STICHTING
MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM
REKENAFDELING

CURSUS: WETENSCHAPPELIJK REKENEN B

Onderdeel: "Proces Analyse"

door

Prof.dr.ir. A. van Wijngaarden

1965

Dat men een naam op zo verschillende manieren kan gebruiken levert in een natuurlijke taal weinig moeilijkheden op.
Als men met formele talen te maken heeft, moet men dit onderscheid goed in acht nemen.
Laat onder de symbolen van de formele taal ook letters voorkomen.
Voorbeelden van namen zijn dan

a

l e t t e r

In de taal kan men de namen gebruiken om objecten aan te duiden.
Men kan de namen ook opvatten als rijtjes symbolen zonder meer.
De zin " a is een letter" zegt iets over het symbool a.
Het woord letter heeft niets te maken met de naam letter in de formele taal. Om dit onderscheid aan te geven kan men bijv. schrijven

a is een <letter>

In de formele taal komt <letter> niet voor.
Het is een begrip uit de taal waarin men over de formele taal spreekt.
Aan dit voorbeeld kan men zien dat voor de beschrijving van een taal een andere taal nodig is, een z.g. metataal.

In deze cursus zullen we processen beschrijven met behulp van Algol 60.
De taalregels van Algol 60 zijn vastgelegd in het Revised Report on the Algorithmic Language Algol 60 by J.W.Backus e.a.

Enkele punten uit dit rapport worden in de volgende paragraaf besproken en met voorbeelden toegelicht.
Voor een meer systematische inleiding in Algol raadplege men de Cursus Programmeren in Algol 60 door prof.dr.E.W.Dijkstra, verkrijgbaar bij het Mathematisch Centrum.

1 Algol 60

In de metataal worden de volgende symbolen gebruikt

< > | ::=

| heeft de betekenis van "of".

::= heeft de betekenis van "is gedefinieerd door".

Met < letter > bedoelen we een metalinguistische variabele. De waarden van een dergelijke variabele zijn rijtjes symbolen.
Met een metalinguistische formule kunnen we aangeven wat voor waarden de variabele kan aannemen.

Voorbeeld

< letter > ::= a|b|c|d

< digit > ::= 0|1|2|3|4|5|6|7|8|9

In woorden: de variabele <letter> heeft de waarde a of b of c of d; de variabele <digit> heeft de waarde 0 of 1 of 2 of 3 enz.

In RR. 2.1 (sectie 2.1 van het revised report) zien we dat het waarde-

bereik van de variabele <letter> heel wat groter is, nl. het gehele alphabet in kleine letters en in hoofdletters. Eventueel kan men het alphabet met andere karakters uitbreiden.
Met behulp van metalinguistische formules kan men de syntax van Algol 60 volledig beschrijven. Onder syntax verstaan we het geheel van regels volgens welke symbolenrijen in de taal kunnen optreden. Daarnaast moet men de betekenis, de bedoeling van deze symbolenrijen aangeven.
Dit noemt men de semantiek.

In RR 2.4.1 wordt gedefinieerd welke symbolenrijen we in Algol 60 als identifier d.w.z. als naam kunnen gebruiken.

< identifier > ::= < letter > | < identifier > < letter > |
< identifier > < digit >

Deze definitie lijkt vreemd, omdat rechts van ::= het te definieren begrip identifier ook voorkomt.
Bij eerste lezing komt men daardoor niet verder dan

< identifier > ::= < letter >

Dit levert 52 voorbeelden van identifiers, n.l. de letters a,A,b,B,... enz.
Lezen we de formule opnieuw, dan vinden we als nieuwe voorbeelden van identifiers ab, ac etc. en ook a1, a2 etc.
Nogmaals lezen geeft abc, ab1, a12 etc.

Dit is een <u>recursieve</u> definitie van het begrip identifier.

We zijn genoopt willekeurig lange identifiers te accepteren. Dit komt door de manier van definieren. Men kan in dit recursieve mechanisme niet gemakkelijk uitdrukken dat een identifier niet langer dan bijv. zes symbolen mag zijn.
We zien hier hoe de metataal invloed kan hebben op de structuur van de taal zelf.

<u>Opmerking</u>

Letters en identifiers hebben geen enkele intrinsieke betekenis (zie RR 2.1 en 2.4.3). Cijfers hebben de gewone betekenis.

Met de cijfers kunnen we getallen opbouwen.
De syntax is gegeven in RR 2.5. Bij wijze van voorbeeld zullen we de syntactische definitie van getallen op de voet volgen.

	Voorbeelden
< unsigned integer >	Na eerste lezing : 1 2 8 Na tweede lezing : 12 23 15
< integer >	123 +123 -123
< decimal fraction >	.123
< exponent part >	$_{10}^{4}$ $_{10}^{-3}$ $_{10}^{+5}$
< decimal number >	3 .14 3.14

< unsigned number >	3.14 $_{10}$-3 3.14$_{10}$-3
< number >	3.14 + 3.14 -3.14$_{10}$-3

Men verwarre het basis symbool $_{10}$ (lees : maal tien tot de macht) niet met het getal 10.
In de syntactische formules komt nergens meer < number > voor. Dat de definitie van < number > nodig is blijkt bij lezing van RR 1.1 .

We behandelen het Algol rapport niet in extenso. We beperken ons tot het geven van toelichtingen en voorbeelden bij een aantal secties.

RR 2.2.2. Logical values

< logical value > ::= true | false

In deze sectie komen we voor het eerst de onderstreping tegen. Algol heeft behoefte aan een groot aantal symbolen. Onderstreping maakt het mogelijk nieuwe symbolen te creëren zonder in typografische moeilijkheden te komen.
true is een nieuw basis symbool; het heeft niets te maken met de letters t, r, u en e.
true en false hebben een intrinsieke betekenis.

RR 2.6. Strings

Als men tekst wil uittypen moet men een rij symbolen hanteren. Deze rij symbolen moet men kunnen onderscheiden van bijv. een identifier. Daartoe plaatst men om de tekst string quotes. Dat zijn de basis symbolen ' quote en ' unquote.
De syntactische definitie van een string houdt in dat de string quotes netjes in paren voorkomen.
Voorbeelden:

'x 'jan' pret'

'klaas'

RR 3.1. Variables

Bij het woord array kan men denken aan een vector of een matrix.
Een array heeft als componenten geindiceerde variabelen.
Voorbeeld.

De vector (het array) a met de componenten

a[-1], a[0], a[1], a[2].

Dit is een een-dimensionaal array. Met de array identifier a duiden we de hele vector aan. Met bijv. a[1] duiden we een bepaald element van de vector aan.

Het aantal indices van een geindiceerde variabele is niet beperkt tot 1 of 2.

RR 3.3. Arithmetic Expressions

We geven een paar voorbeelden.

x + y x z ↑ 3 / a[5] + u

↑ betekent machtsverheffing. De overige tekens in deze expressie hebben hun conventionele betekenis.
Voor de berekening wordt eerst de waarde van de primaries x, y, z, a[5] en u bepaald.
De operaties worden in volgorde van prioriteit uitgevoerd en bij gelijke prioriteit van links naar rechts.
Machtsverheffing heeft de hoogste prioriteit. Daarna volgen deling en vermenigvuldiging met dezelfde prioriteit.

2 ↑ 2 ↑ 3

In deze expressie hebben de operaties dezelfde prioriteit en worden dus van links naar rechts uitgevoerd.
En wordt berekend $(2^2)^3$.
Wil men $2^{(2^3)}$ berekenen, dan schrijft men

2 ↑ (2 ↑ 3)

Door haakjes te plaatsen om 2 ↑ 3 is hiervan een primary gemaakt (zie RR 3.3.1).

(x ↑ 2 + y ↑ 2 + z ↑ 2) ↑ 3/2

Dit is correct Algol, maar desalniettemin verkeerd als men bedoelde op te schrijven

$(x^2 + y^2 + z^2)^{3/2}$.

In een rekenproces moet men vaak verschillende gevallen onderscheiden. Met de if clause kan men dat in de uitdrukking verwerken.
De conditionele uitdrukking is een zeer nuttig element in de taal.

if x > 0 then 3 xy + 5 else 5 x y + 3

Als de waarde van de relatie x > 0 gelijk is aan true, wordt berekend 3xy + 5.
Is de waarde false dan wordt 5 x y + 3 uitgerekend.

if a ≠ b then 6 else if a > 0 then 3 else 0

Voor a ≠ b true heeft de expressie de waarde 6.
Voor a ≠ b false en a > 0 true is de waarde 3.
Voor a ≠ b false en a > 0 false is de waarde 0.

if a ≠ b then if a > 0 then 9 else 8 else if u < v then 3 else 0

Dit is geen Algol. De uitdrukking if a > 0 then 9 else 8 is namelijk niet simpel (zie RR 3.3.1). Door hier haakjes om heen te zetten maakt men er een simpele expressie van.

Het is aan te bevelen om in geval van twijfel het zekere voor het onzekere te nemen en haakjes te gebruiken.

Over de arithmetiek kan men het volgende opmerken.
Men moet zich steeds realiseren wat de precieze betekenis van een uitdrukking is. Als er alleen met gehele getallen wordt gerekend is de uitkomst exact, aangenomen dat men binnen de capaciteit van de machine blijft. Anders opereren we in de zin van de numerieke analyse (zie RR 3.3.6). De uitkomst is niet exact. De consequenties hiervan zijn voor de gebruiker. Hij moet in het programma voldoende maatregelen nemen om de gevolgen van cijferverlies e.d. te ondervangen. De taal helpt ons niet van alle moeilijkheden van de numerieke wiskunde af.

RR 3.4 Boolean expressions

Voorbeelden van Boolean expressies:

true

false

x (variabele)

a[3] (subscripted variabele)

$x = y$, $x \neq y$, $x < y$, $x \leq y$, $x > y$, $x \geq y$

$x + y < z \uparrow 3/u$

(if a > b then x+3 else z) < a ↑ b + 5

De laatste acht expressies zijn voorbeelden van relaties.

Men kan ingewikkelder uitdrukkingen maken met behulp van de logische operatoren

niet	$\neg$	$\neg x$
en	$\wedge$	$x \wedge y$
of	$\vee$	$x \vee y$

$\neg x$ heeft de waarde false als x true is en de waarde true als x false is.

$x \wedge y$ heeft de waarde true als x en y beide true zijn. In alle andere gevallen is de waarde false.

$x \vee y$ heeft de waarde true als van de operanden tenminste een true is.

Zie ook de tabel in RR 3.4.5.

Men noemt $\vee$ (afkomstig van het latijnse woord vel) het niet exclusieve of, d.w.z. $x \vee y$ is ook true als beide operanden true zijn.

Voorbeeld
De voorwaarde $0 < x < 1$ kan men met behulp van logische operatoren schrijven als

$$0 < x \wedge x < 1.$$

RR 4.2 Assignment statements

In een Algoltekst komen statements voor. Een statement is een operatievoorschrift; men moet uitvoeren wat in de statement is aangegeven. Het eenvoudigste type is de assignment statement (toekennings uitspraak). Een assignment statement houdt in dat men de waarde van een uitdrukking uitrekent en aan deze waarde een naam geeft, zodat men in een later stadium van de berekening aan deze waarde kan refereren. De naam noemen we een variabele.

Voorbeelden

```
x := 3.14
```

De uitdrukking rechts van het "wordt" teken wordt uitgerekend. De waarde, zijnde 3.14, wordt toegekend aan de variabele x. Tot nader order, d.w.z. tot een volgende assignment aan deze variabele, houdt x nu de waarde 3.14.

```
x := (3.14×y + 2.81) ↑3/x
```

Rechts komt x ook voor. Hier is niets tegen. Eerst wordt de uitdrukking aan de rechterkant uitgerekend m.b.v. de waarde die x op dat moment heeft. Daarna wordt aan x een nieuwe waarde toegekend.
Een zeer veel voorkomende assignment statement van dit type is x := x + 1. Merk op dat x = x + 1 iets geheel anders is. Dit is een relatie die voor elke waarde van x de waarde false heeft.
Aan de linkerkant van het "wordt" teken is een geindiceerde variabele ook toegestaan.

Voorbeelden

```
A[1,2] := 3.14
A[i]   := i
A[read] := read
```

Hierbij geldt de volgende afspraak: eerst worden de indices van de variabele links van := uitgerekend, dan wordt de waarde van de uitdrukking aan de rechterkant bepaald en daarna vindt de toekenning plaats.
Deze afspraak is wezenlijk. Dit blijkt uit het derde voorbeeld. In dit voorbeeld heeft read de betekenis van "het volgende getal op de band".

Soms wil men de waarde van een expressie aan een aantal variabelen toekennen.
In plaats van

```
x := 3.14×t
y := x
A[3] := x
```

mag men schrijven

```
A[3] := y := x := 3.14×t
```

Nog een voorbeeld

```
A[i] := i := i + 1
```

Hier moet men weer oppaasen. Op grond van bovenstaande afspraak geeft deze assignment voor i = 5 resp. A[5] = 6 en i = 6.

De waarde van een logische uitdrukking kan men eveneens toekennen aan een variabele.

Voorbeelden

```
b := true
b := false
b := x = y
```

RR 4.3 Go to statements

Statements worden in de lexicographische volgorde uitgevoerd, d.w.z. in de volgorde waarin ze in de Algoltekst staan. Opeenvolgende statements worden van elkaar gescheiden door het symbool ";" .

Een go to statement geeft de mogelijkheid deze volgorde te doorbreken. Deze statement bestaat uit het symbool goto gevolgd door een label. Een label is een markering van een bepaald punt in de tekst.

Voorbeeld

```
goto L; i := 0; L : x := a; ; y := 0
```

De statement x := a is met L gelabeld. De statement goto L heeft als opvolger de statement x := a en niet de lexicographisch volgende statement i := 0.
In het voorbeeld staan twee puntkomma's achter elkaar. Tussen deze twee symbolen staat ook een statement nl. de dummy statement (zie RR 4.4.1). De dummy statement heeft geen enkel effect.

RR 4.5 Conditional statements

De conditionele statement maakt het mogelijk in een proces verschillende gevallen te onderscheiden.

```
if x = y then z := 3.14 else goto L;
```

Voor x = y true wordt uitgevoerd de statement z := 3.14. Vervolgens komt de statement na de puntkomma aan bod.
Voor x = y false wordt uitgevoerd goto L.

```
if x = y then z := 3.14 else;
```

Na else kan ook de dummy statement volgen. Voor x = y false doen we niets en gaan we door met de statement achter de puntkomma. Aangezien deze constructie nogal eens voorkomt mag men in dit geval else weglaten.
Men kan dus schrijven

```
if x = y then z := 3.14;
```

Beschouw nu de statement

```
if x = y then if a ≠ b then x := 3.14 else y := 2.81;
```

Het is niet duidelijk welke then bij de else hoort. Men kan de statement op twee manieren lezen. Dit is een gevolg van bovengenoemde conventie.
Een dergelijke dubbelzinnigheid is niet toegestaan. Na een if clause mag geen conditionele statement volgen.
Met behulp van de statement haken begin en end kan men van een conditionele statement een niet-conditionele statement maken. (zie RR 4.1.1)
Al naar gelang de bedoeling krijgt men

```
if x = y then begin if a ≠ b then x := 3.14 end else
y := 2.81;
```

of

```
if x = y then begin if a ≠ b then x := 3.14 else
y := 2.81 end;
```

We merken op dat bij conditionele uitdrukkingen het weglaten van else niet mogelijk is.

RR 4.6 For statements

Ter inleiding geven we een stukje programma dat het inwendige product $s = \sum_{i=1}^{n} a_i b_i$ van twee vectoren berekent.

```
    s := 0;
    i := 1;
L : s := s + a[i]×b[i];
    i := i + 1;
    if i ≤ n then goto L;
```

In s wordt $\Sigma a_i b_i$ gevormd; de variabele i is ingevoerd als teller.
Dit is een voorbeeld van een repetitieve berekening. Dit type berekening komt zo vaak voor dat ten behoeve hiervan een speciale statement is ingevoerd, de for statement.
Met behulp van de for statement kunnen we het stukje programma als volgt schrijven

```
s := 0;
for i := 1 step 1 until n do s := s + a[i]×b[i];
```

De for statement is geen principiëel element in de taal, zoals de assignment statement en de goto statement. Wat de for statement uitdrukt kan ook met de andere statements worden beschreven. Het gebruik van de for statement verhoogt echter aanzienlijk de leesbaarheid van het programma.

De for statement uit het voorbeeld is van het type

```
for i := a step b until c do S
```

Hierin kunnen a, b en c ingewikkelde uitdrukkingen zijn, die zelfs

van de lopende variabele i mogen afhangen.
De statement S wordt herhaald uitgevoerd; deze statement kan de waarde van i wijzigen. In RR 4.6.4.2 is de betekenis van deze for statement nauwkeurig beschreven.

Er zijn nog andere typen mogelijk.

for i := a while b do S

Hierin is a een arithmetische expressie, b een Boolean expressie.
Betekenis: bereken a en ken de waarde toe aan de lopende variabele i, bereken b. Voor b false zijn we klaar; voor b true wordt S uitgevoerd, waarna a opnieuw wordt berekend, de waarde van a aan i wordt toegekend, b opnieuw wordt berekend etc. (zie RR 4.6.4.3)

for i := a do S

a is een arithmetische expressie.
Betekenis: bereken a en ken de waarde toe aan i, voer de statement S uit.
We hebben drie soorten "elementen" gezien, namelijk het step-until element, het while-element en de arithmetische expressie. De algemeen for statement is van de vorm

for i := E_1, E_2, ..., E_n do S

Hierin stellen E_1, E_2, ..., E_n elementen voor. De elementen worden van links naar rechts afgewerkt.
Achter do mag maar één statement volgen. Zonodig kan men met de statement haken begin en end van een aantal statements een enkele statement maken.

Voorbeelden

for i := 1, 2, 4, 7, 11 do S

De statement S wordt uitgevoerd voor i = 1, 2, 4, 7 en 11.

for i := 3 step 2 until 10, 376 do S

De statement S wordt uitgevoerd voor i = 3, 5, 7, 9 en 376.

for x := 0 step 3.14 until 31.4 do s := s + s↑x

De lopende variabele x zij van het type real. De bedoeling is blijkbaar om de statement s := s + s↑x elf maal uit te voeren. De telling werkt hier met niet-gehele getallen, dus niet exact. In een binaire machine is 3.14 afgerond. Is dat een afronding naar boven, dan wordt de statement slechts tien maal uitgevoerd.

Men behoort deze for statement te schrijven in de vorm

for i := 0 step 1 until 10 do s := s + s ↑ (i × 3.14)

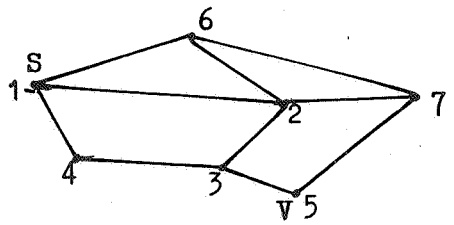
We weten nu zeker dat de telling in orde is.

2 De spin en de vlieg

Hoewel nog enkele belangrijke elementen van Algol 60 behandeld moeten worden, zullen we eerst ter illustratie een toepassing geven. Het gekozen proces is geen rekenprobleem, maar de gedachtengang die we moeten volgen om tot een goede beschrijving te komen is in principe dezelfde als bij een rekenprobleem.

Een spin (S) en een vlieg (V) zitten op een web.
Hoe moet de spin de vlieg vangen ?

We zullen het probleem eerst duidelijk formuleren. Onder een web verstaan we een graph. Een graph is een verzameling punten, genummerd van 1 tot n, waarbij elk tweetal punten al dan niet verbonden is.



In het voorbeeld zijn de punten 1 en 6 verbonden, de punten 1 en 3 niet.
S en V bevinden zich op de punten van de graph. S en V mogen om de beurt een zet doen, d.w.z. ze mogen zich van hun plaats naar een daarmee verbonden punt begeven
In het voorbeeld heeft S de keuze uit zetten naar 4, 2 en 6. V heeft de keuze uit zetten naar 3 en 7.
S heeft gewonnen als S en V zich tegelijk op hetzelfde punt bevinden.
Er zijn situaties waarin S, als hij aan zet is en goed speelt, V altijd kan vangen.
Er zijn ook situaties waarin, als V goed speelt, S de vlieg niet kan vangen.
V kan dus op pat hopen.

De opgave is nu bij een willekeurige graph voor S een strategie op te stellen die S in staat stelt in elke situatie waarin winst voor S mogelijk is, V ook inderdaad te vangen.

We gaan nu formaliseren.
Het web wordt beschreven met een Boolean array pad.

pad [i,j] is true als punt i en punt j verbonden zijn
en false als i en j niet verbonden zijn.

De positie van S en V wordt beschreven met 2 variabelen s en v van het type integer. De waarde van s resp. v is het nummer van het punt waar S resp. V zich bevinden.

Het integer array Zet dient om de strategie voor S te beschrijven.
Voor S aan zet geeft Zet [s,v] de winnende zet aan, als winst mogelijk is.

De waarde van Zet [s,v] is dus het nummer van het punt waar S naar toe moet gaan. Zet [s,v] is nul als er voor S in deze situatie (d.w.z. S op s en V op v) geen winnende zet is.

We hebben nog twee Boolean array's nodig, win en verlies.
Het array win geeft aan of er in een bepaalde situatie met S aan zet winst is voor S.
Het array verlies geeft aan of er in een bepaalde situatie met V aan zet verlies is voor V.

win [s,v] is true als S wint
false als S niet wint.

verlies [s,v] is true als V verliest
false als V niet verliest.

We nemen aan dat de gegevens van het web op een band staan. De band begint met n, het aantal punten van het web. Het web is gegeven door een n × n matrix van nullen en enen. De matrix staat rijsgewijs op de band. Een 1 in rij s en kolom v van de matrix geeft aan dat de corresponderende punten s en v verbonden zijn, een 0 geeft aan dat ze niet verbonden zijn.

In het nu volgende programma worden de array's Zet, win en verlies in een aantal stappen ingevuld.

```
begin integer n;
      n := read;
      begin integer s, t, v, w;
            Boolean klaar;
            integer array Zet [1 : n, 1 : n];
            Boolean array pad, win, verlies [1 : n, 1 : n];
            for s := 1 step 1 until n do
            for v := 1 step 1 until n do
            begin win [s,v] := verlies [s,v] := s = v;
                  pad [s,v] := read = 1;
                  Zet [s,v] := 0
            end;
            spin : klaar := true;
                  for s := 1 step 1 until n do
                  for v := 1 step 1 until n do
                  if ¬ win [s,v] then
                  begin for t := 1 step 1 until n do
                        if pad [s,t] ∧ verlies [t,v] then
                        begin klaar := false;
                              win [s,v] := true;
                              Zet [s,v] := t;
                              goto vangst
                        end;
          vangst : end;
                  if klaar then goto eind;
          vlieg : klaar := true;
                  for s := 1 step 1 until n do
                  for v := 1 step 1 until n do
                  if ¬ verlies [s,v] then
                  begin for w := 1 step 1 until n do
                        if pad [v,w] ∧ ¬ win [s,w] then
                        goto vrij;
                        klaar := false;
                        verlies [s,v] := true;
            vrij : end;
                  if ¬ klaar then goto spin;
          eind : for s := 1 step 1 until n do
                  for v := 1 step 1 until n do
                  print (Zet [ s,v ] )
      end
end
```

Het eerste deel van het programma tot aan de label spin verzorgt de initialisatie.
Eerst wordt n ingelezen. Aangezien de grenzen van de array's van n afhangen, moeten de array's gedeclareerd worden na het inlezen van n. Hiertoe is een binnenblok geïntroduceerd.
De array's win en verlies worden op een beginwaarde gezet. De elementen op de hoofddiagonaal worden true. We weten al zeker dat dit winst resp. verlies situaties zijn. Alle andere elementen worden voorlopig op false gezet.
We kunnen op dit moment nog geen winnende zetten aangeven, daarom wordt het array Zet voorlopig op nul gezet.
De variabele klaar wordt op true gezet. We zeggen dus meteen dat het proces is afgelopen. Als verderop blijkt dat het proces nog niet beeindigd is, wordt klaar op false gezet.

Tussen de label spin en de label vlieg beschouwen we de toestand als de spin aan zet is.
We gaan alle combinaties s, v langs. Vinden we in win dat een bepaalde situatie reeds als winst voor s is geboekt, dan behoeven we niets te doen. Is de situatie nog niet als winnend geboekt, dan gaan we alle mogelijke zetten van S na. Leidt een zet van S tot een nieuwe situatie die reeds als verliezend voor V in verlies is genoteerd, dan hebben we een winnende zet voor S gevonden. Dit wordt geboekt in win en Zet.

Tussen de label vlieg en vrij wordt de toestand beschouwd als de vlieg aan zet is.
Alle combinaties s, v worden bekeken. Als een bepaalde situatie reeds als verlies voor V is geboekt, behoeven we niets te doen. In het andere geval gaan we alle mogelijke zetten van V na. Zijn alle nieuwe situaties die zo kunnen ontstaan als winnend voor S geboekt, dan hebben we een verlies situatie voor V gevonden. Dit wordt genoteerd in verlies.

Het proces eindigt als een van de array's win of verlies niet meer verandert.

3. Algol 60 (vervolg)

RR 5. Declarations

De statement haken begin en end worden gebruikt om van een groepje statements een compound statement te maken. We zijn dit tegengekomen bij de if statement en de for statement.
Een blok is eveneens een groepje statements, omvat door de symbolen begin en end, waarbij echter tussen begin en de eerste statement van het groepje een aantal declaraties voorkomt.
Een blok is zelf weer een statement.

```
begin integer i,j;
      real x,y,z;
      Boolean b1,b2;
      integer array C[1:10];
      real array A, B[1:10, n:m];
      S; S; ...; S;
end
```

De declaratie "integer i,j" betekent dat de identifiers i en j in dit blok gebruikt zullen worden om variabelen van het type integer aan te duiden. Men noemt i en j locale variabelen van het blok.

We zien hier een nieuwe functie van begin en end. Zij geven het gebied aan waarin een declaratie in principe geldt, n.l. vanaf een begin tot aan de corresponderende end. We maken hier een voorbehoud, omdat het mogelijk is in een binnenblok aan identifiers tijdelijk een andere betekenis toe te kennen.

De declaraties "real x, y, x" en "Boolean b1, b2" geven aan dat x,y,z resp. b1, b2 in dit blok gebruikt worden om variabelen van het type real resp. Boolean aan te duiden.

De eerste drie declaraties zijn voorbeelden van type declaraties. Daarna volgen twee array declaraties. De identifier C wordt in het blok gebruikt om een een-dimensionaal array aan te duiden, waarvan de elementen geindiceerde variabelen zijn van het type integer. Tevens is in de declaratie aangegeven dat de index kan lopen van 1 tot 10.

Als index grenzen zijn willekeurige expressies toegestaan. De declaratie van de twee-dimensionale array's A en B geeft hiervan een voorbeeld. De index grenzen worden bij blok binnenkomst uitgerekend. Zij kunnen slechts afhangen van variabelen die non-locaal zijn t.o.v. het blok, d.w.z. die gedeclareerd zijn in een omvattend blok.

```
begin integer i,j,n,m;
      n := 1; m := 3;
      i := 8; j := 9;
      begin integer i,j;
            real x,y,z;
            integer array C[1:10];
            real array A,B[1:10,n:m];
            ...; ...;
      end;
      ...; ...;
end
```

De array elementen A[k,ℓ] zijn gedefinieerd voor $1 \leq k \leq 10$ en $1 \leq \ell \leq 3$. Merk op dat de identifier i in het binnenblok een andere variabele aanduidt dan in het buitenblok. Evenzo de identifier j.

We beschouwen nog het volgende voorbeeld.

```
begin integer i,j;
      i := 1; j := 3;
      begin integer i,j;
            real array A[i:j];
            ...; ...;
      end
      ...; ...;
end
```

In de array declaratie zijn i en j de integer variabelen uit het omvattende blok, die resp. de waarde 1 en 3 hebben gekregen.
Opmerking : het is niet geheel duidelijk of dit correct Algol is (vergelijk RR 5.2.4.2 met de tweede alinea van RR 5).

We behandelen nu de procedure declaratie.

```
begin integer piet; real jan;
      procedure P(x,y,z); value y;
      real x; integer y; procedure z;
      begin ...; ...; x := 3.14; y := 3; ...;
            ...; z; ...
      end;
      ...; P(jan,piet,f); ...
end
```

De procedure declaratie bestaat uit een aantal onderdelen.
P is de naam van de procedure; x,y en z zijn de formele parameters.
Na de formele parameters volgen de value lijst en de specificaties.
De specificaties geven informatie over de soort en het type van de formele parameters. Daarna volgt een statement, de procedure body; in het voorbeeld zijn begin en end weer gebruikt om van een groepje statements één statement te maken.
Door P(jan,piet,f) wordt in het programma de procedure aangeroepen met de actuele parameters jan, piet en f. De parameters jan en piet zijn variabelen, gedeclareerd in hetzelfde blok als P, de parameter f is de naam van een procedure, die elders in het programma is gedeclareerd.
De procedure P wordt hier gebruikt als een zelfstandig statement, een procedure statement.

De betekenis van de procedure statement kan men als volgt beschrijven.
Van de body wordt een copie gemaakt. In de copie vervangen we de formele parameter x door de naam jan en de formele parameter z door de naam f.
De formele parameter y komt in de value lijst voor en wordt daarom anders behandeld. We creëren een nieuw blok dat de body omvat, het fictieve blok. In dit blok wordt een locale variabele y volgens de gegeven specificatie gedeclareerd en de waarde van piet wordt aan y toegekend.
De copie van de body plus het fictieve blok ziet er nu als volgt uit:

```
begin integer y; y := piet;
      begin ...; ...; jan := 3.14; y := 3; ...;
            ...; f; ...
      end
end
```

Tenslotte vervangen we de procedure statement P(jan,piet,f) door dit stukje programma en voeren het uit.

We zien dat er twee manieren zijn om een formele parameter te vervangen door een actuele, en wel door vervanging van namen (call by name) of door toekenning van de waarde van de actuele parameter aan de formele parameter (call by value).

Voorbeeld

De volgende procedure berekent het scalaire product van twee vectoren u en v (indices lopend van 1 tot n) en kent de waarde aan z toe.

```
procedure scapro (u,v,n,z); value n; integer n;
array u,v; real z;
begin integer k; z := 0;
      for k := 1 step 1 until n do
      z := z + u[k]×v[k]
end
```

In het programma kan nu bijvoorbeeld de procedure statement scapro (vec,tor,10,sp) voorkomen.
Als deze procedure statement aan bod is, wordt uitgevoerd:

```
begin integer n; n := 10;
      begin integer k; sp := 0;
            for k := 1 step 1 until n do
            sp := sp + vec[k] × tor[k]
      end
end
```

De procedure scapro is correct, maar de uitvoering vergt meer tijd dan nodig is.
De vervanging van de formele parameters u, v en z door vec, tor en sp, en de vervanging van de procedure statement door de copie doen we slechts in gedachte. In feite zal de procedure behandeld worden als een soort subroutine. In het geval van een call by name zal in de body bij de formele parameter dan de aanwijzing "let op, kijk naar buiten" moeten staan. Dit kost tijd. Het is voordeliger om met een locale variabele te werken. We zullen daarom in de body niet met z werken, maar een locale variabele s invoeren.

```
procedure scapro (u,v,n,z); value n; integer n;
array u,v; real z;
begin integer k; real s; s := 0;
      for k := 1 step 1 until n do
      s := s + u[k] × v[k]; z := s
end
```

Er is een ander bezwaar tegen de procedure aan te voeren.
Als actuele parameters kan men voor u en v slechts echte vectoren geven. Het is niet mogelijk met deze procedure het scalair product te bepalen van een rij van een matrix met een kolom van een matrix. Men zou eerst de rij en de kolom in echte vectoren moeten overnemen. Om aan dit bezwaar tegemoet te komen kunnen we de procedure als volgt modificeren:

```
procedure scapro (k,u,v,n,z); value n;
integer k,n; real u,v,z;
begin real s; s := 0;
      for k := 1 step 1 until n do
      s := s + u × v; z := s
end
```

Voorbeelden van aanroepen:

```
scapro (i,vec[i],tor[i],10,sp)
scapro (i,A[k,i],B[i,m],25,x)
```

We zien hier de macht van het call by name mechanisme. De formele parameters u en v mogen niet alleen door een naam worden vervangen, het is zelfs toegestaan u en v door hele expressies te vervangen. Dit moet dan uiteraard wel tot correct Algol leiden (zie RR 4.7.3.2 en 4.7.5).

In de vorige versie was de teller k een locale variabele van de body. In de nieuwe versie is k naar buiten gehaald, het is een formele parameter geworden. Dit schept de mogelijkheid voor u en v expressies te geven die afhangen van de met k corresponderende actuele parameter. De procedure beschikt daardoor over de formele parameters u en v alsof het functies van k zijn. Men noemt dit gebruik van formele parameters Jensen's device. Het is een uiterst nuttig hulpmiddel om programma's korter en duidelijker te maken.

Een enkele opmerking over de parameter n. In de for statement hebben we de waarde van de met n corresponderende actuele parameter herhaaldelijk nodig. Aangezien deze actuele parameter een ingewikkelde uitdrukking kan zijn, is het zinvol n in de value lijst op te nemen. De uitdrukking wordt dan slechts eenmaal uitgerekend.

We zijn nog niet tevreden met scapro. Dat de waarde van het scalair product in bovenstaande aanroepen toegekend wordt aan de variabele sp resp. x interesseert ons niet zo zeer. Vaak zullen we deze waarde direct in een uitdrukking willen gebruiken.
We kunnen dit realiseren door de declaratie

```
real procedure scparo (k,u,v,n); value n;
integer k,n; real u,v;
begin real s; s := 0;
      for k := 1 step 1 until n do
      s := s + u×v;
      scapro := s
end
```

Voorbeeld van een aanroep:

```
x := 5×scapro (i,A[k,i],B[i,m],25) + 13.4
```

We gebruiken hier scapro (i,A[k,i],B[i,m],25) niet als zelfstandig statement, maar als functie designator. Een functie designator heeft zelf een waarde, die uitgerekend wordt m.b.v. de procedure.
Dat een procedure in expressies aangeroepen wordt komt in de declaratie op twee manieren tot uiting.

1. De procedure declaratie begint met een van de symbolen integer, real of Boolean.
   Hiermede wordt aangegeven van welk type de waarde van de functie designator is.
2. In de body van de procedure wordt aan de procedure identifier een waarde toegekend.

Ter illustratie zullen we nog enkele procedures bekijken.

```
integer procedure fac (n); value n; integer n;
fac := if n = 0 then 1 else n×fac (n-1)
```

Dezeprocedure berekent n! We nemen aan dat de actuele parameter integer is en niet negatief.
In de body roept de procedure fac zichzelf aan. Dit is een voorbeeld van het recursief gebruik van een procedure, hetgeen in Algol is toegestaan. Overigens is dit een weinig efficiente manier om n! te berekenen.

Nu een procedure ter berekening van $\sum_{k=a}^{b} f(k)$.

```
real procedure Sum (k,a,b,fk); value a,b;
integer a,b,k; real fk;
if b < a then Sum := 0
        else begin k := a;
                  Sum := fk + Sum (k,a+1,b,fk)
             end
```

Voorbeeld van een aanroep:

```
Sum (i,0,2 ↑ 27 + 8,1/(i+j) ↑ 2).
```

Hiermee wordt berekend $\sum_{i=0}^{2\uparrow 27+8} \frac{1}{(i+j)^2}$.

We zien hier een voorbeeld van Jensen's device, alsook van het recursief gebruik van een procedure. Ook deze procedure is praktisch onbruikbaar. We geven nog een andere procedure voor Sum.

```
real procedure Sum (k,a,b,fk); value b;
integer a,b,k; real fk;
begin real s; s := 0;
      for k := a step 1 until b do
      s := s + fk;
      Sum := s
end
```

Deze procedure werkt vlugger en kost minder geheugenruimte dan de vorige. In het Algol rapport wordt een aantal standaard functies genoemd die gebruikt mogen worden zonder expliciete declaraties. Voor enkele standaard functies zullen we ter oefening een declaratie opschrijven.

```
real procedure abs(E); value E; real E;
abs := if E ≥ 0 then E else -E
```

De absolute waarde van een integer is van het type real.

```
integer procedure sign(E); value E; real E;
sign := if E > 0 then 1 else
        if E = 0 then 0 else -1
```

Onder het entier van E verstaan we het grootste gehele getal dat niet groter is dan E.

```
integer procedure entier(E); value E; real E;
entier := if 0 ≤ E ∧ E < 1 then 0 else
          if E ≥ 1 then 1 + entier (E-1) else
                        -1 + entier (E+1)
```

Deze definitie is niet correct. De berekening van E-1 en E+1 wordt uitgevoerd in real arithmetiek, dus niet exact.
Wat is nu het resultaat als E bijv. zeer groot is ?
Het is beter een integer op te bouwen (dit kan exact) en te kijken of we E reeds gepasseerd zijn.

```
integer procedure entier(E); value E; real E;
begin integer k; k := 0;
      if E ≥ k then begin L1 : k := k + 1;
                          if k ≤ E then goto L1;
                          k := k-1
                    end
               else begin L2 : k := k - 1;
                          if k > E then goto L2
                    end;
      entier := k
end
```

Dit proces is correct, maar niet efficient.
Voor $E = 10^{10}$ komen we niet klaar.
We eisen van onze processen niet alleen dat ze correct zijn, maar ook efficient.

```
integer procedure entier(E); value E; real E;
begin integer h,j,k; Boolean pos;
      pos := E ⩾ 1;
      if ¬pos ∧ E ⩾ 0 then begin entier := 0; goto end end;
      h := if pos then 1 else -1;
  L1: k := h; h := 2×h;
      if h ⩽ E ≡ pos then goto L1;
      if pos then h := k else begin k := h; h := -k end;
  L2: h := h ÷ 2; j := k + h;
      if j ⩽ E then k := j;
      if h > 1 then goto L2;
      entier := k;
end:
end
```

Dit proces is veel efficienter doordat we een integer opbouwen met stappen 1,2,4,8,16 etc. of -1,-2,-4,-8,-16 etc.

De details van het proces maakt men zich het beste duidelijk aan de hand van enige voorbeelden.

4. Euler transformatie

Bijna alle numerieke werk berust op extrapolatie, omdat men te maken heeft met discrete voorstellingen van oneindige processen uit de analyse. De vraag is hoe men uit een paar schattingen een redelijke extrapolatie vindt naar bijv. stapgrootte nul of oneindig veel termen. We beschouwen om te beginnen het sommeren van een oneindige reeks $\Sigma\, u_k$. Men zou partiële sommen kunnen uitrekenen, tot binnen de vereiste precisie geen verandering meer optreedt. Voor snel convergente reeksen is dat mogelijk. In het algemeen is het een weinig efficiënt proces.
De Euler transformatie is een methode om een bepaald type oneindige reeks te sommeren.
We leiden de desbetreffende formule af door formeel met operatoren te rekenen. Hiervoor hebben we nodig de verschuivingsoperator E en de voorwaartse middelingsoperator M (hoofdletter μ).

$$E\, f_k \overset{\text{def.}}{=} f_{k+1}$$

$$M\, f_k \overset{\text{def.}}{=} \tfrac{1}{2}\,(f_k + f_{k+1})$$

Nu is $\quad M\, f_k = \tfrac{1}{2}\,(f_k + E\, f_k) = \tfrac{1}{2}\,(1 + E)\, f_k$

dus $\quad M = \tfrac{1}{2}\,(1 + E)$

en omgekeerd

$$E = 2\,M - 1.$$

Beschouw de reeks $\sum_{k=0}^{\infty} u_k$.

$$u_0 = E^0\, u_0$$

$$u_1 = E^1\, u_0$$

$$u_2 = E\, u_1 = E\, E\, u_0 = E^2\, u_0$$

$$u_k = E^k\, u_0$$

Formeel kan men schrijven

$$\sum_{k=0}^{\infty} u_k = \sum_{k=0}^{\infty} E^k\, u_0 = \left(\sum_{k=0}^{\infty} E^k\right) u_0.$$

Bewijs zelf:

$$M^k\, u_0 = 2^{-k} \sum_{j=0}^{k} \binom{k}{j}\, u_j.$$

De formule van Euler vindt men nu als volgt:

$$\sum_{k=0}^{\infty} u_k = \left(\sum_{k=0}^{\infty} E^k\right) u_o = \frac{1}{1-E} u_o = \frac{1}{2}\frac{1}{1-M} u_o = \frac{1}{2}\sum_{k=0}^{\infty} M^k u_o .$$

Als Σu_k convergeert, dan convergeert $\Sigma M^k u_o$ ook en wel naar dezelfde som. Dit hebben we thans uiteraard nog niet bewezen. Het bewijs van deze stelling volgt later.

Men kan de gemiddelden rangschikken in een driehoekig schema. Men berekent een kolom, door telkens het gemiddelde te nemen van twee opeenvolgende elementen van de vorige kolom.

$$\begin{array}{llll}
u_o & & & \\
 & M u_o & & \\
u_1 & & M^2 u_o & \\
 & M u_1 & & M^3 u_o \\
u_2 & & M^2 u_1 & \\
 & M u_2 & & \\
u_3 & & &
\end{array}$$

We kunnen nu wel inzien dat de Euler transformatie een geschikte sommatie methode is voor een alternerende reeks die slecht convergeert. Immers de gemiddelden $M u_o$, $M u_1$, $M u_2$ etc. zullen dan kleiner zijn dan de elementen uit de eerste kolom en sneller tot nul naderen. Als de tweede kolom ook weer alterneert, zullen $M^2 u_o$, $M^2 u_1$, nog kleiner zijn, etc.

Voorbeeld

We beschouwen de meetkundige reeks $\sum_{k=0}^{\infty} r^k$.

$$\begin{array}{llll}
1 & & & \\
 & \frac{1+r}{2} & & \\
r & & \left(\frac{1+r}{2}\right)^2 & \\
 & \frac{1+r}{2} r & & \left(\frac{1+r}{2}\right)^3 \\
r^2 & & \left(\frac{1+r}{2}\right)^2 r & \\
 & \frac{1+r}{2} r^2 & & \left(\frac{1+r}{2}\right)^3 r \\
r^3 & & \left(\frac{1+r}{2}\right)^2 r^2 & \\
 & \frac{1+r}{2} r^3 & & \\
r^4 & & &
\end{array}$$

De getransformeerde reeks is $\sum_{k=0}^{\infty} \left(\frac{1+r}{2}\right)^k$.

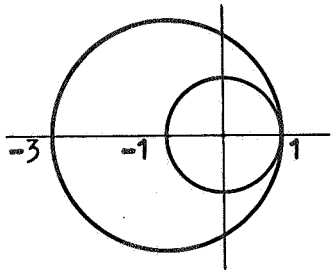
Men kan hier gemakkelijk verifiëren dat voor $|r| < 1$

$$\sum_{k=0}^{\infty} r^k = \frac{1}{2} \sum_{k=0}^{\infty} \left(\frac{1+r}{2}\right)^k .$$

De reeks in het linkerlid convergeert voor $|r| < 1$ en heeft als som $\frac{1}{1-r}$.

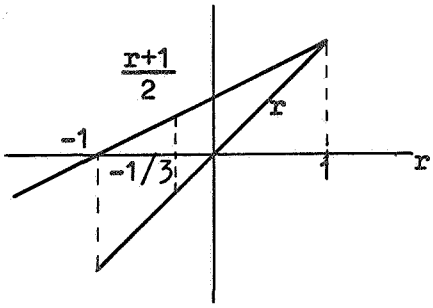
De reeks in het rechterlid convergeert voor $\left|\frac{1+r}{2}\right| < 1$ en heeft als som ook $\frac{1}{1-r}$.

In het complexe r-vlak hebben we het volgende beeld:



Het convergentie gebied van de getransformeerde reeks omvat het convergentie gebied van Σr^k. Ook als de laatste reeks divergent is, is het mogelijk dat de getransformeerde reeks convergeert.

Als we numeriek wilden sommeren zouden we de voorkeur geven aan de reeks met de in absolute waarde kleinste reden.
We gaan dit na voor reële r.



Uit de figuur zien we dat $\left|\frac{r+1}{2}\right| < |r|$ voor $-1 < r < -1/3$.
In dit gebied convergeert de getransformeerde reeks sneller.
Deze reeks convergeert zelfs bijzonder goed als r dicht bij -1 ligt, dus als Σr^k een slecht convergente alternerende reeks is.

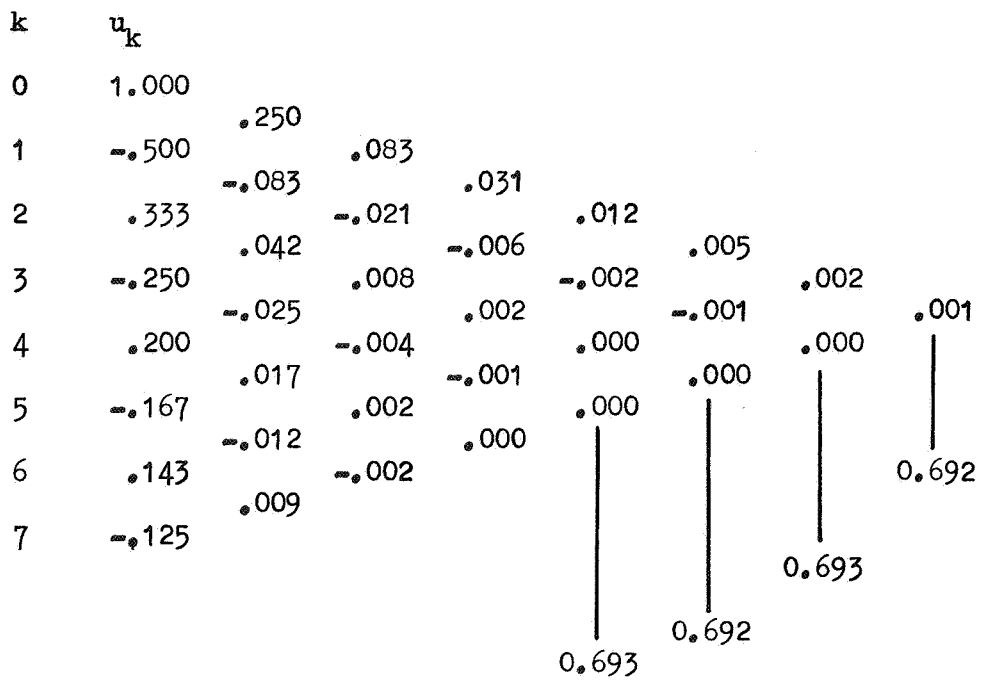
Voorbeeld

$$\ln (1 + x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots$$

We mogen x = 1 invullen (stelling van Abel):

$$\ln 2 = 1 - 1/2 + 1/3 - 1/4 + \ldots$$

Deze reeks convergeert slecht, de verhouding van twee opeenvolgende termen nadert tot -1. Om de som in 3 decimalen te bepalen hebben we 2000 termen nodig. We gaan daarom de reeks "euleren".



We vinden $\ell n\ 2 = \frac{1}{2}\ (1.000 + .250 + .083 + ...) = 0.692$

Kunnen we het nog beter doen ?
We merken op dat de tweede diagonaal van boven sneller convergeert dan de eerste.

Hiervan kan men gebruik maken door te schrijven
$\ell n\ 2 = 1 + (-1/2 + 1/3 - 1/4 + \ldots)$.
De reeks tussen haakjes wordt geeulerd. Men begint dan pas na de eerste term van de reeks te euleren. In het gegeven voorbeeld is het nog beter om pas na twee termen te euleren.

We willen zo weinig mogelijk rekenen. Welke strategie kunnen we volgen om dat te bereiken ?
Het is in ieder geval zaak om op een snel convergerende diagonaal terecht te komen.
We bouwen het schema nogmaals op.

1.000
.250
-.500

Het gemiddelde .250 is kleiner dan .500; we accepteren .250 en gaan verder langs de bovenste diagonaal.

1.000
.250
-.500 .083
-.083
.333

Nu is abs (.083) niet kleiner dan abs (-0.083).
We vergeten daarom de eerste diagonaal verder en stappen over op de tweede m.a.w. we euleren nu na de eerste term.

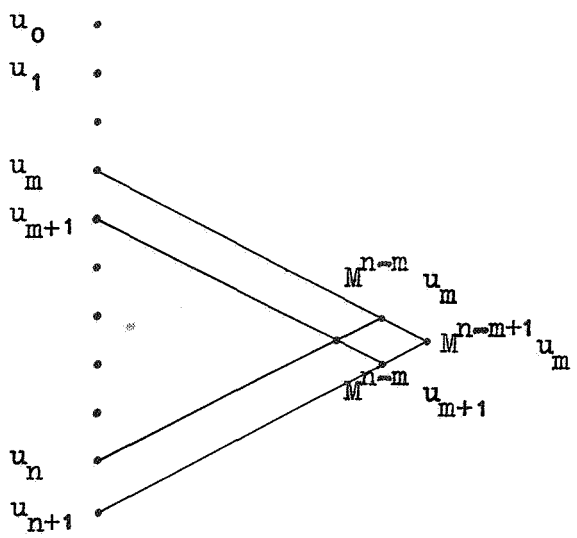
1.000
.250
-.500
-.083
.333 -.021
.042
-.250

Nu is 0.021 kleiner dan .042; we accepteren het gemiddelde -.021 en gaan verder lang de tweede diagonaal.
Op deze manier doorgaand krijgt men tenslotte het volgende schema.

k	u_k				
0	1.000				
		.250			
1	-.500				
		-.083			
2	.333		-.021		
		.042		-.006	
3	-.250		.008		
		-.025		.002	
4	.200		-.004		.000
		.017		-.001	
5	-.167		.002		
		-.012			
6	.143				
7	-.125				

$\ell n\ 2 = 1.000 - .500 + \frac{1}{2}(.333 + .042 + .008 + .002) = .692$

We merken op dat nog een hele driehoek van getallen onthouden moet worden, omdat de eigenlijke sommatie tot het eind is uitgesteld.
Tijdens de opbouw van het schema zijn bij elke nieuwe term u_k van de reeks slechts de getallen van de onderste rij nodig om nieuwe gemiddelden te berekenen.
Kunnen we niet volstaan met het onthouden van deze rij ?
Dit is mogelijk als men tijdens de opbouw reeds kan sommeren.



Als we na m termen euleren is de transformatie formule

$$\sum_{k=o}^{\infty} u_k = \sum_{k=o}^{m-1} u_k + \tfrac{1}{2} \sum_{k=o}^{\infty} M^k u_m .$$

Stel dat de gemiddelden tot $M^{n-m} u_m$ zijn uitgerekend.

Dan is $\sum_{k=o}^{n} u_k$ = (in de zin van Euler) = $\sum_{k=o}^{m-1} u_k + \tfrac{1}{2} \sum_{k=o}^{n-m} M^k u_m = S_{m,n}$

We berekenen nu een nieuwe rij gemiddelden, dus u_{n+1}, $M u_n$, $M^2 u_{n-1}$, ..., $M^{n-m+1} u_m$.

$M^{n-m+1} u_m$ wordt geaccepteerd als $|M^{n-m+1} u_m| < |M^{n-m} u_{m+1}|$; als $M^{n-m+1} u_m$ wordt verworpen gaan we een plaats later euleren.

Om tijdens de opbouw te kunnen sommeren moeten we het verband weten tussen $S_{m,n}$, $S_{m,n+1}$ en $S_{m+1,n+1}$.

$$S_{m,n+1} = S_{m,n} + \tfrac{1}{2} M^{n-m+1} u_m \qquad (4.1)$$

$$S_{m+1,n+1} = S_{m,n} + u_m - \tfrac{1}{2} \sum_{k=o}^{n-m} M^k u_m + \tfrac{1}{2} \sum_{k=o}^{n-m} M^k u_{m+1}$$

We schrijven dit laatste uit

$$S_{m+1,n+1} = S_{m,n} + u_m - \tfrac{1}{2} u_m - \tfrac{1}{2} M u_m - \tfrac{1}{2} M^2 u_m - \ldots - \tfrac{1}{2} M^{n-m} u_m$$

$$+ \tfrac{1}{2} u_{m+1} + \tfrac{1}{2} M u_{m+1} + \tfrac{1}{2} M^2 u_{m+1} + \ldots + \tfrac{1}{2} M^{n-m} u_{m+1}$$

Men kan nu de termen op een handige manier samen nemen.

$$u_m - \tfrac{1}{2} u_m + \tfrac{1}{2} u_{m+1} = M u_m$$

$$M u_m - \tfrac{1}{2} M u_m + \tfrac{1}{2} M u_{m+1} = M^2 u_m$$

etc.

Zo vindt men

$$S_{m+1,n+1} = S_{m,n} + M^{n-m+1} u_m \qquad (4.2)$$

De procedure is nu eenvoudig geworden. Als $M^{n-m+1} u_m$ wordt geaccepteerd, dan wordt de helft van dit gemiddelde bij de som geboekt.
Als $M^{n-m+1} u_m$ wordt verworpen, dan wordt het gemiddelde in zijn geheel bij de som geboekt en verder weggelaten.

We geven nogmaals de berekening.

k	u_k	M u_{k-1}				S
						0.000
0	1.000					0.500
1	-.500	.250				0.625
2	.333	-.083	.083			0.708
3	-.250	.042	-.021			0.698
4	.200	-.025	.008	-.006		0.695
5	-.167	.017	-.004	.002	-.002	0.693
6	.143	-.012	.002	-.001	.000	0.693

De omlijnde gemiddelden zijn verworpen.
Voor de berekening van een rij in het schema zijn slechts de gegevens van de vorige rij nodig.

We merken op dat de formules (4.1) en (4.2) voor de numericus even essentieel zijn als de transformatie formule zelf. Zij vormen het gereedschap waarmee het transformatie proces efficiënt uitgevoerd kan worden.

We laten nu het Algol programma volgen.
Men vergelijke dit met example 1 uit het Algol rapport. In het onderstaande programma is gebruik gemaakt van Jensen's device.

```
real procedure euler (k, uk, eps, tim); value eps, tim;
integer k, tim; real uk, eps;
begin integer i, n, t; array m [0 : 15];
      real mn, mp, ds, sum;
      k := n := t := 0 ; m [0] := uk;
      sum := m [0] /2;
next term: k := k + 1; mn := uk;
      for i := 0 step 1 until n do
      begin mp := (mn + m [i] ) /2;
            m [i] := mn; mn := mp
      end means;
      if (abs (mn) < abs (m [n ])) ∧ (n < 15)
         then begin ds := mn /2; n := n + 1; m [n] := mn
              end accept
         else ds := mn;
      sum := sum + ds;
      if abs (ds) < eps then t := t + 1 else t := 0;
      if t < tim then goto next term;
      euler := sum
end euler
```

De parameter eps is de tolerantie. Aan het eind van het programma wordt getest of de toename ds van de som in absolute waarde kleiner is dan eps. De parameter tim geeft aan hoe vaak in successie dat moet gebeuren voor men in het resultaat gelooft.
De rij gemiddelden is opgeborgen in het array m. Dit array bevat 16 elementen. Als het over dreigt te lopen wordt het nieuwe gemiddelde mn niet geaccepteerd.

Opmerking

Men kan de Euler transformatie op een andere manier beschouwen door uit te gaan van de partiële sommen u_o, $u_o + u_1$, $u_o + u_1 + u_2$ enz. van Σu_k.

u_o	$\frac{1}{2} u_o + M u_o$	$\frac{1}{2} u_o + \frac{1}{2} M u_o + M^2 u_o$
$u_o + u_1$	$\frac{1}{2} u_o + M u_o + M u_1$	$\frac{1}{2} u_o + \frac{1}{2} M u_o + M^2 u_o + M^2 u_1$
$u_o + u_1 + u_2$	$\frac{1}{2} u_o + M u_o + M u_1 + M u_2$	.
$u_o + u_1 + u_2 + u_3$	.	.
.	.	.
.	.	.
.	.	.

Uit de kolom van partiële sommen is een kolom gemiddelden berekend, uit deze kolom weer een nieuwe kolom gemiddelden etc.
Als Σu_k convergeert, dan convergeren de kolommen ook.
We beweren dat ook de rijen convergeren. We merken hiervoor op dat $\lim_{n\to\infty} M^n u_k = 0$, immers de reeks $\sum_{n=o}^{\infty} M^n u_k$ is convergent.
Een gemiddelde in de k-de rij ziet er als volgt uit

$$\frac{1}{2} u_o + \frac{1}{2} M u_o + .. + \frac{1}{2} M^n u_o + M^{n+1} u_o + .. + M^{n+1} u_k$$

Na de eerste n + 1 termen van de Euler som volgen nog k + 1 termen die naar nul gaan voor n naar oneindig.

We passen dit toe op ons voorbeeld.

1.000				
	.750			
.500		.708		
	.667		.698	
.833		.687		.694
	.708		.691	
.583		.695		
	.683			
.783				

We hebben hier geen goede strategie om van de ene diagonaal op de andere over te gaan. Hoewel de methode fraai is vermelden we hem daarom slechts terloops.

Bij het nu volgende bewijs van de formule van Euler zullen we gebruik

maken van hulpmiddelen uit de complexe functietheorie.
We brengen eerst een stelling uit de functietheorie in herinnering.
De som van een machtreeks $\Sigma u_k z^k$ is binnen de convergentiecirkel een reguliere functie van z en op de cirkel ligt tenminste één singulier punt van deze functie (zie syllabus part. diff. verg. pag. 100).

We gaan uit van een willekeurige machtreeks $\sum_{k=o}^{\infty} u_k z^k$ met convergentiestraal ongelijk nul.
Dan is $F(z) = \Sigma u_k z^k$ regulier in een omgeving van de oorsprong.

We kiezen een punt $z_o \neq 0$ in het complexe vlak en voeren een nieuwe variabele t in:

$$t = \frac{2z}{z + z_o}$$

Oplossen van t geeft

$$z = \frac{z_o t}{2 - t} .$$

Voor $|z|$ klein is $|t|$ ook klein.
In een omgeving van de oorsprong is t een reguliere functie van z.

De nu volgende manipulatie met machtreeksen is geoorloofd en levert een resultaat dat goed is voor $|z|$ klein genoeg.

$$\sum_{k=o}^{\infty} u_k z^k = \sum_{k=o}^{\infty} u_k z_o^k \left(\frac{t}{2-t}\right)^k = \frac{z_o}{z} \sum_{k=o}^{\infty} u_k z_o^k \left(\frac{t}{2-t}\right)^{k+1} =$$

$$= \frac{z_o}{2z} \sum_{k=o}^{\infty} u_k z_o^k 2^{-k} t^{k+1} \left(1 - \frac{t}{2}\right)^{-k-1} =$$

$$= \frac{z_o}{2z} \sum_{k=o}^{\infty} u_k z_o^k t^{k+1} 2^{-k} \sum_{h=o}^{\infty} \binom{-k-1}{h} \left(-\frac{t}{2}\right)^h$$

$$= \frac{z_o}{2z} \sum_{k=o}^{\infty} u_k z_o^k t \sum_{h=o}^{\infty} t^{k+h} 2^{-(k+h)} \binom{k+h}{h} .$$

We stellen $k + h = \ell$ en merken op dat $\binom{k+h}{h} = \binom{k+h}{k}$.

$$\sum_{k=o}^{\infty} u_k z^k = \frac{z_o}{2z} \sum_{k=o}^{\infty} u_k z_o^k t \sum_{\ell=k}^{\infty} \binom{\ell}{k} 2^{-\ell} t^{\ell}$$

$$= \frac{z_o}{2z} \sum_{\ell=o}^{\infty} t^{\ell+1} 2^{-\ell} \sum_{k=o}^{\ell} \binom{\ell}{k} u_k z_o^k .$$

Nu is $2^{-\ell} \sum_{k=o}^{\ell} \binom{\ell}{k} u_k z_o^k = M^{\ell} (u_k z_o^k)_{k=o}$.

We vinden tenslotte

$$\sum_{k=0}^{\infty} u_k z^k = \frac{z_o}{2z} \sum_{\ell=0}^{\infty} M^{\ell}(u_k z_o^k)_{k=0} \left(\frac{2z}{z + z_o}\right)^{\ell+1}.$$

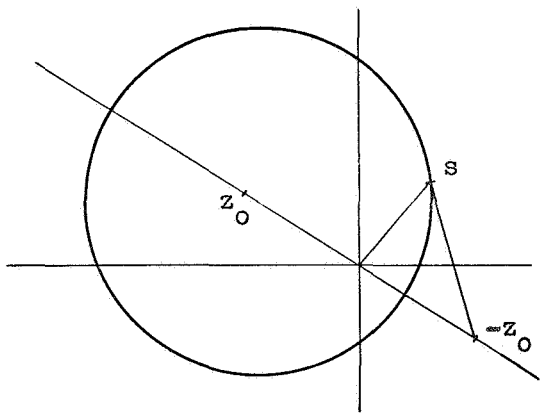
In de omgeving van de oorsprong is de som van beide reeksen gelijk aan $F(z)$.
Voor welke waarden van z convergeert de getransformeerde reeks ?
Het is een machtreeks in $t = \frac{2z}{z + z_o}$, in het convergentiegebied is de som dus overal gelijk aan de reguliere functie $F(z)$.
Als machtreeks in t beschouwd moet er op de convergentiecirkel een singulier punt t_o liggen. Dit punt correspondeert met een singulariteit van $F(z)$ en wel met die singulariteit s waarvoor $\left|\frac{2s}{s + z_o}\right|$ minimaal is.
Dan is dus $|t_o| = \left|\frac{2s}{s + z_o}\right|$ en de reeks in het rechterlid convergeert voor

$$\left|\frac{z}{z + z_o}\right| < \left|\frac{s}{s + z_o}\right|.$$

Er is een divergentie voor $\left|\frac{z}{z + z_o}\right| > \left|\frac{s}{s + z_o}\right|$.

Kritiek is het geval $\left|\frac{z}{z + z_o}\right| = \left|\frac{s}{s + z_o}\right|$.

Hoe ziet de rand van het convergentiegebied eruit ?
Meetkundig geïnterpreteerd staat er dat de verhouding van de afstand van z tot de oorsprong en de afstand van z tot het punt $-z_o$ constant is.
Uit de meetkunde is bekend dat de punten, waarvan de verhouding van de afstanden tot twee vaste punten constant is, op een cirkel liggen, de Apollonius cirkel.



In de figuur is het inwendige van de cirkel het convergentiegebied van de getransformeerde reeks, de oorsprong ligt binnen de cirkel.
Als de oorsprong buiten de cirkel ligt is het uitwendige van de cirkel het convergentiegebied.

We nemen aan dat $F(z)$ een singulariteit heeft in oneindig. De laatst genoemde situatie (het uitwendige van een Apollonius cirkel als convergentiegebied) kan zich nu niet meer voordoen. In deze situatie is $|\frac{2s}{s + z_o}| > 2$; de singulariteit op oneindig neemt daarom de rol van s over.

De constructie van een Apollonius cirkel is eenvoudig.
Trek de binnen- en de buitenbissectrice van de hoek gevormd door de verbindingslijnen van de oorsprong naar s en van $-z_o$ naar s en bepaal de snijpunten met de rechte door z_o en $-z_o$. Dit zijn twee punten van de cirkel. Verder ligt het middelpunt van de cirkel ook op de rechte door z_o en $-z_o$.

We hadden de algemene transformatie formule

$$\sum_{k=o}^{\infty} u_k z^k = \frac{z_o}{z + z_o} \sum_{\ell=o}^{\infty} M^{\ell}(u_k z_o^k)_{k=o} \left(\frac{2z}{z + z_o}\right)^{\ell} . \qquad (4.3)$$

De klassieke Euler transformatie vindt men door $z = z_o$ te nemen.

$$\sum_{k=o}^{\infty} u_k z^k = \tfrac{1}{2} \sum_{\ell=o}^{\infty} M^{\ell}(u_k z^k)_{k=o} .$$

We bepalen het convergentiegebied van de getransformeerde reeks.
Neem een singulariteit s_1.
Voor de convergentie van de geeulerde reeks is nodig

$$|\frac{2z}{z + z_o}| < |\frac{2s_1}{s_1 + z_o}|$$

of wegens $z = z_o$

$$1 < |\frac{2s_1}{s_1 + z}|$$

of $$|z - (-s_1)| < 2|s_1| .$$

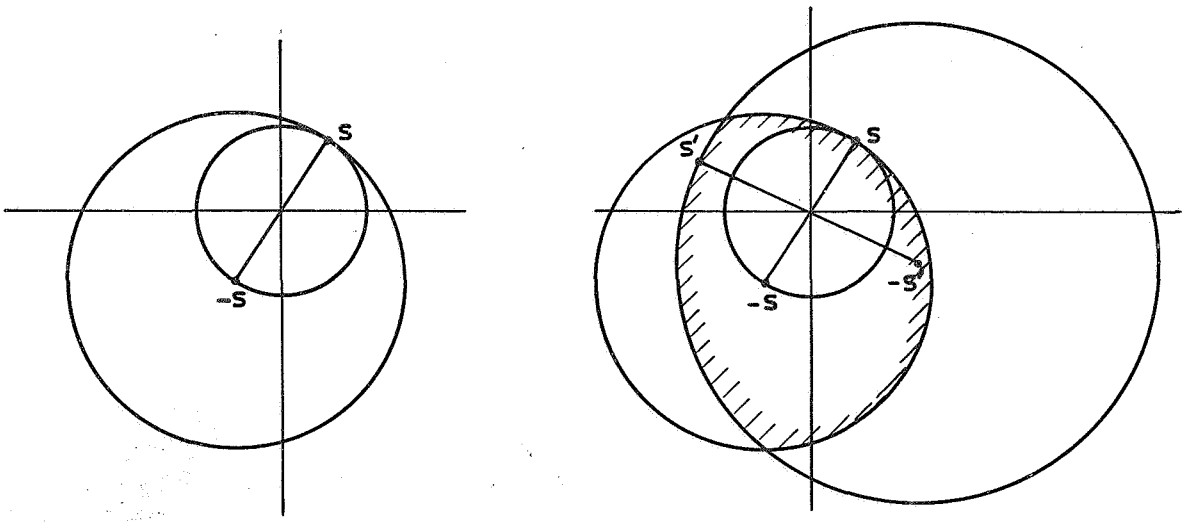
De punten z die hieraan voldoen liggen binnen een cirkel met straal $2|s_1|$ en middelpunt $-s_1$.

Een singulariteit s_2 geeft op dezelfde wijze als voorwaarde

$$|z - (-s_2)| < 2|s_2| .$$

Bij elke singulariteit behoort een cirkel waarbinnen z zeker moet liggen.
De doorsnede van deze cirkels is het convergentiegebied van de getransformeerde reeks.

Voorbeelden



In de figuren is de convergentiecirkel van de machtreeks $\Sigma u_k z^k$ ook getekend. Deze cirkel gaat door de dichtst bij de oorsprong gelegen singulariteit en ligt dus binnen het convergentiegebied van de geeulerde reeks.
Hiermede is de formule van Euler bewezen.
Tevens blijkt dat de transformatie de mogelijkheid biedt een door een machtreeks gegeven functie F(z) analytisch voort te zetten.

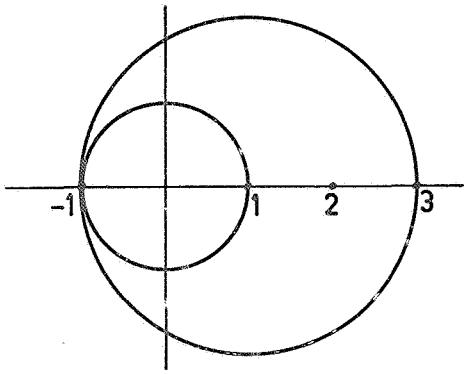
Opmerking

Het bewijs gaat niet op als $\Sigma u_k z^k$ convergeert in de singulariteit s (zie figuur). Het punt s ligt net op de rand. Bovenstaande beschouwing levert echter geen uitsluitsel over het gedrag van de getransformeerde reeks op de rand van het convergentiegebied.

Voorbeeld

$$\ell n\ (1 + z) = z - \frac{z^2}{2} + \frac{z^3}{3} - \frac{z^4}{4} + \ldots \quad \text{voor } -1 < z \leq 1.$$

Het punt -1 is een singulariteit van $\ell n\ (1 + z)$.



Substitutie van z = 2 geeft

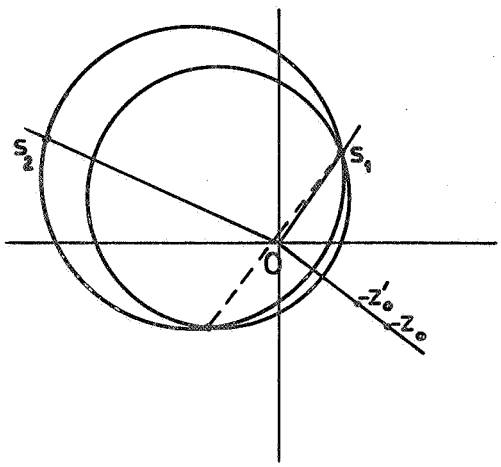
$$\ell n\ 3 = \text{(in de zin van Euler)} = 2 - \frac{2^2}{2} + \frac{2^3}{3} - \frac{2^4}{4} + \frac{2^5}{5} - \ldots$$

Euleren :

2				
-2	0.000			
2.667	0.333	0.167		
-4	-0.667	-0.167	0.000	
6.400	1.200	0.267	0.050	0.025

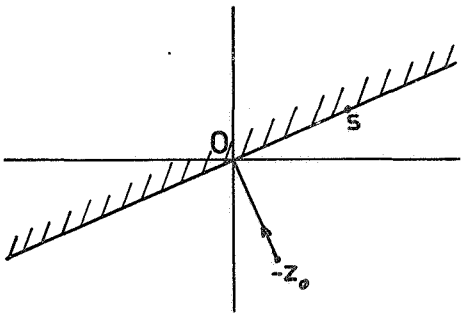
$$\ell n\ 3 = \tfrac{1}{2}\ (2 + 0.167 + 0.025 + \ldots\ ) \approx 1.069$$

De transformatie geeft hier een keurig convergente reeks.
We beschouwen nu weer de algemene transformatie (4.3)
Men kan zich afvragen hoever men de functie F(z) hiermee analytisch kan voortzetten, m.a.w. voor welke punten z kunnen we een z_o kiezen, zodanig dat de getransformeerde reeks convergeert.



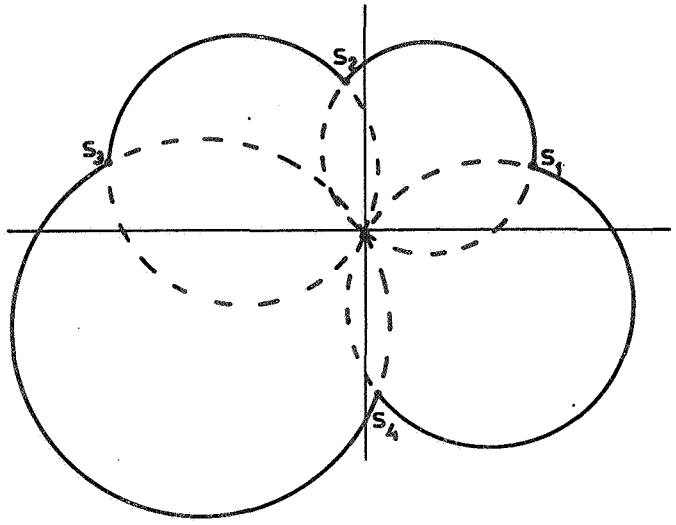
Bij een punt $-z_o$ en een singulariteit s_1 behoort een Apollonius cirkel.
We gaan nu met $-z_o$ langs een rechte naar 0, en wel zo dat 0 in het inwendige van de cirkel ligt. Zolang er nog geen andere singulariteit binnen de cirkel ligt is dit het convergentiegebied van de getransformeerde reeks bij de desbetreffende z_o.
Als $-z_o$ naar 0 gaat, wordt de Apollonius cirkel opgeblazen. Men kan zover gaan tot er een tweede singulariteit s_2 op de omtrek ligt.

In de hoek tussen Os_1 en Os_2 kan men nog verder komen door de rechte waarlangs $-z_o$ naar 0 gaat te draaien (0 steeds binnen de Apollonius cirkel).
Het beste wat men kan halen is een cirkel door s_1, s_2 en 0.



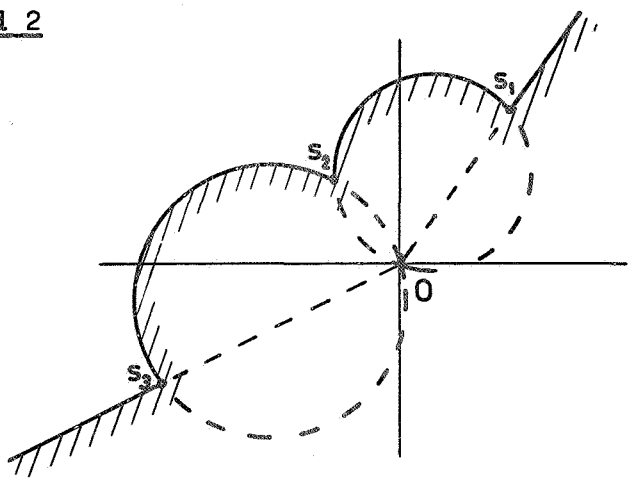
Het gearceerde halfvlak kan men halen door $-z_o$ langs een rechte loodrecht op Os tot O te laten naderen.
Dit geval doet zich voor als in het halfvlak geen andere singulariteiten liggen.

Voorbeeld 1



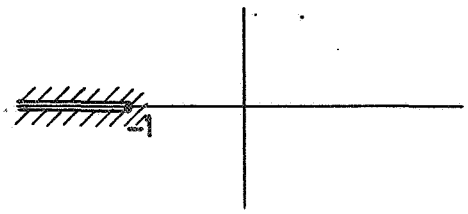
Men kan hier F(z) analytisch voortzetten binnen het dikomlijnde gebied. Buiten dit gebied kan men niet komen.
In dit voorbeeld zijn de hoeken tussen twee opeenvolgende voerstralen van singulariteiten steeds kleiner dan 180° .

Voorbeeld 2



De hoek tussen Os_3 en Os_1 is groter dan 180° .

Voorbeeld 3



De functie $\ell n(1 + z)$ heeft in -1 een singulariteit.
Men kan hier alle punten bereiken, behalve de "schaduw" van de singulariteit.

Opmerking

Voor elk punt z in het convergentiegebied kan men een convergerende reeks voor $F(z)$ opstellen. Men kan dus alles uitrekenen. Dit betekent echter niet dat men alles ook glad kan uitrekenen. De getransformeerde reeks kan zeer slecht convergeren.
Euler verbetert de convergentie van langzaam convergente of divergente alternerende reeksen, maar de convergentie van een reeks, waarvan alle termen hetzelfde teken hebben, wordt door de transformatie niet beter.

In de voorafgaande beschouwingen zijn z en z_o ingevoerd om de complexe functietheorie te kunnen toepassen; achteraf willen we deze grootheden weer kwijt.
We zullen formule (4.3) omvormen; daartoe schrijven we $M^{\ell}(u_k\ z_o^{\,k})_{k=o}$ op een andere manier.

$$
\begin{array}{ll}
u_o & \\
 & \tfrac{1}{2}u_o + \tfrac{1}{2}u_1 z_o \\
u_1 z_o & \qquad\qquad \cdots\cdots \\
 & \tfrac{1}{2}u_1 z_o + \tfrac{1}{2}u_2 z_o^{\,2} \\
u_2 z_o^{\,2} & \\
\vdots & \vdots \\
\vdots & \vdots \\
\vdots & \vdots
\end{array}
$$

We merken op dat $\frac{1}{2}u_o + \frac{z_o}{2}u_1$ een soort gewogen gemiddelde is tussen u_o en u_1, maar
$\frac{1}{2} + \frac{z_o}{2} \neq 1$. Door de factor $\frac{1 + z_o}{2}$ buiten haakjes te brengen maken we dit in orde.

$$\tfrac{1}{2}u_o + \frac{z_o}{2}u_1 = \left(\frac{1}{1 + z_o}u_o + \frac{z_o}{1 + z_o}u_1\right) \cdot \frac{1 + z_o}{2}$$

We krijgen:

$$
\begin{array}{ll}
u_0 & \\
 & \left(\frac{1}{1+z_0}\,u_0 + \frac{z_0}{1+z_0}\,u_1\right)\cdot\frac{1+z_0}{2} \\
u_1 z_0 & \cdots \\
 & \left(\frac{1}{1+z_0}\,u_1 + \frac{z_0}{1+z_0}\,u_2\right) z_0\cdot\frac{1+z_0}{2} \\
u_2 z_0^2 & \cdots \\
 & \left(\frac{1}{1+z_0}\,u_2 + \frac{z_0}{1+z_0}\,u_3\right) z_0^2\cdot\frac{1+z_0}{2} \\
u_3 z_0^3 & \\
\vdots & \vdots
\end{array}
$$

De kolom gemiddelden is op de factor $\frac{1+z_0}{2}$ na van dezelfde structuur als de eerste kolom. De tweede kolom van gemiddelden kan men dus op analoge wijze vormen, etc.

We voeren nu een nieuw type middelingsoperator in.

$$M_q f_k = (1-q) f_k + q f_{k+1}$$

Voor $q = \frac{1}{2}$ levert dit het arithmetisch gemiddelde dus $M = M_{\frac{1}{2}}$.

Neem nu $q = \frac{z_0}{1+z_0}$. Dan heeft men

$$M(u_k z_0^k)_{k=0} = \frac{1+z_0}{2} M_q u_0$$

$$M^2(u_k z_0^k)_{k=0} = \left(\frac{1+z_0}{2}\right)^2 M_q^2 u_0$$

In het algemeen

$$M^\ell(u_k z_0^k)_{k=0} = \left(\frac{1+z_0}{2}\right)^\ell M_q^\ell u_0 .$$

Dit wordt gesubstitueerd in (4.3); als men tevens $z = 1$ neemt, komt er

$$\sum_{k=0}^{\infty} u_k = q \sum_{k=0}^{\infty} M_q^k u_0 .$$

Dit is in feite dezelfde transformatie als (4.3), maar de z_0 is nu in q gestopt.
Men kan de algemene Euler transformatie karakterizeren door een scalar g (eventueel complex).
Symbolisch zullen we de transformatie aangeven met E_q.
Zij S de reeks Σu_k, dan stelt $E_q S$ de getransformeerde reeks voor.

Ter illustratie enkele voorbeelden

$E_{\frac{1}{2}}$ $\quad M_{\frac{1}{2}} = M$, dit is de klassieke Euler transformatie.

E_1 $\quad M_1 f_k = F_{k+1}$

Schema van gemiddelden:

u_o

$\quad u_1$

$u_1 \quad\quad u_2$

$\quad u_2$

$u_2 \quad\quad u_3$

$\quad u_3$

u_3

De getransformeerde reeks is $\Sigma M_1^k u_o = \Sigma u_k$, dus de reeks zelf.
E_1 is een eenheidselement, $E_1 E_q = E_q E_1 = E_q$

E_o $\quad M_o f_k = f_k$

u_o

$\quad u_o$

$u_1 \quad\quad u_o$

$\quad u_1$

u_2

$\Sigma M_o^k u_o = u_o + u_o + u_o + \ldots$

Dit is een divergente reeks (voor $u_o \neq 0$).

E_o is onbruikbaar.

We zullen vervolgens nagaan wat het resultaat is van een herhaalde Euler transformatie $E_{q_2} E_{q_1} S$.

Eerst wordt E_{q_1} toegepast op de reeks S, daarna op het resultaat E_{q_2}.

We schrijven symbolisch $M_q = (1 - q) + q E$.

Dan is

$$M_q^k = \{(1 - q) + q E\}^k = \sum_{h=o}^{k} \binom{k}{h} (1 - q)^{k-h} q^h E^h .$$

$$E_{q_2} E_{q_1} S =$$

$$q_1 q_2 \sum_{k=0}^{\infty} \sum_{h=0}^{k} \binom{k}{h} (1-q_2)^{k-h} q_2^{h} \sum_{j=0}^{h} \binom{h}{j} (1-q_1)^{h-j} q_1^{j} E^{j} u_o .$$

Sommatie volgorde verwisselen geeft

$$E_{q_2} E_{q_1} S =$$

$$q_1 q_2 \sum_{k=0}^{\infty} \sum_{j=0}^{k} q_1^{j} E^{j} \sum_{h=j}^{k} \binom{k}{h}\binom{h}{j} (1-q_2)^{k-h} q_2^{h} (1-q_1)^{h-j} u_o$$

$$= q_1 q_2 \sum_{k=0}^{\infty} \sum_{j=0}^{k} \binom{k}{j} q_1^{j} E^{j} \sum_{h=j}^{k} \binom{k-j}{h-j} (1-q_2)^{k-h} q_2^{h} (1-q_1)^{h-j} u_o$$

wegens $\binom{k}{h}\binom{h}{j} = \binom{k}{j}\binom{k-j}{h-j}$.

Stel $h = j + m$.

$$E_{q_2} E_{q_1} S =$$

$$q_1 q_2 \sum_{k=0}^{\infty} \sum_{j=0}^{k} \binom{k}{j} q_2^{j} q_1^{j} E^{j} \sum_{m=0}^{k-j} \binom{k-j}{m} (1-q_2)^{k-j-m} q_2^{m} (1-q_1)^{m} u_o .$$

Nu is $\sum_{m=0}^{k-j} \binom{k-j}{m} (1-q_2)^{k-j-m} q_2^{m} (1-q_1)^{m} u_o =$

$$= \{(1-q_2) + q_2(1-q_1)\}^{k-j}$$

$$= (1-q_1 q_2)^{k-j}$$

Dus

$$E_{q_2} E_{q_1} S =$$

$$q_1 q_2 \sum_{k=0}^{\infty} \sum_{j=0}^{k} \binom{k}{j} q_2^{j} q_1^{j} E^{j} (1-q_1 q_2)^{k-j}$$

$$= q_1 q_2 \sum_{k=0}^{\infty} (1 - q_1 q_2 + q_1 q_2 E)^{k} u_o =$$

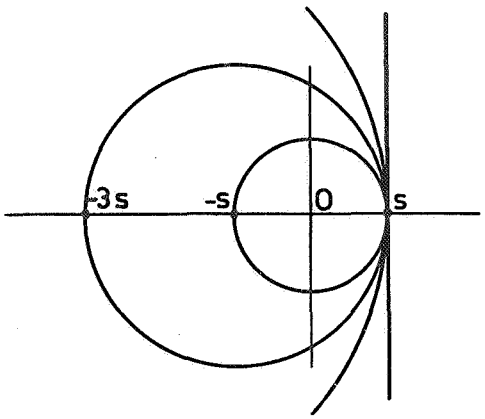
$$= q_1 q_2 \sum_{k=0}^{\infty} M_{q_1 q_2}^{k} u_o = E_{q_1 q_2} S .$$

Dus $E_{q_1} \circ E_{q_2} = E_{q_2} \circ E_{q_1} = E_{q_1 q_2}$.

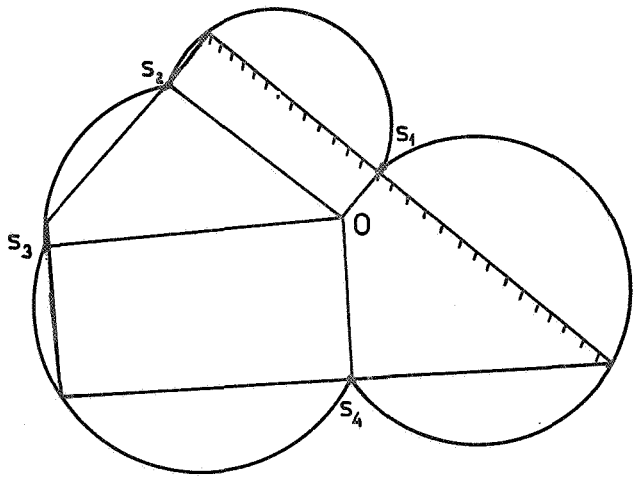
Het product van twee Euler transformaties is weer een Euler transformatie. De transformaties vormen een commutatieve groep. Men kan niet hopen door twee maal een Euler transformatie toe te passen iets nieuws te vinden, men had het ook met één transformatie kunnen doen.

In het bijzonder is $E_{\frac{1}{2}} \circ E_{\frac{1}{2}} = E_{\frac{1}{4}}$,

$E_{\frac{1}{2}} \circ E_{\frac{1}{4}} = E_{\frac{1}{8}}$ etc.



Als men herhaald $E_{\frac{1}{2}}$ toepast, verdubbelt iedere keer de diameter van de convergentie cirkel.
Men komt echter nooit voorbij de raaklijn in de singulariteit s.
Bij dit verdubbelen kan men natuurlijk wel een andere singulariteit passeren.
Door de herhaling wordt de convergentie wel slecht, $E_{\frac{1}{32}}$ bijv. zit dicht bij E_0 die onbruikbaar is.



Wat betreft de singulariteit s_1 kan men door herhaalde $E_{\frac{1}{2}}$ het gearceerde halfvlak dat 0 bevat, halen. Een analoge opmerking geldt voor de andere singulariteiten.

Door de constructieve Euler transformatie $E_{\frac{1}{2}}$ herhaald toe te passen kan men $F(z)$ in het gehele binnengebied van het polygoon analytisch voortzetten. Men noemt dit polygoon het Borel-polygoon. Door Borel is een niet constructieve transformatie van een machtreeks gegeven, waarmede men $F(z)$ ook binnen het polygoon kan voortzetten. Zoals men reeds eerder gezien heeft kan $F(z)$ voortgezet worden in het gehele binnengebied van de cirkels door bij een punt z een passende E_q te kiezen (d.w.z. een geschikt punt z_o).

Opmerking

Men zou kunnen zeggen: de herhaalde Euler transformatie heeft geen zin, in een keer kan met hetzelfde resultaat bereiken. Men mag dit echter niet zo stellen. De decisie om de transformatie te herhalen kan men slechts nemen op grond van gegevens die tijdens het proces beschikbaar komen. Achteraf kan men pas vaststellen dat het in een stap had gekund.
Een soortgelijke opmerking kan men maken naar aanleiding van de formule

$$\sum_{k=0}^{\infty} u_k = \sum_{k=0}^{m-1} u_k + \sum_{k=0}^{\infty} M_q^k u_m \qquad (4.4)$$

De formule zegt dat men kan euleren na m termen. Maar met welke g en welke m moet men werken ? De keuze kan pas gedaan worden op grond van gegevens die het proces oplevert.
Het is bijv. reëel om maar eens te beginnen met $q = \frac{1}{2}$ (tussen E_0 en E_1 in).

Op grond waarvan en hoe men m en g tijdens het proces verandert komt in een formule als (4.4) in het geheel niet tot uiting.
Een proces is veel meer dan een formule.

Tot dusver zijn reeksen $\Sigma u_k z^k$ beschouwd die convergeren in een omgeving van 0. De reeks $\Sigma u_k z^k$ kan ook voor elke $z \neq 0$ divergeren. In dat geval zal de getransformeerde reeks ook voor elke z divergeren. Toch kan de Euler transformatie in een dergelijke situatie bruikbaar zijn. Om dit in te zien moeten we eerst enige aandacht schenken aan asymptotische reeksen.

Beschouw de functie $F(x) = \int_x^{\infty} \frac{e^{-t}}{t}\, dt$. (x reëel).

Voor $x > 0$ is dit een wel gedefinieerde functie.
Voor $x = 0$ heeft $F(x)$ een logarithmische singulariteit. Men kan $F(x)$ -log x in een machtreeks ontwikkelen.

$$F(x) = \ln x + \gamma + \sum_{k=1}^{\infty} (-1)^k \frac{x^k}{k \cdot k!}$$

(γ is de constante van Euler).

De machtreeks convergeert voor alle x, maar is voor grote x voor numerieke berekening onbruikbaar. De termen van de reeks gaan op de duur wel naar nul, maar worden eerst zeer groot.
Euler helpt hier niet. Het begin van de reeks gedraagt zich voor grote x als een snel divergente reeks, de staart gedraagt zich als een snel convergente reeks.
De machtreeks ontwikkeling is dus alleen bruikbaar voor kleine x.

We gaan de integraal omvormen door middel van partiële integratie.

$$\int_x^\infty \frac{e^{-t}}{t}\,dt = -\frac{e^{-t}}{t}\Big|_x^\infty - \int_x^\infty \frac{e^{-t}}{t}\,dt =$$

$$= \frac{e^{-x}}{x} + \frac{e^{-t}}{t^2}\Big|_x^\infty + 2\int_x^\infty \frac{e^{-t}}{t^3}\,dt =$$

$$= e^{-x}\left[\frac{0!}{x} - \frac{1!}{x^2} + \frac{2!}{x^3}\right] - 3!\int_x^\infty \frac{e^{-t}}{t^4}\,dt.$$

In het algemeen

$$F(x) = e^{-x}\left[\frac{0!}{x} - \frac{1!}{x^2} + \frac{2!}{x^3} + \ldots + (-1)^{n-1}\frac{(n-1)!}{x^n}\right] +$$

$$+ (-1)^n\, n!\int_x^\infty \frac{e^{-t}}{t^{n+1}}\,dt.$$

Nu geldt

$$0 < n!\int_x^\infty \frac{e^{-t}}{t^{n+1}}\,dt < \frac{n!}{x^{n+1}}\int_x^\infty e^{-t}\,dt = \frac{n!}{x^{n+1}}e^{-x},$$

dus

$$F(x) = e^{-x}\left[\frac{0!}{x} - \frac{1!}{x^2} + \ldots + (-1)^{n-1}\frac{(n-1)!}{x^n} + (-1)^n\,\theta\,\frac{n!}{x^{n+1}}\right]$$

met $0 < \theta < 1$. (4.5)

Men noemt $\sum_{k=1}^{\infty} (-1)^{k-1}\frac{(k-1)!}{x^k}$ de asymptotische machtreeks van $e^x F(x)$.

Notatie: $e^x F(x) \sim \sum_{k=1}^{\infty} (-1)^{k-1}\frac{(k-1)!}{x^k}$.

Voor elke waarde van x is de asymptotische reeks divergent. Toch is de

reeks zeer geschikt om $F(x)$ voor grote waarden van x te berekenen. Dit blijkt uit formule (4.5). De rest na n termen van de reeks is

$(-1)^n \theta \frac{n!}{x^{n+1}}$ $(0 < \theta < 1)$, de rest ligt dus tussen nul en de eerst verwaarloosde term in. Men kan de eerst verwaarloosde term als schatting van de fout nemen.
Met een partiële som van n termen van de reeks kan men $e^x F(x)$ benaderen met een fout die in absolute waarde kleiner is dan $\frac{n!}{x^{n+1}}$. De benadering is beter, naarmate x groter is.

We nemen nu x vast.
Voor grote x nemen de termen van de reeks in het begin snel af. Op de duur wint de faculteit in de teller en gaan de termen naar oneindig.
De beste schatting voor $F(x)$ krijgt men door zoveel termen te nemen, dat de eerst verwaarloosde term zo klein mogelijk is.
De asymptotische reeks stelt een grens aan de precisie waarmee $F(x)$ bij vaste x berekend kan worden. Vergelijk dit met de convergente machtreeks voor $F(x)$. Hiermee kan men $F(x)$ in elke precisie bepalen door genoeg termen te sommeren.

In het algemeen heet $A_o + A_1 x^{-1} + A_2 x^{-2} + \ldots$ de asymptotische machtreeks van $f(x)$ voor $x \to \infty$ als

$$\lim_{x\to\infty} x^n [f(x) - (A_o + A_1 x^{-1} + \ldots + A_n x^{-n})] = 0$$

van $n = 0,1,2, \ldots .$

Dan is dus $\lim_{x\to\infty} f(x) = A_o$

$\lim_{x\to\infty} x [f(x) - A_o] = A_1$

$\lim_{x\to\infty} x^2 [f(x) - A_o - A_1 x^{-1}] = A_2$ etc.

Voor e^{-x} vindt men

$$e^{-x} \sim 0 + 0.x^{-1} + 0.x^{-2} + 0.x^{-3} + \ldots .$$

De functies $f(x)$ en $f(x) + e^{-x}$ hebben dezelfde asymptotische reeks.
Er is in feite een hele klasse van functies met een zelfde asymptotische ontwikkeling.
Bij het sommeren van een divergente reeks moet men oppassen. Men moet weten voor welke functie uit de klasse de som een benadering is.

In ons voorbeeld bijv. leveren $F(x)$ en $F(x) + 10^{100} e^{-x^2}$ dezelfde asymp-

totische reeks

$$e^{-x} \sum_{k=1}^{\infty} (-1)^{k-1} \frac{(k-1)!}{x^k} .$$

Voor x = 10 levert deze formule een goede benadering voor F(x), maar niet voor

$$F(x) + 10^{100}\ e^{-x^2} .$$

We demonstreren nu hoe men het Euler proces kan toepassen op een asymptotische reeks. De berekening wordt uitgevoerd voor x = 1.
Dit is een extreem lage waarde om in een asymptotische reeks in te vullen. Voor grotere x gaat het proces beter.
De reeks wordt:

$$0 ! - 1 ! + 2 ! - 3 ! + 4 ! - 5 ! + \ldots .$$

We euleren de reeks

1				
	0.0			
-1		0.25		
	0.5		-0.250	
2		-0.75		0.562
	-2		1.375	
-6		3.50		-3.312
	9		-8.000	
24		-19.50		22.625
	-48		53.250	
-120		126		
	300			
720				

De bovenste diagonaal is de getransformeerde reeks. Deze reeks divergeert, maar de staart van de reeks alterneert. De eerste drie termen worden geboekt. De staart wordt opnieuw geeulerd.

-0.250			
	0.156		
0.562		-0.125	
	-0.406		0.181
-1.375		0.488	
	1.382		
4.140			

De geeulerde reeks alterneert nu van meet af aan.
We euleren nog maar eens.

-0.250			
	-0.047		
0.156		-0.016	
	0.016		0.003
-0.125		0.022	
	0.028		
0.181			

Als som van de asymptotische reeks vinden we

$$\tfrac{1}{2}\,(1 + 0.0 + 0.25) + \tfrac{1}{8}\,(-0.250 - 0.047 - 0.016 + 0.003) = 0.586$$

En $$F(1) = e^{-1} \times 0.586 = 0.368 \times 0.586 = 0.216$$

We controleren dit door $F(1)$ met de convergente machtreeks te berekenen.

$$F(1) = -\gamma - \log 1 - \sum_{k=1}^{\infty} (-1)^k \frac{1}{k.k!} =$$

$$= -0.557 - (-1.000 + 0.250 - 0.056 + 0.010 - 0.002) = 0.221.$$

Opmerking.
Stel dat van een functie $f(x)$ een asymptotische reeks is gegeven. Zoals reeds opgemerkt heeft een klasse van functies dezelfde reeks. Hoe vergewist de numericus zich ervan dat de reeks een benadering van $f(x)$ geeft en niet van een andere functie uit de klasse. Daarvoor hebben we van $f(x)$ meer nodig, bijv. een convergente machtreeks.
Voor een aantal waarden van x (bijv. $x = 1(1)\ 5$) wordt $f(x)$ berekend met de convergente reeks. Voor dezelfde waarden past men het Euler proces op de asymptotische reeks toe. Stemmen de waarden overeen, dan kan men voor grotere x doorgaan met de asymptotische reeks.

De asymptotiek is een voor de numericus belangrijk onderdeel van de analyse. Er zijn verschillende technieken om van uitdrukkingen asymptotische ontwikkelingen te maken. We zullen nog een eenvoudig voorbeeld behandelen.

De functie $\Pi(x)$ wordt gedefinieerd door

$$\Pi(x) = \int_0^{\infty} e^{-t}\, t^x\, dt \quad (x \text{ complex, } \mathrm{Re}\ x > -1).$$

Door partiële integratie leiden we een functionaalrelatie af voor $\Pi(x)$.

$$\int_0^{\infty} e^{-t}\, t^x\, dt = -e^{-t}\, t^x \Big|_0^{\infty} + x \int_0^{\infty} e^{-t}\, t^{x-1}\, dt = x \int_0^{\infty} e^{-t}\, t^{x-1}\, dt,$$

dus $$\Pi(x) = x\,\Pi(x-1) \qquad (4.6)$$

voor $\mathrm{Re}\ x > 0$.

$\Pi(x)$ is gedefinieerd als een analytische functie van x voor $\mathrm{Re}\ x > -1$.
We kunnen de functie analytisch voortgezet denken voor $\mathrm{Re}\ x \leq -1$.
Volgens de complexe functie theorie blijft de relatie (4.6) daarbij geldig.

We berekenen enkele waarden van $\Pi(x)$.

$$\Pi(0) = \int_0^\infty e^{-t}\,dt = 1$$

$$\Pi(1) = 1 \cdot \Pi(0) = 1!$$

$$\Pi(2) = 2 \cdot \Pi(0) = 2!$$

$$\Pi(n) = n! \qquad (n \text{ geheel}, \geqslant 0) \qquad (4.7)$$

$\Pi(x)$ is dus een functie die de faculteit interpoleert.

$$\Pi(-\tfrac{1}{2}) = \int_0^\infty e^{-t}\,t^{-\frac{1}{2}}\,dt = 2\int_0^\infty e^{-u^2}\,du \qquad (\text{stel } t = u^2)$$

Nu is

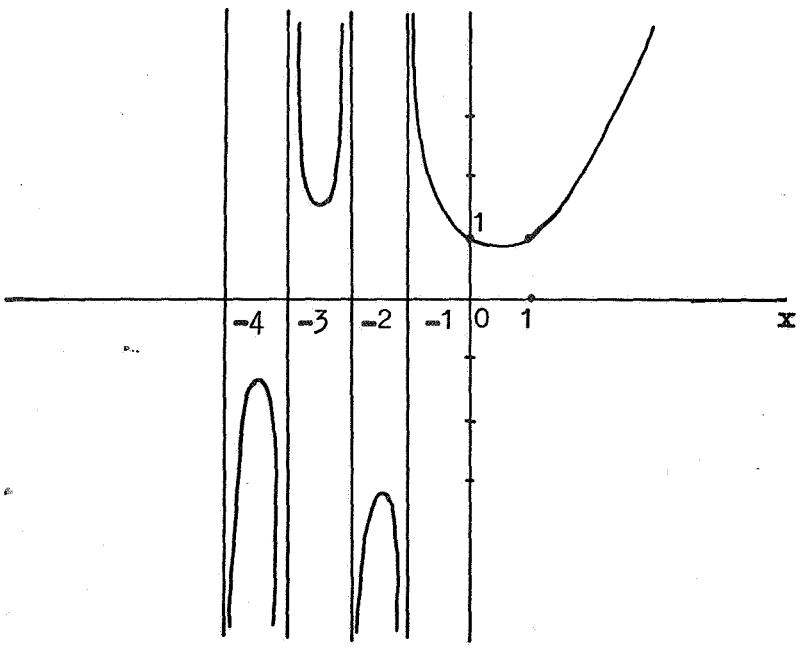
$$\left\{\int_{-\infty}^{\infty} e^{-u^2}\,du\right\}^2 = \int_{-\infty}^{\infty} e^{-u^2}\,du \cdot \int_{-\infty}^{\infty} e^{-v^2}\,dv = \int_0^{2\pi} d\theta \int_0^\infty r\,e^{-r^2}\,dr = \pi$$

Dus $\Pi(-\frac{1}{2}) = \sqrt{\pi}$, $\Pi(\frac{1}{2}) = \frac{1}{2}\sqrt{\pi}$.

$$\Pi(-1) = \infty\,, \quad \Pi(-2) = \infty\,.$$

$\Pi(x)$ heeft singulariteiten voor $x = -1, -2, -3, \ldots$.

Grafiek voor reële x.



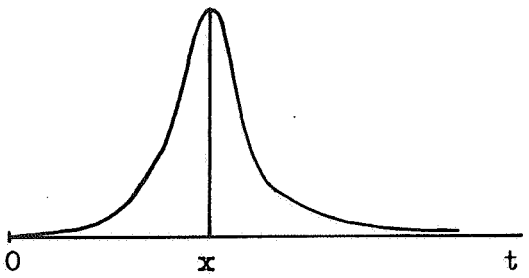
Tenslotte vermelden we nog zonder bewijs de relatie

$$\Pi(x)\ \Pi(-x) = \frac{\pi x}{\sin \pi x} \qquad (4.8)$$

We beperken ons nu tot reële waarden van x.

Hoe gedraagt $\Pi(x) = \int_0^\infty e^{-t}\ t^x\ dt$ zich voor $x \to +\infty$.

We geven in het volgende een schets van de gedachtegang die tot het resultaat voert.
Daartoe beschouwen we het gedrag van de integrand voor zeer grote waarden van x. Dan loopt t^x voor kleine t zeer vlak.



Voor grote t maakte e^{-t} de integrand klein.
Nu is $e^{-t}\ t^x = e^{-t+x \log t}$
De integrand heeft een maximum als $\frac{d}{dt}(-t + x \log t) = 0$ is.

Dit levert $t = x$.

De waarde van $\Pi(x)$ in het maximum is $e^{-x}\ x^x = \left(\frac{x}{e}\right)^x$.

Voor grote x is het maximum dus zeer groot.
In feite draagt alleen de omgeving van x tot de integraal bij. De staarten doen niet mee.

In een omgeving van x willen we nu de integrand door een eenvoudiger functie benaderen die daar hetzelfde verloop heeft.

Stel $t = x + u$.
Dan is

$$-t + x \log t = -x - u + x \log (x + u) =$$

$$= -x - u + x \log x + u - \frac{u^2}{2x} + \frac{u^3}{3x^2} - \frac{u^4}{4x^3} + \ldots$$

Voor $-x < u < x$ is de integrand

$$e^{-t}\ t^x = e^{-x} \cdot x^x \cdot e^{-\frac{u^2}{2x}} \cdot e^{\frac{u^3}{3x^2} - \frac{u^4}{4x^3} + \ldots}$$

In het gebied dat ons interesseert (u klein, bij grote x) is

$$e^{-t}\ t^x \sim e^{-x}\ x^x\ e^{-\frac{u^2}{2x}}$$

en

$$\int_0^\infty e^{-t}\ t^x\ dt \sim e^{-x}\ x^x \int_{-x}^\infty e^{-\frac{u^2}{2x}}\ du \sim e^{-x}\ x^x \int_{-\infty}^\infty e^{-\frac{u^2}{2x}}\ du =$$

$$= e^{-x}\, x^x \sqrt{2x} \int_{-\infty}^{\infty} e^{-\frac{u^2}{2x}}\, d\left(\frac{u}{\sqrt{2x}}\right) = \sqrt{2\pi}\, x^{x+\frac{1}{2}}\, e^{-x} .$$

Dit is de formule van Stirling.
Het bovenstaande is uiteraard geen bewijs van deze formule. Men kan echter alles exact rechtvaardigen.

Opmerking

$\Pi(x) = \Gamma(x+1)$. In het volgende zullen we voor $\Pi(x)$ ook $x!$ schrijven.

We passen Stirling toe op een paar voorbeelden uit de numerieke wiskunde.

Voorbeeld

De interpolatie formule van Newton luidde

$$f_n^*(x) = \sum_{k=o}^{n} \binom{p}{k} \Delta^k f_o \quad \text{met } x = x_o + p\,h$$

Hoe gedraagt zich $\binom{p}{k}$ voor vast p als k groter wordt.

$$\binom{p}{k} = \frac{p!}{k!(p-k)!} \qquad \text{met } p! = \Pi(p).$$

Nu is $p - k < 0$. Via formule (4.8) stappen we over op $k - p$.

$$\binom{p}{k} = \frac{p!}{k!\,\frac{(k-p)\pi}{\sin(k-p)\pi}\cdot\frac{1}{(k-p)!}} =$$

$$= \frac{p!(\sin k\pi \cos p\pi - \cos k\pi \sin p\pi)(k-p)!}{(k-p)\pi \cdot k!}$$

$$= \frac{(-1)^{k+1}\, p!\, \sin p\pi}{\pi(k-p)} \cdot \frac{(k-p)!}{k!} \sim$$

$$\sim \frac{(-1)^{k+1}\, p!\, \sin p\pi}{\pi k} \cdot \frac{\sqrt{2\pi}\,(k-p)^{k-p+\frac{1}{2}} \cdot e^{-k+p}}{\sqrt{2\pi}\, k^{k+\frac{1}{2}}\, e^{-k}} \cdot$$

$$\binom{p}{k} \sim \frac{(-1)^{k+1}\, p!\, \sin p\pi}{\pi k} \cdot \frac{k^{k-p+\frac{1}{2}}\left(1-\frac{p}{k}\right)^{k-p+\frac{1}{2}} e^{p}}{k^{k+\frac{1}{2}}}$$

$$\text{Nu is } \lim_{k\to\infty}\left(1-\frac{p}{k}\right)^{k-p+\frac{1}{2}} = \lim_{k\to\infty}\left(1-\frac{p}{k}\right)^{-p+\frac{1}{2}} \cdot \lim_{k\to\infty}\left(1-\frac{p}{k}\right)^{k} = e^{-p}$$

Dus

$$\binom{p}{k} \sim (-1)^{k+1} \cdot \frac{p!\sin p\pi}{\pi} \cdot k^{-1-p}$$

Als p dicht bij nul ligt gaat $\binom{p}{k}$ als $\frac{1}{k}$ naar nul.

Ligt p dicht bij een, dan gaat $\binom{p}{k}$ als $\frac{1}{k^2}$ naar nul.

De Newton coëfficiënten dalen dus langzaam.

Voorbeeld

De coëfficiënten van even orde in een centrale interpolatie formule zijn van het type $\binom{k+p}{2k}$.

$$\binom{k+p}{2k} = \frac{(k+p)!}{(2k)!(-k+p)!} = \frac{(k+p)!(k-p)!}{(2k)!\,\frac{(k-p)\pi}{\sin(k-p)\pi}}$$

$$\sim(-1)^{k+1}\,\frac{\sin p\pi}{\pi(k-p)}\,\frac{\sqrt{2\pi}\,(k+p)^{k+p+\frac{1}{2}}\,e^{-k-p}\,\sqrt{2\pi}\,(k-p)^{k-p-\frac{1}{2}}\,e^{-k+p}}{\sqrt{2\pi}\,(2k)^{k+\frac{1}{2}}\,e^{-2k}}$$

$$\sim(-1)^{k+1}\,\frac{\sin p\pi}{k}\,\sqrt{\frac{2}{\pi}}\cdot\frac{k^{k+p+\frac{1}{2}}\,(1+\frac{p}{k})^{k+p+\frac{1}{2}}\,k^{k-p+\frac{1}{2}}\,(1-\frac{p}{k})^{k-p+\frac{1}{2}}}{2^{2k+\frac{1}{2}}\,k^{2k+\frac{1}{2}}}$$

$$\sim(-1)^{k+1}\cdot\sin p\cdot\frac{1}{\sqrt{\pi k}}\;2^{-2k}$$

Deze coëfficiënten gaan als 2^{-2k} naar nul, dus veel harder dan de Newton coëfficiënten.

Na deze uitweiding keren we terug tot de sommatie van reeksen.

We beschouwen nu een reeks $\sum_{k=1}^{\infty} u_k$, slecht convergent, maar niet alternerend.
Euler kan ons hier niet helpen. De staart van zo'n reeks geeft een veel grotere bijdrage tot de som dan het geval is bij een alternerende reeks. Men mag niet verwachten dat een aantal termen aan het begin van de reeks voldoende informatie geeft om de som te bepalen.

We zullen eerst formeel werken zonder op convergentie te letten. Later volgt een preciese formulering.

We trachten de reeks alternerend te maken.

$$\begin{array}{llllllll} u_1 - u_2 + u_3 - u_4 + u_5 - u_6 + u_7 - u_8 + \ldots \\ \quad 2u_2 \quad + 2u_4 \quad + 2u_6 \quad + 2u_8 + \ldots \end{array}$$

Als we kolomsgewijs optellen klopt het. De eerste rij alterneert, de tweede echter niet.
We doen nog een stap.

$$\begin{array}{llllllll} u_1 - u_2 + u_3 - u_4 + u_5 - u_6 + u_7 - u_8 + \ldots \\ \quad 2u_2 \quad - 2u_4 \quad + 2u_6 \quad - 2u_8 + \ldots \\ \qquad\quad 4u_4 \qquad\qquad + 4u_8 + \ldots \end{array}$$

Kolomsgewijs optellen klopt, maar nu alterneert de derde rij niet.
Nog een stap.

$$\begin{array}{llllllll} u_1 - u_2 + u_3 - u_4 + u_5 - u_6 + u_7 - u_8 + \ldots \\ \quad 2u_2 \quad - 2u_4 \quad + 2u_6 \quad - 2u_8 + \ldots \\ \qquad\quad 4u_4 \qquad\qquad - 4u_8 + \ldots \\ \qquad\qquad\qquad\qquad\quad 8u_8 + \ldots \end{array}$$

Men kan dit schema voortzetten.
Kolomsgewijs klopt het.

We gaan nu scheef sommeren.

$$\begin{aligned} v_1 &= u_1 + 2u_2 + 4u_4 + 8u_8 + 16u_{16} + \ldots \\ v_2 &= u_2 + 2u_4 + 4u_8 + 8u_{16} + \ldots \\ v_3 &= u_3 + 2u_6 + 4u_{12} + 8u_{24} + \ldots \\ v_4 &= u_4 + 2u_8 + 4u_{16} + 8u_{32} + \ldots \\ v_5 &= u_5 + 2u_{10} + 4u_{20} + 8u_{40} + \ldots \end{aligned}$$

Algemeen:

$$v_k = u_k + 2u_{2k} + 4u_{4k} + 8u_{8k} + \ldots$$

Dan is $\sum_{k=1}^{\infty} u_k = \sum_{k=1}^{\infty} (-1)^{k+1} v_k$. (4.9)

Neem $u_k > 0$, dan is ook $v_k > 0$ en $\Sigma(-1)^{k+1}\ v_k$ alterneert inderdaad.
De bewering is dat $u_k + 2u_{2k} + 4u_{4k} + \ldots$ sneller convergeert dan de oorspronkelijke reeks, hetgeen we aan een voorbeeld zullen demonstreren.
Tevens is $v_k \approx u_k$, de reeks $\Sigma(-1)^{k+1}\ v_k$ convergeert dus even slecht als de oorspronkelijke reeks, maar is alternerend. Men kan dus heel prettig euleren.

Voorbeeld $\displaystyle\sum_{k=1}^{\infty} \frac{1}{k^2}$

$$v_k = \frac{1}{k^2} + 2\,\frac{1}{(2k)^2} + 4\,\frac{1}{(4k)^2} + 8\,\frac{1}{(8k)^2} + \ldots$$

$$= \frac{1}{k^2}\left(1 + \tfrac{1}{2} + \tfrac{1}{4} + \tfrac{1}{8} + \ldots\right) = \frac{2}{k^2}$$

De reeks voor v_k is een meetkundige reeks met reden $\frac{1}{2}$, dus veel sneller convergent dan $\Sigma\frac{1}{k^2}$. Men zou de reeks voor v_k gemakkelijk numeriek kunnen sommeren.

Nu moet gelden $\displaystyle\sum_{k=1}^{\infty} \frac{1}{k^2} = 2 \sum_{k=1}^{\infty} (-1)^{k+1}\ \frac{1}{k^2}$.

We verifiëren dit.

Zij $S = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \ldots$

en $T = 1 - \frac{1}{2^2} + \frac{1}{3^2} - \frac{1}{4^2} + \ldots$

Dan is $S - T = \frac{2}{2^2} + \frac{2}{4^2} + \frac{2}{6^2} + \ldots = \frac{1}{2}\left(1 + \frac{1}{2^2} + \frac{1}{3^2} + \ldots\right) = \frac{1}{2}\,S$,

dus $S = 2\,T$.

Opmerking

Van de v_k's behoeven we alleen die met oneven index te berekenen.

Immers $v_{2k} = u_{2k} + 2u_{4k} + 4u_{8k} + \ldots$,

dus $u_k + 2v_{2k} = u_k + 2u_{2k} + 4u_{4k} + \ldots = v_k$.

Hieruit volgt $v_{2k} = \dfrac{v_k - u_k}{2}$.

<u>Stelling</u> $\sum_{k=1}^{\infty} u_k = \sum_{k=1}^{\infty} (-1)^{k+1} v_k$

dan en slechts dan als

a) $\sum_{k=1}^{\infty} u_k$ een convergente reeks is,

b) $v_k = \sum_{j=0}^{\infty} 2^j u_{2^j k}$ een convergente reeks is voor $k = 1,2,3,\ldots$,

c) $\lim_{n\to\infty} \sum_{k=n+1}^{2n} v_k = 0$.

Bewijs:

$$v_{2k} = \sum_{j=0}^{\infty} 2^j u_{2^{j+1} k} = \tfrac{1}{2} \sum_{j=0}^{\infty} 2^{j+1} u_{2^{j+1} k} =$$

$$= \tfrac{1}{2} \sum_{j=1}^{\infty} 2^j u_{2^j k} = \tfrac{1}{2} \left(-u_k + \sum_{j=0}^{\infty} 2^j u_{2^j k} \right) = \frac{v_k - u_k}{2}$$

of $u_k = v_k - 2 v_{2k}$. Dit wisten we al.

$$\sum_{k=1}^{n} u_k = \sum_{k=1}^{n} (v_k - 2 v_{2k}) = \sum_{k=1}^{n} v_k - 2 \sum_{k=1}^{n} v_{2k} =$$

$$= \sum_{k=1}^{2n} v_k - 2 \sum_{k=1}^{n} v_{2k} - \sum_{k=n+1}^{2n} v_k = \sum_{k=1}^{n} (v_{2k-1} - v_{2k}) - \sum_{k=n+1}^{2n} v_k .$$

We nemen de limiet voor $n \to 0$.
Wegens voorwaarde c) volgt

$$\sum_{k=1}^{\infty} u_k = \sum_{k=1}^{\infty} (v_{2k-1} - v_{2k}) .$$

Uit de convergentie van $\Sigma(v_{2k-1} - v_{2k})$ volgt nog niet de convergentie

van $\Sigma(-1)^{k+1} v_k$. We mogen niet zonder meer de haakjes weglaten.
Dit is wel geoorloofd als $\lim_{k\to\infty} v_k = 0$ is (ga na).

We moeten dus nog aantonen $\lim_{k\to\infty} v_k = 0$

Σu_k is convergent, dus $\lim_{k\to\infty} u_k = 0$. Bij elke ε is een n_1 te vinden zodat voor alle $k > n_1$ geldt $|u_k| < \varepsilon$.

$\Sigma(v_{2k-1} - v_{2k})$ is convergent. Bij elke ε is een n_2 te vinden zodat $|v_{2k-1} - v_{2k}| < \varepsilon$ voor alle $k > n_2$. Neem $n = \max(n_1, n_2)$

$V_n = \max |v_k|$ als k het octaaf n+1, n+2, ..., 2n doorloopt.

$V_{n+1} = \max |v_k|$ als k het octaaf 2n+1, 2n+2, ..., 4n doorloopt.

We proberen een ongelijkheid te vinden tussen V_{n+1} en V_n. We zullen even en oneven indeces onderscheiden.
Neem een v_{2k} (k in n+1, ..., 2n).

$v_{2k} = \frac{1}{2}(v_k - u_k)$, dus

$|v_{2k}| \leq \frac{1}{2}(|v_k| + |u_k|) \leq \frac{1}{2} V_n + \frac{1}{2}\varepsilon$.

Neem een v_{2k-1} (k in n+1, ..., 2n) .

Dan is $|v_{2k-1} - v_{2k}| < \varepsilon$

dus $|v_{2k-1}| < |v_{2k}| + \varepsilon < \frac{1}{2} V_n + \frac{1}{2}\varepsilon + \varepsilon < \frac{1}{2} V_n + 2\varepsilon$.

Er geldt dus $V_{n+1} \leq \frac{1}{2} V_n + 2\varepsilon$.

Zo kan men doorgaan.

$$V_{n+2} \leq \tfrac{1}{4} V_n + 2\varepsilon$$

$$V_{n+3} \leq \tfrac{1}{8} V_n + 2\varepsilon$$

$$V_{n+m} \leq \frac{1}{2^m} V_n + 2\varepsilon$$

Men kan m zo groot kiezen dat

$$V_{n+m} \leq 3\varepsilon$$

Hieruit volgt $\lim_{k\to\infty} v_k = 0$.

Hiermede is bewezen dat de drie voorwaarden voldoende zijn. Dat ze ook nodig zijn is eenvoudig in te zien.

Opmerking

De voorwaarden zijn onafhankelijk.
In het bijzonder volgt uit de convergentie van Σu_k niet dat $\Sigma 2^j u_{2^j k}$ convergeert.

Neem bijv. de reeks

$$1 + \tfrac{1}{2} + 0 + \tfrac{1}{4} + 0 + 0 + 0 + \tfrac{1}{8} + 0 + 0 + 0 + 0 + 0 + 0 + 0 + \tfrac{1}{16} + \ldots$$

Deze reeks is convergent, maar $v_1 = 1 + 1 + 1 + 1 + \ldots$ divergeert.

Bij de toepassing van de transformatie geeft de verificatie van de voorwaarden a en b niet veel moeilijkheden; het is echter lastig om na te gaan of aan voorwaarde c

$$\left(\lim_{n\to\infty} \sum_{k=n+1}^{2n} v_k = 0 \right)$$

is voldaan.

De volgende stelling geeft een voldoende (maar niet nodige) voorwaarde, die praktisch gemakkelijker te hanteren is.

Stelling

Als men bij de reeks Σu_k een monotoon niet stijgende rij van niet negatieve getallen $U_1, U_2, U_3, \ldots$ kan aangeven, zodanig dat $|u_k| \leq U_k$ is en $\sum_{k=1}^{\infty} U_k$ convergent, dan convergeren $\sum_{1}^{\infty} u_k$ en $\sum_{1}^{\infty} (-1)^{k+1} v_k$ en

$$\sum_{k=1}^{\infty} u_k = \sum_{k=1}^{\infty} (-1)^{k+1} v_k .$$

Bewijs:

Wegens $|u_k| \leq U_k$ en de convergentie van $\sum_{1}^{\infty} U_k$ is $\sum_{1}^{\infty} u_k$ convergent.

We tonen vervolgens aan dat aan voorwaarde b is voldaan.

$$|u_k| + 2|u_{2k}| + 4|u_{4k}| + \ldots + 2^n |u_{2^n k}| \leq$$

$$\leq U_k + 2\,U_{2k} + 4\,U_{4k} + \ldots + 2^n U_{2^n k} =$$

$$= U_k + 2(U_{2k} + 2\,U_{4k} + \ldots + 2^{n-1} U_{2^n k}) \leq$$

$$\leq U_k + 2(U_{2k} + U_{3k} + U_{4k} + \ldots + U_{(2^n-1)k} + U_{2^n k}) \leq$$

$$\leq U_k + \frac{2}{k}(U_{k+1} + U_{k+2} + \ldots + U_{2k-1} + U_{2k} + U_{2k+1} + \ldots + U_{3k-1} +$$

$$+ U_{3k} + \ldots + U_{2^n k-1} + U_{2^n k}) .$$

Hier is gebruik gemaakt van de monotonie van de rij $U_1, U_2, \ldots$.

Zij $R_k = \sum_{j=k}^{\infty} U_k$, dan geldt dus

$$|u_k| + 2|u_{2k}| + 4|u_{4k}| + \ldots 2^n|u_{2^n k}| \leq U_k + \frac{2}{k} R_k .$$

Hieruit volgt de convergentie van $\sum_{j=o}^{\infty} 2^j u_{2^j k}$.
Verder blijkt dat

$$|v_k| \leq U_k + \frac{2}{k} R_k .$$

Tenslotte laten we zien dat aan voorwaarde c is voldaan.

$$\left|\sum_{k=n+1}^{2n} v_k\right| \leq \sum_{k=n+1}^{2n} |v_k| \leq \sum_{k=n+1}^{2n} U_k + \sum_{k=n+1}^{2n} \frac{2}{k} R_k \leq \sum_{k=n+1}^{2n} U_k + \sum_{k=n+1}^{2n} \frac{2}{n} R_n =$$

$$= \sum_{k=n+1}^{2n} U_k + 2\,R_n .$$

Wegens de convergentie van ΣU_k is $\lim_{n\to\infty} \sum_{k=n+1}^{2n} U_k = 0$ en $\lim_{n\to\infty} R_n = 0$,

dus $\lim_{n\to\infty} \sum_{k=n+1}^{2n} v_k = 0$.

Men kan de transformatie uitvoeren met behulp van de procedure euler:

```
real procedure euler (k, uk, eps, tim) ; value eps, tim ;
integer k, tim ; real uk, eps ; < body euler > ;
```

Men heeft nu een procedure nodig die v_k berekent uit de u_k's :

real procedure vk (k, uk, eps) ;
integer k ; real uk, eps ; < body vk > ;

Ter illustratie eerst een voorbeeld.
De volgende aanroep berekent de som van de reeks

$$\sum_{k_1=1}^{\infty} \frac{1}{k_1^{\frac{3}{2}} + k_1^{\frac{1}{2}}} .$$

euler (k1, - even (k1) × vk (k1,1/ (k1 ↑(3/2) + sqrt (k1)),$_{10}$ - 6),$_{10}$ - 6, 3)

Opmerking: de procedure even (k) heeft de waarde 1 als k even en -1 als k oneven is.

In de procedure vk zal men gebruik willen maken van de betrekking

$v_{2k} = \frac{1}{2} (v_k - u_k)$.

De procedure moet dus een aantal v_k's en u_k's onthouden die bij vorige aanroepen van vk berekend zijn.
Er zijn in Algol twee manieren om dit te bereiken.

1. Met behulp van own variabelen en own array's.
   Het is echter een bezwaar dat niet alle Algol vertalers hiermee kunnen werken. Bovendien is het niet uitgesloten dat bij de berekening van de u_k's dezelfde reekstransformatie wordt gebruikt. Dit zou tot recursief gebruik van own array's aanleiding geven, hetgeen weinig aanlokkelijk is.

2. Met behulp van enkele array's die van buiten aan de procedure vk worden meegegeven.

Wij kiezen de laatste oplossing.
De procedure wordt dan

real procedure vk (k, uk, eps, A, B) ;
integer k ; real uk, eps ;
integer array A ; real array B ; < body vk > ;

In het array B worden de u_k's en v_k's bewaard; in het array A wordt de administratie bijgehouden.

We geven nu het voorbeeld wat volledi ger.

```
begin integer k1 ; real x ; integer array A1[0 : 1000 ];
      real array B1[0 : 1000] ;
      real procedure euler ( ... ) ; ... ;
      real procedure  vk   ( ... ) ; ... ; ...

      ... ; x := euler (k1, - even (k1) × vk (k1,
      1/ (k1 ↑(3/2) + sqrt (k1)),10 - 6, A1, B1),10 - 6, 3) ; ...
end
```

Nu volgt de tekst van de procedure vk.

```
real procedure vk (k,uk,eps,A,B); integer k;
real uk,eps; integer array A; real array B;
begin integer m,mt,kt,tj; real s,ds;
      if k =0
      then begin A[0] := A[1] := 0; vk := 0
           end
      else if A[0]  = 0
           then begin kt := k; mt := A[kt];
                      tj := 1; s := 0; m := mt + 1;
                  L : B[m] := ds := uk; ds := ds × tj;
                      s := s + ds; m := m + 1;
                      if abs (ds) > eps
                      then begin tj := 2 × tj; k := 2 × k;
                                 goto L
                                 end
                      else vk := B[mt] := s;
                                 A[kt+2] := m;
                                 A[0] := 1; k := kt
                end
           else begin kt := k ÷ 2; m := A[kt];
                      A[k] := m+1;
                      vk := B[m+1] := (B[m] - B[m+1])/2;
                      B[m] := 0; A[0] := 0
                end
end
```

Toelichting

1. De waarde van v_k is te vinden in B[A[k]].
   Het array B wordt successievelijk gevuld met de berekende u_k's en v_k's.
   In het voorbeeld zijn v_1 t/m v_7 reeds berekend.

	0	1	2	3	4	5	6	7	8	9
A		0	1	7	2	13	8	18		

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
v_1	v_2 / u_1	v_4 / u_2	u_4	u_8	u_{16}	u_{32}	v_3	v_6 / u_3	u_6	u_{12}	u_{24}	u_{48}	v_5	u_5	u_{10}	u_{20}	u_{40}	v_7	u_7	u_{14}	u_{28}

v_1 wordt berekend met de reeks $u_1 + 2\,u_2 + 4\,u_4 + \ldots$.
De waarden van v_1 en u_1, u_2, u_4, ... worden in B opgeborgen.
v_2 wordt berekend met $\frac{1}{2}(v_1 - u_1)$; v_2 overschrijft u_1 .
etc.

2. A[o] heeft een bijzondere functie; hier wordt bijgehouden of de index van v_k even of oneven is.

3. We veronderstellen in het voorbeeld dat u_{64} te verwaarlozen is.
Toch kan men u_{64} eventueel nodig hebben voor de berekening van v_{128}.
Men moet dan $u_{64} = 0$ nemen.
Dit bereikt men door na de berekening van v_6 de waarde van v_3 door nul te vervangen.
In het algemeen moet men na de berekening van een v_{2k} de v_k door nul vervangen.

5. EXTRAPOLATIE

Men kan een proces niet uitvoeren met stapgrootte nul of met oneindig veel termen. Men moet werken met kleine stapjes $h > 0$ of met eindig veel (n) termen. Door extrapolatie naar nul of oneindig kan men proberen het verlangde resultaat te vinden.
Extrapolatie geeft aanleiding tot grote onzekerheden of fouten, tenzij men iets weet over het gedrag voor $h \to 0$ of $n \to \infty$.

Voorbeeld

Zij a_1, a_2, a_3, ... een rij getallen die tot een limiet a naderen.
Als men weet dat de afwijking van de limiet met een constante factor wordt vermenigvuldigd kan men extrapoleren.
We vormen differenties.

$$\begin{aligned} a_1 &\approx a + e \\ a_2 &\approx a + k\,e \qquad (k-1)\,e \\ a_3 &\approx a + k^2 e \qquad k(k-1)\,e \qquad (k-1)^2 e \end{aligned}$$

Men kan de afwijking ke elimineren.

$$ke = \frac{(k-1)e \times k(k-1)e}{(k-1)^2 e}$$

Evenzo is

$$e = \frac{\{(k-1)e\}^2}{(k-1)^2 e} \quad \text{en} \quad k^2 e = \frac{\{k(k-1)e\}^2}{(k-1)^2 e} \ .$$

Men vindt zo

$$a \approx a_1 - \frac{(\Delta a_1)^2}{\Delta^2 a_1} \ ; \ a \approx a_2 - \frac{\Delta a_2 \cdot \nabla a_2}{\delta^2 a_2} \ ; \ a \approx a_3 - \frac{(\nabla a_3)^2}{\nabla^2 a_3}$$

Dit is het bekende δ^2-proces van Aitken.
De drie formules zijn mathematisch equivalent, maar niet numeriek.
De laatste formule, die de kleinste correctie geeft, is numeriek de beste.
Door een te kleine noemer kan de toepassing van Aitken gevaarlijk zijn. Dit treedt vooral op als de convergentie van de rij slecht is, dus juist wanneer men extrapolatie werkelijk nodig heeft.

Men past dit proces bijv. toe bij het bepalen van de eigenwaarden van een matrix volgens de powermethode.

Zij A een matrix met eigenwaarden $\lambda_1, \lambda_2, \ldots, \lambda_n \quad (|\lambda_1| > |\lambda_2| > \ldots)$ en eigenvectoren $x_1, x_2, \ldots, x_n$.

Dan is $x = c_1 x_1 + c_2 x_2 + \ldots + c_n x_n$

en $Ax = \lambda_1 c_1 x_1 + \lambda_2 c_2 x_2 + \ldots + \lambda_n c_n x_n$

$$= \lambda_1 \left(c_1 x_1 + \frac{\lambda_2}{x_1} c_2 x_2 + \ldots + \frac{\lambda_n}{\lambda_1} c_n x_n\right) .$$

Na een aantal iteraties doet de staart niet meer mee en men heeft in feite

$$x = c_1 x_1 + c_2 x_2$$
$$Ax = \lambda_1 \left(c_1 x_1 + \frac{\lambda_2}{\lambda_1} c_2 x_2\right) .$$

De fout wordt hier dus vermenigvuldigd met de constante factor $\frac{\lambda_2}{\lambda_1}$ (na normering van Ax).
Op drie opeenvolgende vectoren kan men Aitken toepassen.
Het is beter dit niet element voor element te doen. Een element kan

nul worden.
Men moet de vectoren bij elkaar houden.
Men zou de formule

$$a \approx a_3 - \frac{(\nabla a_3)^2}{\nabla^2 a_3} \qquad (5.1)$$

i.p.v. op scalairen toe willen passen op vectoren. Samelson heeft aangegeven hoe men dit kan doen.

$(\nabla a_3)^2$ wordt vervangen door het inwendig product van de vector ∇a_3 met zichzelf.

Zij x een kolomvector en $\bar{x}$ de toegevoegd complex vector, dan noemt men $\frac{\bar{x}}{\bar{x}^T x}$ de Samelson inverse van de vector x.

Onder $\frac{1}{\nabla^2 a_3}$ verstaan we nu de Samelson inverse van de vector $\nabla^2 a_3$.

Ga na dat met deze interpretatie de formule (5.1) klopt.

We behandelen nu een wat verder gaand extrapolatie proces.
Zij f(h) een functie van h, waarbij h duidt op een stapgrootte ; zij verder bekend dat

$$f(h) \sim a_0 + a_1 h^{r_1} + a_2 h^{r_2} + a_3 h^{r_3} + \ldots \qquad (5.2)$$

met $0 < r_1 < r_2 < r_3 < \ldots$.

We nemen aan dat we r_1, r_2 etc. kennen, maar a_1, a_2, ... niet.
Gevraagd wordt a_0 te berekenen.

Opmerking

De r's behoeven niet geheel te zijn.
In (5.2) schrijven we het asymptotisch teken; bij de toepassing zal men echter vrijwel altijd met een convergente reeks te maken hebben.

We rekenen de functie uit voor een paar waarden van het argument, bijv. h, $k_1 h$ en $k_2 h$ en proberen de hoofdterm van de fout $a_1 h^{r_1}$ te elimineren.

$$f(k_1 h) \sim a_o + a_1 k_1^{r_1} h^{r_1} + a_2 k_1^{r_2} h^{r_2} + \ldots .$$

Dus is

$$f(k_1 h) - f(h) \sim a_1 (k_1^{r_1} - 1) h^{r_1} + a_2 (k_1^{r_2} - 1) h^{r_2} + \ldots .$$

Hieruit volgt

$$f(k_1 h) - \frac{k_1^{r_1}}{k_1^{r_1} - 1} \{ f(k_1 h) - f(h) \} \sim$$

$$\sim a_o + a_2 \left\{ k_1^{r_2} - \frac{k_1^{r_1} (k_1^{r_2} - 1)}{k_1^{r_1} - 1} \right\} h^{r_2} + \ldots .$$

Het extrapolatie resultaat is dus

$$f(k_1 h) - \frac{k_1^{r_1}}{k_1^{r_1} - 1} \{ f(k_1 h) - f(h) \}$$

De hoofdterm van de fout is verdwenen. De fout is nu van de orde h^{r_2}.

Met behulp van $f(k_2 h)$ kan men ook een extrapolatie vinden.
Als men op beide extrapolaties dezelfde bewerking herhaalt wordt de fout van de orde h^{r_3}.

We merken nogmaals op dat men voor dit proces a_1, a_2, etc. niet behoeft te kennen. Men behoeft slechts te weten hoe de storingen zich gedragen.

Dit proces zou de naam van Huygens moeten dragen.
In een artikel van 1654, getiteld "De circuli magnitudinis inventa" bepaalt hij met weinig rekenwerk 9 decimalen van π.

In het artikel bewijst hij de volgende stelling:

elke cirkel is groter dan een ingeschreven regelmatige veelhoek plus het derde gedeelte van wat die veelhoek groter is dan een andere met het halve aantal zijden.

Een soortgelijke stelling voor de omgeschreven regelmatige veelhoek geeft een bovengrens voor de omtrek van de cirkel.
Huygens kon dus π tussen twee grenzen insluiten. Veel belangrijker is echter dat hier het idee van de extrapolatie naar voren komt.

In moderne taal is zijn probleem als volgt te beschrijven.

Neem een cirkel met straal 1.
Zij s_n de omtrek van een ingeschreven regelmatige n-hoek, dan is

$$s_n = 2n \sin \frac{\pi}{n} .$$

We willen $\lim_{n \to \infty} s_n$ berekenen.

Men kan $\sin \frac{\pi}{n}$ in een machtreeks ontwikkelen.

$$s_n = 2\pi - \frac{\pi^3}{3} \frac{1}{n^2} + \frac{\pi^5}{60} \frac{1}{n^4} - \ldots .$$

Men heeft hier de situatie van formule (5.2) met $h = \frac{1}{n}$, $r_1 = 2$, $r_2 = 4$ etc.

De coëfficiënten $\frac{\pi^3}{3}$, $\frac{\pi^5}{60}$ ect. zijn onbekend, maar we kennen het gedrag van s_n voor $n \to \infty$.

Neem een reeks van het type

$$f(h) = a_0 + a_2 h^2 + a_4 h^4 + a_6 h^6 + \ldots$$

en bereken $f(h)$, $f(\frac{1}{2} h)$, $f(\frac{1}{4} h)$ etc.

$$f(\tfrac{1}{2} h) = a_0 + \tfrac{1}{4} a_2 h^2 + \tfrac{1}{16} a_4 h^4 + \tfrac{1}{64} a_6 h^6 + \ldots$$

$$f(\tfrac{1}{4} h) = a_0 + \tfrac{1}{16} a_2 h^2 + \tfrac{1}{256} a_4 h^4 + \tfrac{1}{4096} a_6 h^6 + \ldots$$

We elimineren de term van de orde h^2 .
Dit geeft

$$e(\tfrac{1}{2} h) = f(\tfrac{1}{2} h) + \tfrac{1}{3} \{f(\tfrac{1}{2} h) - f(h)\} = a_0 - \tfrac{1}{4} a_4 h^4 - \tfrac{1}{16} a_6 h^6 + \ldots$$

$$e(\tfrac{1}{4} h) = f(\tfrac{1}{4} h) + \tfrac{1}{3} \{f(\tfrac{1}{4} h) - f(\tfrac{1}{2} h)\} = a_0 - \tfrac{1}{4} \cdot \tfrac{1}{16} a_4 h^4 - \tfrac{5}{16} \cdot \tfrac{1}{64} a_6 h^6 + \ldots$$

We elimineren vervolgens de term van de orde h^4 door te nemen

$$e(\tfrac{1}{4} h) + \frac{1}{15} \{ e(\tfrac{1}{4} h) - e(\tfrac{1}{2} h) \} .$$

Zo kan men doorgaan. Men krijgt in de extrapolatie formules achtereenvolgens de coëfficiënten
$\frac{1}{3}, \frac{1}{15}, \frac{1}{63}, \frac{1}{255}$. In het algemeen $\frac{1}{4^n-1}$.

We passen dit toe op de berekening van 2π, waarbij we niet zullen schromen van de omtrek van een 2-hoek en zelfs van een 1-hoek en een $\frac{1}{2}$-hoek gebruik te maken.

$s_{\frac{1}{4}} = 0.000$

0.000

$s_{\frac{1}{2}} = 0.000$ 0.000

0.000 5.778

$s_1 = 0.000$ 5.688 6.277

5.333 6.275

$s_2 = 4.000$ 6.266

6.208

$s_4 = 5.656$

$(2\pi = 6.283)$

Opmerking

In het schema zijn de coëfficiënten die in de extrapolatie formules optreden slechts afhankelijk van de rij.
We zouden de regelmatige driehoek en zeshoek erbij kunnen nemen.
Daardoor worden de coëfficiënten afhankelijk van rij en kolom.
Men moet van tevoren een patroon maken van deze coëfficiënten. Hier is dat niet erg zinvol, maar bij partiële differentiaal vergelijkingen bijvoorbeeld kan het de moeite lonen een dergelijk schema op te stellen.

In het volgende wordt voor een algemeen geval het extrapolatie proces beschreven.
Stel dat men een berekening kan uitvoeren voor verschillende waarden van een parameter.
$F_k^{(o)}$ zij het resultaat van de berekening voor de waarde x_k van de parameter.
Stel verder dat men weet

$$F_k^{(o)} = f_o + f_1 x_k + f_2 x_k^2 + f_3 x_k^3 + \ldots\ ,$$

waarbij f_o, f_1, f_2 etc. onbekend zijn.

De waarde van f_o wordt gevraagd; men zal dus x_k klein kiezen. We veronderstellen dat de berekening niet uitvoerbaar is voor $x_k = 0$.

Voor x_{k+1} heeft men als resultaat $F_{k+1}^{(o)}$, met

$$F_{k+1}^{(o)} = f_o + f_1 x_{k+1} + f_2 x_{k+1}^2 + f_3 x_{k+1}^3 + \ldots\ .$$

Neem nu

$$F_k^{(1)} = \frac{x_{k+1} F_k^{(o)} - x_k F_{k+1}^{(o)}}{x_{k+1} - x_k}\ .$$

Dan is

$$F_k^{(1)} = f_o - f_2 x_k x_{k+1} + \ldots\ .$$

Uit $F_k^{(1)}$ is dus de eerste orde term geëlimineerd.

Algemeen:

.
.
.

x_k	$F_k^{(o)}$	$F_k^{(1)}$	$F_k^{(2)}$
x_{k+1}	$F_{k+1}^{(o)}$	$F_{k+1}^{(1)}$	
x_{k+}	$F_{k+1}^{(o)}$		

.
.
.

In het driehoekig schema is $F_k^{(i+1)}$ het extrapolatie resultaat van $F_k^{(i)}$

en $F_{k+1}^{(i)}$.

De fout in $F_k^{(i)}$ zal van de orde i+1 zijn.

We trachten de algemene vorm van de extrapolatie formule te vinden.

$$F_k^{(o)} = f_o + x_k \sum_{n=1}^{\infty} f_n x_k^{n-1}$$

$$F_{k+1}^{(o)} = f_o + x_{k+1} \sum_{n=1}^{\infty} f_n x_{k+1}^{n-1}$$

Dus $$F_k^{(1)} = \frac{x_{k+1} F_k^{(o)} - x_k F_{k+1}^{(o)}}{x_{k+1} - x_k} = f_o + x_k x_{k+1} \sum_{n=1}^{\infty} f_n \frac{x_k^{n-1} - x_{k+1}^{n-1}}{x_{k+1} - x_k}$$

of $$F_k^{(1)} = f_o - x_k x_{k+1} \sum_{n=2}^{\infty} f_n \left\{ \frac{x_k^{n-1}}{x_k - x_{k+1}} + \frac{x_{k+1}^{n-1}}{x_{k+1} - x_k} \right\} .$$

Opmerking

Deze formule is symmetrisch in x_k en x_{k+1}. De uitdrukking tussen accoladen is in feite een veelterm in x_k en x_{k+1} , immers

$x_k^{n-1} - x_{k+1}^{n-1}$ is deelbaar door $x_k - x_{k+1}$. Voor n=1 is de uitdrukking nul. Dit was ook de bedoeling.
Voor n=2 is de uitdrukking gelijk aan een. De fout in $F_k^{(1)}$ is van de tweede orde.

Voor $F_{k+1}^{(1)}$ vindt men op dezelfde manier

$$F_{k+1}^{(1)} = f_o - x_{k+1} x_{k+2} \sum_{n=2}^{\infty} f_n \left\{ \frac{x_{k+1}^{n-1}}{x_{k+1} - x_{k+2}} + \frac{x_{k+2}^{n-1}}{x_{k+2} - x_{k+1}} \right\} .$$

Door een combinatie van $F_k^{(1)}$ en $F_{k+1}^{(1)}$ te vormen kan men de tweede orde term elimineren.

Neem $$F_k^{(2)} = \frac{x_{k+2} F_k^{(1)} - x_k F_{k+1}^{(1)}}{x_{k+2} - x_k} .$$

Na enig rekenwerk vindt men

$$F_k^{(2)} = f_o + x_k\, x_{k+1}\, x_{k+2} \sum_{n=3}^{\infty} f_n \left\{ \frac{x_k^{n-1}}{(x_k - x_{k+1})(x_k - x_{k+2})} + \right.$$

$$\left. + \frac{x_{k+1}^{n-1}}{(x_{k+1} - x_k)(x_{k+1} - x_{k+2})} + \frac{x_{k+2}^{n-1}}{(x_{k+2} - x_k)(x_{k+2} - x_{k+1})} \right\} .$$

Opmerking

De uitdrukking tussen accoladen kan men ook schrijven als een veelterm in x_k, x_{k+1} en x_{k+2} .

Voor n=2 is de uitdrukking nul en voor n=3 is de uitdrukking gelijk aan een.
De fout is van de derde orde.

Men kan dit proces zo voortzetten, als men in het algemeen stelt:

$$F_k^{(i+1)} = \frac{x_{k+i+1}\, F_k^{(i)} - x_k\, F_{k+1}^{(i)}}{x_{k+i+1} - x_k} \qquad (5.3)$$

Voor $F_k^{(i)}$ geeft dit de formule

$$F_k^{(i)} = f_o + (-1)^i\, x_k\, x_{k+1} \cdots x_{k+i}\, R_i$$

met

$$R_i = \sum_{n=i+1}^{\infty} \sum_{j=o}^{i} \frac{x_{k+j}^{n-1}}{(x_{k+j} - x_k) \cdots (x_{k+j} - x_{k+j-1})(x_{k+j} - x_{k+j+1}) \cdots (x_{k+j} - x_{k+i})} .$$

Uit (5.3) kan men voor allerlei bijzondere gevallen extrapolatie formules maken.

Voorbeeld 1

Volgens pag. 64 geldt voor de omtrek s_k van de regelmatige k-hoek

$$s_k = 2\pi - \frac{\pi^3}{3}\frac{1}{k^2} + \frac{\pi^5}{60}\frac{1}{k^4} - \cdots$$

We willen π berekenen door k = 1,2,3,4,5 en 6 te nemen.

Neem hier $x_k = \frac{1}{k^2}$, dan is

$$s_k = F_k^{(o)} = 2\pi - \frac{\pi^3}{3} x_k + \frac{\pi^5}{60} x_k^2 - \ldots .$$

Volgens (5.3) wordt in dit geval de extrapolatie formule

$$F_k^{(i+1)} = \frac{(k+i+1)^{-2} F_k^{(i)} - k^{-2} F_{k+1}^{(i)}}{(k+i+1)^{-2} - k^{-2}}$$

$$= \frac{(k+i+1)^2 F_{k+1}^{(i)} - k^2 F_k^{(i)}}{(k+i+1)^2 - k^2}$$

k	$\frac{1}{2} s_k$			
1	0	0	2.6666666667	3.1277709950
2	2	2	3.0765371807	3.1406110524
3	$\frac{3}{2}\sqrt{3}$	2.5980762115	3.1245925845	14310001
4	$2\sqrt{2}$	2.8284271247	3.1353691705	5517768
5	$\frac{5}{4}\sqrt{10-2\sqrt{5}}$	2.9389262612	3.1388039518	
6	3	3		

1	3.1414670562	3.1415921858	3.1415926557
2	5871806	6426	
3	920357		
4			
5			
6			

$\pi = 3.1415926536$

Het driehoekig schema nadert in alle richtingen tot een limiet, maar niet in alle richtingen even hard.

Voorbeeld 2

Zij $F(h) = f_o + f_1 h^2 + f_2 h^4 + f_3 h^6 + \ldots$

en bereken $F(2^{-k}h)$ voor $k = 0,1,2, \ldots$.

Stel $x_k = (2^{-k}h)^2 = 4^{-k}h^2$.

Dan is

$$F(2^{-k}h) = F_k^{(o)} = f_o + f_1 x_k + f_2 x^2{}_k + f_3 x^3{}_k + \ldots .$$

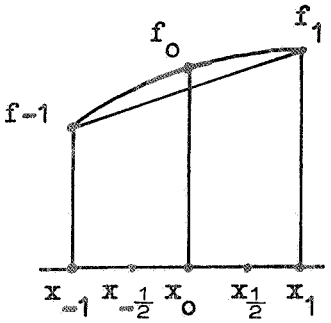
Formule (5.3) geeft

$$F_k^{(i+1)} = \frac{4^{-k-i-1}h^2 F_k^{(i)} - 4^{-k}h^2 F_{k+1}^{(i)}}{4^{-k-i-1}h^2 - 4^{-k}h^2}$$

$$= \frac{4^{i+1}F_{k+1}^{(i)} - F_k^{(i)}}{4^{i+1} - 1} = F_{k+1}^{(i)} + \frac{1}{4^{i+1} - 1}\left(F_{k+1}^{(i)} - F_k^{(i)}\right) .$$

(vergelijk dit met pag. 64 en 65).

We passen het in voorbeeld 2 behandelde toe op numerieke integratie.



Zij gevraagd $\int_{x_{-1}}^{x_1} f(x)dx$ te berekenen.

$(x_1 = x_o + h,\ x_{-1} = x_o - h)$.

De trapeziumregel, toegepast op het hele interval, geeft

$$\frac{1}{2h}\int_{x_{-1}}^{x_1} f(x)dx = \tfrac{1}{2} f_{-1} + \tfrac{1}{2} f_1 + \text{restterm}.$$

We bepalen de restterm door $f(x)$ om het punt x_o in een Taylorreeks te ontwikkelen.

$$f(x) = \sum_{k=o}^{\infty} a_k (x - x_o)^k \text{ met } a_k = \frac{1}{k!} f^{(k)}(x_o).$$

Term voor term integreren geeft

$$\frac{1}{2h}\int_{x_{-1}}^{x_1} f(x)dx = \sum_{k=o}^{\infty} \frac{a_{2k}}{2k+1}\, h^{2k} .$$

Anderzijds is

$$\tfrac{1}{2} f_{-1} + \tfrac{1}{2} f_1 = \sum_{k=o}^{\infty} a_{2k}\, h^{2k} .$$

Alleen de term voor k=o klopt; de termen met h^2 , h^4 enz. kloppen niet.

Dus

$$\frac{1}{2h}\int_{x_{-1}}^{x_1} f(x)dx = \tfrac{1}{2} f_{-1} + \tfrac{1}{2} f_1 + b_2 h^2 + b_4 h^4 + \ldots . \quad (5.4)$$

Hierin is $b_2 = -\frac{2}{3} a_2 = -\frac{1}{3} f^{(2)}(x_o)$.

We zullen nu een nauwkeurigere formule afleiden.
We passen de trapeziumregel tweemaal toe en wel op de deelintervallen (x_{-1}, x_o) en (x_o, x_1).

Dit geeft

$$\frac{1}{2h}\int_{x_{-1}}^{x_1} f(x)dx = \tfrac{1}{4} f_{-1} + \tfrac{1}{2} f_o + \tfrac{1}{4} f_1 + \tfrac{1}{4} b_2^* h^2 + \frac{1}{16} b_4^* h^4 + \ldots . \quad (5.5)$$

Uit de formules (5.4) en (5.5) willen we de term in h^2 elimineren.
We hebben hier niet precies de situatie van voorbeeld 2. Bij de overgang op het halve interval veranderen de coëfficiënten b_{2k} in de restterm in b_{2k}^*.
Toch kan men de extrapolatieformule (met i=o) uit voorbeeld 2 toepassen, omdat nog wel geldt $b_2 = b_2^*$.
We tonen dit aan door wat zorgvuldiger na te gaan hoe men formule (5.5) afleidt.

Men heeft

$$\frac{1}{h}\int_{x_o}^{x_1} f(x)dx = \tfrac{1}{2} f_o + \tfrac{1}{2} f_1 + b_2' \left(\tfrac{h}{2}\right)^2 + b_4' \left(\tfrac{h}{2}\right)^4 + \ldots$$

met

$$b_2' = -\frac{1}{3} f^{(2)}(x_o + \tfrac{1}{2}h) = -\frac{1}{3}\{f^{(2)}(x_o) + \frac{1}{1!}\frac{h}{2} f^{(3)}(x_o) + \tfrac{1}{2}(\tfrac{h}{2})^2 f^{(4)}(x_o) + \ldots\}$$

en

$$\frac{1}{h}\int_{x_{-1}}^{x_o} f(x)dx = \tfrac{1}{2} f_{-1} + \tfrac{1}{2} f_o + b_2''' (\tfrac{h}{2})^2 + b_4'' (\tfrac{h}{2})^4 + \ldots$$

met

$$b_2''' = -\frac{1}{3} f^{(2)}(x_o - \tfrac{1}{2}h) = -\frac{1}{3}\{f^{(2)}(x_o) - \frac{1}{1!}\frac{h}{2} f^{(3)}(x_o) + \frac{1}{2!}(\tfrac{h}{2})^2 f^{(4)}(x_o) + \ldots\}$$

Optellen en door 2 delen:

$$\frac{1}{2h}\int_{x_{-1}}^{x_1} f(x)dx = \tfrac{1}{4} f_{-1} + \tfrac{1}{2} f_o + \tfrac{1}{4} f_1 + \tfrac{1}{2}(b_2' + b_2'')(\tfrac{h}{2})^2 + \ldots \qquad (5.6)$$

Nu is $\frac{1}{2}(b_2' + b_2'') = -\frac{1}{3} f^{(2)}(x_o) + \frac{1}{8} h^2 f^{(4)}(x_o) + \ldots$

$$= b_2 + \tfrac{1}{8} h^2 f^{(4)}(x_o) + \ldots .$$

Men krijgt in (5.6) dus inderdaad als tweede orde term $\frac{1}{4} b_2 h^2$.
Deze term uit (5.4) en (5.5) elimineren volgens $F_{k+1} + \frac{1}{3}(F_{k+1} - F_k)$
geeft

$$\frac{1}{2h}\int_{x_{-1}}^{x_1} f(x)dx = \tfrac{1}{6} f_{-1} + \tfrac{4}{6} f_o + \tfrac{1}{6} f_1 + c_4 h^4 + c_6 h^6 + \ldots . \qquad (5.7)$$

Dit is juist de formule van Simpson met restterm.

We zetten nu deze techniek voort; we passen Simpson tweemaal toe op de deelintervallen (x_{-1}, x_o) en (x_o, x_1).

Dit geeft

$$\frac{1}{2h}\int_{x_{-1}}^{x_1} f(x)dx = \tfrac{1}{12} f_{-1} + \tfrac{4}{12} f_{-\frac{1}{2}} + \tfrac{1}{6} f_o + \tfrac{4}{12} f_{\frac{1}{2}} + \tfrac{1}{12} f_1 + \tfrac{1}{16} c_4 h^4 + \tfrac{1}{64} c_6^* h^6 + \ldots \qquad (5.8)$$

Ook hier worden de coëfficiënten in de restterm gestoord, alleen c_4 blijft ongewijzigd. Dit is voldoende om uit (5.7) en (5.8) de vierde orde term te elimineren volgens $F_{k+1} + \frac{1}{15}(F_{k+1} - F_k)$.

Dit geeft

$$\frac{1}{2h}\int_{x_{-1}}^{x_1} f(x)dx = \frac{1}{90}(7\,f_{-1} + 32\,f_{-\frac{1}{2}} + 12\,f_0 + 32\,f_{\frac{1}{2}} + 7\,f_1) + d_6\,h^6 + \ldots \qquad (5.9)$$

De trapeziumregel integreert een polynoom van de graad 1 exact. De 3-punts formule van Simpson integreert nog een polynoom van de graad 3 exact. De 5-punts formule (5.9) integreert nog een polynoom van de graad 5 exact.

Formule (5.9) tweemaal toepassen en extrapoleren volgens $F_{k+1} + \frac{1}{63}(F_{k+1} - F_k)$

levert een 9-punts formule.
Deze formule integreert een polynoom van de graad 7 exact.
Men krijgt op deze wijze een systeem van (2^n+1)-punts integratie formules $(n=0,1,2,3,\ldots)$, die nog een polynoom van de graad 2n+1 exact integreren.

De 2,3 en 5 punts formules uit het systeem zijn Newton-Cotes formules. De overige formules echter niet meer.
Dit is gemakkelijk in te zien. De 9-punts Newton-Cotes formule bijvoorbeeld integreert een polynoom van de graad 8 (zelfs nog een van de graad 9) exact, terwijl de 9-punts formule uit het systeem niet verder komt dan een polynoom van de graad 7.
Toch zijn de nieuwe formules niet slechter. Men kan namelijk bewijzen dat alle coëfficiënten in een formule positief zijn. Dit is een belangrijke eigenschap zoals uit de volgende stelling blijkt.

Stelling

Stel men heeft een systeem van integratieformules, d.w.z.

$$\int_a^b f(x)dx = \sum_{k=0}^{n-1} A_k^{(n)}\, f(x_k^{(n)}) + R_n\,,$$

waarbij voor $n \to \infty$ de graad van het nog exact geïntegreerde polynoom ook naar oneindig gaat.
Zij verder $A_k^{(n)} \geq 0$, dan geldt voor een continue functie $f(x)$

$$\int_a^b f(x)dx = \lim_{n\to\infty} \sum_{k=0}^{n-1} A_k^{(n)} f(x_k^{(n)}) \ .$$

Bewijs

Het bewijs berust op een stelling van Weierstrasz.
Deze stelling zegt dat, gegeven een continue functie $f(x)$ op het gesloten interval $[a,b]$, er bij elke $\varepsilon > 0$ een polynoom $p(x)$ bestaat zodanig dat $|f(x) - p(x)| < \varepsilon$ voor elke x uit $[a,b]$.

Kies nu een ε en zij $p(x)$ een polynoom met $|f(x) - p(x)| < \varepsilon$.

Volgens de veronderstelling kan men N zo groot kiezen dat alle integratie formules uit het systeem met $n > N$ het polynoom $p(x)$ exact integreren.

Dus $\int_a^b p(x)dx = \sum_{k=0}^{n-1} A_k^{(n)} p(x_k^{(n)})$ voor $n > N$.

Nu is $\left| \int_a^b f(x)dx - \int_a^b p(x)dx \right| \leq \int_a^b |f(x) - p(x)|dx < \varepsilon\,(b-a)$.

en $\left| \sum_{k=0}^{n-1} A_k^{(n)} f(x_k^{(n)}) - \sum_{k=0}^{n-1} A_k^{(n)} p(x_k^{(n)}) \right| \leq$

$$\sum_{k=0}^{n-1} |A_k^{(n)}| \, |f(x_k^{(n)}) - p(x_k^{(n)})| <$$

$$\varepsilon \sum_{k=0}^{n-1} |A_k^{(n)}| = \varepsilon \sum_{k=0}^{n-1} A_k^{(n)} = \varepsilon\,(b-a).$$

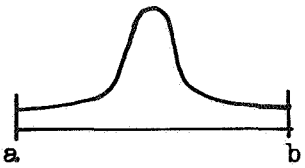
Dus $\left| \int_a^b f(x)dx - \sum_{k=0}^{n-1} A_k^{(n)} f(x_k^{(n)}) \right| \leq 2\varepsilon\,(b-a)$ voor $n > N$.

De hogere Newton-Cotes formules hebben ook negatieve coëfficïenten.
Hiervoor geldt de stelling niet.

De stelling garandeert dat als we het boven beschreven schema voortzetten, d.w.z. als we steeds meer basispunten in het interval (x_{-1},x_1) nemen, het proces convergeert met als limiet $\int_{x_{-1}}^{x_1} f(x)dx$.

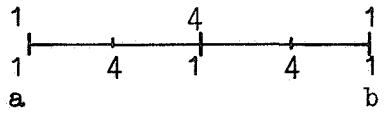
Opmerking

De integratie methode is afkomstig van Romberg.
Bij de Romberg integratie wordt het aantal basispunten bij elke stap verdubbeld. Het is echter onverstandig om dat te doen.



De bewering is dat Romberg verdubbelt waar het niet nodig is.
De piek in de figuur laat zich niet met een grof interval integreren. Tengevolge van de piek wordt het aantal basispunten groot, terwijl in feite de gladde stukken al lang klaar zijn.

Het is beter een schema op te zetten, dat alleen het aantal basispunten verdubbelt op die stukken van het interval waar het nodig is.
Dit kan bijvoorbeeld als volgt.



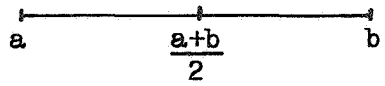
Pas Simpson toe op het hele interval en tweemaal op het halve interval.
Dit geeft $I(h)$ en $I(\frac{1}{2}h)$.

Extrapolatie geeft $I'(\frac{1}{2}h) = I(\frac{1}{2}h) + \frac{1}{15}\{I(\frac{1}{2}h) - I(h)\}$.

Het verschil $I(\frac{1}{2}h) - I(h)$ geeft aan of de vereiste precisie is bereikt. Een fout van deze orde zit niet meer in $I'(\frac{1}{2}h)$. We hopen dat de fout in $I'(\frac{1}{2}h)$ heel klein is t.o.v. dit verschil.

Als het verschil te groot is knippen we het interval definitief in tweeën en beperken de integratie tot

$(a, \frac{a+b}{2})$.



Met dit proces gaat men door tot het verschil klein genoeg is.
We hebben dan geïntegreerd over een interval met a als linker eindpunt.
Daarna zet men, met variabele stapgrootte, de integratie naar rechts voort.
Voordeel: men berekent alleen veel functiewaarden waar het nodig is.

Een andere toepassing van extrapolatie vindt men bij de numerieke differentiatie.

$$\left(\frac{df}{dx}\right)_{x=x_o} = \lim_{h \to o} \frac{f(x_o + h) - f(x_o - h)}{2h} .$$

$$\frac{f(x_o + h) - f(x_o - h)}{2h} = f'(x_o) + c_2 h^2 + c_4 h^4 + \ldots .$$

Dit is weer de situatie van voorbeeld 2.
We berekenen het linkerlid voor h, $\frac{1}{2}$ h, $\frac{1}{4}$ h enz. en extrapoleren met coëfficiënten $\frac{1}{3}$, $\frac{1}{15}$, $\frac{1}{63}$ enz.

Voorbeeld

$f(x) = \sin x \qquad x_o = 1 \qquad \cos 1 = 0.5403023$

h	
1	$\frac{0.9092974 - 0.0000000}{2} = 0.4546487$
0,5	$\frac{0.9974950 - 0.4794255}{1} = 0.5180695$
0.25	$\frac{0.9489846 - 0.6816388}{0.5} = 0.5346916$
0.125	$\frac{0.9022676 - 0.7675435}{0.25} = 0.5388964$
0.0625	$\frac{0.8735749 - 0.8060811}{0.125} = 0.5399504$

Extrapoleren:

0.4546487				
0.5180695	0.5392098			
0.5346916	0.5402323	0.5403005		
0.5388964	0.5402980	0.5403024	0.5403024	
0.5399504	0.5403017	0.5403019	0.5403019	0.5403019

De rijen in het schema convergeren, maar niet naar dezelfde waarde.
Een nieuwe rij zou weer een andere waarde van $f'(x_o)$ opleveren.
Oorzaak: de berekening van $\frac{1}{2h}\{f(x_o + h) - f(x_o - h)\}$ wordt gestoord door de afrondingsfouten in de waarden van sin x.

Naarmate h kleiner is wordt de afrondingsfout in $f(x_o + h) - f(x_o - h)$ met een groter getal vermenigvuldigd (aan de voorkant vallen cijfers weg). Voor h = 0.0625 heeft de laatste decimaal in 0.5399504 geen betekenis meer.

De moeilijkheid is om uit te maken wat het goede antwoord is. Als men een bepaalde precisie eist, volgens welk criterium moet men dan de berekening afbreken.

Er volgen nu enkele overwegingen die aangeven welke strategie men zou kunnen volgen.

1. Zij eps de geëiste precisie. We zoeken een rij waarin een aantal waarden in die precisie met elkaar overeenstemmen.
   De vervulling van deze eis zou men het liefst zien plaats vinden in de rij waar de storingen t.g.v. het wegvallen van cijfers van de orde eps worden.

2. In het voorbeeld is met h = 1 begonnen. Waarom niet met $h = 10^{-3}$?
   Als men een differentiatie procedure maakt voor een vooralsnog onbekende functie f(x) is het niet a priori duidelijk met welke h men het proces moet starten.
   Bij numerieke integratie ligt dit anders. Daar is op natuurlijke wijze een begininterval gegeven.

3. Men behoeft in het schema niet van boven naar beneden te werken.
   In het voorbeeld kan men ook met h = 0.0625 beginnen en het schema naar boven afmaken.

4. Als men weet hoe groot $f'(x_o)$ ongeveer is kan men een critieke h uitrekenen, waarvoor in $f(x_o + h) - f(x_o - h)$ precies zoveel cijfers weg vallen als men nog wil toelaten. Men kan dan met deze h beginnen en het schema naar boven afwerken.
   Nu is $f'(x_o)$ onbekend; gok daarom $f'(x_o) = 1$. Dit geeft een critieke h. Met deze h wordt $\frac{1}{2h}\{f(x_o + h) - f(x_o - h)\}$ berekend. Vallen er te veel cijfers weg dan wordt het schema naar boven voortgezet, anders naar beneden.

Opmerking

De procedure moet nog iets extra doen.
Het is niet altijd mogelijk de afgeleide in de gewenste precisie te bepalen.
De procedure moet hier tegen bestand zijn. De berekening moet gestopt worden als binnen een bepaalde driehoek van het schema de vereiste precisie nog niet bereikt is. Men kan de procedure de wel bereikte precisie laten opgeven.

We komen nog eens terug op het extrapolatie proces van pag. 66 e.v.
Men kan een eenvoudige interpretatie geven van formule (5.3). In de A-cursus zijn we deze formule al eerder tegengekomen, namelijk bij de theorie van de polynoom interpolatie.
We herhalen in het kort deze theorie.

Gegeven een functie $f(x)$ en $n+1$ basispunten $x_0, x_1, \ldots, x_n$.

Gevraagd een polynoom $f_n^*(x)$ van ten hoogste de graad n, met $f_n^*(x_i) = f(x_i)$ voor $i = 0, 1, \ldots, n$.

Het polynoom $f_n^*(x)$ is eenduidig bepaald. De formules van Lagrange en Newton stellen ons in staat $f_n^*(x)$ expliciet op te schrijven.
We leiden de formule van Newton af.
Principe: bouw de formule op door successievelijk een basispunt erbij te nemen.
Stel dat $f_{k-1}^*(x)$ met $f(x)$ overeenstemt in de basispunten $x_0, \ldots, x_{k-1}$.
Voeg het basispunt x_k toe. Bij het polynoom $f_{k-1}^*(x)$ komt nu een term van de graad k. Deze term moet nul zijn in $x_0, x_1, \ldots, x_{k-1}$, immers $f_{k-1}^*(x)$ heeft daar reeds de goede waarde.
De term heeft dus de vorm

$$(x - x_0)(x - x_1) \ldots (x - x_{k-1}) \times \text{constante factor}.$$

Voor de constante factor gebruiken we de notatie $f[x_0, x_1, \ldots, x_k]$.

Voor $f_n^*(x)$ kan men dan schrijven

$$f_n^*(x) = \sum_{k=0}^{n} f[x_0, x_1, \ldots, x_k](x - x_0) \ldots (x - x_{k-1}).$$

Opmerking

$f[x_0, x_1, \ldots, x_k]$ hangt niet af van de volgorde waarin de basispunten $x_0, \ldots, x_k$ zijn ingevoerd, immers $f[x_0, x_1, \ldots, x_k]$ is de coëfficiënt van x^k in $f_k^*(x)$ en dit polynoom is eenduidig bepaald.

$f[x_0, x_1, \ldots, x_k]$ is dus een symmetrische functie van x_0 t/m x_k.

Men kan nu gemakkelijk een recursie vergelijking voor de constante factoren afleiden.

Zij $f_{n-1}^*(x)$ het interpolatie polynoom op de basispunten $x_1, x_2, \ldots, x_n$ en zij $f_{n+1}^*(x)$ het polynoom op de basispunten $x_0, x_1, \ldots, x_n, x_{n+1}$.

Men kan $f^*_{n+1}(x)$ uit $f^*_{n-1}(x)$ afleiden door x_o en x_{n+1} toe te voegen.
Eerst x_o toevoegen geeft

$$f^*_{n-1}(x) + f\,[x_1, x_2, \ldots, x_n, x_o]\,(x - x_1) \ldots (x - x_n)$$

$$= f^*_{n-1}(x) + f\,[x_o, x_1, \ldots\ldots, x_n]\,(x - x_1) \ldots (x - x_n).$$

Vervolgens x_{n+1} toevoegen geeft

$$f^*_{n+1}(x) = f^*_{n-1}(x) + f\,[x_o, x_1, \ldots, x_n]\,(x - x_1) \ldots (x - x_n) +$$

$$+ f\,[x_o, \ldots, x_n, x_{n+1}]\,(x - x_o)(x - x_1) \ldots (x - x_n).$$

Men kan ook eerst x_{n+1} toevoegen en vervolgens x_o.
Dit geeft

$$f^*_{n+1}(x) = f^*_{n-1}(x) + f\,[x_1, \ldots, x_{n+1}]\,(x - x_1) \ldots (x - x_n) +$$

$$+ f\,[x_o, \ldots, x_{n+1}]\,(x - x_1) \ldots (x - x_n)(x - x_{n+1}).$$

Uit de twee uitdrukkingen voor $f^*_{n+1}(x)$ volgt

$$f\,[x_o, \ldots, x_{n+1}] = \frac{f\,[x_1, \ldots x_{n+1}] - f\,[x_o, \ldots, x_n]}{x_{n+1} - x_o} \qquad (5.10)$$

Hierbij is $f\,[x_k] = f(x_k)$.

De methode is efficiënt voor het handrekenen.
Men behoeft de differentie quotiënten (gedeelde differenties) maar eenmaal recursiefuit te rekenen. Wil men interpoleren voor verschillende waarden van x dan behoeven slechts de producten $(x - x_o) \ldots (x - x_k)$ opnieuw berekend te worden.
Voor automatisch rekenen heeft de methode het bezwaar dat een hele driehoek van differentie quotiënten onthouden moet worden.
We zouden liever voor een vaste x de interpolatie polynomen zelf recursief uitrekenen.
Dit kan volgens de methode van Neville, een verbetering van de methode van Aitken.

Notatie: $f(x \mid x_o, \ldots, x_n)$ is het polynoom dat op de basispunten $x_o, \ldots, x_n$ met de gegeven functie $f(x)$ overeenstemt.

Men kan $f(x \mid x_o, \ldots, x_{n+1})$ construeren uit $f(x \; x_o, \ldots, x_n)$ door toevoeging van x_{n+1} of uit $f(x \mid x_1, \ldots, x_{n+1})$ door toevoeging van x_o.

$$f(x \mid x_o, \ldots, x_{n+1}) = f(x \mid x_o, \ldots, x_n) + f\,[x_o, \ldots, x_{n+1}]\,(x - x_o) \ldots$$

$$\ldots (x - x_n)$$

$$f(x \mid x_o, \ldots, x_{n+1}) = f(x \mid x_1, \ldots, x_{n+1}) + f\,[x_o, \ldots, x_{n+1}]\,(x - x_1) \ldots$$

$$\ldots (x - x_{n+1})$$

Vermenigvuldig de eerste vergelijking met $(x - x_{n+1})$ en de tweede met $(x - x_o)$ en trek af. Dit geeft de recursie vergelijking

$$f(x \mid x_o, \ldots, x_{n+1}) = \frac{(x_{n+1} - x) f(x \mid x_o, \ldots, x_n) - (x_o - x) f(x \mid x_1, \ldots, x_{n+1})}{x_{n+1} - x_o}$$

Hierbij is $f(x \mid x_k) = f(x_k)$.

Voor $x = 0$ wordt de recursie vergelijking

$$f(0 \mid x_o, \ldots, x_{n+1}) = \frac{x_{n+1}\, f(0 \mid x_o, \ldots, x_n) - x_o\, f(0 \mid x_1, \ldots, x_{n+1})}{x_{n+1} - x_o} \qquad (5.11)$$

Formule (5.11) levert de waarden van de interpolatie polynomen in het punt nul.

Op pag. 66 waren wij geïnteresseerd in de machtreeks

$$F(x) = f_o + f_1 x + f_2 x^2 + \ldots .$$

De som van deze machtreeks was bekend voor een aantal waarden x_k, x_{k+1}, ... van x.

We willen berekenen $f_o = F(0)$.

Bepaal voor de basispunten x_k, x_{k+1},... volgens (5.11) de waarden van de interpolatie polynomen in het punt nul.
We starten de recursie door te nemen

$$f(x|x_k) = F(x_k) = F_k^{(o)}$$

Dan is de formule (5.11), op de notatie na, precies hetzelfde als formule (5.3).
Men kan (5.3) dus ook zien als een toepassing van polynoom interpolatie met de methode van Aitken-Neville.

Door een goede keuze van de basispunten krijgen de formules een prettige vorm (zie voorb. 2 op pag. 69).

De polynoom interpolatie is een van de methoden om een functie te approximeren. Er zijn heel andere methoden. De vraag is steeds: hoe benadert men een functie met elementen uit een gegeven verzameling van functies.
In het volgende wordt iets verteld over benaderingen met behulp van rationale functies. De rationale functies vormen een rijkere klasse dan de polynomen. Met rationale functies kan men functies met polen approximeren.

Bij deze benaderingen spelen kettingbreuken een rol. De theorie van de kettingbreuken is een vak op zichzelf. Wij zullen zeer in het kort het begrip kettingbreuk toelichten.

Voorbeeld van een oneindige kettingbreuk:

$$a_0 + \cfrac{b_1}{a_1 + \cfrac{b_2}{a_2 + \cfrac{b_3}{a_3 + \ddots + \cfrac{a_n}{b_n + \ddots}}}}$$

Men kan een oneindige kettingbreuk afbreken.

Bijv.

$$a_o + \frac{b_1}{a_1} = \frac{a_o a_1 + b_1}{a_1} = \frac{P_1}{Q_1}$$

$$a_o + \cfrac{b_1}{a_1 + \cfrac{b_2}{a_2}} = \frac{a_o a_1 a_2 + a_o b_2 + a_2 b_1}{a_1 a_2 + b_2} = \frac{P_2}{Q_2} .$$

$$a_o + \cfrac{b_1}{a_1 + \ddots + \cfrac{a_n}{b_n}} = \frac{P_n}{Q_n} .$$

Een eindig beginstuk noemt men een convergent. De limiet (als deze bestaat) van de rij convergenten is de waarde van de oneindige kettingbreuk.

De berekening van convergenten wordt vergemakkelijkt doordat er een recurrente betrekking van de tweede orde bestaat voor de P's; eveneens is er een betrekking voor de Q's.
Probeer deze betrekkingen zelf te vinden.

Er zijn kettingbreuken van een speciale vorm.
Bijv.

$$a_o + \cfrac{x}{a_1 + \cfrac{x}{a_2 + \cfrac{x}{a_3 + \ldots}}}$$

De waarde van de kettingbreuk is een functie f(x) van de variabele x.
Als men afbreekt krijgt men benaderingen voor f(x).

$$a_o + \cfrac{x}{a_1 + \frac{x}{a_2}} = \frac{a_o a_1 a_2 + (a_o + a_2)x}{a_1 a_2 + x}$$

$$a_o + \cfrac{x}{a_1 + \cfrac{x}{a_2 + \frac{x}{a_3}}} = a_o + \frac{a_2 a_3 x + x^2}{a_1 a_2 a_3 + (a_1 + a_3)x} .$$

De convergenten zijn rationale functies van x.
Als men bij de theoretische behandeling van een probleem in staat is een functie in een kettingbreuk te ontwikkelen, dan heeft men dus de beschikking over een rij rationale benaderingen voor die functie.

Zij $f(x)$ gegeven door een machtreeks.

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots .$$

Men kan nu $f(x)$ in een kettingbreuk ontwikkelen.

$$f(x) = a_0 + a_1 x \left(1 + \frac{a_2}{a_1} x + \frac{a_3}{a_1} x^2 + \ldots\right)$$

$$= a_0 + \cfrac{a_1 x}{\cfrac{1}{1 + \frac{a_2}{a_1} x + \ldots}} .$$

De noemer $1/1 + \frac{a_2}{a_1} x + \ldots$ kan weer in een machtreeks $1 + b_1 x + b_2 x^2 + \ldots$ ontwikkeld worden.

Dit geeft

$$f(x) = a_0 + \cfrac{a_1 x}{1 + b_1 x + b_2 x^2 + \ldots} .$$

Op de machtreeks $1 + b_1 x + b_2 x^2$ kan men hetzelfde proces toepassen, enz.

Voorbeeld

$$e^x = 1 + x + \tfrac{1}{2} x^2 + \tfrac{1}{6} x^3 + \ldots$$

$$= 1 + x\left(1 + \tfrac{1}{2} x + \tfrac{1}{6} x^2 + \ldots\right)$$

$$= 1 + \cfrac{x}{\cfrac{1}{1 + \frac{1}{2} x + \frac{1}{6} x^2 + \ldots}} = 1 + \cfrac{x}{1 - \frac{1}{2} x + \frac{1}{12} x^2 + \ldots}$$

$$= 1 + \cfrac{x}{1 - \cfrac{\frac{1}{2} x}{1 + \ldots}}$$

Als approximaties voor e^x vindt men

$$e^x \approx 1$$

$$e^x \approx 1 + x$$

$$e^x \approx 1 + \frac{x}{1 - \frac{1}{2}x} = \frac{2 + x}{2 - x}$$

De partiële sommen van de machtreeks geven als benaderingen voor e^x

$$e^x \approx 1$$

$$e^x \approx 1 + x$$

$$e^x \approx 1 - x + \tfrac{1}{2}x^2 .$$

Locaal (in de omgeving van $x = 0$) gedragen de benaderingen zich op dezelfde manier, maar in het groot is het gedrag van de convergenten geheel anders dan van de partiële sommen.

De kettingbreuk convergeert soms veel beter dan de oorspronkelijke machtreeks. Het convergentiegebied van de kettingbreuk is vaak veel groter dan van de machtreeks.

Er zijn een aantal interpolatiemethoden die uitgaan van rationale functies. Uit de vorige eeuw is er het werk van Thiele. Verder noemen we de Q D algorithme van Rutishauser en de ε algorithme van Peter Wynn. De theorie van deze methoden slaan we over.

Bij de Q D (quotiënt-differentie) algorithme wordt een driehoekig schema gemaakt door om de beurt een differentie te vormen en een deling uit te voeren.
Bij de ε algorithme voert men een uniform proces uit.
We zullen deze algorithme nu beschrijven.

We gaan uit van een rij $f_0, f_1, f_2, \ldots$.

	f_0			
o		εf_0		
	f_1		$\varepsilon^2 f_0$	
o		εf_1		$\varepsilon^3 f_0$
	f_2		$\varepsilon^2 f_1$	
o		εf_2		$\varepsilon^3 f_1$
	f_3		$\varepsilon^3 f_2$	
o		εf_3		
	f_4			

De formeringswijze van het schema is als volgt.

$$\begin{array}{ccc|c} & & \varepsilon^{n} f_k & \\ \varepsilon^{n-1} f_{k+1} & & & \varepsilon^{n+1} f_k \\ & & \varepsilon^{n} f_{k+1} & \end{array}$$

Neem aan dat de grootheden links van de lijn bekend zijn. Dan wordt $\varepsilon^{n+1} f_k$ berekend met de ruitformule

$$\varepsilon^{n+1} f_k = \varepsilon^{n-1} f_{k+1} + \frac{1}{\varepsilon^{n} f_{k+1} - \varepsilon^{n} f_k}$$

Om te starten vult men het schema links van de kolom f_o, f_1, f_2, ... aan met een kolom nullen.

Als f_o, f_1, f_2, ... partiële sommem zijn van een machtreeks, dan zijn f_o, $\varepsilon^2 f_o$, $\varepsilon^4 f_o$ enz. de convergenten van de corresponderende kettingbreuk.

Op deze samenhang met de kettingbreuken gaan we verder niet in. Zoals reeds opgemerkt convergeert de kettingbreuk vaak beter dan de oorspronkelijke machtreeks. Het schema is dan goed voor extrapolatie te gebruiken.

We passen nu de ε algorithme toe op een voorbeeld. Zij weer π te berekenen uit de omtrek s_n van de regelmatige n-hoek.

$$s_n = n \sin \pi/n = \pi - \frac{\pi^3}{3!} n^{-2} + \frac{\pi^5}{5!} n^{-4} + \ldots.$$

(cirkel met middenlijn 1)

We gebruiken nu de regelmatige 2/1, 2/2, 2/3, 2/4 enz. hoek.
Eerst Neville toepassen met $x_n = \frac{1}{n^2}$ geeft:

n	x_n	s_n						
2	1/4	2						
			8/3					
1	1	0		44/15				
			8/15		64/21			
2/3	9/4	-2/3				976/315		
			-32/21				10816/3465	
1/2	4	0						141088/45045
			-32/45					47104/15015
2/5	25/4	2/5						
			72/55					
1/3	9	0						
			72/91					
2/7	49/4	-2/7						
1/4	16	0						

$\frac{47104}{15015} = 3.137$

De rij 2, 8/3, 44/15, ... convergeert inderdaad naar π.
Men kan dit als volgt inzien.

$$\pi/4 = \arctan 1 = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \ldots$$

Deze reeks euleren geeft zoals bekend een convergente reeks. Men vindt gemakkelijk

$$\pi = 2 \sum_{n=0}^{\infty} \frac{n!}{1.3.5.\ldots.(2n+1)} = 2 + \frac{2}{3} + \frac{4}{15} + \frac{4}{35} + \ldots \qquad (5.12)$$

De partiële sommen van deze reeks zijn 2, 8/3, 44/15, 64/21, ...
Neville levert dus juist deze partiële sommen.

In de reeks (5.12) nadert de verhouding van twee opeenvolgende termen tot een $\frac{1}{2}$.
Om 9 decimalen te krijgen zijn ongeveer 30 termen nodig. We hebben echter geen zin om zoveel termen uit te rekenen.
We willen daarom de rij 2, 8/3, 44/15, ... aan een of ander extrapolatie proces onderwerpen. Euler helpt hier niet.

Zij $f_n = a_o + \sum_{k=1}^{m} a_k \lambda_k^n$ en $|\lambda_k| < 1$

Dan is $\lim_{n\to\infty} f_n = a_o$.

Onder deze omstandigheden kan men op de rij f_o, f_1 , f_2 de ε algorithme toepassen.
De rij f_o, $\varepsilon^2 f_o$, $\varepsilon^4 f_o$, ... convergeert dan naar a.

Als f_n een partiële som is van een convergente meetkundige reeks, is aan de voorwaarde voldaan (ga na).
De reeks (5.12) is geen meetkundige reeks, maar gedraagt zich als een meetkundige reeks met reden $\frac{1}{2}$.

We kunnen dus de ε algorithme proberen.

```
      2
0           3/2
      8/3           28/9
0           15/4           195/4
     44/15          47/15          424/135
0           35/4
     64/21
0      .
       .
       .
       .
```

Men krijgt als getransformeerde rij

2 28/9 424/135 190112/60515 541312/172305

541312/172305 = 3.14159 19445
π = 3.14159 26536

Dat de getransformeerde rij inderdaad naar π convergeert is niet zo gemakkelijk te bewijzen.
We proberen in dit voorbeeld uit het aanwezige materiaal te halen wat er in zit. Niets belet ons om op de getransformeerde rij nogmaals de ε algorithme toe te passen.
Dit geeft

2 688/219 412813616/131402655 = 3.14159 26566

Nogmaals transformeren geeft

2 1255 597258576/399669019913 = 3.1415926580

Dit is geen verbetering meer.

Opmerking

De ε algorithme wordt gestart door een kolom nullen toe te voegen.

$$
\begin{array}{lll}
 & f_0 & \\
0 & \dfrac{1}{\nabla f_1} & \\
 & f_1 & f_1 - \dfrac{\Delta f_1 \, \nabla f_1}{\delta^2 f_1} \\
0 & \dfrac{1}{\Delta f_1} & \\
 & f_2 &
\end{array}
$$

We zien dat de kolom $\varepsilon^2 f_0$, $\varepsilon^2 f_1$,... in feite wordt verkregen door op f_0, f_1, f_2,... de extrapolatie van Aitken toe te passen.
Men zou op ieder drietal elementen van $\varepsilon^2 f_0$, $\varepsilon^2 f_1$,... wederom Aitken kunnen toepassen. Dit geeft dan een ander resultaat dan de ε-algorithme.

Men kan aantonen dat de ε algorithme machtiger is dan de herhaalde toepassing van Aitken.

6. Integratie

De integratie methode van Romberg is niet efficient.
Op pag.75 is het principe aangegeven om een integratie proces efficienter uit te voeren.
We geven nu een eenvoudig programma dat volgens dit principe werkt.

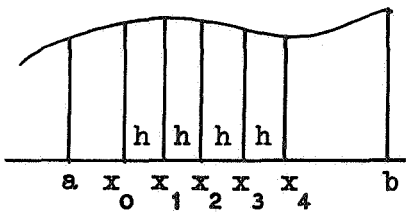
```
real procedure INT (x, a, b, y, eps); value a, b, eps; real x,a,b,y,eps;
begin boolean ls; real h,F,V,f0,f4,I;
      I := 0; x := a; f0 := y;
 m1 : h := (b-a)/4; ls := true;
 m2 : x := a+h; F := f0+4×y;
      x := x+h; V := y; F := F+2×V;
      x := x+h; F := F+4×y;
      x := x+h; f4 := y; F := F+f4;
      V := 8×V + 2×(f0+f4) - F;
      if abs (V) ≥ eps then begin h := h/2; ls := false end else
      begin a := x; f0 := f4; I := h×(F-V/15)+I;
            if ls then goto m3;
            if abs (V) ≤ eps/20 then h := 2×h;
            if sign (b-a-4×h) ≠ sign (h) then goto m1
      end;
      goto m2;
 m3 : INT := I/3
end INT;
```

Toelichting

De procedure INT berekent $\int_a^b y\,dx$.

Zij de integratie gevorderd tot x_0.

Simpson wordt toegepast op (x_0,x_4) en op de deelintervallen (x_0,x_2) en (x_2,x_4).



Het resultaat wordt geaccepteerd als $|V| <$ eps is.

De stapgrootte wordt gehalveerd voor $|V| \geq$ eps en verdubbeld voor $|V| <$ eps/20.

Bij de verdubbeling wordt V ongeveer 16 maal zo groot.

De factor $\frac{1}{20}$ zorgt er voor dat niet al te voorbarig wordt verdubbeld.

Het programma test of x_0+4h de grens b overschrijdt.

De boolean ls geeft aan of x_4 gelijk is aan b.

Het programma is niet efficiënt. Het proces van halveren en verdubbelen is nog te grof.

We trachten een betere strategie te vinden.
Zij discr (discrepantie) de absolute waarde van grootheid V van de procedure, dus

$$\text{discr} = |V| = |2(f_0 + 4f_2 + f_4) - (f_0 + 4f_1 + 2f_2 + 4f_3 + f_4)| \ ,$$

en zij tol een gegeven tolerantie (de parameter eps van de procedure).

Een integratiestap met stapgrootte h levert de waarde van discr. De stap wordt geaccepteerd als discr $<$ tol en verworpen als discr $\geq$ tol.

Men kan nu uitrekenen voor welke stapgrootte h* zou gelden discr* = tol. Immers discr is ongeveer evenredig met h^4 (zie formules (5.7) en (5.8)), dus

$$\left(\frac{h^*}{h}\right)^4 \approx \frac{\text{discr}^*}{\text{discr}} = \frac{\text{tol}}{\text{discr}} \ .$$

Men vindt

$$h^* = h\left(\frac{\text{discr}}{\text{tol}}\right)^{-\frac{1}{4}} \qquad (6.0)$$

Na verwerping wordt de integratiestap opnieuw uitgevoerd met stapgrootte h*; na acceptering wordt de volgende integratiestap met h* uitgevoerd. Het is echter gevaarlijk om h* volgens (6.0) te kiezen. Om het risico van verwerping te verkleinen is het beter de nieuwe stapgrootte iets kleiner te kiezen, bijv.

$$h^* = 0.95\ h\left(\frac{\text{discr}}{\text{tol}}\right)^{-\frac{1}{4}} \qquad (6.1)$$

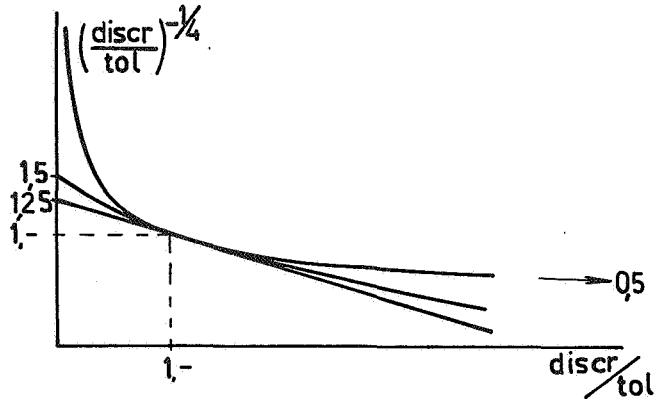
Men heeft dan een veiligheidsmarge van 20% in discr.

Men kan $\left(\frac{\text{discr}}{\text{tol}}\right)^{-\frac{1}{4}}$ vervangen door

$$0.5 + \frac{1}{1 + \frac{\text{discr}}{\text{tol}}} = 0.5 + \frac{\text{tol}}{\text{discr} + \text{tol}} \ .$$

Dit geeft in de omgeving van $\frac{\text{discr}}{\text{tol}} = 1$ een goede benadering voor $\left(\frac{\text{discr}}{\text{tol}}\right)^{-\frac{1}{4}}$.

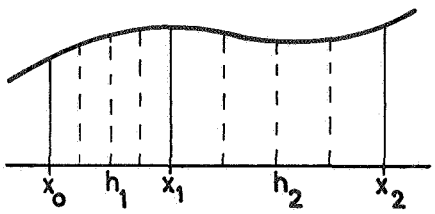
De functie $0.5 + \frac{1}{1+x}$ heeft namelijk in het punt $x = 1$ de zelfde waarde en de zelfde afgeleide als $x^{-\frac{1}{4}}$.



Rekening houdend met de veiligheidsmarge kan men nemen

$$h^* = \left(0{,}45 + \frac{tol}{discr + tol}\right) \times h \qquad (6{,}2)$$

Uit experimenten blijkt dat deze strategie nog niet bevredigend is.
Als de integrand snel van karakter wisselt, bijv. plotseling zeer stijl gaat lopen, werkt de strategie niet goed. Dit kan men als volgt inzien.
Volgens (6.2) is na accepteren van het resultaat $|h^*| \geq 0{,}95\,|h|$.
Het gedrag van de integrand kan zo slecht worden dat de stapgrootte bij iedere stap met meer dan 5% moet afnemen.
Extrapoleren van h met (6.2) heeft dan tot gevolg dat na elke geaccepteerde stap de volgende stap wordt verworpen.
Dit kan worden voorkomen door een verbeterde extrapolatie van de stapgrootte, gebaseerd op de resultaten van twee integratiestappen.



Zij de integratie gevorderd tot x_2. Van twee vorige integratiestappen over intervallen ter lengte van h_1 resp. h_2 zijn de discrepanties $discr_1$ en $discr_2$ bekend.

Op grond hiervan willen we de lengte h van het nieuwe integratie interval bepalen.

Zij H de maximaal toelaatbare stap, als we beginnen te integreren bij x, d.w.z. de benadering voor

$$\int_x^{x+H} f(t)\,dt$$

zal nog juist worden geaccepteerd.

H is een functie van x.
Ontwikkel H(x) in een machtreeks om het punt x_o.

$$H(x) = H_o + H_1 (x - x_o) + H_2 (x - x_o)^2 + \ldots$$

De kwadratische en de hogere termen van de machtreeks zullen we verwaarlozen.

We zijn geïnteresseerd in

$$H(x_2) = H_o + (h_1 + h_2)\, H_1 . \qquad (6.3)$$

Nu is (zie (6.0)

$$H(x_o) = H_o = h_1 \left(\frac{\text{discr}_1}{\text{tol}}\right)^{-\frac{1}{4}}$$

en

$$H(x_1) = H_o + h_1 H_1 = h_2 \left(\frac{\text{discr}_2}{\text{tol}}\right)^{-\frac{1}{4}} ,$$

dus

$$H_1 = \frac{h_2}{h_1} \left(\frac{\text{discr}_2}{\text{tol}}\right)^{-\frac{1}{4}} - \left(\frac{\text{discr}_1}{\text{tol}}\right)^{-\frac{1}{4}} .$$

Substitutie van H_o en H_1 in (6.3) geeft

$$\frac{H(x_2)}{h_2} = \frac{h_2}{h_1} \left(\frac{\text{discr}_2}{\text{tol}}\right)^{-\frac{1}{4}} + \left(\frac{\text{discr}_2}{\text{tol}}\right)^{-\frac{1}{4}} - \left(\frac{\text{discr}_1}{\text{tol}}\right)^{-\frac{1}{4}}$$

We approximeren $\left(\frac{\text{discr}}{\text{tol}}\right)^{-\frac{1}{4}}$ weer door $\frac{\text{tol}}{\text{discr} + \text{tol}} + 0.5$.

Met de notatie $\text{mu} = \frac{\text{tol}}{\text{discr} + \text{tol}}$ geeft dat

$$\frac{H(x_2)}{h_2} = \frac{h_2}{h_1} (\text{mu}_2 + 0.5) + \text{mu}_2 - \text{mu}_1 .$$

Als we tenslotte nog rekening houden met een veiligheidsmarge vinden we voor de nieuwe stap h de extrapolatie formule

$$\frac{h}{h_2} = \frac{h_2}{h_1} (\text{mu}_2 + 0.45) + \text{mu}_2 - \text{mu}_1 . \qquad (6.4)$$

De strategie is nu als volgt.
Als een integratie resultaat wordt verworpen omdat h_2 te groot is, dan wordt een h_2* bepaald volgens (6.2) en opnieuw geïntegreerd.
Als een integratie resultaat wordt geaccepteerd, dan wordt de h voor de volgende stap bepaald volgens (6.4).
Bij de start van het integratieproces is nog geen resultaat geaccepteerd; voor de extrapolatie moet dus (6.2) worden gebruikt.

Er volgt nu een integratie procedure die volgens deze strategie werkt.

```
real procedure int   (x, a, b, y, eps);
value a, b; real x, a, b, y, eps;

begin Boolean ls, first; integer signh;
      real bd, h, H, H1, mu, mu1, fH, fo, f1,
      f2, f3, f4, F, V, I, abs V;

      signh := sign (b - a);
      x := a; fo := y; first := true;
 m1 : bd := b; H := b - a; ls := true;
 m2 : h := H/4;
      x := a + h; f1 := y;
      x := x + h; f2 := y;
      x := x + h; f3 := (y + f1) × 4;
      x := bd; f4 := y; f1 := fo + f4;
      F := f2 × 2 + f1 + f3;
      V := f2 × 6 + f1 - f3; abs V := abs (V);
      mu := eps/(eps + abs V);
      if first then begin if abs V < eps then
                        begin H1 := H; mu1 := mu;
                              first := false;
                              a := bd; fo := f4;
                              I := (F - V/15)× h;
                              if ls then goto m3
                        end; goto reject
                  end
            else if abs V ≥ eps then
reject :            begin H := (mu + 0.45) × H;
                        ls := false; bd := a + H
                  end
                                    else
```

```
                    begin fH := (mu + 0.45) × H/H1 + mu - mu1;
                          mu1 := mu; H1 := H;
                          H := H × fH; a := bd;
                          fo := f4; bd := bd + H;
                          I := (F - V/15) × h + I;
                          if ls then goto m3;
                          if sign (b - bd) ≠ signh then
                          goto m1;
                    end;
          goto m2;
m3 :  int    := I/3
end int .
```

Toelichting

De boolean ls heeft dezelfde betekenis als op pag. 89.
De boolean first is true als nog geen interval is geaccepteerd, dus bij het begin van de integratie.
Merk op dat x_4 (notatie van pag. 89) niet berekend wordt met de statement x := x + h, maar met x := bd, waarbij bd het rechtereindpunt is van het integratie interval.
Dit is een voorzorgsmaatregel, omdat a + H ≠ a + 4 × (H/4) is.
Hierdoor wordt voorkomen dat men bijv. bij

$$\int_0^1 \sqrt{1-x^2}\,dx$$

net over de bovengrens 1 zou komen.

Ter inleiding op de integratie van differentiaal vergelijkingen beschouwen we de vergelijking van van der Pol

$$\frac{d^2y}{dx^2} - c(1 - y^2)\frac{dy}{dx} + y = 0. \qquad (6.5)$$

Dit is een niet lineaire vergelijking.
We trachten een inzicht te krijgen in het gedrag van de oplossingen van deze vergelijking.

Beschouw eerst de lineaire d.v.

$$\frac{d^2y}{dx^2} - c\frac{dy}{dx} + y = 0.$$

Probeer als oplossing e^{tx}.
Dit geeft de karakteristieke vergelijking

$$t^2 - ct + 1 = 0$$

Dus $t = \frac{c}{2} \pm \sqrt{\frac{c^2}{4} - 1}$.

Voor kleine c is

$$y = e^{\frac{c}{2}x} \left(\lambda \cos \sqrt{1 - \frac{c^2}{4}}\, x + \mu \sin \sqrt{1 - \frac{c^2}{4}}\, x\right).$$

Voor $c < 0$ is de oplossing een gedempte trilling,

voor $c > 0$ is de trilling niet gedempt.

Beschouw nu weer (6.5). Neem de c in (6.5) positief.
Zij y heel klein, dan is $1 - y^2 \approx 1$. De oplossing gedraagt zich locaal als een ongedempte trilling; y blijft niet klein.

Zij y wat groter, dan is $1 - y^2$ negatief. De oplossing gedraagt zich locaal als een gedempte trilling.
Wij vermoeden dat de oplossing van (6.5) op de duur het karakter krijgt van een trilling met constante amplitude.

Stel nu $\frac{dy}{dx} = p$.

Dan is $\frac{d^2y}{dx^2} = \frac{dp}{dx} = \frac{dp}{dy} \cdot \frac{dy}{dx} = p\frac{dp}{dy}$.

Hiermee gaat (6.5) over in het stelsel

$$\frac{dy}{dx} = p$$

$$p\frac{dp}{dy} - c(1 - y^2)p + y = 0.$$

Langs grafische weg proberen we een inzicht te krijgen in de oplossingen van de laatste differentiaal vergelijking.

In het (p,y)-vlak tekenen we isoklinen, dat zijn krommen waarop $\frac{dp}{dy}$ constant is.

$$\frac{dp}{dy} = c(1 - y^2) - \frac{y}{p} .$$

Voor $c = 0$ zijn de isoklinen rechten $p = -\frac{1}{\lambda} y$ door de oorsprong.

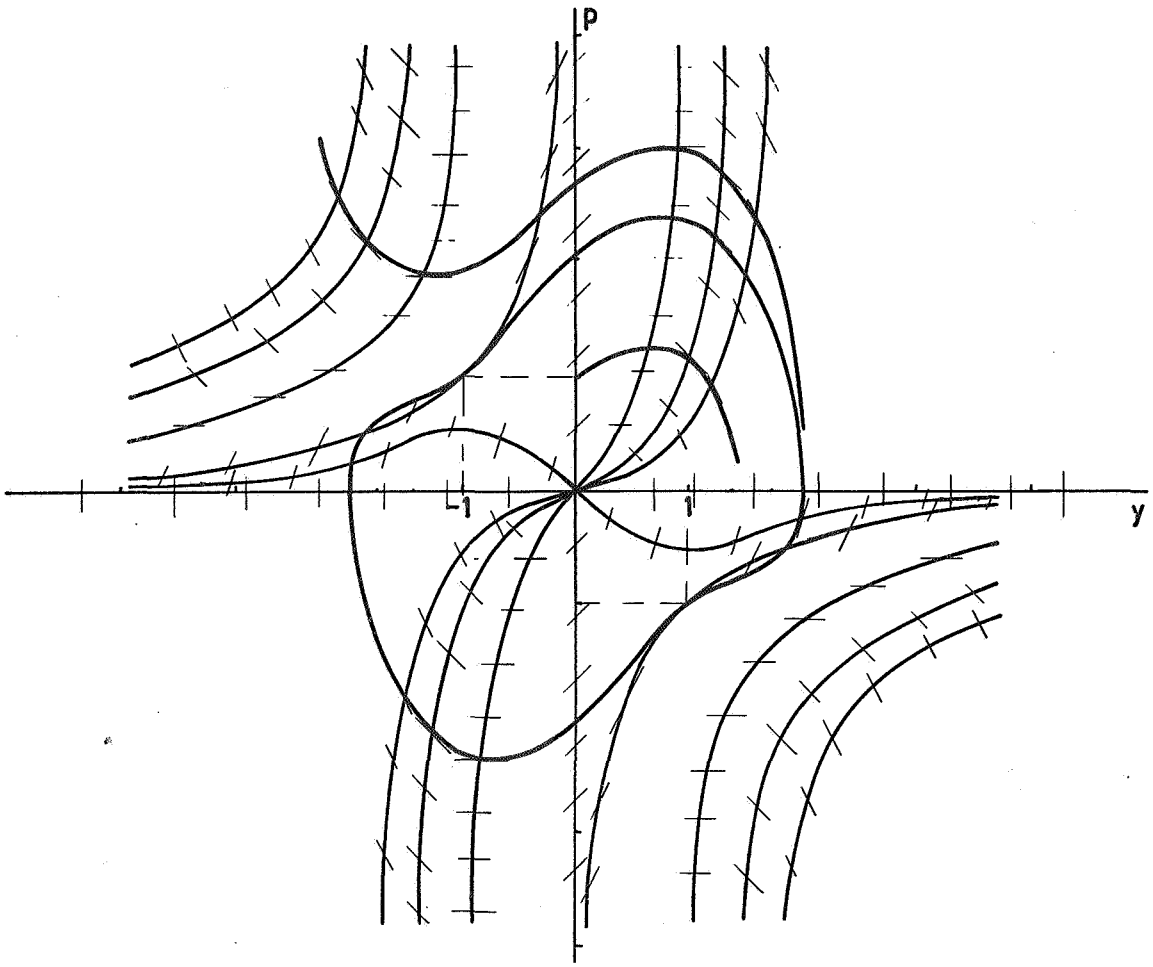
Op zo'n rechte is $\frac{dp}{dy} = \lambda$. In elk punt van de isokline staat de raaklijn aan een oplossing loodrecht op de isokline. De oplossingen zijn concentrische cirkels met de oorsprong als middelpunt. De algemene oplossing van $\frac{dp}{dy} = -\frac{y}{p}$ is inderdaad $p^2 + y^2 = C$.

We beschouwen nu het geval $c = 1$.

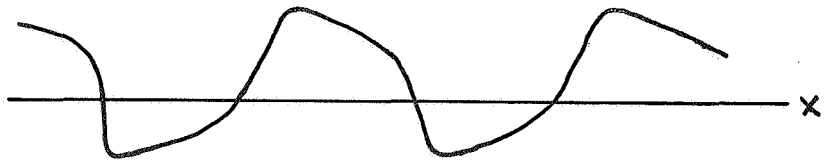
$$\frac{dp}{dy} = 1 - y^2 - \frac{y}{p} .$$

Op de y-as is $\frac{dp}{dy}$ oneindig. Op de y-as heeft elke oplossing een verticale raaklijn. In de figuur zijn verder de isoklinen getekend voor $\frac{dp}{dy} = -2, -1, 0, 1$ en 2.

Hoe men ook start, op de duur komt men altijd terecht op de in de figuur getekende gesloten kromme. Men noemt deze kromme een limiet cyclus.



Aangezien men in het (p,y)-vlak altijd op de limiet cyclus terecht komt zal een oplossing van de vergelijking van van der Pol op de duur periodiek worden. De oplossingen stellen trillingen voor, en wel z.g. relaxatie trillingen.
De grafiek van een oplossing ziet er ongeveer als volgt uit



Voor grotere waarden van de parameter c in de verg. van van der Pol krijgt men scherpe knikken in de grafiek.

We beschouwen nu de integratie van differentiaal vergelijkingen met behulp van de Runge-Kutta methode.
Het principe van de methode (zie ook syll. numerieke analyse pag. 40) is als volgt.

Zij gegeven de vergelijking $y' = f(x,y)$ met $y(x_o) = y_o$.
Gevraagd wordt $y(x_o + h)$ te berekenen.

Zij $\quad k_i = h\, f(x_i, y_i)$

met $\quad x_i = x_o + M_i\, h$

en $\quad y_i = y_o + \sum_{j=o}^{i-1} L_{ij}\, k_j$.

Bereken achtereenvolgens de m grootheden k_o, k_1 t/m k_{m-1} en vorm het gewogen gemiddelde

$$\Delta y = \sum_{i=o}^{m-1} a_i\, k_i.$$

De parameters M_i, L_{ij} en a_i worden zo gekozen dat $y^*(x_o + h) = y_o + \Delta y$ een goede benadering voor $y(x_o + h)$ is.
Daartoe worden $y^*(x_o + h)$ en $y(x_o + h)$ ontwikkeld in machtreeksen.

$$y^*(x_o + h) = y_o + \sum_{i=1}^{\infty} b_i\, h^i$$

$$y(x_o + h) = y_o + \sum_{i=1}^{\infty} c_i\, h^i.$$

Hierin zijn de b_i's ingewikkelde uitdrukkingen in de partiële afgeleiden van $f(x,y)$.
Bij vast gekozen m kiezen we de parameters nu zo dat zoveel mogelijk termen van de beide machtreeksen overeenstemmen. Dit leidt tot een stelsel niet lineaire vergelijkingen voor de parameters.
De exponent van h in de laatste term die nog overeenstemt noemt men de orde van het Runge- Kutta proces.

Met drie punten (m=3) kan men een derde orde R-K proces maken. Aangezien het aantal parameters groter is dan het aantal vergelijkingen zijn een aantal varianten mogelijk.
Voor een vierde orde proces heeft men vier punten nodig. Zie voor de klassieke 4-de orde R-K formule syll. NA pag. 42. Ook hier zijn varianten mogelijk.
Het is niet mogelijk met vijf punten een vijfde orde proces te maken; met zes punten lukt dit echter wel.

Bij de berekening van integralen met Simpson hebben zij gezien hoe men de procedure zelf de stapgrootte kan laten regelen.
Hetzelfde principe kan men ook bij de R-K integratie toepassen.
Integreer over een interval en daarna twee maal over het halve interval.
Op grond van het verschil van beide uitkomsten wordt de stap verworpen of geaccepteerd. Tevens kan men de stapgrootte aanpassen en een correctie op het resultaat in rekening brengen.
De procedure voor de klassieke 4-de orde R-K formule is te vergelijken met de Simpson procedure.
De fout is $O(h^5)$. Een schatting voor de fout, nl. 1/15 van het verschil kan men als correctie in rekening brengen, waardoor het resultaat een orde beter wordt. De strategie voor de verandering van h zoals beschreven op pag. 90 e.v. kan hier op precies dezelfde manier worden toegepast.

De halvering van het interval vereist bij de formule van Simpson de berekening van twee nieuwe punten.
In dat opzicht is hier de situatie veel ongunstiger. Bij de halvering van het interval moet men zeven nieuwe punten uitrekenen (behalve $f(x_o, y_o)$ zijn de reeds berekende waarden van $f(x,y)$ niet te gebruiken). Een integratiestap vergt zo in totaal de berekening van 4 + 7 = 11 punten.

Men kan zich afvragen of het mogelijk is een R-K integratie wat efficiënter uit te voeren.

Zij het R-K proces van de orde r, dan is de term met h^r de laatste die in beide Taylor-reeksen overeenstemt.

Noem deze term $t\,h^r \Delta y$.
Het increment Δy wordt gevormd door een lineaire combinatie van k_i's.
Kan men $t\,h^r \Delta y$ ook door een lineaire combinatie van k_i's berekenen ?

Bij een 2-punts formule lukt dit; bij de hogere orde formules lukt het niet zonder meer. Door nog een extra punt toe te voegen kan men echter

$t\,h^r \Delta y$ inderdaad als lineaire combinatie van k_i's schrijven. Hiermede heeft men dan de laatste term van de Taylorreeks die door het R-K proces in rekening wordt gebracht expliciet in handen.

Als voorbeeld geven we een derde orde formule.

$$k_o = h\, f(x_o,\ y_o)$$

$$k_1 = h\, f(x_o + h/3,\ y_o + k_o/3)$$

$$k_2 = h\, f(x_o + 2h/3,\ y_o + 2k_1/3)$$

$$\Delta y = (k_o + 3k_2)/4$$

Als extra punt wordt berekend

$$k_3 = h\, f(x_o + h,\ y_o + \Delta y).$$

Dan is

$$t\, h^3 \Delta y = (k_o - 3k_2 + 2k_3)/2$$

Hier komt het bijzonder goed uit. Het extra punt is namelijk het begin punt voor de volgende integratiestap en dit punt moest dus toch al berekend worden.

Bij de vierde en vijfde orde formules lukt het niet meer om als extra punt het beginpunt van de volgende stap te kiezen.
We geven hier de desbetreffende formules niet. Men kan ze vinden in het proefschrift van J.A. Zonneveld, Automatic Numerical Integration.

We beschouwen in het bijzonder het vijfde orde proces.
Zoals reeds eerder opgemerkt zijn voor een vijfde orde formule 6 punten nodig. Verder is er een extra punt nodig om $t\, h^5\, \Delta y$ te berekenen, dus in totaal 7 punten per integratiestap.

De grootte van $t\, h^5\, \Delta y$ bepaalt of een stap verworpen of geaccepteerd wordt. De strategie voor de verandering van h is weer dezelfde als op pag. 90 e.v., waarbij men voor discr nu $\mathrm{abs}(t\, h^5\, \Delta y)$ moet nemen.
Men vergelijke dit met het op pag.98 beschreven proces. Daar had men 11 punten per stap nodig, terwijl het proces ook van de vijfde orde was.

Experimenten tonen aan dat de 7-punts R-K procedure in de praktijk goed werkt. Toch moet men bij automatische processen altijd oppassen zoals het volgende voorbeeld leert.

Gegeven de vergelijking $\frac{dy}{dx} = -y$.

met $x_o = 0$ en $y(x_o) = 1$.

Exacte oplossing: $y(x) = e^{-x}$.

Integreer de vergelijking van $x = 0$ tot $x = 2$;
vervolgens van $x = 2$ tot $x = 4$ enz.

Voor de reeksontwikkeling van $y^*(x_o + h)$ vindt men hier

$$y^*(x_o + h) = y(x_o)(1 - h + \tfrac{1}{2}h^2 - \tfrac{1}{6}h^3 + \tfrac{1}{24}h^4 - \tfrac{1}{120}h^5 + \tfrac{1}{1440}h^6). \quad (6.6)$$

De vijfde orde term klopt nog met de reeksontwikkeling van e^{-x} om het punt x_o.

Voor $t\,h^5\,\Delta y$ vindt men

$$t\,h^5\,\Delta y = -h^5(2 - h)\,y(x_o)/240.$$

Inderdaad is dus $t\,h^5\,\Delta y$ gelijk aan de vijfde orde term $-\frac{y(x_o)}{120}h^5$ plus iets van hogere orde.

De procedure probeert de integratie in één stap te doen en neemt dus om te beginnen $h = 2$.
Dan blijkt $t\,h^5\,\Delta y$ nul te zijn en de stap wordt geaccepteerd.
Formule (6.6) geeft als resultaat

$$y^*(x_o + h) = y(x_o)/9.$$

Vervolgens probeert de procedure in één stap van $x = 2$ tot $x = 4$ te integreren en dat lukt ook weer.
In plaats van de exacte oplossing $y(x) = e^{-x}$ krijgt men op deze manier $y(x) = 3^{-x}$.

Zij te integreren de vergelijking

$$\frac{dy}{dx} = -\frac{x}{y} \quad \text{met } x_o = 0 \text{ en } y(x_o) = 2.$$

De oplossing is een cirkel met de oorsprong als middelpunt en straal 2.

De op de vorige pagina beschreven R-K procedure is niet geschikt om deze vergelijking numeriek te integreren.
De punten $(2,0)$ en $(-2,0)$ geven moeilijkheden. In de omgeving van deze punten kan men beter de x als afhankelijk en de y als onafhankelijk variabele kiezen, d.w.z. men integreert de vergelijking $\frac{dx}{dy} = -\frac{y}{x}$.

Men kan dit in de procedure inbouwen. Bij iedere stap kijkt men naar $f(x,y)$.
Is $|f(x,y)| \leq 1$ dan wordt x als onafhankelijk variabele gekozen, is $|f(x,y)| > 1$ dan wordt y als onafhankelijk variabele gekozen.

Er treedt dan nog een additionele moeilijkheid op. Stel namelijk dat men wil integreren tot x = b. In de buurt van x = b kan echter y als onafhankelijk variabele gekozen zijn. Men moet dan in feite het nulpunt bepalen van de functie x(y)-b.
Men kan de procedure nog wat algemener maken door de integratie te laten stoppen als voldaan is aan $\varphi(x,y) = 0$, waarbij $\varphi(x,y)$ een willekeurige expressie in x en y is.

Met een op deze wijze verfijnde integratie procedure kan men bijv. de differentiaal vergelijking

$$\frac{dp}{dy} = \frac{c(1-y^2)p + y}{p}$$

van pag. 95 integreren.
Men kan de amplitude bepalen door de punten te registreren waar p = 0 is. Men moet dan eerst een paar keer rond integreren, zodat men op de limiet cyclus terecht komt.

De verwisselingvan onafhankelijk variabele kan ook toegepast worden bij een stelsel vergelijkingen

$$\frac{dy_j}{dx} = f_j(x, y_1, \ldots, y_n) \qquad j = 1, \ldots, n.$$

Uit $x, y_1, \ldots, y_n$ kiest de procedure een onafhankelijk variabele en wel zo dat de absolute waarden van de afgeleiden van de andere variabelen naar de gekozen variabele kleiner dan 1 zijn.
De procedure beëindigt de integratie als voldaan is aan $\varphi(x, y_1, \ldots, y_n) = 0$.

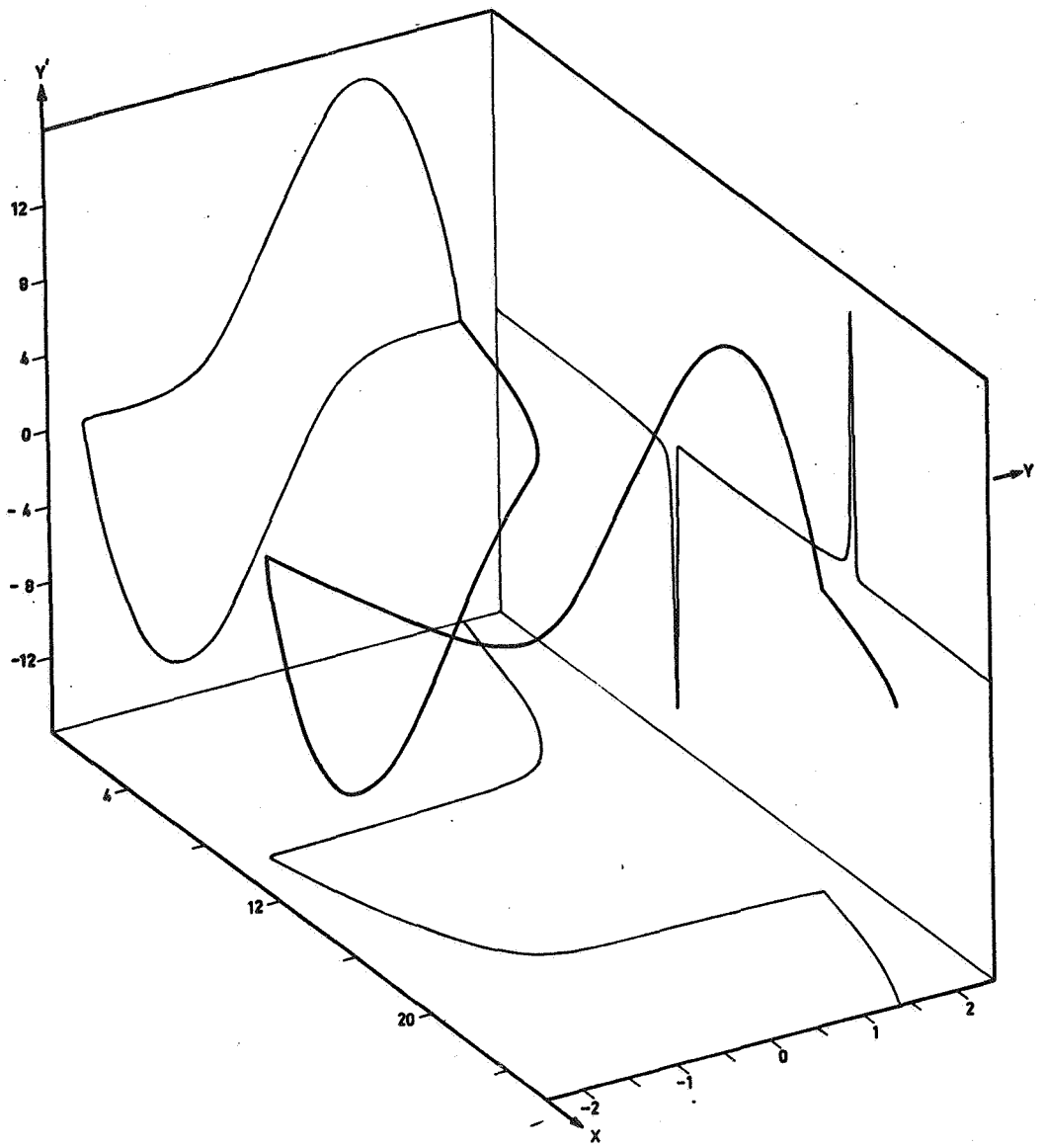
Men kan de procedure gebruiken om de vergelijking van van der Pol

$$y'' - c(1 - y^2)y' + y = 0$$

te integreren, waarbij de vergelijking als een stelsel wordt geschreven.

$$\frac{dy}{dx} = y'$$

$$\frac{dy'}{dx} = c(1 - y^2)y' - y$$



$$y''-10(1-y^2)y'+y=0$$

Uit de figuur (voor c = 10) blijkt al dat de numerieke integratie voor grotere waarden van c een moeilijke opgave is door het bijna singuliere gedrag van de oplossing.
De stapgrootte varieert voor c = 10 met een factor honderd bij het doorlopen van een cyclus.
De toepassing van variabele stapgrootte is hier noodzakelijk.
De amplitude en de periode kan men bepalen door de puntente registreren waar y' = 0 is.

Bij de R-K formules wordt Δy berekend als een lineaire combinatie van k's. Het is echter niet duidelijk waarom men zich tot lineaire combinaties zou beperken. Scraton heeft een 5-punts formule gegeven, waarbij Δy als een rationale functie van k's wordt berekend.

Schematisch zien zijn formules er als volgt uit

$$k_i = h\, f(x_i, y_i) \qquad i = 0, \ldots, 4$$

$$q = \Sigma b_i\, k_i$$

$$s = \Sigma d_i\, k_i$$

$$r = \Sigma d_i\, k_i$$

$$\Delta y = \sum_{o}^{4} a_i\, k_i + \frac{q \cdot r}{s}$$

Door geschikte keuze van de punten (x_i, y_i) en de coefficienten a_i, b_i, c_i en d_i bereikt hij dat de fout in Δy van de orde h^6 is. Tevens is $\frac{q \cdot r}{s}$ juist de vijfde orde term.

Met vijf punten heeft Scraton dus een vijfde orde proces gemaakt. Bij een R-K proces zijn hiervoor ten minste 6 punten nodig.

Een nadeel is dat zijn methode onbruikbaar is voor stelsel differentiaal vergelijkingen, immers bij stelsels worden de k_i's vectoren, dus q, r en s worden ook vectoren.

7. Een repetitief proces.

Beschouw het volgende proces:

$$u_{k+1} = \tfrac{1}{2}(u_k + v_k)$$

$$v_{k+1} = \sqrt{u_{k+1} \cdot v_k}$$

met als beginvoorwaarden

$$u_o = \cot \alpha$$

$$v_o = \operatorname{cosec} \alpha \qquad (0 < \alpha < \pi).$$

Dan is

$$u_1 = \tfrac{1}{2}(\cot \alpha + \operatorname{cosec} \alpha) = \tfrac{1}{2} \frac{\cos \alpha + 1}{\sin \alpha}$$

$$= \tfrac{1}{2} \cdot \frac{2 \cos^2 \alpha/2}{2 \sin \alpha/2 \cos \alpha/2} = \tfrac{1}{2} \cot \frac{\alpha}{2}$$

en

$$v_1 = \sqrt{\tfrac{1}{2} \cot \tfrac{\alpha}{2} \cdot \operatorname{cosec} \alpha} = \sqrt{\tfrac{1}{2} \frac{\cos \alpha/2}{\sin \alpha/2} \cdot \frac{1}{2 \sin \alpha/2 \cos \alpha/2}} = \tfrac{1}{2} \operatorname{cosec} \frac{\alpha}{2}.$$

In het algemeen geldt

$$u_k = 2^{-k} \cot 2^{-k} \alpha$$

$$v_k = 2^{-k} \operatorname{cosec} 2^{-k} \alpha .$$

Hieruit volgt

$$L(\cot \alpha, \operatorname{cosec} \alpha) = \lim_{k \to \infty} u_k = \lim_{k \to \infty} v_k = \frac{1}{\alpha}.$$

Met $L(u_o, v_o)$ bedoelen we hierbij $\lim u_k (= \lim v_k)$ als we het proces starten met de beginvoorwaarden u_o en v_o.

Men kan het proces gebruiken om arctan x te berekenen.
Immers $\arctan x = \operatorname{arccot} \frac{1}{x}$ $\qquad (x > 0)$.
Start het proces met

$$u_o = \cot \alpha = \frac{1}{x} \quad \text{en} \quad v_o = \operatorname{cosec} \alpha = \frac{1}{x} \sqrt{1 + x^2}.$$

Dan is $L\left(\frac{1}{x}, \frac{1}{x}\sqrt{1+x^2}\right) = \frac{1}{\arctan x}$.

Men kan de beginvoorwaarden nog met x vermenigvuldigen, dus

$$u_o = 1 \quad \text{en} \quad v_o = \sqrt{1+x^2} .$$

Men heeft dan $\arctan x = \frac{x}{L(1, \sqrt{1+x^2})}$.

We hebben hiermede een repetitief proces gevonden om α op te lossen uit de vergelijking $\tan \alpha = x$. Evenals bij een iteratief proces herhalen we steeds dezelfde berekening. Nadat echter de beginvoorwaarden 1 en $\sqrt{1+x^2}$ zijn berekend, wordt op de oorspronkelijke vergelijking geen enkel beroep meer gedaan. Dit is een belangrijk verschil met iteratieve processen, waarbij men een gevonden schatting weer in de oorspronkelijke vergelijking substitueert.

Voorbeeld.

Bereken $\arctan 1 = \frac{1}{L(1, \sqrt{2})}$.

$u_o = 1 \qquad v_o = \sqrt{2} = 1.414.$

1	1.414
1.207	1.306
1.257	1.281
1.269	

Het proces convergeert lineair.

Nu is echter $x \cot x = 1 - \frac{1}{3}x^2 + \ldots$,

dus

$$u_k = \frac{x}{\alpha} x_k \cot x_k =$$

$$= \frac{x}{\alpha} + f_1 x_k^2 + f_2 x_k^4 + f_3 x_k^6 + \ldots$$

met $\qquad x_k = 2^{-k} \alpha.$

Vergelijk dit met voorbeeld 2 op pag. 69. Het blijkt dat men op de rij $u_o, u_1, u_2, \ldots$ het extrapolatieproces van Neville-Huygens kan toepassen.

1			
1.207	1.276		
1.257	1.274	1.274	
1.269	1.273	1.273	1.273

Men vindt $\arctan 1 = \pi/4 = \frac{1}{1.273} = 0.785$
$(\pi/4 = 0.7853 \ldots)$.

Het repetitieve proces, gecombineerd met de extrapolatie, is een aantrekkelijk proces om de arctan te berekenen, als de worteltrekking niet veel duurder is dan een vermenigvuldiging of een optelling.
Dit geval doet zich bijvoorbeeld voor bij een computer die geen ingebouwde floating arithmetiek heeft.

Het beschreven proces is beperkt in zijn toepassing. Men kan $L(\cot \alpha, \operatorname{cosec} \alpha)$ analytisch bepalen, maar bij andere keuzen van u_o en v_o kennen we de limiet meestal niet.

We geven nog een toepassing van het proces.
Zij gevraagd log y te berekenen.
We hadden

$$\arctan x = \frac{x}{L(1, \sqrt{1 + x^2})}$$

Substitueer voor x nu i x.
Dit geeft

$$\arctan (ix) = i \operatorname{artanh} x = \frac{ix}{L(1, \sqrt{1 - x^2})} ,$$

dus

$$\operatorname{artanh} x = \frac{x}{L(1, \sqrt{1 - x^2})}$$

Zij $t = \operatorname{artanh} x$, dan is

$$x = \tanh t = \frac{e^t - e^{-t}}{e^t + e^{-t}} = \frac{e^{2t} - 1}{e^{2t} + 1} .$$

Hieruit volgt

$$e^{2t} = \frac{1 + x}{1 - x}$$

en

$$t = \log \sqrt{\frac{1 + x}{1 - x}} = \frac{x}{L(1, \sqrt{1 - x^2})} .$$

Stelt men tenslotte $y = \frac{1 + x}{1 - x}$ dan vindt men

$$\log y = 2 \frac{\frac{y - 1}{y + 1}}{L(1, \frac{2\sqrt{y}}{y + 1})}$$

of $\log y = \dfrac{2(y-1)}{L(1+y,\ 2\sqrt{y})}$.

Voorbeeld

$$\log 2 = \frac{2}{L(3,\ 2\sqrt{2})}$$

3	2.828
2.914	2.871
2.892	

Extrapoleren:

3		
2.914	2.885	
2.892	2.885	2.885

$\log 2 = \dfrac{2}{2.885} = 0.6932$ (log 2 = 0.69314....)

8. Over de definitie van Algol.

In paragraaf 1 van hoofdstuk 1 hebben wij gezien hoe men bij de definitie van Algol gebruik maakt van een metataal.
In deze metataal spelen metalinguistische variabelen en metalinguistische formules een rol.

Met behulp van de metalinguistische formules kan men de syntax van Algol (en van andere formele talen) exact beschrijven. In het Algol rapport is de syntax op deze manier volledig geformaliseerd.
De betekenis van een stuk Algol tekst wordt echter in het Engels aangegeven. Zoals de ervaring leert geeft dat aanleiding tot veel moeilijkheden, omdat men de Engelse tekst vaak op verschillende manieren kan interpreteren.
Men kan deze moeilijkheden vermijden door ook de semantiek te formaliseren.
In dit hoofdstuk zullen we nagaan op welke wijze dit gerealiseerd kan worden. Men houde echter in het oog dat hetgeen nu volgt niet als definitief of zaligmakend beschouwd moet worden.

Een metalinguistische variabele zullen we op dezelfde wijze noteren als in het Algol rapport. Voor de metalinguistische formules voeren we een andere notatie in.
In plaats van

< digit > :: = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

< letter > :: = a | b | c | d

< id > :: = < letter > | < id > < letter > | < id > < digit >

schrijven wij

0 in < digit > ,
1 in < digit > ,
2 in < digit > ,
9 in < digit > ,
a in < letter >,
b in < letter >,
< id > < letter > in < id >,
< letter > in < id >,

De symbolen in en, behoren tot de metataal.

We maken nu een lijst V van "waarheden".
Het bovenstaande rijtje metalinguistische formules zal bijvoorbeeld in de lijst V voorkomen.
Als we iets willen weten zullen we V raadplegen. We komen altijd met een zeer bepaalde vraag bij V, waarbij we V steeds van achter af aan zullen raadplegen. Men kan bijvoorbeeld de vraag stellen

a in < id > ?

We zijn tevreden als we een meer algemene waarheid in de lijst vinden.
Zo is < letter > in < id > een meer algemene waarheid dan a in < id > .
We noemen < letter > in < id > een enveloppe van a in < id >.

We komen nu met onze vraag bij de lijst. Als eerste waarheid vinden we < letter > in < id >. We kijken of a in < id > er in past.
Het antwoord op de vraag is ja als

< letter > een enveloppe is van a
in een enveloppe is van in
< id > een enveloppe is van < id >.

Aan de twee laatste voorwaarden is voldaan.
Blijft over de vraag is a in < letter >?
We beginnen weer onderaan de lijst en komen wederom terecht bij < letter > in < id >. Dit leidt opnieuw tot de vraag is a in < letter >?
Conclusie: bij het opstellen van de lijst moet men oppassen dat men bij de beantwoording van een vraag niet in een kringetje blijft ronddraaien.
Men moet de meer speciale waarheden onderaan in de lijst opnemen.
Het desbetreffende stukje van V moet er als volgt uitzien

< id > < letter > in < id >
< letter > in < id >
a in < letter >
b in < letter >

Komt men nu met de vraag a in < id >? bij V, dan is het antwoord ja.

Op deze wijze kan men de syntax vastleggen. Er is nu echter ook aangegeven hoe men de metalinguistische formules moet raadplegen. Dit laatste ontbreekt bij de Backus notatie van het Algol rapport.

We willen echter niet alleen de syntax, maar ook de semantiek formaliseren.

We gaan van het standpunt uit dat een Algol programma als invoergegevens een rijtje symbolen krijgt, vervolgens wat rekent en daarna een ander rijtje symbolen aflevert.
De semantiek is nu de beschrijving van wat er gebeurt, dus hoe een rijtje symbolen uit een ander rijtje symbolen wordt gevormd.
Op deze wijze opgevat heeft het begrip semantiek niets mystieks.

We zullen nu in het kort beschrijven welke middelen nog in de metataal nodig zijn om het doel te bereiken.
De betekenis van een en ander zal duidelijker worden bij de bestudering van het voorbeeld op pag. 111.

In de eerste plaats willen we de mogelijkheid scheppen om naar een reeds eerder gebruikte identifier te verwijzen. Dit is bij Backus niet mogelijk.

< identifier 1 > , < identifier 2 > enz. zullen metalinguistische variabelen zijn met dezelfde waarden als < identifier > .

Als in een uitdrukking (een waarheid in V) bijvoorbeeld < id 1 > meer dan een keer voorkomt, dan mag men de eerste keer de waarde van < id 1 > volledig vrij kiezen, daarna moet men echter voor < id 1 > steeds de gekozen waarde substitueren.

In de metataal zullen we accoladen { en } gebruiken om aan te geven dat iets bij elkaar hoort. Tussen de accoladen staat een meta primary.

We zullen soms vragen naar de waarde van een stukje tekst (een uitdrukking). Het uitrekenen van de waarde noemen we evalueren.
In de lijst V komen waarheden voor van het type

a = 7,

x = y,

De bedoeling is de volgende: als we geïnteresseerd zijn in de waarde van

een uitdrukking raadplegen we V.
Stel dat we de waarde van x willen weten. We komen terecht bij x = y. We vinden dat de waarde van x gelijk is aan de waarde van y.
Hiermede is de oorspronkelijke vraag vervangen door de vraag wat is de waarde van y.

Algemeen: de uitdrukking die we willen evalueren kan een speciaal geval zijn van hetgeen in een waarheid links van het = teken staat. De substituties die we links moeten maken worden ook rechts van het = teken gemaakt en het oorspronkelijke probleem wordt vervangen door de evaluering van wat er nu rechts is gekomen.

Als men de waarde van a wil bepalen vindt men in de lijst a = 7. Dit betekent dat we nu de waarde van 7 moeten bepalen.
Daartoe worden de metasymbolen ' ' quote en unquote ingevoerd.
Afspraak: de waarde van '....' is hetgeen er tussen het quote en unquote symbool staat.

Men kan dus 7 = '7' lezen als de waarde van 7 is 7.

In V zal men bijv. als waarheid opnemen

< digit 1 > = '< digit 1 >'

of < number 1 > = '< number 1 >'

Zoals reeds eerder gezegd moet de substitutie die we links maken ook rechts worden gemaakt.

In V komen waarheden voor van het volgende type:

< id 1 > + < id 2 > = va < id 1 > + va < id 2 > ,

Stel we willen de waarde van x + y bepalen. Nu is x een speciaal geval van < id 1 > en y is een speciaal geval van < id 2 >. De waarheid is dus toepasbaar. We maken rechts dezelfde substitutie als links en we moeten vervolgens evalueren va x + va y.

va is ook een metasymbool. Zodra men vraagt of va x een bijzonder geval van iets is, heeft dit tot gevolg dat eerst x wordt geëvalueerd. Dit gebeurt dus al direct bij de eerste waarheid van V die men raadpleegt.
Stel dat x de waarde 7 heeft. Dan heeft men vervolgens te evalueren 7 + va y. Vroeg of laat zal men nagaan of va y ergens in past. Dit is het sein om dan eerst y uit te rekenen. Men vindt nu bijv. 7 + 6. Om de waarde hiervan te vinden gaat men V weer raadplegen. (zie het voorbeeld op pag. 111)

Tenslotte bespreken we nog het metasymbool ——> (de implicatie uit de logica).

In V kan iets staan van het type

x ——> y = z,

Bedoeling: we kijken of de uitdrukking die we willen evalueren in y past, daarbij weer links en rechts van het = teken de nodige substituties makend. Als de uitdrukking past maken we dezelfde substituties in de conditie x. Vervolgens raadplegen we V om te zien of er een waarheid is die een bijzonder geval is van de waarheid x. Als dit zo is, dan wordt de evaluatie van y vervangen door de evaluatie van z.

Voorbeeld

Als voorbeeld geven we een onderdeel van de formalisering van Algol, en wel de optelling van twee integers.
Gebruikte afkortingen:

di	digit
ui	unsigned integers
ze	zero

De metavariabele < pm > kan de waarden + of - aannemen.

- < ui 1 > + < ui 2 > = < ui 2 > - < ui 1 > ,
- < ui 1 > - < ui 2 > = - va {< ui 1 > + < ui 2 > },
  < ui 1 > < di 1 > < pm 1 > < ui 2 > < di 2 > =
  va {< ui 1 > < pm 1 > < ui 2 >} 0 +
  va {< di 1 > < pm 1 > < di 2 >},
  < ui 1 > < di 1 > < pm 1 > < di 2 > =
  < ui 1 > 0 + va {< di 1 > < pm 1 > < di 2 >},
  < di 1 > < pm 1 > < ui 2 > < di 2 > =
  < pm 1 > < ui 2 > 0 + va {< di 1 > < pm 1 > < di 2 >},
  < ui 1 > 0 + < di 2 > = < ui 1 > < di 2 > ,
  < di 1 > + < ui 2 > 0 = < ui 2 > < di 1 > ,
  < ui 1 > 0 - < di 2 > = va {< ui 1 > - 1 } 0 +
  va {10 - < di 2 >},
  10 - < di 2 > = 9 - va {< di 2 > - 1 } ,

< di 1 > < pm 1 > < di 2 > =

va{< di 1 > < pm 1 > 1 }< pm 1 >

va{< di 2 > - 1 } ,

< ui 1 > < pm > < ze > = < ui 1 > ,
< ze > + < ui 1 > = < ui 1 > ,
< ze > - < ui 1 > = - < ui 1 > ,
< ze > < pm > < ze > = 0,
0 + 1 = 1, 1 + 1 = 2, 2 + 1 = 3,
3 + 1 = 4, 4 + 1 = 5, 5 + 1 = 6,
6 + 1 = 7, 7 + 1 = 8, 8 + 1 = 9,
9 + 1 = 10,

{< di 1 > + 1 = < di 2 >}——>{< di 2 > - 1 = < di 1 >},

- < ui 1 > = '- < ui 1 >' ,

+ < ui 1 > = '< ui 1 >' ,

< ui 1 > = '< ui 1 >' ,
< ui > < di > in < ui > ,
< di > in < ui > ,
0 in < di > , 1 in < di > , 2 in < di > , 3 in < di > ,
4 in < di > , 5 in < di > , 6 in < di > , 7 in < di > ,
8 in < di > , 9 in < di > ,
< ze > 0 in < ze > ,
0 in < ze > ,
< ze > = '0' ,
< pm > < ze > = '0' ,
< ui 1 > + - < ui 2 > = < ui 1 > - < ui 2 > ,

9. Recursie vergelijkingen

Een recursie (of differentie) vergelijking is een vergelijking van het type

$$f_{k+r} = F(f_{k+r-1}, f_{k+r-2}, \ldots, f_k) \qquad (r \text{ vast}).$$

De oplossing is numeriek eenvoudig te berekenen als de r beginwaarden $f_0, \ldots, f_{r-1}$ bekend zijn.

Voorbeelden:

$$f_{k+1} = p(k) f_k \tag{9.0}$$

$$f_{k+2} = g(k) f_{k+1} + h(k) f_k. \tag{9.1}$$

Men kan deze vergelijkingen ook met differenties schrijven:

$$f_k + \Delta f_k = p(k) f_k$$

$$f_k + \Delta f_k + \Delta^2 f_k = g(k)(f_k + \Delta f_k) + h(k) f_k.$$

De vergelijkingen (9.0) en (9.1) zijn lineaire differentie vergelijkingen; de functie f en de differenties van f komen slechts lineair voor. Vergelijking (9.1) is de algemene homogene lineaire differentie vergelijking van de tweede orde.

De theorie van de lineaire differentie vergelijkingen lijkt op de theorie van de lineaire differentiaal vergelijkingen.
Men heeft bijvoorbeeld de volgende stelling:
Zijn $F_1(k)$ en $F_2(k)$ oplossingen van (9.1) dan is ook $C_1F_1(k) + C_2F_2(k)$ een oplossing. Zijn bovendien $F_1(k)$ en $F_2(k)$ lineair onafhankelijk dan is $C_1F_1(k) + C_2F_2(k)$ de algemene oplossing van (9.1).

Beschouw nu de inhomogene lineaire differentie vergelijking

$$f_{k+2} = g(k) f_{k+1} + h(k) f_k + e(k).$$

Evenals bij de differentiaalvergelijkingen is het voldoende een particuliere oplossing van de inhomogene vergelijking te bepalen, als men de algemene oplossing van de homogene vergelijking kent.
Een oplossing van de inhomogene vergelijking vindt men dan bijvoorbeeld met de methode van de variatie der constanten.

De fundamentele oplossingen van een homogene lineaire differentie vergelijking met constante coefficienten zijn eenvoudig te vinden.

Zij gegeven de vergelijking

$$f_{k+2} = g\, f_{k+1} + h\, f_k \quad (g \text{ en } h \text{ constant}).$$

Stel $f_k = a^k$.
Substitutie in de vergelijking geeft

$$a^{k+2} = g\, a^{k+1} + h\, a^k$$

of $$a^2 = ga + h.$$

Het karakteristieke polynoom $a^2 - ga - h$ heeft twee wortels a_1 en a_2.
Is $a_1 \neq a_2$ dan zijn de fundamentele oplossingen

$$F_1(k) = a_1^k \quad \text{en} \quad F_2(k) = a_2^k.$$

Voorbeeld

Beschouw de recursie vergelijking

$$f_{k+1} = 2 \cos x\, f_k - f_{k-1}. \quad (x \text{ vast})$$

Nu is $\sin (k+1)\, x = \sin kx \cos x + \cos kx \sin x$
en $\sin (k-1)\, x = \sin kx \cos x - \cos kx \sin x.$

Hieruit volgt

$$\sin (k+1)\, x = 2 \cos x \sin kx - \sin (k-1)\, x,$$

dus sin kx is een oplossing van de vergelijking.
Op dezelfde manier blijkt dat cos kx een oplossing is.
Voor $x \neq n\pi$ zijn sin kx en cos kx lineair onafhankelijk; de algemene oplossing is dan

$$f_k = C_1 \sin kx + C_2 \cos kx.$$

Men kan deze oplossing ook vinden met behulp van de substitutie $f_k = a^k$.

De startwaarden $f_0 = 0$ en $f_1 = \sin x$ geven de oplossing $f_k = \sin kx$.
Zijn sin x en cos x bekend dan zijn sin 2x, sin 3x enz. in successie uit te rekenen. Per stap kost dit een vermenigvuldiging. Analoog voor cos kx.

Men kan dit gebruiken voor de berekening van de sommen

$$\sum_{k=1}^{n} a_k \sin kx \qquad \text{en} \qquad \sum_{k=0}^{n} a_k \cos kx.$$

Term voor term uitrekenen van de twee sommen kost per stap 4 vermenigvuldigingen. Het kan echter voordeliger.

Met de startwaarden $f_o = 0$ en $f_1 = a_n$ vindt men als oplossing

$$f_k = a_n \frac{\sin kx}{\sin x} .$$

Beschouw nu de inhomogene recursievergelijking

$$f_{k+1} = 2 \cos x \, f_k - f_{k-1} + a_{n-k}$$

met startwaarden $f_o = 0$ en $f_1 = a_n$.

Dan is

$$f_2 = a_n \frac{\sin 2x}{\sin x} + a_{n-1}$$

$$f_3 = a_n \frac{\sin 3x}{\sin x} + a_{n-1} \frac{\sin 2x}{\sin x} + a_{n-2}$$

$$f_4 = a_n \frac{\sin 4x}{\sin x} + a_{n-1} \frac{\sin 3x}{\sin x} + a_{n-2} \frac{\sin 2x}{\sin x} + a_{n-3}$$

$$\vdots$$

$$f_n = a_n \frac{\sin nx}{\sin x} + a_{n-1} \frac{\sin (n-1)x}{\sin x} + a_{n-2} \frac{\sin (n-2)x}{\sin x} + \ldots + a_1 \frac{\sin x}{\sin x} .$$

Dus $\sin \sum_{k=1}^{n} a_k \sin k x = \sin x \cdot f_n$.

Door per stap a_{n-k} op te tellen bouwt men dus en passant de gehele som $\Sigma a_k \sin kx$ op.

Men kan de recursie echter nog een stap verder voortzetten. Dit geeft

$$f_{n+1} = a_n \frac{\sin (n+1)x}{\sin x} + a_{n-1} \frac{\sin nx}{\sin x} + \ldots + a_1 \frac{\sin 2x}{\sin x} + a_o$$

$$= a_n \frac{\sin nx \cos x + \cos nx \sin x}{\sin x} + a_{n-1} \frac{\sin(n-1)x \cos x + \cos(n-1)x \sin x}{\sin x} +$$

$$\ldots + a_1 \frac{\sin x \cos x + \cos x \sin x}{\sin x} + a_o$$

of

$$f_{n+1} = \cos x . f_n + \sum_{k=o}^{n} a_k \cos kx$$

of

$$\sum_{k=o}^{n} a_k \cos kx = f_{n+1} - \cos x . f_n .$$

De cosinus som heeft men dus door een stap extra erbij gekregen.
Vergeleken met het term voor term uitrekenen van de twee sommen is het aantal vermenigvuldigingen viermaal zo klein geworden.
De recursie vergelijking is hier een machtig hulpmiddel.
In de kristallografie spelen de sinus en cosinus sommen een belangrijke rol.

Hoe zit het met de nauwkeurigheid van dit proces?
Ook als tijdens de berekening geen afrondingsfouten worden gemaakt is het antwoord niet exact; immers cos x kan slechts in een eindig aantal decimalen worden gegeven.
Men kan zeggen: in plaats van de cosinus van x geven we de exacte waarde van de cosinus van x^*, waarbij $x^* \approx x$ is.
Zonder afrondingsfouten zou men dus exact uitrekenen

$$\Sigma a_k \sin k x^* \text{ en } \Sigma a_k \cos k x^*.$$

Voor grote waarden van k kan $k x^*$ vrij veel van $k x$ afwijken. Bovendien merken we op dat $|x - x^*|$ groter is dan $|\cos x - \cos x^*|$, vooral voor x dicht bij nul.
Conclusie: cos x moet zo nauwkeurig mogelijk worden gegeven.

We beschouwen nu het effect van afrondingsfouten tijdens de berekening.
In plaats van f_2 vindt men tengevolge hiervan $f_2 + \varepsilon_2$.

$$f_2 + \varepsilon_2 = a_n \frac{\sin 2x}{\sin x} + a_{n-1} + \varepsilon_2$$

De volgende stap van het proces geeft

$$a_n \frac{\sin 3x}{\sin x} + (a_{n-1} + \varepsilon_2) \frac{\sin 2x}{\sin x} + a_{n-2} + \varepsilon_3.$$

De storing in f_k tengevolge van ε_2 is $\varepsilon_2 \frac{\sin(k-1)x}{\sin x}$.
Deze storingheeft weinig invloed op het resultaat. Men kan zeggen dat i.p.v. $\Sigma a_k \sin k x$ de som $\Sigma a_k^* \sin k x$ exact wordt uitgerekend, waarbij de a_k^*'s licht gestoorde a_k's zijn.

Er kunnen zich bij het oplossen van recursie vergelijkingen ook gevallen voordoen waarbij het effect van afrondingsfouten desastreus is.
Het is mogelijk dat een snel groeiende ongewenste fundamentele oplossing de berekening verstoord.

Ter toelichting beschouwen we eerst een analoge situatie bij differentiaal-vergelijkingen.

$$\frac{d^2y}{dx^2} = y \quad \text{met } y(0) = 1 \quad \text{en} \quad y'(0) = -1.$$

Oplossing: $y = e^{-x}$

Algemene oplossing: $c_1 e^{x} + c_2 e^{-x}$.

De differentiaalvergelijking is onbruikbaar om e^{-x} voor positieve waarden van x te berekenen.
De gezochte oplossing e^{-x} wordt altijd overwoekerd door e^{x}.
Men kan e^{-x} wel uitrekenen als men van het probleem een randwaarde probleem maakt. Men kan bijv. numeriek oplossen

$$\frac{d^2y}{dx^2} = y \quad \text{met} \quad y(0) = 1 \quad \text{en } y(10) = 0.$$

Men vindt zo althans redelijke waarden voor e^{-x} op het interval $(0,10)$.

Een andere mogelijkheid is om de integratie in de andere richting uit te voeren. Dan veroorzaakt e^{x} geen moeilijkheden.

Voorbeeld

De differentiaalvergelijking van Bessel

$$x^2 \frac{d^2y}{dx^2} + x \frac{dy}{dx} + (x^2 - n^2)y = 0$$

heeft als algemene oplossing

$$y = A\, J_n(x) + B\, Y_n(x).$$

$J_n(x)$ is de Bessel functie van de orde n.
$Y_n(x)$ is de Weber functie (of Bessel functie van de tweede soort) van de orde n.
Zie de syllabus partiële diff. verg. pagina 167 e.v.

$J_n(x)$ en $Y_n(x)$ voldoen aan de recursie vergelijking

$$y_{n+1} + y_{n-1} = \frac{2n}{x} y_n \qquad (9.2)$$

Als men $J_o(x)$ en $J_1(x)$ kent, dan zijn $J_2(x)$, $J_3(x)$ etc. uit (9.2) te berekenen.

Dit gaat goed zolang $n < x$ is. Voor grote waarden van n gaan tengevolge van de factor $\frac{2n}{x}$ de afrondingsfouten de zaak bederven.
Dit klopt ook met het gedrag van de fundamentele oplossingen $J_n(x)$ en $Y_n(x)$ van (9.2).
Neem x vast. Als n groot wordt gaat $J_n(x)$ naar nul. Voor kleine n is $Y_n(x)$ klein, bij toenemende n wordt $Y_n(x)$ in absolute waarde zeer groot.

De remedie is de recursie in de andere richting uit te voeren. De ongewenste oplossing $Y_n(x)$ sterft dan uit.

Het nu volgende proces is van J.C.P. Miller.
Zij J_n (1.55) te berekenen voor n = 0, 1, 2, ...
We starten het proces met $y_9 = 0$ en $y_8 = 1$.

n	y_n	J_n (1.55)
9	0	.00000
8	1	.00000
7	10	.00003
6	89	.00028
5	679	.00211
4	4292	.01331
3	21473	.06661
2	78829	.24453
1	181957	.56442
0	155954	.48376

Bij de berekening zijn de y_n's afgerond op het dichtsbijzijnde gehele getal.

De algemene oplossing van (9.2) is $A J_n + B Y_n$, dus

$$y_9 = 0 = A J_9 + B Y_9$$
$$y_8 = 1 = A J_8 + B Y_8 .$$

J_9 (1.55) en J_8 (1.55) zijn klein, Y_9 (1.55) en Y_8 (1.55) groot, dus A is groot t.o.v. B.
Bij de recursie sterft de storing $B Y_n$ uit, dus y_n wordt praktisch $A J_n$ (1.55).
De factor A vindt men door gebruik te maken van de relatie

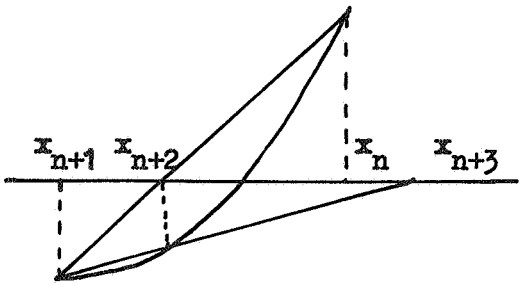
$$J_0(x) + 2 J_2(x) + 2 J_4(x) + \ldots = 1.$$

Nu is $y_0 + 2\,y_2 + \ldots 2\,y_8 = 322376$, dus $A = \frac{1}{322376}$.

Moeilijkheid bij dit proces: met welke n moet men beginnen om Y_n te laten uitsterven. In het voorbeeld is n = 9 gekozen. Men kan nu ook nog eens met n = 10 beginnen en de verhouding van twee opeenvolgende y_n's beschouwen. Is deze verhouding in beide gevallen gelijk, dan is de beginstoring gedempt.

10. Nulpunten van functies

In dit hoofdstuk beschouwen we de functie $y = f(x)$ als een procedure, die bij een gegeven x een getal y aflevert. Dit betekent dat men niet kan differentieren (numerieke differentiatie komt niet in aanmerking). Voor het bepalen van nulpunten is dus bijvoorbeeld de methode van Newton niet bruikbaar. Men kan wel gebruik maken van de Regula Falsi.



$$x_{n+2} = \frac{x_n y_{n+1} - x_{n+1} y_n}{y_{n+1} - y_n}$$

We willen een proces maken, gebaseerd op de Regula Falsi, dat zowel veilig als efficient is.
Uit x_n en x_{n+1} wordt x_{n+2} berekend. Bij de volgende stap kan men x_{n+2} combineren met x_{n+1} of x_n. Nu is in de figuur sign $y_{n+2} \neq$ sign y_n, dus in het interval (x_{n+2}, x_n) ligt zeker een nulpunt. Het is dus veilig om x_{n+2} met x_n te combineren. Als men op deze wijze doorgaat is het proces echter niet efficient, het convergeert slechts lineair.
Een andere methode is om steeds de twee laatst gevonden schattingen x_{n+1} en x_{n+2} te combineren. Dit werkt dicht bij de wortel goed. De convergentie van het proces is dan van de orde 1.6, dus tussen lineair en kwadratisch in. Het is echter niet geheel veilig. In de figuur valt de op deze wijze verkregen schatting x_{n+3} buiten het veilige interval (x_{n+2}, x_n). Deze schatting mag niet geaccepteerd worden.

Om een efficient en veilig proces te krijgen moet men een combinatie van beide methoden nemen.

We laten eerst nog even zien dat de convergentie van de tweede methode inderdaad van de orde 1.6 is. Ter vereenvoudiging van het schrijfwerk nemen we aan dat het nulpunt in de oorsprong ligt, dus $f(0) = 0$.
Dan is $y = C_1 x + C_2 x^2 + C_3 x^3 + \ldots\ldots$.
Stel nu dat er constanten α en β zijn, zodanig dat

$$x_{n+1} = \alpha\, x_n^{\beta}$$

voor elke waarde van n.
Dus ook

$$x_{n+2} = \alpha\, x_{n+1}^{\beta} .$$

Substitutie in de Regula Falsi geeft

$$x_{n+2} = \alpha^{1+\beta}\, x_n^{\beta^2} = \frac{x_n(C_1 x_{n+1} + C_2 x_{n+1}^2 + \ldots) - x_{n+1}(C_1 x_n + C_2 x_n^2 + \ldots)}{C_1(x_{n+1} - x_n) + \ldots}$$

$$= x_n x_{n+1} \frac{C_2}{C_1} + \ldots \quad .$$

Dus met verwaarlozing van hogere orde termen

$$\alpha^{1+\beta} x_n^{\beta^2} = \frac{C_2}{C_1}\, \alpha\, x_n^{\beta+1}$$

of

$$x_n^{\beta^2-\beta-1} = \alpha^{-\beta}\, \frac{C_2}{C_1} .$$

Men kan α en β nu bepalen door op te merken dat het linkerlid onafhankelijk van n moet zijn. Hiervoor is nodig $\beta^2 - \beta - 1 = 0$.
Dus $\beta = \frac{1}{2}(1 + \sqrt{5}) \approx 1.6$.

Opmerking

Regula Falsi op deze wijze 3 maal toepassen (berekening van 3 functiewaarden)geeft

$$x_{n+3} \approx x_n^{1.6^3} \approx x_n^4 .$$

Newton 2 maal toepassen (berekening van 2 functiewaarden en 2 afgeleiden) geeft $x_{n+2} \approx x_n^4$.

Als men van een functie f(x) een nulpunt wil berekenen moet men er rekening mee houden dat de variabele x slechts een rij discrete waarden kan aannemen. Men mag niet verwachten dat f(x) voor een van deze waarden exact nul wordt. We moeten daarom definiëren wat we voor numerieke doeleinden onder een nulpunt zullen verstaan.

Zij gegeven een tolerantie e.
We noemen x een nulpunt van de functie f(x) als er twee waarden x_1 en x_2 zijn aangewezen, zodanig dat

$$x - e \leqslant x_1 \leqslant x_2 \leqslant x + e$$

en

$$\text{sign}(f(x_1)) \times \text{sign}(f(x_2)) \neq 1.$$

We geven nu een Algol procedure ZERO (x,a,b,fx,e) voor de bepaling van een nulpunt van fx tussen a en b.
De expressie fx moet van x afhangen en verschillend teken hebben voor x = a en x = b.

```
real procedure ZERO (x,a,b,fx,e); value a,b; real x,a,b,fx; array e;
begin   real c,fa,fb,fc,m,i,tol,re,ae;
        re:= e[1]; ae:= e[2];
        x:= a; fa:= fx; x:= b; fb:= fx; goto entry;
goon:   if abs (i - b) < tol then i:= b + sign (c - b) × tol;
        x:= if sign (i - m) = sign (b - i) then i else m;
        a:= b; fa:= fb; b:= x; fb:= fx;
        if sign (fc) = sign (fb) then
entry:  begin c:= a; fc:= fa end;
        if abs (fb) > abs (fc) then
        begin a:= b; fa:= fb; b:= c; fb := fc; c := a; fc := fa end
        m:= (b + c) / 2;
        i:= if fb - fa ≠ 0 then (a × fb - b × fa) / (fb - fa) else m;
        tol:= abs (b × re) + ae;
        if abs (m - b) > tol then goto goon;
        ZERO:= x:= b
end     ZERO;
```

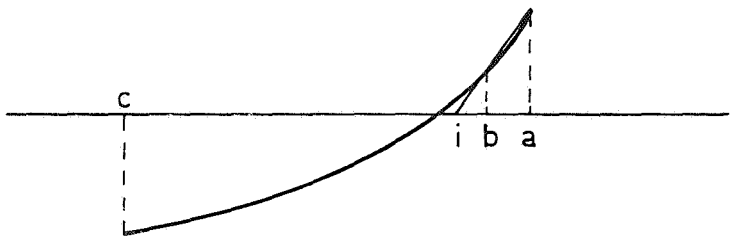
Toelichting.

In het array e wordt de tolerantie gegeven; e[1] is de relatieve tolerantie en e[2] is de absolute tolerantie.
Voor de tolerantie e uit de definitie van een nulpunt wordt in deze procedure genomen e = 2 × (e[2] + abs(b × e[1])); de waarde van b is bij beëindiging van de procedure het gezochte nulpunt.

Het principe van de procedure is als volgt. De Regula Falsi (RF) wordt op de efficiënte manier toegepast, d.w.z. op de laatste schatting (b) en de voorlaatste schatting (a) van het nulpunt. Er wordt echter ook voor gezorgd dat steeds een veilig interval (b,c) aanwezig is, d.w.z.
sign (f(b)) × sign (f(c)) ≠ 1. Bovendien zal steeds $|f(b)| \leqslant |f(c)|$ zijn.
Men mag dus verwachten dat het nulpunt dichter bij b dan bij c ligt. Het resultaat van de RF wordt daarom slechts geaccepteerd als het in de b-helft van (b,c) ligt d.w.z. tussen het midden m = (b+c) / 2 en b. Zo niet, dan wordt als nieuwe schatting m genomen.

De procedure construeert nu een rij inkrimpende veilige intervallen (b, c). Het proces eindigt als (b, c) een lengte $\leq 2 \times (\text{abs}(b \times e[1]) + e[2])$ heeft.

De procedure begint met de berekening van f(a) en f(b) en neemt c = a. Is $|f(b)| > |f(c)|$ dan worden c en b verwisseld (tevens wordt a = c). RF levert een schatting i. Na de label goon wordt i eventueel gewijzigd (zie onder) en vervolgens wordt bekeken of i acceptabel is. Via x wordt de nieuwe schatting b nu gelijk aan i of m, terwijl a de oude waarde van b krijgt.
Als er tekenomslag plaats heeft, d.w.z. als de nieuwe b een ander teken heeft dan de oude b, kan een veilig interval worden aangegeven door c gelijk te kiezen aan a (= oude b).
Als geen tekenomslag plaats heeft verandert c niet.



In de figuur is een situatie getekend waarbij geen tekenomslag meer plaats heeft; b blijft steeds rechts van het nulpunt. De stapjes worden steeds kleiner, maar het veilige interval blijft groot.
We kunnen in dit geval proberen de tekenomslag te forceren door i iets in de richting van c te verschuiven.
Als $|i - b| < \text{tol}$ is, wordt daarom i = b + sign (c - b) × tol genomen (zie bij de label goon).

11. Chebyshev approximatie

In de cursus Wet. Rekenaar A zijn de Chebyshev polynomen en enkele eenvoudige toepassingen van deze polynomen behandeld. Dit hoofdstuk is bedoeld als een aanvulling

Zij K de klasse van de polynomen $p_n(x)$ van de graad $\leq n$ en f(x) een op het interval [a, b] continue functie.
Gevraagd wordt de functie op dit interval te approximeren door een polynoom $p_n(x)$ uit K, zodanig dat $\max_{x \in [a,b]} |f(x) - p_n(x)|$ zo klein mogelijk is.

Stelling (zonder bewijs)

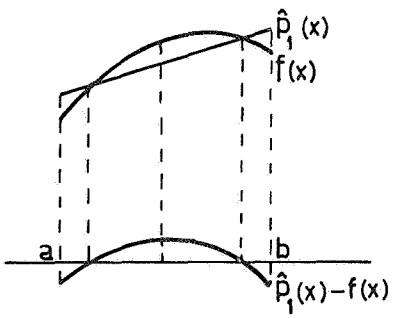
Er is precies één polynoom $\hat{p}_n(x)$ uit K waarvoor $\max_{x \in [a,b]} |f(x) - \hat{p}_n(x)|$ mini-

maal is, d.w.z. voor elk ander polynoom $p_n(x)$ uit K geldt

$$\max_{x \in [a,b]} |f(x) - \hat{p}_n(x)| < \max_{x \in [a,b]} |f(x) - p_n(x)|.$$

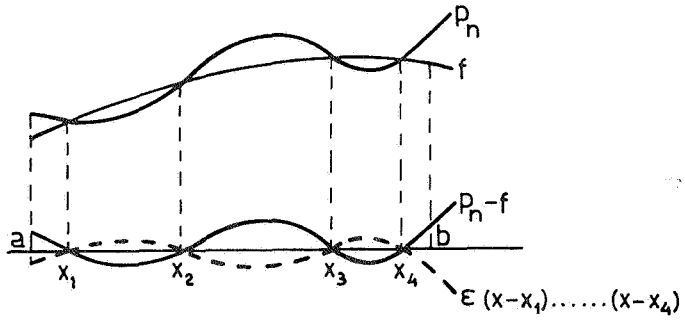
Men noemt $\hat{p}_n(x)$ de beste benadering van $f(x)$ in de zin van Chebyshev.
Het is meestal moeilijk om bij een gegeven functie $f(x)$ het optimale polynoom $\hat{p}_n(x)$ te vinden. Dit gelukt slechts in enkele speciale gevallen.

We beschouwen eerst de approximatie van een functie door een polynoom van de graad ≤ 1. In dit geval kan men het optimale polynoom vaak eenvoudig construeren (zie figuur).



We merken op dat de verschilfunctie $\hat{p}_1(x) - f(x)$ drie extrema heeft, in absolute waarde even groot en alternerend van teken.

We laten nu zien dat in het algemene geval de verschilfunctie $\hat{p}_n(x) - f(x)$ een analoge eigenschap heeft.



Stel dat van het polynoom $p_n(x)$ uit K de verschilfunctie $p_n(x) - f(x)$ ten hoogste n nulpunten heeft, waarvan k nulpunten $x_1 \ldots x_k$ van oneven multipliciteit. In dit geval kan men een beter polynoom aangeven.
Het polynoom $\varepsilon(x - x_1) \ldots (x - x_k)$ is van ten hoogste de graad n.
Door een goede keuze van ε kan men bereiken dat

1) $\varepsilon(x - x_1) \ldots (x - x_k)$ en $p_n - f$ verschillend teken hebben voor x in $[a, b]$

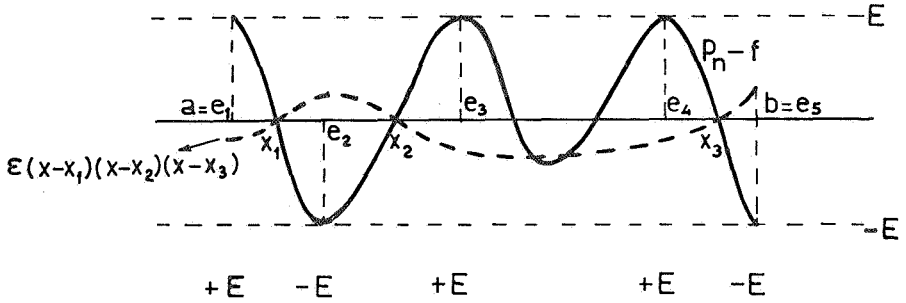
en 2) $|\varepsilon(x - x_1) \ldots (x - x_k)| < |p_n - f|$ voor x in $[a, b]$.

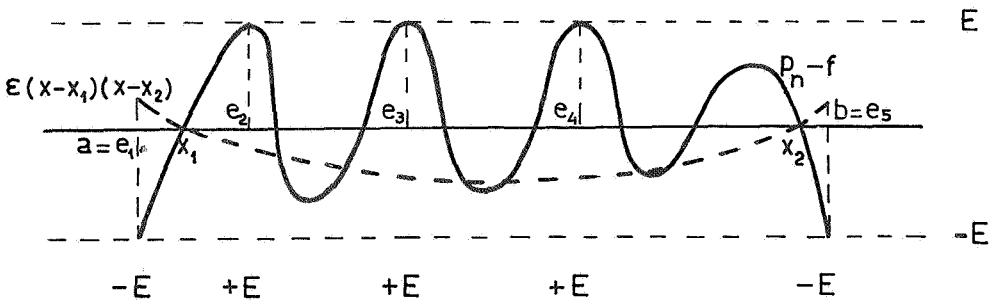
Het polynoom $p_n + \varepsilon(x - x_1) \ldots (x - x_k)$ is dan een betere benadering voor $f(x)$ dan p_n.

Voor het beste polynoom $\hat{p}_n(x)$ moet dus gelden dat de verschilfunctie meer dan n nulpunten heeft.

Opmerking
Door ε wat te laten groeien kan men er voor zorgen dat de verschilfunctie er een nulpunt bij krijgt.

We beperken ons nu tot polynomen $p_n(x)$ waarvan de verschilfunctie ten minste n + 1 nulpunten heeft. Een dergelijk polynoom kan in het algemeen niet meer op de boven beschreven wijze verbeterd worden.
Zij nu $E = \max_{x \in [a,b]} |p_n(x) - f(x)|$. Er is minstens één punt in $[a,b]$ waar $p_n - f$ de extreme waarde E of -E heeft.
Stel dat $e_1 < e_2 < \ldots < e_\ell$ de punten zijn waar dit het geval is. In twee opeenvolgende punten kan $p_n - f$ verschillend teken hebben. Als in de rij punten ten hoogste n tekenwisselingen van $p_n - f$ optreden kan p_n nog verbeterd worden.
Tussen elk tweetal punten e_i en e_{i+1} waar $p_n - f$ verschillend teken heeft kiezen we een nulpunt. We krijgen zo de nulpunten $x_1 \ldots x_k$ met $k \leq n$.
Het polynoom $\varepsilon(x - x_1) \ldots (x - x_k)$ is dus hoogstens van de graad n.
Beschouw nu $p_n - f + \varepsilon(x - x_1) \ldots (x - x_k)$.
Door voor ε het goede teken te kiezen worden de extremen in de punten e_1 t/m e_ℓ door $\varepsilon(x - x_1) \ldots (x - x_k)$ tegengewerkt. Door ε in absolute waarde klein te kiezen voorkomt men dat een van de kleinere extremen te veel groeit.
Het polynoom $p_n + \varepsilon(x - x_1) \ldots (x - x_k)$ is bij geschikte keuze van ε dus een betere benadering voor f(x) dan $p_n(x)$.

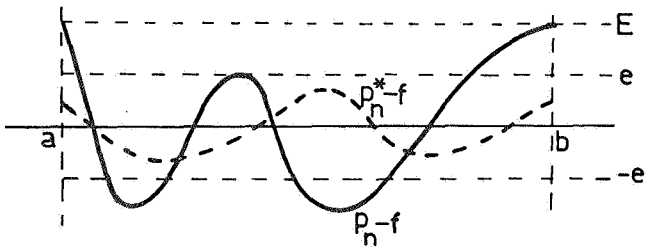




Uit het bovenstaande volgt dat het optimale polynoom $\hat{p}_n(x)$ de volgende eigenschap heeft: In $[a,b]$ zijn er $n+2$ punten $a \leq x_0 < x_1 \ldots < x_{n+1} \leq b$ waar $\hat{p}_n - f$ afwisselend de waarde E en -E heeft.

Deze eigenschap is karakteristiek voor $\hat{p}_n$. Men kan namelijk bewijzen dat $\hat{p}_n$ het enige polynoom uit K is dat deze eigenschap bezit.

Stel dat de verschilfunctie van het polynoom $p_n(x)$ reeds minstens $n+2$ alternerende extrema heeft. Men kan zich dan afvragen wat van een verbetering van $p_n(x)$ nog te verwachten is.

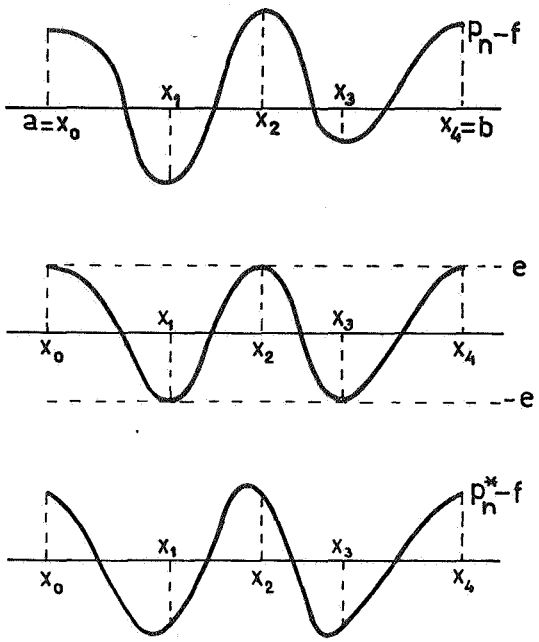


Zij e de absolute waarde van het absoluut kleinste extreem van $p_n - f$.
Neem aan dat er een polynoom $p_n^*(x)$ bestaat, zodat $|p_n^* - f| < e$ voor x in $[a, b]$.
Het polynoom $p_n - p_n^* = (p_n - f) - (p_n^* - f)$ is van de graad $\leq n$ en heeft in de punten waar $p_n - f$ een extreem heeft het teken van $p_n - f$, d.w.z. $p_n - p_n^*$ wisselt ten minste n+2 maal van teken en heeft dus tenminste n+1 nulpunten. Hieruit volgt $p_n - p_n^* \equiv 0$. Tegenspraak.
Het maximum van een verschilfunctie kan dus nooit kleiner dan e zijn.
Voor het optimale polynoom $\hat{p}_n$ geldt

$$e \leq \max_{x \in [a,b]} |\hat{p}_n(x) - f(x)| < E.$$

Dit resultaat is voor de numericus belangrijk. Bij het maken van een Chebyshev approximatie is het niet strikt nodig dat de extremen in absolute waarde gelijk zijn. Als de extremen op 10% na gelijk zijn heeft het meestal weinig zin om verder te gaan.
Als e groter is dan de vereiste precisie is dat een aanwijzing dat voor de approximatie een polynoom van hogere graad gebruikt moet worden.

Tenslotte beschrijven we nog een proces waarmee men het optimale polynoom kan benaderen.
Stel dat $p_n - f$ alternerende extremen heeft in de n+2 punten $x_0, x_1, \ldots x_{n+1}$. We nemen aan dat deze punten ongeveer goed zijn, d.w.z. dat de punten waar $\hat{p}_n(x) - f(x)$ extreem is weinig van $x_0 \ldots x_{n+1}$ afwijken



We zouden klaar zijn als de extremen alternerend waarden e, -e, e, ... hadden.
Men kan nu een polynoom p_n^* construeren waarvan de verschilfunctie in de punten x_0, x_1, ..., x_{n+1} waarden e, -e, e .... heeft, waarbij e voorals-nog onbekend is. Dit geeft nl. n + 2 vergelijkingen voor de n + 1 coëfficiënten van p_n^* en e.

Men kan p_n^* ook als volgt vinden.

Laat één punt weg, bijv. het punt x_m.

Zij $f^*(x)$ het interpolatie polynoom van $f(x)$ op de basispunten x_0, ... x_{m-1}, x_{m+1}, ... x_{n+1}, dus $f^*(x_i) = f(x_i)$ voor $i \neq m$.

Zij verder $k(x)$ het polynoom van de graad $\leq n$ waarvoor geldt $k(x_i) = (-1)^i$ voor $i \neq m$.

Neem $p_n^*(x) = f^*(x) + e\,k(x)$, dan is

$$p_n^*(x_i) - f(x_i) = (-1)^i e \text{ voor } i \neq m.$$

Dit moet ook voor i = m gelden. Hieruit volgt de waarde van e.

$$f^*(x_m) + e\,k(x_m) - f(x_m) = (-1)^m e$$

of

$$e = \frac{f(x_m) - f^*(x_m)}{k(x_m) - (-1)^m}.$$

De punten waar $p_n^* - f$ extreem is zullen iets verschoven zijn t.o.v.

x_o, x_1, ... x_{n+1}. Men kan echter hetzelfde proces nogmaals toepassen op p_n^* etc. De rij polynomen die men aldus construeert convergeert naar het optimale polynoom.

Om dit proces te starten heeft men n + 2 punten x_o, ..., x_{n+1} nodig. Hoe kan men deze punten verstandig kiezen.
Het interval [a, b] kan men altijd op [-1, 1] transformeren. We merken vervolgens op dat iedere voldoend gladde functie f(x) in het interval [-1, 1] te ontwikkelen is in een reeks van Chebyshev polynomen (zie ook cursus Wet. Rek. A), dus

$$f(x) = \sum_{k=o}^{\infty} a_k T_k(x) = \sum_{k=o}^{n} a_k T_k(x) + \sum_{k=n+1}^{\infty} a_k T_k(x).$$

Voor het polynoom $p_n(x)$ kiezen we nu $\sum_{k=o}^{\infty} a_k T_k(x)$, dan is

$f - p_n = \sum_{k=n+1}^{\infty} a_k T_k(x)$. Verder is bekend dat de coëfficiënten in de Chebyshev reeks van een gladde functie snel naar nul gaan, dus

$$f - p_n \approx a_{n+1} T_{n+1}(x).$$

Het polynoom $a_{n+1} T_{n+1}(x)$ heeft n + 2 extremen, nl. in de punten $x_k = \cos \frac{k\pi}{n+1}$ (k = 0, ..., n+1). Men kan nu het proces starten met deze n + 2 punten. Immers de waarde van $a_{n+1} T_{n+1}$ in deze punten is afwisselend a_{n+1} en $-a_{n+1}$; op grond van de karakteristieke eigenschap van $\hat{p}_n$ mag men dus verwachten dat p_n een goede benadering van het optimale polynoom is.

(De syllabus van het college Proces Analyse werd verzorgd door drs.C.Ligtmans, Technische Hogeschool te Eindhoven).

Errata behorende bij Proces analyse.

pag.	regel	oud	nieuw
42	3 v.o.	$F(x) - \log x$	$F(x) + \log x$
	1	$F(x) = \log x + \gamma + \Sigma$	$F(x) = -\log x - \gamma - \Sigma$
54	4, 5 en 6 v.o.	$2\,\varepsilon$	$4\,\varepsilon$
	2	$3\,\varepsilon$	$5\,\varepsilon$
65	6 v.o.	afhankelijk van de rij	afhankelijk van de kolom