

LINEAIRE MEERSTAPSMETHODEN VOOR GEWONE DIFFERENTIAALVERGELIJKINGEN

P.W. HEMKER

Een vergelijking van de vorm

$$(1.1) \quad F(x, y, \frac{dy}{dx}, \frac{d^2y}{dx^2}, \dots, \frac{d^ny}{dx^n}) = 0$$

is een n-de orde gewone differentiaalvergelijking. De orde van de differentiaalvergelijking, n, is de orde van de hoogste afgeleide welke in (1.1) voorkomt. Als alle nevenvoorwaarden, zoals de waarden van $y, y', \dots, y^{(n-1)}$, worden gegeven voor dezelfde waarde van de onafhankelijke variabele x_0 , is het probleem een *beginwaardeprobleem*. Als de noodzakelijke nevenvoorwaarden niet allen voor hetzelfde punt gegeven worden noemen we het probleem een *randwaardeprobleem*.

Iedere n-de orde gewone differentiaalvergelijking kan geschreven worden als een stelsel eerste orde differentiaalvergelijkingen. Hiervoor definiëren we de variabelen

$$\begin{aligned} \frac{dy}{dx} &= v_1 \\ \frac{dv_1}{dx} &= v_2 = \frac{d^2y}{dx^2} \\ &\vdots \\ \frac{dv_{n-1}}{dx} &= v_n = \frac{d^ny}{dx^n} \end{aligned}$$

en substitueren we v_1, v_2, \dots, v_{n-1} en $d^ny/dx^n = dv_{n-1}/dx$ in de vergelijking (1.1). Hieruit blijkt dat we ons kunnen beperken tot het oplossen van stelsels eerste orde differentiaalvergelijkingen.

Opmerking

In enkele gevallen waarin vergelijking (1.1) een bijzondere structuur

bezit, kan het nuttig zijn deze reductie tot een stelsel niet uit te voeren, maar gebruik te maken van deze bijzondere structuur. Dit kan in het bijzonder het geval zijn als een aantal van de afgeleiden $d^i y/dx^i$ ($0 \leq i < n$) niet expliciet in (1.1) voorkomt.

1.1. De constructie van een oplossing

Laten we ons eerst tot een enkele eerste orde differentiaalvergelijking beperken. Het zal later blijken dat uitbreiden tot een stelsel differentiaalvergelijkingen geen extra moeilijkheden oplevert. Zij gegeven de differentiaalvergelijking

$$(1.3) \quad \frac{dy}{dx} = f(x,y), \quad f: \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$$

We kunnen de functie f in een tekening voorstellen als een verzameling van richtingen voor iedere waarde van de onafhankelijke variabele x en van de afhankelijke variabele y . We nemen aan dat f een continue functie is van x en y , welke bovendien aan een Lipschitz-voorwaarde voldoet:

$$(1.4) \quad \exists k > 0 \quad \forall x, y_1, y_2 \quad |f(x, y_1) - f(x, y_2)| \leq k |y_1 - y_2|.$$

Als nu een punt A gegeven is, kunnen we de oplossing eenvoudig tekenen door, met toenemende waarden van x , een baan in het x - y -vlak te volgen waarvan de richting gelijk is aan die welke voorgeschreven wordt door f .

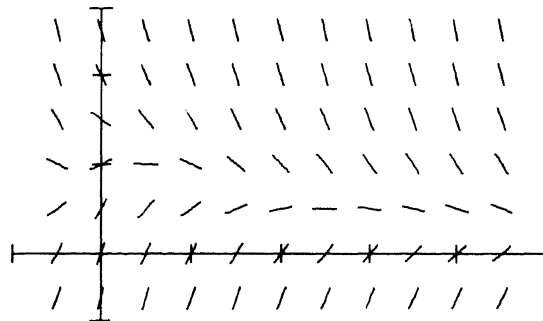


fig. 1.1. Het richtingsveld van de differentiaalvergelijking $dy/dx = -2.5y + (5x+3)(x+1)^{-2}$

Numerieke methoden voor gewone beginwaardeproblemen komen er dan ook op neer dat we de oplossing van de differentiaalvergelijking benaderen met een kromme die zich dicht langs de richtingen van het richtingsveld aansluit. Deze kromme wordt bepaald door stap voor stap een volgend punt (x_{n+1}, y_{n+1}) te berekenen.

Wanneer men te maken krijgt met een beginwaardeprobleem waarvan de structuur niet direct een eigenaardig karakter vertoont, is het een goed gebruik in eerste instantie te trachten dit probleem op te lossen met een standaardmethode. Een aantal algoritmen welke zich hier goed voor lenen, zijn bijvoorbeeld de Runge-Kutta methoden zoals ze staan beschreven in Zonneveld [1964]. (Tijdens deze cursus zullen we op deze methoden niet verder ingaan.) Een aantal moeilijkheden kan zich echter voordoen. Het meest voorkomende is het verschijnsel van de "*stijve differentiaalvergelijking*". Dit treedt op wanneer het gestelde probleem een analytische oplossing heeft waarvan sommige componenten een zeer stabiel karakter hebben. In dat geval zullen deze standaardmethoden, vanwege de eis van numerieke stabiliteit, gedwongen zijn de numerieke integratie voort te zetten met zeer kleine staplengte. De belangrijke begrippen *analytische (inherente) stabiliteit* en *numerieke stabiliteit*, welke hier naar voren komen, zullen we hieronder toelichten.

1.2. Analytische en numerieke stabiliteit

De oplossing $y(x)$ van een differentiaalvergelijking hangt, behalve van de functie f (zie (1.3)), ook af van een gegeven beginvoorwaarde: $y(a) = b$. Beschouwen we twee oplossingen y_1 en y_2 van een differentiaalvergelijking met beginvoorwaarden welke een weinig verschillen

$$\begin{aligned} \frac{d}{dx}y_1(x) &= f(x,y), & y_1(a) &= b \\ \frac{d}{dx}y_2(x) &= f(x,y), & y_2(a) &= b + \epsilon \end{aligned}$$

dan noemen we de differentiaalvergelijking *analytisch* (of *inherent*) *stabil* op het interval (a,c) als geldt

$$x \in (a,c) \Rightarrow |y_1(x) - y_2(x)| < \epsilon$$

Een belangrijke grootheid, welke de stabiliteit in de omgeving van een punt in het x - y vlak bepaalt (*locale stabiliteit*), is de partiële afgeleide f_y .

Schrijven we

$$\begin{aligned} \frac{d}{dx} (y_1(x) - y_2(x)) &= f(x, y_1) - f(x, y_2) = \\ &= f_y(x, y_1)(y_1 - y_2) + o(y_1 - y_2), \end{aligned}$$

dan zien we dat het gedrag van de verschilfunctie $v(x) = y_1(x) - y_2(x)$ weer beschreven wordt door een differentiaalvergelijking en dat voor kleine waarden van $v(a) = \epsilon$ bij benadering geldt

$$(1.5) \quad \frac{d}{dx} v(x) = f_y(x, y) v(x)$$

ofwel

$$v(x) = v(a) \cdot e^{\int_a^x f_y(\xi, y(\xi)) d\xi}.$$

Hieruit volgt direct dat een differentiaalvergelijking in ieder geval stabiel is op die plaats waar $f_y(x, y) < 0$

Voor een stelsel differentiaalvergelijkingen

$$\frac{d}{dx} y_i = f_i(x, y_1, y_2, \dots, y_n), \quad 1 \leq i \leq n,$$

laat het begrip *partiële afgeleide* zich uitbreiden tot het begrip *Jacobiaan*, de matrix van partiële afgeleiden

$$J(x, y) = (\partial f_i / \partial y_j)(x, y).$$

Het stabiliteitsgedrag wordt geheel bepaald door deze Jacobiaan. Het stelsel differentiaalvergelijkingen is nu stabiel als voor alle *eigenwaarden van de Jacobiaan* λ_i geldt $\text{Re } \lambda_i < 0$.

Wanneer de partiële afgeleide $f_y(x, y)$ (of de Jacobiaan $J(x, y)$) onafhankelijk is van de argumenten x en y spreekt men van *lineaire* (een lineair stelsel) *differentiaalvergelijkingen*. Een niet-lineaire vergelijking heeft een stabiliteitsgedrag dat afhankelijk is van de plaats in het x - y vlak.

Numerieke stabiliteit is een wezenlijk ander begrip. Zegt inherente stabiliteit iets over de analytische oplossing, numerieke stabiliteit zegt iets over de numerieke berekening. Bij een rekenproces worden telkens fouten geïntroduceerd zoals *afbrekfouten* en *afrondfouten*. Een numeriek

proces heet nu numeriek stabiel wanneer het min of meer ongevoelig is voor deze fouten, zodat geen opeenhoping van gemaakte fouten ontstaat. We onderscheiden (1) *absolute stabiliteit*: de numerieke fout wordt in absolute waarde steeds kleiner, en (2) *relatieve stabiliteit*: de numerieke fout blijft klein ten opzichte van het gewenste resultaat van de berekening. Een numeriek proces heet instabiel wanneer de gemaakte fouten versterkt worden en daardoor op den duur het gewenste resultaat kunnen verdringen.

We zullen ons op deze plaats niet bezighouden met het analyseren van het stabiliteitsgedrag van de verschillende methoden voor het oplossen van gewone differentiaalvergelijkingen. We volstaan hiertoe met verwijzen naar een aantal boeken: Henrici [1962], Gear [1971] en MC Syllabus 15.1 [1972]. Een enkele opmerking over numerieke stabiliteit moet hier echter toch gemaakt worden.

Bij een theoretische behandeling van de numerieke stabiliteit van methoden voor gewone differentiaalvergelijking, worden bijna uitsluitend lokaal lineaire beschouwingen gegeven, d.w.z. men beperkt zich tot een klein gebiedje in het x - y vlak (juist dat gebiedje waar men de oplossing wil construeren) en men neemt aan dat de Jacobiaan in dat gebiedje praktisch constant is.

1.3. Het stabiliteitsgebied

Het stabiliteitsgedrag van een numerieke methode, bij het oplossen van een bepaald probleem, hangt ten nauwste samen (1) met het karakter van het op te lossen probleem, met name van de Jacobiaan in de omgeving van de oplossing, $J(x, y(x))$, en (2) met de staplengte h_n waarmee de integratie uitgevoerd wordt. De stabiliteit van een bepaalde methode wordt gekarakteriseerd door een gebied S in het complexe vlak. Een numeriek stabiel proces wordt verkregen wanneer voor iedere stap uit het integratieproces en voor iedere eigenwaarde λ_i van de Jacobiaan geldt dat $h_n \lambda_{i,n} \in S$.

Een belangrijk onderscheid valt te maken tussen *expliciete* methoden en *impliciete* methoden om gewone differentiaalvergelijkingen op te lossen. Bij expliciete methoden wordt in iedere stap van het integratieproces y_{n+1} berekend door een vast aantal malen de functie $f(x, y)$ te evalueren waarbij tevoren de argumenten x en y bekend zijn. Bij een impliciete methode moet voor iedere integratiestap een (in het algemeen niet-lineaire) vergelijking opgelost worden van de vorm

$$G(x_{n+1}, y_{n+1}, f(x_{n+1}, y_{n+1})) = 0.$$

Waar expliciete methoden- zoals standaard Runge-Kutta - door hun (betrekkelijke) eenvoud de voorkeur schijnen te verdienen, blijken deze alle een begrensde stabiliteitsgebied te bezitten. Voor beginwaardeproblemen waar $|\lambda_1|$ grote waarden aanneemt, kan het voorkomen dat de integratie met kleinere stappen moet worden uitgevoerd dan men op grond van nauwkeurigheidscriteria wenst (stijve differentiaalvergelijkingen). Sommige impliciete methoden hebben dit nadeel niet.

1.4. Extra informatie voor efficiënter rekenen

Hoewel het in principe mogelijk is een beginwaardeprobleem numeriek te integreren met een algoritme die als enige informatie over het gestelde probleem de definiërende functie f en de bijbehorende startwaarden gebruikt, zullen we dikwijls aan de algoritme extra informatie willen verschaffen om de berekening efficiënter te kunnen uitvoeren. Informatie welke hiervoor o.a. in aanmerking komt is

- (1) een analytische uitdrukking voor de Jacobiaan,
- (2) de ligging van de eigenwaarden in het complexe vlak.

Dikwijls is een redelijke benadering van één van deze twee al voldoende om de efficiency van de berekening aanzienlijk te verhogen. De Jacobiaan wordt gebruikt in semi-impliciete methoden en wordt ook bij impliciete methoden gebruikt om de vergelijking $G = 0$ op te lossen. Wanneer de ligging van de eigenwaarden bekend is, kan dit gebruikt worden om de techniek van het "exponentieel fitten" toe te passen. Deze techniek die ook zeer geschikt is voor het oplossen van stelsels gewone differentiaalvergelijkingen zal behandeld worden in een volgend hoofdstuk.

1.5. Lineaire meerstapsmethoden

In dit hoofdstuk zullen we een beschrijving geven van de meerstapsmethoden volgens Adams-Moulton [1883,1926], en Curtiss-Hirschfelder [1952], zoals ze later zijn uitgewerkt door Nordsieck [1962] en Gear [1968]. Deze methoden behoren tot de lineaire meerstapsmethoden, een klasse van methoden waar een uitgebreide theorie over bestaat. (Henrici [1962] Gear [1971], MC Syllabus 15.1 [1972].) De nieuwere ontwikkelingen zijn vooral gericht op het efficiënt oplossen van *stelsels stijve* differenti-

aalvergelijkingen. We willen hier op de theoretische aspecten niet ingaan, deze kunnen gevonden worden in bovengenoemde boeken. We zullen hier de nadruk leggen op de structuur en het gebruik van procedures zoals ze worden gegeven door Gear [1971] (in FORTRAN) en door Hemker [1971] (in ALGOL 60).

Beschouw het stelsel differentiaalvergelijkingen

$$(1.6) \quad \vec{y}' = \vec{f}(\vec{y}) = \vec{f}(y_1, y_2, \dots, y_m),$$

geschreven in vectornotatie. Zij $h > 0$ een vast gekozen staplengte en laat

$$x_n = x_0 + nh, \quad n = 0, 1, 2, \dots, N,$$

een rij punten zijn met gelijke tussenruimte. We schrijven \vec{y}_n voor de benadering van $\vec{y}(x_n)$ ($n = 0, 1, 2, \dots, N$), de oplossing van (1.6) met een gegeven beginvoorwaarde \vec{y}_0 :

$$\vec{y}_n = \vec{y}(x_n) + \vec{\epsilon}_n = [y_{1n}, y_{2n}, \dots, y_{mn}].$$

Wanneer we aannemen dat $\vec{\epsilon}_0 = \vec{\epsilon}_1 = \dots = \vec{\epsilon}_{n-1} = 0$ dan heet $\vec{\epsilon}_n$ de *locale discretiseringsfout* van de methode en de methode heet *nauwkeurig van de orde p* als geldt

$$\vec{\epsilon}_n = O(h^{p+1}).$$

Een lineaire meerstapsmethode om (1.6) op te lossen, wordt gedefinieerd door de vectorvergelijking

$$(1.7) \quad \sum_{i=0}^k \alpha_i \vec{y}_{n-i} + h \sum_{i=0}^k \beta_i \vec{f}(\vec{y}_{n-i}) = \vec{0} \quad (n \geq k)$$

Waarbij k startwaarden $\vec{y}_0, \vec{y}_1, \dots, \vec{y}_{k-1}$ nodig zijn. Een methode wordt gedefinieerd door de keuze van de parameters α_i, β_i ($i=0, 1, \dots, k$); er bestaan methoden die een hoge orde van nauwkeurigheid bezitten en stabiel zijn voor voldoende kleine h . Twee typen van deze methoden komen als bijzonder gunstig naar voren. Dit zijn (1) de methoden welke de waarden $f(\vec{y}_{n-i})$ ($0 \leq i \leq k$) door een polynoom verbinden (de Adams-Moulton methoden; $\alpha_i = 0$ voor $i = 0, \dots, k$). Deze bereiken de orde van nauwkeurigheid $p = k+1$ maar bezitten een - met hogere orde afnemend - begrensd stabiliteitsge-

bied. En daarnaast (2) de methoden welke de waarden y_{n-1} ($0 \leq i \leq k$) met een polynoom verbinden (de Curtiss-Hirschfelder methoden; $\beta_i = 0$ voor $i = 1, 2, \dots, k$). Deze laatste methoden, met een orde van nauwkeurigheid $p = k$, zijn slechts stabiel voor $p \leq 6$. Het stabiliteitsgebied van deze methoden strekt zich echter (voor $p \leq 6$) uit over de gehele negatieve reële as.

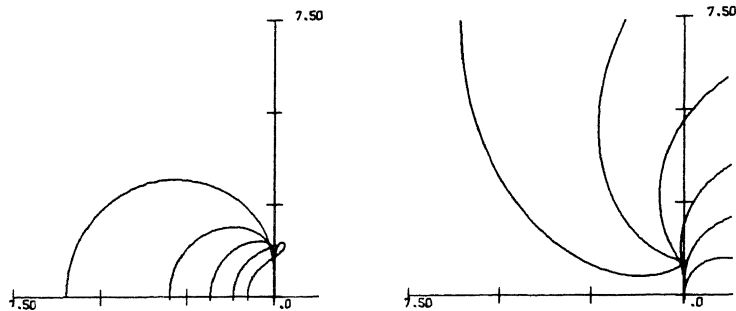


fig. 1.2. Stabiliteitsgebieden voor Adams-Moulton methoden (links) en voor Curtiss-Hirschfelder methoden (rechts)

De voorwaarden voor consistentie en het bepalen van stabiliteitsgebieden voor de lineaire meerstapsmethoden wordt uitgebreid behandeld in hoofdstuk IV van MC Syllabus 15.1 [1972].

We beschouwen de k -stapsmethoden (1.7) met $\alpha_0 \neq 0$, deze kunnen worden geschreven als

$$(1.8) \quad \vec{y}_n = h \vec{f}(\vec{y}_n) + \vec{\phi}, \quad \beta = -\beta_0/\alpha_0,$$

waarin $\vec{\phi}$ een lineaire combinatie is van $\vec{y}_{n-1}, \vec{y}_{n-2}, \dots, \vec{y}_{n-k}, \vec{f}(\vec{y}_{n-1}), \dots, \vec{f}(\vec{y}_{n-k})$. Als $\beta = 0$ is de methode expliciet en levert de berekening van y_n geen moeilijkheden op. Als $\beta \neq 0$ is de methode impliciet en is (1.8) een stelsel van m (niet-lineaire) vergelijkingen met m onbekenden. De gebruikelijke iteratieve methode om dit stelsel op te lossen wordt gegeven door

$$(1.9) \quad {}_{r+1}\vec{y}_n = h\beta\vec{f}({}_r\vec{y}_n) + \vec{\phi}, \quad r = 0, 1, 2, \dots$$

waarin ${}_0\vec{y}_n$ een beginapproximatie is van \vec{y}_n . Het is eenvoudig in te zien dat dit iteratieproces convergeert voor kleine waarden van $|h\lambda_i|$ ($\lambda_1, \lambda_2, \dots, \lambda_n$ de eigenwaarden van de Jacobiaan). Voor stijve stelsels houdt dit echter in dat de staplengte klein gekozen moet worden. Een an-

dere, geschikte, iteratieve methode om (1.8) op te lossen is het gewijzigde Newton-Raphson iteratieproces, dat gedefinieerd wordt door

$$(1.10) \quad \vec{y}_{n+1} = \vec{y}_n - (I - h\beta J^*)^{-1} (\vec{y}_n - \vec{\phi} - h\beta \vec{f}(\vec{y}_n)),$$

waarin J^* een benadering is van de Jacobiaan $J(\vec{y}_n)$. Dit schema convergeert als alle eigenwaarden van de inverse matrix in absolute waarde kleiner dan één zijn. Zijn $\lambda_1, \lambda_2, \dots, \lambda_m$ de eigenwaarden van J^* dan zijn

$$\frac{1}{1 - h\beta\lambda_i} \quad i = 1, 2, \dots, m,$$

de eigenwaarden van $(1 - h\beta J^*)^{-1}$. De iteratiemethode is dan geschikt voor stijve stelsels. We merken op dat men bij het uitvoeren van dit proces de beschikking moet hebben over een benadering van de Jacobiaan J .

De vectorfunctie $\vec{\phi}$, gedefinieerd door (1.8) is in het geval van de Curtiss-Hirschfelder methoden een lineaire combinatie van \vec{y}_{n-i} ($0 < i \leq k$) en in het geval van de Adams-Moulton methoden een lineaire combinatie van y_{n-1} en $f(y_{n-i})$ ($0 < i \leq k$). Een lineaire transformatie voert bij de Curtiss-Hirschfelder methoden de vector $(y_{n-1}, y_{n-2}, \dots, y_{n-k})$ over in de vector der achterwaartse differenties

$$\begin{aligned} & (y_{n-1}, \nabla y_{n-1}, \nabla^2 y_{n-1}/2, \dots, \nabla^{k-1} y_{n-1}/(k-1)!) \cong \\ & \cong (y_{n-1}, h y'_{n-1}, \frac{h^2}{2} y''_{n-1}, \dots, h^{k-1} y_{n-1}^{(k-1)}/(k-1)!). \end{aligned}$$

Bij de Adams-Moulton methode kan $(y_{n-1}, f_{n-1}, f_{n-2}, \dots, f_{n-k})$ worden overgevoerd in

$$\begin{aligned} & (y_{n-1}, f_{n-1}, \nabla f_{n-1}, \dots, \nabla^{k-1} f_{n-1}/(k-1)!) \cong \\ & \cong (y_{n-1}, h y'_{n-1}, \frac{h^2}{2} y''_{n-1}, \dots, h^k y_{n-1}^{(k)}/k!). \end{aligned}$$

We kunnen $\vec{\phi}$ derhalve ook beschouwen als een lineaire combinatie van $h^i y_{n-1}^{(i)}/i!$ ($0 \leq i \leq p-1$)

We kunnen nu eenvoudig, op expliciete wijze, beginschattingen ${}_0 y_n, h {}_0 y_n, \dots, h^{p-1} {}_0 y_n^{(p-1)}/(p-1)!$ berekenen door directe extrapolatie, bij voorbeeld met behulp van de formules

$${}^0\vec{y}_n = y_{n-1} + h y'_{n-1} + h^2 y''_{n-1}/2 + \dots + h^{p-1} y_{n-1}^{(p-1)}/(p-1)!$$

en

$$h {}^0\vec{y}'_n = h y'_{n-1} + h^2 y''_{n-1} + \dots + h^{p-1} y_{n-1}^{(p-1)}/(p-2)!$$

Nadat op deze wijze beginschattingen verkregen zijn, worden \vec{y}_n en $h\vec{y}'_n$ iteratief verbeterd met behulp van (1.10); ${}_r\vec{y}'_n$ ($r > 0$) wordt gedefinieerd door

$$h\beta {}_r\vec{y}'_n \stackrel{d}{=} h\beta \vec{f}({}_{r-1}\vec{y}_n) = {}_r\vec{y}_n - \vec{\phi}.$$

Wanneer dit iteratieproces beëindigd is kunnen de waarden $h^i y_{n+1}^{(i)}/i!$ ($2 \leq i \leq p$) berekend worden door de beginschattingen te corrigeren met een term $a_i \times (y_n - y_n)$. De factor a_i is afhankelijk van de gekozen methode en van de gekozen orde. (Voor het bewijs hiervan en voor de berekening van a_i zij verwezen naar MC Syllabus 15.1 [1972].)

Een volledige rekenproces in één integratiestap luidt dus als volgt:

$$\begin{pmatrix} {}^0\vec{y}_n \\ h {}^0\vec{y}'_n \\ h^2 {}^0\vec{y}''_{n-1}/2 \\ \vdots \\ h^{p-1} {}^0\vec{y}_{n-1}^{(p-1)}/(p-1)! \end{pmatrix} = (A_{ij}) \begin{pmatrix} \vec{y}_{n-1} \\ h \vec{y}'_{n-1} \\ h^2 \vec{y}''_{n-1}/2 \\ \vdots \\ h^{p-1} \vec{y}_{n-1}^{(p-1)}/(p-1)! \end{pmatrix}$$

met $A_{ij} = \begin{cases} \binom{j}{i} & \text{als } i \leq j \\ 0 & \text{als } i > j \end{cases}$

$$\left. \begin{aligned} {}_r\vec{d} &= (I - h\beta J^*)^{-1} (h\vec{f}({}_{r-1}\vec{y}_n) - h {}_r\vec{y}'_n) \\ {}_{r+1}\vec{y}_n &= {}_r\vec{y}_n + \beta {}_r\vec{d} \\ h \cdot {}_{r+1}\vec{y}'_n &= h \cdot {}_r\vec{y}'_n + {}_r\vec{d} \end{aligned} \right\} r = 0, 1, \dots, R-1$$

$$(1.11) \quad \left\{ \begin{array}{l} \vec{y}_n = R \vec{y}_n \\ h \vec{y}'_n = h R \vec{y}'_n \\ h^i \vec{y}_n^{(i)} / i! = h^i \vec{y}_n^{(i)} + a_i (h y'_n - h \vec{y}'_n) \quad (2 \leq i \leq k). \end{array} \right.$$

Wanneer de orde van de formule bij de volgende stap hoger gekozen wordt, wordt bovendien berekend

$$h^{k+1} \vec{y}^{(k+1)} / (k+1)! = a_k (h \vec{y}'_n - h \vec{y}'_n) / (k+1).$$

1.6. Een strategie voor het toepassen van lineaire meerstapsformules

We hebben nog een groot aantal vrijheden als we dit proces willen toepassen. Zo moeten we nog beslissen

- (1) wanneer we een nieuwe J^* zullen berekenen,
- (2) welke methode we zullen toepassen: Adams-Moulton of Curtiss-Hirschfelder, en
- (3) met welke orde en welke staplengte we zullen integreren.

In het kort zullen we de strategie vermelden die in de procedure MULTISTEP gerealiseerd is:

1. We kiezen een minimale en een maximale staplengte (een minimale staplengte is o.a. nodig om ons van een eindig proces te verzekeren).
2. Als we geen reden hebben om aan te nemen dat de vergelijking stijf is, integreren we in eerste instantie met de Adams-Moulton methode.
3. Bij niet-stijve differentiaalvergelijkingen (Adams-Moulton methode) nemen we eerst $J^* = 0$; blijkt dat het iteratieproces niet snel genoeg convergeert met de laatste J^* dan wordt de Jacobiaan ter plaatse geëvalueerd.
4. We starten de integratie met een eerste orde methode en met de minimale staplengte (een hogere orde methode kunnen we niet gebruiken omdat we niet over $h^i \vec{y}^{(i)} / i!$ ($i \geq 2$) beschikken).

5. We nemen minstens k equidistante stappen met een k -stapmethode.
6. We verlagen of verhogen de orde van nauwkeurigheid nooit met meer dan één tegelijk.
7. Na een aantal stappen met een p -de orde methode berekenen we $h^{(i)}$ ($i=p-1, p, p+1$), d.i. de staplengte die een voorgeschreven afbreekfout (ϵ) zal veroorzaken bij het gebruik van de i -de orde methode. De maximale $h^{(i)}$ wordt gekozen als nieuwe staplengte, in combinatie met de bijbehorende orde i . (Enige veiligheidsmarges zijn ingebouwd om overbodig heen- en terugspringen te verhinderen.) We maken er gebruik van dat de i -de orde afbreekfout evenredig is met

$$\begin{aligned} & \| h^p y_n^{(p)} / p! \| && \text{voor } i = p-1 \\ & \| \vec{R}_n - \vec{O}_n \| && \text{voor } i = p, \\ & \| (\vec{R}_n - \vec{O}_n) - (\vec{R}_{n-1} - \vec{O}_{n-1}) \| && \text{voor } i = p+1. \end{aligned}$$

8. In een aantal gevallen zullen we gedwongen zijn een integratiestap te verwerpen en een stap met kleinere staplengte opnieuw uit te voeren. Dit kan optreden (A) als het proces niet (voldoende snel) convergeert terwijl J^* een goede benadering is van J of (B) als de verkregen afbreekfout groter is dan de gewenste.

9. Wanneer bij de minimale staplengte de berekende afbreekfout groter blijft dan de gewenste, wordt dit waarschijnlijk veroorzaakt door stijfheid van de differentiaalvergelijking ^{*)}; wanneer dit optreedt bij het gebruik van een Adams-Moulton methode, wordt overgeschakeld op een Curtiss-Hirschfelder methode, wanneer dit optreedt bij een Curtiss-Hirschfelder methode wordt verder gerekend met de backward-Euler methode met de minimale staplengte. (N.B. de backward-Euler methode is de Curtiss-Hirschfelder methode van orde 1.) Wanneer in dit laatste geval niet aan het locale foutcriterium is voldaan, wordt dit in een output-parameter gemeld.

*) Stijfheid van de differentiaalvergelijking is een begrip dat zowel afhankelijk is van de vergelijking zelf, als van de eisen die de gebruiker aan de oplossing stelt: ϵ s en h_{\min} .

1.7. De procedure MULTISTEP

Tot slot geven we de gebruiksaanwijzing voor ALGOL 60 procedure MULTISTEP

Declaratie

```

procedure MULTISTEP (x,xend,y,hmin,hmax,ymax,eps,
    first,dd,fxyi,i,jacij,j,n,available,
    stiff);
value hmin,hmax,eps,xend,available,n;
Boolean available,stiff,first;
integer i,j,n;
real x,xend,hmin,hmax,eps,fxyi,jacij;
array y,ymax,dd;
<procedure body>;

```

Parameters

x : <variable>;
 Deze wordt o.a. gebruikt als Jensen-parameter voor fxyi en jacij; de onafhankelijke variabele;
 ingang: x0 de beginwaarde van het integratie interval;

xend : <expression>;
 de eindwaarde van het integratie-interval (xend > x);

y : <array identifier>; array y [0:7,1:n];
 de afhankelijke variabele;
 ingang: de beginwaarden voor het stelsel
 differentiaalvergelijkingen
 $y[0,i] := y[i](x_0)$;
 uitgang: y(xend);

hmin,hmax: <expression>;
 de minimale resp. de maximale stap waarmee de integratie wordt uitgevoerd;

eps : <expression>;
 de maximaal toegestane relatieve locale fout;

ymax : <array identifier>; array ymax[1:n];
 ingang : de toegestane absolute locale fout/eps;

uitgang:ymax [i] is de maximale waarde welke y[i]
 tijdens het integratie proces heeft aangenomen;

first : <identifiser>;
 bij een eerste aanroep van de procedure moet first:=
true opdat gestart kan worden met een eerste orde
 Adams-methode. Bij het verlaten van de procedure is
 first:= false zodat bij het voortzetten van de
 integratie de procedure voortgaat op de wijze die
 vanaf de laatste aanroep met first:= true de beste
 gebleken is (een hogere orde Adams- of Curtiss-methode);

dd : <array identifiser>; array dd[0:7,0:n];
 throughput en meldingen:
 dd[0,0] = 0 Adams-methode gebruikt,
 = 1 overgeschakeld op Curtiss-methode;
 dd[1,0] = 0 geen fouten opgetreden tijdens executie,
 = 1 bij de huidige hmin kan de nietlineariteit van het
 probleem niet gevolgd worden;
 dd[2,0] : aantal stappen waarbij met de huidige hmin
 de vereiste locale fout niet binnen de
 gestelde grenzen bleef;
 dd[3,0] : if dd[2,0] = 0 then 0 else
 schatting van de maximale locale fout;

i,j : <identifiser>;
 worden gebruikt als Jensen-parameter voor fxyi en
 jacij;

fxyi : <expression>;
 een uitdrukking afhankelijk van x,y,i welke de
 waarde van dy_i/dx geeft;

jacij : <expression>;
 een uitdrukking afhankelijk van x,y,i,j, welke
 (ad lib.) de waarde van $\partial(dy_i/dx)/\partial y_j$ geeft
 (de Jacobiaan van het stelsel);

available: <Boolean expression>;
 een uitdrukking welke aangeeft of door jacij de
 Jacobiaan ter beschikking gesteld wordt;

stiff : <Boolean expression>;
 als stiff:= true gebruikt de procedure direct
 methoden welke geschikt zijn voor het oplossen van
 stijve differentiaalvergelijkingen, zonder eerst de
 Adams-Moultonmethoden te proberen;

n : <expression>;
 het aantal vergelijkingen waaruit het stelsel bestaat.

1.8. Voorbeeld van een aanroep

Als voorbeeld van het gebruik van de procedure MULTISTEP geven we een gedeelte uit een programma waarin met verschillende waarden van de parameters eps en hmin de oplossing wordt berekend van het beginwaardeprobleem

$$\begin{aligned} dy/dx &= (-1000 \times (y+z-2) - 0.013) \times y \\ dz/dx &= -2500 \times (y+z-2) \quad \times z \end{aligned}$$

met de beginwaarden

$$y(0) = 1; \quad z(0) = 1.$$

De resultaten worden telkens afgedrukt voor $x = 0.005$ en $x = 50$

```

1  BEGIN COMMENT VOORLOOPBAND MULW+STEP (NR 128/72) (NR 3.3.8, MEI 1972);
2
3  PROCEDURE MULW+STEP(X,XEND,Y,HMIN,HMAX,YMAX,EPS, FIRST,DD,
4                      FXYI,I1,JACIJ,JJ,N,AVAILABLE,STIFF);
5  VALUE HMIN,HMAX,EPS,XEND,AVAILABLE,N;
6  BOOLEAN AVAILABLE,STIFF,FIRST; INTEGER I1,JJ,N;
7  REAL X,XEND,HMIN,HMAX,EPS,FXYI,JACIJ; ARRAY Y,YMAX,DD;
8
9  BEGIN OWN BOOLEAN WITH JACOBIAN,ADAMS;
10  OWN INTEGER KOLD; OWN REAL XOLD,HOLD;
11  BOOLEAN EVALUATE,EVALUATED,CONV;
12  INTEGER I,J,L,K,KNEW,MAXORDER,FAILS,SAME;
13  REAL M,CH,CHNEW,C,TOLCONV,TOLUP,TOL,TOLDOWN,ERROR,DFI;
14  ARRAY CONST[1:56],A[0:7],DELTA,LAST DELTA,DF[1:N],JAC[1:N,1:N];
15  INTEGER ARRAY P[1:N];
16
17  PROCEDURE METHOD;
18  BEGIN WITH JACOBIAN:= -ADAMS;
19  MAXORDER:= IF ADAMS THEN 7 ELSE 6;
20  I:= K:= 1;
21  IF ADAMS THEN
22  BEGIN FOR CONST[I]:= 1,1,17,2,1,5,1,5,24,12,1,5/12,1,75,
23          1/6,37,89,24,2,375,1,11/12,1/3,1/24,53,33,37,89,1,
24          251/720,1,29/24,39/72,5/48,1/120,70,08,53,33,3158,
25          95/288,1,137/120,625,17/96,025,1/720,87,97,70,08,
26          07407,19087/60480,1,1,225,203/270,49/192,7/144,
27          7/1440,1/5040,106,9,87,97,0139 DO I:= I + 1
28  END ELSE
29  BEGIN FOR CONST[I]:= 1,1,3,2,1,2/3,1,1/3,6,4,5,1,6/11,1,
30          6/11,1/11,9,167,7,333,0,5,48,1,7,2,02,12,5,
31          10,42,1667,120/274,1,225/274,85/274,15/274,1/274,
32          15,98,13,7,04167,180/441,1,58/63,5/12,25/252,
33          3/252,1/1764,19,6,17,15,008333 DO I:= I + 1
34  END
35  END METHOD;
36
37  PROCEDURE ORDER;
38  BEGIN IF K>MAX ORDER THEN BEGIN DO[1,0]:= 2; GO TO RETURN END;
39  J:= (K-1) * (K+8) / 2 + 1;
40  FOR I:= 0 STEP 1 UNTIL K DO A[I]:= CONST[I+J];
41  TOLUP := (EPS*CONST[J+K+1])^2;
42  TOL := (EPS*CONST[J+K+2])^2;
43  TOLDOWN:= (EPS*CONST[J+K+3])^2;
44  TOLCONV:= EPS/(2*N*(K+2));
45  EVALUATE:= WITH JACOBIAN;
46  SAME:= K+1
47  END ORDER;
48
49  PROCEDURE EVALUATE JACOBIAN;
50  BEGIN REAL R;
51  EVALUATE:= FALSE;
52  IF AVAILABLE THEN
53  BEGIN R:= -A[0] * M;
54  FOR I:= 1 STEP 1 UNTIL N DO
55  FOR JJ:= 1 STEP 1 UNTIL N DO JAC[I,JJ]:= JACIJ * R;
56  END ELSE

```



```

57      BEGIN REAL D; ARRAY FIXDY, FIX Y(1:N);
58      FOR I:=1 STEP 1 UNTIL N DO
59          BEGIN FIX Y(I):= Y(0,I); FIXDY(I):= FIXYI END;
60      FOR I:=1 STEP 1 UNTIL N DO
61          BEGIN D:= IF EPS>ABS(FIX Y(I)) THEN EPS*EPS
62                  ELSE EPS*ABS(FIX Y(I));
63                  Y(0,I):= Y(0,I) + D;
64                  R1:= - A(0) * H/D;
65          FOR I:=1 STEP 1 UNTIL N DO
66              JAC(I,I):=(FIXYI - FIXDY(I)) * R1
67              Y(0,I):= FIX Y(I)
68          END
69      END;
70      FOR I:=1 STEP 1 UNTIL N DO JAC(I,I):= JAC(I,I) + 1;
71      DET(JAC,N,P);
72      EVALUATED:= TRUE
73  END EVALUATE JACOBIAN;
74
75  PROCEDURE CALCULATE STEP AND ORDER;
76  BEGIN REAL A1,A2,A3;
77      SAME:= 10;
78      A1:= IF K<1 THEN 0 ELSE
79            0.75*(TOLDN/SUM(1,1,N,(Y(K,I)/YMAX(I))^2))+0.5/K;
80      A2:= 0.80*(TOL /ERROR) + 0.5/(K+1);
81      A3:= IF K<MAX ORDER - FAILS+0 THEN 0 ELSE
82            0.70*(TOLUP /SUM(1,1,N,((DELTA(I)-LAST DELTA(I))/
83                    YMAX(I))^2))+0.5/(K+2);
84      IF A1>A2 ^ A1>A3 THEN BEGIN KNEW:=K-1; CMNEW:=A1 END ELSE
85      IF A2>A3          THEN BEGIN KNEW:=K ; CMNEW:=A2 END ELSE
86                        BEGIN KNEW:=K+1; CMNEW:=A3 END
87  END CALCULATE STEP AND ORDER;
88
89  PROCEDURE SET;
90  BEGIN XOLD:= X; HOLD:= H; KOLD:= K; CH:= 1;
91      FOR I:=1 STEP 1 UNTIL N DO
92          FOR J:=0 STEP 1 UNTIL K DO DD(J,I):= Y(J,I)
93      END SET;
94
95
96  PROCEDURE RESET STEP;
97  BEGIN REAL C;
98      IF CH < HMIN/HOLD THEN CH:= HMIN/HOLD ELSE
99      IF CH > HMAX/HOLD THEN CH:= HMAX/HOLD;
100     XI:= XOLD; HI:= HOLD * CH; CI:= 1;
101     FOR J:=0 STEP 1 UNTIL K DO
102         BEGIN FOR I:=1 STEP 1 UNTIL N DO Y(J,I):= DD(J,I) * C;
103             CI:= C * CH
104         END;
105     SAME:= K + 1
106  END RESET STEP;
107
108  PROCEDURE BEGIN;
109  BEGIN FAILS:= 0; H:= HMIN;
110     FOR I:=1 STEP 1 UNTIL N DO Y(1,I):= FIXYI * H;
111     KI:= 1; ORDER:= SET
112  END BEGIN;
113
114
115  IF FIRST THEN
116  BEGIN FIRST:= FALSE; ADAMS:= -STIFF; METHOD;

```

```

117          BEGIN; EQB I:= 1,2,3 QD DD(I,0):= 0
118      END ELSE
119      BEGIN METHOD; K:= KOLD; ORDER; CH:= 1; RESET STEP END;
120
121      EQB LI:= 0 WHILE X<XEND DO
122      BEGIN IF X+H<XEND THEN XI:= X+H ELSE
123          BEGIN CH:= (XEND-X)/H; RESET STEP; XI:= XEND END;
124
125      COMMENT PREDICTION;
126      EQB I:=0 STEP 1 UNTIL K-1 DO
127      EQB J:= K-1 STEP -1 UNTIL 1 DO
128      ELMROW(1,N,J,J+1,Y,Y,1)
129      EQB I:= 1 STEP 1 UNTIL N QD DELTA(I):= 0;
130
131
132      COMMENT CORRECTION AND ESTIMATION LOCAL ERROR;
133      EQB LI:=1,2,3 DO
134      BEGIN EQB I:=1 STEP 1 UNTIL N DO DF(I):= FXVI*H - Y(I,1)
135          IF EVALUATE THEN EVALUATE JACOBIAN;
136          IF WITH JACOBIAN THEN SOL(JAC,N,P,DF);
137
138          CONV:= TBUE;
139          EQB I:= 1 STEP 1 UNTIL N DO
140          BEGIN DF(I):= DF(I);
141              Y(0,1):= Y(0,1) + A(0)*DF(I)
142              Y(1,1):= Y(1,1) + DF(I)
143              DELTA(I):= DELTA(I) + DF(I)
144              CONV:= CONV + ABS(DF(I) < TOLCONV * YMAX(I)
145          END;
146          IF CONV THEN
147          BEGIN ERROR:= SUM(1,1,N,(DELTA(I)/YMAX(I))^2);
148              EVALUATED:= EELSE; GO TO CONVERGENCE
149          END
150      END;
151
152      COMMENT ACCEPTANCE OR REJECTION;
153      IF -CONV THEN NO CONVERGENCE;
154      BEGIN IF WITH JACOBIAN + EVALUATED THEN ELSE
155          IF H>HMIN*1.0001 THEN
156          BEGIN WITH JACOBIAN:= WITH JACOBIAN + AVAILABLE;
157              CH:= CH/4
158          END ELSE
159          IF ADAMS THEN GO TO TRY CURTISS ELSE
160          BEGIN DD(1,0):= 1; GO TO RETURN END;
161
162          EVALUATED:= WITH JACOBIAN; RESET STEP
163      END ELSE CONVERGENCE;
164
165      IF ERROR>TOL THEN ERROR TEST NOT OK;
166      BEGIN FAILS:= FAILS + 1;
167          IF H>HMIN*1.0001 THEN
168          BEGIN IF FAILS>2 THEN
169              BEGIN KI:= 0; RESET STEP; BEGIN
170                  END ELSE
171                  BEGIN CALCULATE STEP AND ORDER;
172                      IF KNEW#K THEN BEGIN KI:= KNEW; ORDER END;
173                      CH:= CH#CHNEW/FAILS; RESET STEP
174                  END
175              END ELSE
176              IF ADAMS THEN TRY CURTISS;

```

```

BEGIN ADAMS:= FALSE; METHOD; ORDER; RESET STEP END ELSE
  IE K#1 THEN
  BEGIN KI:=1; ORDER; RESET STEP END ELSE

  BEGIN COMMENT VIOLATE EPS CRITERION;
  C:= EPS * SORT( ERROR/TOL );
  IE C>DD(3,0) THEN DD(3,0):= C;
  DD(2,0):= DD(2,0) * 1;
  GO TO ERROR TEST OK

  END
END ELSE

ERROR TEST OK:
BEGIN FAILS:= 0;
  IE K>2 THEN BEGIN FOR I:=1 STEP 1 UNTIL N DO
    ELMCOLVEC(2,K,I,Y,A,DELTA(I)) END;
  DOB I:= 1 STEP 1 UNTIL N DO IE ABS(Y(0,I))>YMAX(I)
    THEN YMAX(I):=ABS(Y(0,I));
  SAME:= SAME - 1;
  IE SAME=1 THEN BEGIN DOB I:=1 STEP 1 UNTIL N DO
    LAST DELTA(I):= DELTA(I) END ELSE
  IE SAME=0 THEN
  BEGIN CALCULATE STEP AND ORDER;
  IE CHNEW>1.1 THEN
  BEGIN SAME:= K + 1;
  IE KNEW#K THEN
  BEGIN IE KNEW>K THEN
  BEGIN DOB I:=1 STEP 1 UNTIL N DO
    Y(KNEW,I):= DELTA(I)*A(K)/KNEW
  END;
  K:= KNEW; ORDER
  END;
  IE CHNEW> HMAX/H THEN CHNEW:= HMAX/H;
  H:= H * CHNEW; C:= 1;
  DOB J:=1 STEP 1 UNTIL K DO
  BEGIN C:= C * CHNEW;
  DOB I:=1 STEP 1 UNTIL N DO
  Y(J,I):= Y(J,I)*C
  END
  END
  END;
  SET
  END
  END STEP;
RETURN: DD(0,0):= IE ADAMS THEN 0 ELSE 1; DD(4,0):= K
END MUL*+04SP;

```

```

237 COMMENT MIERVOOR STAAT DE DECLARATIE VAN PROCEDURE NUWY+EPS;
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284

```

```

      BEGIN FIRST; INTERSEB I,J,CF,CJ
      BEAL X,XEND,HMIN,EPS,R,TIMI
      ARRAY Y(0:17,1:2),YMAX(1:2),D(0:17,0:12);
      BEAL EQC FXVI
      BEGIN
        CF:=CF+1; FXVI:=I:=1
        THEN -Y(0,1)*0.013+1000*(Y(0,1)+Y(0,2)-2)
        ELSE -2500*(Y(0,1)+Y(0,2)-2)
      END;
      BEAL EQC JAC
      BEGIN
        CJ:=CJ+1; JAC:=I:=1
        THEN -1000*(Y(0,1))
        --(I:=J+2 THEN 0 ELSE (0.013+1000*(Y(0,1)+Y(0,2)-2)))
        ELSE -2500*(Y(0,2))
        --(I:=J+1 THEN 0 ELSE (2500*(Y(0,1)+Y(0,2)-2)))
      END;
      NLCR; PRINTTEXT(
      $ HMIN EPS X Y1 Y2 PUB JACS D(0,1) D(0,2) ORDER TIME MAX.LOC.ERROR$);
      COB HMIN:=4,--5,--6,--7 DO
      COB EPS:=4,--6,--8,--10 DO
      BEGIN
        I:=EPS--4 THEN NLCR; TIME:=TIME;
        FIRST:=I:=I; CJ:=CF:=0;
        X:=0; Y(0,1):=Y(0,2):=1; YMAX(1):=YMAX(2):=2;
        COB XEND:=0.005,50 DO
        NLCR; I:=XEND>1 THEN SPACE(20) ELSE COB R:=HMIN,EPS DO
        BEGIN FLOT(1,1,R); SPACE(3) END;
        NUWY+EPS(X,XEND,Y,HMIN,(XEND-X)/20,YMAX,
        EPS,FIRST,D,XY,1,JAC,J,Z,I:=I,E:=E);
        FLOT(1,1,X); COB R:=Y(0,1),Y(0,2) DO BEGIN SPACE(3); FIXT(1,12,R) END; SPACE(3);
        COB R:=CF/2,CJ/4,D(1,0),D(2,0),D(4,0) DO ABSFIXT(5,0,R) ABSFIXT(5,2,TIME-TIM);SPACE(3);
        I:=D(2,0)+0 THEN FLOT(5,2,D(3,0));
        I:=D(1,0)+0 THEN GO TO AA;
      END;
      AA; NLCR
      END
      END
      END

```

HMIN	EPS	X	Y1	Y2	FUS	JACS	D(0,1)	D(0,2)	ORDER	TIME	MAX,LOC,ERROR
+.1m=3	+.1m=3	+.5m=2	+.999952046592	+.1.000044239492	23	1	0	0	1	.75	
		+.5m=2	+.597692985894	+.1.402305120568	60	7	0	0	3	2.09	
+.1m=3	+.1m=5	+.5m=2	+.999952046573	+.1.000044239445	30	1	0	0	1	.86	
		+.5m=2	+.597653770477	+.1.402344336140	99	8	0	0	4	3.12	
+.1m=3	+.1m=7	+.5m=2	+.999951584187	+.1.000044701768	50	5	0	7	1	1.45	+.64822m= 7
		+.5m=2	+.597654039217	+.1.402344067399	158	15	0	7	9	5.07	+.64822m= 7
+.1m=3	+.1m=9	+.5m=2	+.999952046538	+.1.000044239408	79	1	0	22	1	1.80	+.64822m= 7
		+.5m=2	+.597654368058	+.1.402343738555	364	38	0	22	4	11.53	+.64822m= 7
+.1m=4	+.1m=3	+.5m=2	+.999952037725	+.1.000042591788	27	3	0	0	1	.88	
		+.5m=2	+.597704340240	+.1.402293766167	98	12	0	0	2	2.86	
+.1m=4	+.1m=5	+.5m=2	+.999952510829	+.1.000043775126	29	2	0	0	1	.87	
		+.5m=2	+.597656376877	+.1.402341729730	89	8	0	0	9	2.98	
+.1m=4	+.1m=7	+.5m=2	+.999952510806	+.1.000043775133	62	9	0	0	1	2.00	
		+.5m=2	+.597654698834	+.1.402343407779	160	17	0	0	5	5.33	
+.1m=4	+.1m=9	+.5m=2	+.999952510802	+.1.000043775137	110	8	0	2	6	4.18	+.84550m= 9
		+.5m=2	+.597654698547	+.1.402343408066	443	46	0	2	5	15.73	+.84550m= 9
+.1m=5	+.1m=3	+.5m=2	+.999952500846	+.1.000043750095	25	3	0	0	1	.86	
		+.5m=2	+.597871631695	+.1.402126474080	77	11	0	0	2	2.95	
+.1m=5	+.1m=5	+.5m=2	+.999952510829	+.1.000043775124	29	2	0	0	1	.87	
		+.5m=2	+.597656417520	+.1.402341689087	84	8	0	0	9	2.97	
+.1m=5	+.1m=7	+.5m=2	+.999952510802	+.1.000043775141	61	9	0	0	1	2.00	
		+.5m=2	+.597654692590	+.1.402343414025	176	18	0	0	9	5.74	
+.1m=5	+.1m=9	+.5m=2	+.999952510793	+.1.000043775141	112	8	0	0	4	4.19	
		+.5m=2	+.597654699774	+.1.402343406840	472	54	0	0	4	15.87	
+.1m=6	+.1m=3	+.5m=2	+.999952500102	+.1.000043748221	25	3	0	0	1	.85	
		+.5m=2	+.597870070746	+.1.402128035035	77	11	0	0	2	2.49	
+.1m=6	+.1m=5	+.5m=2	+.999952510831	+.1.000043775124	29	2	0	0	1	.87	
		+.5m=2	+.597656366770	+.1.402341739837	85	8	0	0	9	3.43	
+.1m=6	+.1m=7	+.5m=2	+.999952510813	+.1.000043775137	64	7	0	0	1	1.95	
		+.5m=2	+.597654714180	+.1.402343392434	173	17	0	0	9	5.98	
+.1m=6	+.1m=9	+.5m=2	+.999952510795	+.1.000043775148	116	8	0	0	3	4.17	
		+.5m=2	+.597654699056	+.1.402343407557	404	43	0	0	5	13.63	

1.9. Opgaven

1. Zij gegeven het beginwaardeprobleem

$$\begin{aligned}\dot{s} &= -(1-c)s + qc, \\ \dot{c} &= M((1-c)s - pc), \\ s(0) &= 1, \quad c(0) = 0, \\ M &= 1000; \quad q = 0.99; \quad p = 1.00.\end{aligned}$$

Bereken met een standaard Runge-Kutta procedure $s(t)$ en $c(t)$ voor $t = 1, 2, 3, 4, 5$.

Bereken met behulp van de procedure MULTISTEP $s(t)$ en $c(t)$ voor $t = 1, 2, 3, 4, 5, 10, 15, 20, 25, 50$.

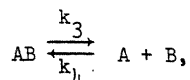
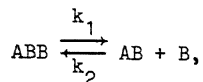
Voer de berekening een aantal malen uit met verschillende nauwkeurigheid.

Hoe nauwkeurig is het verkregen resultaat?

Ga telkens na hoeveel functie-evaluaties en hoeveel evaluatie van de Jacobiaan nodig zijn.

Is hier sprake van een stijve differentiaalvergelijking?

2. Twee gekoppelde chemische reacties worden beschreven door de chemische vergelijkingen



met $k_1 = 0.795$, $k_2 = 0.845$, $k_3 = 0.893$, $k_4 = 0.940$.

Het verloop van de concentraties $[ABB] = y$ en $[AB] = z$ wordt beschreven door het stelsel differentiaalvergelijkingen

$$dy/dt = -k_1 y + k_2 z (b - z - 2y),$$

$$dz/dt = -k_3 z + k_4 (b - z - 2y)(a - z - y) - dy/dt$$

Gegeven is:

$$a = 1; \quad b = 2; \quad y(0) = 0.25; \quad z(0) = 0.5.$$

Bereken $y(t)$ en $z(t)$ voor $t = 0.333, 0.672, 1.012, 100$

Gebruik de procedure MULTISTEP met

$$hmin = 0.002, 0.005, 0.01, 0.02$$

$$eps = 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}$$

$$stiff := \underline{false}.$$

Wordt bij de integratie een Adams-Moulton of een Curtiss-Hirschfelder gebruikt?

Welke uitspraak kan men doen omtrent de nauwkeurigheid van de resultaten?

3. Zij gegeven het beginwaardeprobleem

$$\dot{x} = -0.04x + 10^4 yz,$$

$$\dot{y} = 0.04x + 10^4 yz - 3 \cdot 10^7 y^2,$$

$$\dot{z} = 3 \cdot 10^7 y^2.$$

$$x(0) = z(0) = 0; \quad y(0) = 1.$$

Laat zien dat dit probleem equivalent is met

$$\dot{u} = 0.3v^2,$$

$$\dot{v} = 400(1-u-0.0001v) - 10000v(u+0.3v),$$

$$u(0) = v(0) = 0.$$

Is hier sprake van een stijve differentiaalvergelijking?

Bereken $u(0.4)$, $v(0.4)$, $u(10)$, $v(10)$ en ga na hoeveel functie-evaluaties en hoeveel evaluaties van de Jacobiaan nodig zijn.

1.10. Literatuur

- BASHFORTH, F. & ADAMS, J.C. *Theories of capillary action*, Cambridge Univ. Press. 1883
- CURTISS, C.F. & HIRSCHFELDER J.O., *Integration of stiff equations*, Proc. Nat. Acad. Sci. U.S., 38 235. -1952
- DEKKER, T.J., HEMKER, P.W. & HOUWEN, P.J. VAN DER, *Colloquium Stijve Differentiaalvergelijkingen*, MC Syllabus 15.1, Mathematisch Centrum, Amsterdam. 1972
- GEAR, C.W., *The automatic integration of stiff ordinary differential equations*, Proc. IFIP Congr. 1968, pp. 187
- GEAR, C.W., *Numerical initial value problems in ordinary differential equations*, Prentice-Hall Inc., Englewood Cliffs, N.J. 1971
- HEMKER, P.W., *An ALGOL 60 procedure for the solution of stiff differential equations*, Report MR 128/71, Mathematisch Centrum, Amsterdam, 1971
- HENRICI, P., *Discrete variable methods in ordinary differential equations*, John Wiley and Sons, New York. 1962
- MOULTON, F.R., *New methods in exterior ballistics*, Univ. Chicago Press 1926
- NORDSIECK, A., *On numerical integration of ordinary differential equations*, Math. Comp., 16 22-1962
- ZONNEVELD, J.A., *Automatic numerical integration*, Math. Centre Tracts 8, Mathematisch Centrum, Amsterdam 1964

OPMERKING

Een onlangs (1973) verbeterde en gedocumenteerde versie van de procedure MULTISTEP is te vinden in:

NUMAL, a library of numerical procedures in ALGOL 60 (C. den Heijer, P.W. Hemker, P.J. van der Houwen, N. Temme, D.T. Winter eds.)
Mathematisch Centrum, Amsterdam.