stichting
mathematisch
centrum

MC

REKENAFDELING

CR 19/71 DECEMBER

CURSUS WETENSCHAPPELIJK REKENEN A

T.J. DEKKER

NUMERIEKE WISKUNDE, DEEL II

2e boerhaavestraat 49 amsterdam

Hoofdstuk 5. Oplossen van algebraïsche en transcendente vergelijkingen

0. Inleiding

Voor literatuur zie NPL's Modern Computing Methods, Hoofdstuk 6. In dit hoofdstuk gaan we ons bezig houden met het numeriek oplossen van vergelijkingen met één onbekende. Men spreekt van wortels van een vergelijking of ook nulpunten van een functie. Eerst zullen wij procédé's behandelen, die bruikbaar zijn voor willekeurige continue functies, daarna beschouwen we meer in het bijzonder procédé's ter bepaling van nulpunten van polynomen, m.a.w. wortels van algebraïsche vergelijkingen. Ter inleiding een

0.1 Voorbeeld. Berekening van vierkantswortels.

Zij gevraagd $x = \sqrt{a}$.

We zoeken p en q zodanig, dat $a = pq$, bijv. $p = 1$, $q = a$, maar liever nog zó, dat p en q dicht bij elkaar liggen. Als $p = q$, zijn we klaar, want dan geldt $x = p$. Als $p \neq q$, vinden we een betere schatting door het gemiddelde te nemen: $m = (p+q)/2$. Bij deze m, die te groot is, vinden we een schatting die te klein is, door m op a te delen: $h = a/m$.

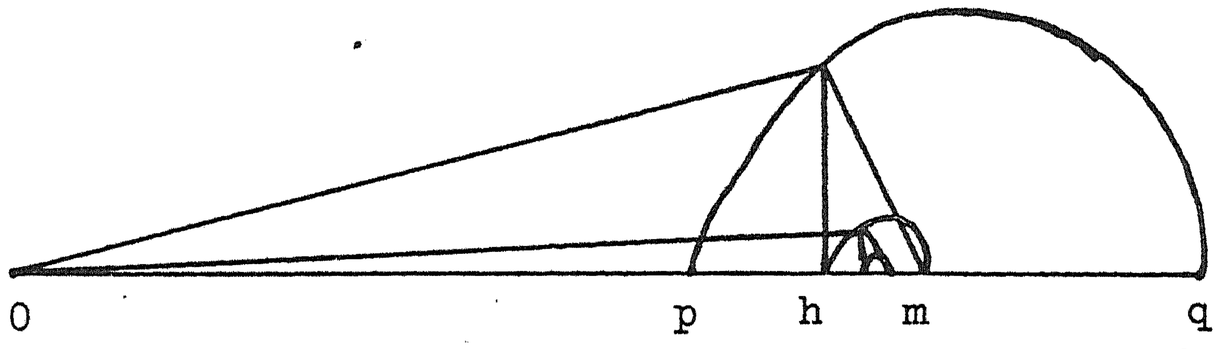
Blijkbaar geldt: $hm = a$, dus h en m kunnen de rol van p en q overnemen.

Met deze nieuwe p en q herhalen (oftewel "itereren") we bovenstaand rekenschema, totdat h en m zó dicht bij elkaar zullen komen, dat voldoende precisie bereikt is. Dit iteratie-proces moge worden toegelicht met het volgende voorbeeld en bijbehorend plaatje.

Voorbeeld. Gevraagd $x = \sqrt{2}$.

p	q
1	2
4/3	3/2 = 1.5
24/17	17/12 ≈ 1.4167
	577/408 ≈ 1.4142157

Vergelijk $\sqrt{2} \approx 1.414213562373$.



Dit iteratie-proces convergeert zeer snel; in elke stap wordt het aantal correcte cijfers ongeveer verdubbeld. Dit heet quadratische convergentie. Het proces zal blijken te zijn de methode van Newton (zie onder).

0.2 Iteratie

Het bepalen van wortels van niet-lineaire vergelijkingen geschiedt meestal iteratief. Daarom nu iets over iteratie in het algemeen.
Een proces heet iteratie-proces, als het bestaat uit het herhaald uitvoeren van een bepaald rekenschema, de z.g.n. "iteratie-stap". Precieser:
Men gaat uit van een startwaarde x_0 (ook beginschatting genoemd). Daarna wordt voor $i = 0, 1, 2, \ldots$ de i-de iteratie-stap uitgevoerd, welke behelst het berekenen van x_{i+1} uit x_i. Deze x_i heten "iterates" of "iteratie-waarden".
Een iteratie-proces heet convergent als $\lim_{i\to\infty} x_i$ bestaat.
Indien het proces voldoende snel convergeert, wordt de iteratie voortgezet, totdat de limiet-waarde in de gewenste precisie benaderd is.
Het kan zijn, dat in de i-de iteratie-stap voor de berekening van x_{i+1} niet alleen x_i, maar ook oudere iteratie-waarden nodig zijn. Dan heeft men natuurlijk ook meer dan één startwaarde nodig (zie bijvoorbeeld Regula falsi).
In dit hoofdstuk zijn de iteratie-waarden en de gezochte limiet steeds (reële of complexe) getallen. Het kunnen echter ook vectoren zijn (zie blz. 146) of zelfs functies (bijv. bij het oplossen van differentiaalvergelijkingen).

Convergentie-snelheid

Een maat voor de convergentie-snelheid is de z.g.n. orde van het iteratie-proces. Zij α de gezochte limiet en $\delta_i = x_i - \alpha$ de fout in de i-de iterate.

Als nu op den duur (voor voldoende grote i) bij benadering geldt:

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

voor constante C en m, dan heet m de orde van het proces.
Als m = 1 of 2 spreken we ook van lineaire resp. quadratische convergentie. Bij lineaire convergentie is ook de convergentie-factor C van belang.

1. Bisectie

Deze methode gaat uit van de

Tussenwaarde-stelling. Een continue functie neemt in een interval (a,b) elke waarde tussen f(a) en f(b) aan.
In het bijzonder: als f(a) en f(b) tegengesteld teken hebben, dan heeft f tussen a en b een nulpunt. Op grond hiervan kan van een continue functie f die in a en b tegengestelde teken heeft, een nulpunt bepaald worden als volgt.
Neem het midden van het interval: m = (a+b)/2 en bereken f(m). Er zijn drie mogelijkheden:

1) f(m) = 0, dan is m dus een nulpunt;
2) f(m) . f(a) > 0, m.a.w. f(m) . f(b) < 0, dus nu ligt er een nulpunt tussen m en b. Vervang nu a door m en itereer, d.w.z. herhaal het bovenstaande met de nieuwe a en b;
3) f(m) . f(a) < 0, dus dan is er een nulpunt tussen a en m. Vervang dan b door m en itereer.

Dit proces heet bisectie. De iterates x_i zijn hier de opeenvolgende middenpunten m. Bij elke stap wordt de lengte van het te beschouwen interval gehalveerd. Het proces convergeert dus altijd en $\lim_{i\to\infty} x_i$ is een nulpunt van f.
(Dit proces kan zelfs als bewijs van de tussenwaarde-stelling dienen.)
De convergentie is lineair met convergentie-factor $\frac{1}{2}$. M.a.w. elke stap levert één bit (= bineair cijfer) meer precisie. Daar $2^{10} \approx 10^3$, gebruiken we de vuist-regel, dat elke 10 bisectie-stappen een winst levert van 3 decimalen.

2. Lineaire (inverse) interpolatie = Regula falsi

Bij bisectie is de enige informatie, die we aan de functie ontlenen, het teken. Bij lineaire interpolatie gebruiken we ook de waarde. Deze methode gaat uit van de onjuiste aanname (vandaar "regula falsi") dat de functie lineair is. Laten twee iterates met bijbehorende functie-waarden (x_{i-1}, f_{i-1}) en (x_i, f_i) gegeven zijn. Dan wordt het snijpunt x_{i+1} van de lijn door deze twee punten met de x-as bepaald:

$$x_{i+1} = x_i - \frac{x_i - x_{i-1}}{f_i - f_{i-1}} f_i .$$

Hierbij zijn dus twee startwaarden x_0 en x_1 nodig. Dit proces gebruikt in elke stap de twee laatste iteratie-waarden. Helaas convergeert het niet altijd, maar als het convergeert, convergeert het wel snel.

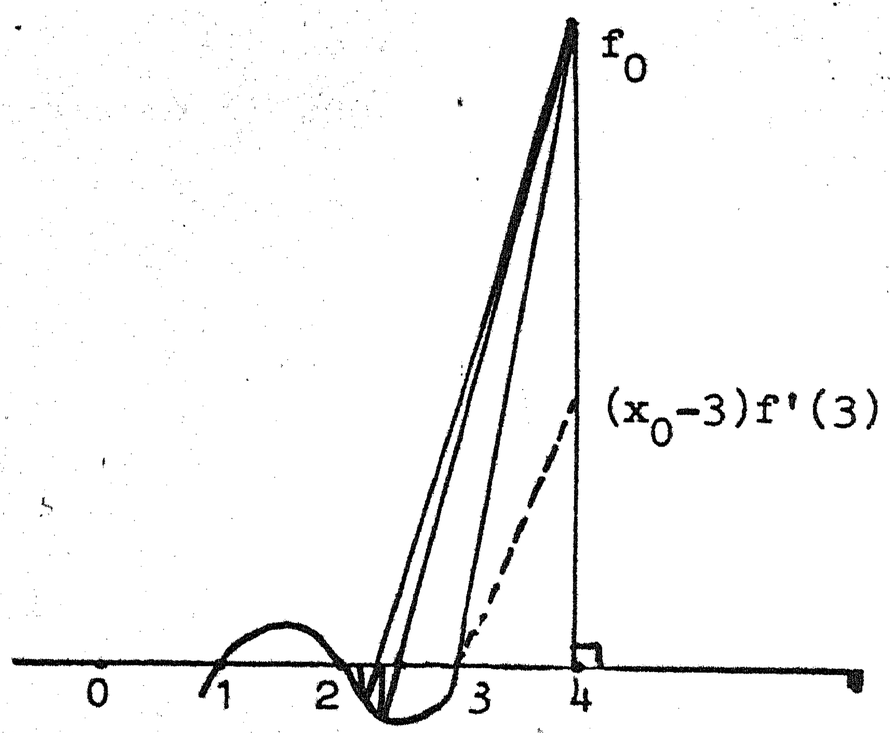
2.1 Stricte interpolatie

Een veilige, maar veel langzamer convergerende methode krijgen we, als we in elke stap interpoleren tussen twee iterates, wier functie-waarden tegengesteld van teken zijn. Dit heet interpolatie in stricte zin; als de gebruikte functie-waarden hetzelfde teken hebben spreken we soms van extrapolatie.

2.2 Voorbeeld. $f(x) = x^3 - 6x^2 + 11x - 6 = (x-1)(x-2)(x-3)$.

Startwaarden: $x_0 = 4$ en $x_1 = 2.5$.

x	f(x)	Δx
4	6	
2.5	-.375	+.09
2.6	-.384	+.08
2.7	-.357	+.07
2.8	-.288	+.05
2.9	-.171	+.030
2.93		



Dit convergeert langzaam, omdat we voor de stricte interpolatie steeds het oude punt $x_0 = 4$ moeten gebruiken. Extrapolatie tussen 2.8 en 2.9 zou leveren $\Delta x = +.146$, dus $x = 3.046$, een veel betere benadering.

2.3 Convergentie-snelheid

Wij beschouwen eerst de stricte interpolatie. Hierbij geldt, dat voor de berekening van x_{i+1} worden gebruikt de laatste iterate x_i en een vrij oude iterate, die onder omstandigheden (zoals in bovenstaand voorbeeld) constant blijft en die we voor het gemak x_0 noemen. Noemen we het nulpunt α en ontwikkelen we f in een Taylorreeks om α, dan krijgen we

$$x_{i+1} - \alpha = x_i - \alpha - \frac{x_i - x_0}{f_i - f_0} f_i$$

$$= x_i - \alpha - \frac{(x_0 - x_i)(f'(\alpha)(x_i - \alpha) + \ldots)}{f_0 - f'(\alpha)(x_i - \alpha) + \ldots}$$

$$= (x_i - \alpha)\left[1 - \frac{(x_0 - \alpha)f'(\alpha)}{f_0} + O(x_i - \alpha)^2\right].$$

Dus voor zo grote i, dat $\delta_i = x_i - \alpha$ klein genoeg is geldt:

$$\delta_{i+1} \approx \delta_i\left(1 - \frac{(x_0 - \alpha)f'(\alpha)}{f_0}\right).$$

Dit proces convergeert dus lineair en soms nog langzamer dan bisectie.

In bovenstaand voorbeeld zou de convergentie-factor worden

$$1 - \frac{(x_0 - 3)f'(3)}{f_0} = 2/3.$$

Nu beschouwen we de eerste geschetste gevaarlijke methode, waarbij in elke stap de twee nieuwste iterates worden gebruikt en vragen: "Als het proces convergeert, hoe snel dan?" We zetten de bovengenoemde Taylor-reeks nog een term verder voort en krijgen:

$$x_{i+1} - \alpha = x_i - \alpha - \frac{x_i - x_{i-1}}{f_i - f_{i-1}} f_i$$

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

$$= (x_i - \alpha) \frac{\frac{1}{2}f''(\alpha)(x_{i-1} - \alpha) + \circ\circ\circ}{f'(\alpha) + \circ\circ\circ} .$$

Ofwel, als i groot genoeg is en $f'(\alpha) \neq 0$ (m.a.w. α is een enkelvoudige wortel):

$$\delta_{i+1} \approx \frac{f''(\alpha)}{2f'(\alpha)} \delta_i \delta_{i-1} .$$

Om de orde m van het proces te bepalen, stellen we $\delta_{i+1} = C \delta_i^m$ en nemen de logarithme:

$$\ln C + m \ln \delta_i = \ln\left(\frac{f''(\alpha)}{2f'(\alpha)}\right) + \ln \delta_i - \frac{1}{m} \ln C + \frac{1}{m} \ln \delta_i .$$

Omdat $\ln \delta_i$ naar $-\infty$ gaat, mogen we alle constante termen verwaarlozen:
$m \ln \delta_i \approx \ln \delta_i + \frac{1}{m} \ln \delta_i$.
Ofwel $m^2 - m - 1 = 0$, dus $m = \frac{1}{2} \pm \frac{1}{2}\sqrt{5}$.
Als het proces convergeert, is alleen het + teken bruikbaar en we vinden dus voor de orde van dit proces:

$$m = \frac{1 + \sqrt{5}}{2} \approx 1.6 .$$

Combinatie van veiligheid en snelheid

Gelukkig is het mogelijk de lineaire interpolatie zodanig uit te voeren, dat zij veilig is en op den duur convergeert met orde 1.6. Bij het handrekenen vindt men meestal wel de juiste tactiek, maar deze tactiek te programmeren is lastiger.

Wij veronderstellen, dat er twee startwaarden x_0 en x_1 zijn waarvoor geldt $f(x_0) \circ f(x_1) < 0$. In elke iteratie-stap bewaren we 3 punten, nl. a = de voorlaatste, b = de laatste iterate en c = het laatste contra-punt, dat is de laatste iterate, waarvoor $f(c) \circ f(b) < 0$.
Het veilige interval is dus (c,b). Voor de interpolatie worden gebruikt a en b. Ligt het resultaat niet tussen c en b, dan wordt dit verworpen

en dan kan men strict interpoleren tussen c en b of, nog beter, gewoon het gemiddelde van c en b nemen.

Tenslotte nog een kleinigheid. We moeten constateren, dat de gewenste precisie bereikt is. Bij het handrekenen ziet men dit wel. In een machine-programma moeten we, als de iterates te dicht bij elkaar komen, een stapje ε opzij gaan, om de teken-omslag te constateren.
Een en ander is beschreven in de Algol-procedure AP 230, van het Mathematisch Centrum (zie onder).

Voorbeeld. $x = \sqrt[3]{2}$ uit de vergelijking $x^3 - 2 = 0$.

x	f	Δx
1	−1	
1.5	+1.375	−.29
1.21	−.22844	+.0413
1.2513	−.040775	+.00897
1.2603	+.00180518	−.0003816
1.2599184		

Vergelijk $\sqrt[3]{2} \approx 1.2599210$.

```
comment                AP 230
ZERO:= x:= a zero of fx between a and b. The expression fx
must depend on x and have different signs for x = a and x = b.
In array e[1 : 2] one must give the relative tolerance e[1] and the
absolute tolerance e[2], both of which must be positive. The method is
a combination of linear inter- and extrapolation and bisection, pro-
ceeding as follows: Starting from the interval (a, b), ZERO constructs
a sequence of shrinking intervals (c, x), each interval having the
property that fx has different signs in its end points. If necessary,
c and x are interchanged, in order to ensure that fx has the smaller
absolute value in x. Subsequently, either interpolation using c and x
or extrapolation using x and a point outside (c, x) takes place, yiel-
ding a new iterate i. If abs (i - x) is too small, i is moved slightly
towards c. Furthermore, the new iterate is accepted only if it is si-
tuated in the x-half of (c,x), otherwise it is replaced by the middle
m of the interval. The process ends as soon as the interval (c, x)
has a length ≤ 2 × (abs (x × e[1]) + e[2]). For a simple zero this
process is of order 1.6;

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;
```

Opgaven

85 a) Schrijf een procedure of functie-procedure in ALGOL 60, die een nulpunt van een functie bepaalt door middel van bisectie in een voorgeschreven precisie. De heading zou kunnen luiden:
"real procedure BISEC (a, b, f, eps, alarm);
value a, b eps; real a, b, eps; label alarm; real procedure f;"

b) Schrijf hieromheen een programma, dat met behulp van deze procedure enige nulpunten berekent in 3 decimalen nauwkeurig. Kies zelf een geschikte vergelijking en bijbehorende a en b, bijv. de vergelijking tg(x) = x.

86) Bereken met behulp van lineaire interpolatie in 5 decimalen de reële wortels van

$$x^3 - 2x - 5 = 0$$

$$x^2 - 3x - 4 \sin^2 x = 0$$

$$x^3 - 9x^2 + 18x - 6 = 0$$

$$x^5 - 5 = 0.$$

3. Formule van Newton

Bij gebruik van deze formule gaat men uit van een schatting x_0 van het nulpunt, trekt de raaklijn aan de kromme en snijdt deze met de x-as. Het snijpunt is de volgende schatting en men itereert dit.
De iteratie-formule luidt dus

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

3.1 Voorbeeld, nogmaals de vergelijking $x^3 - 2 = 0$

x	f(x)	f'(x)	Δx
1	-1	3	$+\frac{1}{3}$
1.3	+.197	5.07	-.039
1.261	+.00514258	4.77036	-.001078
1.259922			

Vgl. $\sqrt[3]{2} = 1.2599210$.

Ook Newton-iteratie convergeert niet altijd. Zelfs bij polynomen kan het voorkomen, dat men in het oneindige terecht komt of in een cyclus (bijv. $x_{i+2} = x_i$) blijft hangen. Convergentie is verzekerd als tussen nulpunt en schatting de functie convex is (d.w.z. met de bolle kant naar de x-as gekeerd). Dit is het geval als tussen nulpunt en schatting $f(x) \cdot f''(x) > 0$. (Bijvoorbeeld geldt dit voor polynomen met uitsluitend reële nulpunten, met schatting groter dan het grootste of kleiner dan het kleinste nulpunt.)

3.2 Convergentie-snelheid

We ontwikkelen f weer in een Taylorreeks om het gezochte nulpunt α.

$$f(x) = f'(\alpha)(x - \alpha) + \frac{1}{2} f''(\alpha)(x - \alpha)^2 + \ldots$$

$$f'(x) = f'(\alpha) + f''(\alpha)(x - \alpha) + \ldots .$$

Dus

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

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

Stel nu $f'(\alpha) \neq 0$, m.a.w. α zij een enkelvoudig nulpunt. Dan

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

m.a.w. $\delta_{i+1} \approx C\, \delta_i^2$, dus voor een enkelvoudig nulpunt is Newton's iteratie-proces kwadratisch convergent. (Bij elke stap wordt het aantal goede cijfers ongeveer verdubbeld.)
Stel vervolgens $f'(\alpha) = 0$, maar $f''(\alpha) \neq 0$, m.a.w. α zij een dubbel nulpunt. Dan geldt:

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

m.a.w. $\delta_{i+1} \approx \frac{1}{2} \delta_i$, dus nu is het proces lineair convergent met factor $\frac{1}{2}$.

<u>Vergelijking lineaire interpolatie met Newton</u>

Vergelijken we nu deze twee processen voor het vinden van een enkelvoudig nulpunt. Newton's proces is van de orde 2 en lineaire interpolatie van de orde 1.6. Wat praktisch het gunstigste is, hangt echter af van de hoeveelheid rekenwerk, nodig voor het berekenen van $f'(x)$. Er zijn twee belangrijke gevallen:
1e) De berekening van $f'(x)$ kost vrijwel niets (bv. als f een oplossing is van een differentiaalvergelijking). Dan is Newton's proces gunstiger, omdat dit proces van hogere orde is.
2e) De berekening van $f'(x)$ kost vrijwel evenveel als de berekening van $f(x)$. (Dit is bv. het geval bij polynomen van niet al te lage graad.)

Noemen we de berekening van een waarde $f(x)$ of $f'(x)$ een evaluatie, dan vergt lineaire interpolatie dus één evaluatie per stap en Newton twee. Per evaluatie is Newton dus effectief van de orde $\sqrt{2} \approx 1.4$ dus iets minder snel, dan lineaire interpolatie.

3.3 Combinatie van veiligheid en snelheid

Ook Newton's formule kan op een veilige en snelle wijze worden uitgevoerd, indien twee punten bekend zijn, waar de functie tegengesteld teken heeft. Men moet dan, evenals boven geschetst voor lineaire interpolatie, een veilig interval bijhouden en elke Newton-stap alleen accepteren als zij een iterate binnen het veilige interval levert. Zo niet, dan kan men weer het gemiddelde nemen (zie opgave 89).

4. Inverse interpolatie van getabelleerde functies

Voor een nulpunt van een functie f geldt $f(z) = 0$, m.a.w. $z = f^{-1}(0)$. We kunnen f^{-1} door een polynoom benaderen en interpoleren. Bij argument 0 vinden we dan een benadering van een nulpunt z.
Als f door een equidistante tabel gegeven is, is dit niet aantrekkelijk 1e) omdat we dan alle voordelen van de equidistantie kwijt zijn, 2e) omdat soms f^{-1} veel minder op een polynoom lijkt dan f, zodat geen redelijke precisie verkregen wordt (zie ook blz. 60). Beter is het de getabelleerde functie f zelf door een polynoom te benaderen en van dit polynoom een nulpunt te bepalen, bijvoorbeeld met regula falsi. Natuurlijk starten we hier met een tabel-interval, waar de functie teken-omslag vertoont.

4.1 Voorbeeld formule van Bessel

$$f^{**}(x_0 + ph) = f_0 + p\delta_{\frac{1}{2}} + B_2(p)(\delta_0^2 + \delta_1^2) + B_3(p)\delta_{\frac{1}{2}}^3 + \ldots .$$

Wij zoeken p zó, dat $f^{**}(x_0 + ph) = 0$. Laten x_i en f_i tabel-argumenten en -waarden aanduiden, waar p_i de successieve iterates zijn met de startwaarden $p_0 = 1$ en $p_1 = 0$. Lineaire interpolatie levert

$$p_2 = -f_0/\delta_{\frac{1}{2}}.$$

Vervolgens berekenen we $f^{**}(x_0 + p_2 h)$ en zetten de volgende stap. Het tabel-interval is vaak zo fijn, dat we de helling van de interpolatielijn constant gelijk aan $1/\delta_{\frac{1}{2}}$ mogen houden.

Dus

$$p_{i+1} = p_i - \frac{f^{*}(x_0 + p_i h)}{\delta_{\frac{1}{2}}}$$

$$= \frac{-f_0 - B_2(p_i)(\delta_0^2 + \delta_1^2) - B_3(p_i)\delta_{\frac{1}{2}}^3 - \ldots}{\delta_{\frac{1}{2}}} .$$

Men zal zien, dat na een paar iteraties de bijdrage

$$B_2(p_i)(\delta_0^2 + \delta_1^2) + B_3(p_i)\delta_{\frac{1}{2}}^3 + \ldots$$

niet meer verandert: men heeft dan p gevonden.
(Het proces convergeert lineair, maar snel, want de convergentie-factor is klein.)

<u>Bij voorbeeld</u> vierde nulpunt van Besselfunctie $J_0(x)$

x	$J_0(x)$	δ	δ^2	δ^3	δ^4
11.4	-.09021450				
11.5	-.06765395				
11.6	-.04461567				
11.7	-.02133128		+ 1406		+ 181
		+ 2329845		- 23024	
11.8	+.00196717		-21618		+ 402
11.9	+.02504944				
12.0	+.04768931				
12.1	+.06966677				

De hogere differenties blijken (voor Bessel's interpolatie) verwaarloosbaar.

$p_2 = -f_0/\delta_{\frac{1}{2}} \approx .02133/.02330 \approx .9155$

$B_2(p_2)(\delta_0^2 + \delta_1^2) + \ldots \approx .00000516$

Dit levert $p_3 \approx .9153450$. Hierna nog slechts een subtiele verandering:

$B_2(p_3)(\delta_0^2 + \delta_1^2) + \ldots \approx .00000515$

$p_4 = .91534458$

Vergelijk vierde nulpunt van $J_0 \approx 11.7915344391$.

Opgaven

87) a) Hoe luidt de formule van Newton voor de vergelijking

$$x^r - a = 0?$$

Toon aan dat de formule convergeert, als $r \neq 0$, $a > 0$ en $x_0 > 0$.
Noem gevallen, waarin geen convergentie optreedt.

b) Belangrijke bijzondere gevallen zijn $r = 2$ en $r = -1$. Laat zien, dat de formule voor $r = 2$ equivalent is met de formule, behandeld in voorbeeld (0.1).
Bereken met de formule voor $r = -1$ ("delen zonder delen") het quotiënt 1/.25 uitgaande van de startwaarde $x_0 = 1$.

88) Schrijf een programma, dat met behulp van procedure ZERO (pag. 158) een nulpunt bepaalt van een polynoom van oneven graad. Laat de tolerantie zijn een gegeven ε maal de som der absolute waarden der polynoom-coëfficiënten. De invoer-gegevens voor dit programma zijn dus: de precisie ε en de graad en coëfficiënten van het polynoom.
Stel de gegevens op voor de polynomen van opgave 86 en voor het polynoom

$$x^5 - x^4 - x^3 - x^2 - x - 1.$$

Doe de berekening tevens met de hand om de resultaten van hand- en machine-berekening te kunnen vergelijken. Laat het programma hiertoe de nodige tussen-resultaten uittypen, om de stappen te kunnen vergelijken.

89) a) Schrijf een Algol-procedure, dat het procédé, beschreven in (3.3), uitvoert en eindigt met een interval ter lengte 2 × tol (vergelijk procedure ZERO, pag. 158). De heading zou kunnen luiden:

```
"real procedure Newton (a, b, f, Df, e);
value a, b; real a, b; array e; real procedure f, Df;
```

b) Schrijf hieromheen een programma, dat met deze procedure een nulpunt berekent van $x^r = a$ voor enige waarden van a en r in 5 cijfers nauwkeurig.

90) Bereken de reële wortels van de vergelijkingen genoemd in opgave 86, maar nu met de formule van Newton.

5. Inverse interpolatie-formule van Muller

Bij lineaire interpolatie, wordt de gegeven functie f in elke stap benaderd door een polynoom van de eerste graad. Hier gaan we uit van een benaderend polynoom van de graad twee. Laten gegeven zijn 3 basispunten x_0, x_1, x_2. De interpolatie-formule van Newton op deze basispunten, evenwel toegevoegd in omgekeerde volgorde, luidt:

$$f^{**}(x) = f[x_2] + f[x_1,x_2](x - x_2) + f[x_0,x_1,x_2](x - x_2)(x - x_1).$$

Van deze f^{**} gaan we nu een nulpunt x_3 berekenen.
Stel ter vereenvoudiging

$$h_i = x_{i+1} - x_i,\ \Delta_1 = f_{i+1} - f_i,\ d_i = h_i/h_{i-1}.$$

Dan krijgen we:

5.0.1 $$f^{**}(x_3) = f_2 + \frac{\Delta_1}{h_1} h_2 + \left(\frac{\Delta_1}{h_1} - \frac{\Delta_0}{h_0}\right) \frac{h_2(h_2 + h_1)}{h_0 + h_1} = 0$$

Vermenigvuldiging met $\frac{h_0 + h_1}{h_0} = 1 + d_1$ levert:

5.0.2 $$(1 + d_1)f_2 + \left[(1 + 2d_1)\Delta_1 - d_1^2\Delta_0\right]d_2 + (d_1\Delta_1 - d_1^2\Delta_0)d_2^2 = 0$$

Dit is een vierkantsvergelijking in d_2. Om convergentie te krijgen, kunnen we het beste de absoluut kleinste wortel kiezen.

5.1. Intermezzo over de vierkantsvergelijking

De vierkantsvergelijking $ax^2 + bx + c = 0$ heeft twee wortels volgens de bekende formule

$$x = -\frac{b \pm \sqrt{b^2 - 4ac}}{2a}$$

Wanneer $|ac| << |b^2|$, dan zullen voor de absoluut kleinste wortel cijfers in de teller wegvallen. Om dit te vermijden, berekent men eerst de absoluut grootste wortel x_1, volgens bovenstaande formule. Dus, als a, b, c reëel zijn:

5.1.1 $$x_1 = -\frac{b + \mathrm{sgn}(b)\sqrt{b^2 - 4ac}}{2a},$$

waarbij sgn(b) = 1 als $b \geq 0$ en -1 als $b < 0$.

Vervolgens berekent men de andere wortel x_2 door x_1 op het product te delen:

5.1.2 $$x_2 = c/a/x_1.$$

Is men alleen geïnteresseerd in de absoluut kleinste wortel x_2, dan berekent men deze direct als volgt:

5.1.3 $$x_2 = -\frac{2c}{b + \mathrm{sgn}(b)\sqrt{b^2 - 4ac}}$$

5.1.4 Complexe arithmetiek

Zijn a, b, c niet reëel, maar complex, dan kiest men het teken van $r = \sqrt{b^2 - 4ac}$ zó, dat in het complexe vlak de vectoren horende bij b en r een niet stompe hoek maken.

5.2. Formule van Muller (vervolg)

Van vergelijking (5.0.2) willen we alleen de absoluut kleinste wortel hebben. Deze berekenen we daarom volgens formule (5.1.3) (althans zolang de zaak reëel blijft):

5.2.1 $$d_2 = -\frac{2(1 + d_1)f_2}{b + \mathrm{sgn}(b)\sqrt{b^2 - 4(1 + d_1)(d_1\Delta_1 - d_1^2\Delta_0)f_2}}$$

waarbij $b = (1 + 2d_1)\Delta_1 - d_1^2\Delta_0$.

Dit is de formule van Muller.

In het geval van complexe arithmetiek wijzigt zich deze formule in zoverre dat het teken van de wortelvorm wordt gekozen volgens opmerking (5.1.4).

Uit deze d_2 worden dan h_2 en x_3 verkregen volgens:

$$h_2 = d_2 \times h_1,\ x_3 = h_2 + x_2.$$

Vervolgens worden x_0, x_1, x_2 vervangen door resp. x_1, x_2, x_3, de functie-waarde in het nieuwe punt x_2 wordt uitgerekend en dan is men gereed voor de volgende iteratie-stap.

5.3. Convergentie-snelheid

Muller's proces convergeert niet altijd. Noemen we de fout weer $\delta_i = x_i - \alpha$, waarbij α een nulpunt is. Op analoge wijze als in 2.3, vinden we nu het volgende.

Indien Muller's proces convergeert naar een enkelvoudig nulpunt α, dan geldt op den duur

$$\delta_{i+1} \approx - \frac{f'''(\alpha)}{6f'(\alpha)} \delta_i \delta_{i-1} \delta_{i-2}.$$

De convergentie-orde m is gelijk aan de grootste wortel van de vergelijking $m^3 - m^2 - m - 1 = 0$, dus $m \approx 1.84$.

5.4. Voordelen van Muller's methode

1) Geen afgeleiden van f nodig (evenals lineaire interpolatie).
2) Hoge convergentie-orde.
3) Uitgaande van reele start-waarden duikt dit proces op natuurlijke wijze in het complexe vlak.
   In tegenstelling tot lineaire interpolatie en Newton, kan deze formule worden gebruikt om complexe nulpunten te bepalen.

Opgaven.

91) De confluente formule van Muller gaat uit van drie samen-vallende basispunten (vgl. pag. 64). M.a.w. f^{**} is het quadratische polynoom, dat in één punt x_i met f functie-waarde en 1^e en 2^e afgeleide gemeen heeft. Leidt voor dit geval af de formule voor x_{i+1} zijnde het dichtst bijzijnde nulpunt van f^{**}.

92) Bereken uit de volgende tabel zo nauwkeurig mogelijk het nulpunt

x	$J_0(x)$	
2.0	+.22389	07791
2.1	+.16660	69803
2.2	+.11036	22669
2.3	+.05553	97844
2.4	+.00250	76833
2.5	-.04838	37765
2.6	-.09680	49544
2.7	-.14244	93700
2.8	-.18503	60334
2.9	-.22431	15458

93) Zij gegeven de volgende tabel van $\Gamma(x)$ en van $\psi(x) = \frac{d}{dx} \ln(\Gamma(x))$

x	$\Gamma(x)$		$\psi(x)$	
1.445	.88572	23397	-.01621	81479
1.450	.88566	13803	-.01131	64226
1.455	.88562	20800	-.00643	72934
1.460	.88560	43364	-.00158	05620
1.465	.88560	80495	+.00325	39677
1.470	.88563	31217	+.00806	64890
1.475	.88567	94575	+.01285	71930
1.480	.88574	69646	+.01762	62684

Bepaal uit deze tabel zo nauwkeurig mogelijk het nulpunt s van $\psi(x)$ en het bijbehorende minimum $\Gamma(s)$ van de Γ-functie.

94) Schrijf procedures ter berekening van

a) het product van twee complexe getallen,

b) het quotiënt van twee complexe getallen,

c) de vierkantswortel uit een complex getal.

Schrijf hieromheen een programma, dat de wortels van een vierkantsvergelijking met complexe coëfficiënten berekent. De coëfficiënten worden ingelezen van de band (bv.: "a1 := read"). De gevonden wortels worden geponst op een band (bv. met "punch (x1)").

6. Twee vergelijkingen met twee onbekenden

Van de boven besproken formules laat die van Newton zich het gemakkelijkst uitbreiden tot een stelsel van n vergelijkingen met n onbekenden. We beperken ons hier tot n = 2.

Zij gevraagd x en y zó, dat

$$f(x,y) = 0 \quad \text{en} \quad g(x,y) = 0.$$

De Taylor-reeksen van f en g om het punt (x_i, y_i) luiden, als $\Delta x = x - x_i$ en $\Delta y = y - y_i$, aldus:

$$f(x,y) \approx f(x_i,y_i) + \frac{\partial f}{\partial x_i}\Delta x + \frac{\partial f}{\partial y_i}\Delta y + \ldots = 0,$$

$$g(x,y) \approx g(x_i,y_i) + \frac{\partial g}{\partial x_i}\Delta x + \frac{\partial g}{\partial y_i}\Delta y + \ldots = 0.$$

We verwaarlozen de termen van hogere orde, m.a.w. f en g worden vervangen door lineaire functies (men spreekt ook van "linearizeren" van het probleem). Zo krijgen we het volgende lineaire stelsel

$$\frac{\partial f}{\partial x_i}\Delta x + \frac{\partial f}{\partial y_i}\Delta y = -f(x_i,y_i)$$

$$\frac{\partial g}{\partial x_i}\Delta x + \frac{\partial g}{\partial y_i}\Delta y = -g(x_i,y_i)$$

De matrix J van dit stelsel heet matrix van Jacobi.
Schrijven we $f_i = f(x_i,y_i)$, $g_i = g(x_i,y_i)$ en

$$D = \det(J) = \frac{\partial f}{\partial x_i} \cdot \frac{\partial g}{\partial y_i} - \frac{\partial f}{\partial y_i} \cdot \frac{\partial g}{\partial x_i},$$

dan luidt de oplossing

$$\Delta x = (g_i \frac{\partial f}{\partial y_i} - f_i \frac{\partial g}{\partial y_i})/D$$

$$\Delta y = (f_i \frac{\partial g}{\partial x_i} - g_i \frac{\partial f}{\partial x_i})/D.$$

Met deze Δx en Δy verkrijgt men nieuwe iterates $x_{i+1} = x_i + \Delta x$, $y_{i+1} = y_i + \Delta y$, waarmee men het proces herhaalt. Men start met gekozen startwaarden x_0, y_0.
Evenals Newton voor een enkele vergelijking, convergeert dit proces niet altijd, maar indien het convergeert, is de convergentie-orde quadratisch.

7. Wortels van Algebraïsche Vergelijkingen

Elk der bovengenoemde methodes is bruikbaar voor algebraïsche vergelijkingen, waar de functie in kwestie dus een polynoom is. Aangezien deze methodes in elke stap een functie-waarde, en Newton bovendien de waarde van de afgeleide, behoeven, zullen we eerst nagaan hoe de waarde van een polynoom en zijn afgeleiden berekend worden.

7.1. Rekenwijze van Horner

Zij gevraagd de waarde voor $x = p$ van een n-de graads polynoom en zijn afgeleiden, als het polynoom wordt gegeven door

7.1.0. $A(x) = a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}x + a_n$.

We berekenen achtereenvolgens voor $i = 0(1)n$:

7.1.1. $b_i = a_i + pb_{i-1}$,

waarbij per definitie $b_{-1} = 0$, dus $b_0 = a_0$.
Deze rekenwijze levert niet alleen de functie-waarde, n.l.:

$$b_n = A(p),$$

maar bovendien geldt:

$$A(x) = (x - p)\,B(x) + b_n,$$

waarbij

$$B(x) = b_0x^{n-1} + \ldots + b_{n-2}x + b_{n-1}.$$

Om de afgeleide te krijgen, doen we hetzelfde met $B(x)$, m.a.w. we berekenen $c_i = b_i + pc_{i-1}$ voor $i = 0(1)n-1$, wederom per definitie stellend $c_{-1} = 0$. Dan vinden we dus

$$B(x) = (x - p)C(x) + c_{n-1},$$

waarbij

$$C(x) = c_0x^{n-2} + \ldots + c_{n-2}.$$

Blijkbaar geldt nu $c_{n-1} = A'(p)$.
Berekenen we vervolgens op dezelfde wijze coëfficiënten d_i, dan vinden we $d_{n-2} = \frac{1}{2}A''(p)$.
Deze rekenwijze heet rekenwijze van Horner.

Men zet de optredende getallen aldus in schema:

$$
\begin{array}{llll}
a_0 & = b_0 & = c_0 & = d_0 \\
a_1 & & b_1 & c_1 & d_1 \\
a_2 & & b_2 & c_2 & d_2 \\
\circ & & \circ & \circ & \circ \\
\circ & & \circ & \circ & \circ \\
a_{n-2} & & b_{n-2} & c_{n-2} & d_{n-2} = \tfrac{1}{2}A''(p) \\
a_{n-1} & & b_{n-1} & c_{n-1} = A'(p) & \\
a_n & & b_n = A(p) & &
\end{array}
$$

Bouwen we het schema verder naar rechts uit, dan vinden we in feite de coëfficiënten van de Taylorontwikkeling van het polynoom A(x) in het punt x = p:

$$A(x) = A(p) + A'(p)(x - p) + \tfrac{1}{2}A''(p)(x - p)^2 + \circ\circ\circ + \frac{1}{n!}A^{(n)}(p)(x - p)^n$$

$$= b_n + c_{n-1}(x - p) + d_{n-2}(x - p)^2 + \circ\circ\circ + a_0(x - p)^n.$$

Opgaven:

95) Bereken zo nauwkeurig mogelijk het nulpunt uit de volgende tabel (uit N.B.S.-A.M.S. 55 p. 494).

x	f(x)
2.3	+.09408798
2.4	+.07218365
2.5	+.05158229
2.6	+.03235987
2.7	+.01457248
2.8	-.00174144
2.9	-.01655931
3.0	-.02987272
3.1	-.04168613
3.2	-.05201554

96) Bepaal met de regel van Newton het reële nulpunt van $f(x) = x + \ln(x)$.

97) Bepaal de reële nulpunten van het polynoom
$x^5 - 9x^4 + 42x^3 - 66x^2 - 43x + 75$.

7.2. Reele nulpunten van een polynoom

Voor het vinden van een reëel nulpunt tracht men eerst een interval te vinden, waar de functie van teken omslaat. Is het polynoom van oneven graad, dan lukt dit gemakkelijk, nl. voor een zeer groot getal m geldt $f(m) \times f(-m) < 0$.
Bij het handrekenen probeert men enige gehele argument-waarden, totdat tekenomslag wordt gevonden tussen twee opeenvolgende gehele getallen.
In een programma gaan we liefst uit van een niet al te grote m. Men kan als m nemen een bovengrens voor de absolute waarde van alle wortels. Een geschikte bovengrens levert de volgende
Stelling. Zij gegeven een polynoom A(x) volgens 7.1.0, waarbij $a_0 \neq 0$. Zij

7.2.0. $$v = \max_{k=1,\ldots,n} \left| \frac{a_k}{a_0} \right|^{\frac{1}{k}} .$$

Dan zijn alle nulpunten van A(x) in absolute waarde kleiner dan 2v. Bovendien is het absoluut grootste nulpunt in absolute waarde minstens v/n.
Voor een polynoom van oneven graad kunnen we dus uitgaan van de bovengrens m = 2v en de veilige lineaire interpolatie toepassen met het veilige begin-interval $[-m, m]$.
Voor een polynoom van even graad kan het moeilijk, of zelfs onmogelijk, zijn een interval te vinden waar tekenomslag optreedt. Met name de gevallen: 2 bijna samenvallende reele wortels, 2 samenvallende reele wortels, 2 bijna samenvallende toegevoegd complexe wortels, zijn moeilijk te onderscheiden en kunnen door futiele wijziging der coëfficiënten in elkaar overgaan.
Vindt men geen tekenomslag, dan zit er niets anders op, dan uitgaande van gekozen benodigde startgegevens een der bovengenoemde iteratieprocessen te proberen en te hopen dat dit convergeert.
Heeft men een wortel gevonden, dan kan men de factor x-p uitdelen met het Horner-schema (zie 7.1) en met het quotiënt B(x) de bereke-

ning voortzetten. Dit kan men herhalen totdat alle wortels gevonden zijn. Om voldoende precisie voor alle wortels te krijgen, is het van belang dit uitdelen in extra precisie te doen, waarbij ook de gebruikte p extra precisie moet hebben. Bij hoge graads polynomen is het bovendien raadzaam de aldus gevonden wortels nog te controleren en te verbeteren (bv. met een of meer Newton-slagen) aan de hand van het oorspronkelijke polynoom.

7.3. Horner schema voor complex argument

Voeren we de Horner rekenwijze (7.1) uit voor complex argument $u + vi$, dan vergt dit $4n$ vermenigvuldigingen. Als het polynoom reele coëfficiënten heeft, kan dit tot $2n$ gereduceerd worden. Hiertoe delen we gelijk met de factor $x - u - vi$ de toegevoegd complexe factor uit. Hun product is reeel en heeft de gedaante

$$(x - u - vi)(x - u + vi) = (x^2 - 2ux + u^2 + v^2) =$$
$$= x^2 - px - q,$$

waarbij we nu stellen $p = 2u$ en $q = -(u^2 + v^2)$.
Uitdelen van deze quadratische factor levert een rest van de graad één, dus we mogen stellen:

$$7.3.1. \quad A(x) = (x^2 - px - q)B(x) + Rx + S,$$

waarbij $B(x) = b_0x^{n-2} + b_1x^{n-3} + \ldots + b_{n-3}x + b_{n-2}$.
Hieruit volgt

$$7.3.2. \begin{cases} b_0 = a_0 \\ b_1 = a_1 + pb_0 \\ b_i = a_i + pb_{i-1} + qb_{i-2} \quad \text{voor} \quad i > 1. \end{cases}$$

Stellen we $b_{-1} = b_{-2} = 0$, dan geldt de b_i-formule ook voor $i = 0$ en 1. Voor $i = n - 1$ levert deze formule juist R, m.a.w.

$$7.3.3. \quad R = b_{n-1} = a_{n-1} + pb_{n-2} + qb_{n-3}$$

en tenslotte

7.3.4. $S = a_n + qb_{n-2}$.

De functie-waarde voor het argument u + vi is dan blijkbaar

7.3.5. $A(u + vi) = Ru + S + Rvi$.

7.4. Complexe nulpunten van een polynoom

Men kan complexe nulpunten berekenen met behulp van de formule van Muller (zie sectie 5), waarbij de functie-waarden voor complexe iteraties worden berekend volgens het complexe Horner-schema (7.3). Deze methode leent zich goed voor automatische berekening, ofschoon convergentie niet is te garanderen. Het is dus zaak, als de gewenste precisie niet bereikt wordt, na een beperkt aantal iteraties het proces te onderbreken.
Het is bovendien van belang iteraties met een al te grote absolute waarde (bv. iteraties > 2v vgl. stelling (7.2.0)) te verwerpen en door iets geschikts te vervangen.
Van de formule van Muller is onlangs een veel eenvoudiger variant van Traub bekend geworden.
Om deze af te leiden gaan we weer uit van Newton's interpolatie-formule op de basispunten x_2, x_1, x_0 en berekenen hiervan een nulpunt x_3. Stellen we weer $h_i = x_{i+1} - x_i$, dan krijgen we (vgl. 5.0.1):

7.4.1. $f^{**}(x_3) = f[x_2] + f[x_1,x_2]h_2 + f[x_0,x_1,x_2]h_2(h_2 + h_1) = 0$

Dit is een vierkantsvergelijking in h_2. De coëfficiënt van h_2 luidt $p = f[x_1,x_2] + f[x_0,x_1,x_2]h_1$ en de kleinste wortel h_2 is (vgl. 5.1):

7.4.2. $$h_2 = -\frac{2f[x_2]}{p \pm \sqrt{p^2 - 4f[x_0,x_1,x_2]\ f[x_2]}}$$

waarbij het teken van de wortelvorm, die we r noemen, zo gekozen wordt, dat p en r een niet-stompe hoek maken.

7.5. Methode van Bairstow.

Deze methode is speciaal voor polynomen, en geschikt zowel voor handrekenen als voor het rekenen met een computer. We trachten een quadratische factor $x^2 - px - q$ te vinden (p en q reeel), waarbij de wortels van deze factor een reeel paar of een toegevoegd complex paar mogen zijn. Hiertoe schrijven we het polynoom A(x) in de vorm (7.3.1). Nu is $x^2 - p\,x - q$ slechts dan een factor van A(x) als de rest $R\,x + S$ verdwijnt, m.a.w. we moeten p en q zodanig bepalen dat R = S = 0. Nu zijn R en S functies van p en q en het probleem is dus herleid tot het oplossen van het stelsel vergelijkingen

$$7.5.1 \qquad \begin{cases} R(p,q) = 0 \\ S(p,q) = 0. \end{cases}$$

De methode van Bairstow bestaat nu uit het oplossen van dit stelsel met de formule van Newton (zie sectie 6).
M.a.w. uitgaande van startwaarden $p = p_o$, $q = q_o$, hebben we in de i-de stap op te lossen het lineaire stelsel

$$7.5.2 \qquad \begin{cases} \dfrac{\partial R}{\partial p}\ \Delta p + \dfrac{\partial R}{\partial q}\ \Delta q = -R \\ \dfrac{\partial S}{\partial p}\ \Delta p + \dfrac{\partial S}{\partial q}\ \Delta q = -S \end{cases}$$

waarbij de waarden van R en S en hun partiele afgeleiden worden genomen voor $p = p_i$ en $q = q_i$.
Om de partiele afgeleiden te vinden, differentieren we eerst formule (7.3.2):

$$7.5.3 \qquad \begin{cases} \dfrac{\partial b_1}{\partial p} = b_o + p\,\dfrac{\partial b_o}{\partial p} = b_o\,, \dfrac{\partial b_1}{\partial q} = \dfrac{\partial b_o}{\partial p} = \dfrac{\partial b_o}{\partial q} = 0 \\ \dfrac{\partial b_i}{\partial p} = b_{i-1} + p\,\dfrac{\partial b_{i-1}}{\partial p} + q\,\dfrac{\partial b_{i-2}}{\partial p}\,, \\ \dfrac{\partial b_i}{\partial q} = b_{i-2} + p\,\dfrac{\partial b_{i-1}}{\partial q} + q\,\dfrac{\partial b_{i-2}}{\partial q} \qquad (i > 1) \end{cases}$$

Vergelijken we deze formule met (7.3.2), dan zien we dat

$\frac{\partial b_i}{\partial p}$ en $\frac{\partial b_i}{\partial q}$ aan dezelfde recurrente betrekking voldoen als b_i mits we hierin a_i vervangen door b_{i-1} respectievelijk b_{i-2}. Daarom passen we het Horner schema nogmaals toe, nu op B(x), en krijgen

$$B(x) = (x^2 - px - q).\ C(x) + c_{n-3}\ x + b_{n-2} + q\ c_{n-4},$$

waarbij $C(x) = c_o\ x^{n-4} + \ldots + c_{n-4}$.

De recurrente relatie voor de c_i luidt nu

7.5.4.
$$\begin{cases} c_o = b_o \\ c_1 = b_1 + p\ c_o \\ c_i = b_i + p\ c_{i-1} + q\ c_{i-2} \quad \text{voor} \quad i > 1. \end{cases}$$

Vergelijking van (7.5.3) en (7.5.4) levert onmiddellijk (mits we $c_{-1} = c_{-2} = 0$ stellen):

$$\frac{\partial b_i}{\partial p} = c_{i-1}\ , \frac{\partial b_i}{\partial q} = c_{i-2}\ .$$

Dus in het bijzonder, wegens $R = b_{n-1}$:

$$\frac{\partial R}{\partial p} = c_{n-2}\ , \frac{\partial R}{\partial q} = c_{n-3}\ .$$

En tenslotte

$$\frac{\partial S}{\partial p} = q\ \frac{\partial b_{n-2}}{\partial p} = q\ c_{n-3}$$

$$\frac{\partial S}{\partial q} = b_{n-2} + q\ \frac{\partial b_{n-2}}{\partial q} = b_{n-2} + q\ c_{n-4}\ .$$

Het op te lossen stelsel (7.5.2) krijgt dus de gedaante:

7.5.5.
$$\begin{cases} c_{n-2}\ \Delta\ p + c_{n-3}\ \Delta\ q = -b_{n-1} \\ q\ c_{n-3}\ \Delta\ p + (b_{n-2} + q\ c_{n-4})\ \Delta\ q = -\ a_n - q\ b_{n-2} \end{cases}$$

Tellen we p x de eerste vergelijking bij de tweede op, dan wordt het rechterlid $-\ b_n$ (volgens formule (7.3.2) met i = n), de coefficient

van Δ q wordt c_{n-2} en de coefficient van Δ p wordt

7.5.6. $$M = p\, c_{n-2} + q\, c_{n-3} \quad (= c_{n-1} - b_{n-1}).$$

Het stelsel luidt dan

7.5.7. $$\begin{cases} c_{n-2}\, \Delta p + c_{n-3}\, \Delta q = -b_{n-1} \\ M\, \Delta p + c_{n-2}\, \Delta q = -b_n \end{cases}$$

De determinant van de matrix is

7.5.8. $$D = c_{n-2}^2 - M\, c_{n-3}$$

en de oplossing luidt

7.5.9. $$\Delta p = (c_{n-3}\, b_n - c_{n-2}\, b_{n-1})/D,$$
$$\Delta q = (M\, b_{n-1} - c_{n-2}\, b_n)/D.$$

Hiermee vindt men een nieuwe schatting

$$p_{i+1} = p_i + \Delta p, \quad q_{i+1} = q_i + \Delta q,$$

waarmee de bewerking wordt herhaald.
Tenzij men iets beters weet, kan men beginnen met $p_o = q_o = 0$.
Convergentie kan ook hier niet worden gegarandeerd. Indien evenwel het proces convergeert, is het quadratisch convergent. Moeilijkheden treden op als D = 0 of zeer klein is.
Heeft men eenmaal een wortelpaar gevonden, dan kan men met quotient B(x) de berekening voortzetten om verdere wortels te vinden.
Bij een dergelijke werkwijze zal men evenwel precisie verliezen. Het is dus raadzaam p en q in extra precisie te berekenen en de laatste complexe Horner ook in extra precisie te doen. Bij hoge graads polynomen doet men goed om ter controle en correctie een aldus gevonden paar p,q als start te nemen voor een paar Bairstow-stappen met het oorspronkelijke polynoom.

Voorbeeld $x^4 - 2x^3 - 4x^2 + 5x - 6$

$p = 0$	$p = 2$	$p = .61$
$q = 0$	$q = -1.5$	$q = -.58$
B C	B C	B C
1 1	1 1	1 1
-2 -2	0 2	-1.3900 -.7800
-4 -4	-5.5 -3	-5.4279 -6.4837
5 M=0	-6 M=-9	+2.4952 M=-3.5027
-6	-9.75	-1.3297
$D = 16$	$D = 27$	$D = 39.31$
$\Delta p = 2$	$\Delta p = -1.39$	$\Delta p = .438$
$\Delta q = -1.5$	$\Delta q = .92$	$\Delta q = -.442$

Hierna vindt men:

p	q
1.048	-1.022
.9985	-. 9991
1.0000	-1.0000

met $b_{1\mathfrak{R}} = 1$ en $b_{2\mathfrak{R}} = 6$.

Dus de 2 quadratische factoren zijn $x^2 - x + 1$ en $x^2 - x - 6$ en de wortels zijn + 3, - 2 en $(1 \pm i\sqrt{3})/2$.

Opgaven

98) Bepaal in 6 decimalen alle nulpunten van

a) $32x^3 - 48x^2 + 18x - 1$

b) $x^4 - 16x^3 + 72x^2 - 96x + 24$

99) Bepaal met de methode van Bairstow in 5 decimalen de nulpunten van:

a) $x^4 - 2x^3 + 7x^2 + x + 1$

b) $x^6 + 8\,x^5 + 24\,x^4 + 36\,x^3 + 72\,x^2 + 36\,x + 30$

100) a) Schrijf een Algol procedure, die de waarde van een polynoom met reele coefficienten berekent voor complex argument met behulp van het complexe Horner-schema (7.3).

b) Schrijf een Algol procedure, die een paar nulpunten van een polynoom bepaalt met de methode van Bairstow.
Deze mag natuurlijk de vorige procedure gebruiken.

101) Schrijf een Algol programma, dat alle nulpunten van een polynoom berekent, waarbij al of niet mag worden aangenomen, dat alle nulpunten reeel zijn.
Stel enige geschikte polynomen op en laat hiervoor het programma uitvoeren.

Hoofdstuk 6. Eigenwaarden en vectoren van matrices

0. Literatuur.

1 National Physical Laboratories, Modern Computing Methods.

22 A.C. Aitken, Determinants and matrices .

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

24 R. Zurmühl, Matrizen.

25 C. Lanczos, Applied analysis (1957).

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

27 V.N. Faddeeva, Computational methods of lineair algebra (1959).

28 D.K. Faddeev & V.N. Faddeeva, Computational methods of linear algebra (1963).

29 J.H. Wilkinson, The algebraic eigenvalue problem (Oxford 1965).

1. Theorie.

Het algebraïsche eigenwaarde-probleem kan als volgt worden geformuleerd. Gegeven een vierkante matrix A van de orde n, zoeken we getallen λ, waarvoor het stelsel lineaire vergelijkingen

1.1 $$Ax = \lambda x$$

een oplossingsvector $x \neq \vec{0}$ heeft. De waarden van λ, die voldoen, heten <u>eigenwaarden</u> van A en een bij zo'n λ horende vector x heet <u>eigenvector</u> van A.

Eigenwaarden

We kunnen het stelsel (1.1.) ook schrijven in de vorm

1.2 $$(A - \lambda I)x = \vec{0} ,$$

waarbij I de eenheidsmatrix van de orde n is. Dit stelsel heeft dan en slechts dan een oplossing $x \neq \vec{0}$ als

1.3 $$\det (A - \lambda I) = 0.$$

Dit is een algebraïsche n^{de}-graads vergelijking. Deze vergelijking heet de karakteristieke vergelijking van A en het polynoom det $(A-\lambda I)$, of ook wel det $(\lambda I-A)$, dat 1 als coëfficiënt van λ^n heeft, heet het karakteristieke polynoom van A.
We beperken ons tot reële matrices. De coëfficiënten van het k... - teristieke polynoom zijn dan kennelijk ook reëel. De eigenwaarden zijn dus reëel of complex en, als ze niet reëel zijn, verschijnen ze als toegevoegd complexe paren.

Eigenvectoren.

Bij elke eigenwaarde hoort minstens één eigenvector en bij een enkelvoudige eigenwaarde (d.i. een enkelvoudige wortel van de karakteristieke vergelijking) hoort ook slechts één eigenvector. Zo'n eigenvector is, wegens de homogeniteit van (1.1.) , op een factor na bepaald. In de praktijk wordt een eigenvector vaak op geschikte wijze genormeerd, bijv. zó dat zijn lengte of zijn absoluut grootste element gelijk aan 1 is. Eigenvectoren horende bij verschillende eigenwaarden zijn lineair onafhankelijk. Zijn alle eigenwaarden $\lambda_1,\ldots,\lambda_n$ verschillend, dan horen daar dus n lineair onafhankelijke eigenvectoren $x_1,\ldots,x_n$ bij, die samen de hele n-dimensionale ruimte opspannen. M.a.w. een willekeurige vector v kan worden geschreven als

$$v = \alpha_1 x_1 + \ldots + \alpha_n x_n.$$

Laat de i^{de} element van eigenvector x_j worden aangeduid met x_{ij}, dan vormen deze elementen samen een matrix $X = \left(x_{ij}\right)$. Als we bovendien de eigenwaarden λ_j, $j=1,\ldots,n$, op de diagonaal van een diagonaal matrix zetten, m.a.w.

$$\Lambda = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{pmatrix} = \operatorname{diag}(\lambda_1,\ldots,\lambda_n),$$

dan krijgt de volledige oplossing van het eigensysteem (1.1.) voor

$\lambda = \lambda_1,\ldots,\lambda_n$ de gedaante

1.4. $AX = X\Lambda$.

Omdat de eigenvectoren x_j lineair onafhankelijk zijn, is X niet-singulier, m.a.w. X^{-1} bestaat. Dus

1.5. $X^{-1}AX = \Lambda$.

Een transformatie, die aan een matrix A toevoegt een matrix $X^{-1}AX$ heet gelijkvormigheids-transformatie en de matrices A en $X^{-1}AX$ heten gelijkvormig, m.a.w.:

Definitie. Twee matrices A en B heten gelijkvormig als er een niet-singuliere matrix X bestaat zó, dat

$B = X^{-1}AX$.

We hebben dus de volgende

Stelling. Als alle eigenwaarden $\lambda_1,\ldots,\lambda_n$ van een matrix A verschillend zijn, dan is A gelijkvormig met de diagonaal-matrix $\Lambda = \text{diag}\,(\lambda_1,\ldots,\lambda_n)$.

Omgekeerd geldt blijkbaar ook de volgende

Stelling. Als A gelijkvormig is met een diagonaalmatrix Λ, m.a.w. voor zekere X geldt $X^{-1}AX = \Lambda$, dan zijn de diagonaal-elementen van Λ de eigenwaarden van A en de kolommen van X zijn de bijbehorende eigenvectoren.

Deze stelling geldt ook, als de diagonaal-elementen van Λ en dus eveneens de eigenwaarden van A niet alle verschillend zijn.

Eigensysteem bij meervoudige eigenwaarden

Bij een meervoudige eigenwaarde horen meestal meer dan één eigenvector. Nu is elke lineaire combinatie van eigenvectoren horende bij eenzelfde eigenwaarde λ eveneens een eigenvector bij λ. De eigenvectoren bij één eigenwaarde λ vormen dus een lineaire deelruimte, de eigenruimte van λ. Laat p de dimensie van deze eigenruimte zijn, dan heeft λ dus precies p lineair onafhankelijke eigenvectoren.
Laat verder λ een m-voudige eigenwaarde zijn. Dan geldt $p \leq m$, m.a.w. de dimensie van een eigenruimte is hoogstens de multipliciteit van de eigenwaarde. Als $p < m$, dan heet de matrix defect, m.a.w.

Definitie. Een matrix heet defect als hij een meervoudige eigenwaarde bezit, wiens multipliciteit groter is dan de dimensie van de bijbehorende eigenruimte.

Bijv. de matrix $\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$ heeft een dubbele eigenwaarde 0, waarbij slechts één lineair onafhankelijke eigenvector $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ hoort. Als de matrix defect is, zijn er blijkbaar te weinig eigenvectoren om de hele n-dimensionale ruimte op te spannen. De matrix is dan niet gelijkvormig met een diagonaal-matrix.
Omdat defecte matrices voor de praktijk minder belangrijk zijn en bovendien numeriek moeilijker te behandelen zijn, zullen wij ons hier beperken tot niet-defecte matrices. Niet-defecte matrices hebben n lineair onafhankelijke eigenvectoren en zijn dus gelijkvormig met een diagonaal-matrix. We noemen ze daarom diagonaliseerbaar, d.w.z.:

Definitie. Een matrix heet diagonaliseerbaar, als zij gelijkvormig is met een diagonaal-matrix.

De begrippen "defect" en "diagonaliseerbaar" zijn dus tegengesteld.
De volledige oplossing van het eigensysteem van een diagonaliseerbare matrix heeft dus de vorm (1.4.) en (1.5.), waarbij nu evenwel de eigenwaarden al of niet verschillend mogen zijn. Bij meervoudige eigenwaarden genieten de eigenvectoren een grotere onbepaaldheid dan

bij enkelvoudige. Niet alleen mogen ze met een willekeurige faktor vermenigvuldigd worden, maar ook mag als verzameling lineair onafhankelijke eigenvectoren bij één meervoudige eigenwaarde elke basis van zijn eigenruimte gekozen worden.
Deze onbepaaldheid moet men bij de numerieke behandeling in de gaten houden.
Numeriek is moeilijk te onderscheiden, of eigenwaarden gelijk zijn of bijna gelijk. Wanneer ze bijna gelijk zijn, dan zijn de bijbehorende eigenvectoren moeilijk te berekenen, omdat ze bijna onbepaald zijn.

Eigenrijen

Naast het stelsel (1.1) is ook van belang het getransponeerde stelsel

1.6 $$A^T y = \lambda y,$$

ook wel geschreven als

1.7 $$y^T A = \lambda y^T$$

Matrix A^T heeft kennelijk dezelfde eigenwaarden als A. De oplossingsvectoren y^T heten eigenrijen van A en ter onderscheiding hiervan heten de eigenvectoren ook eigenkolommen. We hebben dan de volgende
Stelling: Een eigenkolom en een eigenrij horende bij verschillende eigenwaarden staan loodrecht op elkaar.
Als A enkelvoudige eigenwaarden heeft en Y de matrix der eigenvectoren van A^T is, is dus blijkbaar $Y^T X$ een diagonaal-matrix. De eigenkolommen en -rijen worden zó genormeerd dat $Y^T X = I$ is.
Als A meervoudige eigenwaarden heeft en diagonaliseerbaar is, is A^T ook diagonaliseerbaar. Men kiest in dit geval de bases in de eigenruimte en de normering zó, dat $Y^T X = I$. Dan hebben we dus voor diagonaliseerbare matrices:

$$\Lambda = X^{-1} A X = Y^T A X = Y^T A \, (Y^T)^{-1} .$$

we kunnen dus een diagonaliseerbare matrix A schrijven als
$A = X \Lambda X^{-1} = X \Lambda Y^T$, m.a.w.

Stelling: Een diagonaliseerbare matrix A met eigenwaarden λ_i, en bijbehorende eigenkolommen x_i en eigenrijen y_i^T kan worden geschreven in

de gedaante

$$A = \sum_{i=1}^{n} \lambda_i \, x_i \, y_i^T$$

Reële symmetrische matrices

Een belangrijk bijzonder geval zijn de reële symmetrische matrices. Zij komen in de praktijk vaak voor en zijn numeriek prettiger. De eigenwaarden zijn in dit geval altijd reëel. Men normeert de eigenvectoren vaak zó, dat de lengte 1 is. Dit heeft tot gevolg, dat, indien alle eigenwaarden verschillend zijn, de matrix X der eigenvectoren orthogonaal is, d.w.z. $X^T X = I$. Een symmetrische matrix is, ook indien zij meervoudige eigenwaarden heeft, altijd diagonaliseerbaar. Men kiest bij meervoudige eigenwaarden orthogonale bases in de eigenruimtes, zodat de matrix X der eigenvectoren bij normering op lengte 1 ook orthogonaal is.

2. Directe methode

De meest voor de hand liggende methode, die echter numeriek slechts bruikbaar is voor zeer lage orde, is de volgende.
Bereken eerst de coëfficienten van het karakteristieke polynoom

$$2.1 \qquad \det(\lambda I - A) = \lambda^n + c_1 \lambda^{n-1} + \ldots + c_n \,,$$

bereken vervolgens de nulpunten hiervan (zie hoofdstuk 5) en los daarna voor de aldus gevonden eigenwaarden de betreffende stelsels voor de eigenvectoren op.
De coëfficienten vinden we meteen uit (2.1). o.a.:

$$2.2 \qquad \begin{cases} c_1 = -\,\text{spoor}\,(A) = -\sum_{i=1}^{n} a_{ii} \\ c_2 = \sum_{1 \le i < j \le n} \begin{vmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{vmatrix} \\ c_{n-1} = (-1)^{n-1} \sum_{i=1}^{n} m_{ii} \,, \text{ waarbij } m_{ii} = \text{minor van } a_{ii} \\ c_n = (-1)^n \det(A) \end{cases}$$

Voor hogere orde (>5 zeg) wordt de berekening van deze coëfficienten al spoedig tijdrovend. Een veel ernstiger nadeel voor hoge orde is, dat de eigenwaarden vaak veel gevoeliger zijn voor fouten in de coëfficienten c_i , dan voor fouten in de elementen van A.

2.3 Voorbeelden

Voorbeeld 1 uit [25] pag. 65

$$A = \begin{pmatrix} 33 & 16 & 72 \\ -24 & -10 & -57 \\ -8 & -4 & -17 \end{pmatrix}$$

De karakteristieke vergelijking luidt: $\lambda^3 - 6\lambda^2 + 11\lambda - 6 = 0$

De eigenwaarden zijn $\lambda_1 = 1$, $\lambda_2 = 2$, $\lambda_3 = 3$.

De eigenkolommen zijn

$$x_1 = \begin{pmatrix} -15 \\ 12 \\ 4 \end{pmatrix}, \quad x_2 = \begin{pmatrix} -16 \\ 13 \\ 4 \end{pmatrix} \quad x_3 = \begin{pmatrix} 4 \\ -3 \\ -1 \end{pmatrix}$$

De eigenrijen zijn

$y_1^T = (1, 0, 4)$, $y_2^T = (0, 1, -3)$, $y_3^T = (4, 4, 3)$.

De vectoren zijn zó genormeerd, dat $y_i^T x_i = 1$.

Na het berekenen van een eigensysteem is het aan te bevelen te controleren of de relaties $Y^T X = I$ en eventueel ook $\sum \lambda_i x_i y_i^T = A$ gelden.

Voorbeeld 2 symmetrische matrix

$$A = \begin{pmatrix} 1 & 2 & 4 \\ 2 & 4 & 8 \\ 4 & 8 & 16 \end{pmatrix}$$

Karakteristieke vergelijking: $\lambda^3 - 21\lambda^2 = 0$.

Eigenwaarden $\lambda_1 = \lambda_2 = 0$, $\lambda_3 = 21$.

Eigenvectoren:

$$x_1 = \frac{1}{\sqrt{17}}\begin{pmatrix} 4 \\ 0 \\ -1 \end{pmatrix}, \quad x_2 = \frac{1}{\sqrt{357}}\begin{pmatrix} 2 \\ -17 \\ 8 \end{pmatrix}, \quad x_3 = \frac{1}{\sqrt{21}}\begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix}$$

2.4 Complexe eigenwaarden en -vectoren

Laat matrix A van de orde n een complexe eigenwaarde $\lambda + \mu i$ bezitten en laat $r + si$ een bijbehorende eigenvector aanduiden. Dan luidt het op te lossen lineaire stelsel

$$(A - \lambda - \mu i)\ (r + si) = \vec{0}\ .$$

Dit is equivalent met het volgende reële stelsel van de orde 2n:

$$\begin{pmatrix} A - \lambda I & + \mu I \\ - \mu I & A - \lambda I \end{pmatrix}\begin{pmatrix} r \\ s \end{pmatrix} = \begin{matrix} \vec{0} \\ \vec{0} \end{matrix}$$

Als $\lambda + \mu i$ enkelvoudig is, heeft dit systeem de rang $2n - 2$, m.a.w. we kunnen 2 vergelijkingen schrappen en twee onbekenden willekeurig kiezen. Door deze keuze worden reëel en imaginair deel van de normeringsfactor vastgelegd.

Voorbeeld

$$A = \begin{pmatrix} 1 & -2 \\ 2 & 1 \end{pmatrix}$$

Karakteristieke vergelijking: $\lambda^2 - 2\lambda + 5 = 0$

Eigenwaarden: $\lambda_1 = 1 + 2i$, $\lambda_2 = 1 - 2i$

Het op te lossen stelsel voor de eigenvector $x_1 = r + si$ van λ_1 luidt

$$\begin{pmatrix} 0 & -2 & 2 & 0 \\ 2 & 0 & 0 & 2 \\ -2 & 0 & 0 & -2 \\ 0 & -2 & 2 & 0 \end{pmatrix}\begin{pmatrix} r_1 \\ r_2 \\ s_1 \\ s_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

We kunnen kiezen $r_1 = 1$, $s_1 = 0$ en krijgen dan

$$x_1 = \begin{pmatrix} 1 \\ -i \end{pmatrix}\ ,\quad x_2 = \begin{pmatrix} 1 \\ i \end{pmatrix}$$

en de eigenrijen zijn, na normering:

$$y_1^T = \tfrac{1}{2}\ (1,\ i)\ ,\ y_2^T = \tfrac{1}{2}\ (1,\ -i)$$

Opgaven

102) Bereken in 7 cijfers nauwkeurig de eigenwaarden en genormeerde eigenvectoren van de volgende matrices

$$\begin{pmatrix} 6 & 3 \\ 3 & 2 \end{pmatrix}, \quad \begin{pmatrix} 60 & 30 & 20 \\ 30 & 20 & 15 \\ 20 & 15 & 12 \end{pmatrix}$$

103) Bereken in 5 cijfers nauwkeurig de eigenwaarden, eigenkolommen en eigenrijen van

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}, \begin{pmatrix} .45 & -.90 & -.27 & .40 \\ 0 & 1 & .30 & 0 \\ 1 & 0 & 0 & 0 \\ -.58 & 0 & 0 & 1 \end{pmatrix}$$

3. Matrix-maal-vector iteratie (power method)

Deze methode is een vector-iteratie, d.w.z. de iterates zijn vectoren, die we aanduiden met $u^{(0)}$, $u^{(1)}$, $u^{(2)}$, ... (vgl. p. 146 en 152). De start-vector $u^{(0)}$ wordt gekozen en de iteratie-stap luidt:

3.0.1 $$u^{(i+1)} = Au^{(i)}.$$

De methode heeft alleen succes als A een dominante eigenwaarde heeft, d.w.z. een eigenwaarde die in absolute waarde alle andere overtreft. We nemen gemakshalve aan dat deze dominante eigenwaarde enkelvoudig is. Als de eigenwaarden volgens absolute grootte geordend zijn, hebben we dus

$$|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|.$$

Verder nemen we aan, dat A diagonaliseerbaar is en dus n lineair onafhankelijke eigenvectoren $x_1, \ldots, x_n$ bezit. Dan kan dus iedere vector geschreven worden als lineaire combinatie hiervan, in het bijzonder

3.0.2 $$u^{(0)} = \alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n.$$

Gaan we nu itereren, dan krijgen we

$$u^{(1)} = Au^{(0)} = \alpha_1\lambda_1 x_1 + \alpha_2\lambda_2 x_2 + \ldots + \alpha_n\lambda_n x_n, \text{ en algemeen}$$

3.0.3 $$u^{(i)} = \alpha_1\lambda_1^i x_1 + \alpha_2\lambda_2^i x_2 + \ldots + \alpha_n\lambda_n^i x_n.$$

Nu nemen we aan dat α_1 ongelijk aan 0 is. Dan geldt

3.0.4 $$u^{(i)} = \alpha_1\lambda_1^i\left[x_1 + \frac{\alpha_2}{\alpha_1}\left(\frac{\lambda_2}{\lambda_1}\right)^i x_2 + \ldots + \frac{\alpha_n}{\alpha_1}\left(\frac{\lambda_n}{\lambda_1}\right)^i x_n\right].$$

Omdat $\left(\frac{\lambda_k}{\lambda_1}\right)^i$ voor $k > 1$ naar 0 convergeert, convergeert de richting van $u^{(i)}$ naar de richting van de eigenvector x_1. De vector $u^{(i)}$ zelf convergeert echter niet, omdat de factor $\alpha_1\lambda_1^i$ naar oneindig of naar 0 gaat. Deze factor werken we weg, door de vector $u^{(i)}$ te normeren.

3.1 Intermezzo over normen

De norm van een vector v wordt aangeduid met $||v||$.
Het belangrijkste zijn de volgende twee normen:

3.1.1 $$||v|| = \sqrt{v_1^2 + \ldots + v_n^2}$$ (Euclidische norm),

3.1.2 $$||v|| = \max_{k=1,\ldots,n} |v_k|.$$

Delen we een vector $v \neq 0$ door zijn norm, dan ontstaat een genormeerde vector $w = \frac{1}{||v||} v$.
Bij gebruik van de Euclidische norm, krijgen we een vector w ter lengte 1 en bij de tweede norm heeft w een absoluut grootste element 1.

Keren we terug tot onze iteratie-vector $u^{(i)}$.
De genormeerde vector $\frac{1}{||u^{(i)}||} u^{(i)}$ is gelijk aan $\frac{\alpha_1 \lambda_1^i}{||u^{(i)}||}$ maal een vector, die naar x_1 convergeert.

Blijkbaar moet dan $\frac{\alpha_1 \lambda_1^i}{||u^{(i)}||}$ naar 1 convergeren. Samenvattend hebben we de volgende

3.2 Stelling. Zij A een diagonaliseerbare matrix met enkelvoudige dominante eigenwaarde λ_1 en bijbehorende eigenvector x_1. Zij $u^{(0)}$ een start-vector, zodanig gekozen dat $\alpha_1 \neq 0$ in (3.0.2). Dan geldt

3.2.1 $$\lim_{i\to\infty} \frac{1}{||u^{(i)}||} u^{(i)} = x_1.$$

De eigenwaarde hierbij kan op verschillende wijzen worden berekend.
We hebben namelijk

3.2.2 $$\lim_{i\to\infty} \frac{||u^{(i+1)}||}{||u^{(i)}||} = |\lambda_1|.$$

3.2.3 $$\lim_{i\to\infty} \frac{u_k^{(i+1)}}{u_k^{(i)}} = \lambda_1,$$

waarbij k zo genomen wordt, dat $u_k^{(i)}$ een element van $u^{(i)}$ met maximale absolute waarde is.

De afleiding van 3.2.3 verloopt als volgt.
Meestal zal k op den duur costant worden en dan hebben we wegens (3.0.4):

$$\frac{u_k^{(i+1)}}{u_k^{(i)}} = \lambda_1 \frac{x_{k1} + \frac{\alpha_2}{\alpha_1}(\frac{\lambda_2}{\lambda_1})^{i+1} x_{k2} + \ldots}{x_{k1} + \frac{\alpha_2}{\alpha_1}(\frac{\lambda_2}{\lambda_1})^{i} x_{k2} + \ldots} .$$

Omdat x_{k1} de limiet is van $u_k^{(i)}$ en $u_k^{(i)}$ maximaal gekozen was, geldt zeker $x_{k1} \neq 0$. Dus

$$\frac{u_k^{(i+1)}}{u_k^{(i)}} = \lambda_1 \frac{1 + \frac{\alpha_2}{\alpha_1}(\frac{\lambda_2}{\lambda_1})^{i+1} \frac{x_{k2}}{x_{k1}} + \ldots}{1 + \frac{\alpha_2}{\alpha_1}(\frac{\lambda_2}{\lambda_1})^{i} \frac{x_{k2}}{x_{k1}} + \ldots}$$

$$\approx \lambda_1(1 + \frac{\alpha_2}{\alpha_1}(\frac{\lambda_2}{\lambda_1})^{i} (\frac{\lambda_2}{\lambda_1} - 1)\frac{x_{k2}}{x_{k1}} + \ldots)$$

$$= \lambda_1(1 + \delta_i)$$

Waarbij δ_i de relatieve fout is.
Blijkbaar geldt

$$\delta_{i+1} \approx \frac{\lambda_2}{\lambda_1} \delta_i .$$

M.a.w. $\frac{u_k^{(i+1)}}{u_k^{(i)}}$ convergeert lineair naar λ_1 met convergentie-factor $\frac{\lambda_2}{\lambda_1}$.
Beschouwing van (3.0.4) leert dat de convergentie van $\frac{1}{||u^{(i)}||_{\lambda_1}} u^{(i)}$ eveneens lineair geschiedt met dezelfde convergentie-factor $\frac{\lambda_2}{\lambda_1}$.

Als $|\lambda_2|$ maar weinig kleiner is dan $|\lambda_1|$, moeten we dus op een zeer langzame convergentie rekenen.

3.3 Opmerkingen

a) De iteratie convergeert naar λ_1 en x_1 als $\alpha_1 \neq 0$.
Start men toevallig met een $u^{(0)}$, waarvoor $\alpha_1 = 0$, dan zal toch door afrondingsfouten op den duur x_1 erin komen (d.w.z. $\alpha_1 \neq 0$ worden) en de iteratie naar x_1 convergeren (wat natuurlijk wel veel tijd kost).

b) We hebben reeds gezien, dat de vectoren $u^{(i)}$ of naar oneindig of naar 0 gaan. Dit is onaangenaam en kan eenvoudig verholpen worden, door de iteratie-stap (3.0.1) als volgt te wijzigen:

3.3.1
$$v^{(i)} = Au^{(i)}$$
$$u^{(i+1)} = \frac{1}{||v^{(i)}||} v^{(i)} .$$

M.a.w. in elke stap normeren we de geïtereerde vector. Dan convergeert $u^{(i)}$ blijkbaar naar een genormeerde eigenvector en $||v^{(i)}||$ convergeert wegen 3.2.2 naar $|\lambda_1|$.

3.4 Matrix maal vector iteratie voor symmetrische matrices

Stelling (3.2) luidt voor een symmetrische matrix A blijkbaar aldus:
Als A een enkelvoudige dominante eigenwaarde λ_1 met bijbehorende genormeerde eigenvector x_1 heeft en $u^{(0)}$ een startvector is, die niet loodrecht op x_1 staat, dan convergeert $\frac{1}{||u^{(i)}||} u^{(i)}$ naar x_1.
De convergentie is voor symmetrische A ook lineair met dezelfde factor $\frac{\lambda_2}{\lambda_1}$. Voor de eigenwaarde-berekening gebruikt men hetzij (3.2.2) met Euclidische norm, hetzij

3.4.1
$$\frac{u^{(i)T} u^{(i+1)}}{u^{(i)T} u^{(i)}} \to \lambda_1 .$$

Deze formules convergeren eveneens lineair maar nu met factor $(\frac{\lambda_2}{\lambda_1})^2$.

Voor (3.3.1) vinden we namelijk wegens de orthogonaliteit van de eigenvectoren:

$$\frac{u^{(i)T}\,u^{(i+1)}}{u^{(i)T}\,u^{(i)}} = \frac{\alpha_1^2\lambda_1^{2i+1}\,x_1^Tx_1 + \alpha_2^2\lambda_2^{2i+1}\,x_2^Tx_2 + \ldots}{\alpha_1^2\lambda_1^{2i}\,x_1^Tx_1 + \alpha_2^2\lambda_2^{2i}\,x_2^Tx_2 + \ldots}$$

$$= \lambda_1\,\frac{1 + \frac{\alpha_2^2}{\alpha_1^2}\left(\frac{\lambda_2}{\lambda_1}\right)^{2i+1} + \ldots}{1 + \frac{\alpha_2^2}{\alpha_1^2}\left(\frac{\lambda_2}{\lambda_1}\right)^{2i} + \ldots} \approx \lambda_1\left(1 + \frac{\alpha_2^2}{\alpha_1^2}\left(\frac{\lambda_2}{\lambda_1}\right)^{2i}\left(\frac{\lambda_2}{\lambda_1} - 1\right) + \ldots\right).$$

Schrijven we hiervoor weer $\lambda_1(1 + \delta_i)$ dan hebben we nu $\delta_{i+1} \approx \left(\frac{\lambda_2}{\lambda_1}\right)^2 \delta_i$.

3.5 Voorbeeld

$$A = \begin{pmatrix} 261 & -70 & -115 \\ -70 & 36 & 10 \\ -115 & 10 & 101 \end{pmatrix}, \quad u^{(0)} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

$v^{(0)}$	$u^{(1)}$	$v^{(1)}$	$u^{(2)}$	$v^{(2)}$	$u^{(3)}$	$v^{(3)}$	$u^{(4)}$
76	.9524	275.39	.8900	293.64	.8759	293.44	.8734
-24	-.3008	-78.00	-.2521	-75.18	-.2242	-73.66	-.2192
-4	-.0501	-117.59	-.3800	-143.25	-.4273	-146.13	-.4349

$$||v^{(1)}|| = 309.44 \quad ||v^{(2)}|| = 335.26 \quad ||v^{(3)}|| = 335.99$$

Na 7 iteratie-stappen vinden we $\lambda_1 \approx 336.00$ en $x_1 \approx u^{(7)} =$

$$= \begin{pmatrix} .8729 \\ -.2182 \\ -.4364 \end{pmatrix}.$$

3.6 Volgende eigenwaarden en eigenvectoren

De matrix maal vector methode levert de dominante eigenwaarde λ_1 en bijbehorende eigenvector x_1. Om een volgende te vinden staan ons twee mogelijkheden open

1) $\lambda_1 = 0$ maken door deflatie
2) $\alpha_1 = 0$ maken en houden door orthogonalisatie.

Ad 1. Deflatie volgens Hotelling

De deflatie volgens Hotelling verloopt voor symmetrische matrix A aldus. A wordt vervangen door

3.6.1 $$A_2 = A - \lambda_1 x_1 x_1^T ,$$

waarbij x_1 genormeerd is op lengte 1, dus $x_1^T x_1 = 1$.
Matrix A_2 is eveneens symmetrisch en heeft, volgens definitie (1.1), dezelfde eigenvectoren als A met eigenwaarden $0, \lambda_2, \ldots, \lambda_n$.
Waren $\lambda_1, \ldots, \lambda_n$ volgens absolute grootte gerangschikt, dan is nu dus λ_2 dominant, mits $|\lambda_2| \geq |\lambda_3|$.
De matrix maal vector methode toegepast op A_2 levert dan dus λ_2 en x_2.
Hierna kunnen we weer defleren en zo achtereenvolgens alle eigenwaarden en -vectoren vinden.
Als A niet symmetrisch is luidt Hotelling's deflatie-formule

$$A_2 = A - \lambda_1 x_1 y_1^T ,$$

waarbij x_1 de eigenkolom en y_1^T de eigenrij bij λ_1 is en de normering zodanig is, dat $y_1^T x_1 = 1$.
Ook nu heeft A_2 de eigenwaarden $0, \lambda_2, \ldots, \lambda_n$ en dezelfde eigenvectoren als A. Nadeel van deze deflatie-formule is, dat behalve x_1 ook y_1^T bekend moet zijn, wat dus dubbel zoveel werk betekent. Bovendien schijnt de nauwkeurigheid van de volgende eigenwaarden en eigenvectoren bij deze deflatie ernstig te kunnen worden aangetast (zie [29] p.585).

Voorbeeld

Deflatie van de matrix uit voorbeeld (3.5) met $\lambda_1 = 336.00$ en $x_1^T = (.8729, -.2182, -.4364)$ levert

$$A_2 = \begin{pmatrix} 4.98 & -6.00 & 12.99 \\ -6.00 & 20.00 & -21.99 \\ 12.99 & -21.99 & 37.01 \end{pmatrix}$$

Toepassing van de matrix maal vector iteratie met startvector $u^{(0)T} = (1,1,1)$ levert na 5 stappen $\lambda_2 \approx 55.99$,
$x_2^T = (.2670, -.5344, .8019)$.
Vervolgens vormen wij $A_3 = A_2 - \lambda_2 x_2 x_2^T =$

$$\begin{pmatrix} .99 & 1.99 & 1.00 \\ 1.99 & 4.01 & 2.00 \\ 1.00 & 2.00 & 1.01 \end{pmatrix}$$

Iteratie hierop levert $\lambda_3 = 6.00$ en

$$x_3 = (.4067, .8167, .4083).$$

Nogmaals deflatie moet, afgezien van afrondingsfouten, de nulmatrix opleveren. De resulterende matrix kan dus een idee geven van de bereikte precisie.

Ad 2. Orthogonalisatie

We nemen weer aan, dat A symmetrisch is.
We kunnen tijdens het itereren $\alpha_1 = 0$ houden, door in elke stap de geïtereerde vector te orthogonaliseren t.o.v. x_1. Hiertoe wordt de iteratie-stap (vgl. 3.3.1) aldus gewijzigd.

3.6.2
$$\begin{cases} v^{(i)} = Au^{(i)} \\ w^{(i)} = v^{(i)} - (x_1^T v^{(i)})x_1 \\ u^{(i+1)} = \dfrac{1}{\|w^{(i)}\|} w^{(i)} \end{cases}$$

Hierbij nemen we aan, dat x_1 genormeerd is op lengte 1.
Op deze wijze staat $w^{(i)}$, en dus ook $u^{(i+1)}$, inderdaad loodrecht op x_1. Immers $x_1^T w^{(i)} = x_1^T v^{(i)} - (x_1^T v^{(i)})(x_1^T x_1) = 0$, omdat $x_1^T x_1 = 1$ verondersteld is.
Als $|\lambda_1| > |\lambda_2| > |\lambda_3| \geq \ldots \geq |\lambda_n|$, convergeert deze iteratie naar x_2 en λ_2.
Hebben we $\lambda_1, \ldots, \lambda_m$ met hun eigenvectoren $x_1, \ldots, x_m$ bepaald en zoeken we λ_{m+1} met x_{m+1}, dan wordt in (3.6.2) de formule voor $w^{(i)}$ vervangen door:

$$3.6.3 \qquad w^{(i)} = v^{(i)} - (x_1^T v^{(i)})x_1 - \ldots - (x_m^T v^{(i)})x_m .$$

Voor grotere waarde van m betekent dit aanzienlijk meer rekenwerk in elke iteratie-stap.

Daarom zal men meestal de voorkeur geven aan deflatie. Heeft men echter een zeer grote matrix met overwegend nullen, dan wil men niet defleren, omdat deflatie een volle matrix (zonder nullen) levert.
In dit geval is orthogonalisatie aan te bevelen, omdat daarbij de matrix, en dus ook het nul-patroon, gehandhaafd blijft.
Bij niet-symmetrische matrices moet men, om x_2 te bepalen, de vector $v^{(i)}$ orthogonaliseren t.o.v. de eigenrij y_1^T. Hier hebben we dus hetzelfde nadeel als bij Hotelling's deflatie, dat behalve x_1 ook y_1^T bekend moet worden.

<u>Voorbeeld</u>

We beschouwen weer de matrix uit voorbeeld (3.5) met

$$x_1^T = (.8729, -.2182, -.4364).$$

De iteratie volgens (3.6.2) startend met $u^{(0)T} = (1,1,1)$ ziet er zo uit:

$v^{(0)}$	$w^{(0)}$	$u^{(1)}$	$v^{(1)}$	$w^{(1)}$	$u^{(2)}$	$v^{(2)}$	$w^{(2)}$	$u^{(3)}$
76	12.00	.3810	14.986	14.985	.2868	15.338	15.330	.2688
-24	-8.00	-.2540	-26.924	-26.924	-.5153	-30.315	-30.313	-.53 5
-4	28.00	.8890	43.434	43.435	.8312	45.816	45.820	.8033

$x_1^T v^{(0)} = .7332$, $|w^{(0)}| = 31.496$, $x_1^T v^{(1)} = .0015$, $|w^{(1)}| = 52.254$, $x_1^T v^{(2)} = .0092$, $|w^{(2)}| = 57.038$

Zo doorgaande krijgt men na 6 iteraties

$$\lambda_2 \approx 56.00 \ , \quad x_2 \approx \begin{pmatrix} .2673 \\ -.5345 \\ .8018 \end{pmatrix}$$

3.7 Eigenwaarden met gelijke modulus

Wat gebeurt er als de twee grootste eigenwaarden dezelfde modulus hebben, m.a.w. $|\lambda_1| = |\lambda_2|$? Nemen we aan dat de andere eigenwaarden in absolute waarde kleiner zijn en alles volgens absolute grootte geordend, dan hebben we dus

$$|\lambda_1| = |\lambda_2| > |\lambda_3| \geq \ldots \geq |\lambda_n| .$$

Als de matrix reëel symmetrisch is, zijn de eigenwaarden ook reëel. Gelijke modulus houdt dan slechts twee mogelijkheden in, nl.
$\lambda_1 = \lambda_2$ of $\lambda_1 = -\lambda_2$.

3.7.1 Het geval $\lambda_1 = \lambda_2$

In dit geval is met x_1 en x_2 ook elke lineaire combinatie $\alpha x_1 + \beta x_2$ eigenvector bij λ_1. Zij de startvector

$$u^{(0)} = \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \ldots + \alpha_n x_n ,$$

dan convergeert $\frac{1}{||u^{(i)}||} u^{(i)}$ nu naar de eigenvector $\alpha_1 x_1 + \alpha_2 x_2$.
De convergentie is lineair met factor $\frac{\lambda_3}{\lambda_1}$.

De aldus gevonden eigenvector mogen we nu gerust x_1 noemen. Gaan we hiermee defleren of ten opzichte hiervan orthogonaliseren, dan levert de volgende matrix maal vector iteratie een tweede eigenvector bij λ_1, die netjes loodrecht op de eerste staat.

3.7.2 Het geval $\lambda_1 = -\lambda_2$

Dit geval geeft evenmin moeilijkheden, mits men de iteraties om de andere bekijkt. Dan itereert men als het ware met A^2, die twee gelijke eigenwaarden λ_1^2 en λ_2^2 heeft. Hebben we voldoende overeenstemming tussen $u^{(2i-2)}$ en $u^{(2i)}$, dan geldt dus voor de genormeerde $u^{(2i)}$ en voor $Au^{(2i)}$

$$u^{(2i)} \approx \alpha_1 x_1 + \alpha_2 x_2$$

$$Au^{(2i)} \approx \alpha_1 \lambda_1 x_1 - \alpha_2 \lambda_1 x_2 .$$

We mogen aannemen, dat λ_1 de positieve van het paar tegengestelde eigenwaarden is. De waarde hiervan vinden we (itererend volgens 3.0.1) uit

$$\lambda_1 \approx \sqrt{\frac{||u^{(2i)}||}{||u^{(2i-2)}||}} .$$

Met de aldus gevonden λ_1 krijgen we

$$\lambda_1 u^{(2i)} + Au^{(2i)} \approx 2\alpha_1 \lambda_1 x_1$$

$$\lambda_1 u^{(2i)} - Au^{(2i)} \approx 2\alpha_2 \lambda_1 x_2$$

waarna normering de gewenste eigenvectoren oplevert.

3.7.3 Opmerkingen

Hiermee zijn voor symmetrische matrices alle mogelijkheden uitgeput. Gevallen, waarin meer dan twee in absolute waarde gelijke eigenwaarden optreden gaan volkomen analoog. Bij asymmetrische matrices kunnen complexe eigenwaarden optreden.

Dan is met λ_1 ook $\bar{\lambda}_1$ een eigenwaarde, en deze hebben dezelfde absolute waarde. In dit geval convergeert de matrix maal vector iteratie niet.

3.8 Verschuiving van de oorsprong

Men kan trachten de convergentie te versnellen door de oorsprong te verschuiven, d.w.z. in plaats van met A itereert men met A-pI, voor geschikt gekozen p.
Deze matrix heeft immers dezelfde eigenvectoren, terwijl de eigenwaarden met p verminderd zijn. Met behulp van verschuiving kan men bovendien zowel de grootste als de kleinste eigenwaarde bepalen, zonder de hulp van deflatie of orthogonalisatie in te roepen.

3.9 Practische toepassing

Voordeel van de matrix maal vector methode is, dat de formules eenvoudig zijn. Nadeel is, dat convergentie zo langzaam is, als eigenwaarden dicht bij elkaar liggen.
Bij matrices van enige omvang zal men dan ook de methode slechts gebruiken voor het vinden van een betrekkelijk klein aantal eigenwaarden.
Voor grote matrices met overwegend nullen kan de methode aantrekkelijk zijn, omdat zij van deze nullen kan profiteren (zie 3.6 orthogonalisatie).

Opgaven

104 a) Schrijf een procedure, die de dominante eigenwaarde met eigenvector berekent van een gegeven matrix.

b) Schrijf een volledig programma, dat met behulp van deze procedure en met verschuiving van de oorsprong de grootste en kleinste eigenwaarde met eigenvector berekent.

105 a) Schrijf een procedure, die de deflatie volgens Hotelling uitvoert bij gegeven symmetrische matrix.

b) Schrijf een programma, dat van een gegeven symmetrische matrix een zeker aantal meest dominante eigenwaarden en bijbehorende eigenvectoren berekent.

106) Bereken met de matrix maal vector methode en orthogonalisatie (in 3 dec.) de eigenwaarden en -vectoren van de volgende matrices

$$\begin{pmatrix} 60 & 30 & 20 \\ 30 & 20 & 15 \\ 20 & 15 & 12 \end{pmatrix}, \qquad \begin{pmatrix} 1.5884 & 1.3314 & .5811 \\ 1.3314 & -1.0438 & -.8368 \\ .5811 & -.8368 & -.3146 \end{pmatrix}.$$

107) Bereken met de matrix maal vector methode en Hotellings deflatie de eigenwaarden en eigenvectoren, in 3 decimalen, van

$$\begin{pmatrix} 420 & 210 & 140 & 105 \\ 210 & 140 & 105 & 84 \\ 140 & 105 & 84 & 70 \\ 105 & 84 & 70 & 60 \end{pmatrix}$$

4. Methode van Jacobi

4.0 Inleiding

Deze methode is speciaal voor symmetrische matrices en is erop gericht tegelijk alle eigenwaarden en eigenvectoren op te leveren.
De methode is een iteratie, waarbij de iterates nu zijn matrices $A = A^{(0)}, A^{(1)}, A^{(2)}, \ldots$. Elke iteratie-stap is een orthogonale gelijkvormigheids-transformatie:

$$4.0.1 \qquad A^{(k+1)} = R_k^{-1} A^{(k)} R_k ,$$

waarbij R_k een orthogonale matrix van eenvoudig type is, namelijk een vlakke rotatie.
Vanwege de gelijkvormigheid blijven de eigenwaarden invariant en omdat R_k orthogonaal is blijven de iterates symmetrisch.
Wanneer een iterate $A^{(m)}$ in voldoende precisie een diagonaal-matrix is, is het doel bereikt; de eigenwaarden van A zijn dan de diagonaal-elementen van $A^{(m)}$ en de eigenvectoren worden verkregen als de kolommen van het product $R_0 \; R_1 \; \ldots \; R_{m-1}$.
Een vlakke rotatie R_k wordt bepaald door drie gegevens, namelijk twee indices p en q, aangevend dat de draaiing plaats vindt in het vlak door de eenheids-vectoren e_p en e_q, en een draaiingshoek θ. De matrix R_k ziet er dus zo uit

$$R_k = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & \cos\theta & & -\sin\theta & \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & \sin\theta & & \cos\theta & \\ & & & & & & 1 \\ & & & & & & & \ddots \\ & & & & & & & & 1 \end{pmatrix}$$

waarbij de niet-vermelde elementen nul zijn.

Precieser:

$$0.2 \qquad \begin{cases} (R_k)_{pp} = (R_k)_{qq} = \cos\theta, \ (R_k)_{qp} = -(R_k)_{pq} = \sin\theta, \\ (R_k)_{ii} = 1 \text{ voor } i \neq p,q, \ (R_k)_{ij} = 0 \text{ voor alle andere } i \text{ en } j \end{cases}$$

De matrix $R_k^{-1} = R_k^T$ wordt blijkbaar verkregen door sin θ en -sin θ te verwisselen. Vanwege de eenvoudige gedaante van R_k en R_k^{-1} worden in de k-de iteratie-stap slechts de p-de en de q-de rij en kolom van $A^{(k)}$ gewijzigd.
Voor $i \neq p,q$ geldt

$$4.0.3 \qquad \begin{cases} a_{ip}^{(k+1)} = (A^{(k)}R_k)_{ip} = a_{ip}^{(k)}\cos\theta + a_{iq}^{(k)}\sin\theta = a_{pi}^{(k+1)} \\ a_{iq}^{(k+1)} = (A^{(k)}R_k)_{iq} = -a_{ip}^{(k)}\sin\theta + a_{iq}^{(k)}\cos\theta = a_{qi}^{(k+1)} \end{cases}$$

Blijven nog over de 4 elementen, waarvoor beide indices p of q zijn:

$$4.0.4 \qquad \begin{cases} a_{pp}^{(k+1)} = a_{pp}^{(k)}\cos^2\theta + a_{qq}^{(k)}\sin^2\theta + a_{pq}^{(k)}\sin(2\theta) \\ a_{qq}^{(k+1)} = a_{pp}^{(k)}\sin^2\theta + a_{qq}^{(k)}\cos^2\theta - a_{pq}^{(k)}\sin(2\theta) \\ a_{pq}^{(k+1)} = \tfrac{1}{2}(a_{qq}^{(k)} - a_{pp}^{(k)})\sin(2\theta) + a_{pq}^{(k)}\cos(2\theta) = a_{qp}^{(k+1)} \end{cases}$$

We willen uiteindelijk een diagonaal-matrix krijgen, m.a.w. de niet-diagonale elementen moeten nul worden. Daarom maken we in de k-de stap het element $a_{pq}^{(k+1)}$ gelijk aan nul.
Deze voorwaarde bepaalt de draaiingshoek θ, namelijk

$$4.0.5 \qquad \tan(2\theta) = 2\,a_{pq}^{(k)}/(a_{pp}^{(k)} - a_{qq}^{(k)}).$$

Hierbij zullen we θ altijd zo kiezen, dat $|\theta| \leq \pi/4$ en als $a_{pp}^{(k)} = a_{qq}^{(k)}$ nemen we $\theta = \pm \pi/4$.
Het proces is iteratief, omdat elke stap de eerder geïntroduceerde nullen weer kan verstoren.

4.1 De berekening van cos en sin

In feite hebben we niet de hoek θ nodig, maar cos θ en sin θ. Ter afkorting schrijven we

4.1.0 $$2\,a_{pq}^{(k)} = \lambda, \quad a_{pp}^{(k)} - a_{qq}^{(k)} = \mu.$$

Dan gaat (4.0.5) over in

4.1.1 $$\tan(2\theta) = \lambda/\mu.$$

Omdat we θ zo kiezen, dat $|\theta| \leq \pi/4$, zijn cos θ en cos (2θ) minstens 0. Daarom nemen we in de volgende formule de absolute waarde van μ en natuurlijk de positieve wortel. Dus

4.1.2 $$\cos(2\theta) = |\mu|/\sqrt{\lambda^2 + \mu^2},$$

4.1.3 $$\sin(2\theta) = \lambda\ \mathrm{sgn}(\mu)/\sqrt{\lambda^2 + \mu^2},$$

waarbij sgn (μ) = *if* $\mu \geq 0$ *then* 1 *else* -1.
Hieruit vinden we dan

4.1.4 $$\cos\theta = \sqrt{\frac{1+\cos(2\theta)}{2}} = \sqrt{\tfrac{1}{2}\left(1 + \frac{|\mu|}{\sqrt{\lambda^2+\mu^2}}\right)},$$

4.1.5 $$\sin\theta = \frac{\sin(2\theta)}{2\cos\theta} = \frac{\lambda\ \mathrm{sgn}(\mu)}{2\sqrt{\lambda^2+\mu^2}\cos\theta}.$$

Op deze wijze vallen nergens cijfers weg en krijgen we dus een behoorlijke relatieve precisie, ook als θ in de buurt van 0 ligt. In tegenstelling tot sommige andere iteratie-processen, is hier van meet af aan een behoorlijke precisie nodig.

Hierna voeren we de transformatie uit met behulp van (4.0.3) en (4.0.4), behalve wat betreft $a_{pq}^{(k+1)}$, die we natuurlijk zonder meer 0 stellen.

4.2 Convergentie en strategie

Omdat we naar een diagonaal-matrix streven, is het van belang voor de iterates een maat te hebben, die aangeeft hoever we van dit doel verwijderd zijn. Hiertoe beschouwen we de grootheid

4.2.0 $$\eta_k = \sum_{1\leq i<j\leq n} (a_{ij}^{(k)})^2$$

en gaan nu na hoe η_{k+1} eruit ziet. De enige elementen, die veranderen, zijn die in de p-de en q-de rij en kolom. Uit (4.0.3) volgt

4.2.1 $$(a_{ip}^{(k+1)})^2 + (a_{iq}^{(k+1)})^2 = (a_{ip}^{(k)})^2 + (a_{iq}^{(k)})^2 \text{ voor } i \neq p,q.$$

Dus het enige element, dat in η_k verandering kan brengen is $a_{pq}^{(k)}$. Omdat deze 0 wordt, geldt blijkbaar

4.2.2 $$\eta_{k+1} = \eta_k - (a_{pq}^{(k)})^2.$$

M.a.w. in elke Jacobi-stap boeken we een winst $(a_{pq}^{(k)})^2$.
Er zijn nu 3 strategieën mogelijk.

a) Bepaal p en q in elke stap zodanig, dat $|a_{pq}^{(k)}|$ maximaal is. Dan is de winst in deze stap zo groot mogelijk. Nadeel is, dat het zoeken van het maximale element te veel tijd kost.

b) Werk alle niet-diagonale elementen af in een vaste volgorde. Om tijd te sparen, is het echter van belang kleine elementen voorlopig te negeren. Vandaar de volgende tactiek.

c) de drempel-strategie

Werk alle niet-diagonale elementen af in vaste volgorde, maar doe de transformatie alleen, als $|a_{pq}^{(k)}| \geq$ drempel. Als alle niet-diagonale elementen in absolute waarde kleiner dan de drempel blijken te zijn, wordt de drempel verlaagd. Dit wordt herhaald, totdat de drempel klein genoeg is en alle niet-diagonale elementen eronder liggen.

Deze strategie levert altijd convergentie op. Immers voor vaste drempel hebben we

$$0 \leq \eta_{k+1} \leq \eta_k - (\text{drempel})^2 < \eta_k.$$

Dus na een eindig aantal stappen moeten alle niet-diagonale elementen onder de drempel liggen, omdat anders η negatief zou worden.
We kunnen dus onder elke willekeurig kleine drempel komen, m.a.w. het proces convergeert altijd.
Als alle eigenwaarden verschillend zijn, is de convergentie kwadratisch.

4.3 Voorbeeld (van Wilkinson [29] p. 274).

Wegens de symmetrie schrijven we alleen de bovendriehoeken op.

$$A = A^{(0)} = \begin{pmatrix} .6532 & .2165 & .0031 \\ & .4105 & .0052 \\ & & .2132 \end{pmatrix}$$

$p = 1,\ q = 2,\ \tan(2\theta) = \frac{2 \times .2165}{.6532-.4105},\ \cos\theta = .8628,\ \sin\theta = .5055$

$$A^{(1)} = \begin{pmatrix} .7800 & 0 & .0053 \\ & .2836 & .0029 \\ & & .2132 \end{pmatrix}$$

$p = 1,\ q = 3,\ \tan(2\theta) = \frac{2 \times .0053}{.7800-.2132},\ \cos\theta = 1.0000,\ \sin\theta = .0094$

$$A^{(2)} = \begin{pmatrix} .7801 & .0000 & 0 \\ & .2836 & .0029 \\ & & .2132 \end{pmatrix}$$

$p = 2,\ q = 3,\ \tan(2\theta) = \frac{2 \times .0029}{.2836-.2132},\ \cos\theta = .9991,\ \sin\theta = .0411$

$$A^{(3)} = \begin{pmatrix} .7801 & .0000 & .0000 \\ & .2837 & 0 \\ & & .2131 \end{pmatrix}$$

Dus $\lambda_1 \approx .7801,\ \lambda_2 \approx .2837,\ \lambda_3 \approx .2131$

De matrix X der eigenvectoren is gelijk aan het product der transformerende matrices, dus in dit geval $X = R_0R_1R_2$. Willen we alleen de eigenvector x_1 bij λ_1, dan moeten we de eerste kolom hiervan hebben, dus $x_1 = Xe_1 = R_0R_1R_2e_1$. Dit berekenen we het handigst door e_1 successievelijk links te vermenigvuldigen met R_2, R_1, R_0.

$$e_1 = \begin{pmatrix}1\\0\\0\end{pmatrix}, \quad R_2e_1 = \begin{pmatrix}1\\0\\0\end{pmatrix}, \quad R_1R_2e_1 = \begin{pmatrix}1.0000\\0\\.0094\end{pmatrix}, \quad x_1 = R_0R_1R_2e_1 = \begin{pmatrix}.8628\\.5055\\.0094\end{pmatrix}$$

5. Schets van enige andere methodes

5.0 Householders transformatie

Veel methodes beginnen met een gelijkvormigheidstransformatie, die de matrix in een prettige vorm brengt. Deze transformatie is dan niet iteratief, maar behelst, bij gegeven orde, een vast aantal stappen. Op deze wijze is niet de diagonaalvorm bereikbaar, maar wel de tridiagonaalvorm, d.w.z. alleen de hoofddiagonaal en de 2 aanliggende nevendiagonalen bevatten niet-nul elementen.

Transformatie van symmetrische matrices

Bij symmetrische matrices past men altijd orthogonale transformaties toe, omdat alleen deze de symmetrie handhaven. Men kan voor het bereiken van de tridiagonaalvorm vlakke rotaties gebruiken (methode van Givens). Beter en sneller is Householders transformatie, die bestaat uit n-2 transformaties van de gedaante

5.0.1 $$A^{(k)} = P_k A^{(k-1)} P_k, \quad k=1(1)n-2,$$

waarbij P_k orthogonaal en symmetrisch is (dus $P_k^{-1} = P_k^T = P_k$) en er uitziet als volgt

5.0.2 $$P_k = I - 2\, w_k w_k^T .$$

Deze matrix is blijkbaar symmetrisch en als de vector w_k genormeerd is op lengte 1, is zij bovendien orthogonaal.
De vector w_k begint met k nullen, m.a.w.

$$w_k^T = (0, \ldots, 0, w_{k+1,k}, \ldots, w_{n,k})$$

en wordt zo gekozen, dat in de k-de kolom (en vanwege de symmetrie dus ook in de k-de rij) de gewenste nullen ontstaan. Nullen geïntroduceerd voor lagere waarden van k blijven hierbij intact en zo ontstaat na n-2 stappen de tridiagonaalvorm.

Transformatie van niet-symmetrische matrices

Als de matrix niet symmetrisch is, is Householders transformatie ook heel geschikt. De nullen ontstaan nu echter slechts aan een kant van de diagonaal en de vorm, die ontstaat is de bijnadriehoeksvorm oftewel Hessenbergvorm, d.w.z. de niet-nul elementen komen slechts voor in de boven- of onder-driehoek en de aanliggende nevendiagonaal.

Voor asymmetrische matrices bestaat ook een niet-orthogonale transformatie naar Hessenbergvorm; deze verloopt sneller dan Householders transformatie.
Heeft men eenmaal de Hessenbergvorm, dan kan men deze eventueel nog verder reduceren tot een (niet-symmetrische) tridiagonale vorm. Dit proces is numeriek minder stabiel en het is daarom raadzaam dit in extra precisie uit te voeren.

5.1 Berekening der eigenwaarden

Heeft men de matrix getransformeerd in een gelijkvormige matrix van speciale gedaante, nl. tridiagonaal of bijna driehoeks, dan gaat men hiervan de eigenwaarden berekenen met een of ander iteratief proces ter bepaling van nulpunten (zie hoofdstuk 5).
Het voordeel van de speciale gedaante is, dat de berekening der functiewaarden, in dit geval de waarden van de karakteristieke determinant, aanzienlijk sneller gaat. Voor een willekeurige matrix van de orde n is het aantal bewerkingen, nodig voor het berekenen van de determinant, evenredig met n^3. Heeft de matrix de tridiagonaalvorm of de Hessenbergvorm, dan is dit aantal evenredig met n respectievelijk n^2. Dus voor grote n betekent dit een aanzienlijke tijdbesparing.

Eigenwaarden van symmetrische matrices

In het symmetrische geval levert Householders transformatie een tridiagonale matrix

5.1.1
$$S = \begin{pmatrix} a_1 & b_1 & & & 0 \\ b_1 & a_2 & \ddots & & \\ & \ddots & \ddots & b_{n-1} & \\ & & b_{n-1} & a_n & \\ 0 & & & & \end{pmatrix}$$

Het karakteristieke polynoom $\det(\lambda I - S)$ wordt berekend met de recursie-formule:

5.1.2
$$\begin{cases} f_0 = 1 \\ f_1(\lambda) = \lambda - a_1 \\ \text{---------- } \\ f_i(\lambda) = (\lambda - a_i) f_{i-1} - b_{i-1}^2 f_{i-2}, \; i = 2(1)n \end{cases}$$

$f_i(\lambda)$ is juist gelijk aan $\det(\lambda I - S_i)$, waarbij S_i de submatrix van de orde i in de linkerbovenhoek van S is.
Dus $f_n(\lambda) = \det(\lambda I - S)$ is de gevraagde functie-waarde. Bovendien hebben we de volgende

5.1.3 Stelling van Givens

Als in matrix S alle nevendiagonaal-elementen b_i ongelijk aan 0 zijn, dan vormt de rij f_0, $f_1(\lambda)$, ..., $f_n(\lambda)$ een Sturm-rij, d.w.z. het aantal eigenwaarden van S groter dan λ is gelijk aan het aantal tekenwisselingen in deze rij, mits $f_n(\lambda) \neq 0$.

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 Sturm-rij precies één eigenwaarde zit, dan kan men deze bepalen met de veilige regula falsi (zie pag. 156 en AP 230 pag. 158). Bij een meervoudige eigenwaarde vindt men zo'n interval niet, maar dan levert bisectie de eigenwaarde met zijn multipliciteit.

Eigenwaarden van niet-symmetrische matrices

In het asymmetrische geval levert Householders transformatie een Hessenbergmatrix H op. De karakteristieke determinant van H kan bv. met eliminatie worden berekend. Men kan de eigenwaarden berekenen met de gevaarlijke regula falsi, indien men weet dat alle eigenwaarden reëel zijn of anders met Mullers methode.

5.2 Berekening der eigenvectoren

Heeft men eenmaal een benaderde eigenwaarde λ gevonden, dan kan men de eigenvectoren berekenen door het betreffende stelsel op te lossen. Men kan zo nodig de eigenvector nog verbeteren door inverse iteratie. D.w.z. als u een schatting van een eigenvector van A is, dan berekent men een nieuwe schatting v door op te lossen

5.2.1 $$(A - \lambda I)v = u.$$

Ondanks het feit, dat $A - \lambda I$ bijna singulier is, is v meestal veel beter dan u. Neemt men hier voor A de getransformeerde matrix S of H, dan vergt deze berekening veel minder rekenwerk dan voor een volle matrix. Wel moet men dan de gevonden eigenvector v terugtransformeren tot een eigenvector x van de oorspronkelijke matrix. Dit geschiedt door de vector successievelijk voor te vermenigvuldigen met de transformerende matrices, evenwel in omgekeerde volgorde. Dus voor Householders transformatie (5.0.1) wordt dit

5.2.2 $$x = P_1 P_2 \ldots P_{n-2} v.$$

Opgaven

108) Schrijf een Algolprogramma, dat de eigenwaarden en eigenvectoren van een symmetrische matrix berekent volgens de methode van Jacobi. Het programma moet inlezen de gewenste precisie eps, de orde n, en daarna de matrix.

109) Bereken de 2e en 3e eigenvector van de matrix uit voorbeeld 4.3.

110) Bereken de coëfficiënten der karakteristieke vergelijking en de eigenwaarden en eigenkolommen van

$$\begin{pmatrix} .35 & -.70 & .28 & .91 \\ 0 & 1 & -.4 & 0 \\ 1 & 0 & 0 & 0 \\ .77 & 0 & 0 & 1 \end{pmatrix}$$

De eigenkolommen moeten zo worden genormeerd, dat het absoluut maximale element 1 is. De elementen van de gegeven matrix zijn exact. De vereiste precisie is 5 cijfers (relatieve precisie) voor de eigenwaarden en 5 decimalen (absolute precisie) voor de vectoren.

111) Bereken met de matrix maal vector methode en Hotellings deflatie de eigenwaarden en eigenvectoren, in 3 decimalen, van

$$\begin{pmatrix} -7 & -5 & +17 \\ -5 & +31 & -57 \\ +17 & -57 & +97 \end{pmatrix}$$

```
begin comment R 1086 Progr. M.Fretin 270166. Eigenwaarden en -vectoren
van symm.matrices volgens methode van Jacobi. Het gebruikt Alg. 85 van
Th.E.Evans, Comm.ACM. 5(1962)p.208 behoudens enige wijzigingen, vooral
in de drempel-strategie. Op getalband moet staan  precisie EPS, dan de
matrices als onderdriehoek  voorafgegaan door de orde, en tenslotte 0;
integer N,B,C,i,j,count;real EPS;
procedure JACOBI(A,S,n,rho);value n,rho;integer n;real rho;array A,S;
begin integer i,j,p,q,ind;
    real norm1,norm2,thr,cos2t,sin2t,sint,cost,int1,v1,v2,v3,mu,sq;
    for i:=1 step 1 until n do for j:=1 step 1 until n do
    begin if i=j then S[i,j]:=1.0 else S[i,j]:=S[j,i]:=0.0 end;
    ind:=count:=0;int1:=0;
    for i:=1 step 1 until n do for j:=1 step 1 until i do
    int1:=if abs(A[i,j])>int1 then abs(A[i,j]) else int1; norm1:=int1;
MAIN: int1:=0;
    for i:=2 step 1 until n do for j:=1 step 1 until i-1 do
    int1:= if abs(A[i,j]) >int1 then abs(A[i,j]) else int1;
    norm2:=int1/norm1;if norm2<rho then goto END1;thr:=0.1×norm2;
MAIN1: count:=count + 1;
    for q:=2 step 1 until n do for p:=1 step 1 until q-1 do
    begin if abs(A[p,q])> thr then
        begin ind:=1;v1:=A[p,p]; v2:=A[p,q]; v3:=A[q,q];
            mu:=0.5×(v1-v3);sq:=sqrt(v2↑2 + mu↑2);
            sin2t:=(if  mu>0.0 then v2 else -v2)/sq;cos2t:=abs(mu)/sq;
            cost:=sqrt(0.5×(1+cos2t));sint:=sin2t/(2×cost);
            for i:=1 step 1 until n do
            begin int1:=A[i,p]×cost +A[i,q]×sint;
                A[i,q]:=-A[i,p]×sint +A[i,q]×cost;A[i,p]:=int1;
                int1:=S[i,p]×cost +S[i,q]×sint;
                S[i,q]:=-S[i,p]×sint +S[i,q]×cost;S[i,p]:=int1;
            end;
            for i:=1 step 1 until n do
            begin A[p,i]:=A[i,p];A[q,i]:=A[i,q] end;
            A[p,p]:= v1×cost↑2+v3×sint↑2+v2×sin2t;
            A[q,q]:= v1×sint↑2+v3×cost↑2-v2×sin2t; A[p,q]:=A[q,p]:=0.0;
    end end;
    if ind=1 then begin ind:=0;goto MAIN1 end else
        if thr>rho then goto MAIN;
END1:
end JACOBI;
BEGIN:EPS:=read;RUNOUT;          DEBUT:N:=read;if N =0 then goto END;
begin array M,P[1:N,1:N];
    for i:=1 step 1 until N do for j:=1 step 1 until i do
    M[i,j]:=M[j,i] :=read;JACOBI(M,P,N,EPS);PUNLCR;
    FIXP(3,0,N);PUSPACE(3);FIXP(3,0,count);PUNLCR;B:=1;C:=7;
IN: if N>C then goto PRI else C:=N;
PRI:for i:=B step 1 until C do FLOP(12,3,M[i,i]);PUNLCR;PUNLCR;
    for i:=1 step 1 until N do
    begin for j:=B step 1 until C do FLOP(12,3,P[i,j]);PUNLCR end;
    if N=C then goto FIN;B:=B+7;C:=C+7;PUNLCR;goto IN;
FIN:goto DEBUT
end; END: STOPCODE
end
```

Hoofdstuk 7. Gewone differentiaal-vergelijkingen

0. Inleiding

Een vergelijking is een gelijkheids-relatie, waarin een of meer onbekende grootheden voorkomen. De opgave daarbij luidt voor iedere onbekende een, enige of alle mogelijke waarden te vinden, die de relatie in een identiteit overvoeren. De onbekende grootheden kunnen zijn:

a) reële (of complexe) getallen. In dit geval spreken we van een getal-vergelijking; deze kan zijn lineair, algebraïsch of transcendent.

b) vectoren of matrices. In dit geval hebben we een vector- respectievelijk matrix-vergelijking.

Zie bijvoorbeeld in hoofdstuk 4 de vector-vergelijking (1.1) en de matrix-vergelijkingen (5.0.1) en (6.0.1).

c) reële functies van een (of meer) reële veranderlijken. In dit geval spreken we van functie-vergelijkingen.

Hiertoe behoren de differentiaal-vergelijkingen, dat zijn functie-vergelijkingen waarin een of meer afgeleiden van de onbekende functie voorkomen. Wij beperken ons tot gewone (dat is niet-partiële) differentiaal-vergelijkingen. Dan is de onbekende functie y een functie van één veranderlijke x en de voorkomende afgeleiden zijn gewone afgeleiden naar x. De algemene gedaante van een gewone differentiaal-vergelijking luidt dus

$$0.1 \qquad F\left(x, y(x), \frac{dy(x)}{dx}, \ldots, \frac{d^{(n)}y(x)}{dx^n}\right) = 0.$$

Een oplossing van deze vergelijking is een functie $y(x)$, waarvoor deze relatie identiek geldt voor alle waarden van x gelegen in een zeker interval. De orde van de hoogste voorkomende afgeleide heet de orde van de differentiaal-vergelijking. De differentiaal-vergelijking (0.1) heet lineair, als F lineair is in y en alle voorkomende afgeleiden.

Dus de algemene gedaante van een n-de orde lineaire differentiaal-vergelijking is

0.2 $$a_n y^{(n)} + \ldots + a_1 y' + a_0 y = b,$$

waarbij de coëfficiënten $a_0, \ldots, a_n$ en b gegeven functies van x zijn.

Eerste orde differentiaal-vergelijkingen

De algemene gedaante van een differentiaal-vergelijking van de orde 1 is blijkbaar

0.3 $$F(x, y, \frac{dy}{dx}) = 0.$$

Liever schrijven we $\frac{dy}{dx}$ expliciet als functie van x en y, wat altijd wel mogelijk is, en krijgen dan als algemene gedaante:

0.4 $$\frac{dy}{dx} = f(x,y).$$

Ter illustratie beschouwen we nu lineaire eerste-orde vergelijkingen, die dus de algemene gedaante hebben

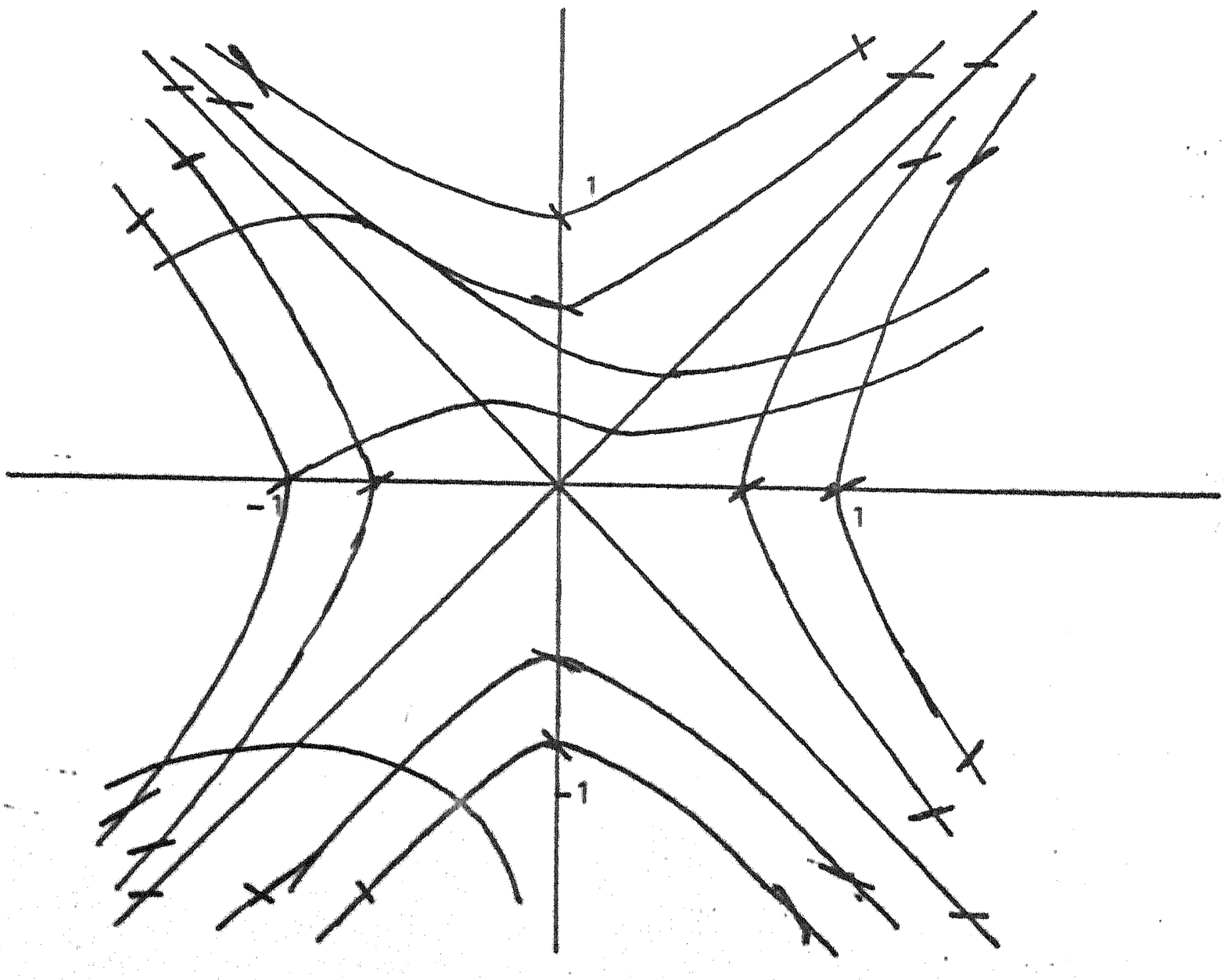
0.5 $$\frac{dy}{dx} = ay + b,$$

waarbij a en b functies van x zijn.
Belangrijke bijzondere gevallen zijn
a) $a = 0$, dus de vergelijking $\frac{dy}{dx} = b$.
De oplossingen zijn $y(x) = \int b(x)dx + C$.
b) $b = 0$, dus $\frac{dy}{dx} = ay$.
De oplossingen hiervan zijn $y(x) = C \exp(\int a(x)dx)$.
Als a en b geen van beide nul zijn, kan het aanzienlijk ingewikkelder worden (zie bv. [1](d.i. Modern Computing Methods) pag. 80).
In deze bijzondere gevallen zien we, dat de differentiaal-vergelijking wordt opgelost door het berekenen van onbepaalde integralen. Naar aanleiding hiervan heeft het woord "integreren" de ruimere betekenis gekregen van "oplossen van differentiaal-vergelijkingen" en de verkregen oplossing heet dan ook "integraal" van de differentiaal-vergelijking.

Grafische methode voor het integreren van een eerste orde vergelijking

Een eenvoudige manier om inzicht te krijgen in de gedaante van de integralen van een differentiaal-vergelijking is het tekenen van de isoklienen, dat zijn lijnen van gelijke helling. Voor de differentiaal-vergelijking $y' = f(x,y)$ zijn de isoklienen de lijnen die voldoen aan $f(x,y) =$ constant. Op een dergelijke lijn is dus de helling y' constant. Men kiest zoveel waarden voor de constante, dat de isoklienen voldoende dicht bij elkaar liggen.
Een manier van oplossen is: start in een punt (x_0, y_0) gelegen op een isoklien en ga met de bijbehorende helling y' verder totdat een andere isoklien bereikt wordt, ga dan met de nieuwe helling verder, enz.
Beter nog is het, tussen twee isoklienen een gemiddelde helling te nemen.

Voorbeeld. (Uit Hamming, pag. 184). Gegeven de differentiaal-vergelijking $y' = x^2 - y^2$. De isoklienen $x^2 - y^2 = C$ zijn gelijkzijdige hyperbolen, die voor $C = 0$ ontaarden in het lijnenpaar $x = \pm y$.



Men ziet dat met ieder startpunt een nieuwe oplossing correspondeert.
Men kan een oplossing van (0.4) vastleggen door het geven van x_0 en een bijbehorende <u>beginwaarde</u> $y_0 = y(x_0)$. Men heeft dan een zgn. <u>beginwaarde-probleem</u> van de orde een:

0.6
$$\begin{cases} y' = f(x,y) \\ y(x_0) = y_0. \end{cases}$$

Een <u>beginwaarde-probleem van de orde n</u> heeft als algemene gedaante

0.7
$$\begin{cases} F(x, y, y', \ldots, y^{(n)}) = 0 \\ y(x_0) = y_0 \\ y'(x_0) = y_1 \\ \vdots \\ y^{(n-1)}(x_0) = y_{n-1} \end{cases}$$

waarbij $x_0, y_0, y_1, \ldots, y_{n-1}$ gegeven getallen zijn.
Deze extra condities leggen, onder vrij algemene voorwaarden, de oplossing eenduidig vast.
Veel lastiger probleem, zgn. <u>randwaarde-problemen</u>, ontstaan als in de extra condities functie-waarden en waarden van de afgeleiden voorkomen in meer dan één punt.
Wij zullen ons voorlopig beperken tot beginwaarde-problemen.

Opgaven

112) Integreer de differentiaal-vergelijking

$$y' = x^2y + 5x^2.$$

Aanwijzing: integreer eerst het homogene deel $y' = x^2y$, vervang in de oplossing hiervan de integratie-constante c door een functie c(x) en bepaal c(x) zodanig, dat aan de gegeven vergelijking wordt voldaan.

113) Integreer evenzo de differentiaal-vergelijking

$$y' = y + e^{-2x}.$$

114) Bereken in 5 decimalen de oplossing van de volgende stelsels lineaire vergelijkingen met behulp van de methode van Crout.

$$\begin{pmatrix} 1 & 2 & -12 & 8 \\ 5 & 4 & 7 & -2 \\ -3 & 7 & 9 & 5 \\ 6 & -12 & -8 & 3 \end{pmatrix} x = \begin{pmatrix} 27 \\ 4 \\ 11 \\ 49 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & -2 & 8 & 4 \\ 2 & 1 & -4 & 7 \\ 4 & 2 & -8 & -4 \end{pmatrix} x = \begin{pmatrix} 20 \\ 26 \\ 10 \\ 2 \end{pmatrix}$$

115) Los het volgende stelsel op met behulp van de relaxatie-methode in 3 decimalen nauwkeurig.

$$\begin{aligned} x_1 + 10x_2 + x_3 &= 10 \\ 2x_1 + 20x_3 + x_4 &= 10 \\ 3x_2 + 30x_5 + 3x_6 &= 0 \\ 10x_1 + x_2 - x_6 &= 5 \\ 2x_4 - 2x_5 + 20x_6 &= 5 \\ x_3 + 10x_4 - x_5 &= 0. \end{aligned}$$

Intermezzo over functies van twee veranderlijken (pag. 216A-D)

Gaat men een functie van meer dan een veranderlijke differentiëren, dan zal men moeten aangeven, naar welke veranderlijke men differentieert. De andere veranderlijken worden daarbij constant gedacht en men spreekt van partiële afgeleiden. Nauwkeuriger gezegd:

Definitie. De partiële afgeleiden van een functie $g = g(x,y)$ worden gedefiniëerd als volgt:

$$g_x = \frac{\partial g}{\partial x} = \text{def.} \lim_{h \to 0} \frac{g(x+h,y) - g(x,y)}{h}$$

$$g_y = \frac{\partial g}{\partial y} = \text{def.} \lim_{k \to 0} \frac{g(x,y+k) - g(x,y)}{k}$$

Hiernaast te spreken van de gewone afgeleide $\frac{dg}{dx}$ heeft alleen zin, als y een functie van x is. Scrijven we deze functie als $y = y(x)$, dan geldt

$$\frac{dg}{dx} = \lim_{h \to 0} \frac{g(x+h,y(x+h)) - g(x,y(x))}{h} .$$

Het verband tussen deze gewone afgeleide en de partiële afgeleiden wordt vastgelegd in de volgende

Stelling. Als in een zeker gebied van het (x,y)-vlak de partiële afgeleiden g_x en g_y bestaan en continu zijn en als bovendien $\frac{dy}{dx}$ bestaat, dan geldt voor dit gebied

$$\frac{dg}{dx} = g_x + \frac{dy}{dx} \quad g_y .$$

Bewijs.

$$\frac{dg}{dx} = \lim_{h \to 0} \frac{g(x+h,y(x+h)) - g(x,y(x))}{h} =$$

$$= \lim_{h \to 0} \frac{g(x+h,y(x+h)) - g(x+h,y(x))}{h} +$$

$$+ \lim_{h \to 0} \frac{g(x+h, y(x)) - g(x, y(x))}{h} .$$

De tweede limiet is kennelijk gelijk aan $g_x(x,y)$; de eerste limiet is volgens de middelwaarde-stelling (wegens het bestaan van g_y) gelijk aan

$$\lim_{h \to 0} \frac{g_y(x+h, p(h))\ (y(x+h) - y(x))}{h} ,$$

waarbij p(h) een punt tussen y(x) en y(x+h) is. Wegens de continuïteit van y(x) convergeert p(h) naar y(x) en de limiet is dus gelijk aan

$$\lim_{h \to 0} g_y(x+h, p(h)) \times \lim_{h \to 0} \frac{y(x+h) - y(x)}{h} = g_y(x,y) \times \frac{dy}{dx}$$

waarmee de stelling bewezen is.

<u>Hogere afgeleiden</u>

De partiële afgeleiden g_x en g_y zijn ook functies van x en y en hiervan kunnen we dus weer de partiële afgeleiden vormen, mits de betreffende limieten bestaan.
Zo hebben we

$$g_{xx} = \frac{\partial^2 g}{\partial x^2} = \text{def.} \frac{\partial}{\partial x} \left(\frac{\partial g}{\partial x}\right),$$

$$g_{xy} = \frac{\partial^2 g}{\partial x \partial y} = \text{def.} \frac{\partial}{\partial y} \left(\frac{\partial g}{\partial x}\right), \text{ enz.}$$

Hierbij is van belang de volgende

<u>Stelling</u>. Als in een zeker gebied van het (x,y)- vlak de partiële afgeleiden g_{xy} en g_{yx} beide bestaan en continu zijn, dan zijn zij aan elkaar gelijk.

Bewijs.

$$g_{xy} = \lim_{k\to 0} \frac{g_x(x,y+k) - g_x(x,y)}{k}$$

$$= \lim_{k\to 0}\lim_{h\to 0} \frac{[g(x+h,y+k) - g(x,y+k)] - [g(x+h,y) - g(x,y)]}{hk}.$$

Volgens de middelwaarde-stelling toegepast op $g(x+h,y) - g(x,y)$, opgevat als functie van y, is dit gelijk aan

$$\lim_{k\to 0}\lim_{h\to 0} \frac{g_y(x+h,y+\theta_1 k) - g_y(x,y+\theta_1 k)}{h},$$

waarbij $0 < \theta_1 < 1$.

Nu passen we op $g_y(x,y+\theta_1 k)$, opgevat als functie van x, weer de middelwaarde-stelling toe en vinden dan dat deze limiet, voor zekere θ_2 tussen 0 en 1, gelijk is aan

$$\lim_{k\to 0}\lim_{h\to 0} g_{yx}(x + \theta_2 h,\ y + \theta_1 k).$$

Deze limiet is wegens de continuïteit van g_{yx} gelijk aan $g_{yx}(x,y)$, waarmee de stelling bewezen is.

Met behulp van deze stellingen vinden we de hogere gewone afgeleiden. Passen we dit toe op de differentiaal-vergelijking $y' = f(x,y)$, dan krijgen we

$$y'' = \frac{df}{dx} = f_x + ff_y.$$

$$y''' = \frac{d^2f}{dx^2} = \frac{\partial}{\partial x}\left(\frac{df}{dx}\right) + \frac{dy}{dx}\frac{\partial}{\partial y}\left(\frac{df}{dx}\right)$$

$$= \frac{\partial}{\partial x}(f_x + ff_y) + f\frac{\partial}{\partial y}(f_x + ff_y)$$

$$= f_{xx} + ff_{yx} + f_xf_y + ff_{xy} + f^2f_{yy} + ff_y^2$$

$$= f_{xx} + 2ff_{xy} + f^2f_{yy} + f_xf_y + ff_y^2.$$

Deze formules worden gebruikt voor het opstellen van de Taylor-reeks van y(x). Zie hieronder (1.1.2) en (1.2.10).

Opgave

116) Bereken bij de differentiaal-vergelijking $y' = f(x,y)$ alle partiële afgeleiden van f van de orde een en twee en de afgeleiden van y tot en met orde drie voor de volgende rechterlidfuncties f(x,y).

a) $x^2 - y^2 + 7xy$

b) $x - y^2$

c) $1/(x + y)$

d) $x^3 - 11x^2y + \cos(x)$.

1. Numerieke oplossing van beginwaarde-problemen

1.1 Taylor-reeks methode

Zij gegeven het begin-waarde probleem

$$1.1.0 \qquad \begin{cases} y' = f(x,y) \\ y(x_0) = y_0 \end{cases}$$

Wij willen nu weten $y(x_0 + h)$ en ontwikkelen hiertoe y in een Taylor-reeks

$$1.1.1 \quad y(x_0+h) = y(x_0) + hy'(x_0) + \frac{h^2}{2} y''(x_0) + \ldots + \frac{h^m}{m!} y^{(m)}(x_0) + \frac{h^{m+1}}{(m+1)!} f^{(m+1)}(\xi),$$

waarbij ξ tussen x_0 en x_0+h ligt.

Uit het gegeven probleem (1.1.0) volgt:

$$1.1.2 \qquad \begin{cases} y(x_0) = y_0 \\ y'(x_0) = f(x_0,y_0) \\ y''(x_0) = \left[\frac{d}{dx} f(x,y)\right]_{x=x_0,y=y_0} = (f_x + f_y f)_{x=x_0,y=y_0} \\ \text{en algemeen} \\ y^{(k+1)}(x_0) = \left[\frac{d^k}{dx^k} f(x,y)\right]_{x=x_0,\ y=y_0}. \end{cases}$$

Op deze wijze berekenen we de afgeleiden tot en met een zekere orde m en verkrijgen daarmee een benadering voor $y(x_0+h)$ volgens (1.1.1).

De stap h moet daarbij zo klein gekozen worden, dat de fout verwaarloosbaar is.

1.1.3 Definitie

De orde van een integratie-formule is per definitie een lager dan de h-exponent in de fout-term. Dus de orde van formule (1.1.1) is gelijk aan m.

N.B. Verwar niet orde van een formule met orde van een differentiaal-vergelijking.

In het bijzonder hebben we de eerste orde formule

1.1.4 $$y(x_0 + h) = y_0 + hf(x_0,y_0) + O(h^2) \ .$$

Deze formule wordt ook wel formule van Euler genoemd.

En de tweede orde formule

1.1.5 $$y(x_0 + h) = y_0 + hf(x_0,y_0) + \frac{h^2}{2} y''(x_0) + O(h^3)$$

Heeft men zo $y(x_0 + h)$ berekend voor $h = h_0$, dan bepaalt men uitgaande van $x_1 = x_0 + h_0$ en $y_1 = y(x_0 + h_0)$ vervolgens $y_2 = y(x_1 + h_1)$, waarbij h_1 niet gelijk aan h_0 behoeft te zijn.
Zo voortgaande zetten we telkens een stapje van (variabele) grootte h_i en berekenen voor het argument $x_{i+1} = x_i + h_i$ de waarde van y met behulp van de Taylor-ontwikkeling in het punt x_i.

Voorbeeld. Los op met formule (1.1.5) en h = 0.1 het beginwaarde-probleem:

$y' = -2xy^2$, $y(0) = 1$, exacte oplossing $y(x) = 1/(1+x^2)$.

Dan geldt dus:

$$y'' = -2y^2 - 4xyy' .$$

x	y	y'	y"/2	fout x10^3
.0	1.000	.000	-1.000	0
.1	.990	-.196	- .941	0
.2	.961	-.369	- .782	-1
.3	.916	-.503	- .562	-1
.4	.860	-.592	- .332	-2
.5	.798	-.637	- .129	-2
.6	.733	-.645	+ .030	-2
.7	.669	-.627	+ .139	-2
.8	.608	-.592	+ .206	-2
.9	.551	-.546	+ .238	-1
1.0	.499			-1

Ter controle kan men ook een stap terug doen, m.a.w. h door -h vervangen. Bijvoorbeeld:

$$y(0.4) = 0.798+0.0637-0.00129 \approx .860.$$

Voor- en nadelen

Het voordeel van de Taylor-reeks methode is, dat men in elke stap de staplengte h vrij kan kiezen. Opeenvolgende stappen zijn slechts verbonden door de functie-waarde, die het resultaat is van de voorafgaande en het beginpunt is voor de volgende stap. Dit soort methodes heten een-stap methodes.

Hiernaast bestaan zgn. meer-stap methodes, die in elke stap voor het berekenen van een nieuwe functie-waarde, meer dan een voorafgaande functie-waarde nodig hebben. Een eenvoudige formule van dit type wordt verkregen door van (1.1.5) de overeenkomstige formule voor $y(x_0 \pm h)$ af te trekken:

1.1.6 $$y(x_0-h) = y_0 - hf(x_0,y_0) + \frac{h^2}{2} y''(x_0) + O(h^3).$$

De formule die dan ontstaat is van de orde twee en luidt:

1.1.7 $$y(x_0+h) = y(x_0-h) + 2hf(x_0,y_0) + O(h^3)$$

Vergelijken we deze formule met (1.1.4), dan zien we, dat beide formules een gelijke hoeveelheid rekenwerk vergen terwijl de orde van (1.1.7) een hoger is.
Deze formule heeft echter twee voorafgaande functie-waarden $y(x_0)$ en $y(x_0-h)$ nodig, zodat in opeenvolgende stappen de staplengte h niet zonder meer gewijzigd kan worden. Later komen we op de meer-stap formules terug.

Het nadeel van de Taylor-reeks methode is, dat voor de formules van de orde < 1 afgeleiden van de rechter-functie f(x,y) nodig zijn. Deze afgeleiden worden bij hogere orde meestal ingewikkeld.
Een hogere orde Taylor formule is daarom slechts bruikbaar ofwel voor zeer eenvoudige differentiaal-vergelijkingen ofwel om een rij start-waarden voor een meer-stap formule te berekenen. De eerste orde formule (1.1.4) is wel altijd bruikbaar, maar heeft het nadeel, dat de gebruiker een kleine staplengte moet kiezen om nog een redelijke precisie te bereiken.

1.2 Runge-Kutta methodes

Deze methodes hebben enorme belangstelling gekregen sinds de opkomst van de automatische rekenmachines. Ze zijn van het een-stap type en hebben dus het voordeel, dat de staplengte h vrij kiesbaar is. Een ander voordeel is, dat ze geen afgeleiden van de rechterlid-functie nodig hebben en desondanks een redelijk hoge orde kunnen hebben.
Wij beschouwen weer het eerste orde beginwaarde-probleem (1.1.0).
Het idee van Runge en Kutta is de rechterlid-functie f(x,y) te berekenen in enige punten van het (x,y)-vlak gelegen in de buurt van (x_0,y_0). Van deze f-waarden wordt dan een lineaire combinatie genomen, die zo goed mogelijk met de Taylor-reeks van $\Delta y = y(x_0+h) - y_0$ overeenstemt.

In formule

$$1.2.1 \qquad y(x_0+h) = y_0 + a_0k_0 + a_1k_1 + \ldots + a_{p-1}k_{p-1},$$

waarbij

$$1.2.2 \qquad \begin{cases} k_0 = hf(x_0,y_0) \\ k_1 = hf(x_0 + m_1h,\ y_0 + l_{10}\ k_0) \\ k_2 = hf(x_0 + m_2h,\ y_0 + l_{20}k_0 + l_{21}k_1) \\ \text{en algemener voor } i = 1(1)p-1 \\ k_i = hf(x_0 + m_ih,\ y_0 + l_{i0}k_0 + \ldots + l_{ii-1}k_{i-1}) \end{cases}$$

Het getal p heet de <u>rang</u> van de Runge-Kutta formule en geeft aan hoe vaak in een stap de rechterlid-functie berekend moet worden. De rang is dus een maat voor de hoeveelheid rekenwerk, mits het rechterlid $f(x,y)$ niet al te eenvoudig is. Bij gegeven rang p worden de coëfficiënten a_i, m_i en l_{ij} nu zo gekozen, dat de orde van de formule zo hoog mogelijk wordt. Hiertoe wordt de Runge-Kutta formule (1.2.1) vergeleken met de Taylor-ontwikkeling van $y(x_0+h)$. Stemmen zij overeen tot en met de m-de term van de Taylor-reeks, m.a.w. is de fout van de orde h^{m+1}, dan is de Runge-Kutta formule van de orde m (vgl. definitie 1.1.3). Het bepalen van de maximale orde m bij gegeven rang p en het berekenen van bijbehorende waarden van de coëfficiënten a_i, m_i, l_{ij} leidt bij hogere waarden van p tot lastige stelsels algebraïsche vergelijkingen. Wij zullen alleen lage p-waarden beschouwen en enige voorbeelden geven.

Wij hebben hierbij nodig de formule van de Taylor-reeks voor een functie met twee variabelen (vgl. pag. 169):

$$1.2.3 \qquad f(x+h,y+k) = f(x,y) + hf_x + kf_y + \tfrac{1}{2}(h^2f_{xx} + 2hkf_{xy} + k^2f_{yy}) + \ldots .$$

Rang p = 1

In dit geval bestaat Δy uit slechts één term a_0k_0 en om overeenstemming met de Taylor-reeks te krijgen moet natuurlijk a_0 gelijk aan een zijn. We krijgen dus als resultaat formule (1.1.4) van de orde een.

Rang p = 2

In dit geval hebben we

$$y(x_0+h) = y_0 + a_0k_0 + a_1k_1$$

1.2.4 $$k_0 = hf(x_0,y_0)$$

$$k_1 = hf(x_0 + m_1h, y_0 + l_{10}k_0).$$

Hieruit volgt met behulp van (1.2.3), waarin we h door m_1h en k door $l_{10}hf(x_0,y_0)$ vervangen:

1.2.5 $$y(x_0+h) = y_0 + a_0hf + a_1hf + a_1h^2m_1f_x + a_1h^2l_{10}ff_y + \ldots,$$

waarbij f en de partiële afgeleiden worden genomen in het punt (x_0,y_0). Om overeenstemming met de Taylor-reeks te hebben, moet dus gelden

$$a_0 + a_1 = 1$$

$$a_1m_1 = \tfrac{1}{2}$$

$$a_1l_{10} = \tfrac{1}{2}$$

Dus $m_1 = l_{10}$. Verder kunnen we m_1 willekeurig $\neq 0$ kiezen en vinden dan als algemene oplossing

$$a_1 = 1/2m_1 \quad , \quad a_0 = 1 - a_1.$$

De h^3-term blijkt niet in overeenstemming met de Taylor-reeks te maken te zijn, zodat de hoogst bereikbare orde in dit geval 2 is.

Voor de hand liggende keuzen van m_1 en bijbehorende Runge-Kutta formules van de tweede orde zijn

1.2.6 $m_1 = 1$: $y(x_0+h) \approx y_0 + \frac{1}{2} h\left[f(x_0,y_0) + f(x_0+h, y_0+hf(x_0,y_0))\right]$

1.2.7 $m_1 = \frac{1}{2}$: $y(x_0+h) \approx y_0 + hf(x_0+\frac{1}{2}h,\ y_0+\frac{1}{2}hf(x_0,y_0))$.

Rang $p = 3$

De ontwikkeling van k_1 moet nu nog een orde verder voortgezet worden, dus

1.2.8 $k_1 = hf + h^2 m_1 f_x + h^2 l_{10} f f_y + \frac{1}{2}h^3(m_1^2 f_{xx} + 2m_1 l_{10} f f_{xy} + l_{10}^2 f^2 f_{yy})$.

Vervolgens moet k_2 ontwikkeld worden:

1.2.9 $k_2 = hf + h^2 m_2 f_x + h^2 l_{20} f f_y + h^2 l_{21}(f + hm_1 f_x + kl_{10} f f_y) f_y$

$+ \frac{1}{2}h^3(m_2^2 f_{xx} + 2m_2(l_{20} + l_{21}) f f_{xy} + (l_{20} + l_{21})^2 f^2 f_{yy}) + \ldots$.

Dit moet overeenstemmen met de Taylor-reeks

1.2.10 $y(x_0+h) = y_0 + hy'(x_0) + \frac{1}{2}h^2 y''(x_0) + \frac{1}{6} h^3 y'''(x_0) + \ldots$

$= y_0 + hf + \frac{1}{2}h^2(f_x + f f_y) + \frac{1}{6} h^3(f_{xx} + 2f f_{xy} + f^2 f_{yy}$

$+ f_x f_y + f f_y^2) + \ldots$.

Van elke term stellen we de coëfficiënt in de Runge-Kutta formule (1.2.1) en in de Taylor-reeks (1.2.10) aan elkaar gelijk en krijgen dan het volgende stelsel algebraïsche vergelijkingen.

term	coëfficiënt in RK-formule	=	coëfficiënt in Taylor-reeks
hf	$a_0 + a_1 + a_2$	=	1
$h^2 f_x$	$a_1 m_1 + a_2 m_2$	=	$\frac{1}{2}$
$h^2 f f_y$	$a_1 l_{10} + a_2(l_{20} + l_{21})$	=	$\frac{1}{2}$
$\frac{1}{2}h^3 f_{xx}$	$a_1 m_1^2 + a_2 m_2^2$	=	$\frac{1}{3}$
$h^3 f f_{xy}$	$a_1 m_1 l_{10} + a_2 m_2(l_{20} + l_{21})$	=	$\frac{1}{3}$
$\frac{1}{2}h^3 f^2 f_{yy}$	$a_1 l_{10}^2 + a_2(l_{20} + l_{21})^2$	=	$\frac{1}{3}$
$h^3 f_x f_y$	$a_2 l_{21} m_1$	=	$\frac{1}{6}$
$h^3 f f_y^2$	$a_2 l_{21} l_{10}$	=	$\frac{1}{6}$

$\Rightarrow m_2 = l_{20} + l_{21}$ (b)

$\Rightarrow m_1 = l_{10}$ (a)

Hieruit volgen meteen de gelijkheden (a) en (b), waarna het stelsel zich reduceert tot het equivalente stelsel

1.2.11
$$\begin{cases} a_0 + a_1 + a_2 = 1 \\ a_1 m_1 + a_2 m_2 = \frac{1}{2} \\ a_1 m_1^2 + a_2 m_2^2 = \frac{1}{3} \\ a_2 l_{21} m_1 = \frac{1}{6} \end{cases}$$

Dit stelsel heeft oneindig veel oplossingen (m_1 en m_2 kunnen vrijwel willekeurig gekozen worden). Twee bijzondere gevallen leiden tot de volgende formules van de orde drie:

Formule van Heun

1.2.12
$$\begin{cases} y(x_0+h) = y_0 + \frac{1}{4}(k_0 + 3k_2) + O(h^4) \\ k_0 = hf(x_0, y_0) \\ k_1 = hf(x_0 + \frac{1}{3}h,\ y_0 + \frac{1}{3}k_0) \\ k_2 = hf(x_0 + \frac{2}{3}h,\ y_0 + \frac{2}{3}k_1) \end{cases}$$

Formule van Kutta

1.2.13
$$\begin{cases} y(x_0+h) = y_0 + \frac{1}{6}(k_0 + 4k_1 + k_2) + O(h^4) \\ k_0 = hf(x_0, y_0) \\ k_1 = hf(x_0 + \frac{1}{2}h,\ y_0 + \frac{1}{2}k_0) \\ k_2 = hf(x_0 + h,\ y_0 - k_0 + 2k_1) \end{cases}$$

Rang p = 4

In dit geval is de hoogste bereikbare orde 4.
Zonder afleiding geven we hier de volgende formule van Kutta

1.2.14
$$\begin{cases} y(x_0+h) = y_0 + \frac{1}{6}(k_0 + 2k_1 + 2k_2 + k_3) + O(h^5) \\ k_0 = hf(x_0, y_0) \\ k_1 = hf(x_0 + \frac{1}{2}h,\ y_0 + \frac{1}{2}k_0) \\ k_2 = hf(x_0 + \frac{1}{2}h,\ y_0 + \frac{1}{2}k_1) \\ k_3 = hf(x_0 + h,\ y_0 + k_2) \end{cases}$$

Opmerking

Als de rechterlid-functie niet van y afhangt, m.a.w. als de gegeven differentiaal-vergelijking luidt: $y' = f(x)$, dan hebben we kennelijk te doen met gewone integratie oftewel quadratuur.
De bovengenoemde Runge-Kutta formules gaan voor dit geval dus over in quadratuur-formules. Zo gaat (1.2.6) over in de trapezium-regel, en (1.2.13) en (1.2.14) gaan beide over in de formule van Simpson.
Ga dit na.

Opgaven

117) Gegeven het beginwaarde-probleem $y' = x - y^2$, $y(0) = 1$.
Gevraagd de oplossing $y(x)$ voor $x = 0(0.2)1$ in 4 decimalen te berekenen met de Taylor-reeks methode.
Bereken tevens (door interpolatie) plaats en grootte van het minimum van $y(x)$.

118) Gegeven het beginwaarde-probleem $y' = 1/(x+y)$, $y(0) = 1$.
Gevraagd de oplossing $y(x)$ voor $x = 0(0.5)2$ te berekenen in 4 decimalen met behulp van de Runga-Kutta formule (1.2.14).

Runge Kutta met Richardson-correctie

Evenals bij gewone integratie kunnen we bij een Runge-Kutta-formule de Richardson correctie-term uitrekenen. Hiertoe integreren we over 2 stappen h en 1 stap 2h. Als voorbeeld beschouwen we de vierde-orde formule (1.2.14). Als $\Delta y_i(h)$ de Runge-Kutta bijdrage in het punt $x_i = x_0 + ih$ met stap-grootte h voorstelt, krijgen we

1.2.15 $$y(x_0 + 2h) = y_0 + \Delta y_0(2h) + 32Ch^5 + O(h^6)$$

$$y(x_0 + h) = y_0 + \Delta y_0(h) + Ch^5 + O(h^6)$$

$$y(x_0 + 2h) = y(x_0 + h) + \Delta y_1(h) + Ch^5 + O(h^6), \text{ dus}$$

1.2.16 $$y(x_0 + 2h) = y_0 + \Delta y_0(h) + \Delta y_1(h) + 2Ch^5 + O(h^6).$$

Aftrekken levert

$$0 = \Delta y_0(h) + \Delta y_1(h) - \Delta y_0(2h) - 30Ch^5 + O(h^6)$$

De correctie-term voor (1.2.16) moet overeenkomen met $2Ch^5$ en de gecorrigeerde formule luidt dus

1.2.17 $$y(x_0 + 2h) = y_0 + \Delta y_0(h) + \Delta y_1(h) + \frac{\Delta y_0(h) + \Delta y_1(h) - \Delta y_0(2h)}{15} + O(h^6)$$

Deze formule is van de orde 5; het aantal benodigde functie-berekeningen bedraagt $3 \times 4 - 1 = 11$.

Fröberg's stap-halvering en -verdubbeling

Daar de Runge-Kutta formules van het een-stap type zijn, kunnen we de stapgrootte h vrij variëren. We kiezen nu h zó, dat de correcties beneden een zekere tolerantie blijven. Zorgen we, bij gegeven ε, dat in elke stap de correctie-term in absolute waarde kleiner is dan $h \times \varepsilon$, dan is de totale correctie, integrerend van a naar b, kleiner dan $(b-a) \times \varepsilon$.

In Fröberg's procédé wordt nu, als de correctie-term te groot is, h gehalveerd en opnieuw geïntegreerd; anders wordt de stap geaccepteerd en door-geïntegreerd, waarbij echter eerst nog wordt overwogen de stap te verdubbelen.
Verdubbeling kan geschieden als de correctie-term al te klein wordt (bv. $< \frac{1}{20}$ maal de toelaatbare waarde), of als vaak genoeg achtereenvolgende stappen geaccepteerd zijn.
Dit procédé is beschreven in de procedure RK in het Revised report on ALGOL 60. Hierin wordt evenwel de Richardson correctie wel gebruikt om h aan te passen, maar niet om de vierde-orde Runge-Kutta-formule te corrigeren tot een vijfde-orde formule.

Een procedure van Zonneveld

Zonneveld heeft zowel de berekening van de correctie-term als het aanpassen van de stap-grootte vereenvoudigd.
Wat het eerste betreft, bij het opstellen van een Runge-Kutta-formule (zie 1.2.1 en 1.2.2) van de orde m kan men trachten de som der hoogste orde termen (dus de termen van de orde h^m), te schrijven als een lineaire combinatie.

$$b_0k_0 + b_1k_1 + \ldots + b_{p-1}k_{p-1}.$$

Bij de tweede orde formules (1.2.6 en 1.2.7) lukt dit zonder meer.
Bij hogere orde hebben we hiervoor minstens één extra k_i nodig, m.a.w. we moeten de rang verhogen.
Het op te lossen algebraïsche stelsel heeft nu p extra onbekenden, nl. de b's, en het aantal vergelijkingen is verdubbeld. We vermelden slechts een enkele oplossing, behorende bij formule (1.2.12).

1.2.12.a
$$\begin{cases} k_3 = hf(x_0 + h,\ y_0 + \frac{1}{4}(k_0 + 3k_2)) \\ \text{som der } h^3\text{-termen} = \frac{1}{2}(k_0 - 3k_2 + 2k_3). \end{cases}$$

De waarde van y nodig voor het berekenen van k_3 is hier juist de berekende waarde van $y(x_0 + h)$, die we toch nodig hebben. Bij hogere orde formule lukt dit niet meer.
In de onderstaande procedure wordt een formule gebruikt van de orde 5 waarin 7 functie-berekeningen nodig zijn, nl. 6 voor de bepaling van Δy en nog een extra voor de som der h^5-termen. Deze formule betekent dus een aanzienlijke besparing vergeleken bij de boven geschetste berekening van de Richardson-correctie, waar totaal 11 functie-berekeningen nodig waren.
De som der hoogste orde termen is nu te beschouwen als de correctie-term, die kan worden gebruikt om de stap-grootte aan te passen.
Beter dan het vrij grove halverings- en verdubbelingsproces is het, na iedere stap te schatten hoe groot de volgende stap moet zijn om juist onder de gevraagde tolerantie te blijven.
De tolerantie "tol" nemen we evenredig met h, laten we zeggen

$$tol = h \times \varepsilon$$

en voor een vijfde-orde formule is de correctie-term "discr" (bij benadering) evenredig met h^5, dus

$$discr \approx Ch^5$$

We willen nu h_1 zo kiezen, dat daarvoor discr ongeveer gelijk aan tol is, dus

$$Ch_1^5 = h_1 \times \varepsilon,$$

dus
$$h_1 = \sqrt[4]{\varepsilon/C} = h \sqrt[4]{tol/discr}.$$

We nemen nog een veiligheidsmarge van 5% en krijgen dan

1.2.18
$$h_1 = 0.95\ h \sqrt[4]{tol/discr}.$$

Er zijn nu in elke stap 2 mogelijkheden:

discr > tol: dan wordt de stap verworpen en opnieuw geïntegreerd met h_1 berekend volgens (1.2.18).
discr $\leq$ tol: dan wordt de stap aanvaard en doorgeïntegreerd eveneens met h_1 uit (1.2.18)
Voor functies die sterk van karakter variëren kan de veiligheidsmarge van 5% te klein zijn. Daarom is het nog mooier, de nieuwe stap-lengte te extrapoleren uit de twee laatste h-waarden. Dit wordt in onderstaande procedure gedaan, waarbij voor de vierdemachts-wortel een lineaire benadering gebruikt wordt.

Description AP 252

RK1 can be used to integrate the equation dy/dx = f(x,y).
First we explain the actual parameters corresponding to the formal parameters:

x : the independent variable; upon completion of a call of RK1, it is equal to b;
a : the starting value of x;
b : a value parameter, giving the end value of x;
y : the dependent variable;
ya : the value of y at x=a;
fxy: an expression, depending on x and y, giving the value of dy/dx;
e : an array of positive tolerances, consisting of e[1] and e[2] ; e[1] is used as a relative tolerance, e[2] as an absolute one; the tolerance in a quantity z is defined, here and in the sequel, as

tolerance(z) = abs(z) * e[1] + e[2] ; (1)

d : an array with elements d[1] , ..., d[4] ; upon completion of a call of RK1:
entier (d[1] + .5) is the number of steps skipped;
d[2] is the step length;
d[3] is equal to b;
d[4] is equal to y(b);

fi : a Boolean; if fi then the integration starts at a, with trial step b-a; if ¬fi then the integration is continued with x=d[3], y=d[4], h=d[2] * sign (b-d[3]) as initial conditions; a and ya are ignored.

RK1 integrates dy/dx=fxy to x=b, with if fi then x=a, y(a) = ya; if ¬fi then x=d[3], y(d[3])=d[4].
Upon completion of a call of RK1 we have x=d[3]=b, y=d[4]=y(b).
RK1 uses as its minimal absolute step length
hmin=e[1] * int+e[2],
where int=abs(b-(if fi then a else d[3])).

If a step of length abs(h) ≤ hmin is rejected, a step sign(h) * hmin is skipped; a step is rejected if
$th^5dy > (abs(fxy) * e[1]+e[2]) * abs(h)/int$.
Procedure RK1 has been copied from J.A. Zonneveld, Automatic Numerical Integration (thesis, Amsterdam 1964).

```
comment AP 252;
procedure RK1(x,a,b,y,ya,fxy,e,d,fi); value b,fi; real x,a,b,y,ya,fxy;
Boolean fi; array e,d;
begin real e1,e2,x1,y1,h,int,hmin,absh,k0,k1,k2,k3,k4,k5,discr,tol,mu,
    mu1,fh,hl;
    Boolean last,first,reject;
    if fi then begin d[3]:= a; d[4]:= ya end; d[1]:= 0; x1:= d[3];
    y1:= d[4]; if fi then d[2]:= b - d[3]; absh:= h:= abs(d[2]);
    if b - x1 < 0 then h:= - h; int:= abs(b - x1);
    hmin:= int × e[1] + e[2];
    e1:= e[1]/int; e2:= e[2]/int; first:= true;
    if fi then begin last:= true; goto step end;
test:  absh:= abs(h); if absh < hmin then
    begin h:= if h > 0 then hmin else - hmin; absh:= hmin end;
    if h ≥ b - x1 = h ≥ 0 then
    begin d[2]:= h; last:= true; h:= b - x1; absh:= abs(h) end
    else last:= false;
step: x:= x1; y:= y1; k0:= fxy × h; x:= x1 + h × 2/9; y:= y1 + k0 × 2/9;
    k1:= fxy × h; x:= x1 + h/3; y:= y1 + (k0 + k1 × 3)/12;
    k2:= fxy × h; x:= x1 + h × .5;
    y:= y1 + (k0 + k2 × 3)/8; k3:= fxy × h; x:= x1 + h × .8;
    y:= y1 + (k0 × 53 - k1 × 135 + k2 × 126 + k3 × 56)/125;
    k4:= fxy × h; x:= if last then b else x1 + h;
    y:= y1 +(k0 × 133 - k1 × 378 + k2 × 276 + k3 × 112 + k4 × 25)/168;
    k5:= fxy × h;
    discr:=abs(k0 × 21 - k2 × 162 + k3 × 224 - k4 × 125 + k5 × 42)/14;
    tol:= abs(k0) × e1 + absh × e2; reject:= discr > tol;
    mu:= tol/(tol + discr) + .45; if reject then
    begin if absh < hmin then
        begin d[1]:= d[1] + 1; y:= y1; first:= true; goto next end;
        h:= mu × h; goto test
    end;
    if first then
    begin first:= false; hl:= h; h:= mu × h; goto acc end;
    fh:= mu × h/hl + mu - mu1; hl:= h; h:= fh × h;
acc:   mu1:= mu;
    y:= y1 + (- k0 × 63 + k1 × 189 - k2 × 36 - k3 × 112 + k4 × 50)/28;
    k5:= fxy × hl;
    y:= y1 + (k0 × 35 + k2 × 162 + k4 × 125 + k5 × 14)/336;
next:  if b ≠ x then begin x1:= x; y1:= y; goto test end;
    if ¬ last then d[2]:= h; d[3]:= x; d[4]:= y
end RK1;
```

Opgaven

119) Schrijf een Algol-programma, dat een van de volgende beginwaarde-problemen oplost met behulp van de procedure RK1 (AP 252).

a) $y' = -2y$, $y(0) = 1$

b) $y' = x^2 - y^2$, $y(0) = 0$

Kies geschikte relatieve en absolute toleranties en laat de oplossing printen voor $x = 0(0.1)2$.

120) Gegeven is het beginwaarde-probleem

$$y' = (x + y)^2$$

$$y(0) = 1$$

Bereken $y(0.05)$ in 6 decimalen door numerieke integratie met behulp van Taylor-ontwikkeling in stappen van $h = 0.01$.

121) Bereken met een derde-orde Runge-Kutta formule de oplossing van het beginwaarde-probleem

$$y' = 2xy$$

$$y(0) = 1$$

in de punten $x = 0.02$, 0.06 en 0.10.

1.3 Hogere orde differentiaal-vergelijkingen en stelsels differentiaal-vergelijkingen

De een-stap methodes laten zich onmiddellijk uitbreiden tot het oplossen van hogere orde differentiaal-vergelijkingen en stelsels. Deze twee hangen trouwens zeer nauw met elkaar samen. Om dit te zien beschouwen we eerst de algemene vorm van een stelsel eerste orde differentiaal-vergelijkingen

$$1.3.1 \qquad \begin{cases} y_1' = f_1(x, y_1, \circ\circ\circ, y_n) \\ \circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ \\ y_n' = f_n(x, y_1, \circ\circ\circ, y_n) \end{cases}$$

Hierbij zijn $y_1, \circ\circ\circ, y_n$ de onbekende functies van x die gezocht worden. Als van al deze functies de waarde in één punt x_0 gegeven is, in formule

$$1.3.2 \qquad y_i(x_0) = \eta_i \quad , \quad i = 1(1)n,$$

dan spreken we van een beginwaarde-probleem (in stelsel-vorm). Nu kan een n-de orde differentiaal-vergelijking gemakkelijk herleid worden tot een stelsel eerste-orde vergelijkingen. Dit gaat als volgt. Zij gegeven de n-de orde vergelijking

$$1.3.3 \qquad y^{(n)} = f(x, y, y', \circ\circ\circ, y^{(n-2)}, y^{(n-1)}).$$

Stellen we nu $y = y_1, y' = y_2, \circ\circ\circ, y^{(n-2)} = y_{n-1}, y^{(n-1)} = y_n$ dan gaat deze vergelijking over in het equivalente stelsel

$$1.3.4 \qquad \begin{cases} y_1' = y_2 \\ y_2' = y_3 \\ \circ\circ\circ\circ\circ\circ \\ y_{n-1}' = y_n \\ y_n' = f(x, y_1, y_2, \circ\circ\circ, y_{n-1}, y_n). \end{cases}$$

Dit is dus een bijzonder geval van (1.3.1). Gaan we uit van een beginwaarde-probleem van de orde n (vgl. 0.7), m.a.w. zijn bij (1.3.3) gegeven de beginvoorwaarden

$$y^{(i)}(x_0) = \alpha_i \quad , \quad i = 0(1)n-1,$$

dan krijgen we blijkbaar een beginwaarde-probleem in stelsel-vorm.

De Taylor-reeks methode voor hogere orde vergelijkingen en stelsels is volkomen analoog aan (1.1). Men moet dan alle functies y_i ontwikkelen en steeds alle vergelijkingen een stap integreren. Voor de toepasbaarheid is wederom nodig, dat men over afgeleiden van de rechterlid-functies beschikt.
We beschouwen nu Runge-Kutta methodes voor stelsels differentiaalvergelijkingen (die dus de hogere orde vergelijkingen als bijzonder geval bevatten).
Het Runge-Kutta schema van een stelsel (1.3.1) met beginvoorwaarden (1.3.2) ziet er als volgt uit, als j = 1(1)n:

$$1.3.5 \qquad \begin{cases} y_j(x_0 + h) = \eta_j + a_0 k_{0j} + a_1 k_{1j} + \circ\circ\circ + a_{p-1} k_{p-1,j} \circ \\ k_{0j} = h f_j(x_0, \eta_1, \circ\circ\circ, \eta_n) \\ k_{1j} = h f_j(x_0 + m_1 h, \eta_1 + l_{10} k_{01}, \circ\circ\circ, \eta_n + l_{10} k_{0n}), \end{cases}$$

enz.

Dit wordt vergeleken met de Taylor-reeksen van y_j, wat weer leidt tot een stelsel algebraïsche vergelijkingen in a_i, m_i em l_{ij}, dat vrijwel hetzelfde is, als voor een enkele differentiaal-vergelijking. (Men neemt altijd $m_i = \sum_j l_{ij}$, wat tot een aanzienlijke vereenvoudiging leidt.)
Zo hebben we analoog aan (1.2.14) de volgende Runge-Kutta formule van de orde 4, voor een stelsel differentiaal-vergelijkingen (j = 1(1)n):

1.3.6

$$\begin{cases} y_j(x_0 + h) = \eta_j + \frac{1}{6}(k_{0j} + 2k_{1j} + 2k_{2j} + k_{3j}) + O(h^5) \\ k_{0j} = hf_j(x_0, \eta_1, \ldots, \eta_n) \\ k_{1j} = hf_j(x_0 + \frac{1}{2}h, \eta_1 + \frac{1}{2}k_{01}, \ldots, \eta_n + \frac{1}{2}k_{0n}) \\ k_{2j} = hf_j(x_0 + \frac{1}{2}h, \eta_1 + \frac{1}{2}k_{11}, \ldots, \eta_n + \frac{1}{2}k_{1n}) \\ k_{3j} = hf_j(x_0 + h, \eta_1 + k_{21}, \ldots, \eta_n + k_{2n}). \end{cases}$$

Voorbeeld (uit C.E. Fröberg, p. 245).

Gegeven is het tweede orde beginwaarde-probleem

$$y'' = x(y')^2 - y^2 \ , \ y(0) = 1 \ ; \ y'(0) = 0.$$

We stellen $y' = z$ en krijgen dan het equivalente stelsel

$$y' = z \qquad y(0) = 1$$

$$z' = xz^2 - y^2 \qquad z(0) = 0.$$

De eerste Runge-Kutta stap volgens (1.3.6, waarbij $n = 2$, $y_1 = y$, $y_2 = z$) ziet er, als $h = 0.2$, zo uit:

x	y	$z = f_1$	$xz^2 - y^2 = f_2$	k_{i1}	k_{i2}
0	1	0	-1	0	-0.2
0.1	1	-0.1	-0.999	-0.02	-0.1998
0.1	0.99	-0.0999	-0.979102	-0.01998	-0.1958204
0.2	0.98002	-0.1958204	-0.9527709	-0.039164	-0.1905542

0.2 0.980146 -0.196966

De eindwaarden van deze stap voor x, y , z staan onder de lijn en deze worden weer gebruikt als beginwaarden voor de volgende stap.

De Runge-Kutta formules voor stelsels differentiaal-vergelijkingen kunnen, evenals bovengeschetst voor een enkele vergelijking, worden gebruikt met Richardson-correctie en Fröberg's stap-halverings- en verdubbelings-techniek (zie de procedure RK in het Revised report on ALGOL 60).

De Runge-Kutta formules voor stelsels zijn eveneens zeer goed bruikbaar met Zonneveld's correctie-term en stap-extrapolatie.
(Voor enige ALGOL-procedures voor stelsels zie J.A. Zonneveld, thesis Amsterdam 1964).

Opgaven

122) De functie y(x) voldoet aan de differentiaal-vergelijking

$$y'' + x^2 y = 0,$$

terwijl $y(0) = 0$ en $y'(0) = 1$.
Bepaal ligging en waarde van het maximum van y(x) gelegen in de buurt van $x = 1.5$. Voer de berekening uit in 6 decimalen.

123) Gegeven het beginwaarde-probleem

$$y'' = xy,\ y(0) = 0.\ y'(0) = 1.$$

Bereken met behulp van reeks-ontwikkeling en ook met de Runge-Kutta-formule (1.3.6) in 4 decimalen nauwkeurig de waarden van y(x) voor $x = 0.5$ en $x = 1$.

124) Schrijf een ALGOL-programma, dat met behulp van de procedure RK uit het Revised report on ALGOL 60 een van de volgende beginwaarde-problemen oplost.

a) $\begin{cases} y' = xz + 1, & y(0) = 0 \\ z' = -xy, & z(0) = 1. \end{cases}$

Gevraagd y en z voor $x = 0.3,\ 0.6,\ 0.9$.

b) $y'' = (x^2 - y^2)/(1 + (y')^2),\ y(0) = 1,\ y'(0) = 0.$

Gevraagd y(x) voor $x = 0.5,\ 1.0,\ 1.5$.

1.4 Meer-stap formules met achterwaartse differenties

De een-stap formules, nl. de Taylorreeks en de Runge-Kutta formules, hebben het nadeel, dat men, om een hogere orde formule te krijgen, meer rekenwerk per stap moet doen. De meer-stap formules bereiken hogere orde door gegevens van verscheidene voorgaande stappen te benutten. Hierbij moet dan wel de staplengte constant gehouden worden.

We beschouwen weer het beginwaarde-probleem

1.4.1 $$y' = f(x,y) \quad , \quad y(x_0) = y_0$$

en nemen aan dat we op een aantal equidistante punten

1.4.2 $$x_i = x_0 + ih \quad (i = 0(1)n)$$

bijbehorende functiewaarden y_i hebben berekend.
We zoeken nu een formule, die bij $x_{n+1} = x_n + h$ een benaderde functiewaarde $y_{n+1} \approx y(x_{n+1})$ levert.

Uit (1.4.1) vinden we door integratie de equivalente integraal-vergelijking

1.4.3 $$y(x) = y_0 + \int_{x_0}^{x} f(t,y(t))dt.$$

Een iteratieve oplossings-methode is de volgende.
Kies als start de lineaire functie

1.4.4 $$y^{[0]}(x) = y_0 + f(x_0,y_0)(x - x_0)$$

en bereken dan voor $k = 0, 1, 2, \ldots$ achtereenvolgens

1.4.5 $$y^{[k+1]}(x) = y_0 + \int_{x_0}^{x} f(t,y^{[k]}(t))dt.$$

Deze methode heet methode van Picard.
Men kan bewijzen, als $f(x,y)$ aan een vrij zwakke eis voldoet, dat het beginwaarde-probleem (1.4.1) een eenduidige oplossing heeft en

dat de rij functie $y^{[k]}(x)$ naar deze oplossing convergeert. De methode van Picard is dus ongetwijfeld van theoretisch belang. Voor practische toepassing is de methode meestal niet geschikt, omdat, als f(x,y) niet al te eenvoudig is, de berekening der integralen spoedig te ingewikkeld wordt.

We gaan nu in formule (1.4.3) de daarin voorkomende integraal vervangen door een numerieke integratie-formule. Omdat, zoals we aannamen, x_i met y_i en dus ook de bijbehorende waarden van f slechts bekend zijn voor $i \leq n$, ligt het voor de hand achterwaartse differenties te gebruiken. Definiëren we

$$f_i = f(x_i,y_i) \text{ en } \nabla_i^k = \nabla^k f_i,\ k = 1, 2, \ldots$$

dan ziet het schema van bekende y- en f-waarden met de differenties van f er zo uit:

x_0	y_0	f_0				
			∇_1			
x_1	y_1	f_1	∘			
∘	∘	∘	∘			
∘	∘	∘	∘			
∘	∘	∘	∇_{n-2}			
x_{n-2}	y_{n-2}	f_{n-2}		∇_{n-1}^2 ∘		
			∇_{n-1}		∇_n^3 ∘	
x_{n-1}	y_{n-1}	f_{n-1}		∇_n^2		
			∇_n			
x_n	y_n	f_n				
x_{n+1}						

We benaderen f met de achterwaartse formule van Newton van de orde m (vgl. formule 18.2 pag. 77), nu echter niet in x_0 maar in x_n beginnend:

1.4.6 $$f(x_n + ph) \approx \sum_{k=0}^{m-1} \frac{p(p+1) \ldots (p+k-1)}{k!} \nabla^k f_n$$

$$= f_n + p\nabla_n + \frac{p(p+1)}{2} \nabla_n^2 + \frac{p(p+1)(p+2)}{6} \nabla_n^3 + \ldots .$$

Vervangen we in (1.4.3) de functie f door deze benadering en stellen we $t = x_n + ph$, dan krijgen we

$$y_{n+1} = y(x_n + h) = y_n + \int_{x_n}^{x+h} f(t,y(t))dt$$

$$= y_n + h \int_0^1 f(x_n + ph)dp$$

$$\approx y_n + h \sum_{k=0}^{m-1} \int_0^1 \frac{p(p+1) \ldots (p+k-1)}{k!} dp\ \nabla^k f_n .$$

We noemen de hiermee verkregen y-waarde y_{n+1}^A en hebben dus

1.4.7 $$y_{n+1}^A \approx y_n + h(f_n + \frac{1}{2} \nabla_n + \frac{5}{12} \nabla_n^2 + \frac{3}{8} \nabla_n^3 + \frac{251}{720} \nabla_n^4 + \ldots).$$

Dit is de formule van Adams. Omdat hierin alleen bekende waarden van y en f gebruikt worden, heet deze formule van het open type. Dit in tegenstelling tot de gesloten formules, die (een schatting van) f_{n+1} met bijbehorende differenties gebruiken.

Om een gesloten formule te krijgen gaan we uit van de achterwaartse Newton-formule (1.4.6), waarin we nu n door n+1 vervangen, stoppen deze benadering in (1.4.3) en krijgen dan met $t = x_{n+1} + ph$:

$$y_{n+1} = y_n + h \int_{-1}^0 f(x_{n+1} + ph)dp$$

$$= y_n + \sum_{k=0}^{m-1} \int_{-1}^0 \frac{p(p+1) \ldots (p+k-1)}{k!} dp\ \nabla^k f_{n+1} .$$

De hiermee verkregen y-waarde y^B_{n+1} noemend krijgen we nu

1.4.8 $$y^B_{n+1} = y_n + h(f_{n+1} - \frac{1}{2}\nabla_{n+1} - \frac{1}{12}\nabla^2_{n+1} - \frac{1}{24}\nabla^3_{n+1} - \frac{19}{720}\nabla^4_{n+1} - \ldots).$$

Dit is de formule van Bashforth. Voor gewone integratie (d.w.z. als $f = f(x)$ onafhankelijk van y is) is deze formule dezelfde als formule (3.2.0) op pag. 117. De coëfficiënten van Bashforth zijn veel kleiner, dan die van Adams, zodat men met Bashfort vaak minder differenties hoeft mee te nemen. Om deze reden gebruikt men graag (1.4.8) in combinatie met (1.4.7). De open formule (1.4.7) heet dan "predictor": hiermee wordt een schatting y^A_{n+1} "voorspeld", vervolgens wordt $f^A_{n+1} = f(x_{n+1}, y^A_{n+1})$ berekend en met deze schatting van f_{n+1} het differentieschema uitgebouwd. Daarna wordt met de gesloten formule (1.4.8), de "corrector", de "gecorrigeerde" waarde y^B_{n+1} berekend. Mocht deze waarde te veel van de voorspelde waarde y^A_{n+1} verschillen, dan berekent men f_{n+1} opnieuw, nu met y^B_{n+1} en herhaalt de berekening, totdat y_{n+1} niet meer verandert.

Andere formules, zowel van open als van gesloten type, worden verkregen, door te integreren vanaf een vroeger punt x_{n-k}, dus

$$y_{n+1} = y_{n-k} + \int_{x_{n-k}}^{x_{n+1}} f(t,y(t))dt$$

$$= y_{n-k} + h\int_{-k}^{1} f(x_n + ph)dp \qquad \text{(open type)}$$

$$= y_{n-k} + h\int_{-k-1}^{0} f(x_{n+1} + ph)dp \qquad \text{(gesloten type).}$$

Zo krijgen we voor k = 1 resp. k = 3 de volgende open formules

1.4.9 $$y_{n+1} = y_{n-1} + h(2f_n + \frac{1}{3}\nabla^2_n + \frac{1}{3}\nabla^3_n + \frac{29}{90}\nabla^4_n + \ldots)$$

1.4.10 $$y_{n+1} = y_{n-3} + h(4f_n - 4\nabla_n + \frac{8}{3}\nabla^2_n + \frac{14}{45}\nabla^4_n + \ldots).$$

In de eerste formule is de coëfficiënt van ∇_n nul. Het is dus gunstig de formule daar af te breken, waarmee we formules (1.1.7) weer gekregen hebben. In formule (1.4.10) is de coëfficiënt van ∇_n^3 nul. Breken we de formule daar af en drukken we de differenties uit in functie-waarden, dan ontstaat de formule

$$1.4.11 \quad y_{n+1} = y_{n-3} + \frac{4h}{3}(2f_n - f_{n-1} + 2f_{n-2}) + O(h^5).$$

Dit is de formule van Milne. Voor gewone integratie ($f = f(x)$) is deze formule dezelfde als de open quadratuur-formule (2.5.3) op pag. 112. De met $k = 1$ ontstaande gesloten formule luidt

$$1.4.12 \quad y_{n+1} = y_{n-1} + h(2f_{n+1} - 2\nabla_{n+1} + \frac{1}{3}\nabla^2_{n+1} - \frac{1}{90}\nabla^4_{n+1} + \ldots).$$

Ook hier ontbreekt de ∇_n^3-term. Breken we de formule daar af en werken we weer de differenties om, dan krijgen we de formule van Simpson (vgl. 2.1.3 pag. 102):

$$1.4.13 \quad y_{n+1} = y_{n-1} + \frac{h}{3}(f_{n+1} + 4f_n + f_{n-1}) + O(h^5).$$

Milne stelde voor de formule van Milne te gebruiken als predictor en die van Simpson als corrector.

1.5 Stabiliteit

Sommige formules zijn instabiel, d.w.z. dat de door het stap-voor-stap integreren veroorzaakte fout-opbouw zo groot is, dat de berekende oplossing meer en meer van de exacte oplossing gaat afwijken. We zullen dit nagaan voor twee eenvoudige formules.

De eerste orde formule (1.1.4)

We gaan het eerste-orde-beginwaarde-probleem (1.1.0) oplossen met behulp van (1.1.4), stap-voor-stap integrerend met constante stapgrootte h. Zij $y = y(x)$ de exacte oplossing en $y_n = y(x_n)$, waarbij $x_n = x_0 + nh$. Zij verder z_n de berekende benadering van y_n. Dan hebben we enerzijds

$$y_{n+1} = y_n + hf(x_n,y_n) + T_n, \tag{1.5.1}$$

waarbij T_n de afbreek-fout ("truncation error") is, dat is de fout, die ontstaat door de hogere termen van de Taylor-reeks te verwaarlozen. Anderzijds geldt voor de berekende waarden

$$z_{n+1} = z_n + hf(x_n,z_n) + R_n, \tag{1.5.2}$$

waarbij R_n de fout is, veroorzaakt door het niet-exact rekenen; als h klein is en f voldoende nauwkeurig berekend wordt, is R_n practisch gelijk aan de afrondingsfout ("rounding error"), ontstaan door het afronden van het resultaat z_{n+1}.
Door aftrekking krijgen we voor de fout $\varepsilon_n = y_n - z_n$ de volgende betrekking:

$$\begin{aligned}\varepsilon_{n+1} &= \varepsilon_n + h\left[f(x_n,y_n) - f(x_n,z_n)\right] + T_n - R_n \\ &= \varepsilon_n + h\varepsilon_n f_y(x_n,\eta_n) + E_n,\end{aligned} \tag{1.5.3}$$

waarbij $E_n = T_n - R_n$ en η_n ligt tussen y_n en z_n.
De laatste gelijkheid geldt volgens de stelling van Rolle, mits de afgeleide $f_y(x,y)$ bestaat.

We beperken ons nu tot het geval $f_y = A$ = constant, m.a.w. het gegeven beginwaarde-probleem luidt

1.5.4 $$y' = Ay \quad , \quad y(x_0) = y_0.$$

De oplossing hiervan is blijkbaar

1.5.5 $$y = y_0 \exp(A(x - x_0)).$$

De betrekking (1.5.3) gaat nu over in

1.5.6 $$\varepsilon_{n+1} = \varepsilon_n + Ah\varepsilon_n + E_n.$$

Beschouw eerst het homogene stuk $\varepsilon_{n+1} = (1 + Ah)\varepsilon_n$. De oplossing hiervan is blijkbaar $\varepsilon_n = \varepsilon_0(1 + Ah)^n$.
Beschouw nu (1.5.6) met $E_n = E$ = constant en probeer een constante C te vinden, zodat $\varepsilon_n = \alpha(1 + Ah)^n + C$ voldoet. Dan moet gelden

$$\alpha(1 + Ah)^{n+1} + C = \alpha(1 + Ah)^{n+1} + C(1 + Ah) + E,$$

dus $C = -E/(Ah)$.
Voor n = 0 vinden we meteen $\alpha = \varepsilon_0 + E/(Ah)$, zodat de oplossing van (1.5.6) met $E_n = E$ luidt

1.5.7 $$\varepsilon_n = (\varepsilon_0 + \frac{E}{Ah})(1 + Ah)^n - \frac{E}{Ah}.$$

Nemen we nu aan $|E_n| \leq E$ = constant, een situatie die zich voordoet bij vaste-komma-rekenen, mits E_n hoofdzakelijk uit de afrondingsfout bestaat. Men kan dan (door volledige inductie) bewijzen, dat voor $Ah > -1$ de fout ε_n voldoet aan

1.5.8 $$|\varepsilon_n| \leq (|\varepsilon_0| + \frac{E}{Ah})(1 + Ah)^n - \frac{E}{Ah}.$$

We onderscheiden nu 2 gevallen

1) $A > 0$.

Dan neemt de fout blijkbaar exponentiëel toe, de oplossing echter ook. Voor $h > 0$ geldt $(1 + Ah)^n < \exp(Ahn) = \exp(A(x - x_0))$, zodat we voor de relatieve fout krijgen

$$\frac{|\varepsilon_n|}{y_n} \leq \frac{|\varepsilon_0| + E/(Ah)}{y_0} \frac{(1 + Ah)^n}{\exp(A(x - x_0)} \leq \frac{|\varepsilon_0| + E/(Ah)}{y_0} .$$

M.a.w. de relatieve fout groeit niet aan.

2) $A < 0$.

Nu moet h voldoen aan $0 < h < \frac{1}{|A|}$. Dan is $0 < 1+Ah < 1$, dus het exponentiële stuk sterft uit en de fout voldoet aan $|\varepsilon_n| \leq \frac{E}{|Ah|}$. M.a.w. de absolute fout groeit niet aan (de relatieve fout wel, omdat de oplossing nu afnemende is).
In beide gevallen is het resultaat zo goed, als we mogen verwachten en de conclusie is, dat formule (1.1.4) stabiel is.

De tweede-orde formule (1.1.7)

Lossen we nu het probleem (1.1.0) op met behulp van formule (1.1.7), wederom stap-voor-stap integrerend met constante h, dan hebben we met dezelfde notatie:

$$y_{n+1} = y_{n-1} + 2hf(x_n, y_n) + T_n .$$

$$z_{n+1} = z_{n-1} + 2hf(x_n, z_n) + R_n .$$

Aftrekking levert nu

1.5.9 $$\varepsilon_{n+1} = \varepsilon_{n-1} + 2h\varepsilon_n f_y(x_n, \eta_n) + E_n .$$

We beperken ons weer tot $f_y = A$ = constant, dus

1.5.10 $$\varepsilon_{n+1} = \varepsilon_{n-1} + 2Ah\varepsilon_n + E_n .$$

We beschouwen eerst het homogene stuk $\varepsilon_{n+1} = \varepsilon_{n-1} + 2Ah\varepsilon_n$ en proberen als oplossing $\varepsilon_n = r^n$. Dan moet r voldoen aan $r^2 - 2Ahr - 1 = 0$, m.a.w.

1.5.11 $$r_{12} = Ah \pm \sqrt{A^2h^2 + 1} .$$

De algemene oplossing van het homogene stuk is dan

1.5.12 $$\varepsilon_n = \alpha r_1^n + \beta r_2^n .$$

Beschouw nu (1.5.10) met $E_n = E$ = constant en probeer een constante C te vinden zodat $\varepsilon_n = \alpha r_1^n + \beta r_2^n + C$ een oplossing is. Dan geldt blijkbaar $C = -E/(2Ah)$ en de oplossing luidt dus

1.5.13 $$\varepsilon_n = \alpha r_1^n + \beta r_2^n - \frac{E}{2Ah}$$

Nu is het triest: $r_1 r_2 = -1$, dus ofwel $|r_1|$ ofwel $|r_2|$ is groter dan 1 (of beide zijn 1, wat alleen optreedt in het geval A = 0). Dus de absolute fout groeit altijd exponentieel. Als A > 0, blijft de relatieve fout begrensd, maar als A < 0, is de oplossing afnemend en groeit dus zowel de absolute als de relatieve fout. De conclusie is, dat formule (1.1.7) instabiel is. Omdat de instabiliteit alleen optreedt voor A < 0, heet zij "conditioneel instabiel".

Welke formules zijn stabiel?

In het algemeen kan men zeggen, dat formules van het type $y_{n+1} = y_n + \Delta$ stabiel zijn. Zo zijn de Taylor-formules en de Runge Kutta-formules stabiel en eveneens de formules van Adams (1.4.7) en van Bashforth (1.4.8). Daarentegen zijn conditioneel instabiel de formules van het type $y_{n+1} = y_{n-k} + \Delta$ voor k > 0, dus in het bijzonder (1.4.9 t/m 13). Natuurlijk is er geen bezwaar tegen een instabiele formule als predictor met een stabiele formule als corrector te gebruiken.

Stabiliteit van beginwaarde-problemen

Ook het gegeven beginwaarde-probleem kan instabiel zijn. Bijvoorbeeld het beginwaarde-probleem

$$y'' = y,\ y(0) = 1,\ y'(0) = -1$$

heeft als oplossing: $y = e^{-x}$. Een kleine storing 2ε in de beginwaarde voor y heeft tot gevolg, dat de oplossing wordt: $y = \varepsilon e^x + (1 + \varepsilon)e^{-x}$. Voor toenemende x gaat e^x overheersen en is de afwijking dus veel te groot. Dit beginwaarde-probleem is instabiel. Geen enkele integratieformule startend in 0 kan de oplossing voor grote x in redelijke precisie leveren (tenzij met exacte arithmetiek)! In de praktijk komt zoiets nogal eens voor. Men kan de instabiliteit vaak vermijden, door van de

andere kant af te beginnen te integreren. Helaas weet men echter aan de andere kant vaak geen beginvoorwaarden.

Ook een eerste-orde beginwaarde-probleem kan instabiel zijn. Bijvoorbeeld $y' = y - x$, $y(0) = 1$ heeft als oplossing $y = x + 1$, maar een fout ε in de beginwaarde levert als fout in de oplossing εe^x, welke voor toenemende x spoedig gaat overheersen.

Opgaven

125) Integreer de differentiaal-vergelijking $y' = y/x + x \sin x$ met de formule van Adams in 5 decimalen voor $x = 1(0.025)1.25$ met de beginvoorwaarde $y(1) = 0$.

126) Bereken met de predictor-corrector methode Adams-Bashforth de oplossing van $y' = y/x - y^2$ met beginpunt $x = 1$, $y = 1$. Reken in 6 decimalen, kies $h = 0.1$ en integreer tot en met $x = 2.0$.

1.6 Gebruik van de meerstap-formules

Start

Om een meerstap-formule te kunnen toepassen, moeten we voor een voldoende aantal equidistante argumenten x_i de bijbehorende y_i en f_i berekend hebben. Het interval h moet hierbij zo klein zijn, dat de hogere differenties snel genoeg afnemen.
De start kan worden verkregen door middel van een eenstap-formule, dus Taylor of Runge-Kutta. Bij gebruik van de Taylor-reeks kan men bovendien een geschikte h vinden, wegens

1.6.1 $$\frac{\Delta^k f_i}{h^k} = k!f[x_i, \ldots, x_{i+k}] = f^{(k)}(\xi),\ \xi \text{ tussen } x_i \text{ en } x_{i+k}.$$

(zie pag. 56 en (15.5) pag. 70).
Dus voor kleine h geldt: $\Delta^k f_0 \approx h^k f^{(k)}(0)$.

1.6.2 Voorbeeld $\qquad y' = x^2 - y,\ y(0) = 1.$

Dus $y_0' = -1$

$y'' = 2x - y',\ y_0'' = +1,$

$y''' = 2 - y'',\ y_0''' = +1,$

$y^{(4)} = -y^{(3)},\ y_0^{(4)} = -1.$

De hogere afgeleiden zijn blijkbaar alternerend $\pm$ 1.
Voor h = 0.1 geldt $h^6 y_0^{(6)} = h^6 f_0^{(5)} = -10^{-6}$, zodat we bij verwaarlozing van $\nabla^5 f$ een precisie van 5 à 6 decimalen mogen verwachten. We berekenen nu uit de Taylor-reeks y(x) voor x = -0.3(0.1)0.3 en gaan daarna met Adams verder integreren.

x	y	hf			
-.3	1.340141	-.125014			
			7154		
-.2	1.218597	-.117860		1223	
			8377		-117
-.1	1.104829	-.109483		1106	12
			9483	-105	
.0	1.000000	-.100000		1001	9
			10484	- 96	
.1	.905163	-.089516		905	11
			11389	- 85	
.2	.821269	-.078127		820	6
			12209	- 79	
.3	.749182	-.065918		741	9
			12950	- 70	
.4	.689682	-.052968		671	6
			13621	- 64	
.5	.643470	-.039347		607	6
			14228	- 58	
.6	.611190	-.025119		549	7
			14777	- 51	
.7	.593416	-.010342		498	2
			15275	- 49	
.8	.590672	+.004933		499	7
			15724	- 42	
.9	.603433	+.020657		407	
			16131		
1.0	.632121	+.036788			

<u>Verandering van de staplengte</u>

Als men zo door-integreert kunnen twee dingen gebeuren:

1) De differenties worden steeds kleiner in absolute waarde. Worden ze al te klein, dan moet de stap vergroot worden zowel om sneller op te schieten als om onnodige opbouw van afrondingsfouten te voorkomen.

Het ligt voor de hand de stap te verdubbelen. Het bijbehorende schema kan dan zonder meer worden opgesteld. Natuurlijk moet men voldoende punten ter beschikking hebben en bedenken dat bij de dubbele stap de k-de differentie van hf ongeveer 2^{k+1} maal groter wordt.

2) De differenties worden steeds groter in absolute waarde. Dan moet men op een gegeven moment de stap verkleinen om voldoende precisie te houden. Meestal zal men de stap halveren. De tussenliggende waarden van f kan men berekenen door interpolatie (Bessel of achterwaartse Newton), waarna men zonder moeite het schema voor de nieuwe staplengte gereed maakt.

1.7 Meerstap-formules met centrale differenties. Iteratieve start-procedure

Zoals we Adams' formule hebben afgeleid uit de achterwaartse Newton-formule leiden we nu een centrale formule af uit de formule van Bessel (19.3 pag. 81), die we als volgt mogen schrijven (wegens $\delta_0^{2k}+\delta_1^{2k}=2\mu\delta_{\frac{1}{2}}^{2k}$):

$$1.7.0 \quad f^{**}(x_0 + ph) = \mu f_{\frac{1}{2}} + (p - \tfrac{1}{2})\delta_{\frac{1}{2}} + 2B_2\mu\delta_{\frac{1}{2}}^2 + B_3\delta_{\frac{1}{2}}^3 + 2B_4\mu\delta_{\frac{1}{2}}^4 + \ldots$$

We vervangen nu in (1.4.3) de functie f door $f^{**}(x_n + ph)$ en krijgen dan

$$y_{n+1} = y_n + \int_{x_n}^{x_{n+1}} f(t,y(t))dt \approx y_n + h\int_0^1 f^{**}(x_n + ph)dp.$$

Werken we dit uit, dan blijken de coëfficiënten van oneven orde nul te zijn en het resultaat luidt:

$$1.7.1 \qquad y_{n+1} = y_n + h(\mu f_{n+\frac{1}{2}} - \frac{1}{12}\mu\delta_{n+\frac{1}{2}}^2 + \frac{11}{720}\mu\delta_{n+\frac{1}{2}}^4 + \ldots)$$

Deze formule is dezelfde als (3.3.0) op pag. 118.

Op dezelfde wijze vinden we door de formule van Stirling (19.2 pag. 80) te integreren:

$$y_{n+1} = y_{n-1} + h\int_{-1}^{+1} (f_n + p\mu\delta_n + S_2(p)\delta_n^2 + S_3(p)\mu\delta_n^3 + \ldots)dp.$$

De coëfficiënten van oneven orde zijn weer 0 en we krijgen

1.7.2 $$y_{n+1} = y_{n-1} + h(2f_n + \frac{1}{3}\delta_n^2 - \frac{1}{90}\delta_n^4 + \ldots).$$

Deze formule is weer conditioneel instabiel, terwijl (1.7.1) stabiel is. Men kan (1.7.1) als corrector gebruiken met (1.7.2) als predictor. Deze centrale formules hebben het voordeel, dat de coëfficiënten veel sneller afnemen, en het nadeel, dat men bij hogere orde formules nogal wat f-waarden en differenties moet voorspellen. Hierdoor zijn de centrale formules ongeschikt voor automatische berekening.

Iteratieve start-procedure

De centrale formules kunnen goed worden gebruikt voor een iteratieve start-berekening. Men begint een tabel van y te maken m.b.v. de eerste orde-formule (1.1.4). Daarna gaat men iteratief (1.7.1) toepassen, totdat de waarden niet meer veranderen. Doen we dit voor het beginwaardeprobleem (1.6.2), dan krijgen we

x	y	hf		
-.3	1.33	-.124		
			7	
-.2	1.21	-.117		1
			8	
-.1	1.10	-.109		1
			9	
.0	1.00	-.100		2
			11	
.1	.90	-.089		1
			12	
.2	.81	-.077		1
			13	
.3	.73	-.064		

Dit was m.b.v. (1.1.4); nu passen we (1.7.1) toe.

x	y	hf						
-.3	1.3380	-.12480			(130)			
			705			(-5)		
-.2	1.2175	-.11775			125			(-5)
			830			-10		
-.1	1.1045	-.10945			115			-5
			945			-15		
.0	1.0000	-.10000			100			0
			1045			-15		
.1	.9055	-.08955			85			+5
			1130			-10		
.2	.8225	-.07825			75			(+5)
			1205			(-5)		
.3	.7520	-.06620			(70)			

De getallen tussen haakjes zijn geschat. Nogmaals (1.7.1) toepassen le-
vert

x	y	hf						
-.3	1.339896	-.124990			(1376)			
			7138			(-143)		
-.2	1.218515	-.117852			1233			(20)
			8371			-123		(-3)
-.1	1.104815	-.109481			1110			17
			9481			-106		-3
.0	1.000000	-.100000			1004			14
			10485			- 92		-3
.1	.905148	-.089515			912			11
			11397			- 81		(-3)
.2	.821181	-.078118			831			(8)
			12228			(-73)		
.3	.748896	-.065890			(758)			

Dit stemt in beide 4 decimalen met de Taylor-start overeen. Na nog twee iteraties hebben we 6 decimalen goed en is de start gereed.

Opgaven

127) Bepaal de exacte oplossing van het beginwaarde-probleem (1.6.2).
Ga (voor enige waarden van x) na hoe goed de met behulp van Adams verkregen oplossing hiermee overeenstemt.

128) De oplossing y van (1.6.2) blijkt een minimum te hebben voor x in de buurt van 0.8. Bereken dit minimum en bijbehorend argument
a) door inverse interpolatie in het boven vermelde differentie-schema
b) met behulp van Newton-iteratie toegepast op de exacte oplossing.

129) Bereken de oplossing van de differentiaalvergelijking $y' = x^2 + y^2 - 2$ met beginwaarde $y(0) = 1$ voor $x = 0.1(0.1)0.6$ in 6 decimalen.
a) Gebruik de Taylor-reeks voor de start en daarna Adams-Bashforth.
b) Bereken de start iteratief en ga daarna integreren met een centrale integratie-formule.

1.8 Meerstap-formules voor tweede-orde differentiaal-vergelijkingen

We hebben gezien, hoe een differentiaal-vergelijking van de tweede orde $y'' = f(x,y,y')$ kan worden geschreven als een stelsel eerste orde vergelijkingen $y' = z$, $z' = f(x,y,z)$. Zijn beginwaarden van y en y' gegeven, dan kan dit stelsel met elk van de boven beschreven methodes worden aangepakt, waarbij dus zowel y als de afgeleide y' berekend worden. Een belangrijk bijzonder geval is een tweede orde vergelijking, waarvan het rechterlid niet van y' afhangt. niet alleen komt dit geval vaak voor in physische problemen, maar ook kunnen vele tweede orde vergelijkingen in deze vorm getransformeerd worden.

Voorbeeld. De algemene lineaire tweede orde vergelijking luidt:

1.8.0 $$y'' + p(x)y' + q(x)y = r(x).$$

Stel nu $y = vz$, dan hebben we $y' = vz' + v'z$ en $y'' = vz'' + 2v'z' + v''z$. Substitutie in (1.8.0) levert

$$vz'' + (2v' + pv)z' + (v'' + pv' + qv)z = r.$$

Om de afhankelijkheid van z' weg te werken stellen we nu $2v' + pv = 0$, m.a.w. $\frac{v'}{v} = -\frac{1}{2}p$, dus

1.8.1 $$v = c \exp\left(-\frac{1}{2}\int p(x)dx\right).$$

Wegens $\frac{v''}{v} = \frac{v'^2}{v^2} - \frac{1}{2}p' = \frac{1}{4}p^2 - \frac{1}{2}p'$ krijgt de differentiaal-vergelijking voor z dan de gedaante

1.8.2 $$z'' + (q - \tfrac{1}{4}p^2 - \tfrac{1}{2}p')z = r/v.$$

Bijvoorbeeld de differentiaal-vergelijking van Bessel luidt

$$x^2y'' + xy' + (x^2 - n^2)y = 0.$$

We vinden dan $v = 1/\sqrt{x}$ en $y = z/\sqrt{x}$, waarbij z voldoet aan de differentiaal-vergelijking

$$z'' + (1 - (n^2 - \tfrac{1}{4})\frac{1}{x^2})z = 0.$$

We beperken ons nu tot het bijzondere tweede orde beginwaarde-probleem

1.8.3 $$y'' = f(x,y),\ y(x_0) = y_0,\ y'(x_0) = z_0.$$

Voor dit geval bestaan speciale meerstap-formules, waarbij y' niet hoeft te worden berekend. Om deze af te leiden gaan we (1.8.3) tweemaal integreren:

$$y'(x_n + th) = y'(x_n) + \int_{x_n}^{x_n+th} f(x,y(x))dx.$$

Schrijven we $y'_{n+p} = y'(x_n + ph)$ en $f_{n+p} = f(x_n + ph, y(x_n + ph))$, dan wordt dit:

$$y'_{n+t} = y'_n + h \int_0^t f_{n+p}\, dp.$$

Nogmaals integreren levert:

1.8.4 $$y_{n+s} = y_n + sh\, y'_n + h^2 \int_0^s \int_0^t f_{n+p}\, dp\, dt.$$

Dit zou met $s = 1$ kunnen worden gebruikt om y_{n+1} te berekenen, maar dan zou in elke stap ook de afgeleide moeten worden berekend. We kunnen echter y'_n wegwerken door (1.8.4) met $s = 1$ en $s = -1$ bij elkaar op te tellen, wat als resultaat heeft:

1.8.5 $$\delta^2 y_n = y_{n+1} - 2y_n + y_{n-1} = h^2\left(\int_0^1\int_0^t f_{n+p}dpdt + \int_0^{-1}\int_0^t f_{n+p}dpdt\right).$$

Benaderen we f met de achterwaartse formule van Newton (1.4.6) dan vinden we

1.8.6 $$\nabla^2 y_{n+1} = y_{n+1} - 2y_n + y_{n-1} = h^2\left(f_n + \frac{1}{12}\nabla_n^2 + \frac{1}{12}\nabla_n^3 + \frac{19}{240}\nabla_n^4 + \ldots\right).$$

Benaderen we f echter met de formule van Stirling (19.2 pag. 80), dan krijgen we de centrale formule:

1.8.7 $$\delta^2 y_n = y_{n+1} - 2y_n + y_{n-1} = h^2\left(f_n + \frac{1}{12}\delta_n^2 - \frac{1}{240}\delta_n^4 + \frac{31}{60480}\delta_n^6 - \ldots\right).$$

Formule (1.8.6) is een open formule en kunnen we, na het optellen van een start, zonder meer gebruiken.
Bij het gebruik van (1.8.7) daarentegen moeten we differenties schatten of (1.8.6) als predictor gebruiken.
Vermenigvuldigen we (1.8.7) met δ^{-2} en vervangen we n door n+1 dan ontstaat de formule

1.8.8 $$y_{n+1} = h^2(\delta^{-2}f_{n+1} + \frac{1}{12} f_{n+1} - \frac{1}{240} \delta^2 f_{n+1} + \ldots)$$

(zie Interpolation and allied Tables, p. 74).

Deze formule is numeriek geenszins equivalent met de vorige (hier wordt nl. de tweede som-functie $\delta^{-2}f$ gebruikt) en heeft enig voordeel bij het handrekenen.

De start

De start kan wederom worden berekend met behulp van de Taylor-reeks of iteratief. De start-berekening moet, evenals in het algemene geval, nauwkeurig geschieden, nu te meer omdat de afgeleide y' met zijn beginwaarde alleen in de start gebruikt worden.

Voorbeeld $y'' + xy = 0$, $y(0) = 0$; $y'(0) = 1$.
Uit de Taylorreeks van y vinden we voor $x = -0.3(0.1)0.3$

x	y	h^2f			
-.3	-.300675(5)	-9020			
			+5017		
-.2	-.2001333	-4003		-2014	
			+3003		11
-.1	-.1000083	-1000		-2003	
			+1000		3
.0	0	0		-2000	
			-1000		3
.1	.0999917	-1000		-1997	
			-2997		11
.2	.1998667	-3997		-1986	
			-4983		
.3	.299325(5)	-8980			

We gaan nu de startwaarden voor $\delta^{-2}f$ en $\delta^{-1}f$ bepalen.
Met behulp van (1.8.8) vinden we

$$h^2\delta^{-2}f_0 = y_0 - \frac{h^2}{12} f_0 + \frac{h^2}{240} \delta^2 f_0 + \ldots \approx -.8_{10}{}^{-6},$$

$$h^2\delta^{-2}f_1 = y_1 - \frac{h^2}{12} f_1 + \frac{h^2}{240} \delta^2 f_1 + \ldots \approx .0999992.$$

Dus $h^2\delta^{-1}f_{\frac{1}{2}} = h^2\delta^{-2}f_1 - h^2\delta^{-2}f_0 \approx .1000000.$

Nu kunnen we het startschema opstellen. We controleren nog even y(.2) en y(.3) (die in 7 decimalen blijken te kloppen) en gaan dan in 6 decimalen verder integreren met (1.8.8).

x	y	$h^2\delta^{-2}f$	$h^2\delta^{-1}f$	h^2f		
0	.000000	-1		0	-200	
			100000	-100	0	
.1	.099992	99999		-100	-200	2
			99900	-300	2	
.2	.199867	199899		-400	-198	1
			99500	-498	3	
.3	.299325	299399		-898	-195	2
			98602	-693	5	
.4	.397869	398001		-1591	-190	6
			97011	-883	11	
.5	.494807	495012		-2474	-179	5
			94537	-1062	16	
.6	.589255	589549		-3536	-163	4
			91001	-1225	19	
.7	.680154	680550		-4761	-144	8
			86240	-1369	27	
.8	.766280	766790		-6130	-117	6
			80110	-1486	33	
.9	.846266	846900		-7616	- 84	(9)
			72494	-1570	(42)	
1.0	.918629	919394		-9186	(-42)	

De getallen tussen haakjes zijn geschatte waarden.

De oplossing hangt samen met de Bessel-functie $J_{1/3}$ en ook met de Airy functies (zie Handbook of Mathematical functions, NBS-AMS 55, hoofdstuk 10). De hier berekende waarden blijken in alle decimalen goed te zijn, behalve voor x = .4, waarvoor de correct afgeronde waarde luidt .397870.

Dubbele integraal

Een nog meer bijzonder geval is de differentiaal-vergelijking $y'' = f(x)$, waarbij de functie f noch van y noch van y' afhangt. Zijn de beginwaarden $y(0) = y'(0) = 0$, dan is de oplossing blijkbaar

$$y(x) = \int_0^x \int_0^t f(u)du\ dt.$$

Deze oplossing kan met formule (1.8.8) berekend worden, waarbij nu niets voorspeld hoeft te worden, omdat f voor elke waarde van x bekend is.

Opgaven

130) Bereken de oplossing y van het beginwaarde-probleem (vgl. opgave 122, pag. 237)

$$y'' + x^2y = 0,\ y(0) = 0,\ y'(0) = 1$$

voor x = .1(.1)1.0 met behulp van (1.8.8). Voer de berekening uit in 6 decimalen.

131) Bereken in 6 decimalen de oplossing y van het beginwaarde-probleem

$$y'' = xy + x^2,\ y(0) = y'(0) = 0$$

voor x = .1(.1)1.0 met behulp van (1.8.8).

132) Bereken in 6 decimalen de dubbele integraal

$$\int_0^1 \int_0^x \cos\left(\frac{\pi}{2}t^2\right)dt\ dx.$$

2. Numerieke oplossing van randwaarde-problemen

We hebben gezien, dat voor het eenduidig vastleggen van een oplossing van een differentiaal-vergelijking extra condities nodig zijn (zie Inleiding). Het aantal extra condities is in het algemeen gelijk aan de orde van de differentiaal-vergelijking. Als deze extra condities de waarde van de oplossing en afgeleiden hiervan in slechts één punt voorschrijven, spreken we van een beginwaarde-probleem. Komen daarentegen in de extra condities de waarden voor van de oplossing en eventueel ook afgeleiden ervan in verschillende punten, dan is het probleem een randwaarde-probleem.
Wij zullen ons beperken tot randwaarde-problemen van de tweede orde. Deze hebben twee extra condities, waarin meestal de waarde van de oplossing in twee punten wordt voorgeschreven, in formule

2.0.0 $$y'' = f(x,y,y'),\ y(x_0) = y_0,\ y(x_e) = y_e.$$

De rand-condities leggen de oplossing helaas niet altijd eenduidig vast. Bijvoorbeeld het randwaarde-probleem

2.0.1 $$y'' + y = 0,\ y(0) = 0,\ y(\pi) = 1$$

heeft geen oplossing. Vervangen we evenwel de laatste randvoorwaarde door $y(\pi) = 0$, dan heeft het probleem oneindig veel oplossingen $c \sin(x)$, waarbij c een willekeurige constante is.

2.1 Herleiding tot beginwaarde-probleem

Voor het oplossen van het randwaarde-probleem (2.0.0) beschouwen we eerste het beginwaarde-probleem

2.1.0 $$y'' = f(x,y,y'),\ y(x_0) = y_0,\ y'(x_0) = \eta.$$

Voor elke gekozen η kunnen we de oplossing $y_\eta(x)$ numeriek bepalen met een van de bovengenoemde methodes.
We zijn hierbij vooral geïnteresseerd in de "eindwaarde" $y_\eta(x_e)$ en zoeken een waarde η, waarvoor $y_\eta(x_e) = y_e$. M.a.w. we moeten een nulpunt van de functie $y_\eta(x_e) - y_e$ bepalen. Dit kan het best geschieden met behulp van de Regula falsi.

Heeft de functie $y_\eta(x_e) - y_e$ in η_1 en η_2 verschillend teken, dan hebben we een veilig interval en kunnen we de veilige methode toepassen (zie pag. 156 en AP 230, pag. 158).
Deze methode kan tamelijk tijdrovend zijn, doordat verscheidene malen een beginwaarde-probleem moet worden opgelost.
Is evenwel de gegeven differentiaal-vergelijking lineair, dan is $y_\eta(x_e)$ een lineaire functie van η zodat slechts voor twee waarden van η hoeft te worden geïntegreerd en de oplossing door eenmaal lineair interpoleren verkregen wordt. Met een kleine wijziging gaat dit als volgt. Een lineaire differentiaal-vergelijking heeft de algemene gedaante

2.1.1 $$y'' + p(x)y' + q(x)y = r(x).$$

Laten de randwaarden zijn

2.1.2 $$y(x_0) = y_0,\ y(x_e) = y_e.$$

Laat $u(x)$ een oplossing zijn van (2.1.1) met beginwaarden $u(x_0) = y_0$, $u'(x_0) = \eta_1$, waarbij η_1 willekeurig gekozen mag zijn; laat $v(x)$ een oplossing zijn van het homogene stuk $y'' + p(x)y' + q(x)y = 0$ met beginwaarden $v(x_0) = 0$, $v'(x_0) = \alpha$, waarbij α een willekeurige waarde $\neq 0$ mag hebben. Deze functie v is in feite het verschil van twee oplossingen van (2.1.1) met dezelfde waarde in x_0. Nu stellen we

2.1.3 $$y(x) = u(x) + cv(x)$$

en trachten c zo te bepalen, dat aan het randwaarde-probleem voldaan is. Blijkbaar voldoet $y(x)$ voor alle c aan (2.1.1) en $y(x_0) = y_0$. Voor de tweede randvoorwaarde hebben we $y(x_e) = u(x_e) + cv(x_e)$, wat gelijk moet zijn aan y_e. Dus

2.1.4 $$c = (y_e - u(x_e))/v(x_e).$$

Hiermee is het randwaarde-probleem opgelost, mits $v(x_e) \neq 0$. Anders krijgen we geen enkele of oneindig veel oplossingen (zie voorbeeld 2.0.1).

Voorbeeld

2.1.5 $y'' + y = x,\ y(0) = 0,\ y(\frac{\pi}{2}) = 1.$

Een particuliere oplossing van de differentiaal-vergelijking met $y(0) = 0$ is $u(x) = x$ en de oplossing van het homogene probleem met $y(0) = 0$ en $y'(0) = 1$ is $v(x) = \sin(x)$.
De eindvoorwaarde levert $c = 1 - \frac{\pi}{2}$ en de oplossing van dit randwaarde-probleem luidt dus $y(x) = x + (1 - \frac{\pi}{2})\sin(x)$.

2.2 Herleiding tot stelsel lineaire algebraïsche vergelijkingen

We beschouwen weer het lineaire tweede-orde randwaardeprobleem

2.2.0 $y'' + p(x)y' + q(x)y = r(x),\ y(x_0) = y_0,\ y(x_e) = y_e.$

We verdelen het interval $[x_0, x_e]$ in n gelijke stukken ter lengte h, dus $h = (x_e - x_0)/n$. Volgens de gebruikelijke notatie $x_i = x_0 + ih$ hebben we dan $x_e = x_n = x_0 + nh$. Dus y_0 en $y_n = y_e$ zijn bekend; daartussen liggen de onbekenden $y_1, \ldots, y_{n-1}$, waarvoor we een stelsel vergelijkingen gaan opstellen. Hiertoe drukken we y' en y'' uit in centrale differenties (vgl. de formules op pag. 98):

2.2.1 $h\ y_i' = (\mu\delta - \frac{1}{6}\mu\delta^3 + \frac{1}{30}\mu\delta^5 - \ldots)y_i,$

2.2.2 $h^2 y_i'' = (\delta^2 - \frac{1}{12}\delta^4 + \frac{1}{90}\delta^6 - \ldots)y_i.$

We gaan dit in de differentiaal-vergelijking invullen, waarbij we schrijven $\mu\delta y_i = \frac{1}{2}(y_{i+1} - y_{i-1})$, $\delta^2 y_i = y_{i+1} - 2y_i + y_{i-1}$, $p_i = p(x_i)$, enz. Alles met h^2 vermenigvuldigend krijgen we dan voor $i = 1(1)n-1$:

2.2.3 $(1 - \frac{1}{2}hp_i)y_{i-1} - (2 - h^2 q_i)y_i + (1 + \frac{1}{2}hp_i)y_{i+1} + Cy_i = h^2 r_i,$

waarbij

$$Cy_i = -\frac{1}{12}\delta^4 y_i + \frac{1}{90}\delta^6 y_i - \ldots - \frac{h}{6}p_i\mu\delta^3 y_i + \frac{h}{30}p_i\mu\delta^5 y_i - \ldots .$$

Zien we voorlopig af van de hogere orde termen Cy_i, dan hebben we het volgende lineaire stelsel

$$2.2.4 \quad \begin{cases} -(2-h^2q_1)y_1 + (1+\frac{1}{2}hp_1)y_2 = h^2r_1-(1-\frac{1}{2}hp_1)y_0 \\ (1-\frac{1}{2}hp_i)y_{i-1} - (2-h^2q_i)y_i + (1+\frac{1}{2}hp_i)y_{i+1} = h^2r_i, \quad i=2(1)n-2 \\ (1-\frac{1}{2}hp_{n-1})y_{n-2} - (2-h^2q_{n-1})y_{n-1} = h^2r_{n-1}-(1+\frac{1}{2}hp_{n-1})y_e. \end{cases}$$

De matrix A van dit stelsel en het rechterlid b zijn

$$2.2.5 \quad A = \begin{pmatrix} -(2-h^2q_1) & 1+\frac{1}{2}hp_1 & & & 0 \\ 1-\frac{1}{2}hp_2 & -(2-h^2q_2) & 1+\frac{1}{2}hp_2 & & \\ & \ddots & \ddots & \ddots & \\ & & & & 1+\frac{1}{2}hp_{n-2} \\ 0 & & & 1-\frac{1}{2}hp_{n-1} & -(2-h^2q_{n-1}) \end{pmatrix},$$

$$b = \begin{pmatrix} h^2r_1 - (1-\frac{1}{2}hp_1)y_0 \\ h^2r_2 \\ \vdots \\ h^2r_{n-2} \\ h^2r_{n-1} - (1+\frac{1}{2}hp_{n-1})y_e \end{pmatrix}.$$

Matrix A is tridiagonaal, wat het belangrijke voordeel heeft, dat Gauss' eliminatie zowel als de methode van Crout veel sneller verlopen dan bij een volle matrix.

De oplossings-vector van dit stelsel noemen we $y^{[0]}$, m.a.w.

$$2.2.6 \quad Ay^{[0]} = b.$$

De elementen van $y^{[0]}$ zijn een eerste schatting voor de functie-waarden $y_1, \ldots, y_{n-1}$. We vinden vervolgens iteratief betere schattingen door oplossing van het stelsel

2.2.7 $$Ay^{[k+1]} = b - Cy^{[k]} \quad , k = 0, 1, 2, \ldots .$$

Om de vector $Cy^{[k]}$ te berekenen, hebben we nog enige functie-waarden nodig buiten het interval $[x_0, x_e]$. Deze vinden we m.b.v. 2.2.3, bv.

$$y_{-1}^{[k]} = (2-h^2q_0)y_0 - (1+\tfrac{1}{2}hp_0)y_1^{[k]} + h^2r_0 - Cy_0^{[k-1]}.$$

Vaak kunnen evenwel de hogere differenties, benodigd voor Cy_0 en Cy_e, geschat worden.

<u>Voorbeeld</u> (uit Fröberg, pag. 256).

$$y'' + \frac{y}{1+x^2} = 7x, \; y(0) = 0, \; y(1) = 2.$$

Kiezen we $n = 5$ en dus $h = 0.2$, dan luidt het lineaire stelsel (2.2.6):

$$\begin{aligned} -1.9615y_1 + y_2 &= .056 \\ y_1 - 1.9655y_2 + y_3 &= .112 \\ y_2 - 1.9706y_3 + y_4 &= .168 \\ y_3 - 1.9756y_4 &= -1.776 \end{aligned}$$

Als oplossing $y^{[0]}$ vinden we

$$y_1 = .208 \; , \; y_2 = .464 \; , \; y_3 = .816 \; , \; y_4 = 1.312,$$

wat goed overeenstemt met de exacte oplossing $y = x^3 + x$.

Voor een ander voorbeeld, zie Modern Computing methods, pag. 95.

<u>Niet-lineaire randwaarde-problemen</u>

Voor niet-lineaire randwaarde-problemen leidt bovenstaande methode tot een niet-lineair stelsel, dat meestal veel lastiger op te lossen is. Men kan hiertoe zijn toevlucht nemen tot Newton's iteratie voor stelsels vergelijkingen (vgl. pag. 169, waar dit proces is beschreven voor 2 vergelijkingen met 2 onbekenden). We gaan hierop niet verder in.

2.3 Eigenwaarde-problemen

Dit zijn homogeen-lineaire randwaarde-problemen met een parameter λ. Een typisch tweede-orde eigenwaarde-probleem ziet er als volgt uit.

2.3.0 $$a_0y'' + a_1y' + (a_2 - \lambda)y = 0,\ y(x_0) = y(x_e) = 0,$$

waarbij a_0, a_1, a_2 functies van x zijn. Waarden van λ, waarvoor het probleem een oplossing y heeft, die niet identiek nul is, heten eigenwaarden en de bijbehorende functie y heet eigenfunctie. De eigenfunctie is op een constante factor na bepaald.
Gaan we, evenals boven, y' en y'' uitdrukken in centrale differenties en verwaarlozen we differenties van de orde 3 en hoger, dan krijgen we een stelsel van de gedaante

2.3.1 $$(A - \lambda I)y = \vec{0},$$

waarbij A wederom een tridiagonale matrix is.

Voorbeeld

2.3.2 $$y'' + \lambda y = 0,\ y(0) = y(1) = 0.$$

De eigenwaarden zijn $\lambda = k^2\pi^2$, $k = 1, 2, \ldots$; de bijbehorende eigenfuncties zijn $y = c \sin(\sqrt{\lambda}\, x) = c \sin(k\pi x)$, waarbij c een willekeurige constante is.
Voor n = 5, dus h = 0.2, vinden we als matrix.

$$A = 25 \begin{pmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & -1 & 2 & -1 \\ & & -1 & 2 \end{pmatrix}$$

De eigenwaarden van A zijn $\lambda_i = 100 \sin^2(\frac{i\pi}{10})$, $i = 1, 2, 3, 4$; dus λ_1 is een aardige benadering van de kleinste eigenwaarde π^2 van (2.3.2).

Opgaven

133) Zij gegeven de matrix A = (a_{ij}) van de orde n, gedefiniëerd door

$a_{ii} = -2$ voor $i = 1(1)n$, $a_{i,i+1} = a_{i+1,i} = 1$

voor i = 1(1)n-1 en alle andere elementen van A zijn 0.
Bereken de inverse van A voor n = 2, 3, 4, 5.

134) Schrijf een ALGOL-procedure, die een stelsel lineaire algebraïsche vergelijkingen oplost met behulp van Gauss' eliminatie, als de matrix van het stelsel een tridiagonale matrix van de orde n is.

135) Bereken in 4 decimalen de oplossing van het randwaarde-probleem

$$y'' + x^2 y = 0 \ , \quad y(0) = 0, \ y(1) = 1$$

voor x = 0.25, 0.50, 0.75.

Literatuur

Aan de lijsten op pag. 21, 121 en 180 voegen we hier toe de volgende boeken op het gebied van numerieke wiskunde.

30) W.E. Milne, Numerical solution of differential equations (1953).

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

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

33) C.E. Fröberg, Introduction to Numerical Analysis (1965).

34) A. Ralston, A first course in Numerical Analysis (1965).

35) R.W. Southworth and S.L. de Leeuw, Digital Computation and Numerical methods (1965).

Practicum Proefwerk

19-10-1965

1) Gevraagd alle 3 wortels in 5 decimalen nauwkeurig van de volgende vergelijking

$$x^3 - 15\,x^2 + 120x - 100 = 0$$

2) Gegeven de volgende tabel van $f(x) = \dfrac{\operatorname{arccosh} x}{\sqrt{x^2 - 1}}$

x	f(x)
1.75	.80689197
1.76	.80489994
1.77	.80291954
1.78	.80095066
1.79	.79899318
1.80	.79704701
1.81	.79511203
1.82	.79318816
1.83	.79127527
1.84	.78937328

a) Gevraagd door interpolatie te berekenen f(1.78656)

b) Gevraagd die waarde van x te bepalen, waarvoor f(x) = 4/5

Practicum Proefwerk

26-10-1965

1) Gevraagd alle 3 nulpunten in 5 decimalen nauwkeurig van de volgende veelterm

$$x^3 - 8x^2 - 28x - 20.$$

2) Zij gegeven de volgende tabel van f(x) = tanh (x)

x	f(x)
1.80	.94680601
1.81	.94783185
1.82	.94883842
1.83	.94982608
1.84	.95079514
1.85	.95174596
1.86	.95267884
1.87	.95359412
1.88	.95449211

a) Bereken door interpolatie f(1.8383838).

b) Bepaal die waarde van x, waarvoor f(x) = .95000000.

Cursus WRA

T.J. Dekker

Proefwerk ALGOL 60 26-10-1965

Maak vraagstuk 1 en 2 en hetzij 3 hetzij 4

1) Hieronder volgen ALGOL statements, waarin syntactische fouten zijn geslopen.
Corrigeer deze fouten.

a) x := if if B then B A else B B then if C
then xa else xb else if D then xc else xd

b) if x = if B then 0 else -1 ∨ 0 < a < 1 then for
i := 1, i+1 while true do
P(a,b) else z := z + if B then 1 else 0

c) begin procedure A(b,c,n); value n, b; array c[1:n];
begin integer i; for i := 1 step 1 until n do
C[i] := b↑ - i end;
array y[1:8.9]; A(3.14,y,9) end

2) a) Schrijf een procedure voor de benadering volgens de trapeziumregel

van $\int_a^b f(x)dx$ met n deelintervallen.

De heading van de procedure moet luiden:
"real procedure trap(x,a,b,fx,n);
value a,b,n; real x,a,b,fx; integer n;"

b) Welke waarde krijgt z na uitvoering van
"z := trap(x,0,1,trap(y,0,x,x+y,2),2)",
waarbij x,y,z variabelen van type real zijn.

z.o.z.

3) Schrijf een functie-procedure, die een nulpunt van een functie f met afgeleide Df bepaalt volgens de iteratie-formule van Newton. Start de iteratie in het gegeven punt x0. De iteratie moet worden beëindigd als twee opeenvolgende iterates minder dan eps uiteenliggen of als het aantal gedane stappen 15 bedraagt. Lever de laatste iterate af als waarde van de functie en het laatste verschil als deltax. De heading moet luiden:
"real procedure Newton (x0,f,Df,eps,deltax);
value x0,eps; real x0,eps,deltax; real procedure f,Df;"

4) Schrijf een procedure, welke de elementen van een integer array A[1:n] ordent in volgorde van niet-afnemende grootte.

Cursus W.R.-A 18-1-1966

T.J. Dekker

Proefwerk

1) a) Hoe luidt de interpolatie-formule van Lagrange van de orde n? Vermeld hierbij ook expliciet de gedaante der Lagrange-coëfficiënten.

b) Bereken de Lagrange-coëfficiënten van de orde 4 als de basispunten zijn 2, 3, 5 en 7 voor het argument $x = 4$.

c) Hoe luidt de restterm, horende bij de n-de orde Lagrange-formule, uitgedrukt in de n-de afgeleide van de te interpoleren functie? In welk interval moet deze n-de afgeleide bestaan?

2) a) Hoe luidt de n-punts integratie-formule van Newton-Cotes? Welke naam heeft deze formule voor de bijzondere gevallen $n = 2$ en $n = 3$? Welke waarde hebben de coëfficiënten in deze gevallen en hoe worden ze berekend?

b) Wat is Richardson-correctie? Welke formule verkrijgt men als men Richardson-correctie toepast op de 3-punts Newton-Cotes formule?

c) Hoe verloopt de berekening van $\int_a^b f(x)dx$ met automatisch variërende staplengte?

3) a) Waarom is Gauss' eliminatie zonder verwisselingen niet goed? Geef als voorbeeld een lineair stelsel van de orde 2, waarvoor het mis gaat.

b) Wat is een pivot? Hoe worden bij Gauss' eliminatie de pivots gekozen?

c) Wat is de inverse van een matrix? Hoe wordt een inverse matrix numeriek berekend?

Z.O.Z.

4) a) Definiëer het begrip "partiële afgeleide".
   b) Beschrijf de iteratie-formule van Newton voor het oplossen van een stelsel van 2 vergelijkingen met 2 onbekenden.
   c) Beschrijf de methode van Bairstow ter berekening van een paar nulpunten van een polynoom.

5) a) Wat is een gelijkvormigheids-transformatie en wanneer heten 2 matrices gelijkvormig?

   b) Hoe luidt een iteratie-stap van de methode van Jacobi ter bepaling van de eigenwaarden van een reële symmetrische matrix? In het bijzonder: welke gedaante heeft de transformerende matrix en hoe wordt de draaiingshoek gekozen?

   c) Beschrijf een strategie, die de volgorde der af te werken matrix-elementen in Jacobi's methode op een geschikte wijze vastlegt.

Cursus WRA

T.J. Dekker

22-3-1966

Proefwerk deel I

1) De functie y(x) voldoet aan de diff. vergelijking

$$y'' + xy' + x^2y = 0,$$

terwijl y(0) = 1 en y'(0) = 0.
Gevraagd wordt met behulp van reeks-ontwikkeling de waarde van y in 4 decimalen nauwkeurig te bepalen voor x = 0.5 en x = 0.6.

2) Bereken met de derde orde formule van Kutta de oplossing van het beginwaarde-probleem

$$y' = -2y/x \quad , \quad y(1) = 1$$

voor x = 1.1(0.1)1.5. Doe de berekening in 5 decimalen.
Bereken vervolgens, eveneens in 5 decimalen, het verschil van de aldus verkregen oplossing met de exacte oplossing.

Cursus WRA 22-3-1966

T.J. Dekker

Proefwerk deel II

Maak opgaven 3 en 4 en hetzij 5 hetzij 6

3) a) Wat is de waarde van de integer variabele a na beëindiging van het volgende stukje programma:

"integer procedure M(k); value k; integer k;
M := if k = 1 then 2 else a × M(k-1);
a := 1; a := M(7)".

b) Dezelfde vraag na het volgende stukje programma:

"procedure EEN(x); value x; integer x; x := 1;
for a := 0 do EEN(a); EEN(a)".

c) Voor welke waarden van de integer variabele n wordt de statement S uitgevoerd in de ALGOL statement

"for n := 2 step if n < 10 then 13 else -5 until 8 do S"?

Hierbij wordt verondersteld, dat S de variabele n niet wijzigt.

4) Zij in een ALGOL-programma gedeclareerd de procedure ZERO (AP 230, pag. 158) en bovendien een array A [0 : 10]. Binnen de scope hiervan komt het volgende blok voor:

```
"begin real w; integer t; array real [1 : 2];
      real procedure fun;
      begin t := t + 1; A [t] := fun := w end;
      t := 0; real [1] := 0; real [2] := 0.001;
      A [t] := ZERO(w, -1, 2, fun, real)
end".
```

Welke getallen staan, na uitvoering van dit blok, in het array A?

5) a) Schrijf een stuk programma in de vorm van een blok dat met behulp van de niet-locale procedure RK1 (AP 252 pag. 232) de oplossing y(x) van het beginwaarde-probleem

$$y' = \frac{1}{x+y}, \quad y(0) = 1$$

berekent in de punten x = 0(0.5)2.

De gewenste absolute precisie is 4 decimalen.
Het blok moet de gevraagde functie-waarden afleveren in een passend niet-locaal array. Alle andere benodigde arrays en variabelen moeten in dit blok gedeclareerd worden.

b) Schrijf de in deze procedure RK1 gebruikte Runge-Kutta-formule uit, met name de formules voor Δy en de benodigde k's. Welke formule wordt gebruikt voor de som der hoogste orde termen? Hoe luidt de formule voor de tolerantie?

6) a) Schrijf een blok, dat met behulp van de niet-locale procedure RK uit het Revised report on ALGOL 60 oplost het stelsel 2e orde differentiaal-vergelijkingen

$$\frac{d^2x}{dt^2} = -\frac{cx}{r^3}, \quad \frac{d^2y}{dt^2} = -\frac{cy}{r^3},$$

waarbij $r^2 = x^2 + y^2$, $c = .2855837$ en de beginvoorwaarden zijn

$$x(0) = 1, \qquad x'(0) = 0$$
$$y(0) = 0, \qquad y'(0) = 1.$$

Gevraagd worden de waarden van x en y en hun eerste afgeleide voor $t = 0(0.2)20$. Deze waarden moeten worden afgeleverd in passende niet-locale arrays.
Alle andere benodigde arrays en variabelen moeten in dit blok worden gedeclareerd.
De gewenste absolute precisie bedraagt 5 decimalen.

b) Welke formule wordt in deze procedure gebruikt en hoe wordt in elke stap de staplengte bepaald? Welke onvolkomenheden merkt U in deze procedure op?

Cursus WRA
T.J. Dekker

Practicum proefwerk

21-6-1966
13.00 - 15.15 uur

(Uit Examen voor Wetenschappelijk Rekenen A, 1963)

1. Van de functie $y = f(x)$ is de volgende tabel gegeven:

x	y
0,00	0,00000
0,01	0,06253
0,04	0,13350
0,09	0,21472
0,16	0,30860
0,25	0,41835
0,36	0,54836
0,49	0,70482
0,64	0,89669
0,81	1,13756
1,00	1,44892

Bereken op verantwoorde wijze:

a) de waarde van y voor $x = 0,29$,

b) de afgeleide $\frac{dy}{dx}$ voor $x = 0,16$,

c) de integraal $\int_0^1 y\,dx$.

2. Tabuleer in vier decimalen de oplossing van de differentiaalvergelijking

$$\frac{dy}{dx} = \frac{x + y^2}{y}$$

over het gebied $x = 1,5(0,1)1,7$ als de volgende waarden van y en $\frac{dy}{dx}$ gegeven zijn:

x	1,0	1,1	1,2	1,3	1,4
y	1,1038	1,3115	1,5347	1,7757	2,0371
$\frac{dy}{dx}$	2,0098	2,1502	2,3166	2,5078	2,7244

Cursus WRA
T.J. Dekker

Theorie proefwerk

21-6-1966
15.30 - 17.30 uur

1. a) Beschrijf de matrix-maal-vector iteratie ter bepaling van een eigenwaarde en bijbehorende eigenvector van een matrix.

   b) Wanneer convergeert dit proces en wat is dan de convergentie-snelheid?

   c) Beschrijf een methode, waarmee men, na het vinden van de absoluut grootste eigenwaarde, een volgende eigenwaarde met eigenvector kan vinden.

2. a) Beschrijf hoe men een differentiaal-vergelijking van de orde n kan herleiden tot een stelsel van n eerste orde differentiaal-vergelijkingen.

   b) Beschrijf de Taylor-reeks methode voor het oplossen van het tweede-orde beginwaarde-probleem

$$y'' = f(x,y,y') \ , \quad y(x_0) = a \ , \quad y'(x_0) = b.$$

   c) Noem voor- en nadelen van de Taylor-reeks methode in vergelijking met Runge Kutta en de meerstap-formules.

3. De interpolatie-formule van Stirling heeft de volgende coëfficiënten voor $i = 0, 1, 2, \ldots$:

$$S_{2i}(p) = \frac{p}{2i} \binom{p + i - 1}{2i - 1} \ , \quad S_{2i+1}(p) = \binom{p + i}{2i + 1} .$$

   Schrijf de formule uit tot en met de δ^4-term en leidt hieruit de volgende formules af

   a) een centrale differentiatie-formule voor $f'(x_n)$, als x_n een tabel-argument is.

   b) een centrale integratie-formule voor $y_{n+1} - y_{n-1}$, als y voldoet aan de differentiaal-vergelijking $y' = f(x,y)$.

   c) een centrale integratie-formule voor $\delta^2 y_n$, als y voldoet aan de differentiaal-vergelijking $y'' = f(x,y)$.

4. Zij gegeven het randwaarde-probleem

$$y'' + xy' - 3x^2y = 2x,\ y(0) = 1,\ y(1) = -4.$$

Hoe luidt, bij verdeling van het interval $[0,1]$ in 5 gelijke stukken, het stelsel lineaire vergelijkingen, waarvan de oplossing een (eerste) benadering geeft van de functie-waarden $y_1, \ldots, y_4$? M.a.w. gevraagd wordt de matrix en de rechterlid-vector op te schrijven, niet echter het stelsel op te lossen.