Energiesystemen

Faculteit der Elektrotechniek

Vakgroep Elektrische Energiesystemen

Prakticumhandleiding bij het college

E N E R G I E S Y S T E M E N

Door: A.J.D. Lambert en P. Massee

Voorjaar 1986

T e c h n i s c h e U n i v e r s i t e i t E i n d h o v e n

KORTE INSTRUCTIE TEMPO VOOR ADM-TERMINALGEBRUIKERS

1. Gereedmaken van het systeem.

Een eenvoudige en duidelijke beschrijving van de ADM-terminal is te vinden in RC-informatie AG-45 ("Het gebruik van de ADM-terminal onder CANDE").

Voor inloggen en uitloggen is de procedure als volgt:

Geef een "sendline".
Er verschijnt een boodschap die eindigt met:
  ENTER USERCODE PLEASE
Tik in:
  <usercode>/<password>
Er volgt een boodschap in de trant van:
  # IT'S DAYTIME .....
  # ....
  # SESSION ......
Het systeem bevindt zich dan in de Cande mode, en we kunnen desgewenst Cande-commando's intikken.
Uitloggen kan geschieden met het Cande-commando:
  BYE
waarna (met een boodschap) uitgelogd wordt, dan wel gevraagd wordt:
  #SAVE OR REMOVE WORKFILE?
Hierna tikken we al naar believen in:
  SAVE
of:
  REMOVE
en nogmaals:
  BYE

Afbreken van een ongewenste langdurige bezigheid geschiedt met het commando:
  ?DS

Om met het l.p.-pakket Tempo te werken, moeten bij de ADM-terminal nog enkele voorbereidingen worden getroffen:

-Zet de "erase"-functie aan, opdat een regel wordt gewist alvorens ze overschreven wordt. Gebruik hiervoor het commando:
  ?+E
-Breng de terminal in de zogenaamde "hardcopy-mode" met behulp van het commando:
  TERM H
Doet men dat niet, dan werkt Tempo niet en geeft het pakket een formatterings-foutmelding.
-Het is in Tempo nodig om met hoofdletters te werken. Zorg daartoe dat het lampje van "caps lock" brandt.

2. Het Tempo-pakket.

Het Tempo-pakket is een voor de Burroughs geschreven applicatiepakket waarmee optimalisatieproblemen kunnen worden

uitgevoerd door een l.p.-algoritme (het zgn. "revised Simplex algorithm".)
Het pakket kan ook voor varianten op het l.p.-probleem worden gebruikt, zoals het duale probleem, integer problemen en niet-lineaire programmering ("separable programming"). Het biedt daartoe tal van procedures alsmede een commandotaal, de zogenaamde Tempo Command Language (TCL).
Het pakket kan worden aangewend in twee modes, de batch mode en de interactieve mode. De batch-mode, gebaseerd op het gebruik van ponskaarten, is verouderd en wordt niet meer behandeld.

Opstarten en verlaten van Tempo.

Vanuit Cande wordt Tempo opgestart met:

```
E$MPS/ALL ON APPL
```

Geven we dit commando dan verschijnt een boodschap die wordt afgesloten met:

```
READY
```

Deze prompt is kenmerkend voor de zgn. command-mode waarin Tempo zich nu bevindt. In deze mode kunnen TCL-commando's worden gegeven.
Dit kunnen aanroepen van Tempo-procedures zijn, waartoe de betreffende procedurenaam moet worden opgegeven. Deze kan eventueel nog worden gevolgd door een modifier, dat is een optie die aan de procedure wordt weergegeven. De modifier komt dan, tussen haakjes, na de procedurenaam. Dus:

```
<procedurenaam> of:
<procedurenaam> (<modifier>)
```

Van belang is allereerst de Tempo-procedure:

```
EXIT
```

waarmee Tempo verlaten wordt en teruggegaan wordt naar Cande.
Soms wordt Tempo ook automatisch verlaten namelijk, indien een fatale fout optreedt.

In de command-mode kunnen ook assignments worden gegeven. De uitvoering van Tempo wordt gecontroleerd door een groot aantal zgn. interne parameters. Deze worden aangegeven met een parameternaam, die begint met een Z. Niet alle identifiers die met een Z beginnen zijn overigens interne parameters. Het kunnen ook filenamen of zogenaamde demands zijn. Op de demands wordt nog teruggekomen.
Interne parameters kunnen zijn van een integer, real, Boolean, of tekst (alpha) type. Een groot aantal van deze parameters wordt automatisch door Tempo geinitialiseerd, sommige echter moeten we zelf initialiseren. Voorbeeld van een interne parameter van het type tekst is ZNAME. We initialiseren deze met:

```
ZNAME="<naam>"
```

Merk op dat de tekst tussen aanhalingstekens moet staan.
Een voorbeeld van een interne parameter van het type Boolean is ZPRINTER. Deze wordt door Tempo automatisch geinitialiseerd als .FALSE. We kunnen haar een andere waarde geven, bijvorbeeld:

```
ZPRINTER=.TRUE.
```

Voorts is het mogelijk om twee interne parameters van hetzelfde type aan elkaar gelijk te stellen, dus:

```
ZDATA=ZNAME
```

Hierbij mogen geen aanhalingstekens worden gebruikt.
Als Tempo verlaten wordt, worden alle initialisaties van interne parameters vernietigd, en bij hernieuwd opstarten krijgen deze parameters weer hun default waarde.
Indien de gebruiker tijdens een Tempo berekening wil bekijken welke waarde een bepaalde parameter heeft, gebruikt hij de Tempo-procedure DISPLAY.
Voorbeeld:

```
DISPLAY <parameternaam>
```

Met de procedure STATUS wordt de waarde van vrijwel alle interne parameters uitgeprint.
Met de procedure RESET herkrijgen alle parameters dezelfde waarde die ze kregen bij de aanroep van Tempo.

Invoer- en uitvoerfiles.

Om te begrijpen hoe het pakket werkt dienen we iets te weten van de organisatie van invoer- en uitvoergegevens.
De invoerfile bevat de gegevens omtrent het probleem. Deze file moet nog worden aangemaakt, ofwel ze bevindt zich reeds op de disk onder de naam <filenaam> als binaire DATA-file.
Met behulp van de Tempo-procedure INPUT wordt de invoerfile op een verzamelfile geschreven. Deze file heeft de naam ZPROF en ieder probleem wordt achteraan op de file bijgschreven. ZPROF wordt bewaard op disk. Staat een probleem er eenmaal op dan hoeft INPUT voor dat probleem niet nogmaals te worden aangeroepen. INPUT doet nog meer. Ze haalt de invoerfile ook op van disk, terwijl het in de interactieve mode mogelijk is om de invoerfile aan te maken respectievelijk te wijzigen.
De procedure INPUT werkt alleen als een tweetal interne parameters een waarde krijgt toegekend. Dit zijn ZNAME en ZDATA. De reden daarvoor is dat zowel het probleem als de dataset (oorspronkelijk: een pak ponskaarten) van een naam worden voorzien en onder deze naam op ZPROF worden geschreven en er weer van worden gelezen.
Als we Tempo opgestart hebben, en de file ZPROF is nog niet aanwezig dan wel het te behandelen probleem bevindt zich niet op ZPROF, dan worden de volgende commando's gegeven:

```
ZNAME="<naam>"
ZDATA="<dataset>"
INPUT(REMOTE)
```

De modifier REMOTE opteert voor de interactieve mode, met DISK als modifier zouden we in de batch-mode terecht komen. De prompt van de interactieve mode is het procent-teken:

```
%
```

De interactieve mode wordt verlaten door de commando's ENDATA of STOP die verderop zullen worden behandeld.

Tenslotte nog een opmerking over de uitvoer. De invoerfiles en de file ZPROF worden in binaire vorm op disk bewaard doch zijn onleesbaar vanuit Cande. Een leesbare file wordt verkregen met behulp van bepaalde Tempo-procedures, voorzien van de modifier

DISK. Betreffende file wordt dan bijgeschreven op de verzamelfile DISKOUT.
Van belang zijn de procedures:

BCDOUT(DISK) en
OUTPUT(DISK)

BCDOUT(DISK) schrijft een probleem dat onder naam ZNAME en dataset ZDATA op ZPROF staat, in leesbare vorm op de file DISKOUT. Het betreft de invoergegevens.
OUTPUT(DISK) schrijft de gegevens betreffende de uitvoer van de berekening op DISKOUT. Evenals bij ZPROF het geval is, wordt op DISKOUT iedere nieuwe file achter de reeds bestaande gegevens gevoegd.

Het statement $FILE

Met het statement $FILE is het mogelijk om Tempo-files een door ons gewenste naam te geven. We hebben er reeds op gewezen dat op de file DISKOUT voortdurend uitvoergegevens worden bijgeschreven. Met de statement:

$FILE DISKOUT=<filenaam>

wordt de uitvoer naar file <filenaam> geschreven. Hetzelfde geldt voor een aantal andere Tempo-files, we noemen:

DISKIN
ZPROF
OLDZPROF
MACROLIB

Dit zijn de zogenaamde "interne filenamen" die door het pakket worden toegekend. Met het $FILE statement kan de gebruiker daar dus een eigen naam aan toekennen. DISKIN is de standaard invoerfile voor Tempo. Zij is vereist bij de batch-mode. Daar de invoerfiles in het algemeen onder een bepaalde <filenaam> staan vermeld, moet deze via een $FILE statement worden gedeclareerd. ZPROF is de problemfile, die in hoofdstuk 2 werd behandeld. OLDZPROF wordt gebruikt bij het copieren van problemen van de ene problemfile naar de andere. MACROLIB komt in hoofdstuk 6 ter sprake.

3. Het ontwerpen van Tempo-files

Het l.p.-probleem

We gaan uit van paragraaf 1.1.2. van het dictaat en beschouwen een probleem met m=3 en n=2. We gaan uit van dezelfde notatie als in dit dictaat en schrijven het vraagstuk als volgt uit:

$$J = c_1 x_1 + c_2 x_2 \qquad \text{vgl.(1.8)}$$
$$z = q_{11} x_1 + q_{12} x_2 >= \hat{z}_1$$
$$z = q_{21} x_1 + q_{22} x_2 >= \hat{z}_2 \qquad \text{vgl.(1.5) en (1.7)}$$
$$z = q_{31} x_1 + q_{32} x_2 >= \hat{z}_3$$

We zien hier een toegestane modificatie t.o.v. (1.7), waar een <= teken voorkwam.
Er kunnen nog meer uitbreidingen op het grondprobleem worden aangebracht.
Een aantal ingangsvariabelen bijvoorbeeld kan een bereik (range) hebben, dat tussen twee grenzen ligt. De ene grens is reeds in bovenstaand probleem meegegeven, hier: een

benedengrens. De andere grens moet nog worden ingevoerd. Stel dat voor z_3 geldt:

$$z_3^{\ell} <= z_3 <= z_3^{u}$$

dan wordt als RANGE opgegeven de grootte van het interval:

$$|z_3^{u} - z_3^{\ell}|$$

De uitgangsvariabelen zijn, indien niet anders is aangegeven, aan de niet-negativiteitsvoorwaarde (1.6) gebonden. Ook strengere voorwaarden zijn mogelijk. Zo is het denkbaar dat x begrensd wordt door een bovengrens (upper BOUND) en een benedengrens (lower bound), dus:

$$x_1^{\ell} <= x_1 <= x_1^{u}$$

In dit geval moeten zowel de boven- als de benedengrens worden ingevoerd.

We merken op dat het probleem uitgeschreven is in (m+1) rijen en n kolommen, waarbij de rijen overeenkomen met de objectfunctie en de ingangsvariabelen, en de kolommen met de uitgangsvariabelen.

De Tempo-file.

In de Tempo-file krijgt iedere rij en iedere kolom een naam. Een naam kan hooguit uit acht karakters bestaan. In de Tempo-file zijn ook indicatoren opgenomen (NAME, ROWS, COLUMNS, RHS, RANGES, BOUNDS en ENDATA). Deze indicatoren geven de secties aan, waarin de Tempo-file is verdeeld.
De NAME-sectie bevat de <dataset>.
De ROWS sectie bevat het type van de betreffende rij en de naam van de betreffende rij. Een van de rijen is de objectfunctie. De volgende codes worden voor het type gebruikt:

- N geen begrenzing (objectfunctie of meerekenfunctie).
- G begrenzing >=
- L begrenzing <=
- E gelijkheid

De COLUMNS-sectie bevat een aantal reeksen die overeenkomen met de elementen van de matrix die gevormd wordt door het l.p.-probleem, waarbij de volgorde is:
Eerst de kolomnaam, dan de rijnaam, vervolgens de waarde van de coefficient.
De RHS-sectie bevat de waarden van de rechterleden in zoverre deze niet van nul afwijken. Ook zij bevat een aantal reeksen, bestaande uit de naam van de RHS-sectie, de naam van de bijbehorende rij, de waarde van het rechterlid of nevenvoorwaarde. In het probleem dat als illustratie behandeld wordt, is de nevenvoorwaarde gegeven door $\hat{z}_i$.
De nu volgende RANGES en BOUNDS secties zijn slechts noodzakelijk als het probleem ook werkelijk ranges en bounds heeft.
De RANGES-sectie bestaat uit: Naam van de RANGES-set, rij waarop de range betrekking heeft, absolute waarde van het interval.
De BOUNDS-sectie bestaat uit: Type van de bound, kolom waarop de bound betrekking heeft, grootte van de begrenzing. Omdat voor de uitgangsvariabelen in de RHS-sectie geen begrenzing is opgegeven, moeten in het algemeen twee regels worden ingevoerd, steeds vooraf gegaan door de naam van de betreffende variabele. Dit moet op verschillende regels gebeuren, terwijl iedere

begrenzing ook van een code moet worden voorzien.
De volgende codes zijn mogelijk:

UP	bovengrens (upper bound)
LO	benedengrens (lower bound)
FX	vaste waarde
MI	geen benedengrens ($-\infty$)
PL	geen bovengrens ($+\infty$)

Hiermee is ook gegeneraliseerd naar gevallen waarin de niet-negativiteitsvoorwaarde niet zonder meer hoeft te worden aangehouden.
Tenslotte wordt de file afgesloten met ENDATA.

4. Interactief werken met Tempo.

Aanmaken van Tempo-files.

In hoofdstuk 2 hebben we gezien dat via de opdracht INPUT(REMOTE) in de interactieve mode kon worden gekomen. In deze mode is de prompt een %-teken. In de interactieve mode kunnen invoergegevens worden ingetikt dan wel bepaalde speciale commando's worden gegeven, welke geen TCL-commando's zijn. We merkten op dat INPUT eerst dan kon worden aangeroepen als aan ZNAME en ZDATA een naam werd toegekend.
De commando's voor de interactieve mode (de zgn. speciale commando's) zijn:

```
CONT/<filenaam>
SAVE/<filenaam>
LIST
STOP
ENDATA
```

Hiernaast zijn nog editing commando's beschikbaar, namelijk:

```
REMOVE
INSERT
```

Stel dat er nog geen file beschikbaar is van het in hoofdstuk 3 besproken l.p.-probleem. Het is mogelijk om in de interactieve mode deze file aan te maken. Reeds nu is de file niet leeg, en dat kan gecontroleerd worden door gebruik te maken van het commando:

```
LIST
```

een commando dat op ieder ogenblik gebruikt kan worden in de interactieve mode. We zien dat reeds de volgende file is gegenereerd:

```
NAME        <dataset>
ENDATA
```

Vervolgens kunnen de gegevens voor de diverse secties worden ingevoerd. Hierbij moet steeds de sectienaam worden opgegeven, opdat de computer weet hoe de betreffende invoer moet worden geinterpreteerd. Na iedere regelinvoer geeft de computer het %-prompt.

We geven een voorbeeld van een Tempo-sessie waarbij de file MODEL wordt aangemaakt:

```
E$MPS/ALL ON APPL
ZNAME="NAAM"
ZDATA=ZNAME
INPUT(REMOTE)
```

Nu volgt de listing van de invoer voor het in hoofdstuk drie behandelde probleem. Het symbool * staat voor een herhaling van dezelfde identifier die reeds in de voorgaande regel is gebruikt. Het symbool / dient als separator.

```
ROWS
N/J
G/Z1
G/Z2
G/Z3
COLUMNS
X1/J/8.0
*/Z1/100.0
*/Z2/2.0
*/Z3/1.0
X2/J/140.0
*/Z1/1000.0
*/Z2/300.0
*/Z3/2.0
RHS
RHS1/Z1/50000.0
*/Z2/1500.0
*/Z3/50.0
RANGES
RNG/Z3/150.0
BOUNDS
UP/BND/X1/60.0
LO/*/X1/40.0
```

Een listing (met LIST) geeft eventueel de gewenste, geformatteerde file.
De input-mode wordt afgesloten met:

```
SAVE/MODEL
ENDATA
```

De laatste twee opdrachten worden nu verklaard:
In het algemeen moet de nieuwe file op disk worden opgeslagen. Wensen we deze onder de naam <filenaam> op te slaan dan geven we het speciale commando:

```
SAVE/<filenaam>
```

Nu is op disk een invoerfile gecreeerd.

De input-mode kan op twee wijzen worden verlaten:
Met het commando:

```
STOP
```

komt men in de command-mode (prompt READY) zonder dat de geredigeerde file op ZPROF wordt opgeslagen.
Met het commando:

```
ENDATA
```

komt men eveneens in de command-mode. De geredigeerde file is nu echter wel op ZPROF opgeslagen.

Ophalen en wijzigen van Tempo-files.

Hoewel de invoerfile via SAVE onder de naam <filenaam> op disk is bewaard, kan het noodzakelijk zijn deze ter wijziging op te halen dan wel opnieuw op ZPROF in te lezen.
Direct nadat we via INPUT(REMOTE) in de interactieve mode terecht zijn gekomen, kunnen we zo'n file ophalen via het speciale commando:

```
CONT/<filenaam>
```

Betreffende file is nu beschikbaar. Ze kan na wijziging onder dezelfde naam of onder een andere naam worden weggeschreven, maar de oorspronkelijke file wordt altijd overschreven. Pas daarmee op! In kritieke gevallen is het beter om in Cande eerst een copie te maken onder een andere naam.
Bij het wijzigen moet er op gelet worden dat de juiste sectie is gedeclareerd, anders wordt de invoer onjuist geinterpreteerd hetgeen meestal in een foutmelding resulteert. Foutmeldingen zijn opmerkingen als:

```
Section type required.
Incorrect number of input fields.
Row type incorrect.
```

De secties zijn: ROWS, COLUMNS, RHS, RANGES, BOUNDS

De volgende wijzigingen kunnen worden doorgevoerd:
Overschrijven van regels. Dit kan ook al op het ogenblik dat de file voor de eerste keer wordt ingevoerd. Als bijvoorbeeld in vergelijking Z1 het type G door E moet worden vervangen, wordt ingetikt:

```
ROWS
E/Z1
```

Een ander commando dat gegeven kan worden is: REMOVE, bijvoorbeeld:

```
REMOVE/Z3
```

Als we na het geven van dit commando listen zien we dat Z3 in alle secties verwijderd wordt. In het voorbeeld verdwijnt de RANGES-sectie, die slechts op Z3 betrekking heeft, zelfs in haar geheel.
Van belang is het snel wijzigen van coefficienten, bijvoorbeeld:

```
COLUMNS
X2/Z2/200.0
```

hiermee wordt de coefficient 300 in 200 veranderd in het voorbeeld.

Dan is er nog de opdracht INSERT. Stel we wensen nog een rij Z0 in te voeren. Die wordt, als we de ROWS-sectie aangegeven hebben, achteraan in de ROWS-sectie geplakt. Als we die echter vooraan willen hebben (of op een andere willekeurige plaats in de sectie), gebruiken we INSERT. In het voorbeeld:

```
ROWS
INSERT/Z1
E/Z0
```

De listing laat zien dat de regel E/Z0 nu voor de regel die Z1 bevat wordt geplaatst.
We merken op dat in de BOUNDS-sectie slechts dan gewijzigd kan worden als de betreffende bounds-set eerst in haar geheel is verwijderd. In ons voorbeeld:

```
BOUNDS
REMOVE/BND/X1
```

Indien de wijzigingen zijn ingevoerd kan desgewenst nog een controlerende listing worden uitgevoerd. Het SAVE-commando wordt gegeven:

```
SAVE/<filenaam>
```

Vervolgens kan de interactieve mode worden verlaten met:

```
ENDATA
```

De gewijzigde file wordt nu weggeschreven naar ZPROF onder naam <naam> en <dataset> en naar disk onder naam <filenaam>.

5. Rekenen met Tempo.

Een nieuwe procedure wordt nu voorbereid en wel SETUP.
We merken op dat met Tempo ook vraagstukken van de algemene vorm kunnen worden behandeld (zie dictaat par.1.1.5). Het is daarbij mogelijk dat de uitgangsvariabelen beperkt zijn door een boven- en een ondergrens (upper en lower bound). In de invoerfile krijgt de reeks begrenzingen een naam die aan een interne parameter moet worden meegegeven, dit is de parameter boundset:

```
ZBNDST="<boundnaam>"
```

Ook is het denkbaar dat zekere ingangsvariabelen aan een bereik (range) gebonden zijn. Indien dat het geval is krijgt ook de range een naam en wordt de parameter rangeset daaraan gelijkgesteld:

```
ZRNGST="<rangenaam>"
```

In de meest elementaire l.p.-vraagstukken komen geen ranges of bounds voor (behalve de bovengrens van de ingangsvariabelen die altijd aanwezig is). In dat geval hoeven de bounds en ranges ook niet verder gedeclareerd te worden, en hoeven aan ZBNDST en ZRNGST geen waarden te worden toegekend.

De procedure SETUP, die vervolgens wordt aangeroepen, leest het probleem met de naam <naam> en <dataset> van ZPROF in. Voorts reserveert. zij geheugen (allocatie). Hoogstens een boundset en een rangeset kunnen worden ingevoerd.
SETUP creeert de initiele oplossing (eerste tableau van dictaat paragraaf 1.1.4). Het is mogelijk dat dit een niet-toegestane oplossing is.
We dienen aan te geven of het om een maximalisatie dan wel een minimalisatieprobleem gaat. Hiertoe dienen de modifiers MAX en MIN. Indien geen modifier wordt meegegeven wordt er automatisch vanuit gegaan dat het om een minimalisatieprobleem gaat. Dus, bijvoorbeeld:

```
SETUP(MAX)
```

Nu komt het eigenlijke rekenwerk: De toepassing van het zogenaamde "herziene simplex-algoritme".
Ter voorbereiding moet nog aan enkele parameters een waarde worden toegekend: De naam van de rechterhelft van de vergelijkingen en de naam van de objectfunctie. Deze namen zijn in de invoerfile opgegeven en we zeggen nu:

```
ZRHS="<naam rechterhelft>"
ZOBJ="<naam objectfunctie>"
```

Vervolgens wordt de rekenprocedure PRIMAL aangeroepen en wordt het simplex-algoritme uitgevoerd.

```
PRIMAL
```

Eerst wordt een toegestane (feasible) oplossing gezocht, waarna wordt geoptimaliseerd. Diverse gegevens over de rekenstappen verschijnen op het scherm. Uiteindelijk wordt dan een optimale oplossing gevonden, tenzij die om een of andere reden niet bestaat. In het laatste geval wordt een boodschap gegeven. Deze is in de trant: "unbounded solution".

Tenslotte wordt de procedure OUTPUT aangeroepen:

```
OUTPUT
```

en een overzicht van het probleem, en de oplossing, verschijnt op het scherm. Aan de procedure-aanroep OUTPUT kan de modifier DISK worden meegegeven:

```
OUTPUT(DISK)
```

In dit geval wordt de output op disk bijgeschreven in de file DISKOUT.

Met EXIT wordt Tempo verlaten.

Een voorbeeld van een TCL-programma dat de bestaande file MODEL op ZPROF schrijft, een minimalisatieprobleem uitvoert en het resultaat naar DISKOUT schrijft:

```
E$MPS/ALL ON APPL
ZNAME="NAAM"
ZDATA=ZNAME
ZOBJ="J"
ZRHS="RHS1"
ZRNGST="RNG"
ZBNDST="BND"
INPUT(REMOTE)
CONT/MODEL
ENDATA
SETUP
PRIMAL
OUTPUT(DISK)
EXIT
```

6. Uitvoer van gegevens.

<u>Uitvoer van files op disk via printer.</u>

Tempo maakt een uitvoerfile aan, DISKOUT genaamd, die desgewenst uitgeprint kan worden. Bepaalde files worden indien gewenst achter elkaar op deze file opgeslagen.
Op DISKOUT kan een leesbare vorm van de invoerfile komen te staan, die via de procedure:

```
BCDOUT(DISK)
```

wordt verkregen. Indien de modifier wordt weggelaten dan verschijnt de uitvoer slechts op het scherm.
Ook uitvoergegevens kunnen naar DISKOUT worden toegeschreven via:

```
OUTPUT(DISK)
```

Uitprinten van een willekeurige file in de directory geschiedt met:

```
U$SERVICE/PRINT ON APPL <filenaam>
```

Uitgebreide uitvoer via printer.

Een andere wijze om geprinte uitvoer te krijgen kan worden ingeschakeld met de Tempo-parameter ZPRINTER. Dit is een Boolean. Ze kan de waarden .FALSE. of .TRUE. aannemen.
Via het commando:

```
ZPRINTER=.TRUE.
```

wordt deze optie aangezet en de schermuitvoer wordt vanaf dit tijdstip opgeslagen in een backup-file. Deze is onzichtbaar in de Cande-listing, doch ze kan worden afgedrukt met:

```
U$SERVICE/BACKUP ON APPL
```

Als dit programma zich gemeld heeft kan een commando worden ingetikt. Met:

```
HELP
```

krijgen we een lijst van de beschikbare commando's. Met:

```
PRINT
```

wordt de backup-file op de printer afgedrukt. Met:

```
STOP
```

kan ten allen tijde teruggekeerd worden naar Cande.

Tijdens de Tempo-sessie kan de optie ZPRINTER steeds naar wens worden aan- en uitgezet. Op deze wijze kan de uitvoer worden geselecteerd. Naast de optie ZPRINTER bestaat ook de optie ZCONSOLE, die de schermuitvoer regelt. ZCONSOLE wordt als .TRUE. geinitialiseerd, doch het kan handig zijn deze optie uit te zetten. Bij grotere berekeningen is het namelijk veel sneller om de uitvoer op disk te zetten zonder deze over het scherm te laten gaan. Met:

```
ZCONSOLE=.FALSE.
```

kan dit worden gerealiseerd. Ook ZCONSOLE kan op ieder ogenblik worden omgeschakeld. Bij bepaalde procedures kan de schermuitvoer worden onderdrukt met een modifier. Bijvoorbeeld:

```
BCDOUT(DISK,NOPRINT)
OUTPUT(DISK,NOPRINT)
```

De modifier SUMMARY geeft extra informatie bij de uitvoer van bepaalde procedures. Bijvoorbeeld:

```
INPUT(REMOTE,SUMMARY)
SETUP(SUMMARY)
```

Ook deze extra informatie kan worden uitgevoerd naar de printerpoort door ZPRINTER aan te zetten.

7. Programmeren in TCL.

7.1. Macro's.

In een TCL-programma kunnen macro's worden gedefinieerd. Een macro is een blok commando's dat met een -door de gebruiker bepaald- commando kan worden uitgevoerd. Het gaat hierbij om een reeks commando's die herhaaldelijk moet worden gebruikt.
Stel we wensen een macro te definieren. Dan voeren we in:

```
MACRO <macronaam> RETAIN
<TCL-statements>
ENDMACRO
```

De tekst RETAIN draagt er zorg voor dat de macro op de disk wordt toegevoegd aan de geformatteerde file MACROLIB, die vele macro's kan bevatten. Deze file kan vanuit Cande eenvoudigweg worden gelist. Ze is van het type SEQDATA. Met een $FILE statement kan aan deze file een willekeurige naam woden toegekend.
De macro kan in een TCL-programma worden aangeroepen met:

```
<macronaam>
```

doch dan dient het TCL-programma te worden voorafgegaan door de declaratie:

```
MACRO <macronaam> RESTORE
```

waarmee deze macro vanuit de bibliotheek ter beschikking komt.
Als voorbeeld van een macro wordt een gedeelte van het TCL-programma uit hoofdstuk 5 gedefinieerd:

```
MACRO INIT RETAIN
ZNAME="NAAM"
ZDATA=ZNAME
ZRHS="RHS1"
ZOBJ="J"
ZBNDST="BND"
ZRNGST="RNG"
INPUT(REMOTE)
ENDMACRO
```

Indien deze macro eenmaal in de bibliotheek is opgenomen kan het TCL-programma van hoofdstuk 5 als volgt worden gegeven, voor een maximalisatieprobleem:

```
MACRO SOLVE RESTORE
INIT
 CONT/MODEL
```

```
 ENDATA
SETUP(MAX)
PRIMAL
OUTPUT(DISK)
EXIT
```

Het is duidelijk dat bij herhaaldelijk wijzigen van de inputfile het gebruik van macro's zeer aantrekkelijk zal zijn.
Let op: op het ogenblik dat de macro door de gebruiker wordt gedefinieerd reageert Tempo gewoon met de READY-prompt. Dat is eigenlijk niet zo handig, daar we ons eigenlijk in een andere mode bevinden. en de TCL-commando's niet worden uitgevoerd, doch als commando in de macro worden opgenomen. Deze speciale macro-edit mode wordt eerst door ENDMACRO verlaten, waarna we weer in de command-mode terecht komen. TCL-commando's worden dan weer als echte commando's geinterpreteerd en uitgevoerd.

7.2. Interrupts of demands, en labels.

Interrupts of demands zijn voorwaarden waaronder de normale volgorde in de uitvoering van het programma wordt gewijzigd.
Onder bepaalde voorwaarden worden de demands geactiveerd (geset). Dit kan bijvoorbeeld het geval zijn als er een fout wordt geconstateerd maar ook kan zulks opzettelijk geschieden door de gebruiker.
De demands worden evenals de interne parameters aangegeven door een variabele die met een Z begint. Zeer bekende zijn de demands ZMAJERR en ZMINERR (fatale resp. geringe fout). ZDONFS wordt geactiveerd als het probleem niet oplosbaar is (No Feasible Solution), bijvoorbeeld als de opgegeven range-set of het toegestane gebied niet bestaat. ZDOUNB wordt geactiveerd als de oplossing onbegrensd is.

Als de demand geactiveerd is kan bijvoorbeeld, in plaats van het normale programmaverloop, een foutmelding worden afgedrukt, het programma worden afgebroken enz.
Het is echter ook mogelijk om in het verloop van de programmatekst, doch ALLEEN in een macro, aan de demand een label toe te kennen. Dit moet dan geschieden voordat de demand geactiveerd wordt.
Het toekennen van een label geschiedt met de macro:

```
<demandnaam>=<labelnaam>
```

Als dan de demand geactiveerd wordt, zullen niet de standaard bij de demand behorende procedures worden afgewerkt, doch wordt naar het label gesprongen.
Een label wordt als volgt in de programmatekst opgenomen:

```
<labelnaam>:<TCL-statements>
```

Vanaf het label worden nu de statements successievelijk uitgevoerd. Dit gaat door totdat een der volgende statements wordt aangetroffen:

```
EXIT
```

Hiermee wordt Tempo verlaten.

```
RETURN
```

Er wordt teruggekeerd naar het statement waar voorheen de betreffende demand werd geactiveerd.

```
NEXT
```

Er wordt teruggekeerd naar het statement dat volgt op het statement waar voorheen de betreffende demand werd geactiveerd.

Voorbeeld:
Beschouw de macro SOLVE.

```
MACRO SOLVE RETAIN
ZDOUNB=LAB
SETUP(MAX)
PRIMAL
OUTPUT
EXIT
LAB: SETUP
ZDOUNB=FINAL
RETURN
FINAL: EXIT
ENDMACRO
```

We hebben hebben hier de mogelijkheid ingebouwd om automatisch over te gaan op een minimalisatieprobleem indien het maximalisatieprobleem een onbegrensde oplossing oplevert.
Eerst wordt voorgeschreven dat activering van de demand ZDOUNB niet meer de standaard-implementatie oplevert (de berekening staken) doch betekent dat naar het label wordt gesprongen. Vervolgens wordt de setup voor het maximalisatieprobleem gemaakt. Met PRIMAL wordt gerekend. Is nu de oplossing onbegrensd dan wordt ZDOUNB geactiveerd, er wordt naar de label LAB gesprongen en de setup voor het minimalisatieprobleem wordt daar gemaakt. Mocht er nog steeds een onbegrensde oplossing zijn dan mag het programma worden verlaten. Met RETURN springt de uitvoer terug naar PRIMAL waarna, als er nu een begrensde oplossing is, het programma van de macro gewoon wordt uitgevoerd.

Er bestaan nog een aantal commando's die ons opzettelijk naar labels kunnen doen springen. Dit zijn:

```
GO TO <labelnaam>
```

en

```
PERFORM (<labelnaam>)
```

Indien PERFORM wordt uitgevoerd gebeurt dat in combinatie met RETURN, NEXT of EXIT, die ergens na de label voorkomen. Indien RETURN wordt aangetroffen wordt PERFORM opnieuw uitgevoerd.
Tenslotte is nog mogelijk:

```
IF (<Boolean uitdrukking>) <statement>
```

Deze draagt zorg voor een voorwaardelijke uitvoering van het statement.
In de Boolean uitdrukkingen kunnen de volgende operatoren worden benut:

```
.AND.
.OR.
.NOT.
.EQ.      (is gelijk)
.NE.      (is ongelijk)
.GT.      (groter dan)
.LT.      (kleiner dan)
```

.GE. (groter dan of gelijk aan)
.LE. (kleiner dan of gelijk aan)
.TRUE.
.FALSE.

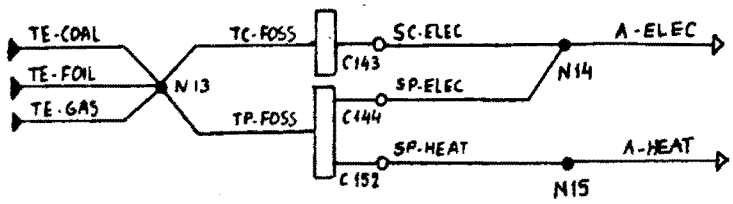
Voorbeeld:

```
IF (.NOT.ZPRINTER) GO TO LAB
```

8. HET OPSTELLEN VAN EEN GROTER MODEL.

8.1. Algemeen.

Modellen zoals die in hoofdstuk 3 van het dictaat zijn behandeld, zijn op betrekkelijk eenvoudige wijze om te zetten in een l.p.-probleem.
Beschouwen we als voorbeeld het volgende deelprobleem:



Het probleem beschrijft elektriciteitsopwekking door middel van twee typen elektrische centrales: De conventionele thermische centrale en de warmte/krachtcentrale. We onderscheiden drie knooppunten, drie procescoefficienten en tien variabelen. Dit levert drie knooppunts- en drie coefficientenvergelijkingen op die we naar de betreffende knooppunten en coefficienten noemen, zoals:

```
N13:  TC-FOSS + TP-FOSS = TE-COAL + TE-FOIL + TE-GAS
                                                        (1)
C144: SP-ELEC = c144 * TP-FOSS
```

We stellen deze vergelijkingen nu zodanig op dat ze passen in het kader van een l.p.-probleem. Bijvoorbeeld:

```
TC-FOSS + TP-FOSS - TE-COAL - TE-FOIL - TE-GAS = 0      (2)
```

Hierbij dient er zorg voor gedragen te worden dat de coefficienten van het juiste teken worden voorzien. Maak hieromtrent een eenduidige afspraak bijvoorbeeld: Van het punt uitgaande stromen positief, naar het punt toevloeiende stromen negatief.
Invoer in Tempo geschiedt via de interactieve mode. Iedere vergelijking wordt in de ROWS-sectie ingevoerd, iedere variabele in de COLUMNS-sectie.
We voeren dit uit voor C144 met voor de coefficient: c144=0.28

```
ROWS
.
.
E/C144
.
.
COLUMNS
.
```

```
.
SP-ELEC/C144/1.0
TP-FOSS/C144/-0.28
.
```

Het rechterlid van (2) is gelijk aan nul en dat hoeft niet in de RHS-sectie te worden ingevoerd aangezien daar de default-waarde voor het rechterlid gelijk is aan nul.
In de praktijk zetten we eerst het gehele stelsel vergelijkingen op en vullen dan achtereenvolgens de secties met de gewenste gegevens. Ook dient een objectfunctie te worden gedefinieerd, die afhankelijk is van de wens van de gebruiker van het betreffende model. Het kan daarbij bijvoorbeeld gaan om maximalisatie van de output:

```
J = A-ELEC + A-HEAT                                    (3)
```

Ook kan het gaan om minimalisatie van de input, maximalisatie van een zekere opbrengst (in geld) of minimalisatie van de kosten. Bij kosten problemen speelt ook de in de diverse processen opgestelde capaciteit een rol in de berekening, zij verschijnt namelijk als een vaste-kostenpost.
Voor het eenvoudige geval (3) moet worden ingevoerd:

```
ROWS
N/J
COLUMNS
A-ELEC/J/1.0
A-HEAT/J/1.0
```

Het probleem is tot nu toe nog steeds niet realistisch aangezien alle variabelen naar boven toe onbegrensd zijn. Het maximalisatieprobleem heeft dan geen oplossing, terwijl de oplossing van het minimalisatieprobleem triviaal, namelijk nul, is.
Er dient daarom een realistisch, samenhangend en niet-tegenstrijdig stelsel beperkingen aan een aantal variabelen te worden opgelegd. Deze beperkingen kunnen van de vorm zijn:

Beperkt aanbod, bijvoorbeeld:

```
G1: TE-COAL <= g1                                      (4)
```

Beperkte capaciteit, bijvoorbeeld:

```
G5: SP-ELEC <= g5
```

Aan een bepaalde vraag moet worden voldaan:

```
D1: A-ELEC  >= d1
```

Aan een bepaalde verhouding (allocatiecoefficient) dient te worden voldaan:

```
A1: A-ELEC  >= a1 * A-HEAT
```

Uiteraard zijn ook intervallen in rijgrootheden (RANGES) en kolomgrootheden (BOUNDS) denkbaar.

Met g1=600 wordt G1 als volgt ingevoerd:

```
ROWS
L/G1
COLUMNS
```

```
TE-COAL/G1/1.0
RHS
RHS1/G1/600.0
```

U kunt met het eenvoudige voorbeeld oefenen door wijzigingen aan te brengen of het model uit te breiden.

8.2. Werken met een bestaand model.

Het conventionele deel van het model-Boonekamp is ingevoerd in het computerbestand onder de naam ZNAME=ZDATA="NEDMOD1".
Copieer het naar uw directory. Er zijn zodanige begrenzingen ingevoerd dat een situatie is gesimuleerd zoals die in 1974 bij benadering was aan te treffen. Met behulp van BCDOUT(DISK) kunt u de coefficienten van het model opvragen. Tracht deze uitvoer te interpreteren, waarbij de diagrammen uit het dictaat (fig. 3.2. t/m 3.8) van nut zijn. De energiestromen zijn in 10 n kcal/yr weergegeven.
(10 kcal/yr = 4,18 PJ/yr)

Verwijder vervolgens G1 t/m G10 (die de ingangsgrootheden vastleggen) en D1 t/m D27 (die hoofdzakelijk betrekking hebben op de eindvraag). Verwijderen van G1 gaat bijvoorbeeld met:

```
ROWS
REMOVE/G1
```

in de interactieve mode (met % als prompt).
Alle rijen, kolommen enz. die G1 bevatten worden dan verwijderd. U kunt vervolgens aan de hand van fig. 3.10 enz. een nieuwe realistische set van beperkingen invoeren, bijvoorbeeld die voor 1984. Ook kunt u experimenteren met andere objectfuncties (kijk welke objectfunctie in NEDMOD1 is genomen).
Let bij SETUP op of u al dan niet (MAX) moet toevoegen.
Bekijk ook de schermuitvoer van PRIMAL waarop naast allerlei iteratiegegevens, ook procedurenamen als PRESOLVE, CRASH, BALANCE, CREATE en INVERT verschijnen. Deze Tempo-procedures brengen de matrices in een optimale vorm. Bedenk dan de matrix zelfs in het beperkte model-Boonekamp reeds de orde 100*100 overschrijdt en dat haar meeste elementen uit nulllen bestaan.
U dient bij het invoeren zorgvuldig te werk te gaan, doch in een complex systeem zijn desondanks fouten mogelijk. Met name fouten die samenhangen met de demands:

```
ZDOUNB (oplossing onbegrensd)
```

en:

```
ZDONFS (ontoelaatbaar probleem)
```

zult u dan zien optreden.
In het eerste geval zijn er te weinig beperkingen opgenomen of diende u een minimalisatieprobleem op te lossen. In het tweede geval zijn er teveel en te strenge beperkingen opgelegd die het toegestane gebied doen verdwijnen. Er is dan sprake van tegenstrijdigheden in de randvoorwaarden bijvoorbeeld: meer output dan input.
Als u de fout niet direct kunt opsporen is het raadzaam om het model als minimalisatieprobleem op te lossen. In de uitvoer, die u via OUTPUT(DISK) kunt verkrijgen, vindt u dan alle energiestromen die u in de diagrammen van het model kunt

invoeren. Tegenstrijdigheden komen dan onmiddellijk aan het licht.
Indien het minimalisatieprobleem ook niet werkt dient u eenn of meer beperkingen te laten vervallen, die als "boosdoener" in aanmerking komen.
Het is raadzaam om de file DISKOUT, onmiddellijk nadat u hem hebt uitgeprint, te verwijderen daar ze cumulatief alle output opneemt die u er naar toe schrijft, die dan steeds opnieuw weer wordt uitgeprint. Een uitvoerfile is namelijk al snel van een aanzienlijke lengte.
Tenslotte nog een opmerking over meerekenfuncties. Dit zijn grootheden die niet rechtstreeks van invloed zijn op de uitkomst van de berekening doch waarin we niettemin geinteresseerd zijn. Zo kunnen we, als we de kostprijs minimaliseren, wel degelijk ook het energetisch rendement willen weten. Dit wordt dan als extra vergelijking met een extra variabele ingevoerd.
Voorbeelden van meerekenfuncties in het model-Boonekamp zijn:
TOT-CONS, TOT-IN, A-TOT enz.

Opgave: Tracht het model NEDMOD1 uit te breiden met een van de niet-conventionele processen uit fig. 3.2 t/m 3.8. Voeg aan het betreffende proces een realistische capaciteitsgrens toe. Deze behoort tot het scenario. Ze wordt namelijk begrensd door technische mogelijkheden en investeringskosten welke niet expliciet in het model zijn opgenomen.
Realistische waarden voor de coefficienten zijn:

c32	0.67	c33	0.70	c53	0.70
c142	0.76	c144	0.28	c145	0.2
c146	0.1	c147	0.2	c152	0.4
c153	0.4	c154	0.4	c231	0.7
c232	0.9	c233	0.9		

9. EXTRA MOGELIJKHEDEN VAN TEMPO

9.1 Selectielijsten.

Bij het werken aan wat grotere modellen worden outputfiles al snel zeer lang en onoverzichtelijk. Dit probleem wordt ondervangen door het gebruik van een selectielijst die de gebruiker in staat stelt om slechts die gegevens uit te printen die hij zich wenst, b.v. betreffende een bepaalde rij, een bepaalde kolom, een reeks uitgangsvariabelen.
Er kan slechts een selectielijst tegelijkertijd worden samengesteld en deze wordt gekenmerkt door <lijstnaam>. Deze ene lijst kan ook worden onthouden en meerdere malen aangeroepen, maar ze verdwijnt als een nieuwe selectielijst wordt opgebouwd.
Wensen we te werken met een selectielijst dan maken we dit kenbaar door de modifier SELIST toe te voegen, bijvoorbeeld:

```
BCDOUT(SELIST)
OUTPUT(SELIST)
```

Geven we deze modifier op dan verschijnt de boodschap:

```
SELECTION LIST BUILDER
SELECTION LIST NAME
%
```

We voeren dan de gewenste <lijstnaam> in. Mocht deze reeds geregistreerd staan dan verschijnt de boodschap:

SELECTION LIST <lijstnaam> ALREADY BUILT-NOW AVAILABLE

Als ze echter nog samengesteld moet worden geschiedt dat op interactieve wijze. In deze mode (met als prompt weer een %) kunnen opnieuw twee soorten invoer worden gegeven en wel speciale commando's en invoergegevens. De speciale commando's zijn:

STOP ENDATA LIST

die dezelfde betekenis hebben als in hoofdstuk 4.

Voorts kunnen sectienamen worden ingevoerd, en wel:

ROWS COLUMNS

Vervolgens worden sleutelwoorden gegeven, gevolgd door een schuine streep en een lijst van de betreffende elementen, gescheiden door komma's.
Sleutelwoorden zijn:

NAMES MASKS LIMITS

Na NAMES komt een lijst van rijen resp. kolommen waarvan we wensen dat ze afgedrukt worden:

NAMES/<naam1>,<naam2>,....

Na MASKS komen maskers. Hierbij wordt gebruik gemaakt van de "wild card" Als we bijvoorbeeld alle variabelen willen bekijken die het woord OIL op de 4e t/m 6e positie hebben, geven we het masker:

***OIL*

en de instruktie wordt:

MASKS/<masker1>,<masker2>,.....

Met LIMITS wordt een lijst van deelbereiken geopend. Stel dat we van de op volgorde staande rijen N1 t/m N312 slechts N10 t/m N15 wensen af te drukken, dan geven we het deelbereik:

N10 N15

en het commando wordt:

LIMITS/<deelbereik1>,<deelbereik2>,...

Tenslotte is er het edit-commando REMOVE, om regels uit de selectielijst te verwijderen:

REMOVE/<sleutelwoord>/<namen,masks of deelbereiken>

voorbeeld:

```
ROWS
REMOVE/NAMES/N2,N5
```

Een voorbeeld van een sessie is:

```
BCDOUT(SELIST)
LIJST
```

```
ROWS
NAMES/N1,N2
COLUMNS
MASKS/***GAS*
ENDATA
```

waarna BCDOUT uitvoer geeft rekening houdend met de zojuist aangemaakte selectielijst met de naam LIJST. We hebben in het voorbeeld alleen de in te tikken tekst vermeld.

9.2. Basismanipulatie.

Algemeen.

Het kan van nut zijn, vooral bij het werken met grotere modellen, om basisoplossingen van een probleem vast te leggen. De theorie van de basisoplossingen is geschetst in paragraaf 1.2.4. van het dictaat energiesystemen.
In twee gevallen is dit handig:
1.Een model is berekend en daarna moet een kleine wijziging worden aangebracht. Door de berekening van het gewijzigde model te starten met de optimale basis van het oorspronkelijke probleem kan rekentijd worden bespaard.
2.Een model is samengesteld uit een aantal deelmodellen. Ieder deelmodel wordt afzonderlijk geoptimaliseerd en de bijbehorende bases worden weggescchreven. Bij de berekening van het samengestelde probleem worden deze basis teruggehaald en als het ware bij elkaar opgeteld. Zij vormen tezamen de beginbasis voor het nieuwe probleem.

Procedures BASIN en BASOUT

De Tempo procedures BASIN en BASOUT zijn procedures voor het bewaren van bases in externe files.
BASOUT schrijft de basis van het probleem weg naar DISKOUT, mits de modifier DISK is gebruikt. Met de modifier PRINT verschijnt de uitvoer ook op het scherm. Met behulp van het $FILE statement is het mogelijk om de basis naar een file <filenaam> te schrijven. Op zo'n file kunnen dan eventueel ook bases verzameld worden die zich van elkaar onderscheiden door de datasetnaam die door ZDATA wordt gegeven.
De basis is nul (leeg) als PRIMAL nog niet is uitgevoerd, daar dan slechts de initiele (nul-) basis is gegenereerd. We kunnen bijvoorbeeld intikken:

```
PRIMAL
$FILE DISKOUT=BASFILE
BASOUT(PRINT,DISK,ZDATA="BAS1")
```

Indien dit gebeurt voor het vraagstuk van paragraaf 1.2. van het dictaat "Energiesystemen", verschijnt de file:

```
NAME     BAS1
XL   X1    Z4
XL   X2    Z2
ENDATA
```

Hierin zijn X1 en X2 de structurele basisvariabelen. In de basis worden ook structurele variabelen vermeld die de bovengrens (upper bound) bereikt hebben, maar dat treedt in dit vraagstuk niet op.
Z2 en Z4 zijn de bijbehorende logische (of rij-) variabelen. ~~Zij~~ De bijbehorende slacks zijn op hun benedengrens (bedenk dat X1=6 en X2=3 bij de

optimale oplossing), want:

$$A2 = \hat{Z}2 - X1 - 2*X2 = 0$$

Dit wordt aangegeven met XL. Met UL wordt de bovengrens aangegeven.

Bovengenoemde basis kan als volgt weer worden opgehaald:

```
SETUP(MAX)
$FILE DISKIN=BASFILE
BASIN(DISK,ZDATA="BAS1")
PRIMAL
```

waarna de l.p.-berekening wordt uitgevoerd, startend vanaf de opgehaalde basis.

Bij het werken met submodellen wordt gebruik gemaakt van de modifier MODIFY. Deze bewerkstelligt dat de basis niet bij de nulbasis wordt "opgeteld" doch bij de reeds ingevoerde basis. Als voorbeeld geven we de procedure voor het ophalen van een basis die bestaat uit de optimale bases van drie submodellen die als dataset BAS1 t/m BAS3 op de externe file BASFILE zijn ondergebracht:

```
SETUP(MAX)
$FILE DISKIN=BASFILE
BASIN(DISK,ZDATA="BAS1")
BASIN(DISK,ZDATA="BAS2",MODIFY)
BASIN(DISK,ZDATA="BAS3",MODIFY)
PRIMAL
```

De procedures SAVE en LOAD.

Het is ook mogelijk om de bases slechts op de probleemfile ZPROF vast te leggen. Dit geschiedt in het eenvoudigste geval onder dezelfde naam die door ZNAME is vastgelegd. Het is echter ook denkbar een eigen naam op te geven hetgeen geschiedt met de interne parameter ZBASNM, dus:

```
PRIMAL
SAVE(ZBASNM="BAS")
```

Deze basis kan later worden opgehaald met:

```
SETUP(MAX)
LOAD(ZBASNM="BAS")
PRIMAL
```

N.B.: De procedure-aanroep SAVE heeft een andere betekenis dan het speciale commando SAVE dat in de interactieve mode wordt gebruikt en reeds werd behandeld.

10. Gevoeligheidsanalyse.

10.1. Algemeen.

Met behulp van de Tempo-procedure RANGE kan een gevoeligheidsanalyse worden uitgevoerd voor de optimala oplossing van een l.p.-probleem. RANGE wordt uitgevoerd na PRIMAL. De output van RANGE geeft aan hoezeer bepaalde coefficienten en variabelen kunnen veranderen alvorens de set van basisvectoren verandert. Totaan dit punt zijn de verbanden

tussen de grootheden lineair, op zo'n punt vertonen de verbanden een knik.
Tevens wordt onderzocht hoe de objectfunctie wordt beinvloed door verandering van de variabelen.

We geven een aantal definities en begrippen.

Objectfunctie of opbrengst- (resp. kosten-) functie:

$$J = C1*X1 + C2*X2 + ...$$

In het geval van een minimalisatieprobleem wordt C1 enz. de (initiele) kostencoefficient genoemd, in geval van een maximalisatieprobleem de (initiele) opbrengstcoefficient.

Een vergelijking in een l.p.-probleem heeft bijvoorbeeld de volgende vorm:

$$Z1 = Q*X1 + Q*X2 + ... <= Z1$$

Z1, Z2 enz. zijn de logische- of rijvariabelen.
X1, X2 enz. zijn de systeem- of kolomvariabelen.
A1, A2 enz. zijn de verschil- of slackvariabelen, gedefinieerd als:

$$A1 = \hat{Z}1 - Z1$$

Range is het bereik van Z1:

$$\hat{Z}1^{L} <= Z1 <= \hat{Z}1^{H}$$

$\hat{Z}1^{L}$ is de benedenwaarde (lower level) en $\hat{Z}1^{H}$ is de bovenwaarde (upper level).

De gereduceerde kosten of opbrengsten zijn de kostencoefficienten D1, D2 enz. zoals ze na het 'vegen' in de benedenrij van het simplex-tableau voorkomen. In de uitvoer worden ze soms met DJ aangegeven. Een synoniem ervoor is 'schaduwprijzen' aangevend dat het om de invloed op de objectfunctie gaat door verandering van een variabele. De schaduwprijs is niet de werkelijke prijs doch de prijs met betrekking tot het systeem.

De uitvoer van RANGE bestaat uit vier secties en wel: Kolommen resp. rijen die limietwaarden (LL resp. UL) bezitten, dan wel basisgrootheden zijn (BS), aangegeven met 'intermediate level'.

10.2. Voorbeeld: minimalisatieprobleem.

Beschouwen we de nuloplossing van het vraagstuk uit hoofdstuk 1 van het dictaat. Dit is de optimale oplossing van het minimalisatieprobleem. De vergelijking Z3 wordt nergens actief, ze speelt geen rol. Van belang is:

```
Z1 =   X1 + 4*X2 <= 20
Z2 =   X1 + 2*X2 <= 12
Z4 = 3*X1 + 2*X2 <= 24
J  =   X1 +   X2
```

ze zijn immers nul, en de basis wordt gevormd door A1,A2,A4. Fig. 10.1 toont de grafische oplossing, en in fig.10.2 zijn de bases weergegeven die bij een aantal oplossingen (toegestaan en niet- toegestaan) behoren.
We beschouwen de output voor de 'rows at intermediate level':

Genoemd worden achtereenvolgens (kolomsgewijs):
-De namen van de logische variabelen en de status van de bijbehorende slacks, bijvoorbeeld Z1, BS.
-De rij activiteit en de slack actviteit (Z1=0, A1=20)
-De beneden- en bovenlimiet voor Z1 zoals opgegeven (GEEN, resp. 20)
-'Lower activity' en 'upper activity'. Deze geven aan tot hoever we de bovenwaarde van Z1 mogen laten afnemen resp. de benedenwaarde laten toenemen voordat de basis wordt veranderd. Dit komt overeen met een wijziging van het rechterlid resp. het type van de vergelijking. Het zegt wat de invloed van veranderde begrenzingen is op de systeemkosten.
We zien dat de bovengrens tot nul (.) mag afnemen en de benedengrens tot 24 mag toenemen. Fig. 10.3 laat zien dat op deze punten het toegestane gebied verdwijnt.
-'Unit cost'. Deze twee getallen geven weer hoeveel de schaduwprijs is per eenheid van de gegeven veranderingen. Zoals de figuur laat zien treedt bij verhoging van de benedenwarde van Z1 een verschijving van het optimum op langs de X2-as, dus X1 blijft nul. Opdat Z1 = X1 + 4*X2 gelijk wordt aan 1 moet dus X2 toenemen van 0 tot 0,25 en J = X1 + X2 is dan 0,25 waarmee de gereduceerde kosten per eenheid toename van Z1 vastliggen.
-'Limiting process' geeft nog twee aanduidingen voor het proces dat aanleiding geeft tot omslag in de basisoplossing. Dat is hier bij de bovengrens Z1=24 de variabele X2 die de basis betreedt, terwijl A2 de basis verlaat. (punt X2,A1,A4). Aan de benedengrens (Z1=0) verandert de basis niet.

De sectie 'columns at limit level' bevat de namen van de kolomvariabelen die op hun benedengrens (LL) zitten (ze zijn beide nul) en in de volgende kolom vinden we de kolomactiviteit (hier: nul) en de initiele kosten ('input cost'), ofwel de coefficienten van de objectfunctie, die in het voorbeeld beide 1 zijn.
De daaropvolgende kolom bevat de eventuele opgelegde begrenzingen ('bounds') waarna in een kolom onderzocht wordt hoezeer de systeemvariabelen kunnen toenemen of afnemen bij gelijkblijvende basis. Voor X1 is dat van 0 tot 8 en voor X2 van 0 tot 5, hetgeen direct uit fig. 10.4 volgt. De daaruit voortvloeiende kosten zijn gelijk aan de initiele kosten, dus +1 voor een toename en -1 voor een afname bij zowel X1 als X2.

10.3 Voorbeeld: Maximalisatieprobleem.

In de optimale oplossing bij het voorgaande vraagstuk, doch nadat een maximalisatieprobleem is bekeken, weten we dat A2 en A4 gelijk zijn aan nul en dat de basis wordt opgespannen door X1, X2 en A1. We krijgen de volgende uitvoer:

De 'Rows at limit level' zijn Z2 en Z4. De rij-activiteit is maximaal en de slack-activiteit is nul. We zien dat een minimum voor Z2 tot 12,8 op kan lopen en een maximum tot 8 af kan nemen voordat de basis verandert. Dit is af te leiden uit de fig.10.5. Ook de marginale opbrengst is vermeld, ze is gelijk aan 0,25 per eenheid toename van Z2 hetgeen als volgt is af te leiden: Als Z2 toeneemt dan neemt A2 af en uit het tableau is te zien dat dan J toeneemt met factor 1/4 (zie benedenrij van eindtableau), dat is de schaduwopbrengst. Ook het 'limiting process' wordt weergegeven: Bij de benedengrens verlaat X2 de basis, bij de bovengrens verlaat A1 de basis (zie figuur). Ook ziet men dat X2 dan op haar benedengrens is (nul) en Z1 op haar bovengrens (namelijk 20).

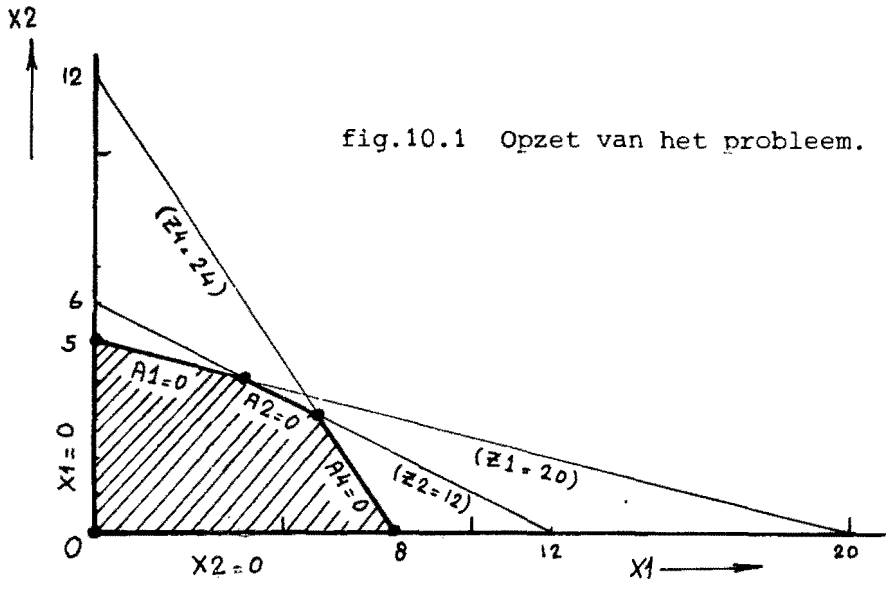


fig.10.1 Opzet van het probleem.

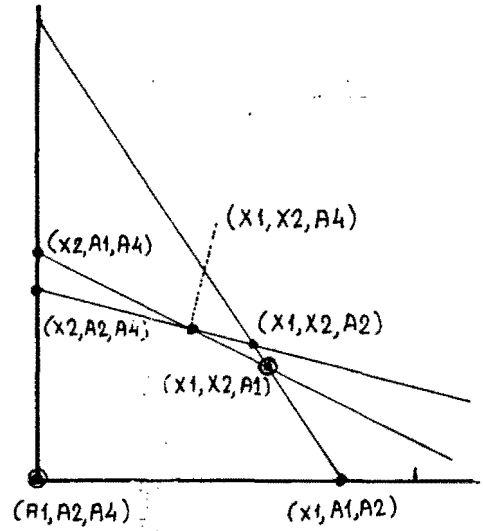


fig.10.2 Basis-oplossingen.

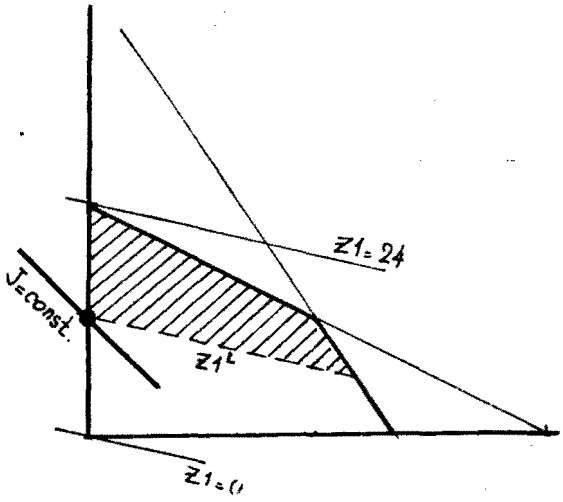


fig.10.3. Minimalisatieprobleem. Invloed van Z1.

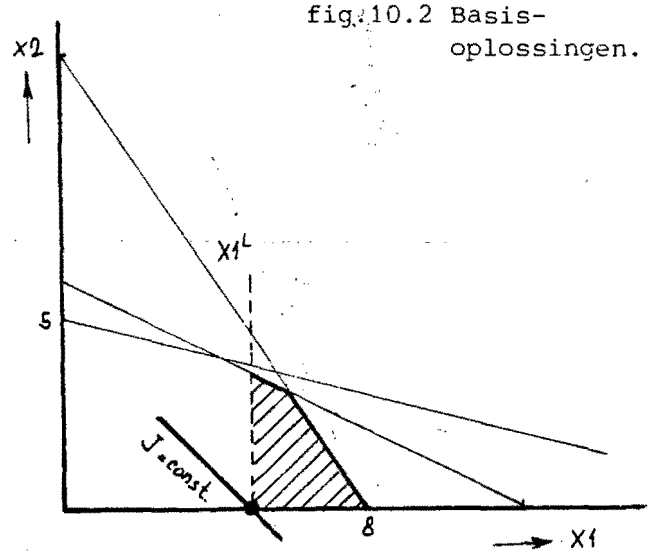


fig.10.4. Id. Invloed van X1

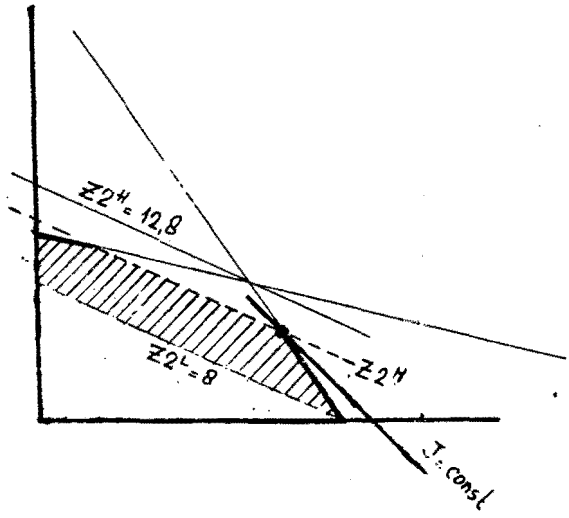


fig.10.5. Maximalisatieprobleem. Invloed van Z2.

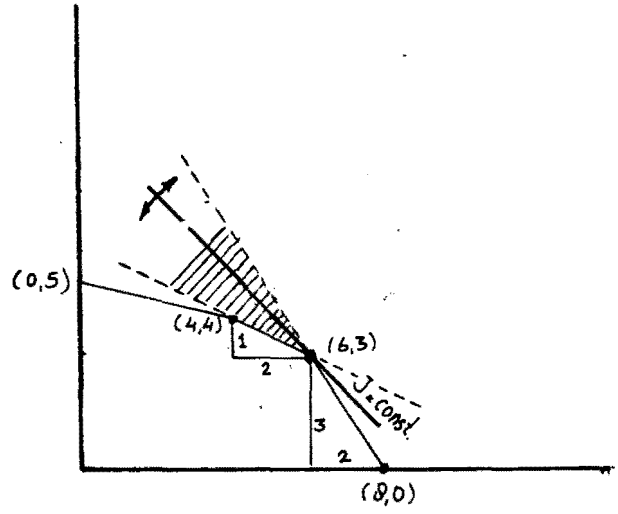


fig.10.6. Maximalisatieprobleem. Variatie initiële kosten.

De 'Rows at intermediate level' bevatten nu bekende informatie, doch de 'columns at intermediate level' geven nog extra gegevens namelijk de initiele kostencoefficienten kunnen varieren bij gelijkblijvende basisoplossing (fig. 10.6)

11. PARAMETRISCHE LINEAIRE OPTIMALISATIE

11.1. Algemeen.

De lineaire optimalisering geeft als uitgangsgrootheid een bepaalde oplossing. Met de gevoeligheidsanalyse kan bekeken worden in hoeverre deze oplossing afhankelijk is van een bepaalde parameter. Aan de hand van het probleem dat bij de behandeling van de gevoeligheidsanalyse gebruikt werd zal ook worden nagegaan hoe de parametrische programmering benut kan worden om het gedrag de optimale oplossing bij een geleidelijke verandering van het probleem te bestuderen.

Parametrisatie kan worden toegepast op de objectfunctie, op rijvariabelen, kolomvariabelen en rechterzijden of een combinatie daarvan.
Beschouwen we nu een optimalisatieprobleem dat door de systeemvariabelen X1, X2 enz., de logische variabelen Z1, Z2 enz., de begrenzingen $\hat{Z}1$, ... en de objectfunctie J wordt bepaald. Het is nu denkbaar om een hulpobjectfunctie JC in te voeren en dan de oplossing van een verwant probleem te bepalen, met:

$$J^* = J + \lambda * JC$$

Hierin is λ een parameter die van 0 tot een zekere op te geven bovengrens varieren kan (ook een negatieve waarde mag worden opgegeven). De berekeningen zijn post-optimaal, d.w.z. dat parametrisatie wordt toegepast nadat de optimale oplossing van het oorspronkelijke probleem is bepaald. De parametrisatie gaat eveneens gepaard met iteraties, d.w.z. met "sprongen" naar de verschillende basisoplossingen, wat de knikpunten zijn in de overigens lineaire verbanden tussen J en parameter λ.

11.2. Tempo-procedures.

In Tempo staan voor parametrische programmering de procedures PARCOS, PARROW, PARCOL, PARRHS en PARRIM ter beschikking. Voor al deze procedures is het nodig dat SETUP en PRIMAL is uitgevoerd en dat bovendien de parameter ZPARMAX wordt gedefinieerd.
ZPARMAX is het maximum tot waar men λ, te beginnen van nul, laat toenemen. Voorts wordt een hulpobjectfunctie, een hulprechterzijde enz. gedefinieerd al naar gelang het type parametrisatie dat men wenst uit te voeren. Zo is daar ZCOBJ (hulp-objectfunctie), ZCRHS, ZCCOL en ZCROW. De letter C staat voor "change".
De uitvoer is zodanig dat de volgende gegevens worden verstrekt:
-iteratietype en -nummer
-waarde der objectfunctie
-variabele die basis verlaat
-variabele die basis betreedt
-waarde van parameter. Een nadeel is hierbij, dat niet de namen der variabelen worden aangegeven doch de interne representatienummers.

We behandelen de verschillende procedures een voor een:

PARCOS

We geven een voorbeeld, waarbij we stellen dat het vraagstuk in de file VR1 is opgeslagen. De objectfunctie wordt zoals gegeven in 11.1. geparametriseerd. Nemen we: ZPARMAX=200 als bovengrens (de "bovengrens" mag ook negatief zijn!) dan kan de volgende file worden ingevoerd:

```
ZPARMAX=200
ZCOBJ="JC"
INPUT(REMOTE)
CONT/VR1
ROWS
N/JC
COLUMNS
X2/JC/1.0
ENDATA
SETUP(MAX)
PRIMAL
PARCOS
```

Dan is hier: JC=X2 en:

$$J^* = J + \lambda * JC = X1 + (1+\lambda) * X2$$

en de output wordt:

IT	OBJ	OUT	IN	PAR
1	12	Z1	Z4	1
2	20	X1	Z2	3

In figuur 11.1 en 11.2 is de interpretatie te zien van deze uitvoer, waarbij ze is uitgebreid tot negatieve .

Deze procedure parametriseert op de kostenfunctie.

PARRHS

De input kan bijvoorbeeld zijn (variatie van het rechterlid van Z3):

```
ZPARMAX=100
ZCRHS="RHS2"
INPUT(REMOTE)
CONT/VR1
RHS
RHS2/Z3/-1.0
ENDATA
SETUP(MAX)
PRIMAL
PARRHS
```

en aan de hand van figuur 11.3 kan de uitvoer onmiddellijk worden geinterpreteerd. Bedenk:

$$\hat{Z3}^* = 30 - \lambda.$$

Voor $\lambda = 30$ wordt een "hard maximum" vermeld en uit de figuur volgt direct wat dat betekent.

PARROW

Deze procedure is voor de parametrisatie van rijvariabelen. Eigenlijk is ze bij uitstek geschikt om matrixcoefficienten te parametriseren. Dit gaat als volgt: Er wordt een interne parameter ZROW gedefinieerd die de gewenste rij aangeeft, bijvoorbeeld Z1. Er wordt een hulprij via ZCROW gedefinieerd, bijvoorbeeld ZC, die in de COLUMNS-sectie een waarde krijgt. Deze hulprij is van het type N. De volgende invoer is denkbaar:

```
ZPARMAX=200
ZCROW="ZC"
ZROW="Z1"
INPUT(REMOTE)
CONT/VR1
ROWS
N/ZC
COLUMNS
X1/ZC/1.0
ENDATA
SETUP(MAX)
PRIMAL
PARROW
```

Het probleem wordt dan uitgevoerd voor:

$Z1^{*} = (1+\ )*X1 + 4*X2$

In figuur 11.4 is te zien dat de lijn Z1 =20 nu om het punt (0,5) heen draait, en dat bij $\lambda = 3$ een ontaarding optreedt, daar de lijn dan evenwijdig met J=const komt te liggen. Het optimum klapt dan om van (5,0) naar (0,5) en de output wordt:

```
1   9   Z1   Z2   0.33
2   8   X2   Z4   1.5
3   5   X1   X2   3
```

PARCOL

Deze procedure verloopt analoog aan PARROW, ook hier wordt de mogelijkheid geboden om coefficienten te parametriseren. We geven een voorbeeld voor verandering van de coefficient voor X1 in Z2.

```
ZPARMAX=200
ZCOL="X1"
ZCCOL="ZC"
INPUT(REMOTE)
CONT/VR1
RHS
XC/Z2/1.0
ENDATA
SETUP(MAX)
PRIMAL
PARCOL
```

met als uitvoer:

```
1   8   X2  Z4   0.5
2   6   X1  X2   1
```

zie de figuur 11.5

PARRIM

Met deze procedure kan zowel de rechterzijde als de

objectfunctie worden geparametriseerd. Zowel ZCOBJ en ZCRHS moeten een waarde krijgen toegekend bijvoorbeeld:

```
ZCOBJ="JC"
ZCRHS="RHS2"
```

en ingevoerd wordt:

```
N/JC
X1/JC/1.0
RHS2/Z3/-1.0
```

met uitvoer:

```
1  12   X2   Z2   0.5
2 120   Z3   Z4  14
```

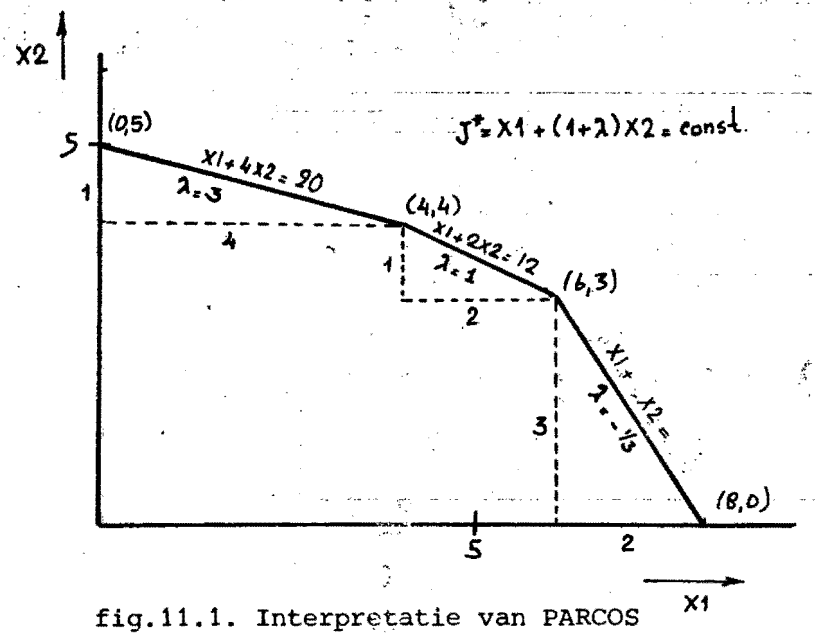


fig.11.1. Interpretatie van PARCOS

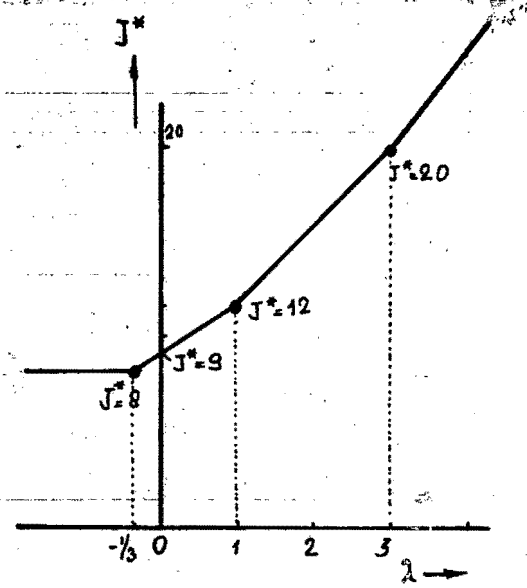


fig.11.2. Resultaat van PARCOS

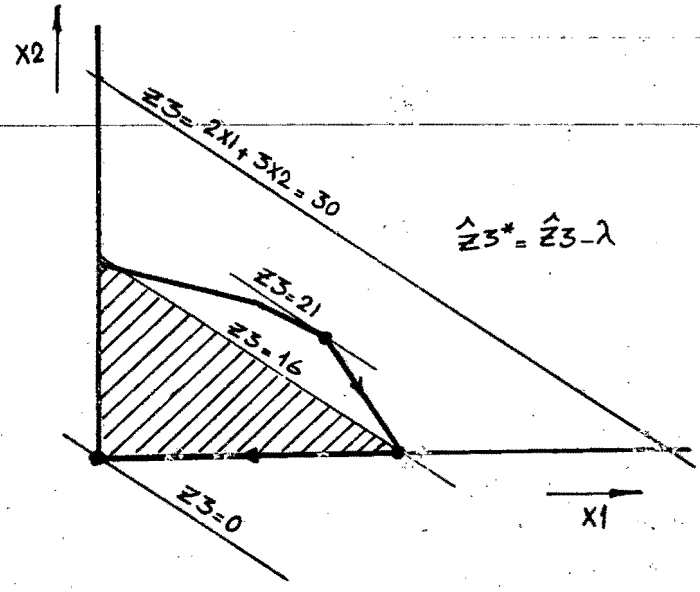


fig.11.3. Interpretatie van PARRHS

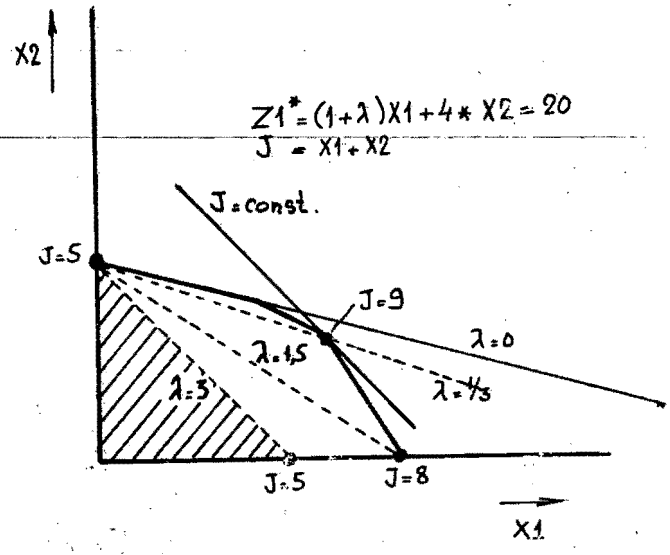


fig.11.4. Interpretatie van PARROW

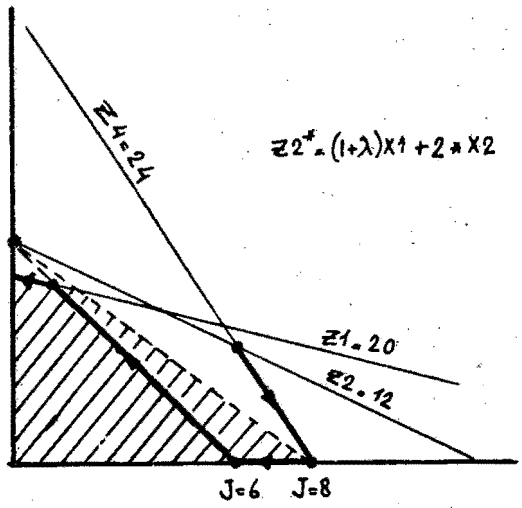


fig.11.5. Interpretatie van PARCOL
Na het punt J=6 wordt de sprong niet meer gemaakt.