RA

stichting
mathematisch
centrum

MC

RA

REKENAFDELING CR 24/71 JUNI

T.J. DEKKER en C.J. ROOTHART
INLEIDING IN DE NUMERIEKE WISKUNDE

Syllabus bij het Oriënterend Colloquium
Informatica 1970-1971

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

Inhoud

1. Het oplossen van vergelijkingen.

De meeste, zo niet alle, methoden voor het benaderen van wortels van vergelijkingen zijn iteratief.
Er wordt gestart met een zekere schatting voor een wortel, de startwaarde x_0, vervolgens wordt een zekere bewerking uitgevoerd waardoor een betere benadering van de betreffende wortel wordt verkregen.
Dit proces wordt herhaald en er ontstaat een rij iteratiewaarden $\{x_i\}_i$.
Het proces heet convergent als $\lim_{i\to\infty} x_i$ bestaat. Indien de limietwaarde voldoende nauwkeurig benaderd is wordt gestopt.
Het succes van de methode hangt vaak af van de startwaarde x_0. Soms is het moeilijk om een geschikte startwaarde te vinden. Het kan zijn, dat voor de berekening van x_{i+1} niet alleen gebruik gemaakt wordt van x_i maar ook van oudere iteratiewaarden. In dit geval is er ook meer dan één startwaarde nodig.
In de te behandelen methoden zijn de iteratiewaarden en de limiet reële of complexe getallen.
Er zijn ook iteratie-methoden waarbij de iteratiewaarden en de limiet vectoren of functies zijn.

1.1. Bisectie.

De tussenwaardestelling luidt:
Zij f een continue functie op [a,b], dan neemt f op (a,b) elke waarde aan tussen f(a) en f(b).
In het bijzonder heeft f een nulpunt tussen a en b als f(a) * f(b) < 0, d.w.z. als f(a) en f(b) tegengesteld teken hebben.
Zij m = (a+b)/2. Er zijn de volgende mogelijkheden:

A. f(m) = 0, m is dus een wortel van f(x) = 0.
B. f(m) * f(a) < 0, er is een nulpunt tussen a en m;
   b wordt vervangen door m en het proces wordt herhaald.
C. f(m) * f(b) < 0, er is een nulpunt tussen m en b;
   a wordt vervangen door m en het proces wordt herhaald.

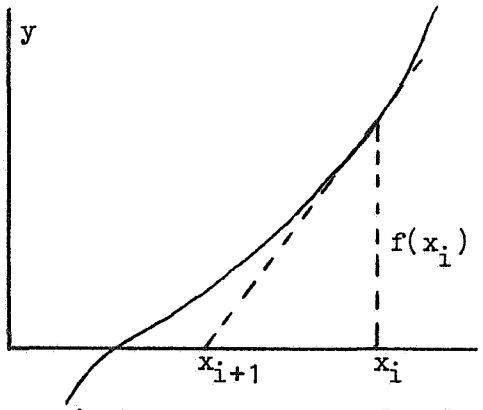
```
real procedure BISECTIE (x,fx,eps,a,b);
value eps,a,b; real x,fx,eps,a,b;
begin integer sa,st; x:=a; sa:=sign(fx);
half: x:=(a+b)/2; st:=sign(fx); if st=0 then goto fini;
      if sa*st > 0 then a:=x else b:=x;
      if abs(b-a) > eps then goto half else x:=(b+a)/2;
fini: BISECTIE:=x
end BISECTIE;
```

Elke stap wordt de lengte van het te beschouwen interval gehalveerd. Het proces is dus convergent.
Daar 2^{10} ongeveer gelijk is aan 10^3 moet ongeveer 10 keer gehalveerd worden om het oorspronkelijke interval met een factor 1000 te verkleinen.

1.2. De iteratieformule van Newton.

De iteratieformule van Newton luidt

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \quad .$$



Meetkundig betekent dit, dat in het punt $(x_i, f(x_i))$ de raaklijn aan de grafiek van de kromme $y = f(x)$ wordt getrokken. Het snijpunt van deze raaklijn en de x-as is de nieuwe iteratiewaarde x_{i+1}.

Analytisch wordt de formule als volgt verkregen.
Taylorreeksontwikkeling van $f(x_{i+1})$ om x_i geeft

$$f(x_{i+1}) = f(x_i) + (x_{i+1}-x_i)\, f'(x_i) + \frac{(x_{i+1}-x_i)^2}{2!}\, f''(x_i) + \ldots \; .$$

Linealiseren en nulstellen van het linkerlid levert de iteratieformule van Newton.

Dit proces convergeert niet altijd. Wel is convergentie verzekerd als tussen startwaarde en de wortel α van $f(x) = 0$ de grafiek van $y = f(x)$ met de bolle kant naar de x-as is gekeerd, in formule $f(x)\, f''(x) > 0$ voor x tussen x_0 en α.

Voorbeeld. Berekening van $\sqrt[N]{a}$.

$\sqrt[N]{a}$ kan worden geinterpreteerd als een wortel van de vergelijking $X^N - a = 0$.

Voor deze vergelijking luidt de iteratieformule

$$x_{i+1} = x_i - \frac{x_i^N - a}{N\, x_i^{N-1}} = \frac{(N-1)x_i^N + a}{N\, x_i^{N-1}} .$$

Voor N = 2, d.w.z. voor de berekening van $\sqrt{a}$,

$$x_{i+1} = \frac{1}{2}\left(x_i + \frac{a}{x_i}\right) .$$

De methode van Newton kan worden gecombineerd met bisectie.
Er moeten dan twee punten bekend zijn, waarvan de functiewaarden tegengesteld van teken zijn. Een stap met de methode van Newton wordt geaccepteerd als het een iteratiewaarde levert, die tussen de twee punten ligt, anders verworpen. Ingeval van verwerping wordt bisectie toegepast. Wordt de Newton-stap geaccepteerd, dan wordt de functiewaarde in het gevonden punt uitgerekend en het veilige interval kan worden verkleind, rekening houdend met het teken van de functie in het laatste punt.

```
real procedure NEWTON (x,a,b,fx,dfdx,tolx,count);
value a,b; real x,a,b,fx,dfdx,tolx; integer count;
begin real t,fa,fb,ft,tol;
      x:=b; fb:=fx; t:=x:=a; ft:=fa:=fx; count:=2; tol:=0;
weer: t:=a-fa/dfdx;
      if sign(a-t)≠sign(t-b) then t:=(a+b)/2 else
      if abs(t-a) < tol then t:=a+sign(b-a)*tol;
      x:=t; ft:=fx; count:=count+1;
      if sign(ft) = sign(fb) then
      begin b:=a; fb:=fa end;
      a:=t; fa:=ft; tol:=abs(tolx);
      if abs(b-a) > tol then goto weer;
      NEWTON:=a
end NEWTON;
```

1.3. Convergentie-snelheid van Newtons proces.

Zij α de gezochte wortel van $f(x) = 0$ en $\delta_i = x_i - \alpha$ de fout in de i-de iteratiestap.
Als voor alle voldoend grote i geldt

$$\delta_{i+1} = C\delta_i^m + O(\delta_i^{m+1})$$

voor constante C en m, dan heet m de orde van het proces en C de convergentiefactor. De orde is een maat voor de convergentie-snelheid van het iteratieproces. Voor m = 2 spreken we ook van kwadratische convergentie en voor m = 1 van lineaire convergentie. (Bisectie is dus lineair convergent met factor $\frac{1}{2}$).
Substitutie van de Taylorreeksen van $f(x_i)$ en $f'(x_i)$ in de iteratieformule van Newton levert

$$x_{i+1} = x_i - \frac{(x_i-\alpha)\, f'(\alpha) + \frac{1}{2}(x_i-\alpha)^2 f''(\alpha) + \ldots}{f'(\alpha) + (x_i-\alpha)\, f''(\alpha) + \ldots}$$

$$x_{i+1}-\alpha = (x_i-\alpha)\left(1 - \frac{f'(\alpha) + \frac{1}{2}(x_i-\alpha)\, f''(\alpha) + \ldots}{f'(\alpha) + (x_i-\alpha)\, f''(\alpha) + \ldots}\right)$$

$$= (x_i-\alpha)\left(\frac{\frac{1}{2}f''(\alpha)\,(x_i-\alpha) + \ldots}{f'(\alpha) + (x_i-\alpha)\, f''(\alpha) + \ldots}\right).$$

Indien $f'(\alpha) \neq 0$, dan geldt

$$x_{i+1}-\alpha = \frac{f''(\alpha)}{2\, f'(\alpha)}(x_i-\alpha)^2 + O(x_i-\alpha)^3.$$

Voor een enkelvoudige wortel van $f(x) = 0$ geldt dus

$$\delta_{i+1} = C\,\delta_i^2 + O(\delta_i^3)\ ,$$

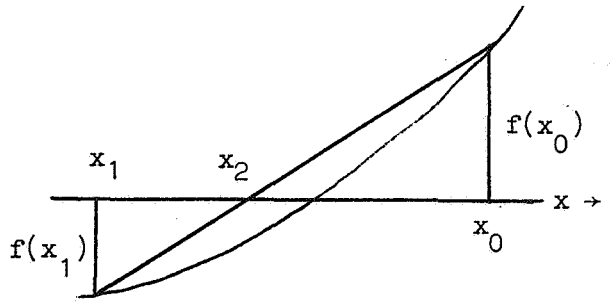
m.a.w., het proces is van de orde 2.

Indien $f'(\alpha) = 0$, maar $f''(\alpha) \neq 0$, dan geldt

$$\delta_{i+1} = \tfrac{1}{2}\delta_i + O(\delta_i^2) ,$$

m.a.w. het proces is van de orde 1. In dit geval is de convergentie-factor, $\frac{1}{2}$, een maat voor de convergentie-snelheid.

1.4. Regula falsi.



Laten x_0 en x_1 twee niet gelijke startwaarden voor het bepalen van een wortel van $f(x) = 0$ zijn. De nieuwe iteratiewaarde x_2 is het snijpunt van de lijn door de punten $(x_0, f(x_0))$ en $(x_1, f(x_1))$ met de x-as:

$$x_2 = x_1 - \frac{x_1 - x_0}{f(x_1) - f(x_0)} f(x_1).$$

De volgende stap wordt uitgevoerd met x_1 en x_2 en de betreffende functiewaarden, enz.
De algemene iteratieformule luidt

$$(1.4.1) \qquad x_{i+1} = x_i - \frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} f(x_i) \quad .$$

Dit proces convergeert niet altijd. Zelfs als gestart wordt met punten x_0 en x_1, waarvoor geldt dat $f(x_0) * f(x_1) < 0$, kan divergentie optreden, terwijl er toch een nulpunt is tussen x_0 en x_1 als f continu is.
De regula falsi kan ook als volgt worden toegepast. Er wordt gestart met startwaarden x_0 en x_1 zodanig, dat $f(x_0) * f(x_1) < 0$. Met formule (1.4.1) wordt x_2 berekend.
Als $f(x_2) = 0$ dan is x_2 een wortel van $f(x) = 0$. Als $f(x_2) * f(x_1) < 0$, dan wordt vervolgd met x_1 en x_2. Is echter $f(x_2) * f(x_0) < 0$, dan wordt de volgende stap uitgevoerd met x_0 en x_2. Deze variant op de regula falsi is veilig. Convergentie naar een tussen x_0 en x_1 gelegen wortel van $f(x) = 0$ is verzekerd.

1.5. Convergentie-snelheid regula falsi.

Stellen we weer $\delta_i = x_i - \alpha$, waarbij α een wortel van $f(x) = 0$ is, dan kan men bewijzen, dat voor alle voldoend grote i bij benadering geldt:

$$\delta_{i+1} \approx C\ \delta_i\ \delta_{i-1}.$$

Bij de zojuist geschetste veilige variant, kan het zijn dat x_{i-1}, en dus ook δ_{i-1}, lange tijd constant blijft, zodat de convergentie dan slechts lineair is.
Kiezen we in formule (1.4.1) steeds x_i en x_{i-1} gelijk aan de twee nieuwste iteratie-waarden, dan geldt: als het proces convergeert, dan convergeert het sneller dan lineair. Men kan aantonen dat de convergentie-orde dan gelijk is aan

$$(1+\sqrt{5})/2 \approx 1.618.$$

Hieronder volgt een ALGOL 60 procedure, waarin regula falsi en bisectie zodanig worden gecombineerd, dat het proces veilig is (d.w.z. altijd convergeert), terwijl toch (op den duur) de bovengenoemde hoge convergentie-orde wordt bereikt.

Description mca 2310

zeroin searches for a zero of a function between the given values of x and y within a certain tolerance. The function and the tolerance are, in this order, given by the actual parameters for fx and tolx, which are expressions depending on the Jensen variable x.

zeroin:= true, if either the function values at the given point x and y have different sign, or the procedure finds some point in between at which the sign of the function value differs from that at x and y. Then zeroin calculates and delivers two values x and y lying within the given interval, having function values of different sign and satisfying abs(x − y) ≤ 2 × tolx.
Moreover, the absolute function value is not greater at x than at y, so that the delivered value of x is the best value for the zero.
If the function has a continuous second derivative, the order of convergence is about 1.6.

zeroin:= false, if the procedure fails to find points at which the function values have different sign. Then the delivered values of x and y satisfy all the above conditions, except the sign change condition.

One has to take care that tolx is never smaller than the machine precision. Then in either case the process is completed after a finite number of steps, an upper bound for the required number of steps being the length of the given interval divided by the minimum of the tolerance.

```
comment mca 2310;
boolean procedure zeroin(x, y, fx, tolx); real x, y, fx, tolx;
begin real a, fa, b, fb, c, fc, tol, m, p, q;
    a:= x; fa:= fx; b:= x:= y; fb:= fx;
interpolate: c:= a; fc:= fa;
extrapolate: if abs(fc) < abs(fb) then
    begin a:= b; fa:= fb; x:= b:= c; fb:= fc; c:= a; fc:= fa
    end interchange;
    tol:= tolx; m:= (c + b) × .5; if abs(m − b) > tol then
    begin p:= (b − a) × fb; if p > 0 then q:= fa − fb else
        begin q:= fb − fa; p:= − p end;
        a:= b; fa:= fb;
        x:= b:= if p < abs(q) × tol then sign(c − b) × tol + b else
        if p < (m − b) × q then p / q + b else m; fb:= fx;
        goto if sign(fb) = sign(fc) then interpolate else
        extrapolate
    end;
    y:= c; zeroin:= sign(fb) × sign(fc) ≤ 0
end zeroin;
```

1.6. Vergelijking van de behandelde methoden.

We hebben gezien dat voor een enkelvoudige wortel de formule van Newton van de orde 2, bisectie is van de orde 1 en de regula falsi van de orde 1.618 is.
Bij bisectie en regula falsi moet per stap één waarde f(x) worden uitgerekend, bij de methode van Newton echter ook een waarde f'(x).
Als de berekening van f'(x) vrijwel niets kost (b.v. als f een oplossing van een differentiaalvergelijking is), dan is het proces van Newton dus het gunstigst van de drie geschetste methoden.
Als de berekening van f'(x) ongeveer evenveel kost als de berekening van f(x), dan kost één Newton-stap ongeveer evenveel als 2 stappen regula falsi.
Uit

$$\delta_{i+2} \approx C\ \delta_{i+1}^{m} \approx C'\ \delta_{i}^{m^2}$$

volgt dat de methode van Newton minder snel is dan de regula falsi, aangezien voor deze formule geldt $m^2 > 2$.
Men zegt ook wel dat de effectieve convergentie-orde van de formule van Newton gelijk is $\sqrt{2} \approx 1.414$.

1.7. Algebraïsche vergelijkingen.

Bovengenoemde methoden worden veelvuldig toegepast voor het oplossen van algebraïsche vergelijkingen. Zij

$$(1.7.1.) \qquad f(x) = x^n + a_1 x^{n-1} + \ldots + a_{n-1} x + a_n .$$

Er zijn diverse formules die een bovengrens voor de modulus van alle wortels van $f(x) = 0$ geven. Een niet al te royale bovengrens is

$$w = 2 \max_{k=1,\ldots,n} |a_k|^{1/k},$$

waarvoor tevens geldt dat er minstens één wortel is, waarvan de modulus minstens $w/2n$ is.

Als de graad n oneven is, is er minstens één reële wortel en de functie f heeft tegengesteld teken voor $x = w$ en $x = -w$, zodat het interval $[-w,w]$ kan worden gekozen als uitgangspunt voor de procedure zeroin of NEWTON om een reële wortel α te bepalen. Zie onderstaand programma. Vervolgens kunnen we de factor $x - \alpha$ uitdelen, waardoor een veelterm van de graad n-1 ontstaat.
Algebraïsche vergelijkingen van even graad hoeven geen reële wortels te hebben. Indien ze wel reële wortels hebben is het niet altijd eenvoudig een interval te bepalen waarin precies één reële wortel gelegen is. Een methode, gebruik makend van Sturm-rijen, komt in de volgende sectie ter sprake. Het bepalen van complexe wortels is in het algemeen ingewikkelder, o.a. omdat dan de tussenwaarde-stelling ons niet kan helpen.
We gaan hier niet verder op in.

```
begin   comment reele wortel van onevengraadsvergelijking.
        Dit programma gebruikt zeroin, zie pag. 7;

        integer n,k; real v,r,x,y,eps,tol;

        real procedure pol(x,graad,a); value x,graad;
        real x; integer graad; array a;
        begin   integer j; real r;
                r:= 1;
                for j:= 1 step 1 until graad do r:= r × x + a[j];
                pol:= r;
                print(x); print(r); nlcr
        end pol;

        eps:= read;
in:     n:= read; if n < 0 then goto einde;
        begin   array a[1:n];
                v:= 0;
                for k:= 1 step 1 until n do
                begin   a[k]:= read;
                        r:= abs(a[k]) ↑ (1/k);
                        if r > v then v:= r
                end;

                x:= 2 × v; y:= -x; tol:= v × eps;
                if zeroin(x,y,pol(x,n,a),tol)
                then
                begin   printtext(‡wortel =   ‡);
                        print(x)
                end
                else printtext(‡geen wortel gevonden‡);
                nlcr; nlcr
        end;
        goto in;
einde:
end
```

Invoergegevens

```
10-10
3        0        0        -2
5        5        10       10       5        -242
0
```

Resultaten

```
+.25198420997880+ 1 +.13999999999960+ 2
-.25198420997880+ 1 -.17999999999970+ 2
+.31498026247570- 0 -.19687500000000+ 1
+.58681254378920- 0 -.17979317099230+ 1
+.15533273217880+ 1 +.17479081891000+ 1
+.10768875660580+ 1 -.75114767232840- 0
+.12200922935100+ 1 -.18373985984360- 0
+.12664653182950+ 1 +.31327293512730- 1
+.12597104915550+ 1 -.10025540268540- 2
+.12599199598720+ 1 -.51909082685600- 5
+.12599210500780+ 1 +.86947693489490- 9
+.12599210498960+ 1 +.36379788070920- 11
+.12599210497700+ 1 -.59299054555590- 9
wortel =    +.125992104989600+ 1
```

```
            +10                  +160808
            -10                   -59292
-.46122671512930+ 1 -.85803426079080+ 3
-.45331546126540+ 1 -.79357085009290+ 3
-.35592470291010+ 1 -.35278955860160+ 3
-.27797574428770+ 1 -.26085681869230+ 3
-.56797550461490- 0 -.24298494981440+ 3
+.47160122476930+ 1 +.58589093772320+ 4
-.35756060229100- 0 -.24289056384350+ 3
+.21792258227000+ 1 +.81793168739420+ 2
+.15401676276230+ 1 -.13724232735020+ 3
+.19405859870690+ 1 -.23128258294430+ 2
+.19931903398100+ 1 -.27454204671080+ 1
+.20002757647090+ 1 +.11170524149200- 0
+.19999987453610+ 1 -.50812843255700- 3
+.19999999997690+ 1 -.93365088105200- 7
+.20000000002690+ 1 +.10919757187370- 6
wortel =    +.199999999976900+ 1
```

1.8. Eigenwaarden van symmetrische matrices.

Zij A een vierkante matrix van de orde n. De vergelijking

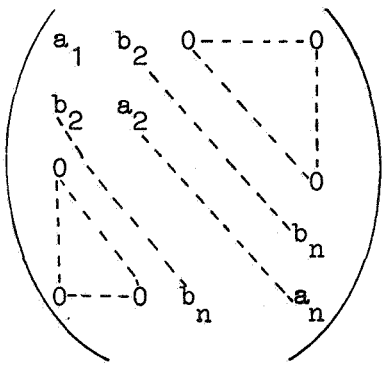
(1.8.1) $\quad Ax = \lambda x,$

waarbij λ een scalar en x een vector met n elementen is, heeft dan en slechts dan een oplossing x ongelijk aan de nulvector, als

(1.8.2) $\quad \det(\lambda I - A) = 0.$

Deze karakteristieke vergelijking van A is een algebraïsche vergelijking van de graad n. De wortels λ heten eigenwaarden van A en de bijbehorende vectoren die voldoen aan (1.8.1) heten eigenvectoren van A.
We nemen aan dat A reëel symmetrisch is. De eigenwaarden zijn dan alle reëel en er zijn n lineair onafhankelijke eigenvectoren die zo gekozen kunnen worden, dat ze een reële orthogonale matrix vormen.
Verder beperken we ons tot het in de praktijk veel voorkomende geval dat A een tridiagonale matrix is, dat is een matrix waarvan alle elementen buiten de hoofddiagonaal en de aanliggende nevendiagonalen nul zijn. Wegens de symmetrie heeft A dan dus de gedaante

(1.8.3) $\quad A =$



.

De waarde van de karakteristieke veelterm $\det(\lambda I - A)$ voor willekeurig gegeven λ wordt berekend met behulp van de recursie-formule

$$f_0 = 1,$$
$$f_1 = \lambda - a_1,$$
$$f_i = (\lambda - a_i)\, f_{i-1} - b_i^2 f_{i-2}, \quad i = 2, \ldots, n.$$

Dan is f_i gelijk aan det $(\lambda I - A_i)$, waarbij A_i de submatrix van A is bestaande uit de eerste i rijen en kolommen van A. dus

$$f_n = \det(\lambda I - A).$$

Bovendien geldt:

(1.8.4) Stelling van Givens.

Als in A, gedefinieerd volgens (1.8.3), alle nevendiagonaalelementen b_i ongelijk aan 0 zijn, dan vormt voor elke λ de rij $f_0, f_1, \ldots, f_n$ een Sturmrij, d.w.z. het aantal eigenwaarden van A groter dan λ is gelijk aan het aantal tekenwisselingen in deze rij, mits $f_n \neq 0$.

Bewijs

Twee opeenvolgende waarden f_{i-1}, f_i kunnen niet beide gelijk aan 0 zijn, anders zouden, volgens bovengenoemde recursie-formule, alle waarden $f_0, f_1, \ldots, f_n$ gelijk aan 0 zijn, wat blijkbaar onmogelijk is.
Als voor zekere $i \leq n$ geldt $f_{i-1} = 0$, dan hebben f_{i-2} en f_i dus tegengesteld teken. Hieruit volgt dat het aantal tekenwisselingen ongewijzigd blijft bij het passeren van een wortel van $f_{i-1} = 0$. Dus het aantal tekenwisselingen kan alleen veranderen bij het passeren van een wortel van $f_n = 0$. Aangezien voor $\lambda = +\infty$ het aantal tekenwisselingen 0 is en voor $\lambda = -\infty$ dit aantal n is, wordt het aantal tekenwisselingen bij het passeren van elke eigenwaarde met 1 verminderd, waaruit de stelling volgt.

Deze stelling is een belangrijk hulpmiddel voor de berekening der eigenwaarden. Men vormt steeds kleinere intervallen met behulp van bisectie. Heeft men eenmaal een interval gevonden, waarin blijkens de tekenwisselingen in de Sturmrijen (berekend op de uiteinden van het interval) precies één eigenwaarde zit, dan kan men deze met behulp van zeroin (zie pag. 7) berekenen. Bij een meervoudige eigenwaarde vindt men zo'n interval niet, maar dan levert bisectie de eigenwaarde met zijn multipliciteit.

1.9. Eigenvectoren van symmetrische matrices.

Bij een eigenwaarde λ van A vindt men een eigenvector x door oplossing van het homogene lineaire stelsel (vgl. 1.8.1)

$$(A-\lambda I)\, x = \underline{0}$$

(hierbij duidt $\underline{0}$ de nulvector aan).
Aangezien de matrix van dit stelsel singulier is, kunnen we een vergelijking schrappen.
Aangezien x op een factor na bepaald is, kunnen we een component van x die ongelijk aan 0 is gelijk aan (bijvoorbeeld) -1 stellen, d.w.z. we kunnen de betreffende kolom van $A - \lambda I$ naar rechts brengen.
Hierdoor ontstaat een inhomogeen lineair stelsel van de orde n - 1, waarvan de matrix vaak niet singulier is, en dan met behulp van eliminatie kan worden opgelost. Het probleem hierbij is, welke vergelijking kunnen we het beste schrappen en welke kolom naar rechts brengen? Bovendien beschikken we in de praktijk slechts over een numerieke benadering van een eigenwaarde.

Inverse iteratie

Deze problemen worden omzeild door het stelsel als volgt iteratief op te lossen. Zij nu λ een numerieke benadering van een eigenwaarde λ_1 van A en $x^{(0)} \neq \underline{0}$ een vector, die als startvector voor de iteratie wordt gekozen.
Hieruit berekenen we een vector x door het inhomogene lineaire stelsel

$$(1.9.1) \qquad (A-\lambda I)\, x = x^{(0)}$$

op te lossen met behulp van eliminatie.
Vervolgens normeren we x (d.w.z. we delen x door zijn Euclidische lengte $||x||$) en noemen de aldus verkregen vector $x^{(1)}$, in formule

$$(1.9.2) \qquad x^{(1)} = x/||x||.$$

(1.9.3) Stelling

De volgens (1.9.1) en (1.9.2) berekende vector $x^{(1)}$ is een goede benadering

van de bij λ_1 behorende eigenvector v_1 van A als aan de volgende voorwaarden is voldaan:

1) λ_1 ligt niet te dicht bij de andere eigenwaarden $\lambda_2, \ldots, \lambda_n$;

2) λ is een goede schatting van λ_1;

3) de component van $x^{(0)}$ in de richting van v_1 is niet klein.

Bewijs

Laten $v_1, \ldots, v_n$ onderling loodrechte genormeerde eigenvectoren van A zijn horende bij respectievelijk $\lambda_1, \ldots, \lambda_n$. We kunnen dan schrijven

$$x^{(0)} = \alpha_1 v_1 + \ldots + \alpha_n v_n. \qquad (1.9.4)$$

Hieruit en uit (1.9.1) volgt

$$x = \frac{\alpha_1}{\lambda_1-\lambda} v_1 + \ldots + \frac{\alpha_n}{\lambda_n-\lambda} v_n. \qquad (1.9.5)$$

Uit bovengenoemde voorwaarden (1) en (2) volgt, dat $|\lambda_1-\lambda|$ veel kleiner is dan $|\lambda_k-\lambda|$ voor $k = 2, \ldots, n$. Uit voorwaarde (3) volgt dat $|\alpha_1|$ niet klein is (met name niet veel kleiner dan $|\alpha_2|, \ldots, |\alpha_n|$). Hieruit kunnen we concluderen dat in (1.9.5) de coëfficiënt van v_1 in absolute waarde veel groter is dan de andere coëfficiënten. Dus $x^{(1)}$, verkregen door x te normeren, is nagenoeg gelijk aan de genormeerde eigenvector v_1, waarmee de stelling bewezen is.

Vervolgens kunnen we op analoge wijze uit $x^{(1)}$ een vector $x^{(2)}$ berekenen, enz. Dit iteratie-proces convergeert meestal zeer snel. In de praktijk zijn 1 of 2 stappen bijna altijd voldoende. Voor meervoudige eigenwaarden is het proces iets gecompliceerder, omdat dan maatregelen moeten worden genomen om onderling loodrechte eigenvectoren te krijgen (bijvoorbeeld d.m.v. Gram-Schmidt-orthogonalisatie).
Als A, en dus ook A - λI, een tridiagonale matrix is, is het eliminatieproces waarmee het stelsel (1.9.1) wordt opgelost betrekkelijk eenvoudig. De onbekenden worden geëlimineerd in de natuurlijke volgorde. Omdat de matrix tridiagonaal is, zijn daardoor telkens bij het elimineren van een

onbekende slechts twee vergelijkingen betrokken. Een geschikt veelvoud van die vergelijking die voor de betreffende onbekende de grootste coëfficiënt in absolute waarde heeft, wordt van de andere vergelijking afgetrokken. De genoemde grootste coëfficiënt heet pivot en het proces wordt genoemd Gauss' eliminatie met pivoten of met (stabiliserende) rijverwisselingen.

2. Numerieke integratie.

Een bepaalde integraal wordt gedefinieerd door een limietproces. De limiet kan analytisch worden bepaald als een primitieve van de integrand gevonden kan worden. Het is de taak van de numerieke wiskunde om een integraal, die niet of moeilijk langs analytische weg verkregen kan worden, te berekenen. Het resultaat moet in een eindig aantal stappen worden verkregen.

2.1. De integratie-formules van Newton-Cotes.

Deze integratie-formules zijn van de volgende vorm

$$\int_{x_0}^{x_{n-1}} f(x)dx = h(A_0f_0+A_1f_1+\ldots+A_{n-1}f_{n-1}) + R_n(h). \tag{2.1.1}$$

Hierin is h een vast getal, $x_i = x_0+ih$ en $f_i = f(x_i)$ voor $i = 0(1)n-1$. De punten x_i heten basispunten en $R_n(h)$ heet restterm. De coëfficiënten A_i hangen niet af van de te integreren functie f en worden bepaald als volgt. We vervangen f door een polynoom f^* van de graad kleiner dan n, die in de basispunten met f overeenstemt en eisen dan dat geldt

$$\int_{x_0}^{x_{n-1}} f^*(x)dx = h(A_0f_0+A_1f_1+\ldots+A_{n-1}f_{n-1}),$$

dus $R_n(h) = 0$ voor $f = f^*$.
Kiezen we $n = 2$ dan wordt formule (2.1.1)

$$\int_{x_0}^{x_1} f(x)dx = h(A_0f_0+A_1f_1) + R_2(h).$$

Het eerstegraadspolynoom, dat in de basispunten x_0 en x_1 met f overeenstemt is

$$f^*(x) = \frac{x-x_1}{x_0-x_1} f_0 + \frac{x-x_0}{x_1-x_0} f_1.$$

Zij $x = x_0+ph$, dan krijgen we

$$\int_{x_0}^{x_1} f^*(x)dx = \int_{x_0}^{x_1} \left(\frac{x-x_1}{x_0-x_1} f_0 + \frac{x-x_0}{x_1-x_0} f_1\right)dx =$$

$$h\left[-\left(\int_0^1 (p-1)dp\right)f_0 + \left(\int_0^1 pdp\right)f_1\right] =$$

$$\frac{h}{2}(f_0+f_1).$$

Dus $A_0 = A_1 = \frac{1}{2}$.
Zij f tweemaal differentieerbaar. Voor de restterm geldt

$$R_2(h) = \int_{x_0}^{x_1} f(x)dx - \int_{x_0}^{x_1} f^*(x)dx =$$

$$\int_{x_0}^{x_0+h} f(x)dx - \frac{h}{2}(f(x_0)+f(x_0+h)).$$

We differentiëren tweemaal naar h:

$$R_2'(h) = f(x_0+h) - \tfrac{1}{2}f(x_0) - \tfrac{1}{2}f(x_0+h) - \tfrac{h}{2}f'(x_0+h).$$

$$R_2''(h) = -\frac{h}{2} f''(x_0+h).$$

Tevens geldt $R_2(0) = R_2'(0) = 0$.
Integreren levert

$$R_2'(h) = -\frac{1}{2}\int_0^h tf''(x_0+t)dt.$$

De middelwaardestelling uit de integraalrekening luidt:
Als de functies f en g continu zijn op [a,b] en op het hele interval geldt of $g(x) \geq 0$ of $g(x) \leq 0$, dan is er een ξ in het interval zodanig, dat

$$\int_a^b f(x)\, g(x)dx = f(\xi)\int_a^b g(x)dx.$$

Voor $R_2'(h)$ levert dit, aannemende dat f" continu is

$$R_2'(h) = -\frac{h^2}{4} f''(x_0+\theta h), \qquad 0 < \theta < 1.$$

Dus

$$R_2(h) = -\frac{1}{4}\int_0^h t^2 f''(x_0+\theta t)dt.$$

Nogmaals toepassen van de middelwaardestelling levert

$$R_2(h) = -\frac{h^3}{12} f''(\xi), \qquad x_0 < \xi < x_1.$$

De integratieformule luidt dus

$$\int_{x_0}^{x_1} f(x)dx = \frac{h}{2}(f_0+f_1) - \frac{h^3}{12} f''(\xi).$$

Deze formule wordt de trapeziumregel genoemd.

Een integratie-formule is van de orde n, als de formule exact is voor alle polynomen van de graad kleiner dan n, maar niet voor polynomen van de graad n.
Uit $R_2(h) = -\frac{h^3}{12} f''(\xi)$ volgt, dat de trapeziumregel exact is voor alle polynomen van de graad kleiner dan twee, maar niet voor tweedegraadspolynomen.
De orde van deze formule is dus twee.
Enkele formules van Newton-Cotes luiden

(2.1.2) $$\int_{x_0}^{x_1} f(x)dx = \frac{h}{2}(f_0+f_1) - \frac{h^3}{12} f''(\xi)$$

(2.1.3) $$\int_{x_0}^{x_2} f(x)dx = \frac{h}{3}(f_0+4f_1+f_2) - \frac{h^5}{90} f^{(4)}(\xi)$$

(2.1.4) $$\int_{x_0}^{x_3} f(x)dx = \frac{3h}{8}(f_0+3f_1+3f_2+f_3) - \frac{3h^5}{80} f^{(4)}(\xi)$$

(2.1.5) $$\int_{x_0}^{x_4} f(x)dx = \frac{4h}{90}(7f_0+32f_1+12f_2+32f_3+7f_4) - \frac{8h^7}{945} f^{(6)}(\xi).$$

Formule (2.1.3) heet de formule van Simpson.

2.2. Richardson-correctie.

De formules van Newton-Cotes kunnen worden aaneengeregen. Voor constante h krijgen we voor de trapeziumregel b.v.

$$\int_{x_0}^{x_m} f(x)dx = h(\tfrac{1}{2}f_0+f_1+\ldots+f_{m-1}+\tfrac{1}{2}f_m) - \frac{x_m-x_0}{12} h^2 f''(\xi).$$

Deze formules hebben echter het nadeel, dat de fout over de verschillende intervallen ter lengte h sterk kan variëren. Liever willen we h variëren en wel zodanig, dat de fout per stap ongeveer constant is.
Zij I_1 een benadering van de integraal over een interval ter lengte 2h, verkregen door 2 stappen ter lengte h met de trapeziumregel en I_2 een benadering van de integraal over hetzelfde interval eveneens verkregen met de trapeziumregel, maar met één stap ter lengte 2h, dus

$$I_1 = \frac{h}{2} (f_0+2f_1+f_2),$$

$$I_2 = h (f_0+f_2).$$

Dan geldt

$$\int_{x_0}^{x_2} f(x)dx = I_1 - \frac{2}{12} h^3 f''(\xi_1), \tag{2.2.1}$$

$$\int_{x_0}^{x_2} f(x)dx = I_2 - \frac{8}{12} h^3 f''(\xi_2). \tag{2.2.2}$$

Nemen we aan, dat f'' constant is op het betreffende interval, dan volgt uit (2.2.1) en (2.2.2) dat

$$-\frac{2}{12} h^3 f''(\xi_1) = \frac{I_1-I_2}{3}.$$

We nemen dit als correctieterm. De benadering van de integraal wordt

$$I = I_1 + \frac{I_1-I_2}{3} = \frac{h}{3} (f_0+4f_1+f_2).$$

Dit is de formule van Simpson, maar nu met een idee van de nauwkeurigheid, n.l. de aangebrachte correctieterm. Deze correctie heet Richardson-correctie. De correctieterm voor de formule van Simpson luidt

$$\frac{I_1-I_2}{15},$$

waarbij

$$I_1 = \frac{h}{3}(f_0+4f_1+2f_2+4f_3+f_4),$$

$$I_2 = \frac{h}{3}(2f_0+8f_2+2f_4).$$

We krijgen dan een formule van de 6e orde equivalent met (2.1.5)

$$\int_{x_0}^{x_4} f(x)dx = I_1 + \frac{I_1-I_2}{15} + O(h^7).$$

In onderstaande procedure qad wordt $\int_a^b f(x)dx$ berekend met de formule van Simpson met Richardson-correctie.

Description of qad

The real procedure qad calculates the definite integral from a to b of fx, where fx is an expression depending on x; both a < b and a > b are allowed.
In array e[1:3], one must give the relative tolerance a[1] and the absolute tolerance e[2].
The procedure delivers the value of the required definite integral as value of the procedure identifier and the number of steps smaller than abs(b-a) × e[1] in e[3].
The integral is calculated by means of Simpson's rule with Richardson correction. If the fourth difference is too big (and thus also the correction term), the total interval is split into two equal parts and the integration process is invoked recursively.
This process uses small steps only in those parts of the integration interval where the integrand or one of its derivatives is (nearly) singular, so that the procedure is useful for smooth integrands as well as for integrands having singularities.

```
real procedure qad(x, a, b, fx, e); value a, b; real x, a, b, fx;
array e;
begin real x0, x1, x2, f0, f1, f2, t, v, sum, hmin, re, ae;

    procedure int;
    begin real x3, x4, f3, f4, h;
        x4:= x2; x2:= x1; f4:= f2; f2:= f1;
    anew: x:= x1:= (x0 + x2) × .5; f1:= fx;
        x:= x3:= (x2 + x4) × .5; f3:= fx; h:= x4 - x0;
        v:= (4 × (f1 + f3) + 2 × f2 + f0 + f4) × 15;
        t:= 6 × f2 - 4 × (f1 + f3) + f0 + f4;
        if abs(t) < abs(v) × re + ae then sum:= sum + h × (v - t)
        else if abs(h) < hmin then e[3]:= e[3] + 1 else
        begin int; x2:= x3; f2:= f3; goto anew end;
        x0:= x4; f0:= f4
    end int;

    re:= e[1]; ae:= e[2] × 180; e[3]:= 0; hmin:= abs(b - a) × re;
    x:= x0:= a; f0:= fx; x:= x2:= b; f2:= fx;
    x:= x1:= (x0 + x2) × .5; f1:= fx; sum:= 0; int; qad:= sum / 180
end qad;
```

```
begin   comment enige aanroepen van de procedure qad;
        integer k;real x,y,z; array e[1:3];

        e[1]:=10-7; e[2]:= 10-8; for k:= 1 step 1 until 10 do
        begin z:=qad(x,0,1,x↑(1/k),e);
              y:=k/(k+1);print(z); print(y); print(z-y);
              FIXT(5,0,e[3]); nlcr
        end
end
```

```
+.5000000000000₁₀-  0 +.5000000000000₁₀-  0          -0          +0
+.6666666664278₁₀-  0 +.6666666666670₁₀-  0 -.2391971065663₁₀-  9 +1
+.7499999992833₁₀-  0 +.7500000000000₁₀-  0 -.7166818249971₁₀-  9 +2
+.7999999979420₁₀-  0 +.8000000000002₁₀-  0 -.2058186510112₁₀-  8 +2
+.8333333263708₁₀-  0 +.8333333333330₁₀-  0 -.6962181942072₁₀-  8 +3
+.8571428437072₁₀-  0 +.8571428571431₁₀-  0 -.1343596522929₁₀-  7 +3
+.8749999812089₁₀-  0 +.8750000000000₁₀-  0 -.1879107003333₁₀-  7 +3
+.8888888644351₁₀-  0 +.8888888888887₁₀-  0 -.2445358404657₁₀-  7 +3
+.8999999698190₁₀-  0 +.8999999999996₁₀-  0 -.3018067218363₁₀-  7 +3
+.9090908732742₁₀-  0 +.9090909090910₁₀-  0 -.3581681085052₁₀-  7 +3
```

3. Gewone differentiaalvergelijkingen

De algemene vorm van een gewone differentiaalvergelijking is

$$F(x,y,y^{(1)},y^{(2)},\ldots,y^{(n)}) = 0,$$

waarbij y en $y^{(i)}$ functies van x zijn, dus

$$y = y(x) \text{ en } y^{(i)} = y^{(i)}(x),\ i = 1(1)n,$$

en $y^{(i)}$ de i^{de} afgeleide van y(x) is, dus

$$y^{(i)} = \frac{d^{(i)}y(x)}{dx^i}.$$

Een differentiaalvergelijking is van de orde n als in de vergelijking de n^{de} afgeleide voorkomt, maar geen afgeleiden hoger dan n.
Een differentiaalvergelijking van de orde 1 heeft dus als algemene vorm

$$F(x,y,y') = 0,$$

waarbij $y' = y^{(1)}$.

Indien y' als functie van x en y wordt geschreven dan ontstaat de expliciete vorm

$$y' = f(x,y).$$

De oplossing van een differentiaalvergelijking is in het algemeen niet eenduidig.

Voorbeeld

De algemene oplossing van y' = xy luidt

$$y = Ke^{\frac{1}{2}x^2}.$$

Een oplossing kan worden vastgelegd door in een punt x_0 de functiewaarde $y_0 = y(x_0)$ te geven.

De oplossing van dit voorbeeld luidt dan

$$y = y_0 e^{\frac{1}{2}(x^2-x_0^2)}.$$

Wordt een oplossing van een differentiaalvergelijking van de orde 1 vastgelegd, dan ontstaat een beginwaarde-probleem van de orde 1:

$$(3.0.1)\quad \begin{cases} y' = f(x,y) \\ y(x_0) = y_0. \end{cases}$$

De algemene vorm van een beginwaarde-probleem van de orde n is

$$\begin{cases} F(x,y,y',\ldots,y^{(n)}) = 0 \\ y(x_0) = y_0 \\ y'(x_0) = y_1 \\ \quad\vdots \\ y^{(n-1)}(x_0) = y_{n-1}. \end{cases}$$

$x_0, y_0, y_1, \ldots, y_{n-1}$ zijn gegeven getallen.
Hier worden alleen in één punt waarden van functie en afgeleiden gegeven. Worden daarentegen dergelijke waarden in meer dan een punt gegeven, dan spreken we van randwaarde-problemen.
We zullen ons beperken tot het numeriek oplossen van het beginwaarde-probleem (3.0.1).

3.1. Meerstap-formules

Zij h een vast getal, $x_i = x_0+ih$ en $g_i = g(x_i)$ voor $i = 1(1)n+1$. In formule (2.1.2), de trapeziumregel, vervangen we x_0 door x_n, x_1 door x_{n+1} en de functie f door de functie g. We krijgen

$$\int_{x_n}^{x_{n+1}} g(x)dx = \frac{h}{2}(g_{n+1}+g_n) - \frac{h^3}{12} g''(\xi).$$

Zij G(x) een primitieve van g(x), dan kan de trapeziumregel als volgt worden geschreven

$$G_{n+1} = G_n + \frac{h}{2}(g_{n+1}+g_n) - \frac{h^3}{12} g''(\xi).$$

We vervangen g(x) door f(x,y(x)) en daar y een primitieve van f is mogen we G door y vervangen:

$$y_{n+1} = y_n + \frac{h}{2}(f_{n+1}+f_n) - \frac{h^3}{12} y^{(3)}(\xi),$$

waarbij $y_i = y(x_i)$ en $f_i = f(x_i,y_i)$ voor i = n, n+1.
Deze formule is weer van de orde 2 (exact voor polynomen van de graad kleiner dan 2, niet voor polynomen van de graad 2).
Er bestaan ook formules van hogere orde van dit type. Enkele formules luiden

(3.1.1) $$y_{n+1} = y_n + \frac{h}{2}(f_{n+1}+f_n) - \frac{h^3}{12} y^{(3)}(\xi)$$

(3.1.2) $$y_{n+1} = y_n + \frac{h}{12}(5f_{n+1}+8f_n-f_{n-1}) - \frac{h^4}{24} y^{(4)}(\xi)$$

(3.1.3) $$y_{n+1} = y_n + \frac{h}{24}(9f_{n+1}+19f_n-5f_{n-1}+f_{n-2}) - \frac{19}{720} h^5 y^{(5)}(\xi)$$

(3.1.4) $$y_{n+1} = y_n + \frac{h}{720}(251f_{n+1}+646f_n-264f_{n-1}+106f_{n-2}-19f_{n-3}) - \frac{3}{160} h^6 y^{(6)}(\xi).$$

In de rechterleden van deze formules komt f_{n+1} voor. Daar f_{n+1} afhankelijk is van y_{n+1} kunnen deze formules niet direkt worden toegepast.
Enkele formules waarbij f_{n+1} in de rechterleden ontbreekt zijn

(3.1.5) $$y_{n+1} = y_n + hf_n + \frac{1}{2} h^2 y^{(2)}(\xi)$$

(3.1.6) $$y_{n+1} = y_n + \frac{h}{2}(3f_n-f_{n-1}) + \frac{5}{12} h^3 y^{(3)}(\xi)$$

$$(3.1.7) \qquad y_{n+1} = y_n + \frac{h}{12}(23f_n - 16f_{n-1} + 5f_{n-2}) + \frac{3}{8} h^4 y^{(4)}(\xi)$$

$$(3.1.8) \qquad y_{n+1} = y_n + \frac{h}{24}(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3}) + \frac{251}{720} h^5 y^{(5)}(\xi).$$

Formules (3.1.1) t.e.m. (3.1.8) zijn met uitzondering van formule (3.1.5) meerstap-formules. Voor het berekenen van y_{n+1} wordt gebruik gemaakt van meer dan één der reeds berekende functiewaarden.
Hiernaast bestaan ook eenstap-formules, die voor het berekenen van y_{n+1} uitsluitend gebruik maken van informatie in het basispunt x_n. Deze formules zijn gebaseerd op de Taylorreeksontwikkeling van de onbekende functie y.
De eenvoudigste eenstap-formule is (3.1.5) en heet de formule van Euler.
De eenstap-formules hebben het voordeel, dat telkens de staplengte h zonder moeite gevarieerd kan worden, maar het nadeel (althans bij formules van hogere orde) dat meer rekenwerk nodig is per stap.

Keren we terug naar bovengenoemde meerstap-formules. We merken op, dat de constanten in de resttermen van (3.1.6) t.e.m. (3.1.8) veel groter zijn dan de constanten in de resttermen van de overeenkomstige formules (3.1.1) t.e.m. (3.1.3).
De formules kunnen als volgt worden gecombineerd.
We berekenen m.b.v. één der formules (3.1.5) t.e.m. (3.1.8) een benaderde waarde voor y_{n+1} en passen vervolgens een of meer malen één van de formules (3.1.1) t.e.m. (3.1.4) toe.
Combineren we formule (3.1.5) met (3.1.1) dan krijgen we

$$(3.1.9) \qquad y_{n+1}^{(0)} = y_n^{(k)} + hf(x_n, y_n^{(k)})$$

$$(3.1.10) \qquad y_{n+1}^{(i)} = y_n^{(k)} + \frac{h}{2}(f(x_{n+1}, y_{n+1}^{(i-1)}) + f(x_n, y_n^{(k)})), \quad i = 1(1)k,$$

waarbij $y_{n+1}^{(i)}$ benaderingen zijn voor y_{n+1}, i = 0(1)k.
In dit geval heet (3.1.9) de predictor-formule en (3.1.10) de corrector-formule.
In de procedure precor, die in onderstaand programma is gedeclareerd, worden de formules (3.1.9) en (3.1.10) gecombineerd. De corrector-formule

wordt één keer toegepast.

Zij $\quad corr = |y_{n+1}^{(1)} - y_{n+1}^{(0)}| = |\frac{1}{2}h^2 y^{(2)}(\xi_1) + \frac{1}{12}h^3 y^{(3)}(\xi_2)|,$

dus $\quad corr = O(h^2).$

Zij $y(x_0) = y(a)$ gegeven en $y(x_n) = y(b)$ gevraagd, dan wordt geeist dat voor een interval ter lengte h geldt

$$(3.1.11) \qquad corr \leq \left|\frac{h*toly}{b-a}\right| = tol,$$

waarbij toly een bepaalde tolerantie is.
Voor het gehele interval geldt dan

$$\sum_{i=0}^{n-1} |y_{i+1}^{(1)} - y_{i+1}^{(0)}| \leq toly.$$

Is niet aan (3.1.11) voldaan dan wordt de berekende functiewaarde $y_{i+1}^{(1)}$ verworpen, de staplengte h wordt gehalveerd en met de nieuwe staplengte en met formules (3.1.9) en (3.1.10) een nieuwe $y_{i+1}^{(1)}$ berekend.
Indien echter $corr \leq tol$ dan wordt $y_{i+1}^{(1)}$ geaccepteerd. Vervolgens wordt nog onderzocht of de staplengte kan worden verdubbeld. Daar $corr = O(h^2)$ en $tol = O(h)$ is dit bij benadering het geval als

$$corr < tol/2.$$

In de procedure precor is een veiligheidsmarge in acht genomen. Voor verdubbelen van de staplengte wordt geeist

$$corr < .4 * tol.$$

```
begin real a,b,x,y;

    procedure precor(x,a,b,toly,y,fxy);
    value a,b; real x,a,b,toly,y,fxy;
    begin real h,z,f,tol,corr,w; boolean last;
       x:= a; h:= b-a; z:= w:= y;
    step: if sign(x+h-b)=sign(b-a)\x+h=b then
    begin last:= true; h:= b-x end;
       f:= fxy;y:= y+h×f; x:= x+h;
       z:= z+h×(fxy+f)/2;tol:= abs(h×toly/(b-a));
       corr:= abs(y-z); if corr>tol then
       begin x:= x-h; h:= h/2; last:= false;
             z:= y:= w; goto step
       end;
       w:= y:= z; if ¬ last then
       begin if corr <.4×tol then h:= h+h; goto step end
    end precor;

    y:= 1; for a:= 0 step.1 until .91 do
    begin b:= a+.1; precor(x,a,b,-4×y+-5,y,-2××y×y);
       nlcr;fixt(2,1,x);fixt(2,10,y);fixt(2,10,1/(1+b×b))
    end
end
```

+.1	+.9900990099	+.9900990099
+.2	+.9615384623	+.9615384615
+.3	+.9174311963	+.9174311927
+.4	+.8620689935	+.8620689655
+.5	+.8000001333	+.8000000000
+.6	+.7352992615	+.7352941176
+.7	+.6711466756	+.6711409396
+.8	+.6097610836	+.6097560976
+.9	+.5524903922	+.5524861878
+1.0	+.5000035013	+.5000000000

3.2. Stelsels differentiaalvergelijkingen en differentiaalvergelijkingen van hogere orde

De algemene vorm van een stelsel differentiaalvergelijkingen van de eerste orde luidt

$$(3.2.1)\quad \begin{cases} y_1' = f_1(x,y_1,y_2,\ldots,y_m) \\ y_2' = f_2(x,y_1,y_2,\ldots,y_m) \\ \qquad\vdots \\ y_m' = f_m(x,y_1,y_2,\ldots,y_m), \end{cases}$$

waarbij y_i functies van x zijn, dus $y_i = y_i(x)$, $i = 1(1)m$.
Zijn van de functies y_i de functiewaarden in een punt x_0 gegeven, dus

$$y_i(x_0) = \alpha_i \text{ voor } i = 1(1)m,$$

dan hebben we weer een beginwaarde-probleem.
De procedure precor kan gemakkelijk worden uitgebreid zodanig, dat de procedure geschikt is voor het oplossen van stelsels differentiaalvergelijkingen van de eerste orde of ook voor het oplossen van een enkele differentiaalvergelijking van hogere orde.

4. Sommeren van reeksen

Wanneer de termen van een reeks snel genoeg naar nul convergeren, biedt het berekenen van de som niet veel moeilijkheden. Men neemt eenvoudig zoveel termen mee, als voor de vereiste precisie nodig zijn. In het algemeen is het evenwel gevaarlijk, op te houden zodra men een verwaarloosbaar kleine term ontmoet. Bijvoorbeeld de reeks

$$\sum_{k=1}^{\infty} \frac{\sin(k\pi/3)}{k!}$$

convergeert weliswaar snel, maar men mag toch niet ophouden bij de derde term, die toevallig nul is. Als wapen hiertegen kan men een ongelovigheidsparameter "tim" invoeren met de betekenis:
neem zoveel termen mee, totdat tim keer achtereen de termen verwaarloosbaar klein blijken te zijn.
Alleen als men weet dat de termen in absolute waarde monotoon afnemen, mag men gewoon tim = 1 kiezen. We gaan nu enige methoden bespreken voor het sommeren van langzaam convergerende reeksen.

4.1. De transformatie van Euler

Op langzaam convergerende alternerende reeksen kan men met succes de transformatie van Euler toepassen.
Beschouwen we de alternerende reeks

$$S = \sum_{k=0}^{\infty} u_k, \tag{4.1.1}$$

waarbij u_k positief is voor k even en anders negatief.
We gaan nu van elk paar opeenvolgende termen de helft bij elkaar voegen aldus:

$$\begin{aligned} S &= u_0 + u_1 + u_2 + u_3 \ \ldots \\ &= \tfrac{1}{2}u_0 + \tfrac{1}{2}(u_0 + u_1) + \tfrac{1}{2}(u_1 + u_2) + \tfrac{1}{2}(u_2 + u_3) + \ldots . \end{aligned}$$

Voeren we de gemiddelde-operator M in gedefinieerd door

$$(4.1.2) \qquad Mu_k = \tfrac{1}{2}(u_k + u_{k+1}),$$

dan kunnen we S schrijven in de vorm

$$S = \tfrac{1}{2}u_0 + \sum_{k=0}^{\infty} Mu_k.$$

Passen we op de aldus ontstane reeks dezelfde transformatie toe, dan krijgen we

$$S = \tfrac{1}{2}u_0 + \tfrac{1}{2}Mu_0 + \sum_{k=0}^{\infty} M^2u_k.$$

Herhalen we dit procédé oneindig vaak, dan krijgen we de getransformeerde reeks

$$(4.1.3) \qquad T = \tfrac{1}{2}u_0 + \tfrac{1}{2}Mu_0 + \tfrac{1}{2}M^2u_0 + \ldots = \tfrac{1}{2}\sum_{j=0}^{\infty} M^ju_0.$$

De omzetting van de reeks (4.1.1) in de reeks (4.1.3) heet de <u>transformatie van Euler</u>. Hiervoor geldt de volgende

(4.1.4) <u>Stelling</u>. Als de reeks (4.1.1) convergeert en de reeks

$$\sum_{k=0}^{\infty} u_k z^k$$

een functie voorstelt die analytisch is in het punt z = 1, dan convergeert de reeks (4.1.3) ook en beide reeksen hebben dezelfde som.

De transformatie van Euler levert vaak grote convergentie-versnelling op voor alternerende langzaam convergerende reeksen.

<u>Euler-transformatie met strategie van Van Wijngaarden</u>

Men kan de Euler-transformatie uitstellen, d.w.z. eerst enige termen gewoon bij elkaar optellen en dan de resterende reeks sommeren volgens Euler. Nog

mooier is het om tijdens de opbouw van het gemiddelden-schema telkens te bepalen of men een Euler-stap zal zetten of de Euler-transformatie een keer zal uitstellen. Zijn op een gegeven moment r termen gewoon opgeteld en vervolgens n Euler-stappen gezet, dan luidt de tot dan toe verkregen partiële som

$$(4.1.5) \qquad S_{r,n} = u_0 + u_1 + \ldots + u_{r-1} + \tfrac{1}{2} \sum_{k=0}^{n} M^k u_r .$$

Accepteren we de volgende Euler-term $M^{n+1}u_r$, dan wordt n opgehoogd en we hebben blijkbaar

$$S_{r,n+1} = S_{r,n} + \tfrac{1}{2} M^{n+1} u_r .$$

Stellen we daarentegen de Euler-sommatie een keer uit, dan wordt r opgehoogd en men kan bewijzen, dat

$$S_{r+1,n} = S_{r,n} + M^{n+1} u_r .$$

In de strategie van Van Wijngaarden wordt de Euler-term $M^{n+1}u_r$ geaccepteerd, dan en slechts dan als $|M^{n+1}u_r| < |M^n u_{r+1}|$.
Dit proces is beschreven in de procedure euler , gepubliceerd in Revised Report on the Algorithmic Language ALGOL 60, door P. Naur (editor).
We lichten dit proces en het gebruik van de procedure euler toe m.b.v. onderstaand programma, waarin een benadering berekend wordt van
$\ln(2) \approx 0.693147$ en $\pi/4 \approx 0.785398$.

```
begin real som; integer q;

        procedure kop;
        begin nlcr; printtext(‹gemiddelden-schema en partiele sommen›);
                nlcr; nlcr
        end;
        procedure print(x); fixt(1, 6, x);
        procedure print1(x); begin nlcr; fixt(7, 10, x); nlcr end;

        real procedure f(n); value n; integer n;
        f:= (if n : 2 × 2 = n then 1 else -1) / (q × n + 1);

        comment Procedure ontleend aan
        P. Naur (editor), Revised report on the algorithmic
        language ALGOL 60.
        Enige statements zijn ingelast om het gemiddelden-
        schema en de partiele sommen af te drukken;
        procedure euler(fct, sum, eps, tim); value eps, tim;
        real procedure fct; real sum, eps; integer tim;
        comment euler computes the sum of fct(i) for i from zero up to
        infinity by means of a suitable refined euler transformation.
        The summation is stopped as soon as tim times in succession the
        absolute value of the terms of the transformed series are found
        to be less than eps. Hence, one should provide a function fct
        with one integer argument, an upper bound eps, and an integer
        tim. The output is the sum sum. euler is particularly efficient
        in the case of a slowly convergent or divergent alternating
        series;
        begin integer i, k, n, t; array m[0: 15]; real mn, mp, ds;
                i:= n:= t:= 0; m[0]:= fct(0); sum:= m[0] / 2;
                        print(m[0]); print1(sum);
        nextterm: i:= i + 1; mn:= fct(i);
                        print(mn);
                for k:= 0 step 1 until n do
                begin mp:= (mn + m[k]) / 2; m[k]:= mn; mn:= mp;
                        print(mn)
                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;
                        print1(sum);
                if abs(ds) < eps then t:= t + 1 else t:= 0;
                if t < tim then go to nextterm
        end euler;

        kop; q:= 1;
        euler(f, som, -4, 2);
        nlcr; printtext( ‹ln(2) :  › ); print(som); nlcr;
        kop; q:= 2;
        euler(f, som, -4, 2);
        nlcr; printtext( ‹pi :     › ); print(som × 4); nlcr
end
```

gemiddelden-schema en partiele sommen

```
+1.000000
       +.5000000000
 -.500000  +.250000
       +.6250000000
 +.333333  -.083333  +.083333
       +.7083333333
 -.250000  +.041667  -.020833
       +.6979166667
 +.200000  -.025000  +.008333  -.006250
       +.6947916667
 -.166667  +.016667  -.004167  +.002083  -.002083
       +.6937500000
 +.142857  -.011905  +.002381  -.000893  +.000595  -.000744
       +.6930059524
 -.125000  +.008929  -.001488  +.000446  -.000223  +.000186
       +.6930989583
 +.111111  -.006944  +.000992  -.000248  +.000099  -.000062  +.000062
       +.6931299603
```

ln(2) : +.693130

gemiddelden-schema en partiele sommen

```
+1.000000
       +.5000000000
 -.333333  +.333333
       +.6666666667
 +.200000  -.066667  +.133333
       +.8000000000
 -.142857  +.028571  -.019048
       +.7904761905
 +.111111  -.015873  +.006349  -.006349
       +.7873015873
 -.090909  +.010101  -.002886  +.001732  -.002309
       +.7849927850
 +.076923  -.006993  +.001554  -.000666  +.000533
       +.7852591853
 -.066667  +.005128  -.000932  +.000311  -.000178  +.000178
       +.7853479853
 +.058824  -.003922  +.000603  -.000165  +.000073  -.000052  +.000063
       +.7854106678
```

pi : +3.141643

4.2. De transformatie van Van Wijngaarden

Deze transformatie is vooral geschikt voor langzaam convergerende reeksen met positieve termen. Zij gegeven de reeks

$$S = \sum_{k=1}^{\infty} u_k. \tag{4.2.1}$$

Stellen we nu, voor k = 1,2,3,...,

$$v_k = u_k + 2u_{2k} + 4u_{4k} + 8u_{8k} + \ldots, \tag{4.2.2}$$

dan geldt

$$u_k = v_k - 2v_{2k},$$

dus

$$S = v_1 - 2v_2 + v_2 - 2v_4 + v_3 + v_3 - 2v_6 + \ldots .$$

Samennemen van gelijksoortige termen levert de getransformeerde reeks

$$T = v_1 - v_2 + v_3 - v_4 + \ldots = \sum_{k=1}^{\infty} (-1)^{k-1} v_k. \tag{4.2.3}$$

Voor deze transformatie geldt

(4.2.4) Stelling. Als de reeks (4.2.1) convergeert, dan convergeert de getransformeerde reeks (4.2.3) dan en slechts naar dezelfde limiet als de reeksen (4.2.2) convergent zijn voor alle k en

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

In het bijzonder is de transformatie geoorloofd als de oorspronkelijke reeks (4.2.1) wordt gemajoreerd door een convergente reeks waarvan de termen monotoon niet-stijgend zijn.
Als de termen van de oorspronkelijke reeks positief zijn, dan is de getransformeerde reeks (4.2.3) alternerend en kan vaak gemakkelijk m.b.v. de

transformatie van Euler gesommeerd worden. Omdat de indices van de termen in de reeksen (4.2.2) met machten van twee oplopen, convergeren deze reeksen meestal sneller dan (4.2.1). Hierin ligt juist het succes van deze transformatie. Bovendien wordt formule (4.2.2) alleen gebruikt om termen v_k met oneven index k te berekenen; de termen met even index volgen uit de relatie

$$v_{2k} = (v_k - u_k)/2.$$

5. Literatuur

M. Abramowitz & I.A. Stegun, Handbook of mathematical functions, National Bureau of Standards, AMS 55 (1964).

S.D. Conte, Elementary numerical analysis, McGraw-Hill (1965).

T.J. Dekker, Cursus Wetenschappelijk Rekenen A, Numerieke wiskunde, 3 delen, Mathematisch Centrum (1967).

T.J. Dekker, ALGOL 60 procedures in numerical algebra, part 1, MC Tracts 22, Mathematisch Centrum (1968).

T.J. Dekker & W. Hoffmann, ALGOL 60 procedures in numerical algebra, part 2, MC Tracts 23, Mathematisch Centrum (1968).

D.K. Faddeev & V.N. Faddeeva, Computational methods of linear algebra (translated from Russian), Freeman & CO. (1963).

G.E. Forsyte & C.B. Moler, Computer solution of linear algebraic systems, Prentice Hall (1967).

L. Fox, An introduction to numerical linear algebra, Oxford Univ. Press (1964).

C.-E. Fröberg, Introduction to numerical analysis, Addison-Wesley (1965).

R.W. Hamming, Numerical methods for scientists and engineers, McGraw-Hill (1962).

P. Henrici, Discrete variable methods in ordinary differential equations, Wiley (1962).

P. Henrici, Elements of numerical analysis, Wiley (1964).

F.B. Hildebrand, Introduction to numerical analysis, McGraw-Hill (1956).

A.S. Householder, The theory of matrices in numerical analysis, Blaisdell (1964).

E. Isaacson & H.B. Keller, Analysis of numerical methods, Wiley (1966).

J.A. Jacquez, A first course in computing and numerical methods, Addison-Wesley (1970).

N. Macon, Numerical analysis, Wiley (1963).

J.M. McCormich & M.C. Salvadori, Numerical methods in FORTRAN, Prentice-Hall (1964).

P. Naur (editor), Revised report on the Algorithmic Language ALGOL 60, Regnecentralen, Copenhagen (1962).

R.H. Pennington, Introductory computer methods and numerical analysis, MacMillan, New York (1965).

A. Ralston, A first course in numerical analysis, McGraw-Hill (1965).

J.J. Seidel, Computerwiskunde, Aula 407, Het Spectrum (1969).

E. van Spiegel, Cursus Wetenschappelijk Rekenen B, Numerieke analyse, 3 delen, Mathematisch Centrum (1964).

E. Stiefel, Einführung in die numerische Mathematik, Teubner (1961).

R.S. Varga, Matrix iterative analysis, Prentice-Hall (1962).

A. van Wijngaarden, Cursus Wetenschappelijk Rekenen B, Proces-analyse, Mathematisch Centrum (1965).

J.H. Wilkinson, Rounding errors in algebraic processes, Her Majesty's Stationary Office (1963).

J.H. Wilkinson, The algebraic eigenvalue problem, Clarendon Press (1965).