keuze een getalwaarde toe die wordt teruggeven. Een case-statement selecteert met dit getal het juiste programma onderdeel. Om te zorgen dat steeds wordt teruggekeerd naar het keuzemenu wordt de functie xpdmenu binnen een do-loop uitgevoerd. De keuze "beëindig het programma" levert de voorwaarde om de do-loop te verlaten, waarna vanzelf wordt teruggekeerd naar het hoofdprogramma. De opbouw van dit programmaonderdeel is hetzelfde als geschetst voor het hoofdpolarisatiemenu (zie **4.1.1** en de figuren 4.1 en 4.2).

6.2 XPD

De functie XPD berekent uit de meetwaarden de XPD, de fase, de infase component van de XPD, XPD-I, en de quadratuur component van de XPD, XPD-Q. Hiertoe roept de functie eerst de functie parameters_xpd aan. Deze zet de benodigde parameters op het scherm en geeft de mogelijkheid deze te wijzigen. Vervolgens wordt de functie checkfile_xpd aangeroepen. Deze checkt de in -en uitvoerfiles. Beide werken zoals beschreven bij CPA (zie **4.2** en fig. 4.4 en fig. 4.5). De daadwerkelijke berekeningen worden dan uitgevoerd door de functie berekenxpd. Deze schrijft ook de uitvoer naar de betreffende files. Als laatste wordt dan het gereedkomen van de functie gemeld. Het flowschema van de functie XPD wordt gegeven in fig. 6.1.

De functie berekenxpd berekent uit de meetwaarden de XPD, de fase, de XPD-I en de XPD-Q. Hiertoe wordt eerst de invoerfile geopend en dan afhankelijk van de waarde van 'uit' de verschillende uitvoerfiles. 'uit' Is nu niet een enkelvoudige variabele maar een array met voor iedere uitvoerfile een aparte waarde die de status van de bijbehorende uitvoerfile weergeeft. Nu wordt steeds een tijdblok ingelezen en met de in de parameterlijst opgegeven conversie gegevens worden de waarden van het hoofdpolarisatiesignaal, het kruispolarisatiesignaal en de fase bepaald. Dan wordt met formule (2.2) de XPD bepaald en met $XPD\text{-}I = \cos\phi \cdot XPD$ en $XPD\text{-}Q = \sin\phi \cdot XPD$ de infase en quadratuur componenten, waarbij voor de berekening van XPD-I en XPD-Q de vermogenswaarde van de XPD genomen wordt en niet de dB waarde. Vervolgens worden de waarden van XPD, XPD-I, XPD-Q en fase ieder in een aparte file opgeslagen in een formaat dat kan worden gebruikt als invoer voor de functie grafiek van tijdreeksen. Deze berekening eindigt als het laatste opgegeven tijdblok is verwerkt. Als laatste sluit de functie dan alle geopende files. Het flowschema van de functie berekenxpd wordt gegeven in fig. 6.2.

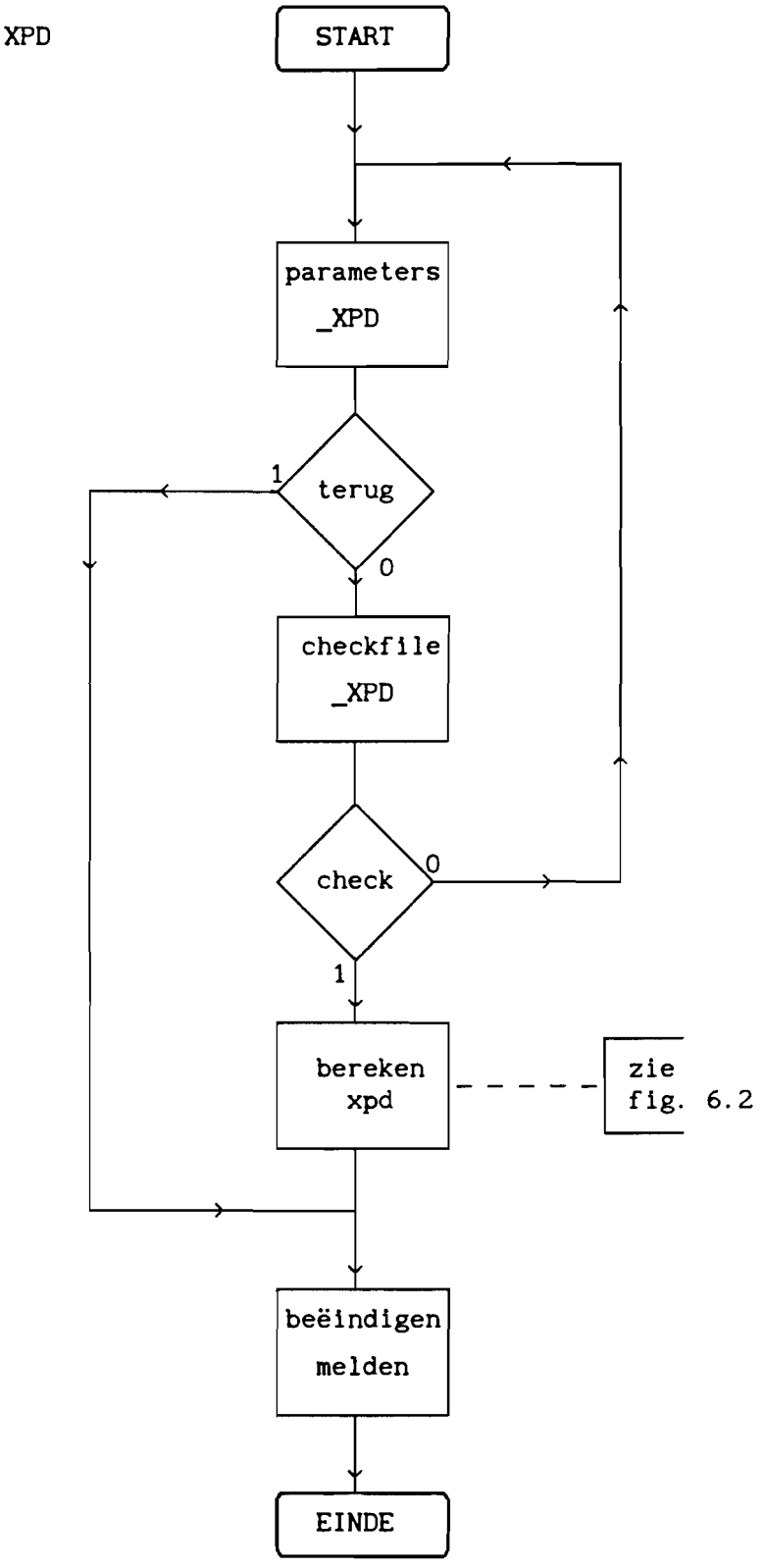


fig. 6.1 Flowschema van de functie XPD

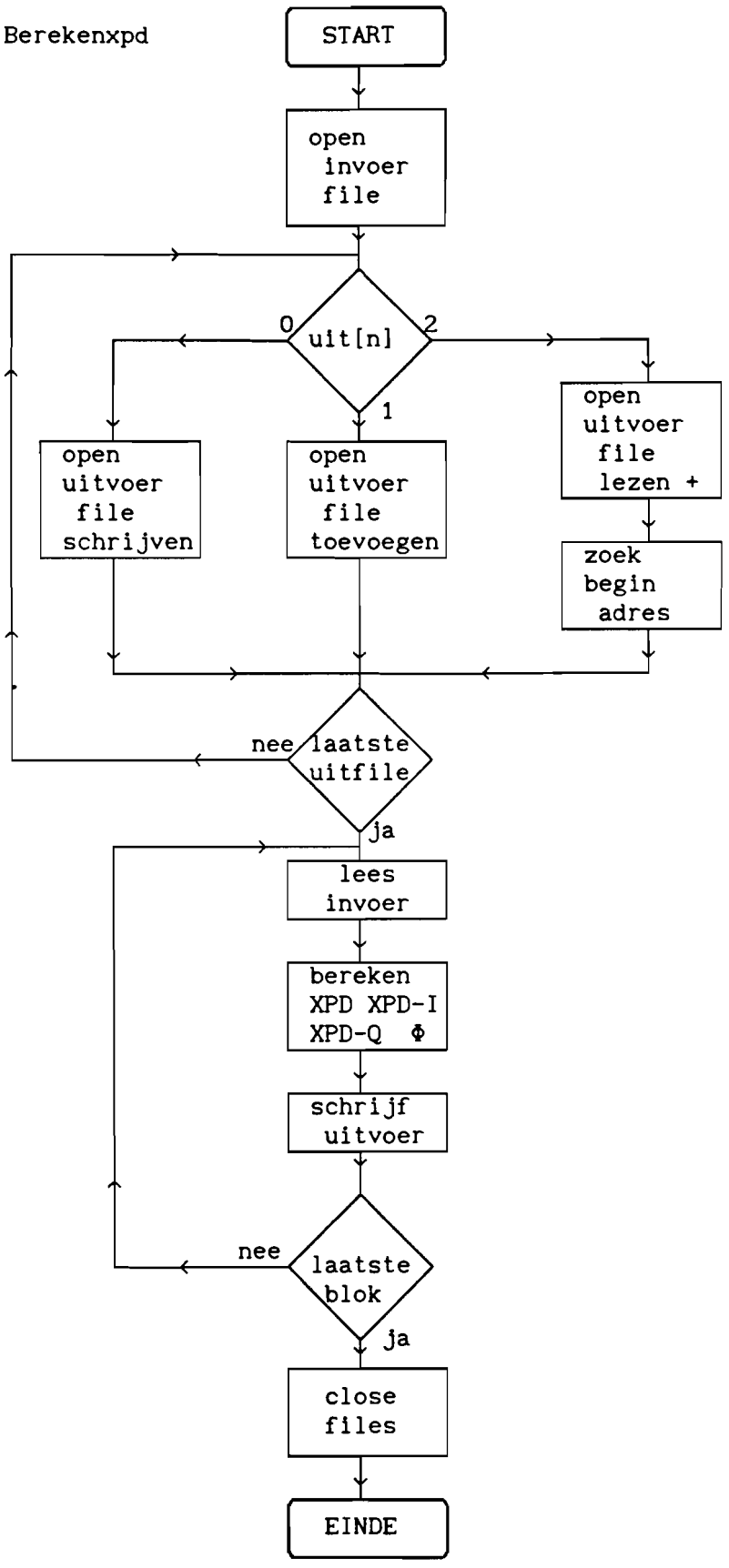


fig. 6.2 Flowschema van de functie berekenxpd

6.3 Template

Om templates te maken door middeling van resultaten van andere dagen wordt gebruik gemaakt van de functie template van het hoofdpolarisatie gedeelte. Deze is beschreven in **4.5** en de figuren 4.14 en 4.15.

6.4 Temp_interpolatie

De functie temp_interpolatie berekent uitgaande van de waarden van een opgegeven begin- en eindpunt de waarden voor de tussenliggende punten door lineaire interpolatie. Hiertoe worden eerst de functies parameters_temp_interpolatie en checkfile_temp_interpolatie aangeroepen. Deze zetten achtereenvolgens de parameters op het scherm om ze eventueel te wijzigen en checken de invoerfile. Beide functioneren zoals beschreven bij CPA (zie **4.2** en de figuren 4.4 en 4.5). Er hoeft echter geen uitvoerfile gecheckt te worden want de berekende waarden worden in de invoerfile gezet. De functie wordt namelijk alleen gebruikt als template berekening door middeling voor een bepaalde periode niet voldoet. De functie wijzigt dan de template voor de periode die niet voldoet. Dan wordt de functie rechtelijn aangeroepen, die lineair interpoleert tussen de waarden bij begin- en eindpunt en de berekende waarden in de templatefile schrijft. Tenslotte wordt aan de gebruiker gemeld dat de bewerking beëindigd is. Het flowschema van de functie temp_interpolatie wordt gegeven in fig. 6.3.

De functie rechtelijn berekent door lineaire interpolatie de waarden tussen een opgegeven begin- en eindpunt. Hiertoe wordt eerst de te corrigeren file geopend. Dan worden de waarden van het opgegeven begin- en eindpunt gelezen, het verschil hiervan berekend, alsmede het aantal punten waarvoor een nieuwe waarde moet worden bepaald en de afstand tussen deze punten. Vervolgens wordt, in een loop, punt voor punt de nieuwe waarde berekend en in de file geschreven. De loop wordt beëindigd als het eindpunt bereikt is. Als laatste sluit de functie dan de invoerfile. Het flowschema van de functie rechtelijn wordt gegeven in fig. 6.4.

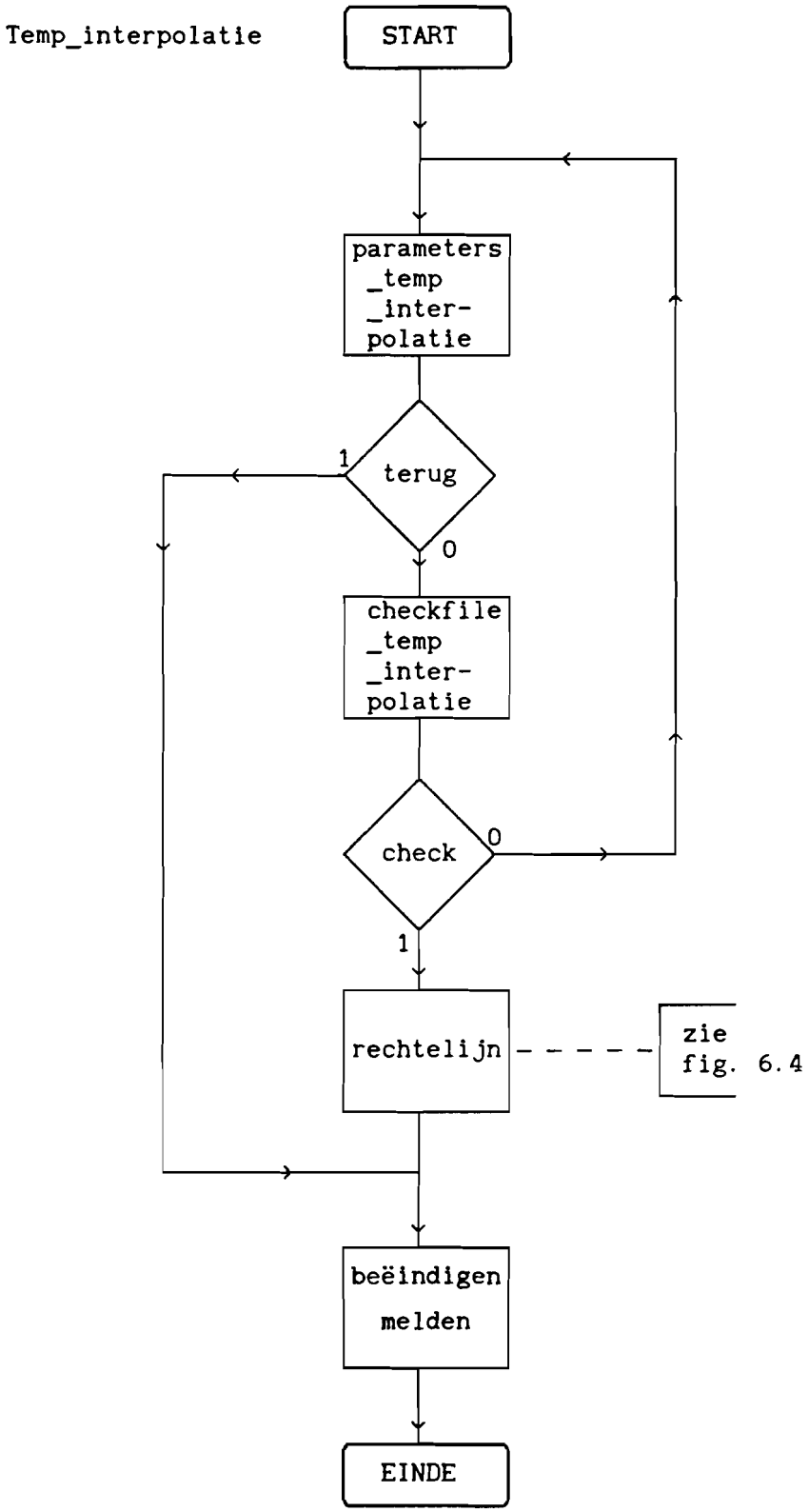


fig. 6.3 Flowschema van de functie temp_interpolatie

Rechtelijn

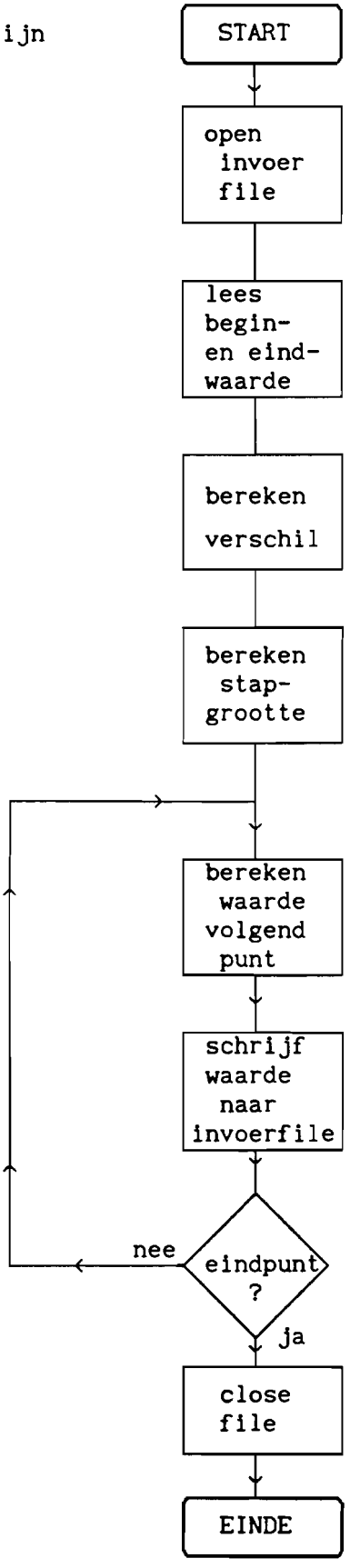


fig. 6.4 Flowschema van de functie rechtelijn

6.5 Verschil

Om het verschil tussen twee tijdreeksen te bepalen wordt gebruik gemaakt van de functie Verschil, zoals beschreven bij het hoofdpolarisatieprogramma. Deze beschrijving is te vinden in **4.6** en in de figuren 4.16 en 4.17.

6.6 Corr_xpd

De functie corr_xpd berekend de gecorrigeerde waarden voor de XPD en Fase uit de gecorrigeerde waarden van XPD-I en XPD-Q. De functie zet eerst met behulp van de functie parameters_corr_xpd de benodigde parameters op het scherm, ter inspectie en wijziging, en checkt met behulp van de functie checkfile_corr_xpd de invoerfiles en de uitvoerfile. De werking van deze functies is zoals beschreven bij de functie CPA (zie **4.2** en de figuren 4.4 en 4.5). Dan wordt de functie reconstrueer aangeroepen. Deze berekent punt voor punt de gecorrigeerde waarde van de XPD en Fase uit de gecorrigeerde I- en Q-component en zet het resultaat in de uitvoerfiles. Als laatste meldt de functie dat de bewerking is beëindigd. Een flowschema van de functie Corr_xpd wordt gegeven in fig. 6.5.

De functie reconstrueer berekent de gecorrigeerde XPD en Fase waarden uit de gecorrigeerde I- en Q-componenten en bergt de resultaten in de uitvoerfiles. Hiertoe opent de functie eerst de invoerfiles en, afhankelijk van de waarde van 'uit', de uitvoerfiles (zie de functie converteer in **4.2**). Dan wordt, in een loop, van beide invoerfiles steeds hetzelfde tijdblok gelezen. Uit de ingelezen waarden wordt met behulp van de formule $XPD=\sqrt{(XPD\text{-}I)^2+(XPD\text{-}Q)^2}$ de XPD berekend en met Fase=arctan(XPD-Q/XPD-I) de Fase en de uitkomsten worden in de uitvoerfiles geschreven. Tevens wordt gekeken of de waarde van van de demping voor het bijbehorende tijdstip een realistische waarde heeft (zie **4.4**) door kijken in de dempingfile. Zoniet dan worden ook de XPD en Fase onrealistisch. De loop wordt beëindigd als voor het laatste opgegeven tijdblok de berekening is voltooid. Als laatste sluit de functie dan de beide invoerfiles en de uitvoerfile. Een flowschema van de functie reconstrueer wordt gegeven in fig. 6.6.

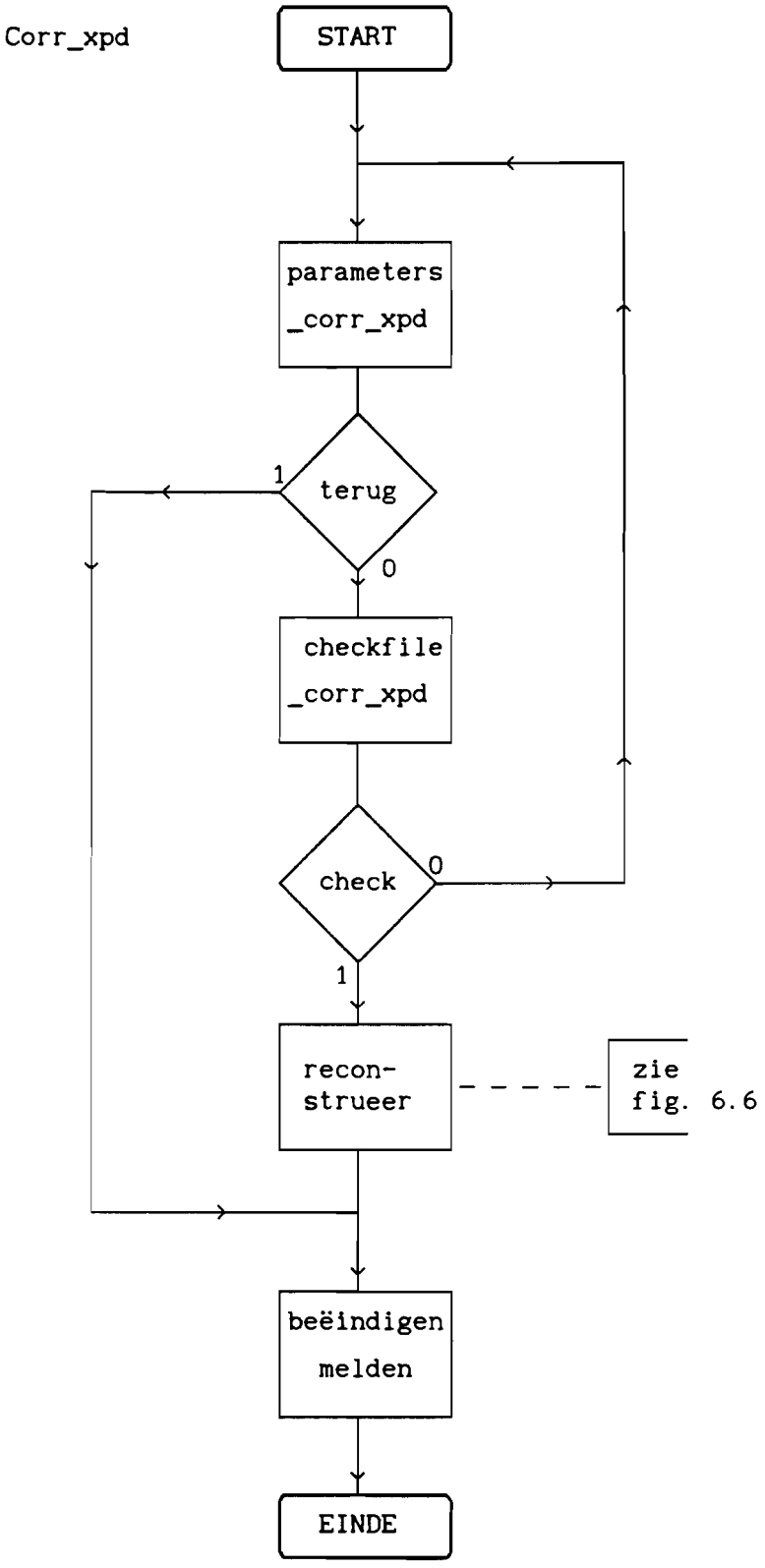


fig. 6.5 Flowschema van de functie Corr_xpd

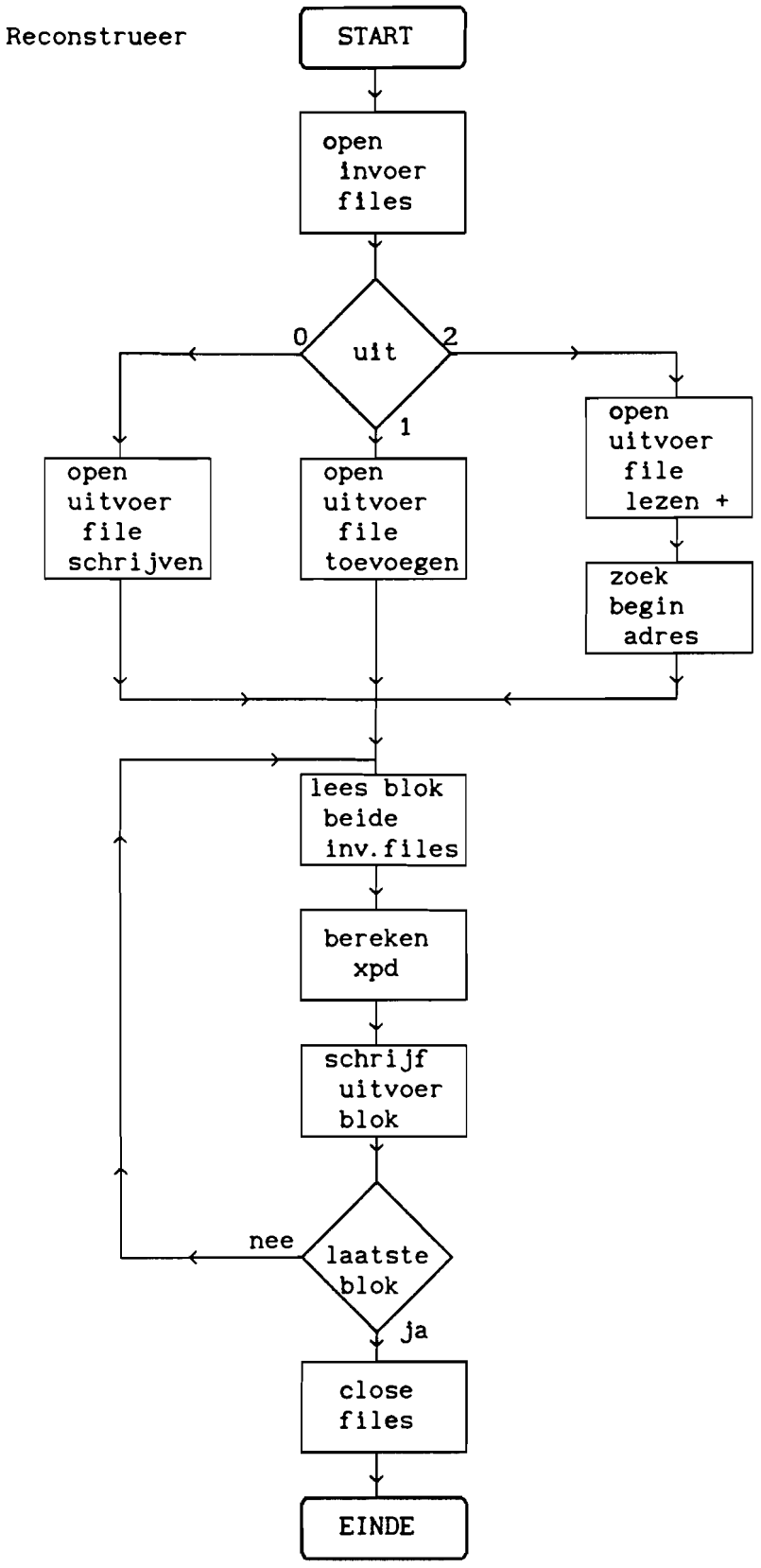


fig. 6.6 Flowschema van de functie reconstrueer

6.7 Grafiek

Om tijdreeksen grafisch te kunnen weergeven wordt gebruik gemaakt van de functie grafiek uit het hoofdpolarisatie gedeelte. Een beschrijving van deze functie is te vinden in **4.7** en de figuren 4.18 en 4.19.

6.8 IvsQ

De functie IvsQ verzorgt een grafische weergave van de quadratuur component, XPD-Q, als functie van de infase component, XPD-I. De functie zet eerst, met behulp van de functie parameters_IvsQ, de benodigde parameters op het scherm, ter inspectie en wijziging. Daarna wordt de functie checkfile_IvsQ aangeroepen. Deze hoeft net als bij grafiek alleen de invoerfiles te checken, waarbij eveneens wordt gekeken of de invoerfiles van de juiste soort zijn, een XPD-I en een XPD-Q file. De werking van deze beide functies is op een zelfde wijze als beschreven bij de functie CPA (zie **4.2** en de figuren 4.4 en 4.5). Tenslotte wordt de functie teken_IvsQ aangeroepen. Deze opent de invoerfiles, tekent het assenkruis, berekent de functiewaarden en tekent deze op het beeldscherm en sluit na afloop de geopende files weer. Het flowschema van de functie IvsQ wordt gegeven in fig. 6.7.

De functie teken_IvsQ verzorgt het daadwerkelijk op het scherm brengen van de gewenste grafiek en het bijbehorende assenkruis. Hiertoe tekent de functie eerst het assenkruis voorzien van de bijbehorende tekst, waaronder de maximale waarde langs de assen. Vervolgens worden de beide invoerfiles geopend. Nu wordt steeds voor ieder tijdblok de waarde uit beide files gelezen en het daarbij horende punt op het scherm bepaald en getekend. De functie is klaar als het laatste opgegeven tijdblok is verwerkt. De grafiek blijft op het scherm staan totdat een toets wordt ingedrukt. Het flowschema van de functie teken_IvsQ wordt gegeven in fig. 6.8.

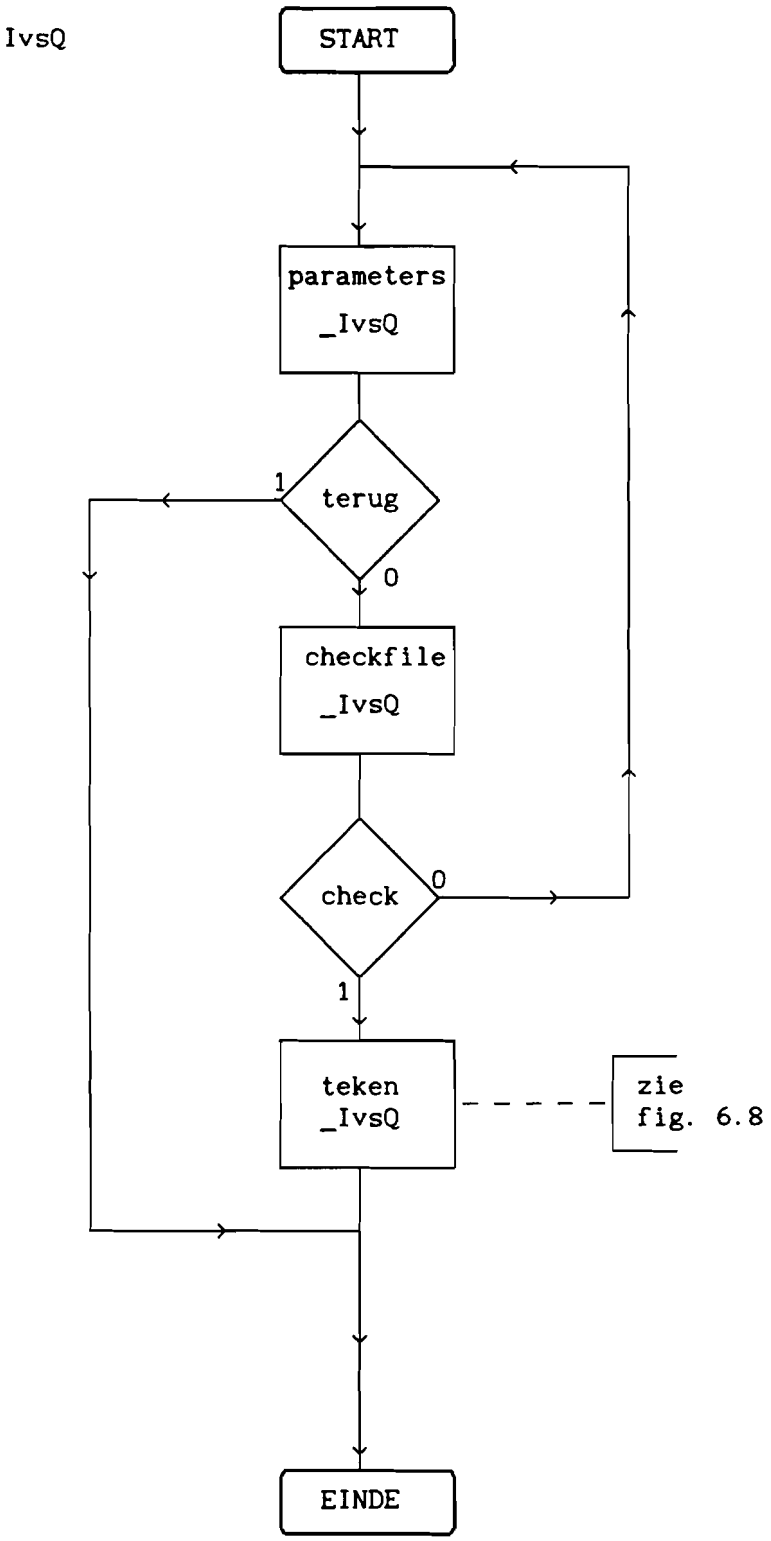


fig. 6.7 Flowschema van de functie IvsQ

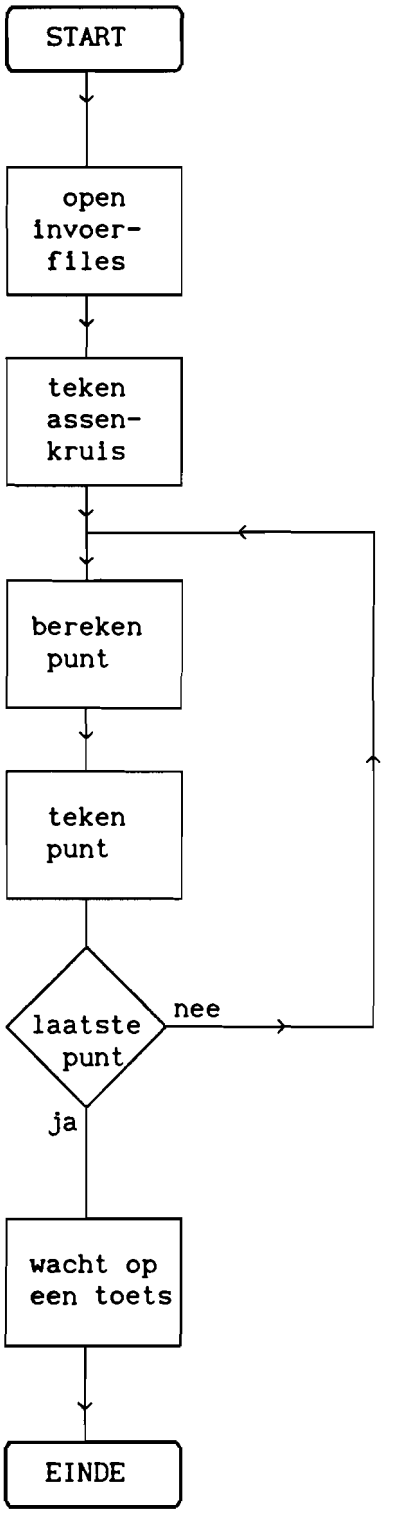


fig 6.8 Flowschema van de functie teken_IvsQ

7 CONCLUSIES EN AANBEVELINGEN

In dit verslag wordt een programma beschreven om ongewenste verstoringen in propagatiemetingen te compenseren. Er zijn een aantal tests gedaan met dit programma waarbij gebruik is gemaakt van data uit radiometermetingen van het PTT diversity experiment. Hieruit blijkt dat het programma naar verwachting functioneert, steeds worden de juiste functies in de juiste volgorde uitgevoerd. Van iedere module zijn de parameter functie en de functie die de in- en uitvoerfiles checkt getest. In alle gevallen bleken deze correct te werken. Er zullen nu nog een aantal tests, met hiervoor geschikte data, gedaan moeten worden, om te kijken of alle berekeningen correct worden uitgevoerd. De data uit het diversity experiment was hiervoor niet geschikt, maar liet wel zien dat de V en L berekening, de dempingomschaling, de template berekening en de verschil module in elk geval realistische waarden opleveren. Verder is hieruit duidelijk geworden dat de module die tijdreeksen grafisch moet weergeven correct functioneert. Om geschikte data te krijgen voor het uitvoeren van een acceptance test, kan zeer goed een file worden gecreëerd met bekende data, zodat de uitkomsten van te voren vaststaan. Deze data kan dan tevens gebruikt worden om anderen te leren het programma te gebruiken.

De gebruikte methode voor compensatie van de XPD metingen, komt neer op de left cancellation methode, zoals die beschreven wordt in appendix-L. De daar beschreven methode van right cancellation zou toegepast kunnen worden bij het, zowel horizontaal als verticaal gepolariseerde, 20 GHz baken. Hiervoor moet dan een aparte module worden toegevoegd. Het is dan wel noodzakelijk dat de verschilfase tussen de beide hoofdpolarisatiesignalen wordt gemeten.

LITTERATUURLIJST

[1] OPEX Olympus Propagation Experiment, "Handbook for Data Preprocessing, Part 1:Theory and basic information", ESA 1986.

[2] Dijk J., Maanders E.J., "Antennes en Propagatie", collegedictaat TU Eindhoven, september 1985.

[3] CCIR, Rec. 525-1 (Section A) of Study Group 5, XVI Plenary Assembly, Dubrovnik, 1986.

[4] CCIR, Rep. 719-2 (Section C) of Study Group 5, XVI Plenary Assembly, Dubrovnik, 1986.

[5] CCIR, Draft Rep. 719-2 (MOD I) of Study Group 5, Interim meeting, Geneva, 1988.

[6] Van Vleck J.H., Purcell E.M., Goldstein H., "Atmosferic Attenuation in Propagation of Short Radio Waves", edited by D.E. Kerr, chap.8, pp 641-692, McGraw-Hill, New York, 1951.

[7] Hawkins G.J., Edwards D.J., McGeehan J.P., "Tracking systems for satellite communications", IEE Proc., Vol. 135, No. 5, October 1988.

[8] Marinčić A.S., Popović B.D., "Radiation properties of microwave reflector antennas covered with a water film", IEE Proc., Vol. 125, No. 10, October 1978.

[9] Blevis B.C., "Losses due to rain on radomes and antenna reflecting surfaces", IEEE Trans. on Ant. & Prop., Vol. AP-13, January 1965.

[10] Dietrich F.J., West D.B., "An experimental radome panel evaluation", IEEE Trans. on Ant. & Prop., Vol. 36, No. 11, November 1988.

[11] Effenberger J.A., Strickland R.R., Joy E.B., "The effects of rain on a radome's performance", Microwave Journal, May 1986.

[12] Ruze J., "More on wet radomes", IEEE Trans. on Ant. & Prop., Vol. AP-13, September 1965.

[13] Anderson I., "Measurement of 20-GHz transmission through a radome in rain", IEEE Trans. on Ant. & Prop., Vol. AP-23, No. 5, September 1975.

[14] Weigand R.M., "Performance of a water-repellant radome coating in an airport surveillance radar", IEEE Proc.,Vol. 61, August 1973.

[15] Siller C.A., "Preliminary testing of teflon as a hydrophobic coating for Microwave radomes", IEEE Trans. on Ant. & Prop., Vol. AP-27, No. 4, July 1979.

[16] Brussaard G., "Radiometry: A useful prediction tool?", ESA SP-1071, 1985.

[17] CCIR, Draft Rep. 563-3 (MOD I) of Study Group 5, Interim meeting, Geneva, 1988.

[18] Shutie P.F., Allnutt J.E., Mackenzie E.G., "Satellite-earth signal depolarisation at 30 GHz in the absence of significant fading", Electronics Letters, Vol. 13, No. 1, January 1977.

[19] McEwan N.J., Watson P.A., Dissanayake A.W., Haworth D.P., Vakili V.T., "Crosspolarisation from high-altitude hydrometeors on a 20 GHz satellite radio path", Electronics Letters, Vol. 13, No. 1, January 1977.

[20] McCormick G.C., Hendry A., "Depolarisation by solid hydrometeors", Electronics Letters, Vol. 13, No. 3, February 1977.

[21] Howell R.G., "Crosspolar phase variations at 20 and 30 GHz on a satellite-earth path", Electronics Letters, Vol. 13, No. 14, July 1977.

[22] Thompson P, British Telecom. research memorandum nr. 76R6/35, Martlesham Heath, UK, 1976.

[23] Evans B.G., Thompson P.T., "Use of cancellation techniques in the measurement of atmospheric crosspolarisation", Electronics Letters, Vol. 9, No. 19, September 1973.

[24] Hendrix C.E., McNally J.E., Monzingo R.A., "Depolarization and attenuation effects of radomes at 20 GHz", IEEE Trans. on Ant. & Prop., Vol. 37, No. 3, March 1989.

[25] ESA software engineering standards, ESA BSSC (84)1, Issue no. 1, 1983.

[26] Chapin N, "Flowcharting with the ANSI standard: a tutorial", Computing Surveys, Vol. 2, No. 2, June 1970.

Appendix A

Hoofdprogramma

```
# include "c:\users\hans\eigen.h"


main()
{ int  ch=1,gdriver=DETECT,gmode,start,hoofdmenu(int k);
  void hoofdpolarisatie(int gmode),kruispolarisatie(int gmode);

  initgraph(&gdriver,&gmode,"\turboc");


  do
  { start=ch;
    ch=hoofdmenu(start);
    clrscr();
    switch(ch)
    { case 1 :
        hoofdpolarisatie(gmode);
        ch++;
        break;
      case 2 :
        kruispolarisatie(gmode);
        ch++;
        break;
      case 3 :
        ch=0;
        break;
    }
  } while (ch!=0);
  closegraph();
}
```

```
int hoofdmenu(int k)
{ int  kleur,b,j=0,key;
  void keerom(int i,char *p),keertrug(int i,char *p);
  char *str[4];
  static char patroon[8]={0x30,0x4A,0x0A,0x0A,0x12,0x22,0x7F,0x02};

  b=k;

  str[0]="HOOFDMENU";
  str[1]="HOOFDPOLARISATIEMENU";
  str[2]="KRUISPOLARISATIEMENU";
  str[3]="BEEINDIG HET PROGRAMMA";

  setvisualpage(1);
  setlinestyle(0,0,3);
  rectangle(7,6,713,337);
  setlinestyle(0,0,1);
  rectangle(260,20,460,50);
  rectangle(160,100,560,130);
  rectangle(160,180,560,210);
  rectangle(160,260,560,290);
  kleur=getcolor();
  setfillpattern(patroon,kleur);
  setfillstyle(12,kleur);
  floodfill(16,17,kleur);

  settextstyle(3,0,2);
  outtextxy(300,25,str[0]);
  outtextxy(240,105,str[1]);
  outtextxy(240,185,str[2]);
  outtextxy(240,265,str[3]);
  setvisualpage(0);

  do
  { keerom(b,str[b]);
    while(bioskey(1)==0);
    key=bioskey(0);
    keertrug(b,str[b]);
    switch(key)
    { case 0x5000 :
        b=b+1;
        if(b==4) b=1;
        break;
      case 0x4800 :
        b=b-1;
        if(b==0) b=3;
        break;
      case 0x1c0d :
        j=1;
        break;
    }
  } while (j!=1);
  return(b);
}
```

```
void keerom(int i,char *p)
{

  setviewport(161,80*i+21,559,80*i+49,0);
  clearviewport();
  setviewport(0,0,getmaxx(),getmaxy(),0);

  setfillstyle(1,7);
  floodfill(165,80*i+25,7);

  setcolor(0);
  outtextxy(240,80*i+25,p);
  setcolor(1);
}


void keertrug(int i,char *p)
{

  setviewport(161,80*i+21,559,80*i+49,0);
  clearviewport();
  setviewport(0,0,getmaxx(),getmaxy(),0);
  outtextxy(240,80*i+25,p);
}



Hoofdpolarisatie keuzemenu


#include "c:\users\hans\eigen.h"


void hoofdpolarisatie(int gmode)
{ int  ch=1,menu(int i,int k),start;
  void cpa(void),VenL(void),omschaling(void),template(void),
       verschil(void),grafiek(int mode),regen(void),tracking(void);

  restorecrtmode();

      /* menu levert een waarde af van 1 tot 9 aan ch      *
       * deze waarde bepaalt welke fuctie wordt uitgevoerd *
       * na uitvoeren van een functie wordt teruggekeerd   *
       * naar het hoofdmenu totdat ch=0 dan wordt het      *
       * programma beeindigd                               */
  do
  { start=ch;
    ch=menu(gmode,start);
    clrscr();
    switch(ch)
    { case 1 :
        cpa();
        ch++;
        break;
      case 2 :
        VenL();
        ch++;
        break;
```

```
      case 3 :
        omschaling();
        ch++;
        break;
      case 4 :
        verschil();
        ch++;
        break;
      case 5 :
        template();
        ch++;
        break;
      case 6 :
        grafiek(gmode);
        break;
      case 7 :
        tracking();
        break;
      case 8 :
        regen();
        break;
      case 9 :
        ch=0;
        break;
    }
  } while(ch!=0);
  setgraphmode(gmode);
}


int menu(int i,int k)
{ int  kleur,b,j=0,key;
  void restore(int i,char *p),reverse(int i,char *p);
  static char patroon[8]={0x30,0x4A,0x0A,0x0A,0x12,0x22,0x7F,0x02};
  char *str[11];

  b=k;

      /* menu tekent een keuzemenu waar met de up en down       *
       * cursorpijlen doorheen gelopen kan worden en            *
       * waaruit met de entertoets een keuze gemaakt kan worden *
       * menu levert een integer af die de gemaakte keuze       *
       * aangeeft                                               */

      /* de tekst wordt in een stringarray gezet         */
  str[0]="HOOFDPOLARISATIEMENU" ;
  str[1]="Omrekenen hoofdpolarisatie naar CPA";
  str[2]="V en L waarden";
  str[3]="Demping omschalen naar bakenfrequentie";
  str[4]="Verschil van twee tijdreeksen berekenen";
  str[5]="Template maken";
  str[6]="Grafisch weergeven";
  str[7]="Tracking";
  str[8]="Regen op antenne";
  str[9]="Beeindig het programma";
```

```
    /* de figuur wordt getekend                          */
  setgraphmode(i);
  setvisualpage(1);
  rectangle(220,20,500,50);
  rectangle(160,70,560,90);
  rectangle(160,100,560,120);
  rectangle(160,130,560,150);
  rectangle(160,160,560,180);
  rectangle(160,190,560,210);
  rectangle(160,220,560,240);
  rectangle(160,250,560,270);
  rectangle(160,280,560,300);
  rectangle(160,310,560,330);
  setlinestyle(0,0,3);
  rectangle(7,6,713,337);
  kleur=getcolor();
  setfillpattern(patroon,kleur);
  setfillstyle(12,kleur);
  floodfill(15,15,kleur);

    /* de tekst wordt in de figuur gezet                 */
  settextstyle(3,0,2);
  outtextxy(240,25,str[0]);
  settextstyle(0,0,1);
  outtextxy(200,76,str[1]);
  outtextxy(200,106,str[2]);
  outtextxy(200,136,str[3]);
  outtextxy(200,166,str[4]);
  outtextxy(200,196,str[5]);
  outtextxy(200,226,str[6]);
  outtextxy(200,256,str[7]);
  outtextxy(200,286,str[8]);
  outtextxy(200,316,str[9]);
  setvisualpage(0);

    /* loop door keuzemenu en selecteer                  */
  do
  { reverse(b,str[b]);      /* benadruk huidige positie          */
    while(bioskey(1)==0);
    key=bioskey(0);
    restore(b,str[b]);      /* herstel achtergrond               */
    switch(key)
    { case 0x5000 :         /* omlaag, keuze wordt een groter    */
      b=(b+1);
      if(b==10) b=1;        /* keuze te groot, terug naar bovenste */
      break;
      case 0x4800 :         /* omhoog, keuze wordt een kleiner   */
      b=(b-1);
      if(b==0) b=9;         /* keuze te klein, terug naar onderste */
      break;
      case 0x1c0d :         /* enter, keuze gemaakt              */
      j=1;                  /* do-loop wordt verlaten            */
    }
  }while(j!=1);
  restorecrtmode();
  return(b);                /* keuze-waarde wordt teruggegeven   */
}
```

```
void reverse(int i,char *p)
{
      /* reverse verandert de achtergrond van de         *
       * huidige keuze                                   */

  setviewport(161,30*i+41,559,30*i+59,0);
  clearviewport();
  setviewport(0,0,getmaxx(),getmaxy(),0);

  setfillstyle(1,7);
  floodfill(199,46+30*i,7);

  setcolor(0);
  outtextxy(200,30*i+46,p);
  setcolor(1);
}



void restore(int i,char *p)
{
      /* restore herstelt de oorspronkelijke achtergrond */

  setviewport(161,30*i+41,559,30*i+59,0);
  clearviewport();
  setviewport(0,0,getmaxx(),getmaxy(),0);
  outtextxy(200,30*i+46,p);
}
```

Kruispolarisatie keuzemenu

```
# include "c:\users\hans\eigen.h"


void kruispolarisatie(int gmode)
{ int  ch=1,xpdmenu(int i,int k),start;
  void xpd(void),template(void),temp_interpol(void),verschil(void),
       corr_xpd(void),grafiek(int mode),IvsQ_grafiek(int mode);

  restorecrtmode();

      /* xmenu levert een waarde af van 1 tot 8 aan ch     *
       * deze waarde bepaalt welke fuctie wordt uitgevoerd *
       * na uitvoeren van een functie wordt teruggekeerd   *
       * naar het keuzemenu totdat ch=0 dan wordt het      *
       * programma beeindigd                               */
do
  { start=ch;
    ch=xpdmenu(gmode,start);
    clrscr();
    switch(ch)
    { case 1 :
        xpd();
        ch++;
        break;
```

```
      case 2 :
        template();
        ch++;
        break;
      case 3 :
        temp_interpol();
        ch++;
        break;
      case 4 :
        verschil();
        ch++;
        break;
      case 5 :
        corr_xpd();
        ch++;
        break;
      case 6 :
        grafiek(gmode);
        ch++;
        break;
      case 7 :
        IvsQ_grafiek(gmode);
        break;
      case 8 :
        ch=0;
        break;
    }
  } while(ch!=0);
  setgraphmode(gmode);
}


int xpdmenu(int i,int k)
{ int  kleur,b,j=0,key;
  void restorexpd(int i,char *p),reversexpd(int i,char *p);
  static char patroon[8]={0x30,0x4A,0x0A,0x0A,0x12,0x22,0x7F,0x02};
  char *str[10];

  b=k;

      /* menu tekent een keuzemenu waar met de up en down       *
       * cursorpijlen doorheen gelopen kan worden en            *
       * waaruit met de entertoets een keuze gemaakt kan worden *
       * menu levert een integer af die de gemaakte keuze       *
       * aangeeft                                               */

      /* de tekst wordt in een stringarray gezet         */
  str[0]="KRUISPOLARISATIEMENU" ;
  str[1]="XPD XPD-I XPD-Q en Phase";
  str[2]="Template maken door middeling";
  str[3]="Template maken door interpolatie";
  str[4]="Verschil van twee tijdreeksen berekenen";
  str[5]="Gecorrigeerde XPD berekenen";
  str[6]="Grafisch weergeven van tijdreeksen";
  str[7]="Grafisch weergeven I vs Q";
  str[8]="Beeindig het programma";
```

```
    /* de figuur wordt getekend                          */
  setgraphmode(i);
  setvisualpage(1);
  rectangle(220,20,500,50);
  rectangle(160,70,560,90);
  rectangle(160,104,560,124);
  rectangle(160,138,560,158);
  rectangle(160,172,560,192);
  rectangle(160,206,560,226);
  rectangle(160,240,560,260);
  rectangle(160,274,560,294);
  rectangle(160,308,560,328);
  setlinestyle(0,0,3);
  rectangle(7,6,713,337);
  kleur=getcolor();
  setfillpattern(patroon,kleur);
  setfillstyle(12,kleur);
  floodfill(15,15,kleur);

    /* de tekst wordt in de figuur gezet                 */
  settextstyle(3,0,2);
  outtextxy(240,25,str[0]);
  settextstyle(0,0,1);
  outtextxy(200,76,str[1]);
  outtextxy(200,110,str[2]);
  outtextxy(200,144,str[3]);
  outtextxy(200,178,str[4]);
  outtextxy(200,212,str[5]);
  outtextxy(200,246,str[6]);
  outtextxy(200,280,str[7]);
  outtextxy(200,314,str[8]);
  setvisualpage(0);

    /* loop door keuzemenu en selecteer                  */
  do
  { reversexpd(b,str[b]);   /* benadruk huidige positie          */
    while(bioskey(1)==0);
    key=bioskey(0);
    restorexpd(b,str[b]);   /* herstel achtergrond               */
    switch(key)
    { case 0x5000 :         /* omlaag, keuze wordt een groter    */
      b=(b+1);
      if(b==9) b=1;         /* keuze te groot, terug naar bovenste */
      break;
      case 0x4800 :         /* omhoog, keuze wordt een kleiner   */
      b=(b-1);
      if(b==0) b=8;         /* keuze te klein, terug naar onderste */
      break;
      case 0x1c0d :         /* enter, keuze gemaakt              */
      j=1;                  /* do-loop wordt verlaten            */
    }
  }while(j!=1);
  restorecrtmode();
  return(b);                /* keuze-waarde wordt teruggegeven   */
}
```

```
void reversexpd(int i,char *p)
{
      /* reverse verandert de achtergrond van de          *
       * huidige keuze                                    */

  setviewport(161,34*i+37,559,34*i+55,0);
  clearviewport();
  setviewport(0,0,getmaxx(),getmaxy(),0);

  setfillstyle(1,7);
  floodfill(199,42+34*i,7);

  setcolor(0);
  outtextxy(200,34*i+42,p);
  setcolor(1);
}



void restorexpd(int i,char *p)
{
      /* restore herstelt de oorspronkelijke achtergrond */

  setviewport(161,34*i+37,559,34*i+55,0);
  clearviewport();
  setviewport(0,0,getmaxx(),getmaxy(),0);
  outtextxy(200,34*i+42,p);
}
```

Myfuncs: een aantal functies die door meerdere modules worden gebruikt

```
# include "c:\users\hans\eigen.h"


void schrijf(int x,int y,char str[50])
{ gotoxy(x,y);
  printf(str);
}


int is_in(char ch,char *s)
{ while (*s)
  { if (ch==*s)
      return(ch);
    else s++;
  }
  return(0);
}
```

```
void caps(char *p)
{ int  i=0;
  char a;

  while (p[i]!='\0')
  { a=toupper(p[i]);
    p[i]=a;
    i++;
  }
}


void tijd(int t,char *s)
{ int  abu,u,min;
  char hulp[5];

  abu=MAXTIJD/24;
  u=t/abu;
  min=((t%abu)+(abu/60)-1)/(abu/60);
  itoa(u,s,10);
  strcat(s,".");
  if (min<10)
    strcat(s,"0");
  itoa(min,hulp,10);
  strcat(s,hulp);
}


void tijd2(int t,char *s)
{ int  abu,u,min;
  char hulp[5];

  abu=MAXTIJD/24;
  u=t/abu;
  min=(t%abu)/(abu/60);
  itoa(u,s,10);
  strcat(s,".");
  if (min<10)
    strcat(s,"0");
  itoa(min,hulp,10);
  strcat(s,hulp);
}


int tijdblok(char *s)
{ double f;
  int    u,m;

  f=atof(s);
  u=(int)f;
  m=(int)((f-u)*100);
  return((MAXTIJD/24)*u+(MAXTIJD/(24*60))*m);
}
```

Eigen.h: zelfgedefinieerde headerfile

```
# include <stdio.h>
# include <bios.h>
# include <conio.h>
# include <graphics.h>
# include <ctype.h>
# include <math.h>
# include <string.h>
# include <stdlib.h>

# define Pi 3.141592
# define MAXTIJD 8640
# define PUNTEN 664
# define EVENTDREMPEL 3.0
```

Appendix B

CPA

```
# include "c:\users\hans\eigen.h"

       /* een structure wordt gedefinieerd waarin alle   *
        * parameters worden opgeborgen die binnen CPA    *
        * worden gebruikt                                *
        * de structure wordt globaal gedefinieerd zodat  *
        * alle parameters in een keer kunnen worden      *
        * ingelezen en voor iedere functie binnen CPA    *
        * bereikbaar zijn                                */

static struct parm { double     freq;    /* bakenfrequentie in GHz */
                     double     EIRP;    /* EIRP                   */
                     double       G0;    /* ontvangantennegain     */
                     double        d;    /* satellietafstand in km */
                     char  filin[15];    /* naam invoerfile        */
                     char filuit[15];    /* naam uitvoerfile       */
                     int        bdag;    /* begin dagnr.           */
                     int       btijd;    /* begin tijdblok         */
                     int        edag;    /* eind dagnr.            */
                     int       etijd;    /* eind tijdblok          */
                     char      s1[6];    /* begintijdstring        */
                     char      s2[6];    /* eindtijdstring         */
                   } par;
```

```
void cpa(void)
{ int    parameters_cpa(struct parm *p),
         checkfile_cpa(struct parm *p, int *uit);
  void   converteer(struct parm *p,int uit);
  int    terug,check,uitvlag;

      /* CPA converteert gemeten dempingswaarden          *
       * naar cpa waarden                                 */

  do
  {   /* de parameters worden uit de parameterfile gelezen    *
       * de gebruiker kan ze wijzigen totdat hij tevreden is  *
       * dan geeft de functie de waarde nul aan terug         *
       * zet de parameters in de structure en schrijft ze     *
       * in de parameterfile                                  *
       * of hij kan teruggaan naar het hoofdmenu dan geeft    *
       * de functie de waarde 1 aan terug                     */
    terug=parameters_cpa(&par);

      /* als terug=1 wordt de functie verlaten en men     *
       * komt vanzelf weer in het hoofdmenu              */
    if (terug==1) return;

      /* het bestaan van de opgegeven files wordt gecheckt    *
       * is alles in orde dan geeft de functie 1 terug en     *
       * kan het programma verder gaan                        *
       * is er iets niet in orde dan geeft de functie 0 terug *
       * en men keert terug naar parameters zodat men de      *
       * parameters kan wijzigen of vandaar terugkeren        *
       * naar het hoofdmenu                                   */
    check=checkfile_cpa(&par,&uitvlag);
  } while (check!=1);

      /* De ingezen meetwaarden worden omgezet naar       *
       * CPA waarden en deze worden in de uitvoerfile     *
       * gezet                                            */
  converteer(&par,uitvlag);

      /* gereedmelding van cpa wordt gegeven              */
  clrscr();
  schrijf(15,12,"Conversie naar CPA waarden voltooid!");
  schrijf(15,25,"Druk op een toets");
  getch();
}
```

```
int parameters_cpa(struct parm *p)
{ long reclen;
  FILE *fparm;
  int i,x,y,key,test,ch;

      /* parameterfile wordt geopend                          */
  reclen=sizeof(struct parm);
  fparm=fopen("CPA.par","r+b");

      /* als de file niet bestaat wordt hij gemaakt           *
       * en gevuld met nullen                                 */
  if (fparm==NULL)
  { fparm=fopen("CPA.par","w+b");
    for (i=0;i<reclen;i++)
    { putc('\0',fparm);
    }
  }

      /* lees file en zet de data in de structure             */
  fseek(fparm,0L,SEEK_SET);
  fread(p,reclen,1,fparm);

      /* de parameternamen worden op het scherm gezet         */
  clrscr();
  printf("\n\tParameterlijst\n\n"
         "\t               frequentie :\n"
         "\t                     EIRP :\n"
         "\t       ontvangantennegain :\n"
         "\t         satellietafstand :\n"
         "\t           invoerfilenaam :\n"
         "\t          uitvoerfilenaam :\n"
         "\t          begin dagnummer :\n"
         "\t               begin tijd :\n"
         "\t           eind dagnummer :\n"
         "\t                eind tijd :\n"
         "\t                                Tevreden\n"
         "\t                                Terug naar hoofdmenu");

      /* de parameterwaarden worden op het scherm gezet       */
  x=39;y=4;
  window(x,y,80,13);
  cprintf("%lf\r\n"
          "%lf\r\n"
          "%lf\r\n"
          "%lf\r\n"
          "%.14s\r\n"
          "%.14s\r\n"
          "%d\r\n"
          "%.5s\r\n"
          "%d\r\n"
          "%.5s"
          ,p->freq,p->EIRP,p->G0,p->d
          ,p->filin,p->filuit,p->bdag,p->s1,p->edag,p->s2);
```

```
    /* met de up en down toetsen kan de cursor op de          *
     * parameterwaarden worden gezet en kunnen deze worden    *
     * veranderd                                              *
     * enter op tevreden zet de configuratie in de par-file   *
     * en geeft 0 terug                                       *
     * enter op terug naar hoofdmenu geeft 1 terug zodat      *
     * teruggegaan wordt naar het hoofdmenu                   */
y=14;
do
{ window(1,1,80,25);
  gotoxy(x,y);
  while(bioskey(1)==0);
  key=bioskey(0);
  switch(key)
  { case 0x5000 :
      y=y+1;
      if (y>15) y=4;
      break;
    case 0x4800 :
      y=y-1;
      if (y<4) y=15;
      break;
    case 0x1c0d :
      if (y==14) test=1;
      if (y==15)
      { fclose(fparm);
        return(1);
      }
      break;
    default :
      if (isalnum(key&0xFF)&&y<14)
      { window(x,y,x+20,y);
        clrscr();
        cprintf("%c",key);
        ungetch(key);
        switch(y)
        { case 4 :
            cscanf("%lf",&p->freq);
            break;
          case 5 :
            cscanf("%lf",&p->EIRP);
            break;
          case 6 :
            cscanf("%lf",&p->G0);
            break;
          case 7 :
            cscanf("%lf",&p->d);
            break;
          case 8 :
            cscanf("%s",p->filin);
            break;
          case 9 :
            cscanf("%s",p->filuit);
            break;
          case 10 :
            cscanf("%d",&p->bdag);
            break;
```

```
            case 11 :
              cscanf("%s",p->s1);
              break;
            case 12 :
              cscanf("%d",&p->edag);
              break;
            case 13 :
              cscanf("%s",p->s2);
              break;
          } ch=getch();y++;
        } ch++;
    }
  } while (test!=1);

     /* de parameters worden wegezet in de parameterfile */
  p->btijd=tijdblok(p->s1);
  p->etijd=tijdblok(p->s2);
  fseek(fparm,0L,SEEK_SET);
  fwrite(p,reclen,1,fparm);
  fclose(fparm);
  return(0);
}


int checkfile_cpa(struct parm *p, int *uit)
{ struct indata { int dag;
                  int tijd;
                  int jaar;
                  int meetwaarde;
                } i1,i2;
  struct uitdata { int dagnr;
                   int tijdblok;
                   int jaartal;
                   double cpa;
                 } u1,u2;
  FILE *fin,*fuit;
  int  ch;
  char t1[6],t2[6];
  long j,k,lengte,aantal;

     /* de invoerfile wordt geopend                    */
  fin=fopen(p->filin,"rb");

     /* als de file niet bestaat wordt een melding gegeven *
      * op het scherm en wordt teruggekeerd naar de        *
      * parameterlijst via het teruggeven van 0 aan check  */
  if (fin==NULL)
  { clrscr();
    gotoxy(1,5);
    printf("\t\tDe invoerfile %s bestaat niet\n"
           "\t\tU keert terug naar de parameterlijst",p->filin);
    gotoxy(10,25);
    printf("Druk op een toets");
    getch();
    return(0);
  }
```

```
     /* begin- en eindpunt van de file worden opgezocht */
j=sizeof(struct indata);
fseek(fin,0L,SEEK_SET);
fread(&i1,j,1,fin);
fseek(fin,0L,SEEK_END);
lengte=ftell(fin);
aantal=lengte/j;
do
{ fseek(fin,(aantal-1)*j,SEEK_SET);
  fread(&i2,j,1,fin);
  aantal--;
} while (i2.dag==0);

     /* gekeken wordt of de opgegeven periode binnen   *
      * de file valt   zoniet dan wordt teruggekeerd   *
      * naar de parameterlijst                         */
if ((p->bdag<i1.dag||(p->bdag==i1.dag&&p->btijd<i1.tijd))||
    (p->edag>i2.dag||(p->edag==i2.dag&&p->etijd>i2.tijd)))
{ clrscr();
  tijd(i1.tijd,t1);
  tijd2(i2.tijd,t2);
  gotoxy(1,5);
  printf("\t\tInvoerfile %s\n"
       "\t\t  begin   dagnr : %d\n"
       "\t\t           tijd : %s\n"
       "\t\t  einde   dagnr : %d\n"
       "\t\t           tijd : %s"
       ,p->filin,i1.dag,t1,i2.dag,t2);

  printf("\n\n\t\tde opgegeven periode valt niet binnen %s\n"
         "\t\tU keert terug naar de parameterlijst!",p->filin);
  gotoxy(10,25);
  printf("Druk op een toets");
  getch();
  fclose(fin);
  return(0);
}

     /* valt de periode wel binnen de file             */
fclose(fin);

     /* de uitvoerfile wordt geopend                   *
      * bestaat de file al dan wordt gekeken of moet   *
      * worden overschreven of toegevoegd en de vlag   *
      * uit wordt gezet: uit=0 file bestaat niet,      *
      * uit=1 toevoegen, uit=2 overschrijven           */
fuit=fopen(p->filuit,"rb");

     /* de file bestaat nog niet                       *
      * openen kan geen kwaad er wordt teruggekeerd en *
      * check krijgt de waarde 1, zodat het programma  *
      * verder kan gaan                                */
if (fuit==NULL)
{ *uit=0;
  return(1);
}
```

```
      /* bestaat de file wel dan worden begin en eindpunt *
       * opgezocht en op het scherm gezet                 */
  k=sizeof(struct uitdata);
  fseek(fuit,0L,SEEK_SET);
  fread(&u1,k,1,fuit);
  fseek(fuit,0L,SEEK_END);
  lengte=ftell(fuit);
  aantal=lengte/k;
  fseek(fuit,(aantal-1)*k,SEEK_SET);
  fread(&u2,k,1,fuit);
  clrscr();
  tijd(u1.tijdblok,t1);
  tijd2(u2.tijdblok,t2);
  gotoxy(1,5);
  printf("\t\tUitvoerfile %s\n"
         "\t\t  begin   dagnr : %d\n"
         "\t\t           tijd : %s\n"
         "\t\t  einde   dagnr : %d\n"
         "\t\t           tijd : %s\n"
         ,p->filuit,u1.dagnr,t1,u2.dagnr,t2);
  fclose(fuit);

      /* ligt het beginpunt voor het einde van de file  *
       * dan wordt gevraagd of men wil overschrijven     *
       * uit=2 zoniet dan wordt gevraagd of men wil      *
       * toevoegen uit=1 wil men niet overschrijven of   *
       * toevoegen dan keert men terug naar de           *
       * parameterlijst                                  */
  if (p->bdag>u2.dagnr||(p->bdag==u2.dagnr&&p->btijd>u2.tijdblok))
  { printf("\n\n\t\tWilt U toevoegen aan het end van de file? (J/N)");
    ch=getch();
    if (ch=='J'||ch=='j')
    { *uit=1;
      return(1);
    }
    return(0);
  }
  printf("\n\n\t\tWilt U de file overschrijven? (J/N)");
  ch=getch();
  if (ch=='J'||ch=='j')
  { *uit=2;
    return(1);
  }
  return(0);
}
```

```
void converteer(struct parm *p,int uit)
{ struct indata { int dag;
                  int tijd;
                  int jaar;
                  int meetwaarde;
                } i;
  struct uitdata { int dagnr;
                   int tijdblok;
                   int jaartal;
                   double cpa;
                 } u;
  FILE   *fin,*fuit;
  long   j,k,x,y,s,t;
  double arg,c,L;

       /* de invoerfile wordt geopend en het eerste blok *
        * dat moet worden gelezen wordt opgezocht        */
  j=sizeof(struct indata);
  fin=fopen(p->filin,"rb");
  x=-100;
  do
  { x+=100;
    fseek(fin,x*j,SEEK_SET);
    fread(&i,j,1,fin);
  } while(i.dag<p->bdag);
  y=p->btijd-i.tijd;
  fseek(fin,(x+y)*j,SEEK_SET);

       /* de uitvoerfile wordt geopend en de             *
        * schrijfpointer wordt op de juiste plaats gezet *
        * afhankelijk van de waarde van uit              */
  k=sizeof(struct uitdata);
  switch(uit)
  { case 0 :
      fuit=fopen(p->filuit,"w+b");
      break;
    case 1 :
      fuit=fopen(p->filuit,"ab");
      break;
    case 2 :
      fuit=fopen(p->filuit,"r+b");
      fseek(fuit,0L,SEEK_SET);
      fread(&u,k,1,fuit);

      if(p->bdag<u.dagnr||(p->bdag==u.dagnr&&p->btijd<u.tijdblok))
      { fseek(fuit,0L,SEEK_SET);
      }
      else
      { s=0;
        while (p->bdag>u.dagnr)
        { s+=100;
          fseek(fuit,s*k,SEEK_SET);
          fread(&u,k,1,fuit);
        }
        t=p->btijd-u.tijdblok;
        fseek(fuit,(s+t)*k,SEEK_SET);
      }
  }
```

```
  clrscr();
  schrijf(35,12,"BEZIG");
  c=3*pow(10,8);
  arg=pow(4*Pi*p->d*p->freq/c,2);
  L=10*log10(arg);
  do
  { fread(&i,j,1,fin);
    u.cpa=p->EIRP+p->GO-L-10*log10(i.meetwaarde);

    u.dagnr=i.dag;
    u.tijdblok=i.tijd;
    u.jaartal=i.jaar;

    fwrite(&u,k,1,fuit);
  } while(u.dagnr<p->edag||u.tijdblok<p->etijd);
  fclose(fin);
  fclose(fuit);
}
```

Appendix C

VENL

```
# include "c:\users\hans\eigen.h"

       /* een structure wordt gedefinieerd waarin alle   *
        * parameters worden opgeborgen die binnen VenL   *
        * worden gebruikt                                *
        * de structure wordt globaal gedefinieerd zodat  *
        * alle parameters in een keer kunnen worden      *
        * ingelezen en voor iedere functie binnen VenL   *
        * bereikbaar zijn                                */
static struct parm { double     freq;    /* radiometer frequentie */
                     double        a;    /* conversie naar Ta     */
                     double        b;
                     double       Af;    /* feed loss factor      */
                     double        h;    /* fractie naar hemel    */
                     double    freq2;    /* idem voor tweede      */
                     double       a2;    /* radiometer            */
                     double       b2;    /*                       */
                     double      Af2;    /*                       */
                     double       h2;    /*                       */
                     double       Tm;    /* mediumtemperatuur     */
                     double       Te;    /* omgevingstemperatuur  */
                     double       rH;    /* relatieve vochtigheid */
                     double       th;    /* elevatiehoek          */
                     char  filin[15];    /* naam invoerfile       */
                     char filuit[15];    /* naam uitvoerfile      */
                     int        bdag;    /* begin dagnr.          */
                     int       btijd;    /* begin tijdblok        */
                     int        edag;    /* eind dagnr.           */
                     int       etijd;    /* eind tijdblok         */
                     char      s1[6];    /* begintijdstring       */
                     char      s2[6];    /* eindtijdstring        */
                   } par;
```

```
void VenL(void)
{ void   VenL1(double drmpl),VenL2(double drmpl),
         VenLmeer(double drmpl);
  int    aantal,keuze;
  double drempel;

      /* de defaulthoogte van de drempel waarboven de uit *
       * de radiometers berekende demping niet meer       *
       * betrouwbaar wordt geacht wordt gegeven en kan    *
       * door de gebruiker worden veranderd daarna vraagt *
       * VenL het aantal radiometers en voert afhankelijk *
       * van dit aantal de juiste functie uit             */

      /* verklarende tekst wordt op het scherm gezet */
  clrscr();
  schrijf(15,10,"Deze procedure berekent");
  schrijf(15,11,"V = de hoeveelheid water en");
  schrijf(15,12,"L = de hoeveelheid waterdamp");
  schrijf(15,13,"per vierkante meter op het satellietpad");
  schrijf(15,14,"m.b.v. radiometermetingen voor een op te");
  schrijf(15,15,"geven aantal radiometers!");
  schrijf(15,25,"Druk op een toets");
  getch();

      /* de drempelwaarde wordt vastgesteld               */
  clrscr();
  drempel=EVENTDREMPEL;
  schrijf(15,10,"De uit radiometers berekende demping boven");
  schrijf(15,11,"een drempelwaarde wordt als niet betrouwbaar");
  schrijf(15,12,"niet meegenomen in de berekeningen");
  schrijf(15,13,"De default waarde voor deze drempel ligt op 3 dB");
  schrijf(15,15,"Wilt U deze waarde wijzigen? (J/N)");
  keuze=toupper(getch());
  if (keuze=='J')
  { schrijf(15,17,"De nieuwe drempelwaarde wordt:     dB");
    gotoxy(47,17);
    scanf("%lf",&drempel);
  }

      /* het aantal radiometers wordt gevraagd en gelezen */
  clrscr();
  schrijf(15,12,"Geef het aantal radiometers : ");
  aantal=getch()-48;

      /* de grootte van aantal bepaalt welke functie     *
       * wordt uitgevoerd                                */
  clrscr();
  switch(aantal)
  { case 1 :
      VenL1(drempel);
      break;
    case 2 :
      VenL2(drempel);
      break;
    default :
      VenLmeer(drempel);
  }
}
```

```
void VenL1(double drmpl)
{ int    parameters_venl1(struct parm *p),
         checkfile_venl1(struct parm *p, int *uit);
  void   berekenL(struct parm *p, int uit,double drempel);
  int    terug,check,uitvlag;

      /* VenL1 berekent V en L voor 1 radiometer          */

  do
  {   /* de parameters worden uit de parameterfile gelezen    *
       * de gebruiker kan ze wijzigen totdat hij tevreden is  *
       * dan geeft de functie de waarde nul aan terug         *
       * zet de parameters in de structure en schrijft ze     *
       * in de parameterfile                                  *
       * of hij kan teruggaan naar het keuzemenu dan geeft    *
       * de functie de waarde 1 aan terug                     */
    terug=parameters_venl1(&par);

      /* als terug=1 wordt de functie verlaten en men    *
       * komt vanzelf weer in het keuzemenu             */
    if (terug==1) return;

      /* het bestaan van de opgegeven files wordt gecheckt    *
       * is alles in orde dan geeft de functie 1 terug en     *
       * kan het programma verder gaan                        *
       * is er iets niet in orde dan geeft de functie 0 terug *
       * en men keert terug naar parameters zodat men de      *
       * parameters kan wijzigen of vandaar terugkeren        *
       * naar het keuzemenu                                   */
    check=checkfile_venl1(&par,&uitvlag);
  } while (check!=1);


      /* uit de radiometermetingen wordt de demping berekend *
       * hiermee en met V en de radiometervariabelen         *
       * wordt L berekend                                    *
       * V en L worden in de uitvoerfile weggezet            */
  berekenL(&par,uitvlag,drmpl);

      /* Gereedmelding wordt gegeven                      */
  clrscr();
  schrijf(15,12,"Bepaling V en L beeindigd");
  schrijf(15,25,"Druk op een toets");
  getch();
}
```

```
int parameters_venl1(struct parm *p)
{ long reclen;
  FILE *fparm;
  int i,x,y,key,test,ch;

     /* parameterfile wordt geopend                          */
  reclen=sizeof(struct parm);
  fparm=fopen("VenL.par","r+b");

     /* als de file niet bestaat wordt hij gemaakt           *
      * en gevuld met nullen                                 */
  if (fparm==NULL)
  { fparm=fopen("VenL.par","w+b");
    for (i=0;i<reclen;i++)
    { putc('\0',fparm);
    }
  }

     /* lees file en zet de data in de structure             */
  fseek(fparm,0L,SEEK_SET);
  fread(p,reclen,1,fparm);

     /* de parameternamen worden op het scherm gezet         */
  clrscr();
  printf("\n\tParameterlijst\n\n"
         "\tradiometer     frequentie :\n"
         "\tconversie naar Ta       a :\n"
         "\t                        b :\n"
         "\tconversie naar Ts      Af :\n"
         "\t                        h :\n"
         "\tconversie naar demping Tm :\n"
         "\t                       Te :\n"
         "\trelatieve vochtigheid  rH :\n"
         "\televatiehoek          th :\n"
         "\t           invoerfilenaam :\n"
         "\t          uitvoerfilenaam :\n"
         "\t          begin dagnummer :\n"
         "\t               begin tijd :\n"
         "\t           eind dagnummer :\n"
         "\t                eind tijd :\n"
         "\t                                Tevreden\n"
         "\t                                Terug naar keuzemenu");

     /* de parameterwaarden worden op het scherm gezet       */
  x=39;y=4;
  window(x,y,80,19);
  cprintf("%lf\r\n"
          "%lf\r\n"
          "%lf\r\n"
          "%lf\r\n"
          "%lf\r\n"
          "%lf\r\n"
          "%lf\r\n"
          "%lf\r\n"
          "%lf\r\n"
          "%.14s\r\n"
          "%.14s\r\n"
          "%d\r\n"
```

```
        "%.5s\r\n"
        "%d\r\n"
        "%.5s"
        ,p->freq,p->a,p->b,p->Af,p->h,p->Tm,p->Te,p->rH,p->th
        ,p->filin,p->filuit,p->bdag,p->s1,p->edag,p->s2);

    /* met de up en down toetsen kan de cursor op de          *
     * parameterwaarden worden gezet en kunnen deze worden    *
     * veranderd                                              *
     * enter op tevreden zet de configuratie in de par-file   *
     * en geeft 0 terug                                       *
     * enter op terug naar keuzemenu geeft 1 terug zodat      *
     * teruggegaan wordt naar het keuzemenu                   */
y=19;
do
{ window(1,1,80,25);
  gotoxy(x,y);
  while(bioskey(1)==0);
  key=bioskey(0);
  switch(key)
  { case 0x5000 :
      y=y+1;
      if (y>20) y=4;
      break;
    case 0x4800 :
      y=y-1;
      if (y<4) y=20;
      break;
    case 0x1c0d :
      if (y==19) test=1;
      if (y==20)
      { fclose(fparm);
        return(1);
      }
      break;
    default :
      if (isalnum(key&0xFF)&&y<19)
      { window(x,y,x+20,y);
        clrscr();
        cprintf("%c",key);
        ungetch(key);
        switch(y)
        { case 4 :
            cscanf("%lf",&p->freq);
            break;
          case 5 :
            cscanf("%lf",&p->a);
            break;
          case 6 :
            cscanf("%lf",&p->b);
            break;
          case 7 :
            cscanf("%lf",&p->Af);
            break;
          case 8 :
            cscanf("%lf",&p->h);
            break;
```

```
            case 9 :
              cscanf("%lf",&p->Tm);
              break;
            case 10 :
              cscanf("%lf",&p->Te);
              break;
            case 11 :
              cscanf("%lf",&p->rH);
              break;
            case 12 :
              cscanf("%lf",&p->th);
              break;
            case 13 :
              cscanf("%s",p->filin);
              break;
            case 14 :
              cscanf("%s",p->filuit);
              break;
            case 15 :
              cscanf("%d",&p->bdag);
              break;
            case 16 :
              cscanf("%s",p->s1);
              break;
            case 17 :
              cscanf("%d",&p->edag);
              break;
            case 18 :
              cscanf("%s",p->s2);
              break;
          } ch=getch();y++;
        } ch++;
    }
  } while (test!=1);

     /* de parameters worden wegezet in de parameterfile */
  p->btijd=tijdblok(p->s1);
  p->etijd=tijdblok(p->s2);
  fseek(fparm,0L,SEEK_SET);
  fwrite(p,reclen,1,fparm);
  fclose(fparm);
  return(0);
}


int checkfile_venl1(struct parm *p, int *uit)
{ struct indata { int dag;
                  int tijd;
                  int jaar;
                  int rm1;
                  int rem1;
                  int rem2;
                  int rem3;
                  int rem4;
                  int rm2;
                } i1,i2;
```

```
struct uitdata { int dagnr;
                 int tijdblok;
                 int jaartal;
                 double v;
                 double l;
                 double te;
               } u1,u2;
FILE *fin,*fuit;
int  ch;
char t1[6],t2[6];
long j,k,lengte,aantal;

     /* de invoerfile wordt geopend                    */
fin=fopen(p->filin,"rb");

     /* als de file niet bestaat wordt een melding gegeven *
      * op het scherm en wordt teruggekeerd naar de        *
      * parameterlijst via het teruggeven van 0 aan check  */
if (fin==NULL)
{ clrscr();
  gotoxy(1,5);
  printf("\t\tDe invoerfile %s bestaat niet\n"
         "\t\tU keert terug naar de parameterlijst",p->filin);
  gotoxy(10,25);
  printf("Druk op een toets");
  getch();
  return(0);
}

     /* begin- en eindpunt van de file worden opgezocht */
j=sizeof(struct indata);
fseek(fin,0L,SEEK_SET);
fread(&i1,j,1,fin);
fseek(fin,0L,SEEK_END);
lengte=ftell(fin);
aantal=lengte/j;
do
{ fseek(fin,(aantal-1)*j,SEEK_SET);
  fread(&i2,j,1,fin);
  aantal--;
} while (i2.dag==0);

     /* gekeken wordt of de opgegeven periode binnen   *
      * de file valt   zoniet dan wordt teruggekeerd   *
      * naar de parameterlijst                         */
if ((p->bdag<i1.dag||(p->bdag==i1.dag&&p->btijd<i1.tijd))||
    (p->edag>i2.dag||(p->edag==i2.dag&&p->etijd>i2.tijd)))
{ clrscr();
  tijd(i1.tijd,t1);
  tijd2(i2.tijd,t2);
  gotoxy(1,5);
  printf("\t\tInvoerfile %s\n"
       "\t\t  begin   dagnr : %d\n"
       "\t\t           tijd : %s\n"
       "\t\t  einde   dagnr : %d\n"
       "\t\t           tijd : %s"
       ,p->filin,i1.dag,t1,i2.dag,t2);
```

```
  printf("\n\n\t\tde opgegeven periode valt niet binnen %s\n"
         "\t\tU keert terug naar de parameterlijst!",p->filin);
  schrijf(10,25,"Druk op een toets");
  getch();
  fclose(fin);
  return(0);
}

     /* valt de periode wel binnen de file            */
fclose(fin);

     /* de uitvoerfile wordt geopend                  *
      * bestaat de file al dan wordt gekeken of moet  *
      * worden overschreven of toegevoegd en de vlag  *
      * uit wordt gezet: uit=0 file bestaat niet,     *
      * uit=1 toevoegen, uit=2 overschrijven          */
fuit=fopen(p->filuit,"rb");

     /* de file bestaat nog niet                      *
      * openen kan geen kwaad er wordt teruggekeerd en *
      * check krijgt de waarde 1, zodat het programma  *
      * verder kan gaan                               */
if (fuit==NULL)
{ *uit=0;
  return(1);
}

     /* bestaat de file wel dan worden begin en eindpunt *
      * opgezocht en op het scherm gezet                 */
k=sizeof(struct uitdata);
fseek(fuit,0L,SEEK_SET);
fread(&u1,k,1,fuit);
fseek(fuit,0L,SEEK_END);
lengte=ftell(fuit);
aantal=lengte/k;
fseek(fuit,(aantal-1)*k,SEEK_SET);
fread(&u2,k,1,fuit);
clrscr();
tijd(u1.tijdblok,t1);
tijd2(u2.tijdblok,t2);
gotoxy(1,5);
printf("\t\tUitvoerfile %s\n"
       "\t\t  begin   dagnr : %d\n"
       "\t\t           tijd : %s\n"
       "\t\t  einde   dagnr : %d\n"
       "\t\t           tijd : %s\n"
       ,p->filuit,u1.dagnr,t1,u2.dagnr,t2);
fclose(fuit);
```

```
        /* ligt het beginpunt voor het einde van de file  *
         * dan wordt gevraagd of men wil overschrijven    *
         * uit=2 zoniet dan wordt gevraagd of men wil     *
         * toevoegen uit=1 wil men niet overschrijven of  *
         * toevoegen dan keert men terug naar de          *
         * parameterlijst                                 */
  if (p->bdag>u2.dagnr||(p->bdag==u2.dagnr&&p->btijd>u2.tijdblok))
  { printf("\n\n\t\tWilt U toevoegen aan het end van de file? (J/N)");
    ch=getch();
    if (ch=='J'||ch=='j')
    { *uit=1;
      return(1);
    }
    return(0);
  }
  printf("\n\n\t\tWilt U de file overschrijven? (J/N)");
  ch=getch();
  if (ch=='J'||ch=='j')
  { *uit=2;
    return(1);
  }
  return(0);
}


void radiometervariabelen(double *a, struct parm *p)
{ double x,y,z,r,s,t,c1,c2,c3;

  c1 = 1 + 0.006 * (288 - p->Te);
  c2 = 1 + 0.001 * (p->Te - 288);
  c3 = 1 + 0.01 * (p->Te - 288);

  x = pow(p->freq,2);
  y = pow(p->freq-22.2,2);
  z = pow(p->freq-57,2);

  r = sin(p->th);
  s = pow(10,-4);
  t = pow(10,-3);

  a[0] = 0.0021*x*s*(1+3.0/(y+5))*c1*c2/1.6;
  a[1] = (0.050+3.6/(y+8.6))*x*s*(1+3.0/(y+5))*c1*c2;
  a[2] = 9.5*s*x;
  a[3] = (7.19*t+6.09/(x+0.227)+4.81/(z+150))*x*t*c3*6.0/r;
}

double berekenV(struct parm *p)
{ double a,b,T6,hulpv;

  a = 20 - 2950/p->Te;
  T6 = pow(p->Te,-6);
  b = pow(10,a);

  hulpv = 20296.988*b*T6*p->rH;

  return(hulpv);
}
```

```
void berekenL(struct parm *p, int uit, double drempel)
{ struct indata { int dag;
                  int tijd;
                  int jaar;
                  int rm1;
                  int rem1;
                  int rem2;
                  int rem3;
                  int rem4;
                  int rm2;
                } i;
  struct uitdata { int dagnr;
                   int tijdblok;
                   int jaartal;
                   double v;
                   double l;
                   double te;
                 } u;
  FILE   *fin,*fuit;
  long   j,k,x,y,s,t;
  double At,L,conversie(struct parm *p, int i),arr[4],V;
  void   radiometervariabelen(double *a, struct parm *p);
  double berekenV(struct parm *p);

       /* de invoerfile wordt geopend en het eerste blok *
        * dat moet worden gelezen wordt opgezocht        */
  j=sizeof(struct indata);
  fin=fopen(p->filin,"rb");
  x=-100;
  do
  { x+=100;
    fseek(fin,x*j,SEEK_SET);
    fread(&i,j,1,fin);
  } while(i.dag<p->bdag);
  y=p->btijd-i.tijd;
  fseek(fin,(x+y)*j,SEEK_SET);

       /* de uitvoerfile wordt geopend en de             *
        * schrijfpointer wordt op de juiste plaats gezet *
        * afhankelijk van de waarde van uit              */
  k=sizeof(struct uitdata);
  switch(uit)
  { case 0 :
      fuit=fopen(p->filuit,"w+b");
      break;
    case 1 :
      fuit=fopen(p->filuit,"ab");
      break;
    case 2 :
      fuit=fopen(p->filuit,"r+b");
      fseek(fuit,0L,SEEK_SET);
      fread(&u,k,1,fuit);

      if(p->bdag<u.dagnr||(p->bdag==u.dagnr&&p->btijd<u.tijdblok))
      { fseek(fuit,0L,SEEK_SET);
      }
```

```
      else
      { s=0;
        while (p->bdag>u.dagnr)
        { s+=100;
          fseek(fuit,s*k,SEEK_SET);
          fread(&u,k,1,fuit);
        }
        t=p->btijd-u.tijdblok;
        fseek(fuit,(s+t)*k,SEEK_SET);
      }
  }

  clrscr();
  schrijf(35,12,"BEZIG");
  do
  { fread(&i,j,1,fin);

      /* de demping wordt berekend                          */
    At=conversie(&par,i.rm1);

      /* komt de berekende demping boven de opgegeven       *
       * drempel dan krijgen V en L een negatieve           *
       * waarde toegekend                                   */
    if (At>drempel)
    { V=-1;
      L=-1;
    }

    else
    {
      /* de frequentieafhankelijke variabelen A,B,C,D       *
       * uit de formule A*V*V + B*V + C*L + D = demping     *
       * worden bepaald voor de radiometerfrequentie        *
       * en opgeborgen in het array arr[4]                  */
      radiometervariabelen(arr,&par);

      /* V wordt berekend uit de omgevingstemperatuur       *
       * en de relatieve vochtigheid                        */
      V=berekenV(&par);

      L=-(arr[0]*V*V + arr[1]*V + arr[3] - At)/arr[2];

      if (V<0||L<0)
        V=L=0;
    }

    u.dagnr=i.dag;
    u.tijdblok=i.tijd;
    u.jaartal=i.jaar;
    u.v=V;
    u.l=L;
    u.te=p->Te;

    fwrite(&u,k,1,fuit);
  } while(u.dagnr<p->edag||u.tijdblok<p->etijd);
  fclose(fin);
  fclose(fuit);
}
```

```
double conversie(struct parm *p,int i)
{ double Ta,Ts,Tc=3.0;

  Ta=(p->a*(double)i*10.0/256.0)-p->b;
  Ts=Ta*p->Af/p->h+(1-p->Af/p->h)*p->Te;

  if (p->Tm>Ts)
  return(10*log10((p->Tm-Tc)/(p->Tm-Ts)));
  else
  return(0);
}
```

```
void VenL2(double drmpl)
{ int    parameters_venl2(struct parm *p),
         checkfile_venl2(struct parm *p, int *uit);
  void   berekenVenL(struct parm *p, int uit, double drempel);
  int    terug,check,uitvlag;

      /* VenL2 berekent V en L voor 2 radiometers        */

  do
  {   /* de parameters worden uit de parameterfile       *
       * gelezen de gebruiker kan ze wijzigen totdat hij *
       * tevreden is dan geeft de functie de waarde nul  *
       * aan terug zet de parameters in de structure en  *
       * schrijft ze in de parameterfile                 *
       * of hij kan teruggaan naar het keuzemenu dan     *
       * geeft de functie de waarde 1 aan terug          */
    terug=parameters_venl2(&par);

      /* als terug=1 wordt de functie verlaten en men    *
       * komt vanzelf weer in het keuzemenu              */
    if (terug==1) return;

      /* het bestaan van de opgegeven files wordt        *
       * gecheckt is alles in orde dan geeft de functie  *
       * 1 terug en kan het programma verder gaan        *
       * is er iets niet in orde dan geeft de functie 0  *
       * terug en men keert terug naar parameters zodat  *
       * men de parameters kan wijzigen of vandaar       *
       * terugkeren naar het keuzemenu                   */
    check=checkfile_venl2(&par,&uitvlag);
  } while (check!=1);



      /* uit de radiometermetingen wordt de demping      *
       * berekend hiermee en met de radiometervariabelen *
       * worden V en L berekend                          *
       * V en L worden in de uitvoerfile weggezet        */
  berekenVenL(&par,uitvlag,drmpl);

      /* Gereedmelding wordt gegeven                     */
  clrscr();
  schrijf(15,12,"Bepaling V en L beeindigd");
  schrijf(15,25,"Druk op een toets");
  getch();
}
```

```
int parameters_ven12(struct parm *p)
{ long reclen;
  FILE *fparm;
  int  i,x,y,key,test,ch;

      /* parameterfile wordt geopend                          */
  reclen=sizeof(struct parm);
  fparm=fopen("VenL.par","r+b");

      /* als de file niet bestaat wordt hij gemaakt           *
       * en gevuld met nullen                                 */
  if (fparm==NULL)
  { fparm=fopen("VenL.par","w+b");
    for (i=0;i<reclen;i++)
    { putc('\0',fparm);
    }
  }

      /* lees file en zet de data in de structure             */
  fseek(fparm,0L,SEEK_SET);
  fread(p,reclen,1,fparm);

      /* de parameternamen worden op het scherm gezet         */
  clrscr();
  printf("\n\tParameterlijst\n\n"
         "\tradiometer 1   frequentie :\n"
         "\tconversie naar Ta       a :\n"
         "\t                        b :\n"
         "\tconversie naar Ts      Af :\n"
         "\t                        h :\n"
         "\tradiometer 2   frequentie :\n"
         "\tconversie naar Ta       a :\n"
         "\t                        b :\n"
         "\tconversie naar Ts      Af :\n"
         "\t                        h :\n"
         "\tconversie naar demping Tm :\n"
         "\t                       Te :\n"
         "\televatiehoek           th :\n"
         "\t           invoerfilenaam :\n"
         "\t          uitvoerfilenaam :\n"
         "\t          begin dagnummer :\n"
         "\t               begin tijd :\n"
         "\t           eind dagnummer :\n"
         "\t                eind tijd :\n"
         "\t                                Tevreden\n"
         "\t                                Terug naar keuzemenu");

      /* de parameterwaarden worden op het scherm gezet       */
  x=39;y=4;
  window(x,y,80,22);
  cprintf("%lf\r\n"
          "%lf\r\n"
          "%lf\r\n"
          "%lf\r\n"
          "%lf\r\n"
          "%lf\r\n"
          "%lf\r\n"
          "%lf\r\n"
```

```
        "%lf\r\n"
        "%lf\r\n"
        "%lf\r\n"
        "%lf\r\n"
        "%lf\r\n"
        "%.14s\r\n"
        "%.14s\r\n"
        "%d\r\n"
        "%.5s\r\n"
        "%d\r\n"
        "%.5s"
        ,p->freq,p->a,p->b,p->Af,p->h,p->freq2,p->a2,p->b2
        ,p->Af2,p->h2,p->Tm,p->Te,p->th
        ,p->filin,p->filuit,p->bdag,p->s1,p->edag,p->s2);

    /* met de up en down toetsen kan de cursor op de   *
     * parameterwaarden worden gezet en kunnen deze    *
     * worden veranderd                                *
     * enter op tevreden zet de configuratie in de     *
     * par-file en geeft 0 terug                       *
     * enter op terug naar keuzemenu geeft 1 terug     *
     * zodat teruggegaan wordt naar het keuzemenu      */
y=23;
do
{ window(1,1,80,25);
  gotoxy(x,y);
  while(bioskey(1)==0);
  key=bioskey(0);
  switch(key)
  { case 0x5000 :
      y=y+1;
      if (y>24) y=4;
      break;
    case 0x4800 :
      y=y-1;
      if (y<4) y=24;
      break;
    case 0x1c0d :
      if (y==23) test=1;
      if (y==24)
      { fclose(fparm);
        return(1);
      }
      break;
    default :
      if (isalnum(key&0xFF)&&y<23)
      { window(x,y,x+20,y);
        clrscr();
        cprintf("%c",key);
        ungetch(key);
        switch(y)
        { case 4 :
            cscanf("%lf",&p->freq);
            break;
          case 5 :
            cscanf("%lf",&p->a);
            break;
```

```
            case 6 :
              cscanf("%lf",&p->b);
              break;
            case 7 :
              cscanf("%lf",&p->Af);
              break;
            case 8 :
              cscanf("%lf",&p->h);
              break;
            case 9 :
              cscanf("%lf",&p->freq2);
              break;
            case 10 :
              cscanf("%lf",&p->a2);
              break;
            case 11 :
              cscanf("%lf",&p->b2);
              break;
            case 12 :
              cscanf("%lf",&p->Af2);
              break;
            case 13 :
              cscanf("%lf",&p->h2);
              break;
            case 14 :
              cscanf("%lf",&p->Tm);
              break;
            case 15 :
              cscanf("%lf",&p->Te);
              break;
            case 16 :
              cscanf("%lf",&p->th);
              break;
            case 17 :
              cscanf("%s",p->filin);
              break;
            case 18 :
              cscanf("%s",p->filuit);
              break;
            case 19 :
              cscanf("%d",&p->bdag);
              break;
            case 20 :
              cscanf("%s",p->s1);
              break;
            case 21 :
              cscanf("%d",&p->edag);
              break;
            case 22 :
              cscanf("%s",p->s2);
              break;
          } ch=getch();y++;
        } ch++;
    }
} while (test!=1);
```

```
      /* de parameters worden weggezet in de parameterfile */
  p->btijd=tijdblok(p->s1);
  p->etijd=tijdblok(p->s2);
  fseek(fparm,0L,SEEK_SET);
  fwrite(p,reclen,1,fparm);
  fclose(fparm);
  return(0);
}


int checkfile_ven12(struct parm *p, int *uit)
{ struct indata { int dag;
                  int tijd;
                  int jaar;
                  int rm1;
                  int rem1;
                  int rem2;
                  int rem3;
                  int rem4;
                  int rm2;
                } i1,i2;
  struct uitdata { int dagnr;
                   int tijdblok;
                   int jaartal;
                   double v;
                   double l;
                   double te;
                 } u1,u2;
  FILE *fin,*fuit;
  int  ch;
  char t1[6],t2[6];
  long j,k,lengte,aantal;

      /* de invoerfile wordt geopend                    */
  fin=fopen(p->filin,"rb");

      /* als de file niet bestaat wordt een melding gegeven *
       * op het scherm en wordt teruggekeerd naar de        *
       * parameterlijst via het teruggeven van 0 aan check  */
  if (fin==NULL)
  { clrscr();
    gotoxy(1,5);
    printf("\t\tDe invoerfile %s bestaat niet\n"
           "\t\tU keert terug naar de parameterlijst",p->filin);
    gotoxy(10,25);
    printf("Druk op een toets");
    getch();
    return(0);
  }

      /* begin- en eindpunt van de file worden opgezocht */
  j=sizeof(struct indata);
  fseek(fin,0L,SEEK_SET);
  fread(&i1,j,1,fin);
  fseek(fin,0L,SEEK_END);
  lengte=ftell(fin);
  aantal=lengte/j;
```

```
do
{ fseek(fin,(aantal-1)*j,SEEK_SET);
  fread(&i2,j,1,fin);
  aantal--;
} while (i2.dag==0);

     /* gekeken wordt of de opgegeven periode binnen   *
      * de file valt   zoniet dan wordt teruggekeerd   *
      * naar de parameterlijst                         */
if ((p->bdag<i1.dag||(p->bdag==i1.dag&&p->btijd<i1.tijd))||
    (p->edag>i2.dag||(p->edag==i2.dag&&p->etijd>i2.tijd)))
{ clrscr();
  tijd(i1.tijd,t1);
  tijd2(i2.tijd,t2);
  gotoxy(1,5);
  printf("\t\tInvoerfile %s\n"
       "\t\t  begin   dagnr : %d\n"
       "\t\t           tijd : %s\n"
       "\t\t  einde   dagnr : %d\n"
       "\t\t           tijd : %s"
       ,p->filin,i1.dag,t1,i2.dag,t2);

     printf("\n\n\t\tde opgegeven periode valt niet binnen %s\n"
         "\t\tU keert terug naar de parameterlijst!",p->filin);
  schrijf(10,25,"Druk op een toets");
  getch();
  fclose(fin);
  return(0);
}

     /* valt de periode wel binnen de file             */
fclose(fin);

     /* de uitvoerfile wordt geopend                   *
      * bestaat de file al dan wordt gekeken of moet   *
      * worden overschreven of toegevoegd en de vlag   *
      * uit wordt gezet: uit=0 file bestaat niet,      *
      * uit=1 toevoegen, uit=2 overschrijven           */
fuit=fopen(p->filuit,"rb");

     /* de file bestaat nog niet                       *
      * openen kan geen kwaad er wordt teruggekeerd en *
      * check krijgt de waarde 1, zodat het programma  *
      * verder kan gaan                                */
if (fuit==NULL)
{ *uit=0;
  return(1);
}

     /* bestaat de file wel dan worden begin en eindpunt *
      * opgezocht en op het scherm gezet                 */
k=sizeof(struct uitdata);
fseek(fuit,0L,SEEK_SET);
fread(&u1,k,1,fuit);
fseek(fuit,0L,SEEK_END);
lengte=ftell(fuit);
aantal=lengte/k;
fseek(fuit,(aantal-1)*k,SEEK_SET);
```

```
  fread(&u2,k,1,fuit);
  tijd(u1.tijdblok,t1);
  tijd2(u2.tijdblok,t2);
  clrscr();
  gotoxy(1,5);
  printf("\t\tUitvoerfile %s\n"
         "\t\t  begin   dagnr : %d\n"
         "\t\t           tijd : %s\n"
         "\t\t  einde   dagnr : %d\n"
         "\t\t           tijd : %s\n"
         ,p->filuit,u1.dagnr,t1,u2.dagnr,t2);
  fclose(fuit);

       /* ligt het beginpunt voor het einde van de file  *
        * dan wordt gevraagd of men wil overschrijven    *
        * uit=2 zoniet dan wordt gevraagd of men wil     *
        * toevoegen uit=1 wil men niet overschrijven of  *
        * toevoegen dan keert men terug naar de          *
        * parameterlijst                                 */
  if (p->bdag>u2.dagnr||(p->bdag==u2.dagnr&&p->btijd>u2.tijdblok))
  { printf("\n\n\t\tWilt U toevoegen aan het end van de file? (J/N)");
    ch=getch();
    if (ch=='J'||ch=='j')
    { *uit=1;
      return(1);
    }
    return(0);
  }
  printf("\n\n\t\tWilt U de file overschrijven? (J/N)");
  ch=getch();
  if (ch=='J'||ch=='j')
  { *uit=2;
    return(1);
  }
  return(0);
}



void radiometervariabelen2(double *a, struct parm *p)
{ double x,x2,y,y2,z,z2,r,s,t,c1,c2,c3;

  c1 = 1 + 0.006 * (288 - p->Te);
  c2 = 1 + 0.001 * (p->Te - 288);
  c3 = 1 + 0.01 * (p->Te - 288);

  x = pow(p->freq,2);
  x2= pow(p->freq2,2);
  y = pow(p->freq-22.2,2);
  y2= pow(p->freq2-22.2,2);
  z = pow(p->freq-57,2);
  z2= pow(p->freq2-57,2);

  r = sin(p->th);
  s = pow(10,-4);
  t = pow(10,-3);
```

```
  a[0] = 0.0021*x*s*(1+3.0/(y+5))*c1*c2/1.6;
  a[1] = (0.050+3.6/(y+8.6))*x*s*(1+3.0/(y+5))*c1*c2;
  a[2] = 9.5*s*x;
  a[3] = (7.19*t+6.09/(x+0.227)+4.81/(z+150))*x*t*c3*6.0/r;
  a[4] = 0.0021*x2*s*(1+3.0/(y2+5))*c1*c2/1.6;
  a[5] = (0.050+3.6/(y2+8.6))*x2*s*(1+3.0/(y2+5))*c1*c2;
  a[6] = 9.5*s*x2;
  a[7] = (7.19*t+6.09/(x2+0.227)+4.81/(z2+150))*x2*t*c3*6.0/r;
}


void berekenVenL(struct parm *p,int uit,double drempel)
{ struct indata { int dag;
                  int tijd;
                  int jaar;
                  int rm1;
                  int rem1;
                  int rem2;
                  int rem3;
                  int rem4;
                  int rm2;
                } i;
  struct uitdata { int dagnr;
                   int tijdblok;
                   int jaartal;
                   double v;
                   double l;
                   double te;
                 } u;
  FILE   *fin,*fuit;
  long   j,k,x,y,s,t;
  double At1,At2,V,L,ratio,aratio,bratio,dratio,Atratio,kwad,disc,
         conversie(struct parm *p, int i),arr[8],
         conversie2(struct parm *p, int i);
  void   radiometervariabelen2(double *a, struct parm *p);

       /* de invoerfile wordt geopend en het eerste blok *
        * dat moet worden gelezen wordt opgezocht        */
  j=sizeof(struct indata);
  fin=fopen(p->filin,"rb");
  x=-100;
  do
  { x+=100;
    fseek(fin,x*j,SEEK_SET);
    fread(&i,j,1,fin);
  } while(i.dag<p->bdag);
  y=p->btijd-i.tijd;
  fseek(fin,(x+y)*j,SEEK_SET);

       /* de uitvoerfile wordt geopend en de             *
        * schrijfpointer wordt op de juiste plaats gezet *
        * afhankelijk van de waarde van uit              */
  k=sizeof(struct uitdata);
  switch(uit)
  { case 0 :
      fuit=fopen(p->filuit,"w+b");
      break;
```

```
  case 1 :
    fuit=fopen(p->filuit,"ab");
    break;
  case 2 :
    fuit=fopen(p->filuit,"r+b");
    fseek(fuit,0L,SEEK_SET);
    fread(&u,k,1,fuit);

    if(p->bdag<u.dagnr||(p->bdag==u.dagnr&&p->btijd<u.tijdblok))
    { fseek(fuit,0L,SEEK_SET);
    }
    else
    { s=0;
      while (p->bdag>u.dagnr)
      { s+=100;
        fseek(fuit,s*k,SEEK_SET);
        fread(&u,k,1,fuit);
      }
      t=p->btijd-u.tijdblok;
      fseek(fuit,(s+t)*k,SEEK_SET);
    }
}

clrscr();
schrijf(35,12,"BEZIG");
do
{ fread(&i,j,1,fin);

    /* de frequentieafhankelijke variabelen A,B,C,D    *
     * uit de formule A*V*V + B*V + C*L + D = demping  *
     * worden bepaald voor de radiometerfrequenties    *
     * en opgeborgen in het array arr[8]               */
  radiometervariabelen2(arr,&par);

    /* V en L worden uitgerekend door het oplossen van *
     * de twee kwadratische vergelijkingen             *
     * At1=arr[0]*V*V+arr[1]*V+arr[2]*L+arr[3] en      *
     * At2=arr[4]*V*V+arr[5]*V+arr[6]*L+arr[7]         *
     * eerst worden de coefficienten berekend          */
  ratio=arr[6]/arr[2];
  aratio=arr[4]-ratio*arr[0];
  bratio=arr[5]-ratio*arr[1];
  dratio=arr[7]-ratio*arr[3];
  kwad=pow(bratio,2);

    /* demping wordt berekend uit de meetwaarden       */
  At1=conversie(&par,i.rm1);
  At2=conversie2(&par,i.rm2);

    /* ligt een van de twee berekende dempingwaarden   *
     * boven de opgegeven drempelwaaarde dan krijgen   *
     * V en L een negatieve waarde toegekend           */
  if (At1>drempel||At2>drempel)
  { V=-1;
    L=-1;
  }
```

```
    else
    {
      /* de vergelijkingen worden opgelost met behulp    *
       * van de wortelformule                            *
       * levert dit negatieve waarden op dan wordt de    *
       * kwadratische term verwaarloosd en de lineaire   *
       * vergelijking opgelost                           *
       * is dan nog V of L negatief dan worden beide     *
       * gelijk aan nul gemaakt                          */
      Atratio=At2-ratio*At1;
      disc=kwad-4*aratio*(dratio-Atratio);
      if (disc<0)
        V=-1;
      else
      { V=(-bratio-sqrt(disc))/(2*aratio);
        if (V<0)
          V=(-bratio+sqrt(disc))/(2*aratio);
      }
      if (V<0)
      { V=(Atratio-dratio)/bratio;
        L=(At1-arr[1]*V-arr[3])/arr[2];
      }
      else
        L=-(arr[0]*V*V + arr[1]*V + arr[3] - At1)/arr[2];
      if (V<0||L<0)
        V=L=0;
    }

       /* de uitkomsten worden in de uitvoerstructure    *
        * gezet en in de uitvoerfile geschreven          */
    u.dagnr=i.dag;
    u.tijdblok=i.tijd;
    u.jaartal=i.jaar;
    u.v=V;
    u.l=L;
    u.te=p->Te;

    fwrite(&u,k,1,fuit);
  } while(u.dagnr<p->edag||u.tijdblok<p->etijd);
  fclose(fin);
  fclose(fuit);
}


double conversie2(struct parm *p,int i)
{ double Ta,Ts,Tc=3.0;

  Ta=(p->a2*(double)i*10.0/256.0)-p->b2;
  Ts=Ta*p->Af2/p->h2+(1-p->Af2/p->h2)*p->Te;

  if (p->Tm>Ts)
  return(10*log10((p->Tm-Tc)/(p->Tm-Ts)));
  else
  return(0);
}
```

```
void VenLmeer(double drmpl)
{ int   parameters_venl2(struct parm *p),
        checkfile_venl2(struct parm *p, int *uit);
  void  berekenVenL(struct parm *p, int uit, double drempel);
  int   terug,check,uitvlag;

  schrijf(15,10,"U dient twee radiometers te kiezen,");
  schrijf(15,12,"waarvoor U de parameters moet opgeven.");
  schrijf(15,14,"De functie zal dan daarmee de waarden");
  schrijf(15,16,"van V en L berekenen!");
  schrijf(15,25,"Druk op een toets!");
  getch();

  do
  { terug=parameters_venl2(&par);
    if (terug==1) return;
    check=checkfile_venl2(&par,&uitvlag);
  } while (check!=1);
  berekenVenL(&par,uitvlag,drmpl);

      /* Gereedmelding wordt gegeven                    */
  clrscr();
  schrijf(15,12,"Bepaling V en L beeindigd");
  schrijf(15,25,"Druk op een toets");
  getch();
}
```

Appendix D

Omschaling

```
# include "c:\users\hans\eigen.h"

       /* een structure wordt gedefinieerd waarin alle   *
        * parameters worden opgeborgen die binnen        *
        * omschaling worden gebruikt                     *
        * de structure wordt globaal gedefinieerd zodat  *
        * alle parameters in een keer kunnen worden      *
        * ingelezen en voor iedere functie binnen        *
        * omschaling bereikbaar zijn                     */

static struct parm { double     freq;   /* bakenfrequentie in GHz */
                     double       th;   /* elevatiehoek           */
                     char  filin[15];   /* naam invoerfile        */
                     char filuit[15];   /* naam uitvoerfile       */
                     int        bdag;   /* begin dagnr.           */
                     int       btijd;   /* begin tijdblok         */
                     int        edag;   /* eind dagnr.            */
                     int       etijd;   /* eind tijdblok          */
                     char      s1[6];   /* begintijdstring        */
                     char      s2[6];   /* eindtijdstring         */
                   } par;
```

```
void omschaling(void)
{ int    parameters_omschaling(struct parm *p),
         checkfile_omschaling(struct parm *p, int *uit);
  void   demping(struct parm *p, int uit);
  int    terug,check,uitvlag;


      /* omschaling bepaalt de demping bij een          *
       * bakenfrequentie uit de met radiometers bepaalde *
       * waarden van V en L                             */

  do
  {   /* de parameters worden uit de parameterfile gelezen    *
       * de gebruiker kan ze wijzigen totdat hij tevreden is  *
       * dan geeft de functie de waarde nul aan terug         *
       * zet de parameters in de structure en schrijft ze     *
       * in de parameterfile                                  *
       * of hij kan teruggaan naar het keuzemenu dan geeft    *
       * de functie de waarde 1 aan terug                     */
    terug=parameters_omschaling(&par);

      /* als terug=1 wordt de functie verlaten en men    *
       * komt vanzelf weer in het keuzemenu             */
    if (terug==1) return;

      /* het bestaan van de opgegeven files wordt gecheckt    *
       * is alles in orde dan geeft de functie 1 terug en     *
       * kan het programma verder gaan                        *
       * is er iets niet in orde dan geeft de functie 0 terug *
       * en men keert terug naar parameters zodat men de      *
       * parameters kan wijzigen of vandaar terugkeren        *
       * naar het keuzemenu                                   */
    check=checkfile_omschaling(&par,&uitvlag);
  } while (check!=1);

      /* met behulp van de waarden voor V en L uit de    *
       * invoerfile en de door dempingvariabelen gegeven *
       * waarden wordt de demping bij de bakenfrequentie *
       * uitgerekend en in de uitvoerfile gezet          */
  demping(&par,uitvlag);

      /* gereedmelding wordt gegeven                     */
  clrscr();
  schrijf(15,12,"Dempingomschaling beeindigd");
  schrijf(15,25,"Druk op een toets");
  getch();
}
```

```
int parameters_omschaling(struct parm *p)
{ long reclen;
  FILE *fparm;
  int i,x,y,key,test=0,ch;

     /* parameterfile wordt geopend                          */
  reclen=sizeof(struct parm);
  fparm=fopen("SCHALING.par","r+b");

     /* als de file niet bestaat wordt hij gemaakt           *
      * en gevuld met nullen                                 */
  if (fparm==NULL)
  { fparm=fopen("SCHALING.par","w+b");
    for (i=0;i<reclen;i++)
    { putc('\0',fparm);
    }
  }

     /* lees file en zet de data in de structure             */
  fseek(fparm,0L,SEEK_SET);
  fread(p,reclen,1,fparm);

     /* de parameternamen worden op het scherm gezet         */
  clrscr();
  printf("\n\tParameterlijst\n\n"
       · "\t               frequentie :\n"
         "\t             elevatiehoek :\n"
         "\t           invoerfilenaam :\n"
         "\t          uitvoerfilenaam :\n"
         "\t          begin dagnummer :\n"
         "\t               begin tijd :\n"
         "\t           eind dagnummer :\n"
         "\t                eind tijd :\n"
         "\t                                Tevreden\n"
         "\t                                Terug naar keuzemenu");

     /* de parameterwaarden worden op het scherm gezet       */
  x=39;y=4;
  window(x,y,80,12);
  cprintf("%lf\r\n"
          "%lf\r\n"
          "%.14s\r\n"
          "%.14s\r\n"
          "%d\r\n"
          "%.5s\r\n"
          "%d\r\n"
          "%.5s"
          ,p->freq,p->th
          ,p->filin,p->filuit,p->bdag,p->s1,p->edag,p->s2);
```

```
    /* met de up en down toetsen kan de cursor op de          *
     * parameterwaarden worden gezet en kunnen deze worden    *
     * veranderd                                              *
     * enter op tevreden zet de configuratie in de par-file   *
     * en geeft 0 terug                                       *
     * enter op terug naar keuzemenu geeft 1 terug zodat      *
     * teruggegaan wordt naar het keuzemenu                   */
y=12;
do
{ window(1,1,80,25);
  gotoxy(x,y);
  while(bioskey(1)==0);
  key=bioskey(0);
  switch(key)
  { case 0x5000 :
      y=y+1;
      if (y>13) y=4;
      break;
    case 0x4800 :
      y=y-1;
      if (y<4) y=13;
      break;
    case 0x1c0d :
      if (y==12) test=1;
      if (y==13)
      { fclose(fparm);
        return(1);
      }
      break;
    default :
      if (isalnum(key&0xFF)&&y<12)
      { window(x,y,x+20,y);
        clrscr();
        cprintf("%c",key);
        ungetch(key);
        switch(y)
        { case 4 :
            cscanf("%lf",&p->freq);
            break;
          case 5 :
            cscanf("%lf",&p->th);
            break;
          case 6 :
            cscanf("%s",p->filin);
            break;
          case 7 :
            cscanf("%s",p->filuit);
            break;
          case 8 :
            cscanf("%d",&p->bdag);
            break;
          case 9 :
            cscanf("%s",p->s1);
            break;
          case 10 :
            cscanf("%d",&p->edag);
            break;
```

```
            case 11 :
              cscanf("%s",p->s2);
              break;
           } ch=getch();y++;
         } ch++;
    }
  } while (test!=1);

      /* de parameters worden wegezet in de parameterfile */
  p->btijd=tijdblok(p->s1);
  p->etijd=tijdblok(p->s2);
  fseek(fparm,0L,SEEK_SET);
  fwrite(p,reclen,1,fparm);
  fclose(fparm);
  return(0);
}


int checkfile_omschaling(struct parm *p, int *uit)
{ struct indata { int dag;
                  int tijd;
                  int jaar;
                  double v;
                  double l;
                  double te;
                } i1,i2;
  struct uitdata { int dagnr;
                   int tijdblok;
                   int jaartal;
                   double demping;
                 } u1,u2;
  FILE *fin,*fuit;
  int  ch;
  char t1[6],t2[6];
  long j,k,lengte,aantal;

      /* de invoerfile wordt geopend                    */
  fin=fopen(p->filin,"rb");

      /* als de file niet bestaat wordt een melding gegeven *
       * op het scherm en wordt teruggekeerd naar de        *
       * parameterlijst via het teruggeven van 0 aan check  */
  if (fin==NULL)
  { clrscr();
    gotoxy(1,5);
    printf("\t\tDe invoerfile %s bestaat niet\n"
           "\t\tU keert terug naar de parameterlijst",p->filin);
    gotoxy(10,25);
    printf("Druk op een toets");
    getch();
    return(0);
  }
```

```
     /* begin- en eindpunt van de file worden opgezocht */
j=sizeof(struct indata);
fseek(fin,0L,SEEK_SET);
fread(&i1,j,1,fin);
fseek(fin,0L,SEEK_END);
lengte=ftell(fin);
aantal=lengte/j;
do
{ fseek(fin,(aantal-1)*j,SEEK_SET);
  fread(&i2,j,1,fin);
  aantal--;
} while (i2.dag==0);
clrscr();

     /* gekeken wordt of de opgegeven periode binnen   *
      * de file valt   zoniet dan wordt teruggekeerd   *
      * naar de parameterlijst                         */
if ((p->bdag<i1.dag||(p->bdag==i1.dag&&p->btijd<i1.tijd))||
    (p->edag>i2.dag||(p->edag==i2.dag&&p->etijd>i2.tijd)))
{ tijd(i1.tijd,t1);
  tijd2(i2.tijd,t2);
  gotoxy(1,5);
  printf("\t\tInvoerfile %s\n"
       "\t\t  begin   dagnr : %d\n"
       "\t\t           tijd : %s\n"
       "\t\t  einde   dagnr : %d\n"
       "\t\t           tijd : %s"
       ,p->filin,i1.dag,t1,i2.dag,t2);

  printf("\n\n\t\tde opgegeven periode valt niet binnen %s\n"
         "\t\tU keert terug naar de parameterlijst!",p->filin);
  schrijf(10,25,"Druk op een toets");
  getch();
  fclose(fin);
  return(0);
}

     /* valt de periode wel binnen de file             */
fclose(fin);

     /* de uitvoerfile wordt geopend                   *
      * bestaat de file al dan wordt gekeken of moet   *
      * worden overschreven of toegevoegd en de vlag   *
      * uit wordt gezet: uit=0 file bestaat niet,      *
      * uit=1 toevoegen, uit=2 overschrijven           */
fuit=fopen(p->filuit,"rb");

     /* de file bestaat nog niet                       *
      * openen kan geen kwaad er wordt teruggekeerd en *
      * check krijgt de waarde 1, zodat het programma  *
      * verder kan gaan                                */
if (fuit==NULL)
{ *uit=0;
  return(1);
}
```

```
      /* bestaat de file wel dan worden begin en eindpunt *
       * opgezocht en op het scherm gezet                 */
  k=sizeof(struct uitdata);
  fseek(fuit,0L,SEEK_SET);
  fread(&u1,k,1,fuit);
  fseek(fuit,0L,SEEK_END);
  lengte=ftell(fuit);
  aantal=lengte/k;
  fseek(fuit,(aantal-1)*k,SEEK_SET);
  fread(&u2,k,1,fuit);
  clrscr();
  tijd(u1.tijdblok,t1);
  tijd2(u2.tijdblok,t2);
  gotoxy(1,5);
  printf("\t\tUitvoerfile %s\n"
         "\t\t  begin   dagnr : %d\n"
         "\t\t           tijd : %s\n"
         "\t\t  einde   dagnr : %d\n"
         "\t\t           tijd : %s\n"
         ,p->filuit,u1.dagnr,t1,u2.dagnr,t2);
  fclose(fuit);

      /* ligt het beginpunt voor het einde van de file  *
       * dan wordt gevraagd of men wil overschrijven     *
       * uit=2 zoniet dan wordt gevraagd of men wil      *
       * toevoegen uit=1 wil men niet overschrijven of   *
       * toevoegen dan keert men terug naar de           *
       * parameterlijst                                  */
  if (p->bdag>u2.dagnr||(p->bdag==u2.dagnr&&p->btijd>u2.tijdblok))
  { printf("\n\n\t\tWilt U toevoegen aan het end van de file? (J/N)");
    ch=getch();
    if (ch=='J'||ch=='j')
    { *uit=1;
      return(1);
    }
    return(0);
  }
  printf("\n\n\t\tWilt U de file overschrijven? (J/N)");
  ch=getch();
  if (ch=='J'||ch=='j')
  { *uit=2;
    return(1);
  }
  return(0);
}



void dempingvariabelen(struct parm *p, double *a, double temp)
{ double x,y,z,r,s,t,c1,c2,c3;

  c1=1+0.006*(288-temp);
  c2=1+0.001*(temp-288);
  c3=1+0.01*(temp-288);

  x=pow(p->freq,2);
  y=pow(p->freq-22.2,2);
  z=pow(p->freq-57,2);
```

```
  r=sin(p->th);
  s=pow(10,-4);
  t=pow(10,-3);

  a[0]=0.0021*x*s*(1+3.0/(y+5))*c1*c2/1.6;
  a[1]=(0.050+3.6/(y+8.6))*x*s*(1+3.0/(y+5))*c1*c2;
  a[2]=9.5*s*x;
  a[3]=(7.19*t+6.09/(x+0.227)+4.81/(z+150))*x*t*c3*6.0/r;
}


void demping(struct parm *p, int uit)
{ struct indata { int dag;
                  int tijd;
                  int jaar;
                  double v;
                  double l;
                  double te;
                } i;
  struct uitdata { int dagnr;
                   int tijdblok;
                   int jaartal;
                   double demping;
                 } u;
  FILE   *fin,*fuit;
  long   j,k,x,y,s,t;
  double arr[4];
  void   dempingvariabelen(struct parm *p, double *a, double temp);

       /* de invoerfile wordt geopend en het eerste blok *
        * dat moet worden gelezen wordt opgezocht        */
  j=sizeof(struct indata);
  fin=fopen(p->filin,"rb");
  x=-100;
  do
  { x+=100;
    fseek(fin,x*j,SEEK_SET);
    fread(&i,j,1,fin);
  } while(i.dag<p->bdag);
  y=p->btijd-i.tijd;
  fseek(fin,(x+y)*j,SEEK_SET);

       /* de uitvoerfile wordt geopend en de             *
        * schrijfpointer wordt op de juiste plaats gezet *
        * afhankelijk van de waarde van uit              */
  k=sizeof(struct uitdata);
  switch(uit)
  { case 0 :
      fuit=fopen(p->filuit,"w+b");
      break;
    case 1 :
      fuit=fopen(p->filuit,"ab");
      break;
    case 2 :
      fuit=fopen(p->filuit,"r+b");
      fseek(fuit,0L,SEEK_SET);
      fread(&u,k,1,fuit);
```

```
      if(p->bdag<u.dagnr||(p->bdag==u.dagnr&&p->btijd<u.tijdblok))
      { fseek(fuit,0L,SEEK_SET);
      }
      else
      { s=0;
        while (p->bdag>u.dagnr)
        { s+=100;
          fseek(fuit,s*k,SEEK_SET);
          fread(&u,k,1,fuit);
        }
        t=p->btijd-u.tijdblok;
        fseek(fuit,(s+t)*k,SEEK_SET);
      }
  }
  clrscr();
  schrijf(35,12,"BEZIG");
  do
  { fread(&i,j,1,fin);

      /* lag de radiometerdemping boven de drempel dan   *
       * wordt de demping nu negatief gemaakt            */
    if (i.v<=-1.0||i.l<=-1.0)
      u.demping=-1000;

    else
    {
      /* de frequentieafhankelijke variabelen van de     *
       * dempingsformule 1.20 worden uitgerekend         */
      dempingvariabelen(&par,arr,i.te);

      u.demping=i.v*i.v*arr[0]+i.v*arr[1]+i.l*arr[2]+arr[3];
    }

    u.dagnr=i.dag;
    u.tijdblok=i.tijd;
    u.jaartal=i.jaar;

    fwrite(&u,k,1,fuit);
  } while(u.dagnr<p->edag||u.tijdblok<p->etijd);
  fclose(fin);
  fclose(fuit);
}
```

Appendix E

Template door middeling

```
# include "c:\users\hans\eigen.h"

       /* een structure wordt gedefinieerd waarin alle   *
        * parameters worden opgeborgen die binnen        *
        * template worden gebruikt                       *
        * de structure wordt globaal static gedefinieerd *
        * zodat alle parameters in een keer kunnen       *
        * worden ingelezen en voor iedere functie binnen *
        * template bereikbaar zijn                       */
static struct parm { char inv1[15];      /* naam invoerfile1 */
                     char inv2[15];      /* naam invoerfile2 */
                     char inv3[15];      /* naam invoerfile3 */
                     char inv4[15];      /* naam invoerfile4 */
                     char inv5[15];      /* naam invoerfile5 */
                     char uitv[15];      /* naam uitvoerfile */
                     int      bdag;      /* begin dagnr.     */
                     int     btijd;      /* begin tijdblok   */
                     int      edag;      /* eind dagnr.      */
                     int     etijd;      /* eind tijdblok    */
                     char    s1[6];      /* begintijdstring  */
                     char    s2[6];      /* eindtijdstring   */
                   } par;
```

```
void template(void)
{ int    parameters_template(struct parm *p,int n),
         checkfile_template(struct parm *p, int *uit,int n);
  void   gemiddelde(struct parm *p,int uit,int n);
  int    terug,check,uitvlag,aantal;

      /* Template geeft een schatting voor de systeem-   *
       * demping door middeling van de gegevens van een  *
       * aantal andere dagen                             */

  do
  { clrscr();
    schrijf(15,10,"Geef het aantal invoerfiles dat U wilt gebruiken");
    schrijf(15,12,"om het gemiddelde te berekenen! (2-5) :");
    aantal=getch()-48;
  } while (aantal<2!!aantal>5);


  do
  {   /* de parameters worden uit de parameterfile gelezen    *
       * de gebruiker kan ze wijzigen totdat hij tevreden is  *
       * dan geeft de functie de waarde nul aan terug         *
       * zet de parameters in de structure en schrijft ze     *
       * in de parameterfile                                  *
       * of hij kan teruggaan naar het keuzemenu dan geeft    *
       * de functie de waarde 1 aan terug                     */
    terug=parameters_template(&par,aantal);

      /* als terug=1 wordt de functie verlaten en men     *
       * komt vanzelf weer in het keuzemenu              */
    if (terug==1) return;

      /* het bestaan van de opgegeven files wordt gecheckt    *
       * is alles in orde dan geeft de functie 1 terug en     *
       * kan het programma verder gaan                        *
       * is er iets niet in orde dan geeft de functie 0 terug *
       * en men keert terug naar parameters zodat men de      *
       * parameters kan wijzigen of vandaar terugkeren        *
       * naar het keuzemenu                                   */
    check=checkfile_template(&par,&uitvlag,aantal);
  } while (check!=1);

      /* het gemiddelde van de waarden in de invoerfiles *
       * wordt berekend en in de uitvoerfile gezet       */
  gemiddelde(&par,uitvlag,aantal);

      /* gereedmelding wordt gegeven                     */
  clrscr();
  schrijf(15,12,"Template vervaardigen beeindigd!");
  schrijf(15,25,"Druk op een toets");
  getch();
  }
```

```
int parameters_template(struct parm *p,int n)
{ long reclen;
  FILE *fparm;
  int  i,x,y,key,test,ch;

      /* parameterfile wordt geopend                    */
  reclen=sizeof(struct parm);
  fparm=fopen("Template.par","r+b");

      /* als de file niet bestaat wordt hij gemaakt     *
       * en gevuld met nullen                           */
  if (fparm==NULL)
  { fparm=fopen("Template.par","w+b");
    for (i=0;i<reclen;i++)
    { putc('\0',fparm);
    }
  }

      /* lees file en zet de data in de structure       */
  fseek(fparm,0L,SEEK_SET);
  fread(p,reclen,1,fparm);

      /* de parameternamen worden op het scherm gezet   */
  clrscr();
  printf("\n\tParameterlijst\n\n"
         "\t          invoerfilenaam1 :\n");
  if (n>1)
  printf("\t          invoerfilenaam2 :\n");
  if (n>2)
  printf("\t          invoerfilenaam3 :\n");
  if (n>3)
  printf("\t          invoerfilenaam4 :\n");
  if (n>4)
  printf("\t          invoerfilenaam5 :\n");
  printf("\t          uitvoerfilenaam :\n"
         "\t          begin dagnummer :\n"
         "\t               begin tijd :\n"
         "\t           eind dagnummer :\n"
         "\t                eind tijd :\n"
         "\t                                Tevreden\n"
         "\t                                Terug naar keuzemenu");

      /* de parameterwaarden worden op het scherm gezet */
  x=39;y=4;
  window(x,y,80,8+n);
  cprintf("%.14s\r\n",p->inv1);
  if (n>1)  cprintf("%.14s\r\n",p->inv2);
  if (n>2)  cprintf("%.14s\r\n",p->inv3);
  if (n>3)  cprintf("%.14s\r\n",p->inv4);
  if (n>4)  cprintf("%.14s\r\n",p->inv5);
  cprintf("%.14s\r\n"
          "%d\r\n"
          "%.5s\r\n"
          "%d\r\n"
          "%.5s"
          ,p->uitv,p->bdag,p->s1,p->edag,p->s2);
```

```
    /* met de up en down toetsen kan de cursor op de          *
     * parameterwaarden worden gezet en kunnen deze worden    *
     * veranderd                                              *
     * enter op tevreden zet de configuratie in de par-file   *
     * en geeft 0 terug                                       *
     * enter op terug naar keuzemenu geeft 1 terug zodat      *
     * teruggegaan wordt naar het keuzemenu                   */
y=9+n;
do
{ window(1,1,80,25);
  gotoxy(x,y);
  while(bioskey(1)==0);
  key=bioskey(0);
  switch(key)
  { case 0x5000 :
      y=y+1;
      if (y>10+n) y=4;
      break;
    case 0x4800 :
      y=y-1;
      if (y<4) y=10+n;
      break;
    case 0x1c0d :
      if (y==9+n) test=1;
      if (y==10+n)
      { fclose(fparm);
        return(1);
      }
      break;
    default :
      if (isalnum(key&0xFF)&&y<9+n)
      { window(x,y,x+20,y);
        clrscr();
        cprintf("%c",key);
        ungetch(key);
        switch(y)
        { case 4 :
            cscanf("%s",p->inv1);
            break;
          case 5 :
            if (n==1)
            cscanf("%s",p->uitv);
            if (n>1)
            cscanf("%s",p->inv2);
            break;
          case 6 :
            if (n==1)
            cscanf("%d",&p->bdag);
            if (n==2)
            cscanf("%s",p->uitv);
            if (n>2)
            cscanf("%s",p->inv3);
            break;
          case 7 :
            if (n==1)
            cscanf("%s",p->s1);
            if (n==2)
            cscanf("%d",&p->bdag);
```

```
  if (n==3)
  cscanf("%s",p->uitv);
  if (n>3)
  cscanf("%s",p->inv4);
  break;
case 8 :
  switch(n)
  { case 1 :
      cscanf("%d",&p->edag);
      break;
    case 2 :
      cscanf("%s",p->s1);
      break;
    case 3 :
      cscanf("%d",&p->bdag);
      break;
    case 4 :
      cscanf("%s",p->uitv);
      break;
    case 5 :
      cscanf("%s",p->inv5);
  }
  break;
case 9 :
  switch(n)
  { case 1 :
      cscanf("%s",p->s2);
      break;
    case 2 :
      cscanf("%d",&p->edag);
      break;
    case 3 :
      cscanf("%s",p->s1);
      break;
    case 4 :
      cscanf("%d",&p->bdag);
      break;
    case 5 :
      cscanf("%s",p->uitv);
      break;
  }
  break;
case 10 :
  switch(n)
  { case 2 :
      cscanf("%s",p->s2);
      break;
    case 3 :
      cscanf("%d",&p->edag);
      break;
    case 4 :
      cscanf("%s",p->s1);
      break;
    case 5 :
      cscanf("%d",&p->bdag);
      break;
  }
  break;
```

```
            case 11 :
              switch(n)
              { case 3 :
                  cscanf("%s",p->s2);
                  break;
                case 4 :
                  cscanf("%d",&p->edag);
                  break;
                case 5 :
                  cscanf("%s",p->s1);
                  break;
              }
              break;
            case 12 :
              switch(n)
              { case 4 :
                  cscanf("%s",p->s2);
                  break;
                case 5 :
                  cscanf("%d",&p->edag);
                  break;
              }
              break;
            case 13 :
              if (n==5)
              cscanf("%s",p->s2);
              break;
          } ch=getch();y++;
        } ch++;
    }
  } while (test!=1);

      /* de parameters worden wegezet in de parameterfile */
  p->btijd=tijdblok(p->s1);
  p->etijd=tijdblok(p->s2);
  fseek(fparm,0L,SEEK_SET);
  fwrite(p,reclen,1,fparm);
  fclose(fparm);
  return(0);
}



int checkfile_template(struct parm *p, int *uit,int n)
{ struct indata { int dag;
                  int tijd;
                  int jaar;
                  int inw;
                } i1,i2;
  struct uitdata { int dagnr;
                   int tijdblok;
                   int jaartal;
                   double template;
                 } u1,u2;
  FILE *fin,*fuit;
  int  ch,i;
  long j,k,lengte,aantal;
  char *infile,t1[6],t2[6];
```

```
for (i=1;i<=n;i++)
{ switch(i)
  { case 1 :
      infile=p->inv1;
      break;
    case 2 :
      infile=p->inv2;
      break;
    case 3 :
      infile=p->inv3;
      break;
    case 4 :
      infile=p->inv4;
      break;
    case 5 :
      infile=p->inv5;
      break;
  }

     /* de invoerfile wordt geopend                    */
  fin=fopen(infile,"rb");

     /* als de file niet bestaat wordt een melding gegeven *
      * op het scherm en wordt teruggekeerd naar de        *
      * parameterlijst via het teruggeven van 0 aan check  */
  if (fin==NULL)
  { clrscr();
    gotoxy(1,5);
    printf("\t\tDe invoerfile %s bestaat niet\n"
           "\t\tU keert terug naar de parameterlijst",infile);
    schrijf(10,25,"Druk op een toets");
    getch();
    return(0);
  }

     /* begin- en eindpunt van de file worden opgezocht */
  j=sizeof(struct indata);
  fseek(fin,0L,SEEK_SET);
  fread(&i1,j,1,fin);
  fseek(fin,0L,SEEK_END);
  lengte=ftell(fin);
  aantal=lengte/j;
  do
  { fseek(fin,(aantal-1)*j,SEEK_SET);
    fread(&i2,j,1,fin);
    aantal--;
  } while (i2.dag==0);
```

```
     /* gekeken wordt of de opgegeven periode binnen   *
      * de file valt   zoniet dan wordt teruggekeerd   *
      * naar de parameterlijst                         */
  if ( p->btijd<i1.tijd || p->etijd>i2.tijd )
  { clrscr();
    tijd(i1.tijd,t1);
    tijd2(i2.tijd,t2);
    gotoxy(1,5);
    printf("\t\tInvoerfile %s\n\n"
           "\t\t  begin  tijd : %s\n"
           "\t\t  einde  tijd : %s"
           ,infile,t1,t2);

    printf("\n\n\t\tde opgegeven periode valt niet binnen %s\n"
           "\t\tU keert terug naar de parameterlijst!",infile);
    schrijf(10,25,"Druk op een toets");
    getch();
    fclose(fin);
    return(0);
  }

     /* valt de periode wel binnen de file             */
  fclose(fin);
}



     /* de uitvoerfile wordt geopend                   *
      * bestaat de file al dan wordt gekeken of moet   *
      * worden overschreven of toegevoegd en de vlag   *
      * uit wordt gezet: uit=0 file bestaat niet,      *
      * uit=1 toevoegen, uit=2 overschrijven           */
fuit=fopen(p->uitv,"rb");

     /* de file bestaat nog niet                       *
      * openen kan geen kwaad er wordt teruggekeerd en *
      * check krijgt de waarde 1, zodat het programma  *
      * verder kan gaan                                */
if (fuit==NULL)
{ *uit=0;
  return(1);
}

     /* bestaat de file wel dan worden begin en eindpunt *
      * opgezocht en op het scherm gezet                  */
k=sizeof(struct uitdata);
fseek(fuit,0L,SEEK_SET);
fread(&u1,k,1,fuit);
fseek(fuit,0L,SEEK_END);
lengte=ftell(fuit);
aantal=lengte/k;
fseek(fuit,(aantal-1)*k,SEEK_SET);
fread(&u2,k,1,fuit);
tijd(u1.tijdblok,t1);
tijd2(u2.tijdblok,t2);
```

```
  clrscr();
  gotoxy(1,5);
  printf("\t\tUitvoerfile %s\n"
         "\t\t  begin   dagnr : %d\n"
         "\t\t           tijd : %s\n"
         "\t\t  einde   dagnr : %d\n"
         "\t\t           tijd : %s\n"
         ,p->uitv,u1.dagnr,t1,u2.dagnr,t2);
  fclose(fuit);

       /* ligt het beginpunt voor het einde van de file  *
        * dan wordt gevraagd of men wil overschrijven    *
        * uit=2 zoniet dan wordt gevraagd of men wil     *
        * toevoegen uit=1 wil men niet overschrijven of  *
        * toevoegen dan keert men terug naar de          *
        * parameterlijst                                 */
  if (p->bdag>u2.dagnr||(p->bdag==u2.dagnr&&p->btijd>u2.tijdblok))
  { printf("\n\n\t\tWilt U toevoegen aan het end van de file? (J/N)");
    ch=getch();
    if (ch=='J'||ch=='j')
    { *uit=1;
      return(1);
    }
    return(0);
  }
  printf("\n\n\t\tWilt U de file overschrijven? (J/N)");
  ch=getch();
  if (ch=='J'||ch=='j')
  { *uit=2;
    return(1);
  }
  return(0);
}



void gemiddelde(struct parm *p,int uit,int n)
{ struct indata { int dag;
                  int tijd;
                  int jaar;
                  double waarde;
                } i1,i2,i3,i4,i5;
  struct uitdata { int dagnr;
                   int tijdblok;
                   int jaartal;
                   double template;
                 } u;
  FILE   *fin1,*fin2,*fin3,*fin4,*fin5,*fuit;
  long   j,k,y,s,t;
  int    extra=0,a,i;
  double m,temp;

       /* de invoerfiles worden geopend en het eerste    *
        * blok dat moet worden gelezen wordt opgezocht   */
  j=sizeof(struct indata);
```

```
fin1=fopen(p->inv1,"rb");
fseek(fin1,0L,SEEK_SET);
fread(&i1,j,1,fin1);
y=p->btijd-i1.tijd;
fseek(fin1,y*j,SEEK_SET);

fin2=fopen(p->inv2,"rb");
fseek(fin2,0L,SEEK_SET);
fread(&i2,j,1,fin2);
y=p->btijd-i2.tijd;
fseek(fin2,y*j,SEEK_SET);

if (n>2)
{ fin3=fopen(p->inv3,"rb");
  fseek(fin3,0L,SEEK_SET);
  fread(&i3,j,1,fin3);
  y=p->btijd-i3.tijd;
  fseek(fin3,y*j,SEEK_SET);
}
if (n>3)
{ fin4=fopen(p->inv4,"rb");
  fseek(fin4,0L,SEEK_SET);
  fread(&i4,j,1,fin4);
  y=p->btijd-i4.tijd;
  fseek(fin4,y*j,SEEK_SET);
}
if (n>4)
{ fin5=fopen(p->inv5,"rb");
  fseek(fin5,0L,SEEK_SET);
  fread(&i5,j,1,fin5);
  y=p->btijd-i5.tijd;
  fseek(fin5,y*j,SEEK_SET);
}


     /* de uitvoerfile wordt geopend en de             *
      * schrijfpointer wordt op de juiste plaats gezet *
      * afhankelijk van de waarde van uit              */
k=sizeof(struct uitdata);
switch(uit)
{ case 0 :
    fuit=fopen(p->uitv,"w+b");
    break;
  case 1 :
    fuit=fopen(p->uitv,"ab");
    break;
  case 2 :
    fuit=fopen(p->uitv,"r+b");
    fseek(fuit,0L,SEEK_SET);
    fread(&u,k,1,fuit);
```

```
    if(p->bdag<u.dagnr||(p->bdag==u.dagnr&&p->btijd<u.tijdblok))
    { fseek(fuit,0L,SEEK_SET);
    }
    else
    { s=0;
      while (p->bdag>u.dagnr)
      { s+=100;
        fseek(fuit,s*k,SEEK_SET);
        fread(&u,k,1,fuit);
      }
      t=p->btijd-u.tijdblok;
      fseek(fuit,(s+t)*k,SEEK_SET);
    }
}

clrscr();
schrijf(35,12,"BEZIG");
do
{ fread(&i1,j,1,fin1);
  fread(&i2,j,1,fin2);
  if (n>2)
    fread(&i3,j,1,fin3);
  if (n>3)
    fread(&i4,j,1,fin4);
  if (n>4)
    fread(&i5,j,1,fin5);

  temp=0.0;
  a=0;
  for (i=1;i<=n;i++)
  { switch(i)
    { case 1 :
        m=i1.waarde;
        break;
      case 2 :
        m=i2.waarde;
        break;
      case 3 :
        m=i3.waarde;
        break;
      case 4 :
        m=i4.waarde;
        break;
      case 5 :
        m=i5.waarde;
        break;
    }
    if (m>-999)
    { temp+=m;
      a++;
    }
  }
  if (a>0)
    temp=temp/a;
  else
    temp=-1000;
```

```
    u.dagnr=p->bdag+extra;
    u.tijdblok=i1.tijd;
    u.jaartal=i1.jaar;
    u.template=temp;
    extra+=i1.tijd/(MAXTIJD-1);

    fwrite(&u,k,1,fuit);
  } while(u.dagnr<p->edag||u.tijdblok<p->etijd);
  fclose(fin1);
  fclose(fin2);
  if (n>2)
    fclose(fin3);
  if (n>3)
    fclose(fin4);
  if (n>3)
    fclose(fin5);
  fclose(fuit);
}
```

Appendix F

Verschilmodule

```
#include "c:\users\hans\eigen.h"

       /* een structure wordt gedefinieerd waarin alle   *
        * parameters worden opgeborgen die binnen        *
        * verschil worden gebruikt                       *
        * de structure wordt globaal gedefinieerd zodat  *
        * alle parameters in een keer kunnen worden      *
        * ingelezen en voor iedere functie binnen        *
        * verschil bereikbaar zijn                      */
static struct parm { char inv1[15];        /* naam invoerfile1  */
                     char inv2[15];        /* naam invoerfile2  */
                     char uitv[15];        /* naam uitvoerfile  */
                     int      bdag;        /* begin dagnr.      */
                     int     btijd;        /* begin tijdblok    */
                     int      edag;        /* eind dagnr.       */
                     int     etijd;        /* eind tijdblok     */
                     char    s1[6];
                     char    s2[6];
                   } par;
```

```
void verschil(void)
{ int    parameters_verschil(struct parm *p),
         checkfile_verschil(struct parm *p, int *uit);
  void   trekaf(struct parm *p,int uit);
  int    terug,check,uitvlag;

      /* verschil berekent het verschil tussen de        *
       * waarden in twee files                           */

  do
  {   /* de parameters worden uit de parameterfile gelezen    *
       * de gebruiker kan ze wijzigen totdat hij tevreden is  *
       * dan geeft de functie de waarde nul aan terug         *
       * zet de parameters in de structure en schrijft ze     *
       * in de parameterfile                                  *
       * of hij kan teruggaan naar het keuzemenu dan geeft    *
       * de functie de waarde 1 aan terug                     */
    terug=parameters_verschil(&par);

      /* als terug=1 wordt de functie verlaten en men    *
       * komt vanzelf weer in het keuzemenu             */
    if (terug==1) return;

      /* het bestaan van de opgegeven files wordt gecheckt    *
       * is alles in orde dan geeft de functie 1 terug en     *
       * kan het programma verder gaan                        *
       * is er iets niet in orde dan geeft de functie 0 terug *
       * en men keert terug naar parameters zodat men de      *
       * parameters kan wijzigen of vandaar terugkeren        *
       * naar het keuzemenu                                   */
    check=checkfile_verschil(&par,&uitvlag);
  } while (check!=1);

      /* blok voor blok worden de waarden van de invoer  *
       * files van elkaar afgetrokken en het resultaat   *
       * inde uitvoerfile gezet                          */
  trekaf(&par,uitvlag);

      /* gereedmelding wordt gegeven                     */
  clrscr();
  schrijf(15,12,"Verschilbepaling beeindigd");
  schrijf(15,25,"Druk op een toets");
  getch();
}
```

```
int parameters_verschil(struct parm *p)
{ long reclen;
  FILE *fparm;
  int i,x,y,key,test=0,ch;

     /* parameterfile wordt geopend                          */
  reclen=sizeof(struct parm);
  fparm=fopen("Verschil.par","r+b");

     /* als de file niet bestaat wordt hij gemaakt           *
      * en gevuld met nullen                                 */
  if (fparm==NULL)
  { fparm=fopen("Verschil.par","w+b");
    for (i=0;i<reclen;i++)
    { putc('\0',fparm);
    }
  }

     /* lees file en zet de data in de structure             */
  fseek(fparm,0L,SEEK_SET);
  fread(p,reclen,1,fparm);

     /* de parameternamen worden op het scherm gezet         */
  clrscr();
  printf("\n\tParameterlijst\n\n"
       ·"\t          invoerfilenaam1 :\n"
        "\t          invoerfilenaam2 :\n"
        "\t          uitvoerfilenaam :\n"
        "\t          begin dagnummer :\n"
        "\t               begin tijd :\n"
        "\t           eind dagnummer :\n"
        "\t                eind tijd :\n"
        "\t                                Tevreden\n"
        "\t                                Terug naar keuzemenu");

     /* de parameterwaarden worden op het scherm gezet       */
  x=39;y=4;
  window(x,y,80,10);
  cprintf("%.14s\r\n"
          "%.14s\r\n"
          "%.14s\r\n"
          "%d\r\n"
          "%.5s\r\n"
          "%d\r\n"
          "%.5s"
          ,p->inv1,p->inv2,p->uitv,p->bdag,p->s1,p->edag,p->s2);

     /* met de up en down toetsen kan de cursor op de        *
      * parameterwaarden worden gezet en kunnen deze worden  *
      * veranderd                                            *
      * enter op tevreden zet de configuratie in de par-file *
      * en geeft 0 terug                                     *
      * enter op terug naar keuzemenu geeft 1 terug zodat    *
      * teruggegaan wordt naar het keuzemenu                 */
```

```
y=11;
do
{ window(1,1,80,25);
  gotoxy(x,y);
  while(bioskey(1)==0);
  key=bioskey(0);
  switch(key)
  { case 0x5000 :
      y=y+1;
      if (y>12) y=4;
      break;
    case 0x4800 :
      y=y-1;
      if (y<4) y=12;
      break;
    case 0x1c0d :
      if (y==11) test=1;
      if (y==12)
      { fclose(fparm);
        return(1);
      }
      break;
    default :
      if (isalnum(key&0xFF)&&y<11)
      { window(x,y,x+20,y);
        clrscr();
        cprintf("%c",key);
        ungetch(key);
        switch(y)
        { case 4 :
            cscanf("%s",p->inv1);
            break;
          case 5 :
            cscanf("%s",p->inv2);
            break;
          case 6 :
            cscanf("%s",p->uitv);
            break;
          case 7 :
            cscanf("%d",&p->bdag);
            break;
          case 8 :
            cscanf("%s",p->s1);
            break;
          case 9 :
            cscanf("%d",&p->edag);
            break;
          case 10 :
            cscanf("%s",p->s2);
            break;
        } ch=getch();y++;
      } ch++;
  }
} while (test!=1);
```

```
    /* de parameters worden wegezet in de parameterfile */
  p->btijd=tijdblok(p->s1);
  p->etijd=tijdblok(p->s2);
  fseek(fparm,0L,SEEK_SET);
  fwrite(p,reclen,1,fparm);
  fclose(fparm);
  return(0);
}


int checkfile_verschil(struct parm *p, int *uit)
{ struct indata { int dag;
                  int tijd;
                  int jaar;
                  double meetwaarde;
                } i1,i2;
  struct uitdata { int dagnr;
                   int tijdblok;
                   int jaartal;
                   double mw;
                 } u1,u2;
  FILE *fin1,*fin2,*fuit;
  int  ch;
  long j,k,lengte,aantal;
  char t1[6],t2[6];

      /* de invoerfiles worden geopend                    *
       * als een file niet bestaat wordt een melding      *
       * gegeven op het scherm en wordt teruggekeerd      *
       * naar de parameterlijst via het teruggeven        *
       * van 0 aan check                                  */
  fin1=fopen(p->inv1,"rb");
  if (fin1==NULL)
  { clrscr();
    gotoxy(1,5);
    printf("\t\tDe invoerfile %s bestaat niet\n"
           "\t\tU keert terug naar de parameterlijst",p->inv1);
    schrijf(10,25,"Druk op een toets");
    getch();
    return(0);
  }
  fin2=fopen(p->inv2,"rb");
  if (fin2==NULL)
  { clrscr();
    gotoxy(1,5);
    printf("\t\tDe invoerfile %s bestaat niet\n"
           "\t\tU keert terug naar de parameterlijst",p->inv2);
    schrijf(10,25,"Druk op een toets");
    getch();
    fclose(fin1);
    return(0);
  }
```

```
     /* begin- en eindpunten van de files worden       *
      * opgezocht                                      */
j=sizeof(struct indata);
fseek(fin1,0L,SEEK_SET);
fread(&i1,j,1,fin1);
fseek(fin1,0L,SEEK_END);
lengte=ftell(fin1);
aantal=lengte/j;
do
{ fseek(fin1,(aantal-1)*j,SEEK_SET);
  fread(&i2,j,1,fin1);
  aantal--;
} while (i2.dag==0);

     /* gekeken wordt of de opgegeven periode binnen   *
      * de file valt                                   *
      * zoniet dan wordt teruggekeerd naar de          *
      * parameterlijst                                 */
if ((p->bdag<i1.dag||(p->bdag==i1.dag&&p->btijd<i1.tijd))||
    (p->edag>i2.dag||(p->edag==i2.dag&&p->etijd>i2.tijd)))
{ clrscr();
  tijd(i1.tijd,t1);
  tijd2(i2.tijd,t2);
  gotoxy(1,5);
  printf("\t\tInvoerfile %s\n"
       "\t\t  begin   dagnr : %d\n"
       "\t\t           tijd : %s\n"
       "\t\t  einde   dagnr : %d\n"
       "\t\t           tijd : %s"
       ,p->inv1,i1.dag,t1,i2.dag,t2);
  printf("\n\n\t\tde opgegeven periode valt niet binnen %s\n"
         "\t\tU keert terug naar de parameterlijst!",p->inv1);
  schrijf(10,25,"Druk op een toets");
  getch();
  fclose(fin1);
  fclose(fin2);
  return(0);
}

fseek(fin2,0L,SEEK_SET);
fread(&i1,j,1,fin2);
fseek(fin2,0L,SEEK_END);
lengte=ftell(fin2);
aantal=lengte/j;
do
{ fseek(fin2,(aantal-1)*j,SEEK_SET);
  fread(&i2,j,1,fin2);
  aantal--;
} while (i2.dag==0);
```

```
     /* gekeken wordt of de opgegeven periode binnen   *
      * de file valt                                   *
      * zoniet dan wordt teruggekeerd naar de          *
      * parameterlijst                                 */
if ((p->bdag<i1.dag||(p->bdag==i1.dag&&p->btijd<i1.tijd))||
    (p->edag>i2.dag||(p->edag==i2.dag&&p->etijd>i2.tijd)))
{ clrscr();
  tijd(i1.tijd,t1);
  tijd2(i2.tijd,t2);
  gotoxy(1,5);
  printf("\t\tInvoerfile %s\n"
       "\t\t  begin   dagnr : %d\n"
       "\t\t           tijd : %s\n"
       "\t\t  einde   dagnr : %d\n"
       "\t\t           tijd : %s"
       ,p->inv2,i1.dag,t1,i2.dag,t2);
  printf("\n\n\t\tde opgegeven periode valt niet binnen %s\n"
         "\t\tU keert terug naar de parameterlijst!",p->inv2);
  schrijf(10,25,"Druk op een toets");
  getch();
  fclose(fin1);
  fclose(fin2);
  return(0);
}

     /* valt de periode wel binnen de file             */
fclose(fin1);
fclose(fin2);

     /* de uitvoerfile wordt geopend                   *
      * bestaat de file al dan wordt gekeken of moet   *
      * worden overschreven of toegevoegd en de vlag   *
      * uit wordt gezet: uit=0 file bestaat niet,      *
      * uit=1 toevoegen, uit=2 overschrijven           */
fuit=fopen(p->uitv,"rb");

     /* de file bestaat nog niet                       *
      * openen kan geen kwaad er wordt teruggekeerd en *
      * check krijgt de waarde 1, zodat het programma  *
      * verder kan gaan                                */
if (fuit==NULL)
{ *uit=0;
  return(1);
}

     /* bestaat de file wel dan worden begin en eindpunt *
      * opgezocht en op het scherm gezet                 */
k=sizeof(struct uitdata);
fseek(fuit,0L,SEEK_SET);
fread(&u1,k,1,fuit);
fseek(fuit,0L,SEEK_END);
lengte=ftell(fuit);
aantal=lengte/k;
fseek(fuit,(aantal-1)*k,SEEK_SET);
fread(&u2,k,1,fuit);
tijd(u1.tijdblok,t1);
tijd2(u2.tijdblok,t2);
```

```
  clrscr();
  gotoxy(1,5);
  printf("\t\tUitvoerfile %s\n"
         "\t\t  begin   dagnr : %d\n"
         "\t\t        tijdblok : %d\n"
         "\t\t  einde   dagnr : %d\n"
         "\t\t        tijdblok : %d\n"
         ,p->uitv,u1.dagnr,t1,u2.dagnr,t2);
  fclose(fuit);

       /* ligt het beginpunt voor het einde van de file  *
        * dan wordt gevraagd of men wil overschrijven     *
        * uit=2 zoniet dan wordt gevraagd of men wil      *
        * toevoegen uit=1 wil men niet overschrijven of   *
        * toevoegen dan keert men terug naar de           *
        * parameterlijst                                  */
  if (p->bdag>u2.dagnr||(p->bdag==u2.dagnr&&p->btijd>u2.tijdblok))
  { printf("\n\n\t\tWilt U toevoegen aan het end van de file? (J/N)");
    ch=getch();
    if (ch=='J'||ch=='j')
    { *uit=1;
      return(1);
    }
    return(0);
  }
  printf("\n\n\t\tWilt U de file overschrijven? (J/N)");
  ch=getch();
  if (ch=='J'||ch=='j')
  { *uit=2;
    return(1);
  }
  return(0);
}



void trekaf(struct parm *p,int uit)
{ struct indata { int dag;
                  int tijd;
                  int jaar;
                  double mw;
                } i1,i2;
  struct uitdata { int dagnr;
                   int tijdblok;
                   int jaartal;
                   double vs;
                 } u;
  FILE   *fin1,*fin2,*fuit;
  long   j,k,x,y,s,t;
```

```
     /* de invoerfiles worden geopend en het eerste    *
      * blok dat moet worden gelezen wordt opgezocht   */
j=sizeof(struct indata);
fin1=fopen(p->inv1,"rb");
fin2=fopen(p->inv2,"rb");
x=-100;
do
{ x+=100;
  fseek(fin1,x*j,SEEK_SET);
  fread(&i1,j,1,fin1);
} while(i1.dag<p->bdag);
y=p->btijd-i1.tijd;
fseek(fin1,(x+y)*j,SEEK_SET);

x=-100;
do
{ x+=100;
  fseek(fin2,x*j,SEEK_SET);
  fread(&i2,j,1,fin2);
} while(i2.dag<p->bdag);
y=p->btijd-i2.tijd;
fseek(fin2,(x+y)*j,SEEK_SET);

     /* de uitvoerfile wordt geopend en de             *
      * schrijfpointer wordt op de juiste plaats gezet *
      * afhankelijk van de waarde van uit              */
k=sizeof(struct uitdata);
switch(uit)
{ case 0 :
    fuit=fopen(p->uitv,"w+b");
    break;
  case 1 :
    fuit=fopen(p->uitv,"ab");
    break;
  case 2 :
    fuit=fopen(p->uitv,"r+b");
    fseek(fuit,0L,SEEK_SET);
    fread(&u,k,1,fuit);

    if(p->bdag<u.dagnr||(p->bdag==u.dagnr&&p->btijd<u.tijdblok))
    { fseek(fuit,0L,SEEK_SET);
    }
    else
    { s=0;
      while (p->bdag>u.dagnr)
      { s+=100;
        fseek(fuit,s*k,SEEK_SET);
        fread(&u,k,1,fuit);
      }
      t=p->btijd-u.tijdblok;
      fseek(fuit,(s+t)*k,SEEK_SET);
    }
}
```

```
  clrscr();
  schrijf(35,12,"BEZIG");
  do
  { fread(&i1,j,1,fin1);
    fread(&i2,j,1,fin2);

    u.dagnr=i1.dag;
    u.tijdblok=i1.tijd;
    u.jaartal=i1.jaar;
    u.vs=i1.mw-i2.mw;

    fwrite(&u,k,1,fuit);
  } while(u.dagnr<p->edag||u.tijdblok<p->etijd);
  fclose(fin1);
  fclose(fin2);
  fclose(fuit);
}
```

Appendix G

Grafiek

```
# include "c:\users\hans\eigen.h"

       /* een structure wordt gedefinieerd waarin alle   *
        * parameters worden opgeborgen die binnen        *
        * grafiek kunnen worden gebruikt                 *
        * de structure wordt globaal static gedefinieerd *
        * zodat alle parameters in een keer kunnen       *
        * worden ingelezen en voor iedere functie binnen *
        * gafiek bereikbaar zijn                         */
static struct parm { char   eenh1[6];      /* eenheid1              */
                     char    max1[5];      /* maxwaarde grootheid1  */
                     char   inv1[15];      /* naam invoerfile1      */
                     char   eenh2[6];      /* eenheid2              */
                     char    max2[5];      /* maxwaarde grootheid2  */
                     char   inv2[15];      /* naam invoerfile2      */
                     char   eenh3[6];      /* eenheid3              */
                     char    max3[5];      /* maxwaarde grootheid3  */
                     char   inv3[15];      /* naam invoerfile3      */
                     int        bdag;      /* begin dagnr.          */
                     int       btijd;      /* begin tijdblok        */
                     int        edag;      /* eind dagnr.           */
                     int       etijd;      /* eind tijdblok         */
                     char      s1[6];      /* begintijdstring       */
                     char      s2[6];      /* eindtijdstring        */
                   } par;
```

```
void grafiek(int mode)
{ int    parameters_grafiek(struct parm *p,int n,char *srt[]),
         checkfile_grafiek(struct parm *p,int sn[],int i);
  void   teken(struct parm *p,int n,int i,char *srt[],int gm,
         int sn[]),sorts(char *srt[],int sn[],char car,int i);
  int    terug,check,aantal,teller,srtnr[3];
  char   *soort[3],cr;

      /* Grafiek kan grafisch de inhoud van een, twee of *
       * drie op te geven files op het scherm zetten     */


  do
  { clrscr();
    schrijf(15,12,"Hoeveel grafieken wilt U tegelijk? (1, 2 of 3)");
    aantal=getche()-48;
  } while (aantal!=1&&aantal!=2&&aantal!=3);
  clrscr();

  do
  { clrscr();
    schrijf(15,5,"Geef de grootheid die U wilt tekenen!");
    schrijf(15,20,"Geef de hoofdletter van Uw keuze");
    schrijf(15,22,"Cpa, V, L, Demping, Template, vErschil,");
    schrijf(15,23,"Xpd, xpd-I, xpd-Q, Gecorrigeerde xpd, Fase.");
    gotoxy(55,5);
    cr=toupper(getche());
  } while(!is_in(cr,"CVLDTEXIQGF"));
  sorts(soort,srtnr,cr,1);


  if (aantal>1)
  { do
    { schrijf(15,7,"Geef ook de tweede grootheid!          ");
      cr=toupper(getche());
    } while(!is_in(cr,"CVLDTEXIQGF"));
    sorts(soort,srtnr,cr,2);
  }

  if (aantal>2)
  { do
    { schrijf(15,9,"Geef ook de derde grootheid!           ");
      cr=toupper(getche());
    } while(!is_in(cr,"CVLDTEXIQGF"));
    sorts(soort,srtnr,cr,3);
  }

  do
  {   /* de parameters worden uit de parameterfile gelezen    *
       * de gebruiker kan ze wijzigen totdat hij tevreden is  *
       * dan geeft de functie de waarde nul aan terug         *
       * zet de parameters in de structure en schrijft ze     *
       * in de parameterfile                                  *
       * of hij kan teruggaan naar het hoofdmenu dan geeft    *
       * de functie de waarde 1 aan terug                     */
    terug=parameters_grafiek(&par,aantal,soort);
```

```
      /* als terug=1 wordt de functie verlaten en men    *
       * komt vanzelf weer in het hoofdmenu              */
    if (terug==1) return;

      /* het bestaan van de opgegeven files wordt gecheckt    *
       * is alles in orde dan geeft de functie 1 terug en     *
       * kan het programma verder gaan                        *
       * is er iets niet in orde dan geeft de functie 0 terug *
       * en men keert terug naar parameters zodat men de      *
       * parameters kan wijzigen of vandaar terugkeren        *
       * naar het hoofdmenu                                   */
    for (teller=1;teller<=aantal;teller++)
    { check=checkfile_grafiek(&par,srtnr,teller);
      if (check==0) break;
    }
  } while (check!=1);

      /* het opgegeven aantal grafieken wordt getekend   */
  teken(&par,aantal,teller,soort,mode,srtnr);
}
```

```
int parameters_grafiek(struct parm *p,int n,char *srt[])
{ long reclen;
  FILE *fparm;
  int i,x,y,key,test,ch;

     /* parameterfile wordt geopend                          */
  reclen=sizeof(struct parm);
  fparm=fopen("grafiek.par","r+b");

     /* als de file niet bestaat wordt hij gemaakt           *
      * en gevuld met nullen                                 */
  if (fparm==NULL)
  { fparm=fopen("grafiek.par","w+b");
    for (i=0;i<reclen;i++)
    { putc('\0',fparm);
    }
  }

     /* lees file en zet de data in de structure             */
  fseek(fparm,0L,SEEK_SET);
  fread(p,reclen,1,fparm);

     /* de parameternamen worden op het scherm gezet         */
  clrscr();
  printf("\n\tParameterlijst\n\n"
         "\t\t%s\n"
         "\t                    eenheid :\n"
         "\t                    maximum :\n"
         "\t                   filenaam :\n",srt[0]);
  if (n>1)
  { printf("\t\t%s\n"
         "\t                    eenheid :\n"
         "\t                    maximum :\n"
         "\t                   filenaam :\n",srt[1]);
  }
  if (n>2)
  { printf("\t\t%s\n"
         "\t                    eenheid :\n"
         "\t                    maximum :\n"
         "\t                   filenaam :\n",srt[2]);
  }
  printf("\t          begin dagnummer :\n"
         "\t              begin tijd :\n"
         "\t           eind dagnummer :\n"
         "\t                eind tijd :\n"
         "\t                                  Tevreden\n"
         "\t                                  Terug naar hoofdmenu");

     /* de parameterwaarden worden op het scherm gezet       */
  x=39;y=4;
  window(x,y,80,7+n*4);
  cprintf("\r\n%.5s\r\n"
          "%.4s\r\n"
          "%.14s\r\n",p->eenh1,p->max1,p->inv1);
  if (n>1)
  cprintf("\r\n%.5s\r\n"
          "%.4s\r\n"
          "%.14s\r\n",p->eenh2,p->max2,p->inv2);
```

```
if (n>2)
cprintf("\r\n%.5s\r\n"
        "%.4s\r\n"
        "%.14s\r\n",p->eenh3,p->max3,p->inv3);
cprintf("%d\r\n"
        "%s\r\n"
        "%d\r\n"
        "%s"
        ,p->bdag,p->s1,p->edag,p->s2);

    /* met de up en down toetsen kan de cursor op de          *
     * parameterwaarden worden gezet en kunnen deze worden    *
     * veranderd                                              *
     * enter op tevreden zet de configuratie in de par-file   *
     * en geeft 0 terug                                       *
     * enter op terug naar hoofdmenu geeft 1 terug zodat      *
     * teruggegaan wordt naar het hoofdmenu                   */
test=0;
y=8+n*4;
do
{ window(1,1,80,25);
  gotoxy(x,y);
  while(bioskey(1)==0);
  key=bioskey(0);
  switch(key)
  { case 0x5000 :
      y=y+1;
      if (y>9+n*4) y=5;
      if ((n==2||n==3)&&y==8) y=9;
      if (n==3&&y==12) y=13;
      break;
    case 0x4800 :
      y=y-1;
      if (y<5) y=9+n*4;
      if ((n==2||n==3)&&y==8) y=7;
      if (n==3&&y==12) y=11;
      break;
    case 0x1c0d :
      if (y==8+n*4) test=1;
      if (y==9+n*4)
      { fclose(fparm);
        return(1);
      }
      break;
    default :
      if (isalnum(key&0xFF)&&(y<8+n*4))
      { window(x,y,x+20,y);
        clrscr();
        cprintf("%c",key);
        ungetch(key);
        switch(y)
        { case 5 :
            cscanf("%s",p->eenh1);
            break;
          case 6 :
            cscanf("%s",p->max1);
            break;
```

```
case 7 :
  cscanf("%s",p->inv1);
  caps(p->inv1);
  if (n==2||n==3) y=8;
  break;
case 8 :
  if (n==1)
  cscanf("%d",&p->bdag);
  break;
case 9 :
  if (n==1)
  cscanf("%s",p->s1);
  if (n>1)
  cscanf("%s",p->eenh2);
  break;
case 10 :
  if (n==1)
  cscanf("%d",&p->edag);
  if (n>1)
  cscanf("%s",p->max2);
  break;
case 11 :
  if (n==1)
  cscanf("%s",p->s2);
  if (n>1)
  { cscanf("%s",p->inv2);
    caps(p->inv2);
  }
  if (n==3) y=12;
  break;
case 12 :
  if (n==2)
  cscanf("%d",&p->bdag);
  break;
case 13 :
  if (n==2)
  cscanf("%s",p->s1);
  if (n>2)
  cscanf("%s",p->eenh3);
  break;
case 14 :
  if (n==2)
  cscanf("%d",&p->edag);
  if (n>2)
  cscanf("%s",p->max3);
  break;
case 15 :
  if (n==2)
  cscanf("%s",p->s2);
  if (n>2)
  { cscanf("%s",p->inv3);
    caps(p->inv3);
  }
  break;
case 16 :
  if (n==3)
  cscanf("%d",&p->bdag);
  break;
```

```
            case 17 :
              if (n==3)
              cscanf("%s",p->s1);
              break;
            case 18 :
              cscanf("%d",&p->edag);
              break;
            case 19 :
              cscanf("%s",p->s2);
              break;
          } ch=getch();y++;
        } ch++;
    }
  } while (test!=1);

      /* de parameters worden wegezet in de parameterfile */
  p->btijd=tijdblok(p->s1);
  p->etijd=tijdblok(p->s2);
  fseek(fparm,0L,SEEK_SET);
  fwrite(p,reclen,1,fparm);
  fclose(fparm);
  return(0);
}



int checkfile_grafiek(struct parm *p,int sn[],int i)
{ struct ineen  { int dag;
                  int tijd;
                  int jaar;
                  double meetwaarde;
                } ie1,ie2;
  struct intwee { int dag;
                  int tijd;
                  int jaar;
                  double waarde;
                  double waarde2;
                  double waarde3;
                } it1,it2;
  FILE  *fin;
  int   ch;
  long  j,k,lengte,aantal;
  char  *infile;

  if (i==1) infile=p->inv1;
  if (i==2) infile=p->inv2;
  if (i==3) infile=p->inv3;

       /* de invoerfile wordt geopend                    */
  fin=fopen(infile,"rb");
```

```
     /* als de file niet bestaat wordt een melding gegeven *
      * op het scherm en wordt teruggekeerd naar de        *
      * parameterlijst via het teruggeven van 0 aan check  */
if (fin==NULL)
{ clrscr();
  gotoxy(1,5);
  printf("\t\tDe invoerfile %s bestaat niet\n"
         "\t\tU keert terug naar de parameterlijst",infile);
  schrijf(10,25,"Druk op een toets");
  getch();
  return(0);
}

     /* afhankelijk van de soort invoerfile wordt      *
      * gekeken of de file van de opgegeven soort is   *
      * en of de opgegeven periode hierin valt         */
clrscr();
switch(sn[i-1])
{ case 1 :

     /* gekeken wordt of de filenaam bij de opgeven    *
      * soort hoort                                    */
    if (strncmpi(infile,"CPA",3)!=0)
    { schrijf(15,10,"Invoerfile is geen CPA-file!");
      schrijf(15,12,"U keert terug naar de parameterfile");
      schrijf(15,25,"Druk op een toets");
      getch();
      fclose(fin);
      return(0);
    }

     /* begin en eindpunt van de file worden gezocht   */
    j=sizeof(struct ineen);
    fseek(fin,0L,SEEK_SET);
    fread(&ie1,j,1,fin);
    fseek(fin,0L,SEEK_END);
    lengte=ftell(fin);
    aantal=lengte/j;
    do
    { fseek(fin,(aantal-1)*j,SEEK_SET);
      fread(&ie2,j,1,fin);
      aantal--;
    } while (ie2.dag==0);
    clrscr();

     /* gekeken wordt of de opgegeven periode binnen   *
      * de file valt                                   */
    if ((p->bdag<ie1.dag||(p->bdag==ie1.dag&&p->btijd<ie1.tijd))||
       (p->edag>ie2.dag||(p->edag==ie2.dag&&p->etijd>ie2.tijd)))
    { printf("\n\n\t\tde opgegeven periode valt niet binnen %s\n"
             "\t\tU keert terug naar de parameterlijst!",infile);
      schrijf(15,25,"Druk op een toets");
      getch();
      fclose(fin);
      return(0);
    }
    break;
```

```
case 2 :
case 3 :

   /* gekeken wordt of de filenaam bij de opgeven    *
    * soort hoort                                    */
  if (strncmpi(infile,"VENL",4)!=0)
  { schrijf(15,10,"Invoerfile is geen VENL-file!");
    schrijf(15,12,"U keert terug naar de parameterfile");
    schrijf(15,25,"Druk op een toets");
    getch();
    fclose(fin);
    return(0);
  }

   /* begin en eindpunt van de file worden gezocht   */
  j=sizeof(struct intwee);
  fseek(fin,0L,SEEK_SET);
  fread(&it1,j,1,fin);
  fseek(fin,0L,SEEK_END);
  lengte=ftell(fin);
  aantal=lengte/j;
  do
  { fseek(fin,(aantal-1)*j,SEEK_SET);
    fread(&it2,j,1,fin);
    aantal--;
    } while (it2.dag==0);
  clrscr();

   /* gekeken wordt of de opgegeven periode binnen   *
    * de file valt                                   */
  if ((p->bdag<it1.dag||(p->bdag==it1.dag&&p->btijd<it1.tijd))||
     (p->edag>it2.dag||(p->edag==it2.dag&&p->etijd>it2.tijd)))
  { printf("\n\n\t\tde opgegeven periode valt niet binnen %s\n"
           "\t\tU keert terug naar de parameterlijst!",infile);
    schrijf(15,25,"Druk op een toets");
    getch();
    fclose(fin);
    return(0);
  }
  break;

case 4 :

   /* gekeken wordt of de filenaam bij de opgeven    *
    * soort hoort                                    */
  if (strncmpi(infile,"demp",4)!=0)
  { schrijf(15,10,"Invoerfile is geen Demping-file!");
    schrijf(15,12,"U keert terug naar de parameterfile");
    schrijf(15,25,"Druk op een toets");
    getch();
    fclose(fin);
    return(0);
  }
```

```
   /* begin en eindpunt van de file worden gezocht   */
  j=sizeof(struct ineen);
  fseek(fin,0L,SEEK_SET);
  fread(&ie1,j,1,fin);
  fseek(fin,0L,SEEK_END);
  lengte=ftell(fin);
  aantal=lengte/j;
  do
  { fseek(fin,(aantal-1)*j,SEEK_SET);
    fread(&ie2,j,1,fin);
    aantal--;
  } while (ie2.dag==0);
  clrscr();

   /* gekeken wordt of de opgegeven periode binnen   *
    * de file valt                                   */
  if ((p->bdag<ie1.dag||(p->bdag==ie1.dag&&p->btijd<ie1.tijd))||
     (p->edag>ie2.dag||(p->edag==ie2.dag&&p->etijd>ie2.tijd)))
  { printf("\n\n\t\tde opgegeven periode valt niet binnen %s\n"
           "\t\tU keert terug naar de parameterlijst!",infile);
    schrijf(15,25,"Druk op een toets");
    getch();
    fclose(fin);
    return(0);
  }
  break;

case 5 :

   /* gekeken wordt of de filenaam bij de opgeven    *
    * soort hoort                                    */
  if (strncmpi(infile,"TEMP",4)!=0)
  { schrijf(15,10,"Invoerfile is geen Template-file!");
    schrijf(15,12,"U keert terug naar de parameterfile");
    schrijf(15,25,"Druk op een toets");
    getch();
    fclose(fin);
    return(0);
  }

   /* begin en eindpunt van de file worden gezocht   */
  j=sizeof(struct ineen);
  fseek(fin,0L,SEEK_SET);
  fread(&ie1,j,1,fin);
  fseek(fin,0L,SEEK_END);
  lengte=ftell(fin);
  aantal=lengte/j;
  do
  { fseek(fin,(aantal-1)*j,SEEK_SET);
    fread(&ie2,j,1,fin);
    aantal--;
  } while (ie2.dag==0);
  clrscr();
```

```
   /* gekeken wordt of de opgegeven periode binnen   *
    * de file valt                                   */
  if ((p->bdag<ie1.dag||(p->bdag==ie1.dag&&p->btijd<ie1.tijd))||
     (p->edag>ie2.dag||(p->edag==ie2.dag&&p->etijd>ie2.tijd)))
  { printf("\n\n\t\tde opgegeven periode valt niet binnen %s\n"
           "\t\tU keert terug naar de parameterlijst!",infile);
    schrijf(15,25,"Druk op een toets");
    getch();
    fclose(fin);
    return(0);
  }
  break;

case 6 :

   /* gekeken wordt of de filenaam bij de opgeven    *
    * soort hoort                                    */
  if (strncmpi(infile,"VERS",4)!=0)
  { schrijf(15,10,"Invoerfile is geen Verschil-file!");
    schrijf(15,12,"U keert terug naar de parameterfile");
    schrijf(15,25,"Druk op een toets");
    getch();
    fclose(fin);
    return(0);
  }

   /* begin en eindpunt van de file worden gezocht   */
  j=sizeof(struct ineen);
  fseek(fin,0L,SEEK_SET);
  fread(&ie1,j,1,fin);
  fseek(fin,0L,SEEK_END);
  lengte=ftell(fin);
  aantal=lengte/j;
  do
  { fseek(fin,(aantal-1)*j,SEEK_SET);
    fread(&ie2,j,1,fin);
    aantal--;
  } while (ie2.dag==0);
  clrscr();

   /* gekeken wordt of de opgegeven periode binnen   *
    * de file valt                                   */
  if ((p->bdag<ie1.dag||(p->bdag==ie1.dag&&p->btijd<ie1.tijd))||
     (p->edag>ie2.dag||(p->edag==ie2.dag&&p->etijd>ie2.tijd)))
  { printf("\n\n\t\tde opgegeven periode valt niet binnen %s\n"
           "\t\tU keert terug naar de parameterlijst!",infile);
    schrijf(15,25,"Druk op een toets");
    getch();
    fclose(fin);
    return(0);
  }
  break;
```

```
case 7 :

   /* gekeken wordt of de filenaam bij de opgeven    *
    * soort hoort                                    */
  if (strncmpi(infile,"XPD",3)!=0)
  { schrijf(15,10,"Invoerfile is geen XPD-file!");
    schrijf(15,12,"U keert terug naar de parameterfile");
    schrijf(15,25,"Druk op een toets");
    getch();
    fclose(fin);
    return(0);
  }

   /* begin en eindpunt van de file worden gezocht   */
  j=sizeof(struct ineen);
  fseek(fin,0L,SEEK_SET);
  fread(&ie1,j,1,fin);
  fseek(fin,0L,SEEK_END);
  lengte=ftell(fin);
  aantal=lengte/j;
  do
  { fseek(fin,(aantal-1)*j,SEEK_SET);
    fread(&ie2,j,1,fin);
    aantal--;
  } while (ie2.dag==0);
  clrscr();

   /* gekeken wordt of de opgegeven periode binnen   *
    * de file valt                                   */
  if ((p->bdag<ie1.dag||(p->bdag==ie1.dag&&p->btijd<ie1.tijd))||
     (p->edag>ie2.dag||(p->edag==ie2.dag&&p->etijd>ie2.tijd)))
  { printf("\n\n\t\tde opgegeven periode valt niet binnen %s\n"
           "\t\tU keert terug naar de parameterlijst!",infile);
    schrijf(15,25,"Druk op een toets");
    getch();
    fclose(fin);
    return(0);
  }
  break;

case 8 :

   /* gekeken wordt of de filenaam bij de opgeven    *
    * soort hoort                                    */
  if (strncmpi(infile,"XPDI",4)!=0)
  { schrijf(15,10,"Invoerfile is geen XPD-I-file!");
    schrijf(15,12,"U keert terug naar de parameterfile");
    schrijf(15,25,"Druk op een toets");
    getch();
    fclose(fin);
    return(0);
  }
```

```
   /* begin en eindpunt van de file worden gezocht   */
  j=sizeof(struct ineen);
  fseek(fin,0L,SEEK_SET);
  fread(&ie1,j,1,fin);
  fseek(fin,0L,SEEK_END);
  lengte=ftell(fin);
  aantal=lengte/j;
  do
  { fseek(fin,(aantal-1)*j,SEEK_SET);
    fread(&ie2,j,1,fin);
    aantal--;
  } while (ie2.dag==0);
  clrscr();

   /* gekeken wordt of de opgegeven periode binnen   *
    * de file valt                                   */
  if ((p->bdag<ie1.dag||(p->bdag==ie1.dag&&p->btijd<ie1.tijd))||
     (p->edag>ie2.dag||(p->edag==ie2.dag&&p->etijd>ie2.tijd)))
  { printf("\n\n\t\tde opgegeven periode valt niet binnen %s\n"
           "\t\tU keert terug naar de parameterlijst!",infile);
    schrijf(15,25,"Druk op een toets");
    getch();
    fclose(fin);
    return(0);
  }
  break;

case 9 :

   /* gekeken wordt of de filenaam bij de opgeven    *
    * soort hoort                                    */
  if (strncmpi(infile,"XPDQ",4)!=0)
  { schrijf(15,10,"Invoerfile is geen XPD-Q-file!");
    schrijf(15,12,"U keert terug naar de parameterfile");
    schrijf(15,25,"Druk op een toets");
    getch();
    fclose(fin);
    return(0);
  }

   /* begin en eindpunt van de file worden gezocht   */
  j=sizeof(struct ineen);
  fseek(fin,0L,SEEK_SET);
  fread(&ie1,j,1,fin);
  fseek(fin,0L,SEEK_END);
  lengte=ftell(fin);
  aantal=lengte/j;
  do
  { fseek(fin,(aantal-1)*j,SEEK_SET);
    fread(&ie2,j,1,fin);
    aantal--;
  } while (ie2.dag==0);
  clrscr();
```

```
   /* gekeken wordt of de opgegeven periode binnen   *
    * de file valt                                   */
  if ((p->bdag<ie1.dag||(p->bdag==ie1.dag&&p->btijd<ie1.tijd))||
     (p->edag>ie2.dag||(p->edag==ie2.dag&&p->etijd>ie2.tijd)))
  { printf("\n\n\t\tde opgegeven periode valt niet binnen %s\n"
           "\t\tU keert terug naar de parameterlijst!",infile);
    schrijf(15,25,"Druk op een toets");
    getch();
    fclose(fin);
    return(0);
  }
  break;

case 10 :

   /* gekeken wordt of de filenaam bij de opgeven    *
    * soort hoort                                    */
  if (strncmpi(infile,"XPDC",4)!=0)
  { schrijf(15,10,"Invoerfile is geen Corr_xpd-file!");
    schrijf(15,12,"U keert terug naar de parameterfile");
    schrijf(15,25,"Druk op een toets");
    getch();
    fclose(fin);
    return(0);
  }
      .
   /* begin en eindpunt van de file worden gezocht   */
  j=sizeof(struct ineen);
  fseek(fin,0L,SEEK_SET);
  fread(&ie1,j,1,fin);
  fseek(fin,0L,SEEK_END);
  lengte=ftell(fin);
  aantal=lengte/j;
  do
  { fseek(fin,(aantal-1)*j,SEEK_SET);
    fread(&ie2,j,1,fin);
    aantal--;
  } while (ie2.dag==0);
  clrscr();

   /* gekeken wordt of de opgegeven periode binnen   *
    * de file valt                                   */
  if ((p->bdag<ie1.dag||(p->bdag==ie1.dag&&p->btijd<ie1.tijd))||
     (p->edag>ie2.dag||(p->edag==ie2.dag&&p->etijd>ie2.tijd)))
  { printf("\n\n\t\tde opgegeven periode valt niet binnen %s\n"
           "\t\tU keert terug naar de parameterlijst!",infile);
    schrijf(15,25,"Druk op een toets");
    getch();
    fclose(fin);
    return(0);
  }
  break;
```

```
    case 11 :

       /* gekeken wordt of de filenaam bij de opgeven    *
        * soort hoort                                    */
      if (strncmpi(infile,"FASE",4)!=0)
      { schrijf(15,10,"Invoerfile is geen Fase-file!");
        schrijf(15,12,"U keert terug naar de parameterfile");
        schrijf(15,25,"Druk op een toets");
        getch();
        fclose(fin);
        return(0);
      }

       /* begin en eindpunt van de file worden gezocht   */
      j=sizeof(struct ineen);
      fseek(fin,0L,SEEK_SET);
      fread(&ie1,j,1,fin);
      fseek(fin,0L,SEEK_END);
      lengte=ftell(fin);
      aantal=lengte/j;
      do
      { fseek(fin,(aantal-1)*j,SEEK_SET);
        fread(&ie2,j,1,fin);
        aantal--;
      } while (ie2.dag==0);
      clrscr();

       /* gekeken wordt of de opgegeven periode binnen   *
        * de file valt                                   */
      if ((p->bdag<ie1.dag||(p->bdag==ie1.dag&&p->btijd<ie1.tijd))||
         (p->edag>ie2.dag||(p->edag==ie2.dag&&p->etijd>ie2.tijd)))
      { printf("\n\n\t\tde opgegeven periode valt niet binnen %s\n"
               "\t\tU keert terug naar de parameterlijst!",infile);
        schrijf(15,25,"Druk op een toets");
        getch();
        fclose(fin);
        return(0);
      }
      break;
  }
  fclose(fin);
  return(1);
}
```

```
void teken(struct parm *p,int n,int i,char *srt[],int gm,int sn[])
{ struct indata1 { int dag;
                   int tijd;
                   int jaar;
                   double mw;
                 } in1;
  struct indata2 { int dag;
                   int tijd;
                   int jaar;
                   double mw1;
                   double mw2;
                   double mw3;
                 } in2;
  FILE     *fin;
  long     j,x,y;
  char     *max,*eenh,*infile,aanvang[6],afloop[6],hulp[5];
  int      u,min,abu,top,bottom,maxx,maxy,beeldgrootte,
           xo,xn,yn,mx,volgnr,d,t,ymax;
  double   m,abp,ptn;

  setgraphmode(gm);
  for (i=1;i<=n;i++)
  {    /* de tekststrings om in de tekening te zetten    *
        * worden gereed gemaakt                          */
    if (i==1)
    { max=p->max1;
      eenh=p->eenh1;
      infile=p->inv1;
    }
    if (i==2)
    { max=p->max2;
      eenh=p->eenh2;
      infile=p->inv2;
    }
    if (i==3)
    { max=p->max3;
      eenh=p->eenh3;
      infile=p->inv3;
    }

    strcpy(aanvang,p->s1);
    strcpy(afloop,p->s2);

       /* het assenkruis wordt getekend                  *
        * en van tekst voorzien                          */
    beeldgrootte=getmaxy()/n;
    top=(i-1)*beeldgrootte+1;
    bottom=i*beeldgrootte;

    setviewport(0,top,getmaxx(),bottom,0);
    maxy=bottom-top;
    maxx=getmaxx();
    line(50,5,50,maxy-10);
    line(47,5,53,5);
    line(47,maxy-13,maxx-5,maxy-13);
    line(maxx-5,maxy-10,maxx-5,maxy-16);
```

```
settextjustify(RIGHT_TEXT,TOP_TEXT);
outtextxy(46,3,max);
outtextxy(46,30,eenh);
outtextxy(46,maxy-20,"0");
outtextxy(70,maxy-9,aanvang);
outtextxy(maxx/2,maxy-9,infile);
outtextxy(maxx-5,maxy-9,afloop);
settextjustify(LEFT_TEXT,TOP_TEXT);
outtextxy(maxx/2+30,maxy-9,srt[i-1]);

   /* de invoerfile wordt geopend en het eerste blok *
    * dat moet worden gelezen wordt opgezocht        */
fin=fopen(infile,"rb");
switch(sn[i-1])
{ case 1 :
  case 4 :
  case 5 :
  case 6 :
  case 7 :
  case 8 :
  case 9 :
  case 10 :
  case 11 :
    j=sizeof(struct indata1);
    x=-100;
    do
    { x+=100;
      fseek(fin,x*j,SEEK_SET);
      fread(&in1,j,1,fin);
    } while(in1.dag<p->bdag);
    y=p->btijd-in1.tijd;
    fseek(fin,(x+y)*j,SEEK_SET);
    break;
  case 2 :
  case 3 :
    j=sizeof(struct indata2);
    x=-100;
    do
    { x+=100;
      fseek(fin,x*j,SEEK_SET);
      fread(&in2,j,1,fin);
    } while(in2.dag<p->bdag);
    y=p->btijd-in2.tijd;
    fseek(fin,(x+y)*j,SEEK_SET);
    break;
}

setviewport(50,top+5,maxx-5,bottom-13,1);
ymax=(bottom-13)-(top+5);
xo=-1;
ptn=(double)PUNTEN;
if (p->etijd<p->btijd)
  abp=((p->edag-p->bdag-1)*MAXTIJD+(MAXTIJD-(p->btijd-p->etijd-1)))/ptn;
else
  abp=((p->edag-p->bdag)*MAXTIJD+(p->etijd-p->btijd+1))/ptn;
volgnr=0;
mx=atoi(max);
moveto(0,0);
```

```
    do
    { xn=(int)(volgnr/abp);

      switch(sn[i-1])
        { case 1 :
          case 4 :
          case 5 :
          case 6 :
          case 7 :
          case 8 :
          case 9 :
          case 10:
          case 11 :
            fread(&in1,j,1,fin);
            m=in1.mw;
            d=in1.dag;
            t=in1.tijd;
            break;
          case 2 :
            fread(&in2,j,1,fin);
            m=in2.mw1;
            d=in2.dag;
            t=in2.tijd;
            break;
          case 3 :
            fread(&in2,j,1,fin);
            m=in2.mw2;
            d=in2.dag;
            t=in2.tijd;
            break;
        }
        if (xn>xo)
        { yn=ymax-(int)(m*ymax/mx);
          lineto(xn,yn);
          xo=xn;
        }
      volgnr++;
    } while(d<p->edag||t<p->etijd);
    fclose(fin);
  }
  getch();
  restorecrtmode();
}

void sorts(char *srt[],int sn[],char car,int i)
{ switch(car)
  { case 'C' :
      srt[i-1]="CPA";
      sn[i-1]=1;
      break;
    case 'V' :
      srt[i-1]="V";
      sn[i-1]=2;
      break;
```

```
    case 'L' :
      srt[i-1]="L";
      sn[i-1]=3;
      break;
    case 'D' :
      srt[i-1]="Demping";
      sn[i-1]=4;
      break;
    case 'T' :
      srt[i-1]="Template";
      sn[i-1]=5;
      break;
    case 'E' :
      srt[i-1]="Verschil";
      sn[i-1]=6;
      break;
    case 'X' :
      srt[i-1]="XPD";
      sn[i-1]=7;
      break;
    case 'I' :
      srt[i-1]="XPD-I";
      sn[i-1]=8;
      break;
    case 'Q' :
      srt[i-1]="XPD-Q";
      sn[i-1]=9;
      break;
    case 'G' :
      srt[i-1]="Gecorrigeerde XPD";
      sn[i-1]=10;
      break;
    case 'F' :
      srt[i-1]="Fase";
      sn[i-1]=11;
      break;
  }
}
```

Appendix H

XPD, XPD-I, XPD-Q en Fase bepalingen

```
# include "c:\users\hans\eigen.h"

       /* een structure wordt gedefinieerd waarin alle   *
        * parameters worden opgeborgen die binnen XPD    *
        * worden gebruikt                                *
        * de structure wordt globaal gedefinieerd zodat  *
        * alle parameters in een keer kunnen worden      *
        * ingelezen en voor iedere functie binnen XPD    *
        * bereikbaar zijn                                */

static struct parm { double       ca;     /* cpa conversie       */
                     double       cb;     /*                     */
                     double       xa;     /* xpd conversie       */
                     double       xb;     /*                     */
                     double       fa;     /* fase conversie      */
                     double       fb;     /*                     */
                     char  filin[15];     /* naam invoerfile     */
                     char   uit1[15];     /* naam XPD file       */
                     char   uit2[15];     /* naam XPD-I file     */
                     char   uit3[15];     /* naam XPD-Q file     */
                     char   uit4[15];     /* naam fase file      */
                     int        bdag;     /* begin dagnr.        */
                     int       btijd;     /* begin tijdblok      */
                     int        edag;     /* eind dagnr.         */
                     int       etijd;     /* eind tijdblok       */
                     char      s1[6];     /* begintijdstring     */
                     char      s2[6];     /* eindtijdstring      */
                   } par;
```

```
void xpd(void)
{ int    parameters_xpd(struct parm *p),
         checkfile_xpd(struct parm *p, int *uit);
  void   berekenxpd(struct parm *p,int *uit);
  int    terug,check,uitvlag[4];

      /* XPD converteert meetwaarden naar xpd, xpd-i,     *
       * xpd-q en fase waarden                            */

  do
  {   /* de parameters worden uit de parameterfile gelezen    *
       * de gebruiker kan ze wijzigen totdat hij tevreden is  *
       * dan geeft de functie de waarde nul aan terug         *
       * zet de parameters in de structure en schrijft ze     *
       * in de parameterfile                                  *
       * of hij kan teruggaan naar het keuzemenu dan geeft    *
       * de functie de waarde 1 aan terug                     */
    terug=parameters_xpd(&par);

      /* als terug=1 wordt de functie verlaten en men     *
       * komt vanzelf weer in het keuzemenu              */
    if (terug==1) return;

      /* het bestaan van de opgegeven files wordt gecheckt    *
       * is alles in orde dan geeft de functie 1 terug en     *
       * kan het programma verder gaan                        *
       * is er iets niet in orde dan geeft de functie 0 terug *
       * en men keert terug naar parameters zodat men de      *
       * parameters kan wijzigen of vandaar terugkeren        *
       * naar het keuzemenu                                   */
    check=checkfile_xpd(&par,uitvlag);
  } while (check!=1);

      /* De ingelezen meetwaarden worden omgezet naar     *
       * XPD, XPD-I, XPD-Q en Fase waarden en deze        *
       * worden in de respectievelijke uitvoerfiles       *
       * gezet                                            */
  berekenxpd(&par,uitvlag);

      /* gereedmelding van xpd wordt gegeven              */
  clrscr();
  schrijf(15,12,"Conversie naar XPD, XPD-I, XPD-Q en Fase voltooid");
  schrijf(15,25,"Druk op een toets");
  getch();
}
```

```
int parameters_xpd(struct parm *p)
{ long reclen;
  FILE *fparm;
  int i,x,y,key,test,ch;

      /* parameterfile wordt geopend                          */
  reclen=sizeof(struct parm);
  fparm=fopen("XPD.par","r+b");

      /* als de file niet bestaat wordt hij gemaakt           *
       * en gevuld met nullen                                 */
  if (fparm==NULL)
  { fparm=fopen("XPD.par","w+b");
    for (i=0;i<reclen;i++)
    { putc('\0',fparm);
    }
  }

      /* lees file en zet de data in de structure             */
  fseek(fparm,0L,SEEK_SET);
  fread(p,reclen,1,fparm);

      /* de parameternamen worden op het scherm gezet         */
  clrscr();
  printf("\n\tParameterlijst\n\n"
         "\t                        ca :\n"
         "\t                        cb :\n"
         "\t                        xa :\n"
         "\t                        xb :\n"
         "\t                        fa :\n"
         "\t                        fb :\n"
         "\t           invoerfilenaam :\n"
         "\t             XPD filenaam :\n"
         "\t           XPD-I filenaam :\n"
         "\t           XPD-Q filenaam :\n"
         "\t            fase filenaam :\n"
         "\t          begin dagnummer :\n"
         "\t              begin tijd :\n"
         "\t           eind dagnummer :\n"
         "\t                eind tijd :\n"
         "\t                                Tevreden\n"
         "\t                                Terug naar keuzemenu");

      /* de parameterwaarden worden op het scherm gezet       */
  x=39;y=4;
  window(x,y,80,18 );
  cprintf("%lf\r\n"
          "%lf\r\n"
          "%lf\r\n"
          "%lf\r\n"
          "%lf\r\n"
          "%lf\r\n"
          "%.14s\r\n"
          "%.14s\r\n"
          "%.14s\r\n"
          "%.14s\r\n"
          "%.14s\r\n"
          "%d\r\n"
```

```
        "%.5s\r\n"
        "%d\r\n"
        "%.5s"
        ,p->ca,p->cb,p->xa,p->xb,p->fa,p->fb,p->filin,p->uit1
        ,p->uit2,p->uit3,p->uit4,p->bdag,p->s1,p->edag,p->s2);

    /* met de up en down toetsen kan de cursor op de   *
     * parameterwaarden worden gezet en kunnen deze    *
     * worden veranderd                                *
     * enter op tevreden zet de configuratie in de     *
     * par-file en geeft 0 terug                       *
     * enter op terug naar keuzemenu geeft 1 terug     *
     * zodat teruggegaan wordt naar het keuzemenu      */
y=19;
do
{ window(1,1,80,25);
  gotoxy(x,y);
  while(bioskey(1)==0);
  key=bioskey(0);
  switch(key)
  { case 0x5000 :
      y=y+1;
      if (y>20) y=4;
      break;
    case 0x4800 :
      y=y-1;
      if (y<4) y=20;
      break;
    case 0x1c0d :
      if (y==19) test=1;
      if (y==20)
      { fclose(fparm);
        return(1);
      }
      break;
    default :
      if (isalnum(key&0xFF)&&y<19)
      { window(x,y,x+20,y);
        clrscr();
        cprintf("%c",key);
        ungetch(key);
        switch(y)
        { case 4 :
            cscanf("%lf",&p->ca);
            break;
          case 5 :
            cscanf("%lf",&p->cb);
            break;
          case 6 :
            cscanf("%lf",&p->xa);
            break;
          case 7 :
            cscanf("%lf",&p->xb);
            break;
```

```
            case 8 :
              cscanf("%lf",&p->fa);
              break;
            case 9 :
              cscanf("%lf",&p->fb);
              break;
            case 10 :
              cscanf("%s",p->filin);
              break;
            case 11 :
              cscanf("%s",p->uit1);
              break;
            case 12 :
              cscanf("%s",p->uit2);
              break;
            case 13 :
              cscanf("%s",p->uit3);
              break;
            case 14 :
              cscanf("%s",p->uit4);
              break;
            case 15 :
              cscanf("%d",&p->bdag);
              break;
            case 16 :
              cscanf("%s",p->s1);
              break;
            case 17 :
              cscanf("%d",&p->edag);
              break;
            case 18 :
              cscanf("%s",p->s2);
              break;
          } ch=getch();y++;
        } ch++;
    }
  } while (test!=1);

     /* de parameters worden weggezet in de parameterfile */
  p->btijd=tijdblok(p->s1);
  p->etijd=tijdblok(p->s2);
  fseek(fparm,0L,SEEK_SET);
  fwrite(p,reclen,1,fparm);
  fclose(fparm);
  return(0);
}
```

```
int checkfile_xpd(struct parm *p, int *uit)
{ struct indata { int dag;
                  int tijd;
                  int jaar;
                  int mw1;
                  int mw2;
                  int mw3;
                  int mw4;
                  int mw5;
                  int mw6;
                } i1,i2;
  struct uitdata { int dagnr;
                   int tijdblok;
                   int jaartal;
                   double uitw;
                 } u1,u2;
  FILE *fin,*fuit;
  int  ch,i;
  char t1[6],t2[6],*uitv;
  long j,k,lengte,aantal;

       /* de invoerfile wordt geopend                    */
  fin=fopen(p->filin,"rb");

       /* als de file niet bestaat wordt een melding gegeven *
        * op het scherm en wordt teruggekeerd naar de        *
        * parameterlijst via het teruggeven van 0 aan check  */
  if (fin==NULL)
  { clrscr();
    gotoxy(1,5);
    printf("\t\tDe invoerfile %s bestaat niet\n"
           "\t\tU keert terug naar de parameterlijst",p->filin);
    gotoxy(10,25);
    printf("Druk op een toets");
    getch();
    return(0);
  }

       /* begin- en eindpunt van de file worden opgezocht */
  j=sizeof(struct indata);
  fseek(fin,0L,SEEK_SET);
  fread(&i1,j,1,fin);
  fseek(fin,0L,SEEK_END);
  lengte=ftell(fin);
  aantal=lengte/j;
  do
  { fseek(fin,(aantal-1)*j,SEEK_SET);
    fread(&i2,j,1,fin);
    aantal--;
  } while (i2.dag==0);
```

```
     /* gekeken wordt of de opgegeven periode binnen   *
      * de file valt   zoniet dan wordt teruggekeerd   *
      * naar de parameterlijst                         */
if ((p->bdag<i1.dag||(p->bdag==i1.dag&&p->btijd<i1.tijd))||
    (p->edag>i2.dag||(p->edag==i2.dag&&p->etijd>i2.tijd)))
{ clrscr();
  tijd(i1.tijd,t1);
  tijd2(i2.tijd,t2);
  gotoxy(1,5);
  printf("\t\tInvoerfile %s\n"
       "\t\t  begin   dagnr : %d\n"
       "\t\t           tijd : %s\n"
       "\t\t  einde   dagnr : %d\n"
       "\t\t           tijd : %s"
       ,p->filin,i1.dag,t1,i2.dag,t2);

  printf("\n\n\t\tde opgegeven periode valt niet binnen %s\n"
         "\t\tU keert terug naar de parameterlijst!",p->filin);
  gotoxy(10,25);
  printf("Druk op een toets");
  getch();
  fclose(fin);
  return(0);
}

     /* valt de periode wel binnen de file             */
fclose(fin);

     /* de uitvoerfiles worden geopend                 *
      * bestaat een file al dan wordt gekeken of moet  *
      * worden overschreven of toegevoegd en de vlag   *
      * uit wordt gezet: uit=0 file bestaat niet,      *
      * uit=1 toevoegen, uit=2 overschrijven           */
for (i=1;i<5;i++)
{ if (i==1) uitv=p->uit1;
  if (i==2) uitv=p->uit2;
  if (i==3) uitv=p->uit3;
  if (i==4) uitv=p->uit4;
  fuit=fopen(uitv,"rb");

     /* de file bestaat nog niet                       *
      * openen kan geen kwaad er wordt teruggekeerd en *
      * check krijgt de waarde 1, zodat het programma  *
      * verder kan gaan                                */
  if (fuit==NULL) uit[i-1]=0;

  else
  {
     /* bestaat de file wel dan worden begin en eindpunt *
      * opgezocht en op het scherm gezet                 */
    k=sizeof(struct uitdata);
    fseek(fuit,0L,SEEK_SET);
    fread(&u1,k,1,fuit);
    fseek(fuit,0L,SEEK_END);
    lengte=ftell(fuit);
    aantal=lengte/k;
    fseek(fuit,(aantal-1)*k,SEEK_SET);
    fread(&u2,k,1,fuit);
```

```
      clrscr();
      tijd(u1.tijdblok,t1);
      tijd2(u2.tijdblok,t2);
      gotoxy(1,5);
      printf("\t\tUitvoerfile %s\n"
             "\t\t  begin   dagnr : %d\n"
             "\t\t           tijd : %s\n"
             "\t\t  einde   dagnr : %d\n"
             "\t\t           tijd : %s\n"
             ,uitv,u1.dagnr,t1,u2.dagnr,t2);
      fclose(fuit);

       /* ligt het beginpunt voor het einde van de file  *
        * dan wordt gevraagd of men wil overschrijven    *
        * uit=2 zoniet dan wordt gevraagd of men wil     *
        * toevoegen uit=1 wil men niet overschrijven of  *
        * toevoegen dan keert men terug naar de          *
        * parameterlijst                                 */
      if(p->bdag>u2.dagnr||(p->bdag==u2.dagnr&&p->btijd>u2.tijdblok))
      { printf("\n\n\t\t");
        printf("Wilt U toevoegen aan het eind van de file? (J/N)");
        ch=getch();
        if (ch=='J'||ch=='j') uit[i-1]=1;
        else return(0);
      }
      else
      { printf("\n\n\t\tWilt U de file overschrijven? (J/N)");
        ch=getch();
        if (ch=='J'||ch=='j') uit[i-1]=2;
        else return(0);
      }
    }
  }
  return(1);
}
```

```
void berekenxpd(struct parm *p,int *uit)
{ struct indata { int dag;
                  int tijd;
                  int jaar;
                  int mw1;
                  int mw2;
                  int mw3;
                  int mw4;
                  int mw5;
                  int mw6;
                } i;
  struct uitdata { int dagnr;
                   int tijdblok;
                   int jaartal;
                   double uitw;
                 } u,u1,u2,u3,u4;
  FILE   *fin,*fuit[4];
  long   j,k,x,y,s,t;
  double hp,kp,phase;
  char   *uitv;
  int    teller;

       /* de invoerfile wordt geopend en het eerste blok *
        * dat moet worden gelezen wordt opgezocht        */
  j=sizeof(struct indata);
  fin=fopen(p->filin,"rb");
  x=-100;
  do
  { x+=100;
    fseek(fin,x*j,SEEK_SET);
    fread(&i,j,1,fin);
  } while(i.dag<p->bdag);
  y=p->btijd-i.tijd;
  fseek(fin,(x+y)*j,SEEK_SET);

       /* de uitvoerfile wordt geopend en de             *
        * schrijfpointer wordt op de juiste plaats gezet *
        * afhankelijk van de waarde van uit              */
  k=sizeof(struct uitdata);
  for(teller=1;teller<5;teller++)
  { if (teller==1) uitv=p->uit1;
    if (teller==2) uitv=p->uit2;
    if (teller==3) uitv=p->uit3;
    if (teller==4) uitv=p->uit4;
    switch(uit[teller-1])
    { case 0 :
        fuit[teller-1]=fopen(uitv,"w+b");
        break;
      case 1 :
        fuit[teller-1]=fopen(uitv,"ab");
        break;
      case 2 :
        fuit[teller-1]=fopen(uitv,"r+b");
        fseek(fuit[teller-1],0L,SEEK_SET);
        fread(&u,k,1,fuit[teller-1]);
```

```
        if(p->bdag<u.dagnr||(p->bdag==u.dagnr&&p->btijd<u.tijdblok))
        { fseek(fuit[teller-1],0L,SEEK_SET);
        }
        else
        { s=0;
          while (p->bdag>u.dagnr)
          { s+=100;
            fseek(fuit[teller-1],s*k,SEEK_SET);
            fread(&u,k,1,fuit[teller-1]);
          }
          t=p->btijd-u.tijdblok;
          fseek(fuit[teller-1],(s+t)*k,SEEK_SET);
        }
    }
  }

  clrscr();
  schrijf(35,12,"BEZIG");
  do
  { fread(&i,j,1,fin);
    hp=p->ca*i.mw1+p->cb;
    kp=p->xa*i.mw2+p->xb;
    phase=p->fa*i.mw6+p->fb;
    u1.uitw=10*log10(hp/kp);
    u2.uitw=cos(phase)*(hp/kp);
    u3.uitw=sin(phase)*(hp/kp);
    u4.uitw=phase;

    u1.dagnr=u2.dagnr=u3.dagnr=u4.dagnr=i.dag;
    u1.tijdblok=u2.tijdblok=u3.tijdblok=u4.tijdblok=i.tijd;
    u1.jaartal=u2.jaartal=u3.jaartal=u4.jaartal=i.jaar;

    fwrite(&u1,k,1,fuit[0]);
    fwrite(&u2,k,1,fuit[1]);
    fwrite(&u3,k,1,fuit[2]);
    fwrite(&u4,k,1,fuit[3]);
  } while(u1.dagnr<p->edag||u1.tijdblok<p->etijd);
  fclose(fin);
  fclose(fuit[0]);
  fclose(fuit[1]);
  fclose(fuit[2]);
  fclose(fuit[3]);
}
```

Appendix I

Template door interpolatie

```
# include "c:\users\hans\eigen.h"

       /* een structure wordt gedefinieerd waarin alle   *
        * parameters worden opgeborgen die binnen        *
        * temp_interpol worden gebruikt                  *
        * de structure wordt globaal gedefinieerd zodat  *
        * alle parameters in een keer kunnen worden      *
        * ingelezen en voor iedere functie binnen        *
        * temp_interpol bereikbaar zijn                  */
static struct parm { char  inv[15];        /* naam invoerfile  */
                     int      bdag;        /* begin dagnr.     */
                     int     btijd;        /* begin tijdblok   */
                     int      edag;        /* eind dagnr.      */
                     int     etijd;        /* eind tijdblok    */
                     char    s1[6];        /* begintijdstring  */
                     char    s2[6];        /* eindtijdstring   */
                   } par;
```

```
void temp_interpol(void)
{ int    parameters_temp_interpol(struct parm *p),
         checkfile_temp_interpol(struct parm *p);
  void   rechtelijn(struct parm *p);
  int    terug,check;

      /* temp_interpol maakt een lineaire interpolatie   *
       * tussen de waarden bij het begin en eindpunt     */

  do
  {   /* de parameters worden uit de parameterfile gelezen    *
       * de gebruiker kan ze wijzigen totdat hij tevreden is  *
       * dan geeft de functie de waarde nul aan terug         *
       * zet de parameters in de structure en schrijft ze     *
       * in de parameterfile                                  *
       * of hij kan teruggaan naar het keuzemenu dan geeft    *
       * de functie de waarde 1 aan terug                     */
    terug=parameters_temp_interpol(&par);

      /* als terug=1 wordt de functie verlaten en men    *
       * komt vanzelf weer in het keuzemenu             */
    if (terug==1) return;

      /* het bestaan van de opgegeven files wordt gecheckt    *
       * is alles in orde dan geeft de functie 1 terug en     *
       * kan het programma verder gaan                        *
       * is er iets niet in orde dan geeft de functie 0 terug *
       * en men keert terug naar parameters zodat men de      *
       * parameters kan wijzigen of vandaar terugkeren        *
       * naar het keuzemenu                                   */
    check=checkfile_temp_interpol(&par);
  } while (check!=1);

      /* blok voor blok worden de waarden tussen begin-  *
       * eindpunt berekend door interpolatie en het      *
       * resultaat in de file gezet                      */
  rechtelijn(&par);

      /* gereedmelding wordt gegeven                     */
  clrscr();
  schrijf(15,12,"Interpolatieberekening beeindigd");
  schrijf(15,25,"Druk op een toets");
  getch();
}
```

```
int parameters_temp_interpol(struct parm *p)
{ long reclen;
  FILE *fparm;
  int i,x,y,key,test=0,ch;

      /* parameterfile wordt geopend                          */
  reclen=sizeof(struct parm);
  fparm=fopen("Interpol.par","r+b");

      /* als de file niet bestaat wordt hij gemaakt           *
       * en gevuld met nullen                                 */
  if (fparm==NULL)
  { fparm=fopen("Interpol.par","w+b");
    for (i=0;i<reclen;i++)
    { putc('\0',fparm);
    }
  }

      /* lees file en zet de data in de structure             */
  fseek(fparm,0L,SEEK_SET);
  fread(p,reclen,1,fparm);

      /* de parameternamen worden op het scherm gezet         */
  clrscr();
  printf("\n\tParameterlijst\n\n"
         "\t           invoerfilenaam :\n"
         "\t          begin dagnummer :\n"
         "\t               begin tijd :\n"
         "\t           eind dagnummer :\n"
         "\t                eind tijd :\n"
         "\t                                Tevreden\n"
         "\t                                Terug naar keuzemenu");

      /* de parameterwaarden worden op het scherm gezet       */
  x=39;y=4;
  window(x,y,80,10);
  cprintf("%.14s\r\n"
          "%d\r\n"
          "%.5s\r\n"
          "%d\r\n"
          "%.5s"
          ,p->inv,p->bdag,p->s1,p->edag,p->s2);
```

```
    /* met de up en down toetsen kan de cursor op de          *
     * parameterwaarden worden gezet en kunnen deze worden    *
     * veranderd                                              *
     * enter op tevreden zet de configuratie in de par-file   *
     * en geeft 0 terug                                       *
     * enter op terug naar keuzemenu geeft 1 terug zodat      *
     * teruggegaan wordt naar het keuzemenu                   */
y=9;
do
{ window(1,1,80,25);
  gotoxy(x,y);
  while(bioskey(1)==0);
  key=bioskey(0);
  switch(key)
  { case 0x5000 :
      y=y+1;
      if (y>10) y=4;
      break;
    case 0x4800 :
      y=y-1;
      if (y<4) y=10;
      break;
    case 0x1c0d :
      if (y==9) test=1;
      if (y==10)
      { fclose(fparm);
        return(1);
      }
      break;
    default :
      if (isalnum(key&0xFF)&&y<9)
      { window(x,y,x+20,y);
        clrscr();
        cprintf("%c",key);
        ungetch(key);
        switch(y)
        { case 4 :
            cscanf("%s",p->inv);
            break;
          case 5 :
            cscanf("%d",&p->bdag);
            break;
          case 6 :
            cscanf("%s",p->s1);
            break;
          case 7 :
            cscanf("%d",&p->edag);
            break;
          case 8 :
            cscanf("%s",p->s2);
            break;
        } ch=getch();y++;
      } ch++;
  }
} while (test!=1);
```

```
      /* de parameters worden wegezet in de parameterfile */
  p->btijd=tijdblok(p->s1);
  p->etijd=tijdblok(p->s2);
  fseek(fparm,0L,SEEK_SET);
  fwrite(p,reclen,1,fparm);
  fclose(fparm);
  return(0);
}



int checkfile_temp_interpol(struct parm *p)
{ struct indata { int dag;
                  int tijd;
                  int jaar;
                  double mw;
                } i1,i2;
  FILE *fin;
  int  ch;
  long j,k,lengte,aantal;
  char t1[6],t2[6];

       /* de invoerfile wordt geopend                    *
        * als de file niet bestaat wordt een melding     *
        * gegeven op het scherm en wordt teruggekeerd    *
        * naar de parameterlijst via het teruggeven      *
        * van 0 aan check                                */
  fin=fopen(p->inv,"rb");
  if (fin==NULL)
  { clrscr();
    gotoxy(1,5);
    printf("\t\tDe invoerfile %s bestaat niet\n"
           "\t\tU keert terug naar de parameterlijst",p->inv);
    schrijf(10,25,"Druk op een toets");
    getch();
    return(0);
  }

       /* begin- en eindpunt van de file wordt opgezocht */
  j=sizeof(struct indata);
  fseek(fin,0L,SEEK_SET);
  fread(&i1,j,1,fin);
  fseek(fin,0L,SEEK_END);
  lengte=ftell(fin);
  aantal=lengte/j;
  do
  { fseek(fin,(aantal-1)*j,SEEK_SET);
    fread(&i2,j,1,fin);
    aantal--;
  } while (i2.dag==0);
```

```
      /* gekeken wordt of de opgegeven periode binnen   *
       * de file valt                                   *
       * zoniet dan wordt teruggekeerd naar de          *
       * parameterlijst                                 */
  if ((p->bdag<i1.dag||(p->bdag==i1.dag&&p->btijd<i1.tijd))||
      (p->edag>i2.dag||(p->edag==i2.dag&&p->etijd>i2.tijd)))
  { clrscr();
    tijd(i1.tijd,t1);
    tijd2(i2.tijd,t2);
    gotoxy(1,5);
    printf("\t\tInvoerfile %s\n"
         "\t\t  begin   dagnr : %d\n"
         "\t\t           tijd : %s\n"
         "\t\t  einde   dagnr : %d\n"
         "\t\t           tijd : %s"
         ,p->inv,i1.dag,t1,i2.dag,t2);
    printf("\n\n\t\tde opgegeven periode valt niet binnen %s\n"
           "\t\tU keert terug naar de parameterlijst!",p->inv);
    schrijf(10,25,"Druk op een toets");
    getch();
    fclose(fin);
    return(0);
  }


      /* valt de periode wel binnen de file             */
  fclose(fin);
  return(1);
}



void rechtelijn(struct parm *p)
{ struct indata { int dag;
                  int tijd;
                  int jaar;
                  double mw;
                } i1;
  FILE   *fin;
  int    t1,t2,a;
  long   j,x,y;
  double mw1,mw2,c;
```

```
      /* de invoerfile wordt geopend en van het begin_  *
       * en eindpunt worden tijd en meetwaarde gelezen  *
       * om de helling van de rechte te bepalen         */
  j=sizeof(struct indata);
  fin=fopen(p->inv,"r+b");
  x=-100;
  do
  { x+=100;
    fseek(fin,x*j,SEEK_SET);
    fread(&i1,j,1,fin);
  } while(i1.dag<p->edag);
  y=p->etijd-i1.tijd;
  fseek(fin,(x+y)*j,SEEK_SET);
  fread(&i1,j,1,fin);
  t2=i1.tijd;
  mw2=i1.mw;
  x=-100;
  do
  { x+=100;
    fseek(fin,x*j,SEEK_SET);
    fread(&i1,j,1,fin);
  } while(i1.dag<p->bdag);
  y=p->btijd-i1.tijd;
  fseek(fin,(x+y)*j,SEEK_SET);
  fread(&i1,j,1,fin);
  t1=i1.tijd;
  mw1=i1.mw;

      /* de helling wordt bepaald                       */
  c=(mw2-mw1)/(double)(t2-t1);

  a=1;
  clrscr();
  schrijf(35,12,"BEZIG");
  do
  { fseek(fin,(x+y+a)*j,SEEK_SET);
    fread(&i1,j,1,fin);

    i1.mw=mw1+c*(i1.tijd-t1);

    fseek(fin,(x+y+a)*j,SEEK_SET);
    fwrite(&i1,j,1,fin);
    a++;
  } while(i1.dag<p->edag||i1.tijd<p->etijd);
  fclose(fin);
}
```

Appendix J

Gecorrigeerde XPD en Fase waarden

```
# include "c:\users\hans\eigen.h"


       /* een structure wordt gedefinieerd waarin alle   *
        * parameters worden opgeborgen die binnen        *
        * corr_xpd worden gebruikt                       *
        * de structure wordt globaal gedefinieerd zodat  *
        * alle parameters in een keer kunnen worden      *
        * ingelezen en voor iedere functie binnen        *
        * corr_xpd bereikbaar zijn                       */

static struct parm { char inv1[15];        /* naam invoerfile1  */
                     char inv2[15];        /* naam invoerfile2  */
                     char test[15];        /* naam dempfile     */
                     char uit1[15];        /* naam uitvoerfile1 */
                     char uit2[15];        /* naam uitvoerfile2 */
                     int      bdag;        /* begin dagnr.      */
                     int     btijd;        /* begin tijdblok    */
                     int      edag;        /* eind dagnr.       */
                     int     etijd;        /* eind tijdblok     */
                     char    s1[6];        /* begintijdstring   */
                     char    s2[6];        /* eindtijdstring    */
                   } par;
```

```
void corr_xpd(void)
{ int    parameters_corr_xpd(struct parm *p),
         checkfile_corr_xpd(struct parm *p, int *uit);
  void   reconstrueer(struct parm *p,int *uit);
  int    terug,check,uitvlag[2];

      /* corr_xpd berekent de gecorrigeerde waarden voor *
       * de XPD en Fase uit de gecorrigeerde XPD-I en -Q *
       * waarden                                        */

  do
  {   /* de parameters worden uit de parameterfile gelezen    *
       * de gebruiker kan ze wijzigen totdat hij tevreden is  *
       * dan geeft de functie de waarde nul aan terug         *
       * zet de parameters in de structure en schrijft ze     *
       * in de parameterfile                                  *
       * of hij kan teruggaan naar het hoofdmenu dan geeft    *
       * de functie de waarde 1 aan terug                     */
    terug=parameters_corr_xpd(&par);

      /* als terug=1 wordt de functie verlaten en men     *
       * komt vanzelf weer in het hoofdmenu              */
    if (terug==1) return;

      /* het bestaan van de opgegeven files wordt gecheckt    *
       * is alles in orde dan geeft de functie 1 terug en     *
       * kan het programma verder gaan                        *
       * is er iets niet in orde dan geeft de functie 0 terug *
       * en men keert terug naar parameters zodat men de      *
       * parameters kan wijzigen of vandaar terugkeren        *
       * naar het hoofdmenu                                   */
    check=checkfile_corr_xpd(&par,uitvlag);
  } while (check!=1);

      /* blok voor blok wordt uit de waarden van de      *
       * invoer files de waarde van de XPD berekend en   *
       * het resultaat in de uitvoerfile gezet           */
  reconstrueer(&par,uitvlag);

      /* gereedmelding wordt gegeven                     */
  clrscr();
  schrijf(15,12,"Gecorrigeerde XPD en Fase berekend!");
  schrijf(15,25,"Druk op een toets");
  getch();
}
```

```
int parameters_corr_xpd(struct parm *p)
{ long reclen;
  FILE *fparm;
  int i,x,y,key,test=0,ch;

      /* parameterfile wordt geopend                          */
  reclen=sizeof(struct parm);
  fparm=fopen("Corr_xpd.par","r+b");

      /* als de file niet bestaat wordt hij gemaakt           *
       * en gevuld met nullen                                 */
  if (fparm==NULL)
  { fparm=fopen("Corr_xpd.par","w+b");
    for (i=0;i<reclen;i++)
    { putc('\0',fparm);
    }
  }

      /* lees file en zet de data in de structure             */
  fseek(fparm,0L,SEEK_SET);
  fread(p,reclen,1,fparm);

      /* de parameternamen worden op het scherm gezet         */
  clrscr();
  printf("\n\tParameterlijst\n\n"
         "\t     invoerfilenaam XPD-I :\n"
         "\t     invoerfilenaam XPD-Q :\n"
         "\t          dempingfilenaam :\n"
         "\t      uitvoerfilenaam XPD :\n"
         "\t     uitvoerfilenaam Fase :\n"
         "\t          begin dagnummer :\n"
         "\t               begin tijd :\n"
         "\t            eind dagnummer :\n"
         "\t                 eind tijd :\n"
         "\t                                Tevreden\n"
         "\t                                Terug naar hoofdmenu");

      /* de parameterwaarden worden op het scherm gezet       */
  x=39;y=4;
  window(x,y,80,12);
  cprintf("%.14s\r\n"
          "%.14s\r\n"
          "%.14s\r\n"
          "%.14s\r\n"
          "%.14s\r\n"
          "%d\r\n"
          "%.5s\r\n"
          "%d\r\n"
          "%.5s"
          ,p->inv1,p->inv2,p->test,p->uit1,p->uit2,p->bdag
          ,p->s1,p->edag,p->s2);
```

```
    /* met de up en down toetsen kan de cursor op de          *
     * parameterwaarden worden gezet en kunnen deze worden    *
     * veranderd                                              *
     * enter op tevreden zet de configuratie in de par-file   *
     * en geeft 0 terug                                       *
     * enter op terug naar hoofdmenu geeft 1 terug zodat      *
     * teruggegaan wordt naar het hoofdmenu                   */
y=13;
do
{ window(1,1,80,25);
  gotoxy(x,y);
  while(bioskey(1)==0);
  key=bioskey(0);
  switch(key)
  { case 0x5000 :
      y=y+1;
      if (y>14) y=4;
      break;
    case 0x4800 :
      y=y-1;
      if (y<4) y=14;
      break;
    case 0x1c0d :
      if (y==13) test=1;
      if (y==14)
      { fclose(fparm);
        return(1);
      }
      break;
    default :
      if (isalnum(key&0xFF)&&y<13)
      { window(x,y,x+20,y);
        clrscr();
        cprintf("%c",key);
        ungetch(key);
        switch(y)
        { case 4 :
            cscanf("%s",p->inv1);
            break;
          case 5 :
            cscanf("%s",p->inv2);
            break;
          case 6 :
            cscanf("%s",p->test);
            break;
          case 7 :
            cscanf("%s",p->uit1);
            break;
          case 8 :
            cscanf("%s",p->uit2);
            break;
          case 9 :
            cscanf("%d",&p->bdag);
            break;
          case 10 :
            cscanf("%s",p->s1);
            break;
          case 11 :
```

```
            cscanf("%d",&p->edag);
            break;
          case 12 :
            cscanf("%s",p->s2);
            break;
        } ch=getch();y++;
      } ch++;
    }
  } while (test!=1);

     /* de parameters worden wegezet in de parameterfile */
  p->btijd=tijdblok(p->s1);
  p->etijd=tijdblok(p->s2);
  fseek(fparm,0L,SEEK_SET);
  fwrite(p,reclen,1,fparm);
  fclose(fparm);
  return(0);
}


int checkfile_corr_xpd(struct parm *p, int *uit)
{ struct indata { int dag;
                  int tijd;
                  int jaar;
                  double meetwaarde;
                } i1,i2;
  struct uitdata { int dagnr;
                   int tijdblok;
                   int jaartal;
                   double mw;
                 } u1,u2;
  FILE *fin1,*fin2,*ftest,*fuit;
  int  ch,i;
  long j,k,lengte,aantal;
  char t1[6],t2[6],*uitv;

      /* de invoerfiles worden geopend                    *
       * als een file niet bestaat wordt een melding      *
       * gegeven op het scherm en wordt teruggekeerd      *
       * naar de parameterlijst via het teruggeven        *
       * van 0 aan check                                  */
  fin1=fopen(p->inv1,"rb");
  if (fin1==NULL)
  { clrscr();
    gotoxy(1,5);
    printf("\t\tDe invoerfile %s bestaat niet\n"
           "\t\tU keert terug naar de parameterlijst",p->inv1);
    schrijf(10,25,"Druk op een toets");
    getch();
    return(0);
  }
```

```
fin2=fopen(p->inv2,"rb");
if (fin2==NULL)
{ clrscr();
  gotoxy(1,5);
  printf("\t\tDe invoerfile %s bestaat niet\n"
         "\t\tU keert terug naar de parameterlijst",p->inv2);
  schrijf(10,25,"Druk op een toets");
  getch();
  fclose(fin1);
  return(0);
}


     /* begin- en eindpunten van de files worden       *
      * opgezocht                                      */
j=sizeof(struct indata);
fseek(fin1,0L,SEEK_SET);
fread(&i1,j,1,fin1);
fseek(fin1,0L,SEEK_END);
lengte=ftell(fin1);
aantal=lengte/j;
do
{ fseek(fin1,(aantal-1)*j,SEEK_SET);
  fread(&i2,j,1,fin1);
  aantal--;
} while (i2.dag==0);

     /* gekeken wordt of de opgegeven periode binnen   *
      * de file valt                                   *
      * zoniet dan wordt teruggekeerd naar de          *
      * parameterlijst                                 */
if ((p->bdag<i1.dag||(p->bdag==i1.dag&&p->btijd<i1.tijd))||
    (p->edag>i2.dag||(p->edag==i2.dag&&p->etijd>i2.tijd)))
{ clrscr();
  tijd(i1.tijd,t1);
  tijd2(i2.tijd,t2);
  gotoxy(1,5);
  printf("\t\tInvoerfile %s\n"
       "\t\t  begin   dagnr : %d\n"
       "\t\t           tijd : %s\n"
       "\t\t  einde   dagnr : %d\n"
       "\t\t           tijd : %s"
       ,p->inv1,i1.dag,t1,i2.dag,t2);
  printf("\n\n\t\tde opgegeven periode valt niet binnen %s\n"
         "\t\tU keert terug naar de parameterlijst!",p->inv1);
  schrijf(10,25,"Druk op een toets");
  getch();
  fclose(fin1);
  fclose(fin2);
  return(0);
}

fseek(fin2,0L,SEEK_SET);
fread(&i1,j,1,fin2);
fseek(fin2,0L,SEEK_END);
lengte=ftell(fin2);
aantal=lengte/j;
```

```
do
{ fseek(fin2,(aantal-1)*j,SEEK_SET);
  fread(&i2,j,1,fin2);
  aantal--;
} while (i2.dag==0);

     /* gekeken wordt of de opgegeven periode binnen   *
      * de file valt                                   *
      * zoniet dan wordt teruggekeerd naar de          *
      * parameterlijst                                 */
if ((p->bdag<i1.dag||(p->bdag==i1.dag&&p->btijd<i1.tijd))||
    (p->edag>i2.dag||(p->edag==i2.dag&&p->etijd>i2.tijd)))
{ clrscr();
  tijd(i1.tijd,t1);
  tijd2(i2.tijd,t2);
  gotoxy(1,5);
  printf("\t\tInvoerfile %s\n"
       "\t\t  begin   dagnr : %d\n"
       "\t\t           tijd : %s\n"
       "\t\t  einde   dagnr : %d\n"
       "\t\t           tijd : %s"
       ,p->inv2,i1.dag,t1,i2.dag,t2);
  printf("\n\n\t\tde opgegeven periode valt niet binnen %s\n"
         "\t\tU keert terug naar de parameterlijst!",p->inv2);
  schrijf(10,25,"Druk op een toets");
  getch();
  fclose(fin1);
  fclose(fin2);
  return(0);
}

     /* valt de periode wel binnen de file             */
fclose(fin1);
fclose(fin2);

     /* gekeken wordt of de testfile bestaat           *
      * als de file niet bestaat wordt een melding op  *
      * het scherm gegeven en wordt teruggekeerd naar  *
      * de parameterlijst                              */
ftest=fopen(p->test,"rb");
if (ftest==NULL)
{ clrscr();
  gotoxy(1,5);
  printf("\t\tDe dempingfile %s bestaat niet\n"
         "\t\tU keert terug naar de parameterlijst",p->test);
  schrijf(10,25,"Druk op een toets");
  getch();
  return(0);
}

      /* begin en eindpunt van de file wordt opgezocht */
fseek(ftest,0L,SEEK_SET);
fread(&i1,j,1,ftest);
fseek(ftest,0L,SEEK_END);
lengte=ftell(ftest);
aantal=lengte/j;
```

```
do
{ fseek(ftest,(aantal-1)*j,SEEK_SET);
  fread(&i2,j,1,ftest);
  aantal--;
} while (i2.dag==0);

     /* gekeken wordt of de opgegeven periode binnen   *
      * de file valt                                   *
      * zoniet dan wordt teruggekeerd naar de          *
      * parameterlijst                                 */
if ((p->bdag<i1.dag||(p->bdag==i1.dag&&p->btijd<i1.tijd))||
    (p->edag>i2.dag||(p->edag==i2.dag&&p->etijd>i2.tijd)))
{ clrscr();
  tijd(i1.tijd,t1);
  tijd2(i2.tijd,t2);
  gotoxy(1,5);
  printf("\t\tDempingfile %s\n"
       "\t\t  begin   dagnr : %d\n"
       "\t\t           tijd : %s\n"
       "\t\t  einde   dagnr : %d\n"
       "\t\t           tijd : %s"
       ,p->test,i1.dag,t1,i2.dag,t2);
  printf("\n\n\t\tde opgegeven periode valt niet binnen %s\n"
         "\t\tU keert terug naar de parameterlijst!",p->test);
  schrijf(10,25,"Druk op een toets");
  getch();
  fclose(ftest);
  return(0);
}

     /* valt de periode wel binnen de file             */
fclose(ftest);

     /* de uitvoerfiles worden geopend                 *
      * bestaat een file al dan wordt gekeken of moet  *
      * worden overschreven of toegevoegd en de vlag   *
      * uit wordt gezet: uit=0 file bestaat niet,      *
      * uit=1 toevoegen, uit=2 overschrijven           */
for (i=1;i<3;i++)
{ if (i==1) uitv=p->uit1;
  if (i==2) uitv=p->uit2;
  fuit=fopen(uitv,"rb");

     /* de file bestaat nog niet                       *
      * openen kan geen kwaad er wordt teruggekeerd en *
      * check krijgt de waarde 1, zodat het programma  *
      * verder kan gaan                                */
  if (fuit==NULL) uit[i-1]=0;

  else
  {
     /* bestaat de file wel dan worden begin en eindpunt *
      * opgezocht en op het scherm gezet                 */
    k=sizeof(struct uitdata);
    fseek(fuit,0L,SEEK_SET);
    fread(&u1,k,1,fuit);
    fseek(fuit,0L,SEEK_END);
    lengte=ftell(fuit);
```

```
      aantal=lengte/k;
      fseek(fuit,(aantal-1)*k,SEEK_SET);
      fread(&u2,k,1,fuit);
      tijd(u1.tijdblok,t1);
      tijd2(u2.tijdblok,t2);
      clrscr();
      gotoxy(1,5);
      printf("\t\tUitvoerfile %s\n"
             "\t\t  begin   dagnr : %d\n"
             "\t\t        tijdblok : %d\n"
             "\t\t  einde   dagnr : %d\n"
             "\t\t        tijdblok : %d\n"
             ,uitv,u1.dagnr,t1,u2.dagnr,t2);
      fclose(fuit);

       /* ligt het beginpunt voor het einde van de file  *
        * dan wordt gevraagd of men wil overschrijven     *
        * uit=2 zoniet dan wordt gevraagd of men wil      *
        * toevoegen uit=1 wil men niet overschrijven of   *
        * toevoegen dan keert men terug naar de           *
        * parameterlijst                                  */
      if(p->bdag>u2.dagnr||(p->bdag==u2.dagnr&&p->btijd>u2.tijdblok))
      { printf("\n\n\t\t");
        printf("Wilt U toevoegen aan het eind van de file? (J/N)");
        ch=getch();
        if (ch=='J'||ch=='j') uit[i-1]=1;
        else return(0);
      }
      else
      { printf("\n\n\t\t\tWilt U de file overschrijven? (J/N)");
        ch=getch();
        if (ch=='J'||ch=='j') uit[i-1]=2;
        else return(0);
      }
    }
  }
  return(1);
}



void reconstrueer(struct parm *p,int *uit)
{ struct indata { int dag;
                  int tijd;
                  int jaar;
                  double mw;
                } i1,i2,i3;
  struct uitdata { int dagnr;
                   int tijdblok;
                   int jaartal;
                   double uitw;
                 } u,u1,u2;
  FILE   *fin1,*fin2,*ftest,*fuit[2];
  long   j,k,x,y,s,t;
  char   *uitv;
  int    teller;
```

```
     /* de invoerfiles worden geopend en het eerste    *
      * blok dat moet worden gelezen wordt opgezocht   */
j=sizeof(struct indata);
fin1=fopen(p->inv1,"rb");
fin2=fopen(p->inv2,"rb");
x=-100;
do
{ x+=100;
  fseek(fin1,x*j,SEEK_SET);
  fread(&i1,j,1,fin1);
} while(i1.dag<p->bdag);
y=p->btijd-i1.tijd;
fseek(fin1,(x+y)*j,SEEK_SET);

x=-100;
do
{ x+=100;
  fseek(fin2,x*j,SEEK_SET);
  fread(&i2,j,1,fin2);
} while(i2.dag<p->bdag);
y=p->btijd-i2.tijd;
fseek(fin2,(x+y)*j,SEEK_SET);

     /* de dempingfile wordt geopend en het eerste     *
      * dat moet worden gelezen wordt opgezocht        */
ftest=fopen(p->test,"rb");
x=-100;
do
{ x+=100;
  fseek(ftest,x*j,SEEK_SET);
  fread(&i3,j,1,ftest);
} while(i3.dag<p->bdag);
y=p->btijd-i3.tijd;
fseek(ftest,(x+y)*j,SEEK_SET);

     /* de uitvoerfiles worden geopend en de           *
      * schrijfpointer wordt op de juiste plaats gezet *
      * afhankelijk van de waarde van uit              */
k=sizeof(struct uitdata);
for (teller=1;teller<3;teller++)
{ if (teller==1) uitv=p->uit1;
  if (teller==2) uitv=p->uit2;
  switch(uit[teller-1])
  { case 0 :
      fuit[teller-1]=fopen(uitv,"w+b");
      break;
    case 1 :
      fuit[teller-1]=fopen(uitv,"ab");
      break;
```

```
      case 2 :
        fuit[teller-1]=fopen(uitv,"r+b");
        fseek(fuit[teller-1],0L,SEEK_SET);
        fread(&u,k,1,fuit[teller-1]);

        if (p->bdag<u.dagnr||
           (p->bdag==u.dagnr&&p->btijd<u.tijdblok))
        { fseek(fuit[teller-1],0L,SEEK_SET);
        }
        else
        { s=0;
          while (p->bdag>u.dagnr)
          { s+=100;
            fseek(fuit[teller-1],s*k,SEEK_SET);
            fread(&u,k,1,fuit[teller-1]);
          }
          t=p->btijd-u.tijdblok;
          fseek(fuit[teller-1],(s+t)*k,SEEK_SET);
        }
    }
  }

  clrscr();
  schrijf(35,12,"BEZIG");
  do
  { fread(&i1,j,1,fin1);
    fread(&i2,j,1,fin2);
    fread(&i3,j,1,ftest);

    u1.dagnr=u2.dagnr=i1.dag;
    u1.tijdblok=u2.tijdblok=i1.tijd;
    u1.jaartal=u2.jaartal=i1.jaar;

    if (i3.mw>-999)
    { u1.uitw=10*log10(sqrt(pow(i1.mw,2)+pow(i2.mw,2)));
      u2.uitw=atan2(i2.mw,i1.mw);
    }
    else
    { u1.uitw=u2.uitw=-1000;
    }

    fwrite(&u1,k,1,fuit[0]);
    fwrite(&u2,k,1,fuit[1]);

  } while(u1.dagnr<p->edag||u1.tijdblok<p->etijd);
  fclose(fin1);
  fclose(fin2);
  fclose(ftest);
  fclose(fuit[0]);
  fclose(fuit[1]);
}
```

Appendix K

Grafiek van infase versus quadratuur component van de XPD

```
# include "c:\users\hans\eigen.h"

       /* een structure wordt gedefinieerd waarin alle   *
        * parameters worden opgeborgen die binnen        *
        * grafiek kunnen worden gebruikt                 *
        * de structure wordt globaal static gedefinieerd *
        * zodat alle parameters in een keer kunnen       *
        * worden ingelezen en voor iedere functie binnen *
        * gafiek bereikbaar zijn                         */
static struct parm { char    max1[5];      /* maxwaarde grootheid1 */
                     char   inv1[15];      /* naam invoerfile1     */
                     char    max2[5];      /* maxwaarde grootheid2 */
                     char   inv2[15];      /* naam invoerfile2     */
                     int        bdag;      /* begin dagnr.         */
                     int       btijd;      /* begin tijdblok       */
                     int        edag;      /* eind dagnr.          */
                     int       etijd;      /* eind tijdblok        */
                     char      s1[6];      /* begintijdstring      */
                     char      s2[6];      /* eindtijdstring       */
                   } par;
```

```
void IvsQ_grafiek(int mode)
{ int    parameters_ivsq(struct parm *p),
         checkfile_ivsq(struct parm *p);
  void   teken_ivsq(struct parm *p,int gm);
  int    terug,check;

     /* IvsQ_grafiek kan grafisch een XPD-I file tegen  *
      * een XPD-Q file op het scherm zetten             */

  do
  {  /* de parameters worden uit de parameterfile gelezen    *
      * de gebruiker kan ze wijzigen totdat hij tevreden is  *
      * dan geeft de functie de waarde nul aan terug         *
      * zet de parameters in de structure en schrijft ze     *
      * in de parameterfile                                  *
      * of hij kan teruggaan naar het hoofdmenu dan geeft    *
      * de functie de waarde 1 aan terug                     */
    terug=parameters_ivsq(&par);

      /* als terug=1 wordt de functie verlaten en men    *
       * komt vanzelf weer in het hoofdmenu             */
    if (terug==1) return;

      /* het bestaan van de opgegeven files wordt gecheckt    *
       * is alles in orde dan geeft de functie 1 terug en     *
       * kan het programma verder gaan                        *
       * is er iets niet in orde dan geeft de functie 0 terug *
       * en men keert terug naar parameters zodat men de      *
       * parameters kan wijzigen of vandaar terugkeren        *
       * naar het hoofdmenu                                   */
    check=checkfile_ivsq(&par);
  } while (check!=1);

     /* het opgegeven aantal grafieken wordt getekend   */
  teken_ivsq(&par,mode);
}
```

```
int parameters_ivsq(struct parm *p)
{ long reclen;
  FILE *fparm;
  int i,x,y,key,test,ch;

     /* parameterfile wordt geopend                          */
  reclen=sizeof(struct parm);
  fparm=fopen("ivsq.par","r+b");

     /* als de file niet bestaat wordt hij gemaakt           *
      * en gevuld met nullen                                 */
  if (fparm==NULL)
  { fparm=fopen("ivsq.par","w+b");
    for (i=0;i<reclen;i++)
    { putc('\0',fparm);
    }
  }

     /* lees file en zet de data in de structure             */
  fseek(fparm,0L,SEEK_SET);
  fread(p,reclen,1,fparm);

     /* de parameternamen worden op het scherm gezet         */
  clrscr();
  printf("\n\tParameterlijst\n\n"
         "\t             maximum XPD-I :\n"
         "\t            filenaam XPD-I :\n"
         "\t             maximum XPD-Q :\n"
         "\t            filenaam XPD-Q :\n"
         "\t           begin dagnummer :\n"
         "\t               begin tijd :\n"
         "\t            eind dagnummer :\n"
         "\t                 eind tijd :\n"
         "\t                                Tevreden\n"
         "\t                                Terug naar hoofdmenu");

     /* de parameterwaarden worden op het scherm gezet       */
  x=39;y=4;
  window(x,y,80,11);
  cprintf("%.4s\r\n"
          "%.14s\r\n"
          "%.4s\r\n"
          "%.14s\r\n"
          "%d\r\n"
          "%s\r\n"
          "%d\r\n"
          "%s"
          ,p->max1,p->inv1,p->max2,p->inv2
          ,p->bdag,p->s1,p->edag,p->s2);
```

```
    /* met de up en down toetsen kan de cursor op de          *
     * parameterwaarden worden gezet en kunnen deze worden    *
     * veranderd                                              *
     * enter op tevreden zet de configuratie in de par-file   *
     * en geeft 0 terug                                       *
     * enter op terug naar hoofdmenu geeft 1 terug zodat      *
     * teruggegaan wordt naar het hoofdmenu                   */
test=0;
y=12;
do
{ window(1,1,80,25);
  gotoxy(x,y);
  while(bioskey(1)==0);
  key=bioskey(0);
  switch(key)
  { case 0x5000 :
      y=y+1;
      if (y>13) y=4;
      break;
    case 0x4800 :
      y=y-1;
      if (y<4) y=13;
      break;
    case 0x1c0d :
      if (y==12) test=1;
      if (y==13)
      { fclose(fparm);
        return(1);
      }
      break;
    default :
      if (isalnum(key&0xFF)&&(y<12))
      { window(x,y,x+20,y);
        clrscr();
        cprintf("%c",key);
        ungetch(key);
        switch(y)
        { case 4 :
            cscanf("%s",p->max1);
            break;
          case 5 :
            cscanf("%s",p->inv1);
            break;
          case 6 :
            cscanf("%s",p->max2);
            break;
          case 7 :
            cscanf("%s",p->inv2);
            break;
          case 8 :
            cscanf("%d",&p->bdag);
            break;
          case 9 :
            cscanf("%s",p->s1);
            break;
          case 10 :
            cscanf("%d",&p->edag);
            break;
```

```
            case 11 :
              cscanf("%s",p->s2);
              break;
          } ch=getch();y++;
        } ch++;
    }
  } while (test!=1);

     /* de parameters worden wegezet in de parameterfile */
  caps(p->inv1);
  caps(p->inv2);
  p->btijd=tijdblok(p->s1);
  p->etijd=tijdblok(p->s2);
  fseek(fparm,0L,SEEK_SET);
  fwrite(p,reclen,1,fparm);
  fclose(fparm);
  return(0);
}



int checkfile_ivsq(struct parm *p)
{ struct ineen  { int dag;
                  int tijd;
                  int jaar;
        .         double meetwaarde;
                } ie1,ie2;
  FILE  *fin1,*fin2;
  int   ch;
  long  j,k,lengte,aantal;


      /* de invoerfiles worden geopend als een van de files *
       * niet bestaat wordt een melding gegeven op het      *
       * scherm en wordt teruggekeerd naar de               *
       * parameterlijst via het teruggeven van 0 aan check  */
  fin1=fopen(p->inv1,"rb");
  if (fin1==NULL)
  { clrscr();
    gotoxy(1,5);
    printf("\t\tDe invoerfile %s bestaat niet\n"
           "\t\tU keert terug naar de parameterlijst",p->inv1);
    schrijf(10,25,"Druk op een toets");
    getch();
    return(0);
  }
  fin2=fopen(p->inv2,"rb");
  if (fin2==NULL)
  { clrscr();
    gotoxy(1,5);
    printf("\t\tDe invoerfile %s bestaat niet\n"
           "\t\tU keert terug naar de parameterlijst",p->inv2);
    schrijf(10,25,"Druk op een toets");
    getch();
    fclose(fin1);
    return(0);
  }
```

```
     /* gekeken wordt of filenaam1 bij XPD-I hoort     */
if (strncmpi(p->inv1,"XPDI",4)!=0)
{ schrijf(15,5,"Invoerfile1 is geen XPD-I-file!");
  schrijf(15,7,"U keert terug naar de parameterfile");
  schrijf(15,25,"Druk op een toets");
  getch();
  fclose(fin1);
  fclose(fin2);
  return(0);
}

     /* begin en eindpunt van de file worden gezocht   */
j=sizeof(struct ineen);
fseek(fin1,0L,SEEK_SET);
fread(&ie1,j,1,fin1);
fseek(fin1,0L,SEEK_END);
lengte=ftell(fin1);
aantal=lengte/j;
do
{ fseek(fin1,(aantal-1)*j,SEEK_SET);
  fread(&ie2,j,1,fin1);
  aantal--;
} while (ie2.dag==0);
clrscr();

     /* gekeken wordt of de opgegeven periode binnen   *
      * de file valt                                   */
if ((p->bdag<ie1.dag||(p->bdag==ie1.dag&&p->btijd<ie1.tijd))||
   (p->edag>ie2.dag||(p->edag==ie2.dag&&p->etijd>ie2.tijd)))
{ printf("\n\n\t\tde opgegeven periode valt niet binnen %s\n"
         "\t\tU keert terug naar de parameterlijst!",p->inv1);
  schrijf(15,25,"Druk op een toets");
  getch();
  fclose(fin1);
  fclose(fin2);
  return(0);
}


     /* gekeken wordt of filenaam2 bij XPD-Q hoort     */
if (strncmpi(p->inv2,"XPDQ",4)!=0)
{ schrijf(15,5,"Invoerfile2 is geen XPD-Q-file!");
  schrijf(15,7,"U keert terug naar de parameterfile");
  schrijf(15,25,"Druk op een toets");
  getch();
  fclose(fin2);
  fclose(fin1);
  return(0);
}
```

```
      /* begin en eindpunt van de file worden gezocht   */
  j=sizeof(struct ineen);
  fseek(fin2,0L,SEEK_SET);
  fread(&ie1,j,1,fin2);
  fseek(fin2,0L,SEEK_END);
  lengte=ftell(fin2);
  aantal=lengte/j;
  do
  { fseek(fin2,(aantal-1)*j,SEEK_SET);
    fread(&ie2,j,1,fin2);
    aantal--;
  } while (ie2.dag==0);
  clrscr();

      /* gekeken wordt of de opgegeven periode binnen   *
       * de file valt                                   */
  if ((p->bdag<ie1.dag||(p->bdag==ie1.dag&&p->btijd<ie1.tijd))||
     (p->edag>ie2.dag||(p->edag==ie2.dag&&p->etijd>ie2.tijd)))
  { printf("\n\n\t\tde opgegeven periode valt niet binnen %s\n"
           "\t\tU keert terug naar de parameterlijst!",p->inv2);
    schrijf(15,25,"Druk op een toets");
    getch();
    fclose(fin2);
    fclose(fin1);
    return(0);
  }
  fclose(fin1);
  fclose(fin2);
  return(1);
}



void teken_ivsq(struct parm *p,int gm)
{ struct indata1 { int dag;
                   int tijd;
                   int jaar;
                   double mw;
                 } in1,in2;
  FILE     *fin1,*fin2;
  long     j,x,y;
  char     duur[20],dag[17];
  int      maxx,maxy,xn,yn,max1,max2,xmax,minx;
  double   dBppx,dBppy;

  setgraphmode(gm);

      /* het assenkruis wordt getekend                  *
       * en van tekst voorzien                          */

  minx=190;
  maxx=getmaxx();
  maxy=getmaxy();
  setviewport(0,0,minx,maxy,1);
  strcpy(duur,p->s1);
  strcat(duur," tot ");
  strcat(duur,p->s2);
  strcpy(dag,"dagnummer : ");
```

```
itoa(p->bdag,dag+12,10);

rectangle(0,0,minx-10,maxy);
settextjustify(RIGHT_TEXT,TOP_TEXT);
outtextxy(minx-30,50,"XPD-I vs XPD-Q");
outtextxy(minx-30,80,dag);
outtextxy(minx-30,100,duur);
outtextxy(minx-30,230,"Invoerfiles :");
outtextxy(minx-30,250,p->inv1);
outtextxy(minx-30,270,p->inv2);

setviewport(minx,0,maxx,maxy,1);
xmax=maxx-minx;
line(xmax/2,5,xmax/2,maxy-5);
line(5,maxy/2,xmax-10,maxy/2);
line(5,maxy/2-3,5,maxy/2+3);
line(xmax-10,maxy/2-3,xmax-10,maxy/2+3);
line(xmax/2-3,5,xmax/2+3,5);
line(xmax/2-3,maxy-5,xmax/2+3,maxy-5);
outtextxy(xmax/2-5,5,p->max1);
outtextxy(xmax/2-25,5,"(dB)");
outtextxy(xmax-5,maxy/2+5,p->max2);
outtextxy(xmax-5,maxy/2+15,"(dB)");


     /* de invoerfiles worden geopend en het eerste     *
      * blok dat moet worden gelezen wordt opgezocht    */
fin1=fopen(p->inv1,"rb");
fin2=fopen(p->inv2,"rb");

j=sizeof(struct indata1);
x=-100;
do
{ x+=100;
  fseek(fin1,x*j,SEEK_SET);
  fread(&in1,j,1,fin1);
} while(in1.dag<p->bdag);
y=p->btijd-in1.tijd;
fseek(fin1,(x+y)*j,SEEK_SET);

do
{ x+=100;
  fseek(fin2,x*j,SEEK_SET);
  fread(&in2,j,1,fin2);
} while(in2.dag<p->bdag);
y=p->btijd-in2.tijd;
fseek(fin2,(x+y)*j,SEEK_SET);

max1=atoi(p->max1);
max2=atoi(p->max2);
dBppx=(double)(2*max1)/(double)(xmax-15);
dBppy=(double)(2*max2)/(double)(maxy-10);
```

```
  do
  { fread(&in1,j,1,fin1);
    fread(&in2,j,1,fin2);
    xn=xmax/2+(int)(in1.mw/dBppx);
    yn=maxy/2-(int)(in2.mw/dBppy);
    moveto(xn,yn);
    putpixel(xn,yn,1);
  } while(in1.dag<p->edag||in1.tijd<p->etijd);
  fclose(fin1);
  fclose(fin2);
  getch();
  restorecrtmode();
}
```

Appendix-L

Het gebruik van de cancellation matrix voor zowel horizontaal als verticaal gepolariseerde signalen.

De matrix met meetwaarden wordt gegeven als [1]

$$\bar{\bar{U}} = \bar{\bar{A}}\ \bar{\bar{T}}\ \bar{\bar{E}}^{o}$$

Onder clear-sky condities wordt dat dan

$$\bar{\bar{U}} = \bar{\bar{A}}\ \bar{\bar{E}}^{o}$$

De cancellation matrix wordt nu gedefinieerd als

$$\bar{\bar{K}} = (\bar{\bar{U}}^{o})^{-1} = \bar{\bar{E}}^{o-1}\bar{\bar{A}}^{-1}$$

Er kan op twee manieren gecanceld worden

Left cancellation

$$\bar{\bar{T}}' = \bar{\bar{K}}\ \bar{\bar{U}} = \bar{\bar{E}}^{o-1}\bar{\bar{A}}^{-1}\bar{\bar{A}}\ \bar{\bar{T}}'\bar{\bar{E}}^{o} = \bar{\bar{E}}^{o-1}\bar{\bar{T}}'\bar{\bar{E}}^{o}$$

Right cancellation

$$\bar{\bar{T}}'' = \bar{\bar{U}}\ \bar{\bar{K}} = \bar{\bar{A}}\ \bar{\bar{T}}''\bar{\bar{E}}^{o}\bar{\bar{E}}^{o-1}\bar{\bar{A}}^{-1} = \bar{\bar{A}}\ \bar{\bar{T}}''\bar{\bar{A}}^{-1}$$

De verschillende matrices kunnen als volgt uitgeschreven worden

$$T' = \begin{bmatrix} T_{xx}' & T_{xy}' \\ T_{yx}' & T_{yy}' \end{bmatrix} \qquad T'' = \begin{bmatrix} T_{xx}'' & T_{xy}'' \\ T_{yx}'' & T_{yy}'' \end{bmatrix}$$

$$\bar{\bar{U}} = \begin{bmatrix} U_x(x) & U_x(y) \\ U_y(x) & U_y(y) \end{bmatrix} \qquad \bar{\bar{U}}^{o} = \begin{bmatrix} U_x^{o}(x) & U_x^{o}(y) \\ U_y^{o}(x) & U_y^{o}(y) \end{bmatrix}$$

$$\bar{\bar{K}} = \bar{\bar{U}}^{o-1} = \frac{1}{D}\begin{bmatrix} U_y^{o}(y) & -U_x^{o}(y) \\ -U_y^{o}(x) & U_x^{o}(x) \end{bmatrix}$$

met $D = U_x^{o}(x)U_y^{o}(y) - U_y^{o}(x)U_x^{o}(y)$

Bij left cancellation geldt nu

$$T' = \bar{\bar{K}}\ \bar{\bar{U}} = \frac{1}{D}\begin{bmatrix} U_y^{o}(y) & -U_x^{o}(y) \\ -U_y^{o}(x) & U_x^{o}(x) \end{bmatrix}\begin{bmatrix} U_x(x) & U_x(y) \\ U_y(x) & U_y(y) \end{bmatrix}$$

$$= \frac{1}{D}\begin{bmatrix} U_y^{o}(y)U_x(x) - U_x^{o}(y)U_y(x) & U_y^{o}(y)U_x(y) - U_x^{o}(y)U_y(y) \\ -U_y^{o}(x)U_x(x) + U_x^{o}(x)U_y(x) & -U_y^{o}(x)U_x(y) + U_x^{o}(x)U_y(y) \end{bmatrix}$$

Verwaarlozen van tweede orde effecten geeft

$$D \approx U_x^o(x)U_y^o(y) \quad , \quad T_{xx}' \approx \frac{U_x(x)}{U_x^o(x)} \quad , \quad T_{yy}' \approx \frac{U_y(y)}{U_y^o(y)}$$

zodat

$$T_{yx}' = \frac{-U_y^o(x)U_x(x)+U_x^o(x)U_y(x)}{U_x^o(x)U_y^o(y)} = -\left(\frac{U_y^o(x)}{U_y^o(y)}\right)\left(\frac{U_x(x)}{U_x^o(x)}\right)+\left(\frac{U_y(x)}{U_y^o(y)}\right)$$

$$= \left(\frac{U_x^o(x)}{U_y^o(y)}\right)\left(\frac{U_x(x)}{U_x^o(x)}\right)\left\{-\left(\frac{U_y^o(x)}{U_x^o(x)}\right)+\left(\frac{U_y(x)}{U_x(x)}\right)\right\}$$

en

$$\frac{T_{yx}'}{T_{xx}'} = \left(\frac{U_x^o(x)}{U_y^o(y)}\right)\left\{\left(\frac{U_y(x)}{U_x(x)}\right)-\left(\frac{U_y^o(x)}{U_x^o(x)}\right)\right\}$$

ofwel de gecorrigeerde XPD fasor is, afgezien van een balansterm, het verschil tussen de ongecorrigeerde XPD fasor en de clear-sky XPD fasor

Bij right cancellation geldt

$$T'' = \underline{\underline{U}}\,\underline{\underline{K}} = \frac{1}{D}\begin{bmatrix} U_x(x) & U_x(y) \\ U_y(x) & U_y(y) \end{bmatrix}\begin{bmatrix} U_y^o(y) & -U_x^o(y) \\ -U_y^o(x) & U_x^o(x) \end{bmatrix}$$

$$= \frac{1}{D}\begin{bmatrix} U_x(x)U_y^o(y)-U_x(y)U_y^o(x) & -U_x(x)U_x^o(y)+U_x(y)U_x^o(x) \\ U_y(x)U_y^o(y)-U_y(y)U_y^o(x) & -U_y(x)U_x^o(y)+U_y(y)U_x^o(x) \end{bmatrix}$$

Verwaarlozen van tweede orde effecten geeft

$$D \approx U_x^o(x)U_y^o(y) \quad , \quad T_{xx}'' \approx \frac{U_x(x)}{U_x^o(x)} \quad , \quad T_{yy}'' \approx \frac{U_y(y)}{U_y^o(y)}$$

zodat

$$T_{yx}'' = \frac{U_y(x)U_y^o(y)-U_y(y)U_y^o(x)}{U_x^o(x)U_y^o(y)} = \frac{U_y(x)}{U_x^o(x)} - \left(\frac{U_y(y)}{U_y^o(y)}\right)\left(\frac{U_y^o(x)}{U_x^o(x)}\right)$$

$$= \left(\frac{U_x(x)}{U_x^o(x)}\right)\left(\frac{U_y(x)}{U_x(x)}\right) - \left(\frac{U_y(y)}{U_y^o(y)}\right)\left(\frac{U_y^o(x)}{U_x^o(x)}\right)$$

en

$$\frac{T_{yx}''}{T_{xx}''} = \frac{U_y(x)}{U_x(x)} - \frac{T_{yy}''}{T_{xx}''}\left(\frac{U_y^o(x)}{U_x^o(x)}\right)$$

$$= \frac{U_y(x)}{U_x(x)} - \frac{U_y(y)U_x^o(x)}{U_x(x)U_y^o(y)}\left(\frac{U_y^o(x)}{U_x^o(x)}\right)$$

ofwel de gecorrigeerde XPD fasor wordt berekend door van de ongecorrigeerde fasor de clear-sky fasor af te trekken, nadat deze laatste is gecorrigeerd voor het verschil in versterking in de beide polarisatierichtingen.

Om op deze wijze te corrigeren is het echter wel nodig om het faseverschil tussen de twee hoofdpolarisatierichtingen te meten.