Universiteit Utrecht

Department
of Mathematics

Numerieke Wiskunde

Deel 2: praktische Numerieke Wiskunde

door

G. Sleijpen en P. Zegeling

24 januari 2006

Voorwoord

Wat is Numerieke Wiskunde?
De mathematische problemen die men in de fysische en technische praktijk tegenkomt [1] zijn gewoonlijk niet met analytische middelen oplosbaar. Van bijvoorbeeld de functie e^{-t^2} kan de primitieve niet gegeven worden in termen van bekende functies als e-machten, logaritmen, sinussen, ... Toch is het voor statistische toepassingen van belang numerieke waarden te hebben voor $\int_{-\infty}^{x} e^{-t^2}\,dt$. In dit soort gevallen is men aangewezen op numerieke berekeningen. Numeriek wiskundigen ontwerpen en analyseren algoritmen, waarmee met behulp van de computer, op een *efficiënte* manier de voornoemde problemen *nauwkeurig* en *betrouwbaar* opgelost kunnen worden. Numerieke uitkomsten zijn altijd met *fouten* belast (rekenfouten en benaderingsfouten). Met een nauwkeurige uitkomst bedoelen we een uitkomst waarvan de fout kleiner is dan een opgegeven waarde. Uiteraard zijn de fouten zelden exact te bepalen (dan zouden de exacte resultaten bekend zijn), maar kunnen op zijn best geschat worden. Die schattingen moeten dan wel gegeven kunnen worden en betrouwbaar zijn. De problemen zijn vaak zeer rekenintensief en om een algoritme te ontwerpen waarmee een nauwkeurig betrouwbare resultaten efficiënt, d.w.z. binnen redelijke (korte) tijd verkregen kunnen worden moeten de nodige wiskundige inspanningen geleverd worden.

Waarom numerieke wiskunde?
Voor diverse mathematische problemen zijn er standaard routines [2] en programmapakketten te krijgen [3]. De belangrijkste zijn ook nog ingebouwd in een gebruikersvriendelijke omgeving [4]. Helaas is er niet één algoritme, of één routine die bijvoorbeeld alle differentiaalvergelijkingen betrouwbaar kan oplossen. De ene routine is geschikt voor het ene type differentiaalvergelijking, terwijl de andere beter werkt voor een ander type. In iedere routine moet uiteraard het mathematisch probleem gespecificeerd worden maar ook nog extra parameters. Kortom je moet verstand hebben van de theorie achter het mathematisch probleem, maar ook van de numerieke theorie. De situatie doet denken aan autorijden: een auto is een handig vervoermiddel. Een automaat (denk aan MATLAB) maakt autorijden gemakkelijker, maar ook voor zo'n auto moet het rijden geleerd worden. Als universitair geschoolde wil je ook van de goedgebaande wegen af. Dan is het prettig als je ook verstand hebt van het mechaniek.

Inhoud van dit deel van de cursus.
In dit deel van de cursus maken we kennis met de fundamentele begrippen en een aantal fundamentele gedachten uit de Numerieke Wiskunde. De begrippen en gedachten worden telkens geformuleerd in een praktische context. In het eerste hoofdstuk betreft dat die van gewone differentiaalvergelijkingen (DV's): we gebruiken de problemen die optreden bij het numeriek oplossen van DV's als motivatie en als inspiratiebron. In tweede hoofdstuk bepalen we nulpunten van niet-lineaire vergelijkingen. Vaak leidt dit tot resultaten en inzichten die ook nuttig zijn buiten de context van DV's en nulpuntberekening. Resultaten en inzichten worden telkens getoetst aan de praktijk.

Een elementaire calculus cursus geeft voldoende voorkennis: Taylor reeksen van functies van één veranderlijke zijn de meest geavanceerde objecten die we zullen gebruiken.

Structuur van dit deel van de cursus.
Ieder hoofstuk sluit af met praktische toepassingen waarin alle inzichten uit het hoofdstuk gecombineerd worden.

Doel van dit deel van de cursus:
(1) het aanleren van de numerieke basisbegrippen (fouten, foutvoortplanting, gestructureerde fout, convergentie, nauwkeurigheid, stabiliteit, efficiëntie);
(2) het aanleren van basistechnieken (Euler methoden, Newtons methode, ...); vertrouwd maken met de wiskundige aanpak die tot die technieken leidt; bijbrengen van voldoende vaardigheid in het uitvoeren van kleine aanpassingen aan standaard technieken om eigen problemen die niet geheel standaard zijn toch ook te kunnen oplossen;
(3) het geven van zelfvertrouwen in het omgaan met standaardpakketten: weten dat je ze kan gebruiken, maar ook dat ze beperkingen hebben; het bijbrengen van een kritische houding t.o.v. numerieke resultaten; handigheid bijbrengen in het opsporen van de oorzaak van eventuele numerieke problemen; leren resultaten naar waarde kunnen schatten; laten inzien dat het maken van een 'fool-proof' code erg moeilijk is.

Om dit doel te bereiken vragen we van de cursisten een aktievere inzet dan gebruikelijk is bij andere cursussen. De docent speelt in feite alleen een begeleidende rol. Al het werk wordt door de cursist zelf gedaan.

Werkwijze:
Het gebruik van de computer speelt een dominante rol in deze cursus. De cursist brengt tijdens de "contact uren" de hele tijd door achter de computer (geen hoorcollege, geen werkcollege). Numerieke resultaten die door de cursist zelf met behulp van de computer bepaald worden roepen vragen op en nodigen uit tot theoretische beschouwingen. Inzichten worden onmiddellijk aan de praktijk getoetst. De resultaten roepen weer nieuwe vragen op, etc.. Het formaliseren van de inzichten (omzetten in correcte wiskunde) gebeurt thuis.

In het diktaat gebruiken we het volgende sjabloon:

Vraag. Hierin formuleren we een vraag die we willen beantwoorden. Idealiter is de "vraag" slechts een expliciete verwoording van wat de cursist zichzelf al afvroeg naar aanleiding van resultaten in eerdere experimenten: de vragen vormen de rode draad door de cursus. Vragen zijn genummerd.

Experiment. Hierin geven we suggesties om al experimenterend inzicht te krijgen in het probleem dat in **Vraag** geopperd is. De experimenten zullen ook nieuwe vragen oproepen. De experimenten worden uitgevoerd met MATLAB. Een groot deel van het benodigd programmatuurwerk is al uitgevoerd (dit is geen cursus programmeren). De routine die nodig is voor vraag X is te vinden in MATLAB m-file `X.m` (waarbij X het nummer van de vraag). In de heading van de m-file staan ("uitgecommentarieerd", d.w.z. achter %) de aanwijzigingen die betrekking hebben op details van het experiment (geschikte parameter keuze, te volgen MATLAB instructies, etc.).

Verklaring. In de onderdelen "**Verklaring**" staan wat suggesties/vragen die, naast de experimenteel verkregen resultaten, zouden kunnen helpen om te begrijpen wat de oplossing is van het in **Vraag** geopperde probleem. Bij deze onderdelen is het niet de bedoeling om formele bewijzen te leveren: het gaat hier puur om inzicht.

Opgave. *Van een aantal inzichten is het nuttig om ze te formaliseren. Aanwijzingen om dat te doen staan in dit onderdeel. De bedoeling is om dit soort werk thuis te doen.*

De vragen zijn genummerd. De 'experimenten', 'verklaringen' en 'opgaven' die bij een vraag horen zijn dat niet: ze hebben impliciet hetzelfde nummer als de vraag. Opgaven die onafhankelijk zijn van de vragen zijn wel genummerd.

Aan het eind van ieder hoofdstuk staan een aantal "**opdrachten**", gevolg door een

samenvatting van de belangrijkste resulaten. De stof van het hoofdstuk is van toepassing op de opdrachten. De opdrachten betreffen realistische en praktische problemen.

De cursus wordt afgesloten met een verslag over een deel van de opdrachten.
In het verslag moeten onder meer de vragen in de opdrachten verwerkt zijn. Het verslag is een soort artikel. Verwerk de vragen dus niet op een 'tentamenachtige' manier. Het verslag moet leesbaar zijn voor iemand die de stof bestudeerd heeft maar niet alles meer paraat heeft en die niet beschikt over de bronnen (diktaat en codes) van de schijver. Er mag wel verwezen worden. Referenties zijn zelfs gewenst (om wetenschappelijke redenen van verifiëerbaarheid, om 'credit' te geven aan je bronnen, om de lezer de mogelijkheid te bieden zich meer in details te verdiepen), maar het verslag moet ook begrijpelijk zijn zonder de referenties te raadplegen.

Referenties

[1] F. Beukers, Modellen en Computers, College Diktaat, Mathematisch Instituut, Universiteit Utrecht, Utrecht, 1994.

[2] W.H. Press, S.A. Teukolsky, W.T. Vettering, and B.P. Flannery, Numerical Recipes in C: The Art of Scientific Computing, Cambridge University Press, Cambridge, 2nd. ed., 1992.

[3] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, LAPackUsers' Guide – Release 2.0, SIAM,Philadelpha, 1995. `http://www.netlib.org/lapack`

[4] The Student Edition of MATLAB, version 4, User's Guide, Prentice-Hall, 1995

Dankwoord

De ontwikkeling van het materiaal in dit deel van de cursus heeft plaats gevonden binnen het K&S-project "Introductie IT in het wiskunde onderwijs".

Inhoudsopgave

Fouten

In alle numerieke berekeningen worden fouten gemaakt.
Als een grootheid berekend wordt uit een mathematische benadering spelen **benaderingsfouten**, ook wel *approximatie fouten* genoemd, een rol. Als we bijvoorbeeld $f'(a)$ berekenen volgens $\widetilde{f'(a)} := \frac{1}{h}(f(a+h) - f(a))$, met zeg $h = 0.0001$, dan is $f'(a) - \widetilde{f'(a)}$ de benaderingsfout.
Hoewel de computer met een behoorlijk aantal decimalen werkt is het aantal decimalen eindig en hebben we gewoonlijk ook nog te maken met **afrondfouten**, ook wel *evaluatie fouten* genoemd: $-0.1\ldots\ 10^{-12}$ is de afrondfout in de benadering $3.141\,592\,653\,589\,8$ van π.
Het numeriek resultaat is een benadering van de exacte waarde en is gewoonlijk belast met zowel benaderingsfouten als afrondfouten. Als we van een benaderende grootheid geen betrouwbare schatting kunnen geven voor (een bovengrens van) de fout dan heeft de benadering geen enkele betekenis (immers ook 7 benadert π). Foutschattingen en een analyse van de betrouwbaarheid van de foutschatting zijn daarom van cruciaal belang.

In de praktijk zal de fout niet exact bekend zijn. Immers als de fout bekend is, dan kan de exacte grootheid gemakkelijk bepaald worden. Het begrip "fout" speelt echter een grote rol in theoretische beschouwingen, bovendien kan de fout nogal eens redelijk betrouwbaar geschat worden.

Als we een benaderende waarde $\widetilde{a}$ voor een grootheid a hebben verstaan we onder
de fout: $a - \widetilde{a}$
de absolute fout: $|a - \widetilde{a}|$
de relatieve fout: $|(a - \widetilde{a})/a|$
de onbetrouwbaarheid: een bovengrens voor de absolute fout
de relatieve onbetrouwbaarheid: een bovengrens voor de relatieve fout.
Als de fout of de onbetrouwbaarheid in %'en wordt uitgedrukt wordt automatisch de relatieve fout of onbetrouwbaarheid bedoeld. In plaats van "fout" gebruiken we vaak "onnauwkeurigheid". Het woord "absoluut" wordt nogal eens weggelaten als er geen misverstand dreigt.

Met bijvoorbeeld $3.141\,6 \pm 0.002\,3$ voor π bedoelen we dat $3.141\,6$ een benaderende waarde is van π met een onbetrouwbaarheid van $0.002\,3$. Verder schrijven we bijvoorbeeld $2.008\,6\,10^1$ voor e^3 als we $2.008\,6\,10^1 \pm 0.000\,05\,10^1$ bedoelen: zondere verdere specificatie van de onbetrouwbaarheid is de onbetrouwbaarheid de helft van het laatst gegeven decimaal.

Experiment. Welke conventies hanteert MATLAB? (raadpleeg `help format` om de representatie van getallen in MATLAB te veranderen.)

Als bijvoorbeeld de benadering $2.712\,8$ van e een onbetrouwbaarheid $0.000\,05$ heeft dan zeggen we ook wel dat "de benadering correct is in 5 decimalen". Deze uitdrukking moet niet altijd naar de letter genomen worden, immers $1.000\,0$ is een benadering van $0.999\,999\,999$ met een onbetrouwbaarheid van $0.000\,05$, terwijl in feite geen enkel decimaal correct is. Met "correct in 5 decimalen" bedoelen we dus "correct afgerond op 5 decimalen".

Als je resultaten van een berekening presenteert aan andere mensen moet je natuurlijk de betrouwbaarheid van de resultaten vermelden, maar voorkom dat je de mensen

vermoeit met teveel decimalen: rond resultaten af op een relevant aantal decimalen, geef de onbetrouwbaarheid in maar 1 of 2 (interessante) decimalen.

Experiment. Van de functie $f(x) = \tan(x)$ willen we de waarde in een punt a weten. We beschikken over de benadering $\widetilde{a}$ die belast is met een onbetrouwbaarheid van 1%. Wat is de onbetrouwbaarheid in $f(\widetilde{a})$ voor het geval $\widetilde{a} = 1.4$ en voor het geval $\widetilde{a} = 3.0$? (Je kan dit uitzoeken met MATLAB, maar ook met een pocket calculator).

Verklaring. De relatieve onbetrouwbaarheden in $f(a)$ zijn (veel) groter dan in a voor deze waarden van a. Kan je dat verklaren?

Opgave. *Schrijf $\widetilde{a} = a + h$ en gebruik een eerste orde Taylor reeks om een uitdrukking te krijgen voor de relatieve fout in $f(\widetilde{a})$ in termen van de relatieve fout in $\widetilde{a}$. Hoe pakt dit uit in geval $f = \tan$ en $\widetilde{a} = 1.4$, $\widetilde{a} = 3.0$ beide met een onbetrouwbaarheid van 1%?*

Met $a \approx b$ geven we aan dat de benadering in relatieve zin is: $|(a - b)/b| \ll 1$.
Als a en b functies zijn en b heeft geen of maar een paar nulpunten dan schrijven we ook $a \approx b$ als $|[a(t) - b(t)]/b(t)| \ll 1$ voor alle t behalve voor de t in kleine intervalletjes rond de nulpunten van b.

Hoofdstuk I

Numeriek oplossen van differentiaalvergelijkingen

door G. Sleijpen

SLEUTELBEGRIPPEN: *benaderingsfout, foutvoortplanting, gestructureerde fout, foutschatting, betrouwbaarheid foutschatting, efficiëntie, stabiliteit*

Inleiding

Problemen die je in de praktijk tegenkomt kunnen nogal eens gemodelleerd worden als een differentiaalvergelijking of een stelsel differentiaalvergelijkingen [1]. Gewoonlijk zijn die differentiaalvergelijkingen niet analytisch oplosbaar. Middels analytische technieken [1, Hoofdstuk 2] kan je deelaspecten analyseren die enig inzicht kunnen geven in het "kwalitatieve" gedrag van de oplossing: via bijvoorbeeld linearisatie kan je iets te weten komen over de oplossing in de buurt van evenwichten, soms zijn er asymptotische uitspraken mogelijk, etc.. Helaas zijn die analytische technieken niet altijd toepasbaar of geven ze een onvolledig beeld. Bovendien is men in de praktijk geïnteresseerd in "kwantitatieve" resultaten: al in een ontwerpfase van bijvoorbeeld een gebouw zal men zeker willen weten of de constructie stormen en aardbevingen zal doorstaan; een raket wordt pas gelanceerd als de baan in grote nauwkeurigheid berekend is, etc.. In dit soort gevallen is men aangewezen op numerieke berekeningen. De belangrijke vraag is dan of de numerieke resultaten betrouwbaar zijn en hoe je kan achterhalen hoe betrouwbaar ze zijn. Verder vraag je je af of er één numerieke methode is waarmee iedere differentiaalvergelijking nauwkeurig opgelost kan worden en als dat niet zo is waar het van afhangt welke methode voor welk type differentiaalvergelijking het meest geschikt is. Zijn er standaard pakketten? Hoe roep je ze aan? Werken ze altijd? Als dat niet het geval is, hoe pas je ze aan voor je eigen probleem?

1.1 Differentiaalvergelijkingen

Gewone differentiaalvergelijkingen (GDVs), het type differentiaalvergelijkingen dat we in deze cursus bekijken, zijn van de vorm

$$u'(t) = f(t, u(t)), \quad t \in [t_0, t_0 + T]. \tag{1.1.a}$$

Hierin is f een gegeven functie en u een onbekende functie. u moet bepaald worden. De functies u en f zijn reëelwaardig of vectorwaardig: u is een C^1-afbeelding van het

'integratie' interval $[t_0, t_0+T]$ naar $\mathbb{R}^d$, f is gedefinieerd op een relevant deel van $\mathbb{R}\times\mathbb{R}^d$ en is ook $\mathbb{R}^d$-waardig. De oplossing u hangt ook nog af van een *beginvoorwaarde*

$$u(t_0) = u_0, \tag{1.1.b}$$

met u_0 een bekende vector. GDV (1.1.a) met beginvoorwaarde (1.1.b) is een *beginwaarde probleem.*
In toepassingen staat t vaak voor de tijd, maar er zijn ook toepassingen waarbij t een andere interpretatie heeft. Voor het gemak noemen we in deze cursus t de tijd: t_0 is dan het aanvangstijdstip, T is de tijdsspanne waarover de oplossing van de GDV gevolgd wordt, $t_0 + T$ is de eindtijd. Voor het schrijfgemak nemen we verder $t_0 = 0$.

De GDVs die we hieronder beschrijven zullen we gebruiken om onze numerieke inzichten te testen. Andere voorbeelden zijn te vinden in diverse tekstboeken over GDVs (zie bijvoorbeeld [1]). De tekst hieronder bevat wat achtergrondinformatie bij de GDVs.[1]

Voorbeeld 1A. [`Linear`] We bekijken eerst een eenvoudige GDV:

$$\begin{cases} u'(t) = \lambda(t)u(t) + g(t), \quad t \in [0, 20] \\ u(0) = u_0, \end{cases} \tag{1.2}$$

hierin zijn u, g en λ reëelwaardige functies ($d = 1$), g en λ zijn gegeven functies gedefinieerd op $[0, 20]$, u_0 is een bekende scalair.

De GDV in (1.2) is eendimensionaal en *lineair*. Als λ constant is en g niet al te ingewikkeld is kan u analytisch bepaald worden. Het beginwaarde probleem (1.2) is niet direct een voorbeeld om de behoefte aan een numerieke methode te illustreren. Maar juist omdat de GDV zo eenvoudig is, is ze geschikt voor numerieke experimenten. Als een numerieke methode faalt voor deze eenvoudige GDV dan zal hij het zeker laten afweten voor gecompliceerdere. Verder kunnen we, als we de exacte oplossing hebben, controleren of onze uitspraken betreffende de fout in een numeriek resultaat correct zijn. In een experimenteel stadium is het wel prettig om de exacte oplossing ter beschikking te hebben. We kiezen g in onze experimenten daarom als volgt:

$$g(t) \equiv w'(t) - \lambda\, w(t), \qquad u_0 \equiv w(0) \quad \text{met} \quad w(t) \equiv \cos(t). \tag{1.2.c}$$

Dan is de oplossing van (1.2) precies gelijk aan w: $u = w$.
We kunnen overigens iedere GDV modificeren om iedere gewenste oplossing te forceren: met $g(t) \equiv w'(t) - f(t, w(t))$ is w de oplossing van $u'(t) = f(t, u(t)) + g(t)$, $u(0) = w(0)$.

In onze testen kiezen we λ constant (o.a. $\lambda = 0.3$, $\lambda = -1$ en $\lambda = -1000$) of voor een $\varepsilon > 0$, $\varepsilon \ll 1$,

$$\lambda(t) = \frac{-2}{\varepsilon + (\sin \frac{1}{4}t)^2}. \tag{1.2.d}$$

[1] Ieder voorbeeld heeft een naam die tussen rechte haakjes vermeld staat achter het nummer. Deze naam correspondeert met de naam van de MATLAB file waarin de GDV gedefinieerd is. Zo staat het voorbeeld "Linear" in de file `Linear.m`. Als je een indruk wilt hebben over de oplossing kan je met de MATLAB functie `odeExact` een plaatje proberen te maken van de exacte oplossing. Zo levert `odeExact('Linear')` een plaatje van de exacte oplossing van de GDV in voorbeeld 1A. Voor $d = 2$ en voor $d = 3$ krijg je een faseplaatje, voor andere dimensies worden alle coördinaten uitgezet tegen de tijd.

Voorbeeld 1B. Het lineaire eendimensionale beginwaarde probleem (1.2) is ook van het type (1.1) als λ en g complexwaardige functies zijn. De oplossing u is dan ook complexwaardig. Door alle grootheden te splitsen in reële en imaginaire delen kan de GDV geschreven worden als een lineaire tweedimensionale reële GDV. Voor de analyse is het overigens handiger om met de complexe grootheden te werken.

In het algemeen is het zo dat iedere 'complexe' d-dimensionale GDV geschreven kan worden als een reële $2d$-dimensionale GDV.

Voorbeeld 1C. [`Flame`] Het volgende eendimensionale beginwaarde probleem is een model voor het verloop van de temperatuur in een chemische reactie. κ en μ zijn constanten, $u(t)$ is de temperatuur op tijdstip t.

$$\begin{cases} u' = \mu\,(2-u)\,e^{\kappa(1-\frac{1}{u})} \quad \text{op} \quad [0,1] \\ u(0) = 1. \end{cases} \tag{1.3}$$

Deze GDV is autonome. In een *autonome GDV* hangt de functie f die het rechterlid definieert niet af van de tijd: de GDV hangt via het rechterlid alleen via $u(t)$ van de tijd af. In (1.3) is $f(t,u) = \mu\,(2-u)\,\exp(\kappa(1-1/u)$. Voor een autonome GDV is het begintijdstip niet essentieel: als $u = v$ en $u = w$ de oplossing zijn van dezelfde autonome GDV met dezelfde beginwaarde u_0, maar met verschillende begintijd, dan is v gelijk aan w op een vertraging na: als $v(t_0) = u_0$ en $w(t_0 - s) = u_0$ dan is $v(t) = w(t-s)$.

Voorbeeld 1D. [`Immuun`] Antigenen ("ziekmakers") worden in ons lichaam gebonden door antilichamen en vervolgens onschadelijk gemaakt. De GDV in dit voorbeeld modelleert de verhouding in concentraties tussen deze twee. $x(t)$ is de concentratie antigenen in het lichaam op tijdstip t, $y(t)$ is de concentratie antilichamen. Het verloop van de concentraties kan, onder zekere aanname, beschreven worden door

$$\begin{cases} x' = 2(1 + \kappa x - \kappa y)x \\ y' = (-1 + 3\kappa x - \kappa y - \vartheta\kappa x y)y \end{cases} . \tag{1.4}$$

Hierin zijn κ en ϑ nog twee nader te specificeren parameters.

De GDV (1.4) is tweedimensionaal en met

$$u = \begin{bmatrix} x \\ y \end{bmatrix}, \quad f(t,u) = \begin{bmatrix} 2(1+\kappa x - \kappa y)x \\ (-1 + 3\kappa x - \kappa y - \vartheta\kappa x y)y \end{bmatrix}$$

zien we dat deze GDV van het type is als in (1.1.a).

Voorbeeld 1E. [`vdPol`] De *van der Pol vergelijking* speelt een rol in de medische biologie bij het modelleren van de hartslag. De vergelijking bevat niet alleen eerste orde afgeleiden (u'), maar ook een hogere orde (u''). Met een eenvoudige truc kan de hogere orde afgeleide omgeschreven worden naar een afgeleide van de eerste orde. De GDV wordt hierdoor wel hoger dimensionaal.

Voor zekere reële constanten μ, β en ε en u_0, w_0 wordt de van der Pol vergelijking gegeven door

$$\begin{cases} \varepsilon u'' = \mu(u' + \beta) - u - (u' + \beta)^3 \quad \text{op} \quad [0,T] \\ u(0) = u_0, \quad u'(0) = w_0. \end{cases} \tag{1.5}$$

Met $v \equiv u' + \beta$ is (1.5) equivalent met

$$\begin{cases} u' = v - \beta \\ v' = \frac{1}{\varepsilon}(\mu v - u + v^3) \end{cases} \tag{1.6.a}$$

met beginvoorwaarde

$$\begin{cases} u(0) = u_0 \\ v(0) = w_0 + \beta. \end{cases} \tag{1.6.b}$$

Deze GDV is tweedimensionaal en weer van het type als in (1.1).

In de experimenten nemen we $\mu = 1$, $\beta = 0.57$ en $\varepsilon = 1/19$, $u_0 = 0$, $w_0 = -\beta$.

Voorbeeld 1F. [Emigration] De GDV $\gamma\, u' = (\alpha - u)\, u$ is een eenvoudig model om de bevolkingsgroei te modelleren (*logistische groei*). Hierin zij α en γ positieve constanten en stelt $u(t)$ de gemiddelde bevolkingsdichtheid van een bepaalde populatie dieren voor op tijdstip t. Hierbij is gemiddeld over een zeker (leef)gebied. Als een klein deel van de populatie wegtrekt (*emigratie*) dan leidt een verfijning van het model (onder zekere aanname), waarin met dit kleine effect rekening wordt gehouden, tot $\gamma\, u' = (\alpha - u)\, u + \varepsilon u''$, met $\varepsilon < 0$, $|\varepsilon| \ll 1$. Deze GDV kan, met $v = u'$ geschreven worden als

$$\begin{cases} u' = v \\ v' = \frac{1}{\varepsilon}[\gamma\, v - (\alpha - u)\, u]. \end{cases} \tag{1.7}$$

Voorbeeld 1G. [Body3] De volgende GDV modelleert de baan die een kunstmaan K beschrijft die zich vrij (geen raketaandrijving) beweegt in een stelsel van een planeet A (aarde) met een satelliet M (maan). De massa van M is $\mu = 0.012\,277\,471$ maal de massa van A. Zwaartekrachtinvloeden van andere hemellichamen (en van de kunstmaan) verwaarlozen we. Verder nemen we aan dat de kunstmaan zich alleen maar in het vlak van planeet en satelliet beweegt. We kiezen de oorsprong van dit '(x, y)-vlak' in het zwaartepunt van het stelsel. We leggen de x-as door het centrum van A en dus ook door het centrum van M. Merk op dat het (x, y)-assenstelsel met A en M meeroteert. Planeet A ligt op positie $(-\mu, 0)$ en saelliet M op $(\nu, 0) \equiv (1 - \mu, 0)$. Als op tijdstip t de kunstmaan positie $(x(t), y(t))$ heeft dan geldt

$$\begin{cases} x'' = x + 2y' - \nu\dfrac{x+\mu}{d_A} - \mu\dfrac{x-\nu}{d_M}, \\ y'' = y - 2x' - \nu\dfrac{y}{d_A} - \mu\dfrac{y}{d_M}, \end{cases} \tag{1.8.a}$$

waarbij $\quad d_A \equiv ((x+\mu)^2 + y^2)^{\frac{3}{2}} \quad$ en $\quad d_M \equiv ((x-\nu)^2 + y^2)^{\frac{3}{2}}$.

Als u de snelheid is van K in de x-richting, $u = x'$, v de snelheid in de y-richting, $v = y'$, dan leidt dit tot een vierdimensionale GDV.

Het is niet de bedoeling dat de kunstmaan in de kosmos verdwijnt of te pletter slaat op A of op M: we zijn dus geïnteresseerd in een gesloten baan, d.w.z. in een gesloten oplossingskromme $t \to (x(t), y(t))$. Zo'n baan wordt een *Arenstorf baan* genoemd. De beginwaarden

$$\begin{cases} x(0) = 0.994, & x'(0) = 0, \\ y(0) = 0, & y'(0) = -2.001\,585\,106\,379\,082\,522\,405\,378\,622\,24\ldots, \end{cases} \tag{1.8.b}$$

passen bij een Arenstorf baan met periode $T = 17.065\,216\,560\,157\,962\,558\,891\,720\,624\,9$. Het berekenen van een Arenstorf baan vergt hoge precisie en de startwaarden moeten in een groot aantal cijfers bekend zijn.

1.2 Kennismaking met numerieke oplosmethoden

SLEUTELBEGRIPPEN: *Betrouwbaarheid numeriek antwoord, efficiëntie*

In deze paragraaf proberen we te achterhalen of we blindelings mogen vertrouwen op de oplossingen die de computer ons levert.

Vraag 2A. Kunnen computers GDVs eigenlijk wel oplossen? Differentiëren is een limietoperatie. Computers kunnen alleen met getallen uit een eindige collectie werken. Krijg je wel de oplossing te zien?

Experiment. Los (1.2) (`Linear`) met λ als in (1.2.d) en $\varepsilon = 0.02$ op in MATLAB met de routine '`odeEF`'.[2] Plot ook de exacte oplossing.[3] Bekijk de figuren die MATLAB produceert. Is het numeriek antwoord in alle tijdstippen berekend? (Zoom in als je meer details wilt zien). Valt het numerieke antwoord samen met het exacte antwoord in relevante tijdstippen? Welke zijn de relevante tijdstippen?

Verklaring. De afgeleide $u'(t)$ is benaderd door $\frac{1}{h}[u(t+h) - u(t)]$ of een ander differentie-quotiënt. Verder is er alleen gerekend in een beperkte set van tijdstippen t_n (is $t_n = nh$ voor $n = 0, 1, 2, \ldots$?). Door te benaderen introduceer je fouten.

Vraag 2B. Wat gebeurt er als je in meer tijdstippen rekent, d.w.z. als je de afstand h tussen de rekenpunten verandert?

Experiment. Verklein h eens met, zeg, een factor 3.[4] Wat gebeurt er met de fout? Verklein h eens met een factor 10 (bereken nu niet de exacte oplossing).[5] Wat gebeurt er met de rekentijd? Wat gebeurt er als je h vergroot?

Verklaring. Met een grotere h zijn er minder rekenpunten en dus zal het rekenwerk ook wel minder zijn. Echter voor grotere h is de benadering 'grover', d.w.z. de benaderingsfouten zijn groter. h wordt de **stapgrootte** genoemd.

Vraag 2C. Zijn de antwoorden altijd betrouwbaar? Of hangt de betrouwbaarheid samen met de stapgrootte? Wordt de fout kleiner als je h verkleint? Kan je aan de resultaten zien of ze betrouwbaar zijn?

Experiment. We experimenteren met twee routines. De eerste `odeEF` is door onszelf gecodeerd en berust op een eenvoudige methode die we in de volgende paragrafen uitgebreid zullen bestuderen. De tweede routine `ode23` is zeer geavanceerd en behoort tot de ODE Toolbox van MATLAB.

[2] 'Run' hiertoe de m-file `Vraag2.m`. Kijk wel eerst of het goede beginwaarde probleem gekozen is: `ODE='Linear2'` en de goede oplosmethode: `method='odeEF'`. Check ook eerst of in de function file `Linear2.m` de GDV en de beginvoorwaarden goed gedefinieerd zijn. Een uitgebreidere handleiding bij de MATLAB files staat in de file `READ.ME`.

[3] D.w.z, neem, in de file `Vraag2.m`, `PlotExactSolution=1` .

[4] h heet `dt` in `Vraag2.m` en in de andere '`Vraag`-files'.

[5] Neem `PlotExactSolution=0`.

Los (1.5) (`vdPol`) eens op met $h = 0.03$ en `odeEF` en plot een faseplaatje van de numerieke oplossing.[6] (Laat ook eens de tussenresultaten plotten[7]). Vertrouw je het antwoord? Probeer ook eens $h = 0.01$. Zou je h nog kleiner moeten nemen? Los het beginwaarde probleem ook eens op met `ode23`. Welke oplossing is de nauwkeurigste denk je? Hoe kan je daar enige zekerheid over krijgen?

Met `odeEF` kan je een hogere nauwkeurigheid krijgen door h te verkleinen. De professionele code `ode23` kiest zelf een "optimale" stapgrootte (en past die aan tijdens het rekenproces). Je kan dus zelf h niet aan passen. Je kan wel opgeven hoe nauwkeurig je je antwoord wil middels de parameter `RelTol`. De suggestie is dat `ode23` het probleem oplost met een relatieve onbetrouwbaarheid `RelTol`. De default waarde is `RelTol=1.e-3`. Is de relatieve onbetrouwbaarheid inderdaad 0.001? Neem ook eens een hogere "relatieve tolerantie" (d.w.z. met kleinere `RelTol`).[8]

Verklaring. Gaat de rest van het eerste hoofdstuk over.

> **Conclusie.** *Je kan er niet blindelings op vertrouwen dat de computer correcte antwoorden produceert.*

Bovenstaande experimenten roepen een aantal vragen op.
Hoe weet je of de h voldoende klein is, of daaraan gerelateerd, wanneer kan je het numeriek antwoord vertrouwen? Hoe betrouwbaar is dat antwoord precies? Hoe kan je een betrouwbaar antwoord te krijgen en veel rekenwerk vermijden? Hoe hangt dit samen met de numerieke methode en met de GDV? Deze vragen staan centraal in dit hoofdstuk.

1.3 Convergentie

SLEUTELBEGRIPPEN: *lokale fout, cumulatie lokale fouten, globale fout, convergentie, gestructureerde globale fout, foutschatten, betrouwbaarheid foutschatting*

In programmapakketten worden geavanceerde methoden gebruikt. Om te begrijpen wat de problemen zijn waarmee je bij het numeriek oplossen van GDVs te maken krijgt, bekijken we eerst een eenvoudige methode, die overigens voor talloze GDVs nog steeds een van de efficiëntste is.

Een computer kan niet differentiëren. Omdat voor kleine $h > 0$

$$u'(t) \approx \frac{u(t+h) - u(t)}{h}, \tag{3.1}$$

kunnen we, als we de waarde van u in $t + h$ en in t nauwkeurig kennen, $u'(t)$ toch nauwkeurig berekenen. Helaas kennen we die functiewaarden niet. Maar we hebben ook nog de GDV en we kunnen kijken waar een combinatie van (1.1.a) met (3.1) toe leidt. Voor een $h > 0$ die voldoende klein is geldt

$$u(t+h) \approx u(t) + h\,u'(t) = u(t) + h\,f(t, u(t)).$$

Als we in een tijdstip t_n een nauwkeurige benadering, zeg u_n, van $u(t_n)$ tot onze beschikking hebben dan kunnen we u in het tijdstip $t_n + h$ nauwkeurig benaderen door

[6] Neem `J=[1,2]`.

[7] `ShowSolutionProcess=1`

[8] Je kan al een indruk krijgen van een deel van de problemen door inspectie van het richtingsveld: neem `PlotFlowField=1`.

$u_n + hf(t_n, u_n)$: de grootheden u_n en t_n zijn bekend, en $f(t_n, u_n)$ kan uitgerekend worden. Er kan dus gemakkelijk een benadering $u_{n+1} \equiv u_n + hf(t_n, u_n)$ van $u(t_n + h)$ berekend worden. We kunnen vervolgens het spelletje herhalen: we kunnen nu op dezelfde manier in het tijdstip $t_n + 2\,h$ de oplossing schatten, etc.. Beginnen we met deze aanpak in het tijdstip $t_0 = 0$ dan weten we dat $u(0) = u_0$ (zie (1.1.b)) en komen we tot het volgend rekenproces:

$$u_{n+1} = u_n + hf(t_n, u_n), \qquad t_{n+1} = t_n + h, \quad n = 0, 1, 2, \ldots, \tag{3.2}$$

waarin u_n een benadering is voor $u(t_n)$. Dit rekenschema heet de **Euler forward** (EF) methode.

Vraag 3A. De oplossing van een beginwaarde probleem kan grafisch gerepresenteerd worden. Hoe kan je de EF methode grafisch interpreteren?

Experiment. Plot voor (1.2) (`Linear`) met $\lambda = -1$ en g als in (1.2.c),[9] de grafiek van de exacte oplossing en plot, voor niet al te kleine h, de numerieke oplossing. Verbind de punten (t_n, u_n) met lijnstukjes.[10] Plot ook de exacte oplossing van de GDV (1.2.a) door (t_1, u_1), d.w.z. plot de grafiek van die u waarvoor $u' = \lambda\, u + g$ en $u(t_1) = u_1$. Doe dat ook voor de exacte oplossingen van de GDV door achtereenvolgens (t_2, u_2),..., (t_6, u_6).[11] Bekijk de grafische resultaten in de buurt van de eerste paar t_n (zoom in).

Verklaring. EF volgt telkens over korte stukjes (stukjes met, langs de t-as, lengte h) de raaklijn en stapt daarbij telkens over op een andere 'naburige' oplossing.

Vraag 3B. Convergeert EF? Wat betekent convergentie hier?

Experiment. Onderzoek door h te verkleinen voor de testvoorbeelden (1.2) (`Linear`) met $\lambda = -1$ en (1.4) (`Immuun`) met $\kappa = 1.3$ en $\vartheta = 0.06$ of EF convergeert en als er convergentie is, in welke zin dat gebeurt.[12]

Verklaring. Er geldt

$$u(t_{n+1}) = u(t_n) + h\, f(t_n, u(t_n)) + \tau_n \quad \text{waarbij} \quad \tau_n \equiv \tfrac{1}{2}\,(h)^2\, u''(\xi_n) \tag{3.3}$$

voor geschikte ξ_n tussen t_n en t_{n+1}.
EF kan je zien als een methode waarin in ieder tijdstip t_n het **defect** τ_n weggemoffeld wordt (vergelijk (3.3) en (3.2)). De defecten τ_n gaan naar 0 als $h \to 0$ en ze gaan sneller naar 0 dan de hf-term (waarom?). Het is duidelijk dat de defecten de nauwkeurigheid van het numeriek resultaat beïnvloeden, het is niet duidelijk hoe ze dat doen. Het is evenmin duidelijk of er niet nog andere factoren een rol spelen.

Bij convergentiebeschouwingen moet je je realiseren dat voor iedere t de n in de vergelijking $t_n = t$ afhangt van h.

Opgave. *Bewijs* (3.3). *Hint: gebruik een Taylor ontwikkeling.*

We onderzoeken eerst hoe de defecten de numerieke nauwkeurigheid beïnvloeden. Als h klein is, dan is

$$\tau_n \approx \tfrac{1}{2}\, h^2\, u''(t_n). \tag{3.4}$$

[9] Dit beginwaarde probleem is gedefinieerd in `Linear.m`.

[10] Gebeurt vanzelf met `Vraag3A.m`. De `Vraag*.m`-files zijn copiën van `main.m` met waarden voor parameters passend bij de corresponderende Vraag.

[11] Neem `PlotExactNearbySol=1`.

[12] Het tweede beginwaarde probleem is gedefinieerd in `Immuun.m`.

Als we voor kleine h de stapgrootte h halveren dan worden de defecten met ongeveer een factor 4 kleiner, immers $\frac{1}{2}(\frac{1}{2}h)^2u''(t) = \frac{1}{4}[\frac{1}{2}h^2u''(t)]$. Geldt dat ook voor de fout in het tijdstip t? Deze **fout** is gegeven door

$$e_n \equiv u(t_n) - u_n \quad \text{met } n \text{ zodat} \quad t_n = t. \tag{3.5}$$

Vraag 3C. Hoeveel kleiner wordt de fout als je h halveert?

Experiment. Plot de exacte oplossing. Los (1.2) (`Linear`) met $\lambda = -1$ op met EF met stapgrootte h, $h/2$, $h/4$, ... Wat gebeurt er met de fout?

Verklaring. Defecten cumuleren.
De defecten worden weliswaar een factor vier kleiner, maar het aantal stappen om in eenzelfde tijdstip te komen verdubbelt.
De fout in het tijdstip t (waarvoor t/h geheel is) lijkt evenredig met h.

Opgave. *Bekijk een tijdstip $t \in [0, T]$ waarvoor t/h geheel is. Vergelijk de fout in twee stappen EF vanuit het tijdstip t met stapgrootte $h/2$ met een stap EF ook vanuit het tijdstip t maar nu met stapgrootte h.*

Vraag 3D. Is de fout e_n een som van defecten in voorgaande tijdstippen of hebben nog andere factoren een invloed op de fout?

Experiment a. Los weer (1.2) (`Linear`) op, maar neem nu eens $\lambda = 0.3$. Kijk met name naar het verschil tussen de numerieke oplossing voor $\lambda = -1$ en $\lambda = 0.3$ in $t = 20$. Hoe kan je dat verschil verklaren? Plot ook weer eens de exacte oplossingen door (t_n, u_n) voor $n = 1, \ldots, 6$ en kijk naar de resultaten in $t = 20$.
Tussen haakjes: halveert hier de fout nog steeds als je h halveert?

Verklaring. De GDV zelf kan fouten vergroten.
Met EF maak je telkens foutjes (nl. de defecten) die de GDV kan vergroten.
Als u de oplossing is van (1.1.a-b) en $\widetilde{u}$ voldoet aan (1.1.a), maar nu met $\widetilde{u}(0) = u_0 + \varepsilon$ voor een $\varepsilon \neq 0$, dan geeft $e(t) \equiv u(t) - \widetilde{u}(t)$ aan hoe de GDV de verstoring ε van de beginwaarde voortplant.

$$e(t + h) \approx e^{h\,\lambda(t)}\, e(t). \tag{3.6}$$

Over een tijdstraject(je) ter lengte h vergroot of verkleint de GDV de fout, d.w.z. het effect van de verstoring, met ongeveer een factor $e^{h\,\lambda(t)}$.

Opgave. *Bekijk* (1.2) *met onverstoorde en verstoorde beginwaarde.*

a. *Bewijs dat $e'(t) = \lambda(t)e(t)$ voor alle $t \in [0, T]$ en $e(0) = -\varepsilon$.*
Stel nu dat $\lambda(t) = \lambda_0$ voor alle t (λ is constant)

b. *Bewijs dat $|e(t)| = |\varepsilon|\, e^{t\lambda_0}$, in geval $\lambda \in \mathbb{R}$. Spoort dit met je observaties in je experiment? In het geval dat λ_0 complex is en niet reëel (zie Voorbeeld 1B) geldt $|e(t)| = |\varepsilon|\, e^{tRe(\lambda_0)}$.*

c. *Bewijs dat $e(t + h) = e^{h\lambda_0}\, e(t)$ (zie* (3.6)*).*

Experiment b. Neem nu eens $\lambda = -125$ en $h \geq 0.02$. Is het resultaat ook zo beroerd voor kleinere h? Halveert de fout dan weer bij halverende h?
Vergroot de GDV fouten? Plot ook weer eens de exacte oplossingen door de eerste paar numerieke resultaten en zoom in $t = 0$?

Verklaring. EF kan fouten vergroten.

Voor EF toegepast op (1.2) geldt

$$e_{n+1} = (1 + h\,\lambda_n)\,e_n + \tau_n \quad \text{met} \quad \lambda_n \equiv \lambda(t_n). \tag{3.7}$$

Over een tijdstraject ter lengte h vergroot of verkleint EF de 'oude' fout met een factor $1 + h\,\lambda_n$. (Hoe groot was deze factor in de beroerde gevallen?) Verder komt er in dat traject nog een 'nieuwe' fout bij, een 'lokale' onnauwkeurigheid, nl. het defect.

Opgave. *Bekijk EF toegepast op* (1.2).

a. *Bewijs* (3.7).

b. *Als* $|h\,\lambda_n| \ll 1$ *(denk aan* $|h\,\lambda_n| \leq 0.2$*) dan geldt* $1 + h\,\lambda_n \approx e^{h\,\lambda_n}$*. Bewijs dit en vergelijk dit resultaat met het resultaat in onderdeel c. van de vorige opgave.*

Conclusie.
a. *Defecten (al of niet vergroot; zie* b. *en* c.*) cumuleren.*
b. *De GDV zelf kan fouten vergroten.*
c. *De numerieke methode kan fouten vergroten.*

Als $|h\,\lambda_n| \not\ll 1$ dan springt EF anders om met 'oude' fouten dan de GDV. Dit is gewoonlijk fataal als ook nog $|1 + h\,\lambda_n| > 1$. We zullen deze situatie uitgebreid bestuderen in §1.4. Voorlopig concentreren we ons op de situatie waarin h zo klein is dat EF de fouten op min of meer dezelfde manier voortplant als de GDV zelf. In dat geval is de conclusie in 3C correct.

Vraag 3E. Kan je het inzicht uit 3C gebruiken om de fout te schatten? Kan je dit inzicht gebruiken om de stapgrootte zo aan te passen dat je een fout krijgt die kleiner is dan bv. 10^{-3}?

Experiment. Los weer (1.2) (`Linear`) met $\lambda = -1$ op met EF voor $h = 0.2$ en $h = 0.1$ en schat de fout. Bepaal ook de exacte oplossing. Hoe goed is je schatting? Met welke waarde voor h verwacht je een fout te krijgen die in alle (relevante) tijdstippen kleiner is dan 10^{-3}. Kijk of je verwachting uitkomt.

Deze manier om de fout te schatten is *a posteriori*: de fout wordt achteraf (nadat er gerekend is) uit numerieke resultaten geschat. Met behulp van de formule voor τ_n is het ook mogelijk (maar overigens niet erg practisch) om de fout te schatten voordat er gerekend wordt (zie 3K). De schatting is dan *a priori*.

Vraag 3F. Geldt die evenredigheid van de fout met h ook voor grotere h (met h toch nog zo klein dat EF fouten niet beroerder voortplant dan de GDV zelf)? Geldt die evenredigheid in alle (relevante) tijdstippen?

Experiment. Los weer (1.2) (`Linear`) met $\lambda = -1$ op met EF maar nu met $h = 1$ en $h = 0.5$.

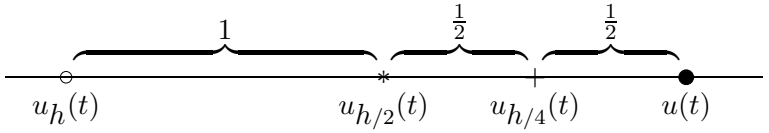
Verklaring. Als h niet klein is (of preciezer, als u'' veel varieert op $[t, t + h]$) dan is (3.4) niet correct: hogere orde Taylor reeks zijn dan niet verwaarloosbaar. Soms is de fout overigens "per ongeluk" ≈ 0.

Als in een numerieke oplosmethode van GDVs de fouten min of meer evenredig zijn met h^p voor alle h die klein genoeg zijn dan is de methode van **orde** p: EF is een methode van orde 1.

Vraag 3G. Als je de fout wilt schatten (en het exacte antwoord niet hebt), hoe weet je of h voldoende klein is?

Experiment. De EF oplossing die verkregen is met stapgrootte h noemen we u_h. Dus in $t_n = n\,h$ is $u_h(t_n) = u_n$. Wat kan je voor het tijdstip t (met t/h geheel) zeggen over de verhouding tussen $u_h(t) - u_{h/2}(t)$ en $u_{h/2}(t) - u_{h/4}(t)$? Hoe hangt die verhouding samen met h? Gebruik de resultaten uit de experimenten in 3C en 3F.

Verklaring. Als voor een zekere c geldt $u_h(t) - u_{h/2}(t) \approx c\,h$ dan kan je de resultaten als volgt weergeven op de getallenrechte. De getallen geven verhoudingen weer.



Vraag 3H. Zijn die inzichten ook correct voor ingewikkeldere GDVs?

Experiment a. Pas EF toe op het niet-lineaire beginwaardebprobleem (1.3) (`Flame`) met $\mu = \frac{1}{4}$ en $\kappa = 2$.[13] Hoe gedraagt de fout zich?

Verklaring. Als EF fouten niet beroerder voortplant dan de GDV zelf dan gedragen de fouten zich min of meer als de oplossing van een lineaire GDV en wel als de oplossing van

$$\begin{cases} e'(t) = \lambda(t)e(t) + g(t) \quad \text{met} \quad \lambda(t) \approx \frac{\partial f}{\partial u}(t, u(t)) \quad \text{en} \quad g \approx \frac{1}{2}h\,u''(t) \\ e(0) = 0. \end{cases} \tag{3.8}$$

Opgave. *Bekijk de eendimensionale versie van* (1.1).

a. *Bewijs dat, voor het geval dat $\varepsilon \equiv e(t)$ klein is, geldt*

$$f(t, u(t) + \varepsilon) = f(t, u(t)) + \lambda(t)\varepsilon, \quad \textit{waarbij} \quad \lambda(t) \approx \frac{\partial f}{\partial u}(t, \widetilde{u}(t))$$

en $\widetilde{u}(t)$ een geschikte waarde is tussen $u(t)$ en $u(t) + \varepsilon$.

b. *Geef de versie van* (3.6) *en van* (3.7) *voor de 1-d. versie van* (1.1).

Experiment b. Hoe zit het met meerdimensionale beginwaarde problemen? Pas EF eens toe op (1.4) (`Immuun`) met $\kappa = 1.3$ en $\vartheta = 0.06$ en $T = 5$. $h = 0.1$ lijkt een geschikte 'start' voor het halveringsproces. Kijk naar de oplossingen als functie van t maar ook naar het faseplaatje. Doe hetzelfde eens voor $T = 40$.

Verklaring. Bovenstaande beschouwingen kunnen coördinaatsgewijs gehouden worden.

Experiment c. Pas EF weer toe op (1.2) (`Linear`) met $\lambda = -500$.

Verklaring. Ook hier is het antwoord ja, maar h moet wel erg klein gekozen worden. De verklaring zit in de grootte van de factor $|1 + h\,\lambda_n|$ in (3.7). Je kan je hier afvragen hoe groot h mag/moet zijn. We bewaren deze vraag tot §1.4.

Conclusie.

a. *Is h voldoende klein, dan zijn de EF fouten min of meer evenredig met h.*

b. *Je mag er op vertrouwen dat h 'voldoende klein' is als bij herhaald halveren van h het geobserveerde foutgedrag past bij evenredigheid met h.*

[13] in `Flame.m`.

Zijn de inzichten die we voor EF opgedaan hebben ook van toepassing voor andere oplosmethoden? We bekijken ondermeer de Trapeziumregel. Deze methode ontstaat door in iedere t_n het defect τ_n^{trap} in de relatie

$$\begin{cases} u(t_{n+1}) = u(t_n) + \frac{1}{2}h\,[f(t_n, u(t_n)) + f(t_{n+1}, u(t_{n+1}))] + \tau_n^{\text{trap}} \\ \text{met} \quad \tau_n^{\text{trap}} = -\frac{1}{12}h^3\,u'''(\xi_n) \end{cases} \tag{3.9}$$

weg te poetsen.

Vraag 3I. Hoe verandert de fout in de Trapeziumregel als je h verandert? Zijn er methoden die nog nauwkeuriger zijn?

Experiment a. Pas de Trapeziumregel toe op (1.4) (`Immuun`) met $\kappa = 1.3$ en $\vartheta = 0.06$ en kijk of je verwachting uitkomt.[14]

Opgave. *Bewijs* (3.9) *(het is ook al voldoende als je kan bewijzen dat* $\tau_n^{\text{trap}} = -\frac{1}{12}h^3\,u'''(t_n) + \mathcal{O}(Dt^4)$*).*

Hint: bedenk dat $f(t, u(t)) = u'(t)$ *en ontwikkel de voorkomende functies rond* t_n*.*

Experiment b. Welke orde heeft RK4?[15][16] De orde is lastig af te lezen uit de plaatjes. Hoe komt dat?

Conclusie. *De Trapeziumregel is van orde 2,* RK4 *is van orde 4.*

Opgave 3J. *Bewijs dat* (3.9) *ontstaat door de Trapeziumregel voor numeriek integreren toe te passen op de integraal in de relatie* $u(t+h) = u(t) + \int_t^{t+h} u'(s)\,\mathrm{d}s$*.*

Opgave 3K. *Hieronder een schets van het bewijs dat de EF "convergeert". Details worden als opgave aan de lezer over gelaten.*

Neem $d = 1$*. Bewijs de volgende uitspraken.*

a. $f(t_n, u(t_n)) - f(t_n, u_n) = \frac{\partial f}{\partial u}(t_n, \widetilde{u}_n)\,e_n$ *voor zekere* $\widetilde{u}_n$ *tussen* $u(t_n)$ *en* u_n*.*

b. $e_0 = 0, \quad e_{n+1} = e_n + h\,\frac{\partial f}{\partial u}(t_n, \widetilde{u}_n)\,e_n + \tau_n = \left(1 + h\,\frac{\partial f}{\partial u}(t_n, \widetilde{u}_n)\right)e_n + \tau_n.$

c. *Als* $\left|\frac{\partial f}{\partial u}(t_n, \widetilde{u}_n)\right| \le M$ *voor alle* n *dan geldt*

$$|e_{n+1}| \le (1 + h\,M)|e_n| + |\tau_n| \le \textstyle\sum_{j=0}^{n}(1 + h\,M)^{n-j}|\tau_j|$$

d. *Omdat* $1 + h\,M \le e^{h\,M}$ *is* $(1 + h\,M)^i \le e^{ih\,M}$ *en geldt*

$$|e_n| \le e^{n\,h\,M}\textstyle\sum_{j<n}|\tau_j| \le e^{TM}\sum_{j<n}|\tau_j| \quad \text{voor alle} \quad t_n,\ t_n - t_0 = n\,h \le T.$$

e. $\sum_{j<n}|\tau_j| \le n\,\max_{j<n}|\tau_j| \le n\,h^2\,\frac{1}{2}\,\max_\xi|u''(\xi)| \le \left(T\,\frac{1}{2}\,\max|u''(\xi)|\right)h.$

f. *EF convergeert als de functies* $|u''|$ *en* $|\frac{\partial f}{\partial u}|$ *begrensd zijn.*

g. *Stel nu dat* $\frac{\partial f}{\partial u}$ *ook nog negatief is. Laat zien dat dan* $|1 + h\frac{\partial f}{\partial u}(t_n, \widetilde{u}_n)| \le 1$ *voor iedere* h*,* $0 < h \le 2/M$*. Gebruik dit om je schattingen in c. en d. aan te scherpen.*

We onderzoeken of we bovenstaande schattingen praktisch kunnen gebruiken.

h. *Bepaal een* h *zodat de fout in EF toegepast op* (1.2) (`Linear`) *met* $\lambda = -1$ *(en* g *als in* (1.2.c) *zo dat* $u(t) = \cos(t)$*) over het hele integratietraject* $[0, 20]$ *ten hoogste* 10^{-2} *is. Gebruik hierbij eerst de oorspronkelijke schattingen en daarna de aangescherpte.*

[14] De Trapeziumregel roep je aan met `odeIT`.

[15] `odeRK` maakt gebruik van RK4. Het heeft hier geen zin om de exacte oplossing te laten plotten omdat de "exacte" oplossing berekend wordt met `odeRK`: neem `PlotExactSolution=0`.

[16] RK4 is 'n Runge–Kutta methode. Maar er zijn er vele meer. De RK4 die wij hier bekijken wordt wel de 'klassieke' Runge–Kutta methode genoemd en is degene die voorgesteld is door Runge en Kutta.

1.4 Stabiliteit

SLEUTELBEGRIPPEN: *Stabiliteit, groeifactor fout, foutvoortplanting, stijve GDV, impliciet versus expliciet, efficiëntie*

De **stabiliteitsanalyse** in de Numerieke Wiskunde houdt zich bezig met de vraag hoe kleine fouten (zowel benaderingsfouten als afrondfouten), die gemaakt zijn in de afzonderlijke rekenstappen van een rekenproces, doorwerken in het eindantwoord.

> *Een rekenmethode is* **instabiel** *als kleine fouten in de afzonderlijke rekenstappen een grote invloed hebben op het eindantwoord, anders wordt de methode* **stabiel** *genoemd.*

In deze formulering zijn de begrippen 'instabiel' en 'stabiel' nogal subjectief: wat groot is en wat klein is hangt erg af van de toepassing. Een relatieve fout van bijvoorbeeld 0.1 % in de weersvoorspelling voor het weer van overmorgen ($300 \pm 0.3^\circ$ K) is klein terwijl een fout van 0.1% in de berekening van de baan van een raket naar Mars fataal is. Voor mathematische toepassingen wordt in de numerieke wiskunde het stabiliteitsbegrip dat in een bepaalde context gehanteerd wordt telkens geprecisiëerd. Het precieze begrip wordt dan versierd met bijwoorden als 'sterk stabiel', 'asymptotisch stabiel', etc.. Het heuristisch begrip speelt echter telkens op de achtergrond een rol. Merk op dat de heuristiek die aan de numerieke stabiliteitsbegrippen ten grondslag ligt precies dezelfde is als voor de stabiliteitsbegrippen in de mathematische analyse. Zo is een stabiel evenwicht van een GDV een evenwicht dat de oplossing van de GDV opzoekt als diezelfde evenwichtsoplossing licht verstoord wordt (kleine verstoringen hebben uiteindelijk een klein effect).

De oorzaak van een grote amplificatie van kleine fouten in de afzonderlijke stappen van een rekenproces hoeft overigens niet altijd te liggen bij de numerieke methode: zie bv. experiment a in 3D. Als bijvoorbeeld het beginwaarde probleem van een autonome GDV een instabiel evenwicht als exacte oplossing heeft dan zal dat probleem met geen enkele numerieke methode stabiel opgelost kunnen worden (waarom niet?). Door een mathematische herformulering (in dit voorbeeld, door de tijd 'om te draaien', d.w.z. door t door $-t$ te vervangen) kan het evenwicht wel toegankelijk gemaakt worden voor een stabiele numerieke berekening (zie dat in).

Vraag 4A. In experiment c in 3H hebben we gezien dat we voor een acceptabel numeriek resultaat de h wel eens erg klein moeten nemen. Zeker voor meerdimensionale GDVs geven kleine h aanleiding tot lange rekentijden. Om efficiëntie redenen is het gewenst h zo groot mogelijk te kiezen. Hoe groot kan h maximaal zijn in experiment c in 3H? Hoe kritisch is die maximale waarde?

Experiment a. Probeer voor (1.2) (`Linear`) met $\lambda = -500$ zo goed mogelijk de maximale waarde van de h te vinden.

Verklaring. Kijk eens naar (3.7) en naar de waarde van de factor $|1 + \lambda h|$ waarmee EF voor deze GDV oude fouten voortplant.

Experiment b. Voor $\lambda = -400$ en $h = 0.005$ krijgen we een nauwkeurige numerieke oplossing. Voor $h = 0.005\,01$ is $|1+h\,\lambda| = 1.004$. Wat denk je: is de numerieke oplossing ook nauwkeurig?

Verklaring. Hoeveel stappen heb je nodig om in $T = 20$ te komen?

Experiment c. Met $\lambda = 0.1$ en $h = 0.04$ is $|1 + \lambda h| = 1.004$. Dit is precies de waarde in ons vorige experiment waarvoor we slechte resultaten kregen. Hoe pakt dat hier uit?

Verklaring. Hoeveel stappen heb je nu nodig?

> **Conclusie. a.** *Als* $|h\,\lambda| \ll 1$ *(denk aan* $|h\,\lambda| \leq 0.2$*) dan plant EF de fouten voort op ongeveer dezelfde manier als de GDV dat doet.*
> **b.** *Als* $|h\,\lambda| \not\ll 1$ *dan kan je alleen een nauwkeurige numerieke oplossing krijgen als (vrijwel) exact voldaan is aan de* stabiliteitseis $|1 + h\,\lambda| \leq 1$.

Vraag 4B. Bovenstaande conclusie geldt voor de eendimensionale lineaire GDV in (1.2) voor $\lambda(t) = \lambda_0$ is constant. Wat hebben we eraan voor gecompliceerdere GDVs? Blijft dan nog iets van de conclusie overeind? Kunnen we gemakkelijk zien of we met instabiliteiten te maken hebben?

Experiment a. Pas EF weer toe op (1.2) maar neem nu een variabele λ: neem λ als in (1.2.d) met $\varepsilon = 0.02$ (`Linear2`).[17] Voor welke h heb je geen stabiliteitsproblemen? Bekijk voor $h = 0.04$ ook eens de grafiek van $h\lambda(t)$.[18] Kan je de instabiliteit verklaren? Kan je, aan de hand van de grafiek van $h\lambda$ schatten voor welke h je van de instabiliteit af bent?

Verklaring. Op trajecten waar $|1 + h\,\lambda(t)| > 1$ blaast de fout op.

Experiment b. We kijken eens wat er van onze inzichten overblijft voor niet-lineare GDVs. Pas EF toe op (1.3) (`Flame`) met $\mu = \frac{1}{4}$ en $\kappa = 20$. Probeer de "kritische" h te vinden. Plot ook eens de grafiek van $h\,\lambda(t)$, nu met $\lambda(t) \equiv \frac{\partial f}{\partial u}(t, u(t))$.[19]

Verklaring. De fouten gedragen zich min of meer als de oplossing van een lineaire GDV (en wel de GDV (3.8)).

Experiment c. Los (1.6) (`vdPol`) op met EF en stapgrootte $h = 0.04$. Plot een faseplaatje. Voor welke waarde van h raak je de 'wiggels' kwijt? Bekijk voor deze h ook $h\,\lambda_i(t_n)$ met $\lambda_1(t_n)$ en $\lambda_2(t_n)$ de eigenwaarden van de Jacobi matrix $\frac{\partial f}{\partial u}(t_n, u_n)$.[20]

Verklaring. In de meerdimensionale situatie hangt de stabiliteit samen met $h\,\lambda_i(t)$ waarbij $\lambda_i(t)$ de i-de eigenwaarde is van de Jacobi matrix $h\,\frac{\partial f}{\partial u}(t, u(t))$.

> **Conclusie.** *Ook voor niet-lineaire (meerdimensionale) GDVs manifesteert instabiliteit zich in de vorm van 'wiggels' in de numerieke oplossing. In zo'n geval moet de stapgrootte h kleiner genomen worden. Wiggels wijzen op instabiliteit. In geval van instabiliteit krijgen we gewoonlijk niet een onnauwkeurige oplossing die er toch "redelijk" uitziet.*

We kunnen nu ook beter omschrijven wanneer, in de conclusie na Vraag 3H, 'h voldoende klein' is.

> **Conclusie.** *De EF fout is min of meer evenredig met h als h voldoende klein is, en dat is het geval als* (i) *én* (ii) *geldt. Hierbij is:*
> (i) *voor alle i en vrijwel alle t geldt dat* $|h\lambda_i(t)| \ll 1$ *of* $|1 + h\lambda_i(t)| \leq 1$
> (ii) *hogere orde Taylor reeks termen in het defect verwaarloosbaar zijn.*

[17] Deze GDV staat in `Linear2.m`.

[18] Neem in `Vraag4.m`, `ShowEigenvaluesJacobian = 1;`. Na de volgende twee experimenten zul je begrijpen waarom we de naam 'ShowEigenvaluesJacobian' gekozen hebben voor het plotten van de grafiek van $h\lambda(t)$.

[19] Neem `ShowEigenvaluesJacobian = 1;`.

[20] Neem `ShowEigenvaluesJacobian = 1;`.

Conditie (i) betreft stabiliteit (foutvoortplanting), (ii) is een eis op de lokale nauwkeurigheid (op het defect). Voor sommige GDVs (en nauwkeurigheidswaarde) leidt de stabiliteitseis al tot voldoende nauwkeurigheid (d.w.z., voor die h's waarvoor aan de stabiliteiteis voldaan is zijn de defecten τ_n al vanzelf zo klein dat de grootte van de resulterende fouten e_n onder een verlangde waarde liggen). In andere gevallen leidt het eisen van een klein defect vanzelf tot stabiliteit (d.w.z., de h waarvoor τ klein is, is al zo klein dat aan de stabiliteitseis voldaan is). Dus, stabiliteit dicteert soms de grootte van h, en in andere gevallen doet de nauwkeurigheid dat.

Vraag 4C. Voor welk type GDVs kunnen we stabiliteitsproblemen verwachten met EF?

Experiment. Los (1.2) (`Linear`) met $\lambda = -40$ op met EF en $h = 0.05$. Plot ook de exacte oplossingen door (t_n, u_n) voor de eerste paar n en kijk eens hoe die exacte oplossingen zich gedragen: zoom in bij $t = 0$. Neem nu eens $h = 0.07$ en $T = 0.5$. Zoom in, nu rond de hele exacte oplossing. Kan je grafisch zien waarom EF instabiel is?

Het beginwaarde probleem in bovenstaand experiment is stijf. In **stijve** problemen zijn verstoringen van de oplossing snel niet meer te zien. Wat precieser, als de oplossing u in een zeker tijdstip t_1 verstoord wordt en $\widetilde{u}$ is de verstoorde oplossing, d.w.z. $\widetilde{u}$ is de oplossing van de GDV waarvoor $\widetilde{u}(t_1) = u(t_1) + \varepsilon$, dan is $|\widetilde{u}(t) - u(t)| \ll |\varepsilon|$ voor $t \geq t_1 + \delta$, met δ 'klein'. Verstoorde of 'naburige' oplossingen veranderen in geval van een stijf probleem in een kort tijdsbestek zeer 'snel' maar zijn daarna niet meer te onderscheiden van de onverstoorde oplossing (dus als $\widetilde{u}(t_1) = u(t_1) + \varepsilon$, dan is, voor $t \geq t_1, t \approx t_1$, de waarde $|\widetilde{u}'(t)|$ groot in vergelijking met $|u'(t)|$, terwijl voor $t \geq t_1 + \delta$, $\widetilde{u}'(t)$ en $u'(t)$ min of meer identiek zijn. Hetzelfde geldt voor hogere afgeleiden van $\widetilde{u}$ en u).

> **Conclusie.** *In geval van een stijf beginwaarde probleem dicteren bij EF de naburige 'snel' variërende oplossingen welke stapgrootte er genomen moet worden ook al variëert de oplossing waarin we geïnteresseerd zijn 'langzaam'.*

Langzaam variërend wil voor de numerieke methode zeggen dat de defecten τ_n in absolute waarde klein zijn voor niet al te kleine stapgrootte h (denk aan $|\tau_n| \leq 10^{-3}$ voor $h = 0.02$). Voor EF betekent dat dat $|u''|$ niet al te groot mag zijn: immers voor EF is $\tau_n = \frac{1}{2}h^2 u''(\xi_n)$.

Stijve GDVs komen in de praktijk nogal eens voor. Ze ontstaan typisch als aan een goed model, d.w.z. aan een GDV die het praktijk probleem al behoorlijk goed modelleert, ter perfectionering hogere orde afgeleiden met een klein gewicht worden toegevoegd. Je kan hierbij denken aan de (0-de orde) "GDV" $u(t) - g(t) = 0$ waaraan de term $\varepsilon u'(t)$ wordt toegevoegd: $\varepsilon u' + u - g = 0$. Dus $u' = -\frac{1}{\varepsilon}(u - g)$ met $u(0) = g(0)$. Voor $0 < \varepsilon \ll 1$ is $u \approx g$ terwijl naburige oplossingen van de vorm $w = u + c\,e^{-t/\varepsilon}$ zijn. Hier is c een zekere constante. Omdat $w''(0) = u''(0) + \frac{c}{\varepsilon^2}$ zal $|w''(0)|$ groot zijn (tenzij $|c| \ll \varepsilon$). In de praktijk zijn de modellen (GDVs) natuurlijk wel wat ingewikkelder.

Vraag 4D. Kunnen we voor stijve GDVs met een nauwkeurige numerieke methode met grotere stapgroottes toe?

Experiment. Vergelijk de grootte van de stapgrootte die EF vereist met die voor RK4 voor de GDV (1.2) met $\lambda = -500$ en weer $T = 20$ (zie experiment c in 4A). Hoeveel groter kan de stapgrootte met RK4 zijn? Zie je deze factor ook terug bij ingewikkeldere GDVs? Bekijk ook eens (1.3) (zie experiment b in 4B).

Vraag 4E. We hebben hierboven gezien dat een nauwkeurigere methode voor stijve GDVs ook wel eens een heel kleine stapgrootte kan vergen. Je kan je afvragen of er überhaupt wel methoden bestaan waarmee je ook met een grotere stapgrootte nauwkeurige resultaten kan behalen?

We proberen de Euler backward methode (EB). EB is gebaseerd op de 'terugwaartse' differentie (vergelijk dit met (3.1))

$$u'(t) \approx \frac{u(t) - u(t-h)}{h}, \tag{4.1}$$

Dit leidt tot de relatie

$$u(t_n) = u(t_{n-1}) + h\, f(t_n, u(t_n)) + \tau_n^{\mathrm{EB}} \quad \text{waarbij} \quad \tau_n^{\mathrm{EB}} = -\tfrac{1}{2}\, h^2\, u''(\xi_n). \tag{4.2}$$

Wegpoetsen van de defecten τ_n^{EB} geeft EB.

Experiment a. Gebruik de routine '`odeEB`' om de GDV uit experiment a in 4B (`Linear2`) op te lossen. Met welke h (zo groot mogelijk) krijg je nog redelijke oplossingen? Vergelijk dit met de situatie voor EF. Hoe pakt het uit voor de GDV (`vdPol`) in experiment c in 4B? Neem ook eens in `Linear2` $\varepsilon = 10^{-6}$ en $h = 0.1$.

Verklaring. Probeer EB eens grafisch te interpreteren.
Verder geldt voor de fouten

$$e_n = \tfrac{1}{1-h\,\lambda_n}(e_{n-1} + \tau_n^{\mathrm{EB}}). \tag{4.3}$$

Opgave. *Bewijs* (4.3).

Experiment b. Wat verwacht je van EB voor stijve niet-lineaire GDVs? Pas EB ook eens toe op (1.7) (`Emigration`) met $\alpha = 2$, $\gamma = 1$, $\varepsilon = -0.01$. Wat is de "optimale" stapgrootte? Hoe ligt dat voor EF? Moet je de "optimale" stapgrootte voor EB en EF nog herzien als je $\varepsilon = -0.001$ neemt?

Verklaring. De fouten gedragen zich min of meer als de oplossing van een lineaire GDV (zie (3.8)).

Conclusie. *EB is aantrekkelijk met name voor stijve GDVs.*

Opgave 4F. *Bewijs, voor $d = 1$, dat EB "convergeert". Ga daarbij te werk als in opgave 3K.*

Vraag 4G. Waarom werken we niet altijd met EB?

Experiment a. Probeer eens een stap EB uit te voeren "met de hand" voor (1.3).

Experiment b. Los (1.4) (`Immuun`), nu met $\kappa = 1$ and $\vartheta = 0.25$, eens op met EB met $h = 0.1$ en ook met EF met $h = 0.1$. Hoe kan je de verschillen verklaren? Plot ook $h\,\lambda_i(t_n)$ als functie van t_n. Neem nu eens voor h in EB de waarde 0.11.

Verklaring. Kijk eens naar de foutvoortplantingsfactor $\frac{1}{1-h\,\lambda_n}$ van EB voor de λ's die we hier tegen kwamen.

Experiment c. Los (1.4) (`Immuun`) op met EF en met EB in beide gevallen met een betrouwbaarheid van 0.01 over het hele interval $[0, T]$, $T = 40$. Welke methode is het efficiëntst hier?

Verklaring. Hint: Is stabiliteit een probleem voor EF in dit voorbeeld?

> **Conclusie. a.** *Per stap is EB, zeker voor niet-lineaire en meerdimensionale GDVs veel 'duurder' dan EF. Het gebruik van EB betaalt alleen af als EB met veel minder stappen toe kan.*
> **b.** *Ook EB kan stabiliteitsproblemen hebben. Die treden met name op als er grotere positieve eigenwaarden zijn.*

In experiment a hierboven hebben we gezien dat het niet eenvoudig is om de EB stappen uit te voeren. De code `odeEB` speelt dat toch voor elkaar. In een volgend hoofdstuk zullen we zien hoe dat mogelijk is.

Vraag 4H. Zijn er methoden die een hogere orde nauwkeurigheid hebben dan EB (en EF) en waarmee je ook stijve GDVs met wat grotere stapgrootte nauwkeurig kan oplossen?

Experiment. Pas de trapeziumregel toe op (1.2) (`Linear2`) met λ als in (1.2.d) en $\varepsilon = 10^{-6}$ en $h = 0.1$ (zie experiment c in 4E). Vergelijk de nauwkeurigheid die je nu krijgt met die die je krijgt met EB.

Opgave. *Leid een uitdrukking voor de foutvoortplanting met de Trapeziumregel af analoog aan* (4.3).

Opgave 4I. *EF komt tot stand via een eerste orde Taylor reeks. We onderzoeken in deze opgave waar een tweede orde Taylor reeks toe leidt.*

a. *Laat zien dat*

$$u(t_{n+1}) = u(t_n) + h\, f(t_n, u(t_n)) + \tfrac{1}{2}\, h^2\, [\tfrac{\partial f}{\partial t}(t_n, u(t_n)) + \tfrac{\partial f}{\partial u}(t_n, u(t_n))\, f(t_n, u(t_n))] + \tau_n$$

geldt en geef een uitdrukking voor τ_n.

b. *Van welke orde is de resulterende methode?*

c. *Analyseer de stabiliteit van de methode.*

d. *Vergelijk de voor– en nadelen van deze methode met die van EF en van de Trapeziumregel.*

1.5 Efficiëntie

SLEUTELBEGRIPPEN: *Efficiëntie, kosten per stap, totale kosten, variabele stapgrootte, lokale fouten schatten, stapgroottebesturing*

Als twee rekenprocessen, zeg A en B, dezelfde grootheid X berekenen dan is A **efficiënter** als A minder rekentijd nodig heeft dan B om X even nauwkeurig of nauwkeuriger te bereken. De toevoeging 'even nauwkeurig' is crusiaal. Geen enkel rekenprocess produceert exacte uitkomsten:[21] de uitkomst zal dus altijd met een fout belast zijn. Het heeft geen zin om te zeggen dat A sneller rekent dan B als je van de uitkomst van proces A bijvoorbeeld niet meer weet dan dat één cijfer goed is, terwijl je kan aantonen dat

[21] tenzij de uitkomst uit gehele getallen of machine getallen bestaat en je deze informatie van te voren had en ook hebt kunnen uitbuiten in het ontwerpen van je rekenproces.

B 12 cijfers nauwkeurigheid produceert: je zit dan 'appels met peren' te vergelijken![22] Of een rekenproces efficiënter is kan overigens ook afhangen van de machine waarop je rekent en van de software (compiler) die je gebruikt: sommige rekenprocessen kunnen deels parallel worden uitgevoerd en daar kan het proces van profiteren als je machine meerdere processoren heeft en je compiler dat weet uit te buiten. Om de machine afhankelijkheid zullen we ons in deze cursus verder niet bekommeren (we doen alle berekening op een standard PC of SUN systeem en we gebruiken MATLAB).

Is EF efficiënter dan EB? Dit is een zinloze vraag. Je zult eerst moeten vaststellen welk (beginwaarde) probleem je wilt oplossen (wat is X?) en hoe nauwkeurig je dat wilt doen (en eigenlijk ook op welke machine). Dus: voor 'n specifieke GDV en 'n specifiek nauwkeurigheid zou EF efficiënter kunnen zijn dan EB en in 'n ander geval wellicht niet. In het algemeen is het overigens onmogelijk om *exact* vast te stellen met welke nauwkeurigheid het rekenproces daadwerkelijk de uitkomst berekend heeft. Gelukkig kunnen we wel de nauwkeurigheid schatten (zie Vraag 3C) en kunnen we vaststellen of die schatting betrouwbaar is (zie Vraag 3G). Onze efficiëntie uitspraken baseren we daarom op dit soort schattingen. Het schatten van de fout en het inzien of de schatting betrouwbaar is kost overigens ook rekentijd. Je kan je afvragen of je die tijd in je efficiëntie analyse mee moet tellen. In onze experimenten en opdrachten kiezen we ervoor om dat niet te doen, en wel om de volgende reden. In de praktijk voert men vaak een efficiëntie analyse uit voor een *model probleem*, dat wil zeggen voor een lager dimensionale vereenvoudigde versie. Of, als je een familie van problemen hebt die afhangen van 'n parameter, doe je de analyse voor één specifieke goedgekozen waarde voor de parameter. Je hoopt dan dat de conclusies ook correct zijn voor het echte probleem of voor het probleem met de andere waardes voor de parameter. De rekentijd die in het betrouwbaar schatten van de fout gaat zitten (en jouw tijd om alles te analyseren? :-)) is dan verwaarloosbaar ten opzichte rekentijd die nodig is om het echte probleem, of de familie van problemen, op te lossen. De echte 'sterke' codes schatten overigens ook de fout en zijn daardoor trager dan hun 'zwakkere broeders'. Als je professionele codes wilt gebruiken moet je besluiten of je voor betrouwbaarheid kiest (de sterke maar langzamere code met betrouwbare foutschatting) of voor efficiëntie (de snellere code waarbij je zelf maar moet zien te achterhalen hoe nauwkeurig de antwoorden zijn).

EF kan je met diverse stapgroottes toepassen. De efficiëntste variant van EF voor een specifiek probleem met een specifieke nauwkeurigheid krijg je uiteraard als je met de grootste stapgrootte werkt die de oplossing produceert met de gewenste nauwkeurigheid.[23] In iedere efficiëntie analyse voor het oplossen van een specifiek probleem, moet je dus eerst, nadat je de gewenste nauwkeurigheid hebt vastgeteld, achterhalen wat de grootste stapgrootte is die je je, gezien de gewenste nauwkeurigheid, kunt permiteren. Eenzelfde opmerking geldt uiteraard ook voor EB en voor andere numerieke oplosmethoden.

Je kunt natuurlijk toch de foute vraag stellen: is EF efficiënter dan EB? Alleen kan je dan geen eenduidig antwoord verwachten.

Vraag 5A. Is, voor een nauwkeurigheid van 10^{-3}, EF efficiënter dan EB voor het oplossen van (1.2) met λ als in (1.2.d)?

[22] Als je je niet om nauwkeurigheid bekommert zou je, ongeacht het probleem, altijd kunnen zeggen dat de uitkomst π is en voor die uitkomst hoef je zelfs niet te rekenen :-)

[23] de rekentijd is ruwweg evenredig met het aantal stappen dat je nodig hebt om van t_0 in $t_0 + T$ te komen, en is dus ongeveer omgekeerd evenredig met de stapgrootte h

Experiment. Als in de experimenten a in de Vragen 4E en 4G passen we EF en EB toe op `Linear2`. Neem eerst $\varepsilon = 10^{-3}$ en vind de grootste stapgrootte h_{eb} waarvoor EB de gewenste nauwkeurigheid haalt.[24] Hoeveel rekentijd kost het oplossen bij deze stapgrootte h_{eb}?[25] (Schrijf de timings op: je hebt ze weeer nodig voor het experiment in Vraag 5C.) Gebruik de eigenwaardencurve die bij EB hoort om een geschikte stapgrootte voor EF te schatten. Hoeveel tijd heeft EF nodig? Neem nu eens $\varepsilon = 10^{-5}$. Kan je voor EB weer de h_{eb} als voorheen nemen? Verklaring? Hoeveel tijd heeft EB nu nodig? Kan je schatten hoeveel tijd EF nodig zal hebben? Neem tenslotte $\varepsilon = 10^{-1}$ en schat de benodigde stapgroottes voordat je de computer laat rekenen.

Onderstaand antwoord op de vraag of EF efficiënter is vat onze bevindingen uit bovenstaand experiment samen.

> **Conclusie.** *Als voor EF nauwkeurigheid de factor is die de grootte van h bepaalt en niet stabiliteit dan* is *EF veel efficiënter dan EB. Als stabiliteit de bepalende factor is voor EF dan* kan *EB (heel veel) efficiënter zijn.*

Als stabiliteit de stapgrootte bepaalt bij EF dan kan EB met een grotere stapgrootte werken, en dus met minder stappen. Of hierdoor de totale rekentijd ook korter wordt hangt af van de kosten per stap: EB stappen zijn gewoonlijk duurder dan EF stappen.

Als we kijken waar de problemen liggen voor EF bij het oplossen van (1.2) met λ als in (1.2.d) voor kleine ε, dan zien we dat we alleen om stabiliteitsredenen in een klein tijdsinterval met een kleine stapgrootte moeten werken. Dat roept de vraag op of we de stapgrootte niet zouden kunnen aanpassen per tijdsinterval. Dit is precies wat de methode `ode23` doet: in `ode23` wordt in ieder tijdstip t_n de stapgrootte h (zeg h_n) zo gekozen dat de geschatte waarde van het defect τ_n in absolute waarde (of in norm) ietjes kleiner is dan een zekere tolerantie *tol*.[26] Als het geschatte defect met $h_n = h_{n-1}$ ongeveer de waarde *tol* heeft, dan wordt de stapgrootte geaccepteerd (d.w.z. de stap wordt gezet: $t_{n+1} = t_n + h_n$). De stapgrootte wordt ook geaccepteerd als het geschatte defect met $h_n = h_{n-1}$ veel kleiner is dan *tol*, maar dan wordt voor de volgende stap de stapgrootte groter gekozen. Is daarentegen het geschatte defect groter dan *tol* dan wordt de stapgrootte verkleind voordat de stap daadwerkelijk gezet wordt en wellicht wordt eerst de stapgrootte nogmaals aangepast. De factor waarmee vergroot of verkleind wordt, wordt gekozen op grond van het feit dat het defect in `ode23` evenredig is met h^3 (de methode is van orde 2. Voor het schatten van de grootte van het defect wordt een verwante methode van orde 3 gebruikt. Vandaar de 2 en 3 in `ode23`).

Vraag 5B. EF (en ook EB en de trapeziumregel) zoals wij die gebruiken werkt met één en dezelfde stapgrootte in ieder tijstip t_n. Voor GDVs waarvan de afgeleide van de oplossing maar een deel van de tijd groot is, is dat wellicht niet handig. Zou `ode23` hier efficiënter zijn?

Experiment. Voor welke stapgrootte berekent EF de waarde $u(t)$ voor (1.5) (`vdPol`)

[24] In het algemeen is de exacte oplossing niet bekend en moeten we de fout schatten. Voor deze GDV is de exacte oplossing wel bekend en eenvoudigheidshalve maken we daar gebruik van: neem `PlotError=1`. Dan verschijnt de oplossing en de fout in figure 6. Neem eerst `dt = 0.01;`. Kijk naar de grootte van de fout en gebruik dit om de stapgrootte te schatten die de gewenste nauwkeurigheid haalt.

[25] Maak het verschil of je `ShowSolutionProcess = 0;` hebt? Hoe komt dat?

[26] De waarde voor *tol* moet door de gebruiker van de methode gespecificeerd worden. In ons programma is dat de parameter `RelTol`.

in $t = 10$ met een nauwkeurigheid van 10^{-2}?[27] Kies de toleratie *tol* in `ode23` zodat `ode23` dezelfde nauwkeurigheid haalt in $t = 10$. Welke methode is het efficiëntst?

Verklaring. Kijk eens naar de stapgroottes die `ode23` genomen heeft.[28]

We hebben de mogelijkheid van variabele stapgrootte aangekaart omdat EF in (1.2) met λ als in (1.2.d) maar in een beperkt deel van het tijdsinterval 'behoefte' had aan een kleine stapgrootte. Die behoefte was er om stabiliteitsredenen. In de strategie waarmee `ode23` zijn stapgrootte bestuurt wordt echter geen rekening gehouden met de stabiliteit.

Vraag 5C. Werkt de stapgroottebesturings strategie van `ode23` ook goed in de gevallen waarin EF stabiliteitsproblemen heeft? Welke methode is dan het efficiëntst?

Experiment. Hoe efficiënt is `ode23` voor de GDV in het experiment in Vraag 5A (`Linear2`) voor de waarden 10^{-3} en 10^{-5} voor ε en een nauwkeurigheid van 10^{-3}? Stel eerst vast hoe groot de toleratie *tol* moet zijn om met `ode23` de gewenste nauwkeurigheid te kunnen halen. (Haalt `ode23` de 'beloofde' nauwkeurigheid?) Welke methode (EF, EB of `ode23`) is het efficiëntst? Hangt dat nog van ε af?

Voor $\varepsilon = 10^{-3}$ lijkt `ode23` te winnen. Kijk ook eens naar de nauwkeurigheid in de afgeleide bij `ode23` en EB.[29]

Hoewel `ode23` mischien minder nauwkeurig is dan het op het eerste gezicht lijkt (we hadden overigens alleen een zekere nauwkeurigheid in de oplossing verlangd en niet in diens afgeleide!), lijkt de methode eventuele stabiliteitsproblemen automatisch op te lossen. Hoe komt dat?

Verklaring. Laat je inspireren door de exacte oplossingen door de numerieke verkregen waarden.[30]

Vraag 5D. De strategie waarmee `ode23` de stapgrootte bestuurt is gericht op (lokale) nauwkeurigheid, maar lijkt impliciet stabiliteitstproblemen op te lossen. Is het verstandig om `ode23` voor stijve problemen te gebruiken?

Experiment. Los (1.2) met $\lambda = -400$ (`Linear1`) zo efficiënt mogelijk op met EF en ode23 met een nauwkeurigheid van 10^{-3}.[31] Welke methode is het efficiëntst? Aan welke methode geef je voor deze GDV de voorkeur, EF, EB of `ode23`?

> **Conclusie.** *Stapgroottebesturing gebaseerd op controle van de lokale nauwkeurigheid leidt tot een efficiënte methode in geval de oplossing in delen van het tijdsinterval sterk variëert. De strategie kan ook effectief gebruikt worden voor problemen die maar in een beperkt tijdsinterval stijf zijn en waarbij de stijfheid niet al te groot is.*

---

[27] De exacte oplossing is niet bekend, maar met `PlotError = 1;` krijg je een schatting voor de fout in $t = T$.

[28] Met `PlotDensityStepsizes = 1;` krijg je in figuur 7 de $\log_{10}(1/h_n)$ te zien in iedere t_n. Dit geeft een beeld van de 'dichtheid' van de stappen: in een gebiedje waar de stapgrootten klein zijn (d.w.z. waar de $1/h_n$ groot zijn), worden veel stappen gezet. In hetzelfde figuur staat ook de oplossing. Let op het verband tussen de stapgrootten en veranderingen in de functiewaarden (d.w.z. de grootte van de afgeleiden).

[29] Neem `J = [0,1,2];` en laat de fout plotten: `PlotError = 1;`.

[30] Neem `PlotExactNearbySol = 1;` en kijk naar de gevonden oplossingen voor $t \approx 0$. Tip: inzoomen kan handig met het commando `axis([0,0.08,0.997,1.003])`.

[31] Neem weer `PlorError =1;`.

De stapgroottebesturing gebaseerd op controle van de lokale nauwkeurigheid werkt overigens ook voor echte stijve problemen, alleen moet de strategie ingebouwd worden in een methode met sterke stabiliteitseigenschappen als EB; de methode waar `ode23` op berust heeft stabiliteitseigenschappen die vergelijkbaar zijn met EF.

1.6 Toepassingen

In onderstaande "praktijk problemen" moet je proberen de inzichten, die je je in de vorige paragrafen eigen gemaakt hebt, toe te passen.

Opdracht 6A. Onderzoek welke numerieke methode, EF of EB, het efficiëntst het beginwaarde probleem (1.3) (`Flame`) met $\mu = \frac{1}{4}$ en $\kappa = 20$ met een nauwkeurigheid van 0.03 kan oplossen (d.w.z., met een betrouwbaarheid van 0.03. Let op: de exacte oplossing is niet bekend.[32] Leg uit hoe je de fouten schat en waarom dat je denkt dat de schattingen betrouwbaar zijn. Hangt een en ander nog af van het tijdstip t waarin je schat?). Doet EB het veel beter dan EF? Verklaar de verschillen. (Betrek in je verklaring de grootheden die de efficiëntie bepalen: het stabiliteitsgedrag, de grootte van de defecten, de kosten per stap. Moet je conclusies herzien bij een andere eindtijd T?).

Opdracht 6B. Los het beginwaarde probleem (1.8) (`Body3`) op met RK4 met $h = 0.007$.[33] We krijgen geen Arenstorf baan (gesloten kromme). Hoe komt dat? Is RK4 bij deze stapgrootte nog niet nauwkeurig genoeg of heeft RK4 problemen met de stabiliteit (zoom eens in rond de positie $(\nu, 0)$ van M. Wanneer is $d_M \approx 0$? Wat betekent dat voor x'' en de defecten? Wat leer je van de eigenwaarden? Is 'n grote eigenwaarde in alleen de eerste stap fataal?)? Probeer ook eens $h = 0.0068$. Voor welke h krijg je wel een Arenstorf baan? Kan je ook met een praktische stapgrootte met EF een Arenstorf baan krijgen? Verwacht je dat het beter gaat met EB?

Probeer ook eens `ode23`. Heb je een hogere `RelTol` nodig? Kan je zien hoe `ode23` het klaar speelt om zo efficiënt een redelijk nauwkeurige oplossing te krijgen? Plot de coördinaten van de oplossing ook eens als functie van de tijd en probeer te achterhalen met welke stapgrootte `ode23` werkt. Is die overal hetzelfde? Is er een 'fysische' verklaring waarom in sommige stukken van de baan 'nauwkeuriger' rekenwerk vereist is?

Opdracht 6C. Onderzoek voor de van der Pol vergelijking (1.5) (`vdPol`) het numerieke gedrag van EF, EB en `ode23`. Besteed aandacht aan stapgrootten die aspecten laten zien die voor de numerieke theorie interessant zijn.[34] Betrek in je analyse niet alleen de stabiliteit van de methoden maar ook de stabiliteit van het beginwaarde probleem.[35] Draai ook eens de tijd om.[36]

Opdracht 6D. Analyseer de convergentie van EF en EB (zie 3K) en vergelijk het numerieke gedrag van deze twee methoden. Gebruik `linear2` ter illustratie.

---

[32] Met `PlotExactSolution = 1;` kan je MATLAB vragen om de exacte oplossing te plotten, maar het programma 'slaat er maat een slag naar'.

[33] `ODE='odeRK';`.

[34] Bekijk ook een voor `ode23` $h = 0.01$, $h = 0.001$, `RelTol`$= 10^{-3}$ en 10^{-4}

[35] Inspecteer het richtingsveld: neem `PlotFlowField=1`.

[36] Hiervoor hoef je maar alleen in `vdPol.m` de eindtijd te vervangen door -1 maal de eindtijd.

1.7 Samenvatting

Bij GDVs wordt de fout in het numeriek resultaat bepaald door

1. *de wijze waarop de GDV fouten voortplant,*
2. *de fouten die in iedere stap van het rekenproces geïntroduceerd worden,*
3. *de wijze waarop de numerieke oplosmethode fouten voortplant.*

In §§1.7.1-1.7.3 analyseren we achtereenvolgens deze drie aspecten voor het eendimensionale beginwaarde probleem

$$\begin{cases} u'(t) = \lambda(t)u(t) + g(t), \quad t \in [0,T] \\ u(0) = u_0. \end{cases} \tag{7.1}$$

Hierin zijn λ en g gegeven reëel (eventueel complex) waardige functies en u_0 is een gegeven scalair. u is de exacte oplossing van (7.1). De functie f is dus in deze §§ gegeven door $f(t,u) = \lambda(t)u + g(t)$. Daarna geven we in §1.7.4 aan hoe de resultaten er voor niet-lineaire – en meerdimensionale GDVs er uit zien.

1.7.1 Foutvoortplanting door de GDV

Een verstoring van bijvoorbeeld de beginvoorwaarde leidt tot een andere oplossing. In sommige gevallen kan die oplossing flink afwijken van de beoogde oplossing. Dat kan zelfs gebeuren voor heel kleine verstoringen. In zo'n geval mogen we niet verwachten dat we de GDV numeriek nauwkeurig kunnen oplossen: immers in een numeriek proces zijn fouten onvermijdelijk (en fouten worden niet alleen in het begin gemaakt!).

Om te achterhalen wat we op zijn minst aan fouten in de numerieke resultaten mogen verwachten analyseren we het effect van een verstoring ε op de beginvoorwaarde in (7.1).

Als $\widetilde{u}$ de 'verstoorde' oplossing is van (7.1), d.w.z. de oplossing van

$$\begin{cases} u'(t) = \lambda(t)u(t) + g(t), \quad t \in [0,T] \\ u(0) = u_0 + \varepsilon, \end{cases} \tag{7.2}$$

dan geeft

$$e(t) \equiv u(t) - \widetilde{u}(t) \tag{7.3}$$

aan hoe de verstoring van de beginvoorwaarde doorwerkt op de oplossing u in het tijdstip t. Trekken we (7.2) af van (7.1) dan zien we dat

$$\begin{cases} e'(t) = u'(t) - \widetilde{u}'(t) = \lambda(t)\, e(t), \\ e(0) = -\varepsilon. \end{cases} \tag{7.4}$$

Dus

$$e(t) = -c(t)\varepsilon, \quad \text{met} \quad c(t) \equiv \exp\left(\int_0^t \lambda(s)\, ds\right) : \tag{7.5}$$

$c(t)$ geeft aan hoe de verstoring door de GDV voortgeplant wordt.
Met $\lambda_{\min} \equiv \min_t \mathrm{Re}(\lambda(t))$ en $\lambda_{\max} \equiv \max_t \mathrm{Re}(\lambda(t))$ geldt $e^{\lambda_{\min} T} \leq |c(t)| \leq e^{\lambda_{\max} T}$. Als $\lambda_{\min} > 0$ dan wordt de verstoring vergrotend voortgeplant, als $\lambda_{\max} < 0$ is dat verkleinend. Met bv $\lambda_{\min} = 2$ en $T = 20$ is $e^{\lambda_{\min} T} = 2.4\,10^{17}$ en de GDV vergroot een fout ε in het begin minstens tot een fout $2.4\,10^{17}\varepsilon$ in de eindtijd $T = 20$.

Een zelfde soort uitspraak geldt met betrekking tot verstoringen gemaakt op latere tijdstippen. Afhankelijk van de functie λ kan de voortplanting van een verstoring op sommige delen van $[0,T]$ groeiend zijn en op andere delen krimpend.

Als $\lambda(t)$ over een tijdstraject ter lengte h vrijwel constant is, dan geldt

$$e(t+h) \approx e^{h\,\lambda(t)}\, e(t)\,: \tag{7.6}$$

over een tijdstraject ter lengte h wordt de oude fout, d.w.z. het effect van een eerdere verstoring, door de GDV voortgeplant met een factor $e^{h\,\lambda(t)}$. We mogen niet verwachten dat een numerieke oplosmethode fouten gunstiger zal voortplanten.

1.7.2 Lokale fouten in het rekenproces

Voor $h > 0$ voldoende klein geldt $u(t+h) \approx u(t) + h\,u'(t)$. Verder is $u'(t) = f(t,u(t))$. Dit leidt tot het volgend rekenproces:

$$u_{n+1} = u_n + hf(t_n,u_n), \qquad t_0 = 0, \quad t_{n+1} = t_n + h, \quad n = 0,1,2,\ldots, \tag{7.7}$$

waarin u_n een benadering is voor $u(t_n)$. Dit rekenschema heet de **Euler forward** (EF) methode. Voor de exacte oplossing in het tijdstip $t_{n+1} = t_n + h$ geldt:

$$u(t_{n+1}) = u(t_n) + h\,f(t_n,u(t_n)) + \tau_n, \quad \text{met} \quad \tau_n \equiv \tfrac{1}{2}\,h^2\,u''(\xi_n) \tag{7.8}$$

voor een zekere ξ_n in $[t_n,t_{n+1}]$ (gebruik een eerste orde Taylor benadering). De EF methode ontstaat dus door in ieder tijdstip t_n het **defect** τ_n weg te poetsen[37] en de exacte oplossing in t_n te vervangen door een benaderende.

In de numerieke literatuur schrijft men gewoonlijk f_n in plaats van $f(t_n,u_n)$. Deze notatie maakt de formules beknopter en we zullen deze notatie ook gebruiken. De afhankelijkheid van h wordt wel onder de tafel geveegd (maar dat was ook al in de notatie t_n gebeurd!). We schrijven λ_n in plaats van $\lambda(t_n)$.

De benadering u_{n+1} is gebaseerd op de benadering u_n van $u(t_n)$, die al met een zekere fout belast was. Bovendien wordt er nog een nieuwe fout gemaakt door het defect weg te poetsen. De **fout** (ook wel **globale fout** genoemd) $u(t_{n+1}) - u_{n+1}$ in het tijdstip t_{n+1} hangt dus af van de fout op het voorgaande tijdstip t_n en van een nieuwe fout, nl. het defect: EF maakt nieuwe fouten, maar "haalt ook nog eens oude koeien uit de sloot". Dit doet overigens iedere numerieke methode voor het oplossen van GDVs. Noteren we de fout in t_n met e_n,

$$e_n \equiv u(t_n) - u_n,$$

dan geldt (trek (7.7) af van (7.8))

$$e_0 = 0, \qquad e_{n+1} = (1 + h\,\lambda_n)\,e_n + \tau_n, \quad n = 0,1,2,\ldots. \tag{7.9}$$

Als $\lambda = 0$, d.w.z. , als $u' = g$ voor zekere $g = g(t)$, dan volgt uit (7.9) dat $e_n = \sum_{j<n} \tau_j$: de fout op een tijdstip is een cumulatie van defecten van voorgaande tijdstippen. Merk

[37] De grootheid $\delta_n \equiv \tau_n/h$ noemt men ook wel **lokale discretisatiefout**. In deze visie wordt $(u(t+h) - u(t))/h$ gezien als benadering voor $u'(t)$. Ga na dat $(u(t+h) - u(t))/h - \delta(t) = u'(t) = f(t,u(t))$ tot EF leidt.

op dat een cumulatie van defecten een orde h groter kan zijn dan de defecten afzonderlijk:

$$|\sum_{j<n} \tau_j| \leq \sum_{j<n} |\tau_j| \leq n \max_j |\tau_j| \leq \frac{T}{h} \max_j |\tau_j|.$$

Onder omstandigheden kunnen de ongelijkheden gelijkheden zijn.
In het algemeen is $\lambda \neq 0$ en hebben we niet simpel weg te maken met een cumulatie van defecten maar ook nog met de *manier* waarop oude fouten worden voortgeplant. Afhankelijk van de grootte in absolute waarde van $1+h\,\lambda_n$ worden oude fouten groeiend of krimpend voortgeplant. Deze factor wordt daarom de **foutvoortplantings factor** voor EF genoemd. Merk op dat

$$1 + h\,\lambda_n \approx e^{h\,\lambda_n} \quad \text{is als} \quad |h\,\lambda_n| \ll 1 \qquad (\text{denk aan } |h\,\lambda_n| \leq 0.2);$$

in dit geval planten EF en de GDV zelf fouten op min of meer dezelfde manier voort (zie (7.6)).

Als λ begrensd is onafhankelijk van h dan kunnen we de fout majoreren door een constante maal h.

Stelling 1.7.1 *Als $|\lambda|$ begrensd is op $[0,T]$ dan is er een $C > 0$ zodat voor iedere h, die voldoende klein is, en iedere n waarvoor $t_n \leq T$, geldt dat*

$$|u(t_n) - u_n| \leq C \sum_{j<n} |\tau_j| \leq C \left(\tfrac{1}{2} T \max_{\xi \leq T} |u''(\xi)| \right) h. \tag{7.10}$$

Door h naar 0 te laten gaan volgt **convergentie**.

Stelling 1.7.2 *Als λ begrensd is dan convergeert EF* **uniform** *op $[0,T]$, d.w.z.,*

$$\sup_n |u(t_n) - u_n| \to 0 \quad \textit{voor} \quad h \to 0$$

Hierbij is het supremum genomen over alle n waarvoor $t_n \leq T$.

Schatting (7.10) is niet erg praktisch: in de praktijk kennen we u'' niet. Schatting (7.10) geldt ook voor niet lineaire GDVs en dan kennen we zelfs C niet (zie §1.7.3). Gelukkig vertoont de fout, voor h voldoende klein, een "h-gedrag": er is een functie w, die niet afhangt van h en n, zodanig dat

$$u(t_n) - u_n \approx h\,w(t_n), \tag{7.11}$$

voor h voldoende klein (zie stelling 1.7.4). Omdat de fout zo fraai gestructureerd is kunnen we aan de hand van de numerieke oplossingen zelf de fout schatten:

Stelling 1.7.3 *Als $\tau \in [0,T]$ en h is zo dat $\ell h = \tau$ en voldoende klein is, dan geldt*

$$u(\tau) - u_{2\ell}^{(2)} \approx \tfrac{1}{2} \left[u(\tau) - u_{\ell}^{(1)} \right], \tag{7.12}$$

hierin is $u_n^{(1)}$ de EF benadering met stapgrootte h en $u_n^{(2)}$ die met stapgrootte $\frac{1}{2}h$.

De fout halveert dus min of meer als de stapgrootte halveert. De fout moet uiteraard wel gemeten worden in geschikte punten.

Stelling 1.7.4 *Als f voldoende glad is ($f \in C^2$), dan is er een functie w op $[0,T]$ en een $C > 0$ zodanig dat voor iedere h die voldoende klein is, geldt*

$$|u(t_n) - u_n - h\,w(t_n)| \leq Ch^2, \quad \textit{voor alle } n \textit{ waarvoor} \quad t_n \in [0,T]. \tag{7.13}$$

In de volgende sektie gaan we in op de vraag wat "h voldoende klein" betekent.
In plaats van "er is een C zodat voor iedere voldoende kleine h (7.13) geldt" zeggen we

$$u(t_n) - u_n = h\,w(t_n) + \mathcal{O}(h^2) \quad \text{uniform op} \quad [0,T]. \tag{7.14}$$

Het woord "uniform" wijst erop dat dezelfde C gebruikt moet worden voor alle h, voldoende klein, en *alle* n. In woorden luidt (7.14):

"de fout $u(t_n) - u_n$ is gelijk aan $h\,w(t_n)$ op een orde h^2-term na.

De stellingen gelden, in aangepaste vorm, ook voor de andere numerieke oplosmethoden die we in dit hoofdstuk gezien hebben: Euler Backward (EB), Trapezium regel en RK4 convergeren uniform op $[0,T]$ (zie St.1.7.2); voor voldoende kleine h worden de fouten gemajoreerd door Dh^p, waarbij D een zekere constante is en $p = 1$ voor EB, $p = 2$ voor de Trapezium regel, $p = 4$ voor RK4 (St.1.7.1); voor voldoende kleine h zijn de fouten evenredig met h^p (bv. $u(t_n) - u_n^{\text{trap}} = h^2 w^{\text{trap}}(t_n) + \mathcal{O}(h^3)$ uniform op $[0,t]$) en nemen de fouten af met een factor 2^p als h halveert (zie St.1.7.4). p is de **orde** van de methode.

1.7.3 Stabiliteit

Hoe groot moeten we de stapgrootte h kiezen om een numerieke oplossing te krijgen met een door ons gewenste nauwkeurigheid? Om efficiëntie redenen willen we h zo groot mogelijk nemen. De uitspraken in de stellingen 1.7.3 en 1.7.4 geven een hanteerbare foutschatting, maar vertellen niet wat "voldoende klein" is.

Schatting (7.10) geldt voor $C = e^{MT}$ waarbij $M = \max_t |\lambda(t)|$. De constante e^{MT} is vaak waanzinnig groot, ook in situaties waarin de GDV zelf verstoringen dempend voortplant. De waarde e^{MT} voor de constante C is tot stand gekomen door de foutvoortplantingsfactoren $|1 + h\,\lambda_n|$ af te schatten op $1 + h\,M \leq e^{hM}$. In het algemeen is die schatting te pessimistisch. Als $|h\,\lambda_n|$ klein is voor alle n dan is $1 + h\,\lambda_n \approx e^{h\,\lambda_n}$ en plant EF de fout op min of meer dezelfde manier voort als de GDV zelf. In dit geval is C min of meer gelijk aan de factor waarmee de GDV zelf verstoringen kan vergroten en beter mogen we niet verwachten. Als $|1 + h\,\lambda_n| \leq 1$ voor (vrijwel) alle n dan worden oude fouten (vrijwel) niet vergroot. In dit geval is $C = 1$ ($C \approx 1$). C is echt groot (en heeft niet alleen maar een grote waarde door te grove schattingen) als $|h\,\lambda_n| \not\ll 1$ en $|1 + h\,\lambda_n| > 1$ voor een groot aantal tijstippen t_n (voor zeer veel n).

Voor een nauwkeurige numerieke oplossing moeten de defecten voldoende klein zijn en de vergroting van de fout (door de foutvoortplanting) moet binnen de perken blijven. Zoals we net gezien hebben is dit laatste het geval als

(i) voor iedere n geldt (i.1) $|h\,\lambda_n| \ll 1$ (denk aan $|h\,\lambda_n| \leq 0.2$) of
(i.2) $|1 + h\,\lambda_n| \leq 1$.

Voorwaarde (i.2) is een *stabiliteitseis*: oude fouten mogen niet vergroot worden. In zo'n geval is $C = 1$ in (7.10) en wordt de nauwkeurigheid in de numerieke oplossing gemajoreerd door een cumulatie van defecten. Als (i.1) of (i.2) geldt dan wordt de fout in essentie volledig bepaald door Taylor resttermen en "voldoende klein" in de stellingen 1.7.3 en 1.7.4 betekent dan dat

(ii) hogere orde Taylor resttermen verwaarloosbaar zijn ten opzichte van de laagste.

Als $-M \leq \lambda(t) \leq 0$ voor alle t dan geldt (i.2) als $h \leq 2/M$ en voor dit soort h hangt verder het foutgedrag nog maar alleen af van Taylor resttermen. Voor bv. $M = 5$ is $2/M = 0.4$ en vormt de stabiliteitseis (i.2) niet echt een extra beperking. Voor grote M (denk aan $M \geq 10^4$) kan dat wel het geval zijn. Andere numerieke oplosmethoden met betere stabiliteitseigenschappen kunnen dan uitkomst bieden. Een variant op de afleiding van EF leidt tot zo'n methode.

Voor kleine h geldt ook $u(t-h) \approx u(t) - h\,u'(t)$ of, equivalent hiermee, $u(t) \approx u(t-h) + h\,f(t,u(t))$. Dit leidt tot de **Euler Backward** (EB) methode:

$$\begin{cases} u_n = u_{n-1} + h\,f(t_n, u_n), \\ u(t_n) = u(t_{n-1}) + h\,f(t_n, u(t_n)) + \tau_n \quad \text{met} \quad \tau_n \equiv -\frac{1}{2}\,h^2\,u''(\xi_n). \end{cases} \tag{7.15}$$

De eerste relatie is het rekenschema. Voor de fout $e_n \equiv u(t_n) - u_n$ geldt

$$e_n = (1 - h\,\lambda_n)^{-1}\,(e_{n-1} + \tau_n)\,. \tag{7.16}$$

Fouten in EB worden niet vergroot als

$$\left|(1 - h\,\lambda_n)^{-1}\right| \leq 1 \quad \text{voor alle} \quad n. \tag{7.17}$$

In dat geval kan de fout gemajoreerd worden door $\sum_{j<n} |\tau_j|$.
Als bv. $\mathrm{Re}(\lambda(t)) \leq 0$ voor alle t, dan geldt (7.17) voor iedere h en de nauwkeurigheid in de numerieke oplossing wordt volledig bepaald door de defecten, ongeacht hoe groot $|\lambda(t)|$ is. Als $\mathrm{Re}(\lambda(t)) \leq 0$ dan hoeven we ons met EB alleen om de grootte van de defecten te bekommeren, terwijl we met EF ook nog rekening moeten houden met de stabiliteit. Voor zogenaamde **stijve** GDVs, GDVs waarbij $\mathrm{Re}(\lambda(t))$ negatief is maar absoluut groot en $|u''|$ niet al te groot, kunnen we met EB met veel grotere stappen werken dan met EF. Helaas kleeft er ook een nadeel aan EB. In iedere stap moet u_n uit de vergelijking $u_n - h\,f(t_n, u_n) = u_{n-1}$ *opgelost* worden, terwijl in EF u_n simpel volgt door *evaluatie* van $f(t_{n-1}, u_{n-1})$. EF wordt daarom een **expliciete** methode genoemd, en EB noemt men een **impliciete** methode.

1.7.4 Niet-lineaire en meerdimensionale GDVs

Vervangen we in geval de eendimensionale GDV niet-lineair is de functie $\lambda(t)$ in de uitspraken over fouten door

$$\lambda(t) = \frac{\partial f}{\partial u}(t, u(t))$$

dan vinden we in essentie de correcte uitspraken voor niet-lineaire GDVs. Dit laat zich als volgt verklaren. Stellen we voor het niet-lineaire geval relaties op voor de fout dan lopen we aan tegen $f(t,u(t)) - f(t,u(t)+e)$ hetgeen min of meer gelijk is aan $\frac{\partial f}{\partial u}(t,u(t))e$.

Het rekenschema voor meerdimensionale problemen is exact gelijk aan dat voor een-dimensionale. De discussie betreffende nauwkeurigheid en stabiliteit loopt ook niet wezenlijk anders. In plaats van scalairen komen in de discussie vectoren en matrices voor. De grootheden u_n^* en u_n zijn dan d-dimensionale vectoren terwijl voor $\frac{\partial f}{\partial u}$ de Jacobi matrix $(\frac{\partial f_j}{\partial u_i})$ gelezen moet worden: $\frac{\partial f_j}{\partial u_i}$ is het (i,j)-de element van die matrix. De absolute waarde $|v|$ van de vector $v = (v_1, \ldots, v_d)^T$ moet men, voor 'n geschikte "norm",

lezen als de norm van de vector: bv. $(|v| =) \|(v_1, \ldots, v_d)^T\| = \max_{i \leq d} |v_i|$. Evenzo wordt de absolute waarde van een matrix de norm van die matrix. Normen van matrices zijn lastiger te interpreteren dan normen van vectoren. Omdat normen van matrices wellicht geen vertrouwde objecten zijn werken we dit niet verder uit. We volstaan met de opmerking dat de norm van een matrix gerelateerd kan worden aan de eigenwaarden van de matrix. Ruwweg gesproken betekent de stabiliteitsvoorwaarde (i.2) in §1.7.3 voor het meerdimensionale geval dat $|1 + h\,\lambda| \leq 1$ moet zijn voor alle eigenwaarden λ van de Jacobi matrix $(\frac{\partial f_j}{\partial u_i}(t, u))$ voor vrijwel alle t (en dat voor alle relevante (t, u)).

Referenties

[1] F. Beukers, Modellen en Computers, College Diktaat, Mathematisch Instituut, Universiteit Utrecht, Utrecht, 1994.

[2] L.F. Shampine Numerical Solution of Ordinary Differential Equations, Chapman & Hal, New York, 1994.

[3] L.F. Shampine and M.W. Reichelt, *The* Matlab *ODE Suite*, SIAM J. Sc. Comput., 18 (1997), pp. 1–22.

[4] *The Student Edition of MATLAB, version 4, User's Guide*, Prentice-Hall, 1995

Hoofdstuk II

Nulpuntsbepaling

door P. Zegeling

1 Inleiding

2.1 Niet-lineaire vergelijkingen

Het vinden van één of meer wortels α van een niet-lineaire vergelijking (of anders gezegd: nulpunten van een niet-lineaire functie)

$$f(x) = 0, \quad x \in \mathbb{R}, \quad \text{of } x \text{ in het definitiegebied van } f \tag{1.1}$$

is een van de meest voorkomende problemen in de toegepaste wiskunde en in vele andere toepassingsgebieden. Indien $x \in \mathbb{R}^n$ en $f : \mathbb{R}^n \to \mathbb{R}^n$ dan hebben we te maken met niet-lineaire stelsels. We zullen ons voornamelijk bezig gaan houden met het geval $n = 1$: een scalaire niet-lineaire vergelijking. Sommige niet-lineaire vergelijkingen hebben een expliciete formule voor de oplossing; de bekendste is natuurlijk de *abc*-formule voor een kwadratische vergelijking. Er bestaan (wat minder overzichtelijke) formules, ofschoon weinig gebruikt in de praktijk, voor 3e-graads- en 4e-graadsvergelijkingen.
Het moge ook duidelijk zijn dat het *aantal* oplossingen van een algemene niet-lineaire vergelijking elke waarde $\in \mathbb{N}$ kan aannemen:

- $\exp(x) + 1 = 0$ heeft geen enkele oplossing
- $\exp(-x) - x = 0$ heeft precies één oplossing
- $x^2 - 4\sin(x) = 0$ heeft twee oplossingen
- $x^3 + 6x^2 + 11x - 6 = 0$ heeft drie oplossingen
- $\sin(x) = 0$ heeft ∞-veel oplossingen

Behalve voor enige speciale situaties, zoals eerder beschreven, zijn i.h.a. geen expliciete oplossingen van (1.1) voorhanden en moeten we tevreden zijn met de mogelijkheid om een numerieke benadering voor de wortel te vinden met een zekere van tevoren gespecificeerde nauwkeurigheid. Deze numerieke benaderingen van de oplossingen worden vaak verkregen met behulp van iteratieve methoden. Met een iteratieve methode wordt uitgaande van een geschatte startwaarde, die natuurlijk bij voorkeur zo dicht mogelijk bij het gezochte nulpunt ligt, een rij getallen x_i gegenereerd, met uiteraard de

bedoeling dat deze rij getallen convergeert naar het gevraagde nulpunt. Iedere volgende iteratiewaarde wordt op basis van zijn voorganger(s) volgens een bepaalde iteratieve formule berekend. In dit hoofdstuk zullen we voornamelijk de situatie beschouwen van één of twee voorgangers, dus $x_{i+1} = \phi(x_i)$, of $= \phi(x_{i-1}, x_i)$ voor een zekere functie ϕ. Als deze rij convergeert, dan bereiken we nooit de exacte waarde ('toevalstreffers' uitgesloten), maar kunnen we deze onder 'gunstige' omstandigheden wel zo dicht mogelijk benaderen. De grootheid $x_i - \alpha$ geeft dan de gemaakte fout aan in de i-de iteratie van het proces.
Bij het gebruik van iteratieve methoden voor nulpunten moet je je direkt de volgende zaken gaan afvragen. Bijvoorbeeld:

Vraag 1A. Wat is, numeriek gezien, een nulpunt eigenlijk? Plot n.a.v. deze vraag de functie $f(x) = (1-x)^6$ eens met Matlab (gebruik hiervoor de file `nulpvb1.m`).
NB. In hoofdstuk II kun je de variabelen en schermpjes 'schoonvegen' m.b.v. de file `schoon.m`; d.w.z. in Matlab: `schoon` ($\hookleftarrow$).

Vraag 1B. Probeer ook eens `nulpvb2.m` voor het plotten van de functie $f(x) = x - 3 + 10^{15}(\tan(\arctan(x)) - x)$. Wat is hier aan de hand?

Andere interessante vragen die we tegen zullen komen zijn bijvoorbeeld: hoeveel nulpunten zijn er (zie de eerdergenoemde voorbeelden), en naar welk van die nulpunten convergeert de iteratieve methode? Convergeert de methode wel en, zo ja, hoe snel? Hoe moet er gestart worden met het proces (startwaarde) en wanneer moet je stoppen (stopcriterium)? En, uiteindelijk, natuurlijk de belangrijkste vraag: hoe nauwkeurig is de berekende oplossing?
Een aantal van deze vragen zien we straks terug bij de experimenten.

Terugverwijzend naar Hoofdstuk I wordt het nut van numerieke methoden voor vergelijkingen als (1.1) op een andere manier duidelijk, indien we bijvoorbeeld Euler-Backward willen toepassen op een niet-lineaire DV.

Opgave 1C. *Schrijf nog eens Euler-Backward uit voor de niet-lineaire DV* (1.3) *uit Hoofdstuk I en zie in dat we een numerieke methode nodig hebben voor het oplossen van een niet-lineaire vergelijking om überhaupt van t_n naar t_{n+1} te kunnen komen.*

Nu volgen eerst enkele voorbeelden van niet-lineaire vergelijkingen en 'praktijkproblemen' die we nog tegen zullen komen.

Voorbeeld 1D.

$$f(x) = x^3 + 3x - 4 = 0. \tag{1.2}$$

Deze vergelijking heeft de unieke wortel $x = 1$. Dit is een simpel testprobleem voor numerieke methoden, waarbij we het gedrag van de fout tijdens het iteratieproces kunnen bijhouden.

Voorbeeld 1E.

$$f(x) = x^3 - 3x + 2 = 0. \tag{1.3}$$

Deze vergelijking lijkt op de voorgaande, maar heeft twee oplossingen $x_1 = -2$ en $x_2 = 1$, waarbij x_2 een zogenaamde 'dubbele' wortel is. Het is op dit moment nog

onduidelijk naar welk van de twee nulpunten een iteratief proces convergeert en welke invloed het speciale karakter van x_2 mogelijk heeft.

Voorbeeld 1F.

$$f_B(x) = x + \exp(-Bx^2)\cos(x) = 0. \tag{1.4}$$

waarbij $B > 0$ een parameter is. Deze vergelijking heeft een unieke oplossing, waarvan de exacte waarde onbekend is. Het ophogen van de parameter B verhoogt de 'moeilijkheidsgraad' van het probleem navenant.

Voorbeeld 1G. Een interessant 'praktijk'probleem voor nulpuntsbepaling treedt op bij het berekenen van annuiteiten: een bedrag P_1 aan € wordt op een rekening gezet aan het begin van de jaren $1, 2, \ldots, N_1$. Het wordt jaarlijks vergroot met een factor r (bijv. $r = 0.05$ betekent een rentevoet van 5%). Aan het begin van de jaren $N_1+1, \ldots, N_1+N_2$ wordt een bedrag van P_2 in € opgenomen van de rekening. Na de laatste transactie staat er precies € 0,– op de rekening. De relatie die dit proces beschrijft wordt gegeven door de volgende vergelijking:

$$P_1[(1+r)^{N_1} - 1] = P_2[1 - (1+r)^{-N_2}]. \tag{1.5}$$

Als gegeven is dat $N_1 = 30$ jaar, $N_2 = 20$ jaar, $P_1 =$ € 2000,– en $P_2 =$ € 8000,–, wat is de rente r dan? Om dit te berekenen moeten we een niet-lineaire vergelijking oplossen.

Voorbeeld 1H. Beschouw het probleem van het richten van een kanon om een doel te raken op afstand d. Het kanon geeft een beginsnelheid V_0 mee aan de kanonskogel onder een hoek ϑ. Dit wordt beschreven door

$$f(\vartheta) = \frac{2V_0^2 \cos(\vartheta)\sin(\vartheta)}{g} - d = 0. \tag{1.6}$$

Gevraagd is de hoek ϑ te bepalen bij gegeven V_0 en d ($g = 9.8\,\mathrm{m/s^2}$).

Voorbeeld 1I. Een pijp ter lengte L heeft een dwarsdoorsnede in de vorm van een halve cirkel met straal r. Als deze gevuld is met water tot een afstand h vanaf de bovenkant gerekend, geldt voor het volume V van het water:

$$V = L(0.5\pi r^2 - r^2 \arcsin(h/r) - h\sqrt{r^2 - h^2}). \tag{1.7}$$

Stel dat $L = 10$ meter, $r = 1$ meter, and $V = 12.4$ kubieke meter. Gevraagd is: bepaal de diepte van het water in de pijp tot op een nauwkeurigheid van 0.01 meter.

Voorbeeld 1J. Een object valt verticaal door de lucht naar beneden en ondervindt zowel wrijving als zwaartekracht. Stel dat men het object met massa m laat vallen van een hoogte S_0 en dat de hoogte van het object na t seconden wordt gegeven door de volgende formule:

$$S(t) = S_0 + \frac{mg}{k}t - \frac{m^2 g}{k^2}(1 - e^{-kt/m}), \tag{1.8}$$

waarbij g de versnelling t.g.v. de zwaartekracht is en k coefficient van de luchtweerstand weergeeft in kg sec/m.
Stel dat $S_0 = 300$ meter, $m = 0.25$ kg en $k = 0.1$ kg sec/m.

De vraag is: bepaal met een nauwkeurigheid van 0.01 sec, de tijdsduur die het object nodig heeft om de grond te raken.

Voorbeeld 1K. De groeifactor in grote populaties kan worden gemodelleerd (over niet al te lange tijdperioden) door aan te nemen dat de populatie continu in de tijd groeit met de factor die evenredig is met het aantal individuen die er op dat moment zijn. Als we met $N(t)$ het aantal individuen op tijdstip t aangeven en met λ de constante geboorte factor van de populatie, dan voldoet de populatie aan de volgende DV

$$\frac{dN(t)}{dt} = \lambda N(t) + v, \tag{1.9}$$

waarbij v de constante immigratie factor beschrijft.
De oplossing van deze DV is

$$N(t) = N_0 e^{\lambda t} + \frac{v}{\lambda}(e^{\lambda t} - 1). \tag{1.10}$$

Stel nu dat een zekere populatie aanvankelijk 1 miljoen individuen telt, dat er, bijv., 435 000 individuen immigreren in het eerste jaar, en dat er 1 564 000 individuen aanwezig zijn aan het einde van het eerste jaar.
De vraag is: wat is de geboortefactor van deze populatie? Hiervoor moet eerst een niet-lineaire vergelijking voor λ uitgewerkt worden!

2.2 Een 'standaardroutine' in Matlab

Vraag 2A. Hoe benadert Matlab nulpunten van een functie en zijn deze uitkomsten betrouwbaar?

Experiment. Probeer beide nulpunten van vergelijking (1.3) (de functie wordt gedefinieerd in `f.m`) te vinden met de Matlab-function `fzero.m` (evt. `help fzero`). Speel met verschillende startwaarden (start ook eens 'zeer dicht' bij de wortel(s)) .

Verklaring. Om één of andere reden kan Matlab het nulpunt x_2 niet vinden! Op dit moment is het onduidelijk waarom. We zullen dus de eigenschappen van numerieke methoden moeten gaan onderzoeken om in te kunnen zien welke criteria van belang zijn.

2.3 De halveringsmethode

Als een expliciete uitdrukking voor een nulpunt α niet bestaat, zou je kunnen zoeken naar een interval, waar de functie in beide eindpunten een tegengesteld teken heeft!

Idee. We kunnen in dit geval gebruik maken van de tussenwaardestelling. Door twee x-waarden x_0 en x_1 te kiezen waarvoor $f(x_0)f(x_1) < 0$, weten we dan dat er in ieder geval één punt $\alpha \in [x_0, x_1]$ moet zijn waarvoor $f(\alpha) = 0$! Op basis van deze gedachte zouden we een iteratieve methode kunnen uitwerken die ons een benadering van een nulpunt van f oplevert.

Vraag 3A. Is het mogelijk met dit idee een algoritme te verzinnen dat convergeert naar een α met $f(\alpha) = 0$?

Experiment. Pas de file `halverng.m` toe op vergelijking (1.2). Kijk ook eens in de file en probeer er achter te komen hoe deze methode werkt.

Verklaring. Deze techniek heet de **halveringsmethode** (of **bisectie**) en convergeert naar de wortel van (1.2).

Opgave 3B. *Beschrijf het algoritme in een zogenaamd 'stroomdiagram', m.a.w. geef een schets van het algoritme in de vorm van een schema of diagram (kijk ook eens in de Matlab-file).*

Vraag 3C. Hoe 'snel' convergeert de halveringsmethode?

Experiment. Herhaal het vorige experiment, maar let nu op het aantal iteraties dat nodig is bij een voorgeschreven 'nauwkeurigheid' (of 'tolerantie').

Verklaring. Bij iedere iteratieslag kom je, door het halveren van het interval, hooguit 2 keer zo dicht bij de exacte oplossing.

Opgave 3D. *Controleer dat*

$$|x_{i+1} - x_i| = \frac{1}{2}|x_i - x_{i-1}| \tag{3.1}$$

en dus dat de onbetrouwbaarheid in slag i voldoet aan:

$$|x_{i+1} - \alpha| \leq |x_{i+1} - x_i| = \frac{|x_1 - x_0|}{2^i}, \quad \textit{voor} \quad i \geq 1. \tag{3.2}$$

D.w.z. de onbetrouwbaarheid wordt per iteratieslag gehalveerd. Vanwege deze eigenschap wordt de halveringsmethode een (gedomineerd) lineair convergent proces genoemd.

Vraag 3E. Aangezien vergelijking (1.3) bijzonder veel op (1.2) lijkt qua moeilijkheidsgraad, zou je ongeveer hetzelfde gedrag van de halveringsmethode verwachten. We gaan proberen het nulpunt $\alpha = 1$ te vinden.

Experiment. Gebruik `halverng.m`. Probeer, indien nodig, meerdere startwaarden (kies als default 'precisie' 0.001 om de verschillende methoden later goed met elkaar te kunnen vergelijken).

Verklaring. De halveringsmethode kan het 'dubbele' nulpunt $\alpha = 1$ niet detecteren, en convergeert dus, of helemaal niet, of naar het andere nulpunt $\alpha = -2$.

We zien dus dat deze methode niet altijd het gewenste resultaat oplevert. Bovendien is de convergentie 'slechts' lineair. Verder is het niet duidelijk hoe we dit principe zouden moeten uitbreiden naar stelsels van niet-lineaire vergelijkingen. Redenen genoeg om wellicht een andere weg in te slaan...

2.4 De methode van successieve substitutie

Idee. We kunnen vergelijking (1.1) herschrijven naar de vorm

$$x = \phi(x) \tag{4.1}$$

om vervolgens via, bijvoorbeeld, het iteratieve proces

$$x_{i+1} = \phi(x_i) \tag{4.2}$$

en een gekozen startwaarde x_0 een nulpunt van f proberen te vinden. Dit principe heeft de naam methode van successieve substitutie en is gebaseerd op de zogenaamde 'vaste-punt-theorie' (vaste punten van ϕ, d.w.z. oplossingen van (4.1), komen dan overeen met wortels van (1.1)). Het moge intuitief al duidelijk zijn, dat niet iedere keuze van ϕ tot een goed resultaat hoeft te leiden, ook al zijn oplossingen van (1.1) per definitie wel vaste punten van $\phi(x)$.

Opgave 4A. *Laat met een simpel voorbeeldje zien dat $f(x) = 0$ altijd te herschrijven is naar $x = \phi(x)$.*

Vraag 4B. Laten we eerst eens bekijken hoe een dergelijk proces werkt op het eerste testprobleem (1.2). Convergeert het, en zo ja, hoe snel, en naar welke wortel?

Opgave 4C. *Check eerst dat de functie $\phi_1(x) = \frac{4}{x^2+3}$ een kandidaat is voor de functie ϕ uit* (4.1).

Experiment. Gebruik de file `xisphix.m` met bijbehorende file `phi.m` (hierin kan de functie $\phi(x)$ desgewenst aangepast worden). Doe enkele experimenten met verschillende startwaarden. 'Meet' ook de convergentiesnelheid; d.w.z. welke (gehele) $p > 0$ voldoet (ongeveer) aan $(x_{i+1} - \alpha)/(x_i - \alpha)^p \approx$ constant? ($\hookrightarrow$ maak zelf een tabelletje bij enkele runs voor een paar voordehand liggende waarden van p; probeer $p = 1$ en $p = 2$ bijv.)? Hoe wordt de 'precisie' hier gemeten? Heb je hiervoor andere (betere?) suggesties?

Verklaring. De functie ϕ_1 heeft blijkbaar zodanige eigenschappen dat er voor iedere startwaarde convergentie optreedt. Overigens is de convergentie lineair (d.w.z. $p = 1$): de convergentieorde is één (vergelijk met de halveringsmethode; wat is het subtiele verschil?).

Opgave 4D. *Laat zien dat de fout voldoet aan de volgende formule:*

$$x_{i+1} - \alpha = \phi'(\xi_i)(x_i - \alpha) \approx \phi'(\alpha)(x_i - \alpha) \quad \textit{voor } i \textit{ 'ver genoeg' (d.w.z. bij convergentie)}$$

voor zekere $\xi_i \in [\alpha, x_i]$. We zien dus dat de afgeleide van ϕ, en i.h.b. de waarde ervan in α, een cruciale rol speelt in dit iteratieve proces. Check dat in de bovenstaande situatie de ϕ' in absolute waarde altijd kleiner is dan $\frac{1}{2} < 1$. Dus?
Misschien hebben we geluk gehad met deze keuze van ϕ. We hadden net zo goed, zonder voorkennis, een andere functie kunnen kiezen!

Opgave 4E. *Laat zien dat vaste punten van de volgende functies ook nulpunten van f uit* (1.2) *zijn:*

$$\phi_2(x) = \frac{4 - x^3}{3},$$

$$\phi_3(x) = \frac{4 - 3x}{x^2},$$

$$\phi_4(x) = x^3 + 4x - 4,$$

$$\phi_5(x) = (4 - 3x)^{1/3},$$

$$\phi_6(x) = \frac{2x^3 + 4}{3x^2 + 3}.$$

Echter, niet alle keuzes zullen even 'goed' blijken te zijn (zie verderop voor een verklaring)!

Vraag 4F. Een 'snellere' convergentie, m.a.w. een hogere waarde van p, heeft het effect dat de fout $x_i - \alpha$ sneller klein wordt, en dus dat we eerder een goede benadering van het nulpunt hebben. Welke functie uit dit rijtje moeten we nemen om (snellere) convergentie te verkrijgen?

Experiment. Doe enkele experimenten met al deze functies om het nulpunt van (1.2) zo snel en nauwkeurig mogelijk te benaderen (pas hiertoe, per geval, de file `phi.m` aan). Kies een functie $\phi(x)$ en een 'tolerantie' en experimenteer met een paar verschillende startwaarden x_0. Ga pas daarna naar de volgende $\phi(x)$. Noteer onderweg alle bijzonderheden die je tegenkomt (convergentie, divergentie, snelheid van convergentie, afhankelijkheid van startwaarde x_0, etcetera).

Verklaring. De afgeleide van ϕ (en speciaal de waarde $\phi'(\alpha)$) is de overheersende factor in alle experimenten. Deze grootheid kan de numerieke resultaten verpesten ('langzamere' convergentie of zelfs divergentie), maar ook aanzienlijk verbeteren ('snellere' convergentie)!

Opgave 4G.

a. *Bereken $\phi'(x)$ en i.h.b. $\phi'(\alpha)$ voor alle gebruikte functies ϕ_j, $j = 1, \ldots, 6$. Wat voor gevolgen heeft dit voor de 'convergentie-snelheid? Teken ook in een $(x, \phi(x))$-diagram het iteratie proces voor minstens twee ϕ_j's.*

b. *Laat zien dat, als $|\phi'(\alpha)| > 1$, er nooit convergentie naar α kan optreden (toevalstreffers uitgesloten).*

Uit het voorafgaande experiment concluderen we dat er keuzes voor ϕ mogelijk zijn, waarvoor de convergentiesnelheid veel groter is dan misschien verwacht!

Vraag 4H. Is dit toeval of zit er een structuur in? Oftewel, kunnen we een dergelijke ϕ ook construeren voor het algemene geval (1.1)?

Opgave 4I. *Ontwikkel een aantal termen van een Taylorreeks voor de functie $\phi(x)$ in x_i rond het nulpunt α. Laat vervolgens zien dat voor de keuze $\phi(x) = x - f(x)/f'(x)$ met $f'(\alpha) \neq 0$ (welke ϕ_j uit het voorbeeld voldoet hieraan?) geldt: $\phi'(\alpha) = 0$ en dus:*

$$x_{i+1} - \alpha = \frac{\phi''(\xi_i)}{2}(x_i - \alpha)^2. \tag{4.3}$$

Geef een uitdrukking voor $\phi''(\xi_i)$ in termen van f. De fout voor de onderliggende iteratiemethode neemt dan, in principe (wanneer misschien niet bijvoorbeeld?), **kwadratisch** *af per iteratieslag. Een nadere analyse van deze speciale methode is dan ook gewenst.*

Opgave 4J. *Stel dat de iteratieve methode $x_{i+1} = \phi(x_i)$ lineair convergeert naar een oplossing α. Laat zien dat de iteratie*

$$x_{i+1} = \Phi(x_i), \qquad \textit{met} \quad \Phi(x) := \frac{x\ \phi(\phi(x)) - [\phi(x)]^2}{\phi(\phi(x)) - 2\phi(x) + x}$$

dan kwadratisch convergeert naar α.

2.5 Newton-Raphson

De methode van **Newton**–**Raphson** (NR) voor (1.1) wordt gegeven door de iteratie (zie de voorlaatste opgave van het vorige hoofdstuk):

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}, \quad i \geq 0, \qquad x_0 \text{ te kiezen.} \tag{5.1}$$

Vraag 5A. Uit (4.3) kunnen we afleiden dat NR alleen kwadratisch convergeert voor $i \to \infty$, indien $f'(\alpha) \neq 0$. Kunnen we NR ook op een andere manier (dus niet in termen

van een functie ϕ), bijvoorbeeld grafisch, interpreteren? Het gebruik van de afgeleide van f suggereert een relatie met de raaklijn in verschillende x-waarden.

Experiment. Gebruik de file `newton.m` voor een zelfgekozen f (bijv. $\sin^2(\pi x)$ met $x_0 \in (0, 1/4)$ en een precisie van 10^{-4}; zie de files `f.m` en `facc.m`) om enkele iteratieslagen van NR te zien. Desgewenst kan m.b.v. `graffun.m` en `graffacc.m` de grafiek van $f(x)$, respectievelijk $f'(x)$, getekend worden.

Verklaring. In iedere iteratieslag maakt Newton-Raphson gebruik van de raaklijn aan de grafiek van f in de oude x-waarde om het volgende iteratiepunt te bereiken.

Opgave 5B. *Laat zien dat, om van x_0 naar x_1 te komen, we de raaklijn in x_0 kunnen nemen, en dat vervolgens het nulpunt van deze lineaire functie gelijk is aan de x_1 uit NR. Door vervolgens verder te gaan met x_1 als nieuwe 'startwaarde' en het proces te herhalen krijgen we het iteratieve proces horend bij NR. Ga dit na.*

Vraag 5C. In praktische situaties kennen we het nulpunt α natuurlijk niet, maar zou het toch prettig zijn om te weten hoever we ongeveer van α af zitten tijdens het proces. Zouden we, zonder het nulpunt α te kennen, een schatting kunnen geven van de onderweg gemaakte fout? We hebben immers al in een eerdere opgave gezien dat voor NR geldt:

$$\frac{x_{i+1} - \alpha}{(x_i - \alpha)^2} \approx \frac{f''(\alpha)}{2f'(\alpha)} \equiv C \qquad \text{voor } i \text{ groot genoeg}$$

Derhalve zouden we de constante C kunnen schatten met $\frac{f''(x_{i+1})}{2f'(x_{i+1})} \approx \frac{f''(\alpha)}{2f'(\alpha)}$ en via

$$x_{i+1} - \alpha \quad (= \text{de fout in slag } i+1) \approx C\ (x_i - \alpha)^2 \approx C\ (x_i - x_{i+1})^2, \tag{5.2}$$

waarbij x_i en x_{i+1} berekend zijn met NR, een indruk kunnen krijgen van de fout.

Experiment. Doe enkele experimenten met `newton.m` toegepast op (1.2). Vergelijk de geschatte fout uit formule (5.2) met de werkelijke fout $x_{i+1} - \alpha$; NB. de functie $f''/(2f')$ staat in de functionfile `f2f.m`.

Verklaring. De boven gegeven schatting voor de fout in NR geldt inderdaad (voor i groot genoeg); let wel: voor deze vergelijking met $f'(\alpha) \neq 0$.

Vraag 5D. In het NR-proces komt de afgeleide van f in de noemer voor. Wat gebeurt er als $f'(\alpha) = 0$ (immers, teller en noemer zijn dan beide gelijk aan nul; en tijdens de berekeningen kan derhalve $f'(x_i) \approx f'(\alpha)$ voor i groot genoeg, en dus delen we dan door een zeer klein getal...)? Hebben we dan nog steeds convergentie, laat staan kwadratische convergentie?

Experiment. Pas `newton.m` toe op vergelijking (1.3) en probeer het nulpunt $\alpha = 1$ (met $f'(1) = 0$) te benaderen. Noteer per iteratieslag de grootheid $(x_{i+1} - \alpha)/(x_i - \alpha)^p$ voor $p = 1$ en $p = 2$.

Verklaring. Voor $f'(\alpha) = 0$ wordt NR een lineair convergent proces...!

Opgave 5E. *Laat zien dat voor het geval $f(\alpha) = f'(\alpha) = 0$, $f''(\alpha) \neq 0$ geldt:*

$$\lim_{i \to \infty} \frac{x_{i+1} - \alpha}{x_i - \alpha} = \frac{1}{2}. \tag{5.3}$$

Vraag 5F. Zou in speciale situaties het NR-proces i.p.v. een langzamere (lineaire) juist een nog snellere (sneller dan kwadratische) convergentie kunnen opleveren?

Experiment. Pas `newton.m` toe op $f(x) = \tan(x-1) = 0$ en onderzoek het gedrag van de fout in de buurt van $\alpha = 1$. Is het proces nu lineair of kwadratisch convergent? Plot deze functie en probeer er achter te komen wat er rond $x = 1$ aan de hand is.

Verklaring. In $x = 1$ heeft deze speciale f een buigpunt, terwijl de eerste afgeleide daar niet gelijk aan nul is. Deze eigenschap versnelt het NR-proces blijkbaar.

Opgave 5G. *Laat zien dat geldt: als $f(\alpha) = f''(\alpha) = 0$ en $f'(\alpha) \neq 0$, dan convergeert Newton-Raphson minstens* **kubisch** *($\Longrightarrow$ convergentie-orde 3) naar het nulpunt α.*
(hint: werk o.a. $\phi''(\alpha)$ en $\phi'''(\alpha)$ uit voor $\phi(x) \equiv x - f(x)/f'(x)$.)

Vraag 5H. In de vorige experimenten hebben we de invloed van eigenschappen van de functie f zelf op het convergentie-gedrag van NR onderzocht, wanneer we dicht genoeg bij het nulpunt α zitten. Maar, hoe komen we dichtbij het nulpunt zonder kennis vooraf? Oftewel, welke invloed heeft de startwaarde x_0 op het iteratieproces?

Opgave 5I. *Onderzoek de functie f uit* (1.4) *en ga na wat de invloed is van de parameter B op de 'vorm' van de grafiek van f en dus op NR (denk aan het raaklijnen-verhaal uit een eerdere opgave).*

Experiment. Gebruik `newton.m` om het nulpunt van (1.4) te vinden voor verschillende startwaarden (dichtbij α, ver weg van α en probeer ook eens $x_0 = 0$). Doe dit voor enkele waarden van B (bijv. $B = 1, 5, 10, 25, 50$).

Verklaring. Numerieke methoden voor het berekenen van nulpunten zijn i.h.a. zeer gevoelig voor de keuze van de startwaarde.

Opgave 5J. *Laat zien dat voor NR geldt: als $f(\alpha) = 0$ voor $f \in \mathcal{C}^2$, $f(x) \neq 0$ en $f''(x)f(x) \geq 0 \quad \forall x \in (\alpha, b]$, dan convergeert NR monotoon naar α* voor iedere startwaarde $x_0 \in (\alpha, b]$. (NB. dit resultaat geldt natuurlijk voor een interval $[a, \alpha)$ 'links' van het nulpunt)

Opgave 5K. *Er kan bewezen worden dat voor het vinden van het nulpunt $\alpha = 0$ van de functie $f(x) = \arctan(x)$ ($x \in \mathbb{R}$) m.b.v. NR het geen zin heeft om startwaarden te kiezen die voldoen aan de ongelijkheid*

$$\arctan(|x_0|) > \frac{2|x_0|}{1 + x_0^2}.$$

Vind een concrete startwaarde x_0 die hieraan voldoet en bewijs dat in dat geval de NR-rij $(x_i)_{i \geq 0}$ divergeert, en i.h.b. $\lim_{i \to \infty} |x_i| = \infty$.

Opgave 5L. *Beschouw de vergelijking $x^2 \ln x = a$.*
Maak eerst een grafiek van $g(x) = x^2 \ln x$, op het interval $(0, 2)$.

a. *Toon aan, dat voor iedere $a \geq 0$ de vergelijking precies één positieve oplossing, zeg $\alpha(a)$, heeft, en voor $a \in (-1/(2e), 0)$ de vergelijking twee oplossingen heeft, zeg $\beta(a) \leq \alpha(a)$.*

b. *Toon aan, dat voor $a > 0$, $1 < \alpha(a) < 1 + a$ terwijl het NR-proces, toegepast op $g - a$ gestart in $x_0 = 1$, op de eerste stap na, monotoon naar $\alpha(a)$ convergeert en wel kwadratisch.*

c. *Wat kun je zeggen wat betreft convergentie en monotonie van de NR-iteranden, als $a \in (-\frac{1}{2e}, 0)$ en $x_0 = 1$?*

d. *Zij nu $a = -\frac{1}{2e}$. Hoe is het convergentie gedrag van NR gestart in $x_0 = 1$?*

e. *Waar moet NR gestart worden, opdat gegarandeerd monotone convergentie naar $\beta(a)$ optreedt, als $a \in [-\frac{1}{2e}, 0) \setminus \{-\frac{3}{2e^3}\}$? Toon de juistheid van je bewering aan.*

f. *Laat nu* $a = -\frac{3}{2e^3}$. *Dan is* $\beta(a) = e^{-3/2}$ *en* $g''(\beta(a)) = 0$. *Op een onmiddellijke omgeving van deze wortel geldt nu* $ff'' < 0$, *met* $f(x) = g(x) - a$. *Laat zien, dat er van monotone convergentie bij het NR-proces geen sprake kan zijn, wel van alterneren om de wortel, op den duur. Zie dit ook grafisch in.*

Opgave 5M. *Generaliseer één van de argumenten die tot NR hebben geleid zodanig, dat dit de kubisch convergente methode*

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} - \frac{1}{2}\frac{[f(x_i)]^2 f''(x_i)}{[f'(x_i)]^3}$$

oplevert.

Opgave 5N. *Stel dat de functie* $f(x)$ *een nulpunt* α *heeft met multipliciteit* q, *i.e.,*

$$f(\alpha) = f'(\alpha) = ... = f^{(q-1)}(\alpha) = 0, \quad f^{(q)}(\alpha) \neq 0.$$

We hebben eerder gezien dat NR hiermee zijn kwadratische convergentie zou kunnen verliezen. Echter, beschouw het geval $q = 2$. *Laat zien dat de volgende, ietwat aangepaste, versie van NR toch kwadratisch convergeert naar* α*:*

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

(extra: je kunt eventueel algemener laten zien dat voor een wortel met multipliciteit q *geldt dat de iteratie* $x_{i+1} = x_i - \frac{qf(x_i)}{f'(x_i)}$ *kwadratisch naar* α *convergeert.)*

Vraag 5O. Zou het mogelijk zijn om een methode te bedenken waarbij we de afgeleide f' niet nodig hebben en toch de voordelen van NR, zoals het meer dan lineair convergente karakter van het proces, gedeeltelijk kunnen behouden?

2.6 Koorden-Newton

Idee. Om de laatste vraag te beantwoorden zouden we bijvoorbeeld de afgeleide $f'(x_i)$ in NR kunnen benaderen door een differentiequotiënt.

Opgave 6A. *Laat m.b.v. Taylorontwikkelingen zien dat*

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} + E. \tag{6.1}$$

Bepaal dus ook de foutterm E.

Vraag 6B. Indien we $f'(x_i)$ in NR vervangen door $(f(x_i)-f(x_{i-1}))/(x_i-x_{i-1})$ krijgen we de zogenaamde **Koorden**–**Newton** methode (KN). Wat gebeurt er met het i.h.a. kwadratisch convergente karakter van de methode? Treedt er dan nog wel convergentie op naar het nulpunt?

Experiment. Probeer met de file `koordnwt.m` toegepast op (1.2) de convergentie-orde p te schatten in $(x_{i+1}-\alpha)/(x_i-\alpha)^p$. Merk op dat we nu 2 startwaarden x_0 en x_1 nodig hebben!

Verklaring. De methode KN verliest het kwadratisch convergente karakter van NR door de extra benadering die we maken, maar is nog steeds 'sneller dan' lineair convergent!

Opgave 6C. *Verklaar de naam Koorden-Newton met een plaatje.*
NB. de orde van convergentie van KN blijkt $\frac{1}{2}(1+\sqrt{5})$ *te zijn. Voor een bewijs m.b.v. Fibonaccigetallen verwijzen we naar de literatuur (zie referenties achterin).*

2.7 Toepassingen

Onderstaande 'praktijkproblemen' gaan we met numerieke methoden oplossen. Bij iedere opdracht is het niet alleen de bedoeling dat er simpelweg de oplossing van het probleem wordt bepaald, maar ook dat er enig inzicht wordt getoond o.a. in het gedrag van de fout, de convergentiesnelheid van de gebruikte methode, en, indien mogelijk, een vergelijking met een andere methode in termen van 'kosten' (hoeveel goedkoper of duurder?) en 'efficiëntie' (hoeveel sneller of langzamer?).

Opdracht 7A. Bereken de oplossing van het beschreven model uit Voorbeeld1G m.b.v. Koorden-Newton of Newton-Raphson.

Opdracht 7B. Bepaal de diepte van het water in de pijp uit Voorbeeld1I. Kies zelf een 'geschikte' methode.

Opdracht 7C. Bereken voor een aantal zelfgekozen waarden van V_0 en d de benodigde hoek om d te bereiken (zie Voorbeeld1H). Houd hierbij rekening met het feit dat niet iedere combinatie een (unieke) oplossing heeft!

Opdracht 7D. Bedenk zelf een functie ϕ voor de methode van successieve substitutie om het probleem uit Voorbeeld1J op te lossen (en los het probleem ook daadwerkelijk op).

Opdracht 7E. Gebruik Newton-Raphson voor het bepalen van de geboortefactor uit Voorbeeld1K met de aldaar gegeven parameters.

Opdracht 7F. Gegeven is de matrix

$$A = \begin{pmatrix} 10 & -1 & 0 \\ -1 & 2 & 2 \\ 0 & 2 & 3 \end{pmatrix}$$

Benader met Newton-Raphson de eigenwaarde λ_1 van A die op het interval $[9, 11]$ ligt (tot op 3 correcte decimalen).
Bepaal daarna, *zonder NR te gebruiken*, ook de twee overige eigenwaarden λ_2 en λ_3 van A (hint: gebruik eigenschappen van het spoor, $\mathrm{sp}(A)$, en de determinant, $\det(A)$, in relatie tot de eigenwaarden).
Controleer je antwoorden met die van Matlab, maar dan berekend met de Matlab-function `eig.m` (evt. `help eig`).

2.8 Extra opgaven

Opdracht 8A. In de 'prehistorie' van het computertijdperk konden computers alleen maar optellen, aftrekken en vermenigvuldigen. Voor een deling als $\frac{1}{a}$ moest men gebruik maken van een iteratieve methode. Pas nu NR toe om een iteratie te construeren voor het berekenen van $\frac{1}{a}$, $a \neq 0$, *zonder* een deling te gebruiken. Voor welke startwaarden x_0 convergeert deze methode?

Opdracht 8B. Pas eenzelfde 'truc' als in de vorige opgave toe voor het berekenen van $y = \frac{1}{\sqrt{a}}$, $a > 0$, zonder dat gebruik wordt gemaakt van de bewerking $\sqrt{}$ en delingen.

Opdracht 8C. Tracht met Newton-Raphson een nulpunt te vinden van de functie $f(x) = c\ \mathrm{sign}(x)\ \sqrt{|x|}$. Wat is hier aan de hand?

Opdracht 8D. Het Gerlach-schema voor het benaderen van $\sqrt{A}$, $A > 0$, is gegeven door

$$x_{i+1} = x_i - \frac{(x_i^2 - A)(3x_i^2 + A)(3x_i^6 + 27Ax_i^4 + 33A^2x_i^2 + A^3)}{2x_i(x_i^4 + 10Ax_i^2 + 5A^2)(5x_i^4 + 10Ax_i^2 + A^2)}, \quad i \geq 0$$

met startwaarde $x_0 = \frac{A}{2}$. Welke orde van convergentie heeft deze methode? (schrijf een Matlab-file `gerlach.m`, en test de methode voor enkele waarden van A; hoeveel extra correcte cijfers zie je na elke iteratieslag?)

Opdracht 8E. Het vinden van een juiste startwaarde voor Newton-Raphson is i.h.a. niet gemakkelijk. Stel we willen met NR een nulpunt vinden voor een 'ingewikkelde' functie $h(x)$, en we weten de oplossing, noem deze alvast x_0, van een 'simpeler' probleem $g(x) = 0$. De methode van Davidenko maakt handig gebruikt van deze kennis door het volgende model te beschouwen

$$(1 - \vartheta)\ g(x) + \vartheta\ h(x) = 0,$$

met een (continuerings-)parameter $\vartheta \in [0, 1]$. Voor $\vartheta = 0$, respectievelijk $\vartheta = 1$, zien we de afzonderlijke vergelijkingen $g(x) = 0$ en $f(x) = 0$ terug. Het algoritme van de methode van Davidenko luidt dan: start met $\vartheta = 0$, dit geeft x_0 als oplossing. Deel nu het interval $\vartheta = [0, 1]$ op in m stukken ter grootte $\Delta\vartheta = 1/m$. Bepaal met NR het nulpunt van $\Delta\vartheta h(x) + (1 - \Delta\vartheta)g(x) = 0$ met startwaarde x_0. Dit levert $x^{(1)}$. Ga verder met $\vartheta = 2\Delta\vartheta$, pas NR toe met startwaarde $x^{(1)}$, etcetera t/m $\vartheta = m\Delta\vartheta = 1$ met startwaarde $x^{(m-1)}$. Uiteindelijk geeft dit een benadering $x^{(m)}$ voor het probleem $h(x) = 0$.
Implementeer deze methode in Matlab (noem de file `davidenko.m`) en pas deze toe op (1.4), bijvoorbeeld voor $h(x) = f_{100}(x)$ en $g(x) = f_{10}(x)$.

Opdracht 8F. Beschouw voor *complexe* z de vergelijking $f(z) = z^3 - 1$. Deze vergelijking heeft drie oplossingen in het complexe vlak, n.l. $z^{(1)} = 1$, $z^{(2)} = \frac{1}{2}(-1 + i\sqrt{3})$ en $z^{(3)} = \frac{1}{2}(-1 - i\sqrt{3})$. De NR-methode luidt dan $z_{i+1} = z_i - \frac{f(z_i)}{f'(z_i)}$. Kies een startwaarde $z_0 = x + iy$ en laat NR hierop los. Geef, afhankelijk van naar welk van de drie punten de methode convergeert, een kleur aan het startpunt z_0: *rood* $\leftrightarrow z^{(1)}$, *blauw* $\leftrightarrow z^{(2)}$, en *groen* $\leftrightarrow z^{(3)}$. Doe dit voor meerdere startpunten z_0 in het complexe vlak, bijv. voor alle punten $z_0 := (x_m, y_n)_{m=1,...,M}^{n=1,...,N}$ met N en M *niet te klein* gekozen. Uiteindelijk levert de figuur een zogenaamde 'Newton-fractal' op. De opdracht is: schrijf een Matlab-code `newtonc.m` die de NR-iteraties uitvoert voor de gehele set van beginwaarden en die de fractal voor je plot.

Opdracht 8G. Vind alle nulpunten van de functie

$$f(x) = 720x^6 - 1764x^5 + 1624x^4 - 735x^3 + 175x^2 - 21x + 1.$$

2.9 Stelsels van niet-lineaire vergelijkingen

Opdracht 9A. In deze sectie zijn we geïnteresseerd in het oplossen van stelsels van niet-lineaire functies $f_j(x_1, x_2, ..., x_n) = 0$ voor $j = 1, ..., n$. Maak gebruik van de

literatuur uit de bibliotheek om numerieke methoden te beschrijven die dit probleem kunnen oplossen: hoe zou je bijvoorbeeld NR of successieve substitutie op dit soort stelsels toepassen?
Beschouw nu, voor het geval $n = 2$, het stelsel $f_1(x, y) := x^2 - y = 0$, $f_2(x, y) := y^2 - x = 0$. Teken deze situatie in $\mathbb{R}^2$. Welke nulpunten heeft dit stelsel? (hoe zie je dit in je tekening?) Pas NR toe op dit stelsel met een zelf gekozen startwaarde (x_0, y_0). Maak een tabel met benaderingen als functie van de iteratieslag i. Teken de convergentie of divergentie in je 2d-figuur en geef commentaar.

Opdracht 9B. Voor $n = 2$ willen we alle reële nulpunten gaan bepalen van het stelsel

$$f_1(x, y) := 2x^2 - xy - 5x + 1 = 0, \quad f_2(x, y) := x + 3\ln|x| - y^2 = 0.$$

Maak een schets in $\mathbb{R}^2$ van deze situatie. Hoeveel nulpunten zijn er eigenlijk? Pas nu zowel NR als de methode van successieve substitutie toe op dit stelsel. Gebruik in dat laatste geval, bijvoorbeeld

$$x = \phi_1(x, y) := \sqrt{\frac{x(y+5)-1}{2}}, \quad y = \phi_2(x, y) := \sqrt{x + 3\ln|x|}.$$

Kies zelf geschikte startwaarden om dit probleem op te lossen. Beschrijf je numerieke testwerk en trek conclusies.

Opdracht 9C. Benader met NR de nulpunten van de volgende stelsels (teken bovendien in een figuur de nul-contourlijnen van f_1 en f_2, d.w.z. de krommen $f_1 = 0$ en $f_2 = 0$):

1. $f_1(x, y) := \sin(x - y) = 0, \quad f_2(x, y) := \cos(x + y) = 0$
2. $f_1(x, y) := \frac{\sin(\sqrt{2x^2+y^2})}{\sqrt{x^2+2y^2}} = 0, \quad f_2(x, y) := \ln(\frac{x^2+y^4}{8}) = 0.$

Opdracht 9D. Bekijk het stelsel $y + x^{10}y^3 = 1 - x, \quad ye^y = x + x^3$.

1. Laat zien dat er een unieke oplossing bestaat in $(x, y) \in [0, 1]^2$.
2. Benader de oplossing numeriek met NR.

Opdracht 9E. Bepaal met NR de snijpunten van de cirkel $x^2 + y^2 - 1 = 0$ en de parabool $x^2 - y = 0$.

Opdracht 9F. Beschouw de ellips met excentriciteit 0.5 gegeven door de vergelijking

$$x^2 + 4y^2/3 = 1.$$

We willen in deze opgave onderzoeken of een cirkel met hetzelfde middelpunt en dezelfde oppervlakte snijpunten heeft met de ellips. Laat eerst zien dat mogelijke snijpunten moeten voldoen aan het niet-lineaire stelsel

$$3x^2 + 4y^2 - 3 = 0, \quad x^2 + y^2 - \frac{\sqrt{3}}{2} = 0.$$

Bepaal met NR de potentiele snijpunten van deze twee krommen.

Opdracht 9G. Bepaal met de methode van successieve substitutie een nulpunt van het stelsel $f_1(x,y) := x^3 + 10x - y - 5 = 0, \quad f_2(x,y) := x + y^3 - 10y + 1 = 0$.

Opdracht 9H. Het niet-lineaire systeem

$$x^2 + y^2 - 4 = 0, \quad e^x + y - 1 = 0$$

heeft twee reële oplossingen. Teken beide krommen en noteer bij benadering de ligging van de oplossingen. Pas NR toe achtereenvolgens startend met de beginwaarden $(1,1)$, $(1,-1)$ en $(0,0)$ en onderzoek de convergentie van de iteratie (gebruik in Matlab `format long`).

Opdracht 9I. Construeer zelf (**!**) een stelsel van 3 niet-lineaire vergelijkingen met 3 onbekenden (x,y,z) (d.w.z. $n = 3$), waarvan je de exacte oplossing (x^*, y^*, z^*) kent. Pas NR toe om dat nulpunt numeriek te bepalen. Hoe zou je NR voor $n \gg 1$ in de praktijk toepassen? Leg uit wat in dat geval de extra (numerieke) complicaties zijn, t.o.v. kleinere waarden van n (denk hierbij aan de behandeling van de inverse van de Jacobiaan van de vectorfunctie $\vec{f}$ die voorkomt in NR).

2.10 Samenvatting

2.10.1 De halveringsmethode

• De halveringsmethode, ook wel bisectiemethode genoemd, is gebaseerd op de tussenwaardestelling:

Stelling 2.10.1 *Als $f(x)$ een continue functie is op het interval $[a,b]$ en $f(a) \neq f(b)$, dan geldt voor elk getal y tussen $f(a)$ en $f(b)$, dat er een getal x bestaat met $x \in [a,b]$ waarvoor $f(x) = y$ (lees in ons geval: $y = 0$).*

Zoek eerst een interval $[a,b]$ waarvoor $f(a)$ en $f(b)$ tegengesteld teken hebben. Neem het middelpunt m van $[a,b]$ en controleer of $f(m)$ hetzelfde teken heeft als $f(a)$ of $f(b)$. In het eerste geval ga je verder met het gehalveerde interval $[m,b]$, in het tweede met $[a,m]$. Hierna neem je het middelpunt van het volgende (kleinere) interval en je herhaalt bovenstaande procedure. Dit levert je het bisectie-algoritme op voor het bepalen van nulpunten.
Noem de startwaarden x_0 en x_1. Het voordeel van deze methode is dat, indien $f(x_0)f(x_1) < 0$, er altijd convergentie naar een nulpunt van f plaatsvindt. De vraag is alleen, in het geval van meerdere nulpunten, welk nulpunt uiteindelijk berekend wordt...
Verder hoeft 'de fout' niet monotoon naar nul te gaan. Oftewel, zonder stopcriterium zou bijv. iteratieslag 12 een betere benadering kunnen geven dan iteratieslag 14! De methode convergeert 'lineair' naar een nulpunt van f. Het is helaas niet mogelijk om de halveringsmethode uit te breiden naar stelsels van niet-lineaire vergelijkingen.

2.10.2 De methode van successieve substitutie

Een andere techniek is gebaseerd op het volgende principe:
Herschrijf $f(x) = 0$ naar $x = \phi(x)$ (dit is altijd mogelijk; vaak bestaan er zelfs meerdere mogelijkheden!). Oplossingen van $x = \phi(x)$ worden 'vaste punten' van de functie ϕ genoemd. Een iteratieve methode voor het bepalen van een vast punt van ϕ, en dus een nulpunt van f, wordt gegeven door

$$x_{i+1} = \phi(x_i), \quad i = 0, 1, \ldots \qquad x_0 \text{ te kiezen.} \tag{10.1}$$

Het is mogelijk om enkele nuttige lemma's en stellingen over deze methode te formuleren:

Lemma 2.10.2 *Als $\phi(x)$ continu is op $[a,b]$ en $\phi(x) \in [a,b]$ voor $x \in [a,b]$, dan heeft de vergelijking $x = \phi(x)$ minstens één oplossing $\alpha \in [a,b]$.*

Bewijs. m.b.v. de tussenwaardestelling. □

Lemma 2.10.3 *Als $\phi(x)$ continu is op $[a,b]$ en $\phi(x) \in [a,b]$ voor $x \in [a,b]$, en er bestaat een constante k*

$$0 < k < 1 \quad met \quad |\phi(x) - \phi(y)| \leq k\,|x - y| \quad \forall x, y \in [a,b],$$

dan heeft de vergelijking $x = \phi(x)$ precies één oplossing $\alpha \in [a,b]$ en de rij $x_{i+1} = \phi(x_i) \quad (i \geq 0)$ convergeert voor iedere startwaarde $x_0 \in [a,b]$ naar α.

Bewijs. Stel dat ook $\beta \neq \alpha$ een oplossing is van $x = \phi(x)$. Er volgt een tegenspraak vanwege:

$$|\alpha - \beta| = |\phi(\alpha) - \phi(\beta)| \leq k|\alpha - \beta|.$$

Convergentie volgt door diezelfde relatie meerdere malen toe te passen (via volledige inductie). □

Lemma 2.10.4 *Als $\phi(x) \in C^1[a,b]$, dan geldt met $K \equiv max_{x\in[a,b]}|\phi'(x)|$ en gebruikmakend van de middelwaardestelling: $|\phi(x) - \phi(y)| \leq K|x - y|, \quad \forall x, y \in [a,b]$. Indien nu deze $K < 1$, dan kunnen de resultaten uit Lemma 2.10.3 toegepast worden.*

Een stelling over 'lokale' convergentie:

Stelling 2.10.5 *Als $\phi'(\alpha)$ bestaat en $|\phi'(\alpha)| < 1$, dan is er een $\delta > 0$ zodanig, dat voor elke startwaarde x_0 met $|x_0 - \alpha| < \delta$, (x_0 "ligt voldoende dicht" bij α) geldt:*

$$\lim_{i\to\infty} x_i = \alpha \quad en \quad \lim_{i\to\infty} \frac{x_{i+1} - \alpha}{x_i - \alpha} = \phi'(\alpha).$$

Bewijs. Opgave of zie literatuur. □

Een stelling over 'globale' convergentie:

Stelling 2.10.6 *Als $\phi(x) \in C^1[a,b]$, $\phi(x) \in [a,b]$ voor $x \in [a,b]$ en*

$$K \equiv \max_{x\in[a,b]} |\phi'(x)| < 1,$$

dan geldt:

i) *$x = \phi(x)$ heeft een unieke oplossing $\alpha \in [a,b]$*

ii) *$\forall x_0 \in [a,b]$ convergeert het proces $x_{i+1} = \phi(x_i)$ naar α, d.w.z. $\lim_{i\to\infty} x_i = \alpha$.*

iii) *een afschatting voor de fout: $|x_{i+1} - \alpha| \leq K^i|x_0 - \alpha|$*

iv) $\lim_{i\to\infty} \frac{x_{i+1} - \alpha}{x_i - \alpha} = \phi'(\alpha)$.

NB. Als $\phi'(\alpha) \neq 0$ en $K < 1$, dan noemen we het proces **lineair convergent** en $\phi'(\alpha)$ is de zogenaamde **asymptotische convergentiefactor**.

'Divergentie'. *Als $|\phi'(\alpha)| > 1$, dan zal het proces $x_{i+1} = \phi(x_i)$ nooit naar de oplossing α kunnen convergeren, tenzij x_i voor een of andere i per ongeluk gelijk is aan α.*

Een stelling over 'snellere' convergentie (van orde p):

Stelling 2.10.7 *Als 'x_0 voldoende dicht bij α' gekozen wordt, $\phi(x)$ p-keer continu differentieerbaar is $\forall x$ rond α (voor zekere $p \geq 2$), en er geldt $\phi'(\alpha) = \phi''(\alpha) = \ldots = \phi^{(p-1)}(\alpha) = 0$ met $\phi^{(p)}(\alpha) \neq 0$, dan heeft het iteratieproces $x_{i+1} = \phi(x_i) \quad (i \geq 0)$ orde van convergentie p en er geldt:*

$$\lim_{i \to \infty} \frac{x_{i+1} - \alpha}{(x_i - \alpha)^p} = \frac{\phi^{(p)}(\alpha)}{p!}.$$

Voor $p = 1$, $p = 2$, $p = 3$ heet het proces respectievelijk **lineair**, **kwadratisch** en **kubisch** convergent.
De analyse van de methode van successieve substitutie hangt sterk samen met de theorie van 'vaste-punten'. In een vector-vorm kan deze methode ook voor stelsels van niet-lineaire vergelijkingen geformuleerd worden. Bij een 'slimme' keuze van de afbeelding ϕ kan snellere convergentie (zie bovengenoemde stelling) bewerkstelligd worden. *Echter*, het is niet altijd eenvoudig om de 'juiste' ϕ te bepalen.

2.10.3 Newton-Raphson

Voor de speciale keuze $\phi(x) = x - \frac{f(x)}{f'(x)}$ in de methode van successieve substitutie krijgt men de methode van **Newton**–**Raphson**.
Als $f'(\alpha) \neq 0$ en $f''(\alpha) \neq 0$, dan heeft dit proces convergentie-orde 2 met asymptotische convergentiefactor $\frac{1}{2}\frac{f''(\alpha)}{f'(\alpha)}$.
Een stelling over 'globale' en monotone convergentie van Newton-Raphson:

Stelling 2.10.8 *Als $f \in C^2$, $f(\alpha) = 0$, $f(x) \neq 0$ en $f''(x)f(x) \geq 0 \quad \forall x \in (\alpha, b]$, dan convergeert Newton-Raphson* monotoon *naar α voor iedere startwaarde $x_0 \in (\alpha, b]$.*

NB. een dergelijk resultaat geldt natuurlijk ook voor het interval $[a, \alpha)$.

Speciale situaties: voor $f'(\alpha) = 0$ en $f''(\alpha) \neq 0$ is NR lineair convergent, terwijl voor $f'(\alpha) \neq 0$, $f''(\alpha) = 0$ en $f'''(\alpha) \neq 0$ NR kubisch convergent is.

Deze methode (en varianten ervan) is in de praktijk de meest populaire methode voor het oplossen van (stelsels) niet-lineaire vergelijkingen. Een nadeel is echter het berekenen van f' en de Jacobiaan in het geval van stelsels.

2.10.4 Koorden-Newton

Deze methode is gebaseerd op Newton-Raphson en benadert de daarin voorkomende afgeleide f' alsvolgt:

$$f'(x_i) \approx \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}. \tag{10.2}$$

Door deze extra benadering verliest Koorden-Newton het kwadratisch-convergente karakter van Newton-Raphson. De orde van convergentie wordt (afleiding via Fibonacci-getallen, zie literatuur): $p = \frac{1}{2}(1 + \sqrt{5})$, dus nog steeds 'sneller' dan lineair convergent.

Voor deze methode hebben we geen f' nodig tijdens de iteraties, maar wel twee startwaarden. Het is onduidelijk hoe Koorden-Newton moet worden toegepast op stelsels van niet-lineaire vergelijkingen (dit kan slechts voor zeer speciale gevallen). Bovenstaande stelling 2.8 over globale convergentie voor NR geldt ook voor Koorden-Newton.

2.10.5 Slotopmerkingen

Voor alle behandelde methoden geldt: het kiezen van de juiste startwaarde(n) is essentieel voor de convergentie en nauwkeurigheid van het benaderingsproces. Verder speelt het stopcriterium in alle gevallen een belangrijke rol; bij een verkeerde interpretatie van het numerieke resultaat kunnen onjuiste conclusies getrokken worden over de nauwkeurigheid van de oplossing.

Opgave 10A. *(terugkoppeling naar Euler-Backward)*
Beschrijf het toepassen van Newton-Raphson en die van de methode van successieve substitutie op de niet-lineaire DV (1.3) *uit Hoofdstuk I. Bespreek de voor- en nadelen van beide methoden. Wanneer kunnen we beter Euler-Forward gebruiken, i.p.v. Euler-Backward gecombineerd met een nulpuntsmethode?*

Referenties

[1] K.E. Atkinson, *An Introduction to Numerical Analysis*, John Wiley & Sons, Inc., 1978.

[2] G.J. Borse, *Numerical Methods with Matlab, A Resource for Scientists and Engineers*, PWS Publishing Company, Boston, 1997.

[3] R.L. Burden and J.D. Faires, *Numerical Analysis*, PWS-KENT Publishing Company, Boston, 1989.

[4] K. Chen, P. Giblin and A. Irving, *Mathematical Explorations with Matlab*, Cambridge University Press, Cambridge, 2000.

[5] S.D. Conte and C. de Boor, *Elementary Numerical Analysis: An Algorithmic Approach*, McGraw-Hill, New York, 1980.

[6] D.J. Higham and N.J. Higham, *Matlab Guide*, SIAM, Philadelphia, 2000.

[7] E. Isaacson and H.B. Keller, *Analysis of Numerical Methods*, John Wiley & Sons, Inc., 1966.

[8] C.F. van Loan, *Introduction to Scientific Computing, A Matrix-Vector Approach Using Matlab*, Prentice Hall, Inc., 2000.

[9] J.H. Mathews and K.D. Fink, *Numerical Methods Using Matlab*, Pearson Prentice Hall, 2004.

[10] S.S. Rao, *Applied Numerical Methods for Engineers and Scientists*, Prentice Hall, Inc., 2002

[11] L.F. Shampine, R.C. Allen Jr. and S. Pruess, Fundamentals of Numerical Computing, John Wiley & Sons, Inc., 1997.

[12] R.D. Skeel and J.B. Keiper, *Elementary Numerical Computing with Mathematica*, McGraw-Hill, Inc., 1993.

Index