Eindhoven University of Technology

MASTER

Interaktieve optimalisering van laddernetwerken m.b.v. de Grazor-search

Derks, A.B.H.

Award date:
1977

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

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

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

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 20. Mar. 2025

2234 bse
TECHNISCHE HOGESCHOOL
EINDHOVEN
STUDIEBIBLIOTHEEK
ELEKTROTECHNIEK

AFDELING DER ELEKTROTECHNIEK
TECHNISCHE HOGESCHOOL
EINDHOVEN
Groep Automatisch Systeem Ontwerpen

INTERAKTIEVE OPTIMALISERING VAN LADDERNETWERKEN M.B.V. DE GRAZOR-SEARCH

door: A.B.H. Derks

Rapport van het afstudeerwerk
uitgevoerd van nov.76 tot okt.77
in opdracht en onder leiding van
prof.dr.ing.J.Jess

INHOUD blz

1. Inleiding.

Omdat het in dit verslag beschreven systeem zijn toepassing kan vinden in het ontwerpproces van elektrische filters, zullen we eerst op het algemene ontwerpproces van filters ingaan.
De verschillende stappen die bij het ontwerpen van een filter meestal doorlopen worden, zijn weergegeven in bijgaand stroomschema.

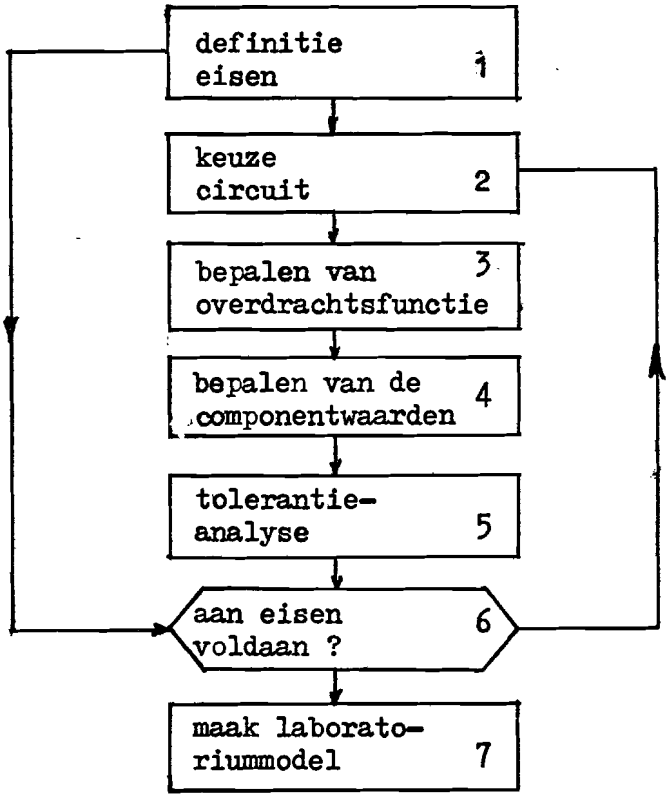


fig.1.1

Blok 1 bevat de eisen die aan het filter gesteld moeten worden, zoals het verlangde verlies- en fasegedrag maar ook andere eisen, zoals het gedrag van impedantie aan de poorten, signaalniveaux, afmetingen, kosten, elementtoleranties, enz. kunnen hier worden opgenomen.
In blok 2 wordt de keuze van het filtercircuit bepaald. Het spreekt voor zich dat hier een grote ervaring van de filterontwerper vereist is om in elk speciaal geval een verstandige keuze te doen. Wanneer eenmaal de filterklasse* is vastgesteld kan het meer gedetailleerde ontwerpproces beginnen. De eerste stap hiervan is weergegeven in blok 3. Alhoewel er voor verschillende filterklassen verschillende methoden bestaan is er in het algemeen een syntheseproces om een geschikte over-

*(bandpass, lowpass, highpass, bandstop, enz)

drachtsfunctie te contrueren of deze op zijn minst te specificeren. Er zijn vele standaardmethoden hiervoor.[1].
De stap om te komen van de geformuleerde overdrachtsfunctie tot de componentwaarden wordt gemaakt in blok 4. Wanneer het filter gedefineerd is in termen van de componentwaarden, is de volgende stap (blok 5) om het filter te analyseren. Hier wordt eerst een analyse gedaan met de in blok 4 bepaalde norminale componentwaarden om de stappen van blok 3 en blok 4 te controleren. Daarna kunnen berekeningen worden uitgevoerd met afwijkingen in de componentwaarden: bv. gevoeligheidsberekeningen zg. worst-case of random. Het een en ander wordt gedaan om te testen of het ontwerp met praktische componenten is te verwezenlijken. In dit stadium wordt het bereikte resultaat vergeleken met de oorspronkelijke specificaties, blok 6. Als niet wordt voldaan de specificaties van blok 1 moet de ontwerper de stappen 2 t/m 5 opnieuw doorlopen met andere keuze van de vrijheden die hij heeft. In bovenstaande schema is nog een blok 7 opgenomen: de praktische stap in het laboratorium.
Tegenwoordig zijn de beschikbare wiskundige modellen voor elektrische filters meestal zo goed dat het maken van een laboratoriummodel niet meer zoals vroeger, vóór het beschikbaar zijn van de digitale computer, diende om door metingen vast te stellen of het ontwerp voldeed. Inplaats daarvan wordt het laboratoriummodel meer gebruikt als een eerste prototype voor de productie. Ook het onderzoek betreffende overspraak tussen in- en uitgang van het filter, en tussen het filter en andere apperatuur, overbelasting en niet lineaire vervorming wordt meestal aan een laboratoriummodel gedaan.
In het bovenstaande schema speelt de digitale computer reeds lange tijd een belangrijke rol. Vooral in de blokken 3, 4 en 5 is veel numeriek werk waarbij de computer wordt ingeschakeld echter meestal in zg. batchmode. Programma's worden ingeleverd en een tijd later ontvangt men de uitkomsten van een vaak uitvoerige analyse. Door o.a. de opkomst van terminals en vooral de grafical display en het gebruik van minicomputers gaat het zg. interaktief werken steeds meer tot de mogelijkheden behoren. Met in dit verslag beschreven systeem wordt beoogd een bijdrage in deze richting te leveren.

In het boven omschreven ontwerpstrategieschema is nog steeds sprake geweest van een zekere recht toe recht aan aanpak, laat ons zeggen van een klassieke synthese. Wanneer deze klassieke aanpak niet tot het gewenste resultaat leidt kunnen tegenwoordig iteratieve automatische optimaliseringsmethoden worden gebruikt om te komen tot een ontwerp dat aan de opgegeven eisen voldoet of deze zelfs zoveel mogelijk overtreft.
Vaak kunnen daar met succes exakte methode worden gebruikt om een "startoplossing" voor de optimaliseringsmethoden te genereren.
Voor deze automatische optimaliseringsmethoden is vereist de formulering van een kostenfunctie en natuurlijk moet er een analyseprogramma aanwezig zijn. We kunnen t.a.v. filterontwerpen komen tot een volgorde zoals in fig1.2. weergegeven. Hierin is binnen het gestippelde blok het automatische gedeelte weergegeven, terwijl de rest gezien kan worden als een menselijke ingreep.

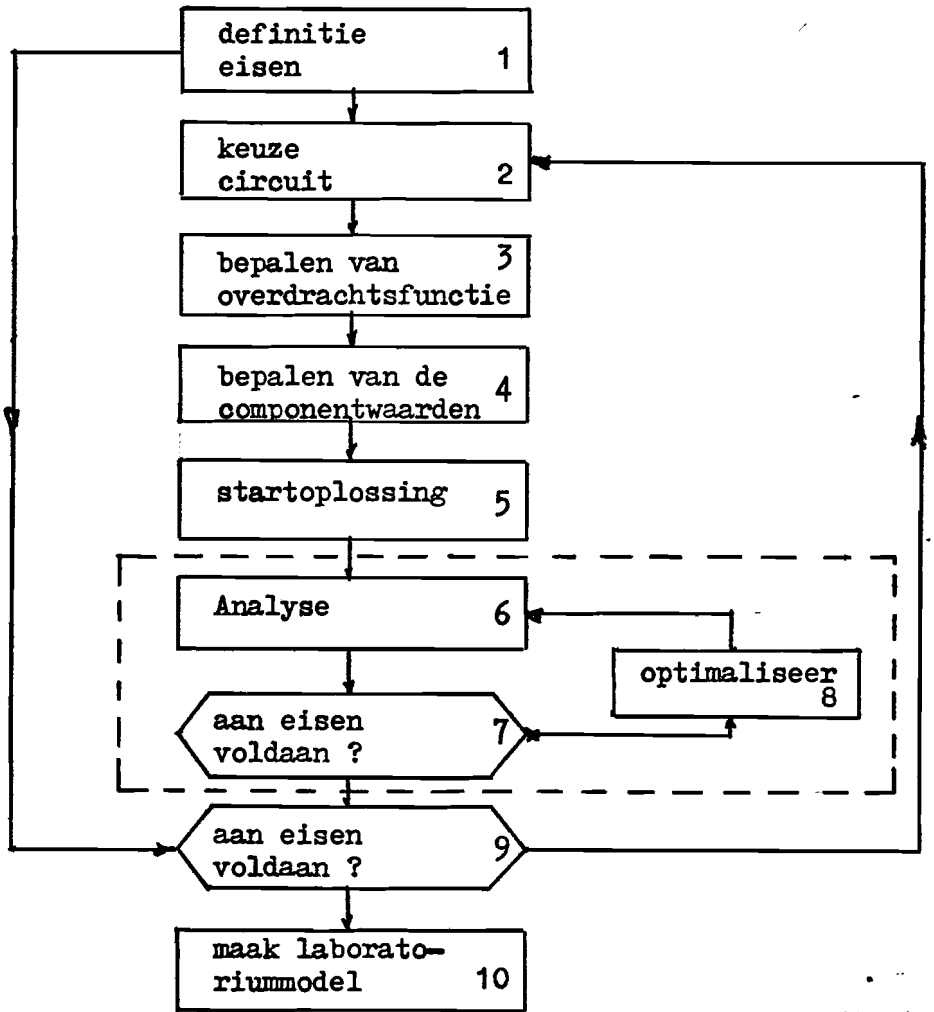


fig.1.2

Het schema van fig.1.2 dient als uitgangspunt voor het in dit verslag beschreven systeem. We hebben ons echter de volgende berperkingen opgelegd:

1. De cicuitkeuze is beperkt tot laddernetwerken. De keuze bestaat hierin dat het aantal en de configuratie (L's en C's) van de takken variabel is.
2. We beperken ons tot de overdrachtsfunctie van deze laddernetwerken.
3. De defenitie van de eisen is in de vorm van maskers waarbinnen de overdrachtsfunctie zich moet bevinden. zie fig.1.3

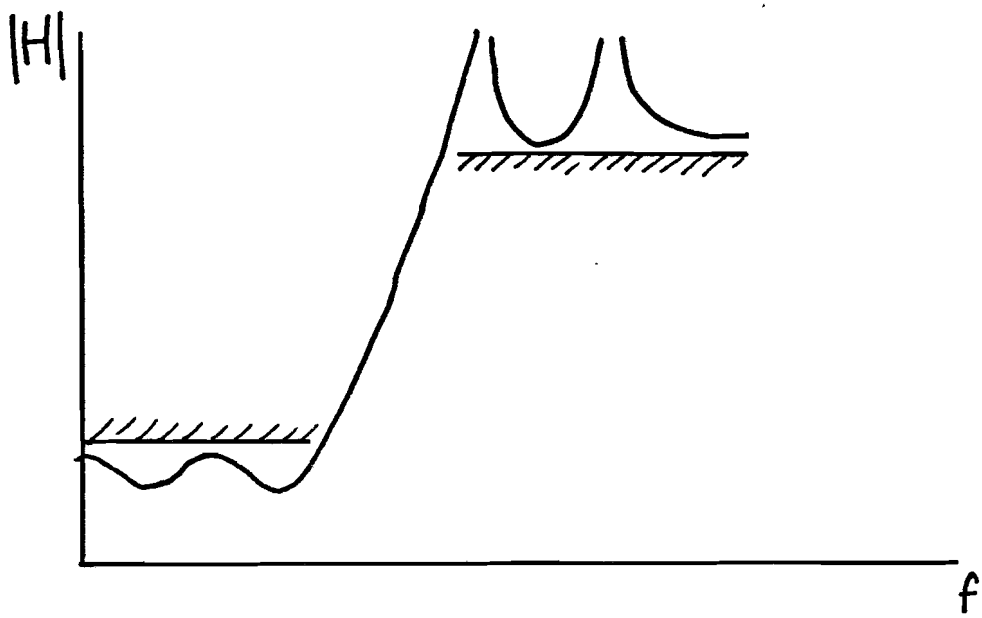


fig.1.3

Het ontworpen programma heeft hoofdzakelijk betrekking op het gestippelde blok. In hoofdst.2 wordt nader ingegaan op de formulering van het optimaliseringsprobleem en de daarvoor noodzakelijke objectfunctie.

2. Formulering van het optimaliseringsprobleem.

2.1 Zonder randvoorwaarden (unconstrained)

Uitgangspunt voor het optimaliseringsprogramma in het gestippelde blok van fig.1.2 is dat we beschikken over een redelijke startoplossing, dwz. een netwerkrealisatie waarmee de geëiste overdrachtsfunctie redelijk benaderd is.

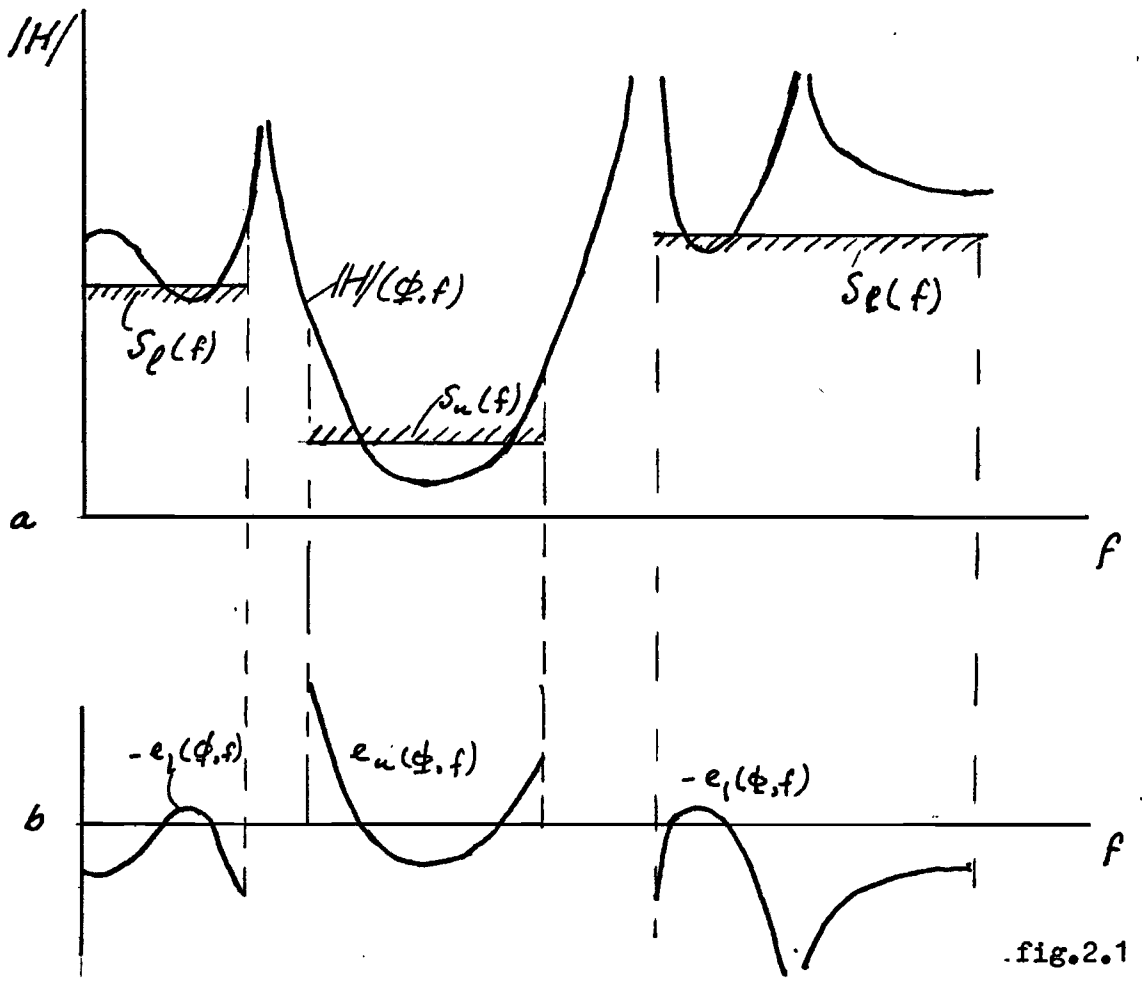


fig.2.1

Er blijft dan nog een afwijking t.o.v. specificaties over die door optimalisering geminimaliseerd moet worden.
In fig.2.1 is de overdrachtsfunctie H (Φ,f) weergegeven evenals boven- en onderspecificaties resp. $S_u(f)$ en $S_l(f)$. Φ is de parametervektor, f is de frequentie. In fig.2.1b is de foutfunctie weergegeven. Aangenomen is dat H (Φ,f), S(f) reële functies zijn.

We kunnen dan schrijven:

$$e_u(\phi,f) \triangleq w_u(f)\{H(\phi,f) - S_u(f)\} \tag{2.1}$$

$$e_l(\phi,f) \triangleq w_l(f)\{H(\phi,f) - S_l(f)\} \tag{2.2}$$

met w_u en w_l als weegfuncties, zodat

$$e_{ui}(\phi) \triangleq e_u(\phi,f_i),\ i \in I_u \tag{2.3}$$

$$e_{li}(\phi) \triangleq e_l(\phi,f_i),\ i \in I_l \tag{2.4}$$

De indices u en l staan voor resp. upper- en lower specificatie.
De indexsets I_u en I_l behoeven niet disjunct te zijn.
Het optimaliseringsprobleem kan nu geformuleerd worden als:

minimaliseer U onder voorwaarde

$$U \geq e_{ui}(\phi),\ i \in I_u \tag{2.5}$$

$$U \geq -e_{li}(\phi),\ i \in I_i \tag{2.6}$$

Dit kan ook geschreven worden als:

minimaliseer U met

$$U = \max\{(e_{ui}(\phi), i \in I_u),(-e_{li}(\phi), i \in I_l)\} \tag{2.7}$$

Hierbij betekent:

$U > 0$: de specificatiesworden overschreden.
$U = 0$; er wordt juist voldaan aan de specificaties.
$U < 0$: er wordt voldaan aan de specificaties, de overschrijding wordt geoptimaliseerd.

Met de weegfuncties kan bereikt worden dat we aan een of meerdere overschrijdingen extra aandacht besteden, door hiervoor een grotere weegfunctie te kiezen. Als aan de specificaties is voldaan, kunnen we aandacht aan de rest besteden.

Vooruitlopend op de in het volgende hoofdstuk beschreven GRAZOR-search schrijven we het optimaliseringsprobleem als:

minimaliseer

$$U(\underline{\phi}) = \max_{1 \leq i \leq n} y_i(\underline{\phi}) \tag{2.8}$$

waarbij $\underline{\phi}$ de vektor van k onafhankelijke parameters voorstelt en y_i is de foutfunctie op het samplepunt i, die een reële, niet lineaire differentieerbare functie is.
Laat $\hat{y}_l(\underline{\phi})$, $l=1,\ldots,n_r$ de grootste locale discrete maxima (ripples) [3] van $y_i(\underline{\phi})$, $i=1,..,n$ zijn, dan gaat (2.8) over in

minimaliseer

$$U'(\underline{\phi}) = \max_{1 \leq l \leq n_r} \hat{y}_l(\underline{\phi}) \tag{2.9}$$

2.2 Met randvoorwaarden (constrained).

We beschouwen het probleem

minimaliseer $U(\underline{\phi})$

onder voorwaarde

$$g_j(\underline{\phi}) \geq 0, \quad j=1,..,m \tag{2.10}$$

waarbij g_j in het algemmeen een niet lineaire functie van de parameters is.
We formuleren dan de te minimaliseren functie met randvoorwaarden als volgt:

$$W(\underline{\phi}, \underline{w}) = \max_{\substack{1 \leq i \leq n \\ 1 \leq j \leq m}} \{ y_i(\underline{\phi}), y_j(\underline{\phi}) \} \tag{2.11}$$

waarbij

$$y_j(\underline{\phi}) = -w_j g_j(\underline{\phi}) \tag{2.12}$$

$$\underline{w}=(w_1,w_2,\ldots\ldots w_m)^t \tag{2.13}$$

$$w_j>0, j=1,2,\ldots.m \tag{2.14}$$

Voor implementatie in GRAZOR is gekozen voor de volgende formulering:

$$W'(\phi,\underline{w}')=\max_{\substack{1\le l\le n_r \\ j\in J_v}}\left\{\hat{y}_1(\phi),\hat{y}_j(\phi)\right\} \tag{2.15}$$

waarbij

$$\hat{y}_j(\phi)=-w_j'.g_j(\phi) \tag{2.16}$$

$$J_v\subset J \text{ met } J:\{1,\ldots.,m\} \tag{2.17}$$

$$J_v := \left\{j \mid j\in J\wedge\hat{y}_j(\phi)>0\right\} \tag{2.18}$$

$$\underline{w}'=(w_1,w_2,\ldots\ldots w_m)^t \tag{2.19}$$

$$w_j'>0 \tag{2.20}$$

Bij deze formulering blijft gelden dat als W'>0 niet aan de specificaties of aan de randvoorwaarden is voldaan terwijl W'=0: eris juist aan voldaan, en W'<0: betekent dat aan alle eisen is voldaan en de overschrijding kan dan nog geoptimaliseerd worden.

In het geval van parameterconstraints (box-constraints) kunnen boven- en ondergrenzen als volgt beschreven worden:

$$g_{2l-1}(\phi)=\phi_i-\phi_{il}\geq 0 \tag{2.21}$$

$$l=1,2,\ldots.,m$$

$$g_{2l}(\phi)=-(\phi_i-\phi_{iu})\geq 0 \tag{2.22}$$

en overeenkomstig (2.16)

$$\hat{y}_j(\phi)=-w_j'\not\propto g_j(\phi) \tag{2.23}$$

Dan kunnen ook de elementen van de gradiëntvektor $\nabla \hat{y}_j(\phi)$ eenvoudig geformuleerd worden als:

$$\partial \hat{y}_j / \partial \phi_k = \begin{cases} 0 & \text{voor } j \neq k \\ \left.\begin{matrix} w_j & \text{voor bovengrens} \\ -w_j & \text{voor ondergrens} \end{matrix}\right\} & \text{en } j=k \end{cases}$$

3. De optimaliseringsmethode: Grazor-seareh.

3.1 Inleiding.

In hoofdstuk 2 is het optimaliseringsprobleem geformuleerd als een zg.minimax-probleem:

Minimaliseer

$$U(\phi) = \max_{1 \leq i \leq n_r} \hat{y}_i(\phi) \qquad (3.1)$$

Hierbij is $\hat{y}_i$ een lokaal discreet maximum (ripple) van de te minialiseren functie. De objectfunctie U wordt gekenmerkt door "scherpedalen" voor die waarden van ϕ waar twee of meer toppen ($\hat{y}_i$'s) aan elkaar gelijk zijn. Deze scherpe dalen vormen paden van discontinue partiele afgeleiden. Daardoor zijn gradientmethoden zoals bv.van Fletcher-Powell niet zonder meer te gebruiken.
Bandler [2] heeft eerst een pattern-search methode ontwikkeld om bovenstaand minimax-probleem te optimaliseren, genaamd de RAZOR - search. De naam is ontleend aan scherpe dalen (scheermes) van discontinue afgeleiden, die door de methode gevonden en gevolgd worden om te komen tot een minimum in het hierboven geformuleerde minimax probleem. Hierbij werden geen afgeleiden gebruikt.
In 1972 publiceerde Bandler een tweede methode: de Grazor-search [3]. Deze is afgeleid van de Razor-search met als uitbreiding het gebruik van gradiënten.
Deze methode is hier geïmplementeerd en zal nu nader beschreven worden.
De strategie is als volgt:
Beginnend met de maximale $\hat{y}_i$ bij een startpunt ϕ_{start} wordt een zoekrichting bepaald tegengesteld aan de gradiëntrichting om een minimum te vinden. Verondersteld is dan dat een punt dicht bij een pad van discontinue afgeleiden is bereikt. Er wordt dan een lineare methode gebruikt om de bergafwaartse richting langs het pad van de discontinue afgeleiden te volgen.

Verder wordt in deze richting dan een lijnminimum gezocht. Als dit bereikt is, wordt verondersteld dat de subset "actieve" $\hat{y}_i$'s met één is uitgebreid. Dit gaat zo door totdat een minimax oplossing met een zekere nauwkeurigheid is bereikt.

3.2 Uitvoeriger beschrijving van de Grazor-search.

De grazor-search strategie is in het stroomschema van fig.3.1 weergegeven. Voor de theoretische beschouwingen wordt verwezen naar het oorspronkelijke artikel [3]. Om een inzicht te krijgen in de werking is het van belang te weten wat de belangrijkste subroutines doen. Daarom wordt hier eerst een overzicht van de functies van de belangrijkste subroutines gegeven.

3.2.1 Overzicht van de belangrijkste routines.

1. SELEC.
   Deze routine bepaalt de locale discrete maxima $\hat{y}_i, i = 1,\ldots,n_r$ en sorteert deze in volgorde met afnemende grootte.
2. GRADIE.
   Hier wordt gradient $\nabla\hat{y}_1$ bepaald.
3. Lineair programma.
   Wanneer twee of meer functies aktief zijn (op een pad discontinue afgeleiden), wordt met behulp van dit lineaire programma een zoekrichting $\Delta\phi$ bepaald. Het programma -een simplexalgorithme- lost het volgende lineare optimaliseringsprobleem op bij een punt ϕ^j:

$$\text{Maximaliseer: } \alpha_{k_r+1}(\phi^j) \geq 0 \tag{3.2}$$

Onder voorwaarde

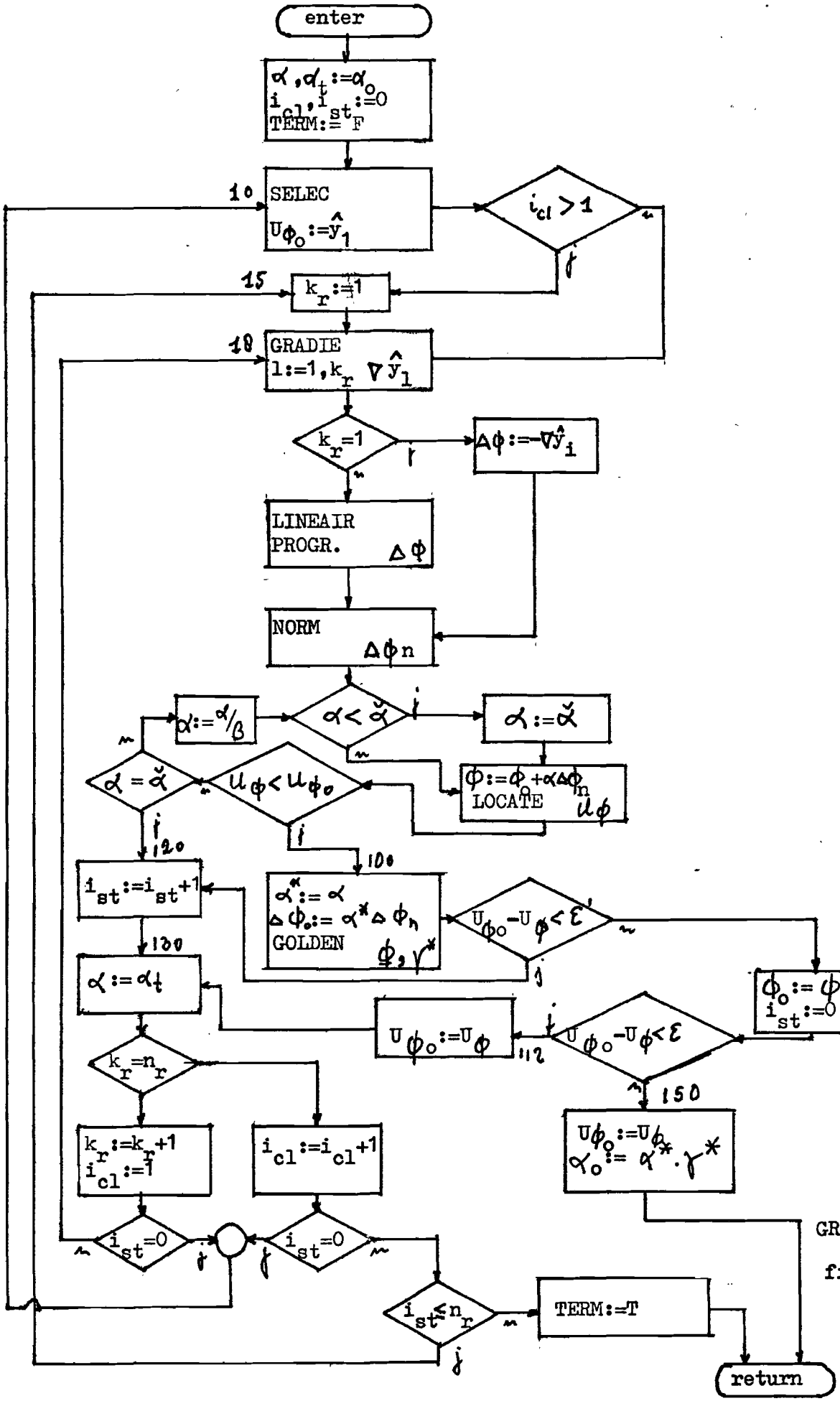
$$-\underline{\nabla}^t\hat{y}_i(\phi^j) \sum_{l=1}^{k_r} \alpha_l^j \underline{\nabla}\hat{y}_l(\phi^j) \leq -\alpha_{k_r+1}^j \;, \quad i=1,\ldots,k_r \tag{3.3}$$

$$\alpha_i^j \geq 0, \quad i=1,\ldots,k_r \tag{3.4}$$

$$\sum_{l=1}^{k_r} \alpha_l^j = 1 \tag{3.5}$$

waarbij $\hat{y}_l(\phi^j)$, $l=1,\ldots,k_r$ de hoogste in beschouwing genomen toppen zijn ($k_r \leq n_r$).
Dan wordt:

$$\Delta\phi^j = -\sum_{l=1}^{k_r} \alpha_l^j \underline{\nabla} y_l(\phi^j) \tag{3.6}$$



GRAZOR

fig.3.1

4. NORM.
De zoekvektor $\Delta\Phi^j$ wordt genormaliseerd tot:

$$\Delta\Phi_n^j = \Delta\Phi^j / \|\Delta\Phi^j\|$$

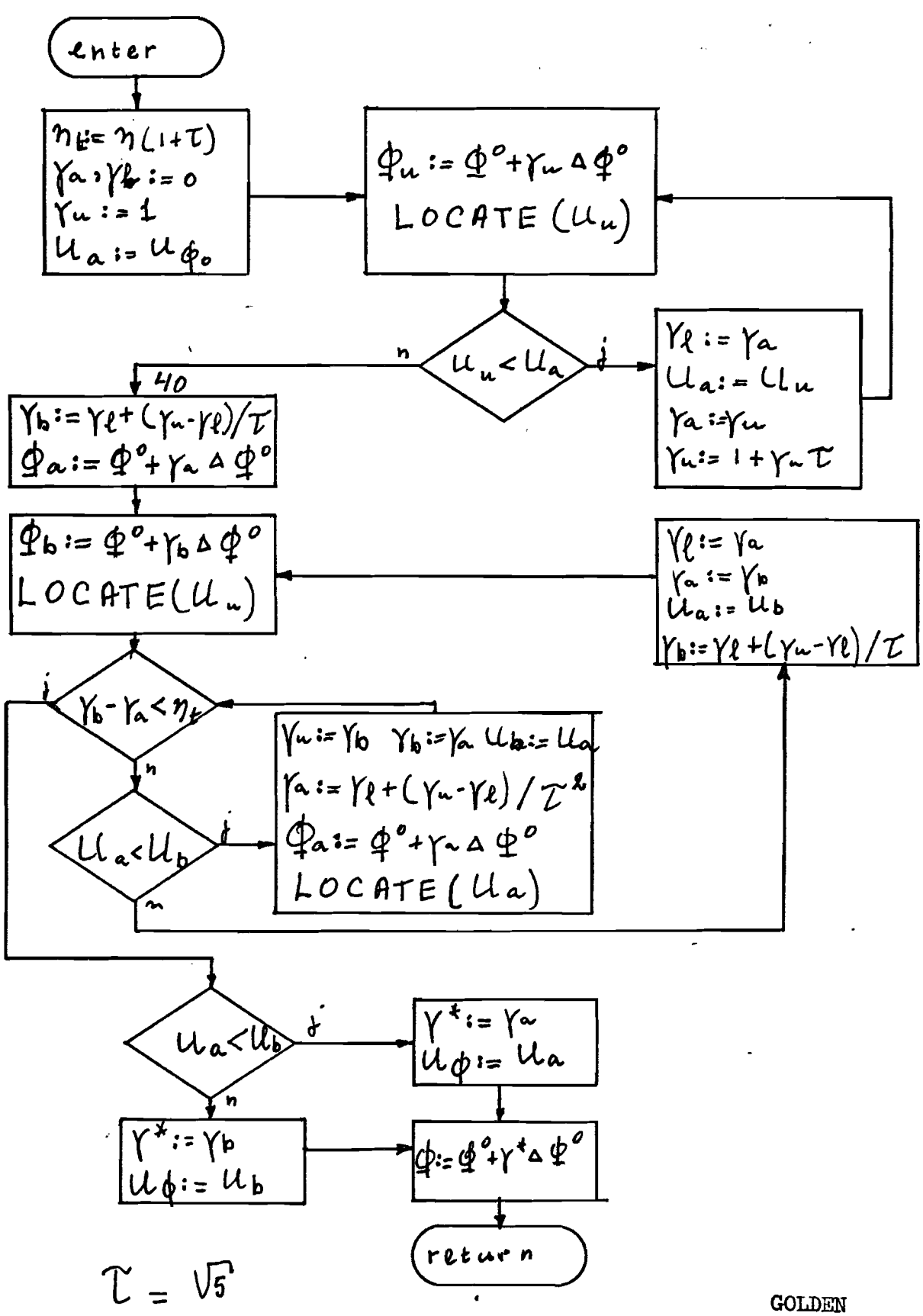
5. LOCATE.
Met deze routine wordt telkens (voor elke nieuweΦ) de echte object-functie U_Φ bepaald als het globale maximum van alle n samplepunten n n_r, zie form.(2.8).

6. GOLDEN.
Van de subroutine GOLDEN is in fig.3.2 een stroomschema gegeven. Het opmerkelijke van deze methode, die is voorgesteld door Temes [7] , is, dat eerst d.m.v. verlenging van de zoekvektor een gebied wordt bepaald, waarop de functie unimodaal is. Als dit interval is bepaald, begint bij 40 de eigenlijke Gulden- snede methode.
Wanneer het interval tot een zekere resolutie (η_t) is verkleind, wordt de gulden-snede zoekmethode afgebroken en is er een faktor γ^* bepaald.
De vektor Φ wordt dan: $\Phi = \Phi^0 + \gamma^* \Delta\Phi^0$.

3.2.2 De strategie.

In de Grazor strategie zijn drie hoofdfasen te onderscheiden.

1. Eerst wordt geprobeerd een dalende richting te vinden tegengesteld aan de gradiënt van de grootste functie (kr = 1), dan wordt in deze richting een minimum gezocht en dit proces wordt herhaald (return).
2. Wanneer in een bepaalde stap dit proces of het lineaire programma geen richting meer oplevert waarin U verlaagd kan worden, of een verbetering optreedt die kleiner is dan ζ wordt de procedure herhaald echter met meenemen van de functie corresponderend met de volgende grootste top ($\hat{y}_i$) van de n_r discete locale maxima die in selec gesorteerd zijn.
3. Als alle locale maxima op deze manier zijn meegenomen ($k_r = n_r$) en U kan nog steeds niet verlaagd worden of U kan met niet meer dan ζ verbeterd worden, herhalen we het proces ($i_{cl} := i_{cl} + 1$) met k_r functies corresponderend met de eerste k_r grootste kandidaten, beginnend met $k_r = 1$.
Het algorithme wordt pas afgebroken als het laatste met alle n_r



GOLDEN

fig. 3.2

functies is geprobeerd en er geen verbeteringen ($U_{\phi} < U_{\phi_0}$) of geen verbeteringen kleiner dan een zekere resolutie zijn opgetreden over een hele cyclus van k_r van 1 tot n_r dus als $k_r = n_r$ en $i_{st} > n_r$.

3.2.2.1 Enkele opmerkingen hierover die het lezen van het stroomschema van fig. 3.1 mogelijk vergemakkelijken.

In dit en alle volgende stroomschema's zijn labelnummers opgenomen. Deze nummers komen overeen met de labelnummers in de betreffende programma-subroutines.

1. Nadat een zoekrichting bepaald en genormaliseerd is wordt een α bepaald, zodanig dat bij $\phi = \phi^0 + \alpha \Delta \phi$ een verbetering (vermindering) van U_{ϕ} optreedt. Als hier een verbetering wordt gevonden is er een voldoende startinterval voor Golden gevonden. Dit interval wordt eventueel binnen Golden uitgebreid (zie 3.2.1 onder GOLDEN).
   Er wordt dan een minimum gezocht in Golden. Als de gevonden U_{ϕ} met meer dan ε is afgenomen ($U_{\phi_0} - U_{\phi} > \varepsilon$) wordt Grazor opnieuw gestart (return) met $\alpha_0 := \alpha^* \cdot \gamma^*$. Er is in deze stap dus "geleerd".
   Dat we hier terug moeten naar het begin heeft twee redenen:
   a. Bij deze betrekkelijk grote stap kunnen de toppen wel eens heel anders liggen (dus opnieuw SELEC).
   b. Vanwege de nietlineaire functies kunnen de gradiënten bij deze nieuwe ϕ wel eens heel anders zijn (dus opnieuw GRADIE).

2a. Wanneer $U_{\phi_0} - U_{\phi} < \varepsilon$ wordt er verondersteld dat een scherp dal is bereikt en dat er dus een top is bijgekomen (mits $k_r < n_r$). Dan wordt $k_r := k_r + 1$. Er geldt: $i_{st} = 0$ (omdat er een verbetering van U_{ϕ} is opgetreden) dus wordt er opnieuw geselekteerd in SELEC en de procedure wordt herhaald met k_r toppen.

2b. Als U_{ϕ} door verkleinen van α niet verbeterd kan worden of als de verbetering minder bedraagt dan een zekere resolutie ε' komen we bij 120 en wordt i_{st} opgehoogd.
   Dan wordt k_r opgehoogd en we proberen op deze manier (d.w.z. zonder nieuwe SELEC) een verbetering te krijgen door de volgende top bij het proces te betrekken (18.GRADIE, $1 := 1, k_r$).

3. Wanneer $k_r = n_r$ wordt i_{cl} opgehoogd. Verondersteld is dan dat het globale minimum is bereikt. Er begint dan een tweede cyclus.
   Hier kunnen we weer twee gevallen onderscheiden:

a. $i_{st}=0$:Dit betekent dat we alle toppen in de set aktieve y_i's hebben betrokken want de laatste gaf een verbetering $\leq \varepsilon$ van U_ϕ . Er wordt dan opnieuw geselekteerd.
De tweede cyclus begint dan d.w.z. we gaan nog een laatste poging doen om een verbetering te krijgen. Omdat $i_{cl}>1$ wordt $k_r:=1$. Dus we beginnen met één top. Als we op deze manier geen verbetering van U_ϕ krijgen wordt steeds i_{st} opgehoogd totdat we de set beschouwde toppen hebben uitgebreid tot alle n_r toppen.
Als dit laatste gebeurt ($k_r=n_r \wedge i_{st}>n_r$) wordt Grazor afgebroken.

b. i_{st} 0:Dit betekent dat in de vorige cyclus reeds één- of meermalen geen verbetering van U_ϕ is verkregen. We maken dan de onder 3a beschreven tweede cyclus af door uitbreiding van de set beschouwde toppen tot $i_{st}>n_r$. Dan wordt Grazor afgebroken.

4. Organisatie programma.

4.1 Algemeen

Enkele uitgangspunten.

Het programma is geschreven in FORTRAN-IV - PLUS voor gebruik op een PDP -11/55. Er wordt gebruik gemaakt van een grafical display en wel de T 4014 van Tektronix. Bij deze T 4014 is aanwezig een hard- copy apparaat, waarmee copiën kunnen worden gemaakt van wat er op het scherm staat.

Het optimaliseringsprogramma vormt het belangrijkste onderdeel van het beschreven systeem. In feite is de rest rondom dit optimaliserings-programma gebouwd. Voor de hiervoor gekozen GRAZOR- search en de daaruit overgenomen nomenclatuur wordt verwezen naar artikel van Bandler [3].
Een belangrijke benodigheid voor dit optimaliseringsprogramma is het netwerkanalyseprogramma, beschreven in 4.2.
Voor de gekozen GRAZOR- search en de in hoofdst. 2 geformuleerde uitgangspunten zijn voor dit programma enkele specifieke arrays gedeclareerd, waarmee de organisatie van het programma in grote lijnen is bepaald. Deze worden in 4.3 besproken.

4.2. Het netwerkanalyseprogramma.

Voor het netwerkanalyseprogramma kan een dankbaar gebruik worden gemaakt van het door V.F.Nicola [4] in een stage in de groep ES gemaakte programma. De voor ons noodzakelijke procedures zijn vanuit Algol naar Fortran vertaald en eventueel aangepast aan onze situatie. We zullen hier volstaan met het bespreken van de hoofdzaken. Voor de details wordt verwezen naar [4].
Met het programma, binnen ons programma is het de subroutine ANAL, is het mogelijk om de overdrachtsfunctie ABSH en de partiele afgeleiden van ABSH naar de netwerkelementen te bepalen. De struktuur van het netwerk is beperkt tot de in fig.4.1 weergegeven ladderstruktuur, waarbij elke Z en Y opgebouwd kan zijn uit L's en C's (maximaal 4 elementen), met dien verstande dat de eerste Z en de laatste Y een weerstand moet zijn. Hiermee wordt bereikt, dat voor het betreffende filter behalve de spanningsoverdracht (hier ABSH) ook de vermogensoverdracht en bv.ook de reflexiecoëfficiënten in beschouwing kunnen worden genomen.
(zie bv.[1] blz.14 e.v.)

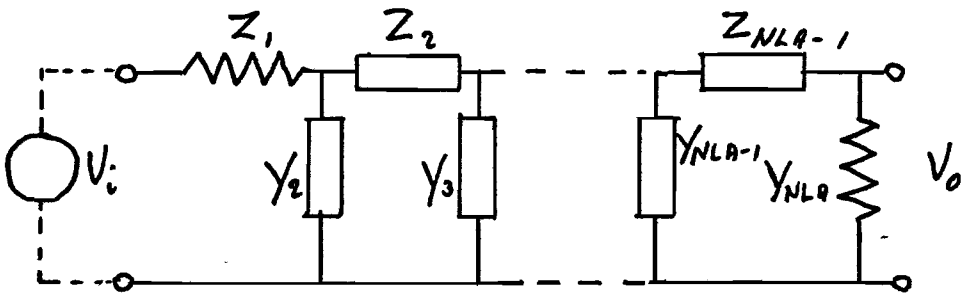


fig.4.1

Nicola [4] heeft de componentwaarden, ter onderscheiding tussen een L en een C, resp.een + of een - teken meegegeven. Dit teken dient alléén ter onderscheiding en men mag er dus geen arithmetische betekenis aan toekennen. Voor het inlezen worden Z en Y identiek behandeld. Met behulp van de subroutine ZTY worden de even Z's getransformeerd naar Y's. Het laddernetwerk bestaat uit NLA (N-ladder) Z's met een maximum van 10. Voor Z_1 en Z_{NLA} is een vast programma binnen CAZDZ geschreven.
De elementswaarden worden opgeslagen in de array ZN resp.ZNO (komen we

nog op terug in 4.3)

De netwerkstruktuur van $Z_i, i = 1,\ldots,$ NLA wordt opgeslagen in de array IZO en wel met de volgende betekenis:

1 = data

2 = parallel

3 = serie

Dit kan het beste met een voorbeeld geïllustreerd worden. zie fig. 4.2

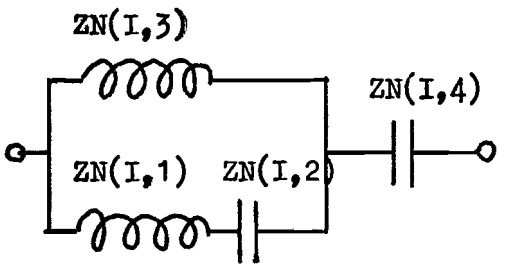


fig.4.2

De getalwaarden van L's en C's staan in ZN(I,J)

Voor dit schema ziet IZO er als volgt uit (1,1,3,1,2,1,3).

Met de subroutine CAZDZ worden dan dan de $Z_i, i = 1,\ldots,$NLA berekend en tevens de partiële afgeleiden van Z naar het element $p_j, j=1,\ldots,4$

Deze worden opgeslagen in het array Z(10,5) en wel als volgt:

$Z(I,5) = Z_i$

$Z(I,J) = \partial Z_i / \partial p_j$, J = 1,...,4

Met de subroutine CAHG kan ABSH = V_o/V_i bepaald worden.

4.2.1 Gradiëntberekening.

Met behulp van adjoint-analysis [5] worden indien nodig in de subroutine CAHG de afgeleiden $\partial|H|/\partial Z_i$ berekend. Dan is $\partial|H|/\partial p_j = \partial|H|/\partial Z_i * \partial Z_i/\partial p_j$.

4.3 De belangrijkste arrays.

De afmeting van de betreffende array is bij de naam gegeven.

1. BOU (20,4)

   In deze array worden de gegevens van de maskers (zie fig.2.1), de boundaries, opgeslagen.

   Bou (I,1): laagste frequentie van grens
   Bou (I,2): hoogste frequentie van grens
   Bou (I,3): waarde van de grens
   Bou (I,4): weegfactor van deze I_e grens.

   Op de adressen 1 t/m 10 kunnen max. 10 bovengrenzen en op 11 t/m 20 kunnen max. 10 ondergrenzen worden opgeslagen.

2. CON (10,5)

   In CON worden gegevens over de variabele componentenvektor opgeslagen.

   CON (I,1): pointer voor I van ZN (I,J) wijst dus ladderelement aan.
   CON (I,2): pointer voor J van ZN (I,J) wijst de L of C aan.
   CON (I,3): bovengrens van L of C
   CON (I,4): ondergrens van L of C
   CON (I,5): weegfactor.

3. PSI (100)

   Een eendimensionale array, waarin opgeslagen alle voor Grazor noodzakelijke frequentiepunten. Eerst alle frequentiepunten m.b.t. de bovengrenzen; daarachter alle m.b.t. de ondergrenzen.

4. IPSI (20,2)

   Tweedimisionale array waarin is opgeslagen de informatie over welke frequentiepunten bij een bepaalde boundary horen.

   IPSI (I,1): adres van PSI waar het eerste frequentiepunt van de betreffende boundary staat.
   IPSI (I,2): adres van PSI waar laatste punt van betreffende boundary staat.
   I : nummer van de grens (= eerste index van BOU)

Voorbeeld:

Stel we hebben 2 bovengrenzen en 1 ondergrens en de gegeven frequentiepunten. zie fig. 4.3

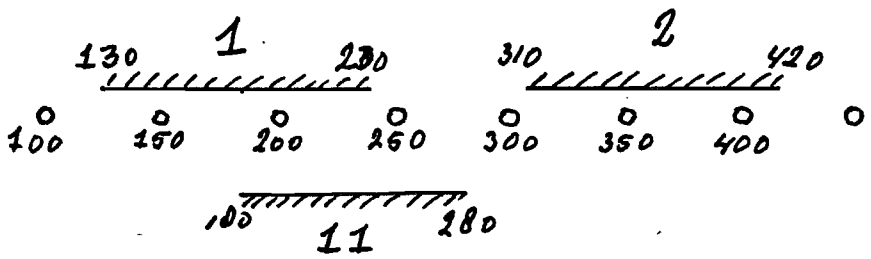


fig.4.3

Dan is de inhoud van onderstaande arrays bv. als volgt:

PSI (130,150,200,230,310,350,400,420,180,200,250,280)
1 2 3 4 5 6 7 8 9 10 11 12

IPSI
1 [1, 4]
2 [5, 8]
11 [9,12]

BOU
1 [130,230,50, 2]
2 [310,420,30,100]
11 [180,280,20, 5]

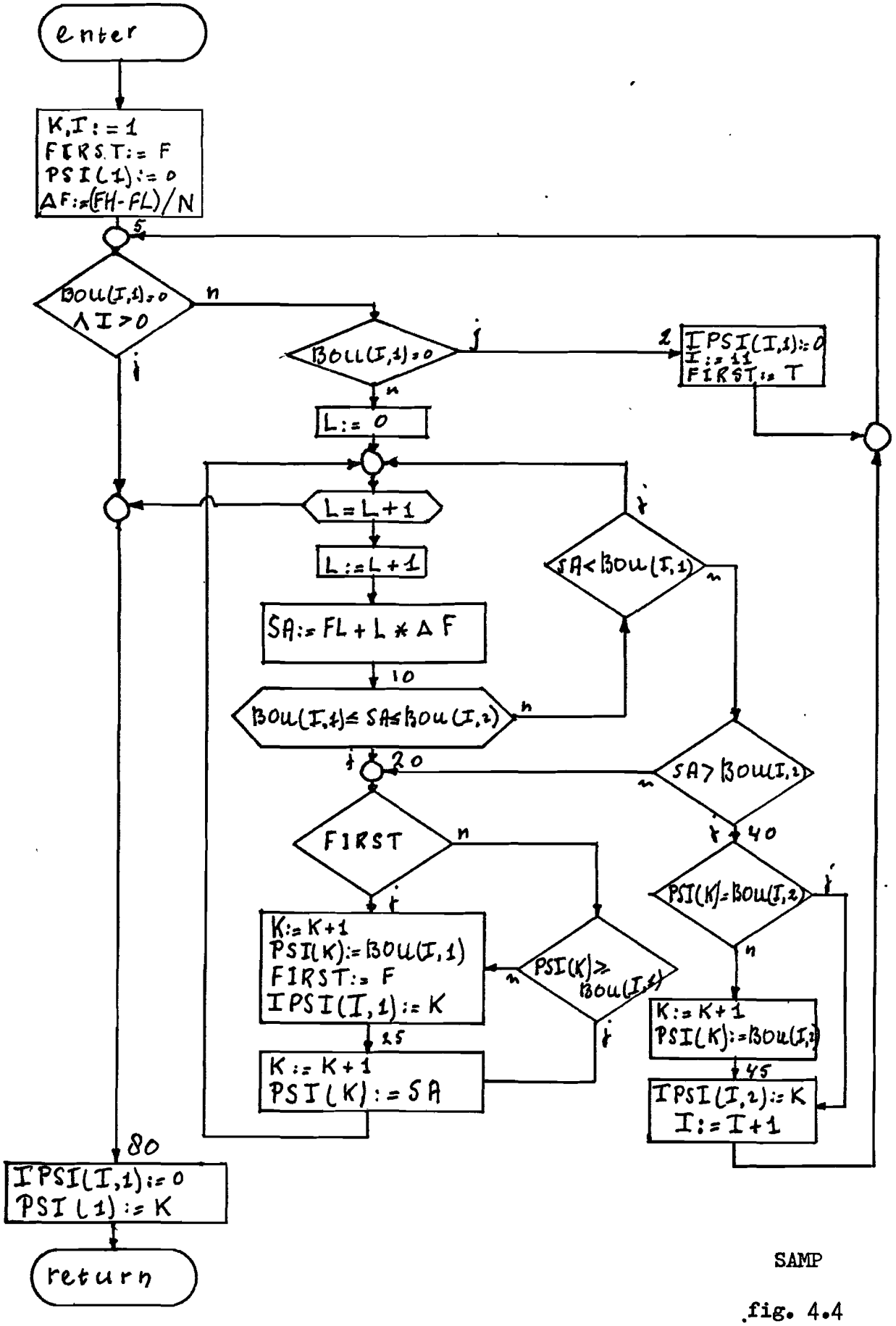
5. ZN (10,4) en ZNO (10,4)
   Deze arrays (reeds behandeld in Hoofdst. 4.2) bevatten de componentwaarden. De 0 in ZNO staat voor optimaal. De array ZN dient als "klad"-array tijdens excursies met de componentvektor ϕ binnen GRAZOR. Na elke Grazor-slag wordt ZNO veranderd.
6. IZO (10,7)
   Hierin is opgeslagen de struktuur van het netwerk. zie 4.2.
7. PH (10) en PHO (10)
   Deze arrays vormen de variabele componentvektor ϕ resp. ϕ_0 voor GRAZOR. zie hoofdst . 4.5.
8. YM (16)
   Hierin worden opgeslagen de locale discrete maxima $\hat{Y}_i, i = 1,\ldots,n_r$ die in SELEC bepaald en gesorteerd worden.
9. IYM (16)
   In deze integer-array staat op de I_e plaats het nummer van de bij

YM (I) behorende boundary, als YM (I) een ABSH-functie is. Als YM (I) een parameterconstraint-functie is staat in IYM (I) 21 voor een bovengrens of 22 voor een ondergrens.

10. PSIM (16)
Hierin staat het frequentiepunt voor YM (I) of het nummer van de parameter als YM (I) een parameterconstraint-functie is.

4.4 SAMP. de routine om de samplepunten te bepalen.

Met behulp van deze routine worden de voor GRAZOR noodzakelijke samplepunten bepaald, die dan in PSI worden opgeslagen.
We zijn er hierbij vanuit gegaan, dat door de gebruiker een N = "Number of samples for Grazor" wordt opgegeven. Deze N bepaalt in eerste instantie het frequentieraster.(zie fig.4.2). De frequentiepunten waarvoor de overdrachtsfunctie niet begrensd is, worden niet meegenomen. Echter de randpunten van de grenzen zijn voor Grazor belangrijk omdat hier vaak een top in de foutfunctie optreedt.(zie fig.2.1.)
In de gemaakte subroutine (fig.4.4) is er voor gezorgd dat de hierboven genoemde frequentiepunten in PSI worden opgeslagen en wel slechts éénmaal, eerst voor de bovengrenzen ($I \leq 10$) en daarna voor de ondergrenzen ($11 \leq I \leq 20$). In PSI (1) komt het totaal aantal in PSI opgeslagen aantal frequentiepunten. Tevens worden in IPSI de adressen van PSI opgeslagen waar een bepaalde I_e grens de begin- en eindfrequentiepunten van de betreffende grens staan. Het een en ander zoals in hoofdst.4.3 in een voorbeeld is geïllustreerd.
IPSI (I,1) met I volgend op de laatst aanwezige boven-resp. ondergrens wordt "0" gemaakt voor testdoeleinden in de routines waar IPSI gebruikt wordt (SELEC en LOCATE).


enter
K,I:=1
FIRST:=F
PSI(1):=0
ΔF:=(FH-FL)/N
5
BOU(I,1)=0 ∧ I>0
n
i
BOU(I,1)=0
j
n
1
IPSI(I,1):=0
I:=I1
FIRST:=T
L:=0
L=L+1
L:=L+1
SA:=FL+L*ΔF
10
BOU(I,1)≤SA≤BOU(I,2)
n
i
SA<BOU(I,1)
n
i
SA>BOU(I,2)
n
i
20
40
FIRST
n
i
PSI(K)=BOU(I,2)
j
n
K:=K+1
PSI(K):=BOU(I,1)
FIRST:=F
IPSI(I,1):=K
n
PSI(K)≥BOU(I,2)
i
25
K:=K+1
PSI(K):=SA
K:=K+1
PSI(K):=BOU(I,2)
45
IPSI(I,2):=K
I:=I+1
80
IPSI(I,1):=0
PSI(1):=K
return

SAMP

fig. 4.4

4.5 CHANGE en RECHAN de transformatieroutines.

Bij de bespreking van de belangrijkste arrays zijn de arrays ZN en PH resp. ZNO en PHO genoemd. Ook in hoofdst. 4.1 is er reeds op gewezen dat we binnen GRAZOR mogelijk een subset van de netwerkcomponenten gebruiken. Echter bij elke functieëvaluatie (lees: netwerkanalyse) hebben we de hele set componenten (ZN of ZNO) nodig. Met CHANGE en RECHAN is het mogelijk om naar believe van PH (PHO) naar ZN (ZNO) resp. van ZN (ZNO) naar PH (PHO) te transformeren.

4.6 GRAZOR:de optimaliseringsroutine.

In hoofdstuk 3 is de strategie reeds besproken aan de hand van het in fig. 3.1 gegeven stroomschema. Dit stroomschema is door Bandler [3] in zijn artikel gegeven. Dit schema is in ons programma geïmplementeerd. Er is slechts één kleine wijziging gemaakt nl.de test $k_r = N_r$ is vervangen door $k_r \geq n_r$. Hierover het volgende:
De meegenomen objectfunctie t.g.v. overschreden parameterconstraints zijn niet continu. (alleen gedefinieerd voor overschrijding) zie hoofdstuk 2.2.
Daardoor is bijv.de volgende situatie denkbaar (en ook een keer voorgekomen):
Na een bepaalde stap treedt het volgende op:

SELEC NR=5, met par. constraint
KR=4
KR:=KR+1 dus KR=5
$I_{st} > 0$
GRADIE
$U_{\phi_0} - U > \varepsilon$
RETURN
GRAZOR opnieuw
SELEC NR=4 ,constraint niet meer aktief

Dan gaat het bij de test $k_r = n_r$ mis.Immers $k_r=5$ en $n_r=4$. K_r wordt dan ten onrechte opgehoogd en er wordt (als $i_{st} > 0$) een poging gedaan om een gradiënt te bepalen van een functie, die er niet is (PSIM (6) =0)
De aanpassing van GRAZOR aan ons specifieke geval zit hoofdzakelijk in de "probleemspecifieke" subroutines die in 4.6.1 t/m 4.6.4 worden besproken.

4.6.1 FUNCT de functie

Het geïmplementeerde algorithme is in het in fig.4.5 gegeven stroomschema weergegeven. Deze subroutine wordt als volgt aangeroepen.

FUNCT (↓ZN,↓F,↑Y,↑GRADY,↓KB,↓KC,↓DERIV)

↓: ingangsvariabelen

↑: uitgangsvariabelen (worden bepaald)

KB = nummer van grens voor overdrachtsfunctie of 21 of 22 voor parameterconstraints.

KC = nummer van parameter.

Het onderscheid tussen een functie en constraint en tussen boven- of-onderconstraint wordt getest met KB. Zoals in hoofdst. 2 reeds vermeld geldt de objectfunctie voor de constraints slechts bij overschrijding (Y=-W.g). Y kan alléén positief zijn. Deze functie is dus discontinue. Dit kan mogelijk moeilijkheden oproepen bij het bepalen van de gradiënt m.b.v. GRADIE, als bij de actuële ϕ_0 de betreffende constraint ineens net niet meer aktief is. Een mogelijke oplossing hiervoor zou kunnen zijn dat toch de gradiënt wordt bepaald en wel als zou de functie continu voortgezet worden. De functie is immers toch lineair.
Deze oplossing is eigenlijk vanzelf in het programma aanwezig. (nl.door er niets tégen te doen) Het laten vervallen van de Y als deze ≤ 0 is gebeurt in de aanroeproutines (SELEC en LOCATE).

Het probleem van de + en - van Nicola (zie hoofdst.4.2) is t.a.v. het bepalen van de objectfunctie voor de constraints opgelost door de grenzen van de parameters CON (KC,3) en CON (KC,4) ook een - teken te geven voor C's. Door gebruikmaking van SIGN en de weegfactor CON (KC,5) die altijd pos. is, is het een en ander in goede banen te leiden.

Voorbeeld.

ZN(ϕ)= -0.1179782 dus condensator

CON (KC,3) = -10^8

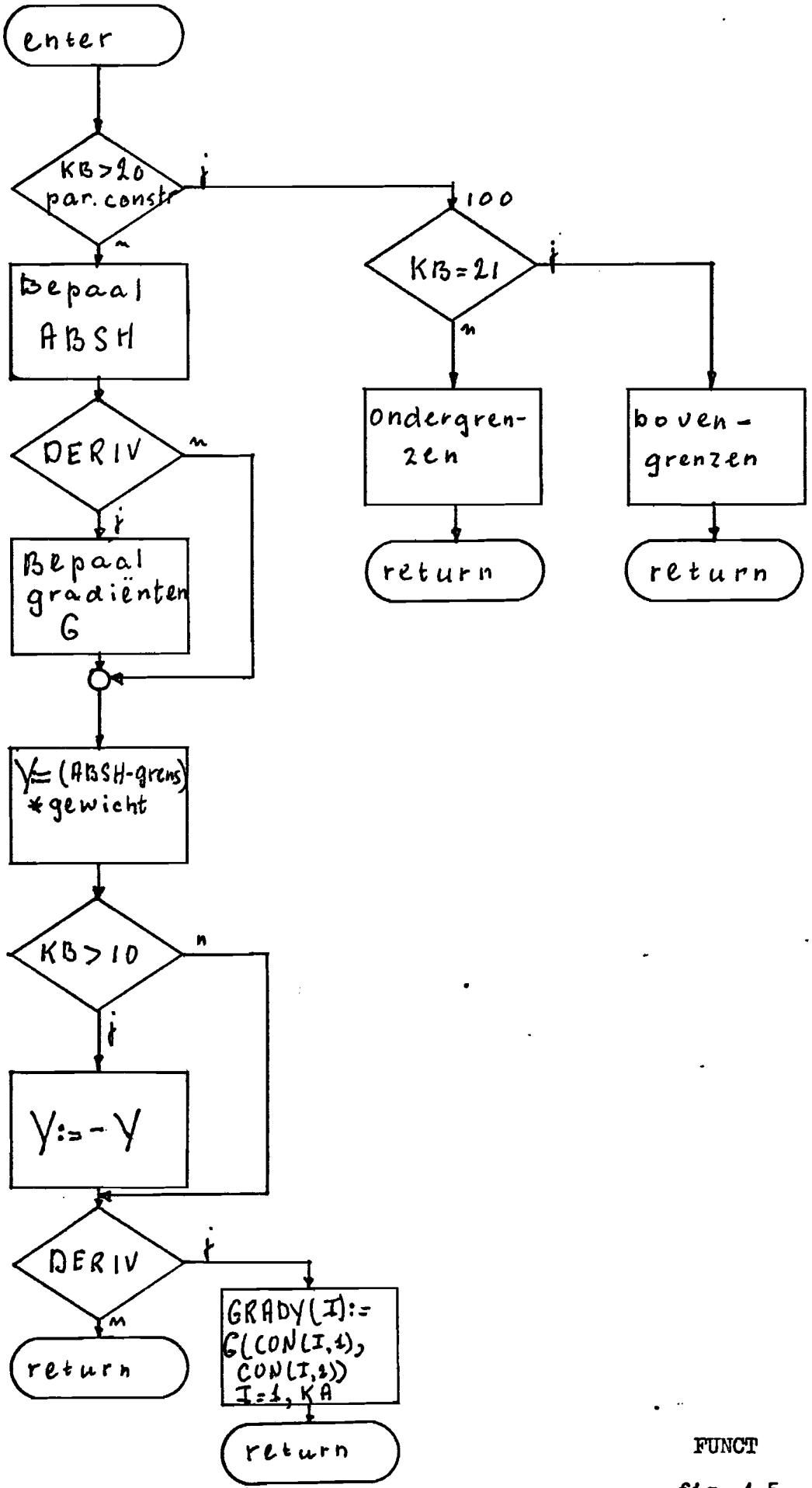
CON (KC,4) = - 0.118

CON (KC,5) = 100

Dan is:

$Y = (-0.118-(-0.1179782) * \text{sign}(100,-10^8) = + 0.00218$

Y>0, dus een overschrijding en dit klopt.


enter
KB>20
par. constr
j
100
n
KB=21
j
n
Bepaal
ABSH
DERIV
n
j
Bepaal
gradiënten
G
Ondergren-
zen
boven-
grenzen
return
return
Y:= (ABSH-grens)
*gewicht
KB>10
n
j
Y:=-Y
DERIV
j
n
return
GRADY(I):=
G(CON(I,1),
CON(I,2))
I=1, KA
return

FUNCT

fig. 4.5

4.6.2 SELEC en SORT het bepalen en sorteren van de toppen.

In deze routine worden de toppen (locale maxima) in de objectfunctie en de discrete $\hat{Y}_i$'s t.g.v. de parameterconstraints bepaald en gesorteerd.

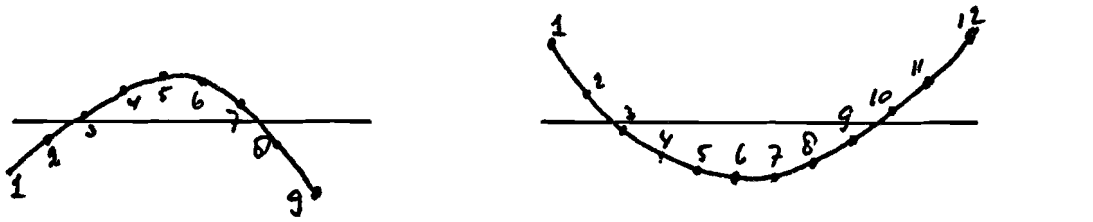
In het stroomschema van fig. 4.7 is het algorithme weergegeven.

1. Het bepalen van de toppen.

a. Het bepalen van een top gaat in principe als volgt:
We vergelijken twee samples als in het voorgaande de functie stijgend was (KD = 1) en nu dalend (YX < YZ) dan is er een top bij het vorige sample (30).

b. Dit gaat ook op voor het eerste sample binnen een begrenzing mits we de functie vooraf stijgend veronderstellen (initieel is KD = 1).

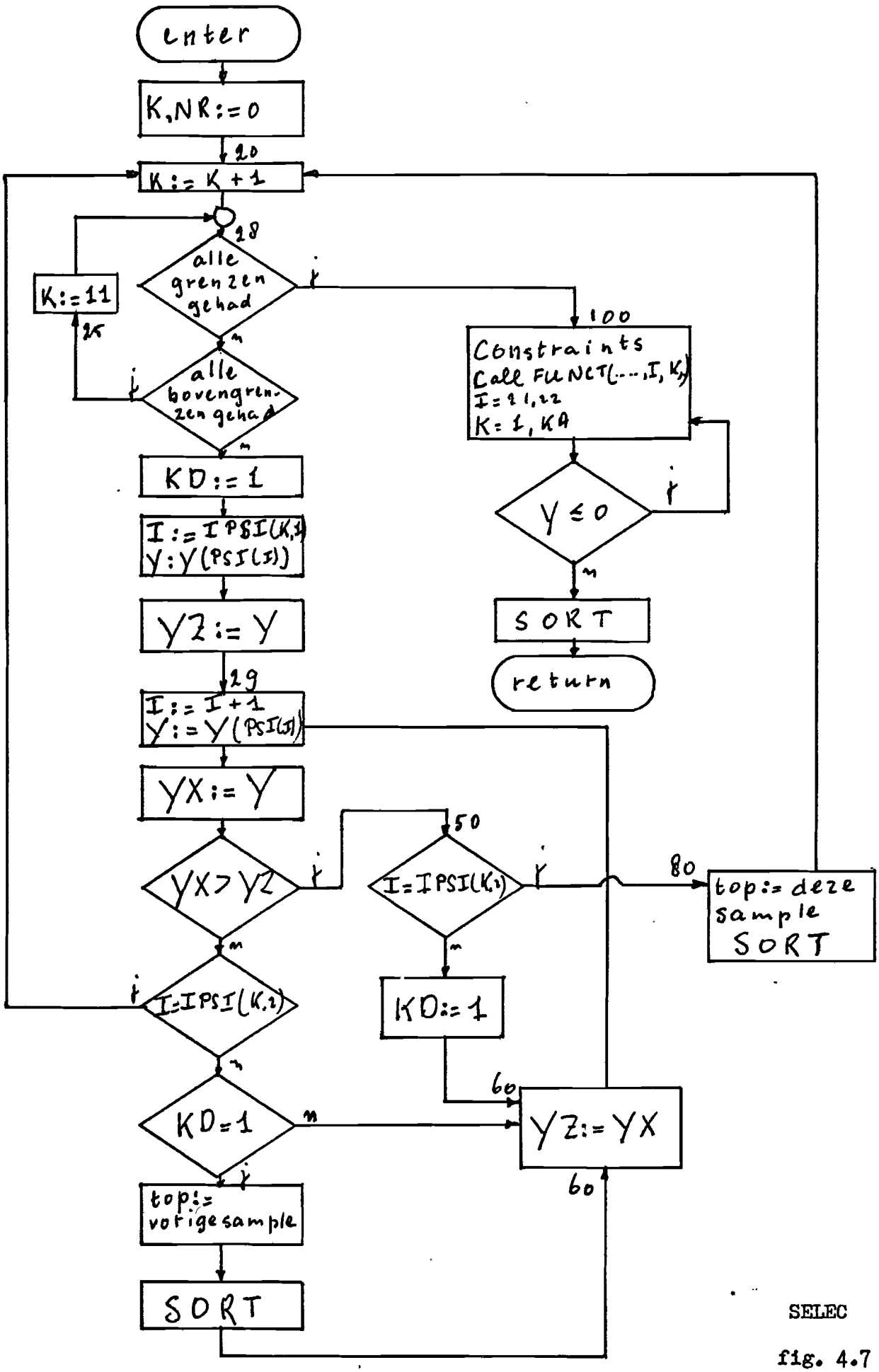
c. Als de functie stijgend was en het huidige punt is een eindpunt van van de grens (I = IPSI(K,2)) willen we dit punt ook als een top beschouwen (80).
Met twee voorbeelden kunnen we dit demonstreren. In fig. 4.6 a en b zijn twee foutfuncties gegeven.



a b fig.4.6

In fig. a is alleen punt 5 een top.
In fig. b de punten 1 en 12.
Telkens als er een top bijgekomen is, KR = KR + 1 wordt er gesorteerd met SORT. Als PSIM vol is (15 toppen) moeten er nog wel nieuwe toppen worden bepaald en gesorteerd, alleen moet dan de laagste afvallen.

2. De constraints.
De parametergrens- overschrijdingen worden in SELEC getest (100) en indien aanwezig (y>0) opgenomen en gesorteerd met de andere toppen.
Zoals reeds opgemerkt in hoofdst 4.3 en 4.6.1.


enter
K,NR:=0
20
K:=K+1
28
alle grenzen gehad
j
n
K:=11
24
alle bovengrenzen gehad
j
n
KD:=1
I:=IPSI(K,1)
Y:Y(PSI(I))
YZ:=Y
29
I:=I+1
Y:=Y(PSI(I))
YX:=Y
YX>YZ
j
50
I=IPSI(K,1)
j
80
top:=deze sample
SORT
n
I=IPSI(K,1)
j
n
KD:=1
60
KD=1
n
YZ:=YX
60
j
top:=vorige sample
SORT
100
Constraints
Call FUNCT(...,I,K)
I=11,22
K=1,KA
Y≤0
j
n
SORT
return

SELEC

fig. 4.7

4.6.3 LOCATE het bepalen van de objectfunctie: U

De objectfunctie wordt in dit programma bepaald overeen komstig W in (2.11), dus als het globale maximum over alle samplepunten (van functie + constraints).
Het algorithme is eenvoudig:
Alle frequentiepunten van PSI worden achtereenvolgens getest en de laatst grootste y_i wordt vastgehouden. Daarna (35) worden alle constraints op dezelfde manier getest (alleen y>0)

4.6.4 GRADIE het bepalen van de gradiënt

Deze routine roept FUNCT aan met DERIV = .TRUE. hierbij het onderscheid makend tussen functies en constraints (IYM(I)>20). Omdat ϕ_0 na SELEC en vóór GRADIE dezelfde is (ga maar na), en PHO in SELEC naar ZNO getransformeerd wordt, kan GRADIE werken met de in common (MAIN) opgenomen ZNO.

4.6.5 NORM

In NORM wordt de Euclidische norm van $\Delta\phi$ bepaald.

4.6.6 GOLDEN

De subroutine GOLDEN is letterlijk geïmplementeerd zoals door Bandler [3] gegeven.

4.6.7 SIMPLEX

Het simplexalgorithme is geïmplementeerd zoals in het collegedictaat Lineaire Programmering [6] is gegeven als tweefasen- methode.
Omdat het algorithme hier toch helemaal geprogrammeerd moest worden is het algorithme wat betreft het starttableau en de specifieke kostenvektor (zie [3]) aangepast. Het programma is dienovereenkomstig van commentaar voorzien.

4.7 MAIN het hoofdprogramma

In het hoofdprogramma wordt de in- en uitvoer (al- of niet grafisch) geregeld en wordt met OPTIM het optimaliseringsprogramma bestuurd.

a. De in- en uitvoer via het scherm van de T 4014 (unit number : 1) wordt in principe zo geregeld dat links de tekst verschijnt en rechts de grafische informatie en eventueel een schema.
b. Er kunnen ook data- files gebruikt worden voor in- en uitvoer.
c. Met het menu kan ingegrepen worden in het programmaverloop.

Het hoofdprogramma is ten alle tijde gemakkelijk te wijzigen of aan te vullen. In het menu wordt naar het verlangde programmaonderdeel gesprongen en na afloop weer teruggesprongen naar het menu.

4.7.1 Invoer via eigen terminal(1)

Deze invoer zal weinig moeilijkheden voor de gebruiker opleveren omdat alles met een write- opdracht gevraagd wordt. Het betreft hier vooral de variabele invoer.

4.7.2 Invoer via data- file (2)

Er is een datafile voor bv. de netwerkgegevens. Deze file kan via EDI gemaakt worden. Voor meer informatie wordt verwezen naar Apendix.

4.7.3 De DEBUG output

Als de betreffende programmafiles met een /DE worden vertaald (zie Apendix) wordt de desbetreffende debugoutput geschreven naar (3). Deze debugoutput is zodanig dat hiermee GRAZOR goed nagegaan kan worden.

4.7.4 PLOTH de overdrachtsfuntie.

Met deze subroutine wordt een grafiek van de overdrachtsfunctie gemaakt voor een opgegeven aantal samplepunten. Er wordt een rechte lijn tussen twee punten gemaakt. De assen zijn (nog) niet van een schaalverdeling voorzien. Te zijner tijd kan dit nog gebeuren, als in het rekencentrum hiervoor standaardroutines binnen GINO-F zijn gemaakt.

4.7.5 ERROR

Hiermee wordt een errorfunctie in grafiek gebracht van de laatste overdrachtsfunctie met de opgegeven specificaties.

4.7.6 BOUND

Met deze routines worden de specificaties uit de invoerfile (2) ingelezen.

4.7.7 BOUNDC

Hiermee worden de met BOUND ingelezen waarden overschreven. Vooral voor demonstratiedoelenden is het interessant dat de filterspecificaties met de cursor kunnen worden ingelezen.

4.7.8 CONSTR

Hiermee wordt de array CON ingelezen, dus de variabele parameters met grenzen en weegfactor.

4.7.9 DRAW

Met deze routine is het mogelijk om een netwerk te tekenen. Het ligt in de bedoeling (is op dit moment nog niet gerealiseerd) om als volgt te werk te gaan.

a. Het netwerk wordt met de hand (cursor) getekend.
b. Het getekende netwerk wordt in een file (2) geschreven.
c. Bij een volgende analyse van ditzelfde netwerk komt de tekening automatisch op het scherm.

4.7.10 OPTIM

Met deze routine kan GRAZOR bestuurd worden. Voorlopig is geprogrammeerd dat GRAZOR maximaal een op te geven aantal malen wordt aangeroepen. Het is hier ook denkbaar dat een ingewikkelder besturing van GRAZOR wordt geautomatiseerd bv. veranderen van ε en ε' het automatisch besturen van de weegfuncties in afhankelijkheid van de met GRAZOR verkregen U.

5. Resultaten en conclusies.

De belangrijkste conclusie die we t.a.v. het geïmplementeerde systeem kunnen trekken is, dat we een werkend systeem hebben verkregen waarmee langs interaktieve weg filters met een ladderstruktuur kunnen worden geoptimaliseerd.
Dit laatste dan wat betreft de overdrachtsfunctie t.o.v. opgegeven specificaties.
Op de minicomputer PDP-11/55 zijn voor het interaktief werken redelijke rekentijden te bereiken. Voor het systeem met de hier gegeven omvang ligt de rekentijd voor een complete optimalisering in de grootteorde van seconden tot enkele minuten.
Een belangrijk voordeel van het interaktieve aspekt en vooral van de direct beschikbare grafische informatie is bij het gebruik al een keer opgevallen.
Men kan hier nl. afhankelijk van de middels analyse verkregen grafiek direct een sampleraster opgeven.
Mocht dit raster te grof zijn en daarvoor bv. bij de optimalisering een top in de frequentiekarakterestiek over het hoofd zijn gezien, dan ziet men dit bij het resultaat in grafiekvorm direct.

Ofschoon nog geen systematisch onderzoek is gedaan naar de efficiëncy en betrouwbaarheid van de geïmplementeerde optimaliseringsmethode is de tot nu toe opgedane ervaring met een eenvoudig voorbeeld hoopgevend in die zin dat we een praktisch bruikbaar systeem hebben gerealiseerd.
Willen we dit laatste bevestigd krijgen dan zal een netwerkspecialist er mee moeten gaan werken.

6. Literatuurlijst

1. G.C. Temes,S.K. Mitra,"Modern Filter Theory and Design",Wiley 1973.
2. J.W. Bandler en P.A. Macdonald, "Optimization of microwave networks by razor search," IEEE Trans. Microwave Theory and Techniques, MTT-17,552-562 (aug. 1969).
3. J.W. Bandler, T.V. Srinivasan en C. Charalambous, "Minimax optimization of networks by grazor search", IEEE Trans. Microwave Theory and Techniques, MTT-20, 596-604 (sept 1972).
4. V.F. Nicola, "An approach to interactive optimization for ladder networks", Stageverslag groepES afd. Elektrotechniek TH-Eindhoven.
5. S.W. Director en R.A. Rohrer, "The generalized adjoint network and network sensitivities", IEEE Trans. on circuit theory, CT-16, 318-323 (1969).
6. J.F. Benders, "Lineaire programmering" college dictaat TH-Eindhoven, 2.202.
7. G.C. Temes, "Optimization methods in circuit design", Computer Oriented Cicuit Design, F.F. Kuo en W.G. Magnuson, Jr., Eds. Englewood Cliffs, N.J.: Pretice-Hall,1969.