Hoofdstuk I

Kennismaking met de Numerieke Wiskunde

door G. Sleijpen

SLEUTELBEGRIPPEN: *benaderingsfout, foutvoortplanting, gestruktureerde fout, foutschatting, betrouwbaarheid foutschatting, efficiëntie, stabiliteit.*

1 Numeriek oplossen van differentiaalvergelijkingen

Problemen die je in de praktijk tegenkomt kunnen nogal eens gemodelleerd worden als een differentiaalvergelijking (DV) of een stelsel DVs [1]. Gewoonlijk zijn die DVs niet analytisch oplosbaar. Middels analytische technieken [1, Hoofdstuk 2] kan je deelaspekten 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 ontwerp fase van bijvoorbeeld een gebouw zal men zeker willen weten of de konstruktie 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 DV nauwkeurig opgelost kan worden en als dat niet zo is waar het van afhangt welke methode voor welk type DV het meest geschikt is. Zijn er standaard paketten? 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, het type DVs dat we in deze kursus 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 funktie en u een onbekende funktie. u moet bepaald worden. De funkties u en f zijn reëel-waardig of vektor-waardig: u is een C^1-afbeelding van het 'integratie' interval $[t_0, t_0 + T]$ naar $\mathbb{R}^d$, f is gedefineerd 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 vektor.
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 kursus t de tijd: t_0 is dan het aanvangstijdstip, T is de tijdsspanne waarover de oplossing van de DV gevolgd wordt, $t_0 + T$ is de eindtijd. Voor het schrijfgemak nemen we verder $t_0 = 0$.

De DVs die we hieronder beschrijven zullen we gebruiken om onze numerieke inzichten te testen. Andere voorbeelden zijn te vinden in diverse tekstboeken over gewone DVs (zie bijvoorbeeld [1]). De tekst hieronder bevat wat achtergrondinformatie bij de DVs die je voor de kursus niet per sé hoeft te lezen.[1]

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

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

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

De DV (1.2) is 1-dimensionaal en lineair. Als λ konstant is en g niet al te ingewikkeld is kan u analytisch bepaald worden. DV (1.2) is niet direkt een voorbeeld om de behoefte aan een numerieke methode te illustreren. Maar juist omdat de DV zo eenvoudig is, is ze geschikt voor numerieke experimenten. Als een numerieke methode faalt voor deze eenvoudige DV dan zal hij het zeker laten afweten voor gekompliceerdere. Verder kunnen we, als we de exacte oplossing hebben, controleren of onze uitspraken betreffende de fout in een numeriek resultaat korrekt 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 DV 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 konstant (o.a. $\lambda = 0.3$, $\lambda = -1$ en $\lambda = -1000$) of voor een $\varepsilon > 0, \varepsilon \ll 1$,

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

Voorbeeld 1B. De lineaire 1-dimensionale DV (1.2) is ook van het type (1.1) als λ en g complex-waardige funkties zijn. De oplossing u is dan ook complex-waardig. Door alle grootheden te splitsen in reële en imaginaire delen kan de DV geschreven worden als een lineaire 2-dimensionale reële DV. 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 DV geschreven kan worden als een reële $2d$-dimensionale DV.

[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 DV gedefineerd 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 funktie `odeExact` een plaatje proberen te maken van de exacte oplossing. Zo levert `odeExact('Linear')` een plaatje van de exacte oplossing van de DV 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 1C. [`Flame`] De volgende 1-dimensionale DV is een model voor het verloop van de temperatuur in een chemische reaktie. κ en μ zijn konstanten, $u(t)$ is de temperatuur op tijdstip t.

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

Voorbeeld 1D. [`Immuun`] Antigenen ("ziekmakers") worden in ons lichaam gebonden door antilichamen en vervolgens onschadelijk gemaakt. De DV 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

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

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

De DV (1.4) is 2-dimensionaal en met

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

zien we dat deze DV 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 truuk kan de hogere orde afgeleide omgeschreven worden naar een afgeleide van de eerste orde. De DV wordt hierdoor wel hoger dimensionaal.

Voor zekere reële konstanten 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 DV is 2-dimensionaal en weer van het type als in (1.1).

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

Voorbeeld 1F. [`Emigration`] De DV $\gamma\,u' = (\alpha - u)\,u$ is een eenvoudig model om de bevolkingsgroei te modelleren (*logistische groei*). Hierin zij α en γ positieve konstanten 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 DV 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 DV modelleert de baan die een kunstmaan K beschrijft die zich vrij (geen raket aandrijving) beweegt in een stelsel van een planeet A (aarde) met een sateliet 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 sateliet 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 het centrum van M): het (x, y)-assenstelsel roteert dus met A en M mee. A ligt op positie $(\mu, 0)$ en M op $(\nu, 0) \equiv (1 - \mu, 0)$. Als op tijdstip t de kunstmaan positie $(x(t), y(t))$ heeft dan geldt

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

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

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 4-dimensionale DV.

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

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

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 moet in een groot aantal cijfers bekend zijn.

1.2 Kennismaking met numerieke oplosmethoden

SLEUTELBEGRIPPEN: *Betrouwbaarheid numeriek antwoord, efficiëntie*

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

Vraag 2A. Kunnen computers DVs eigenlijk wel oplossen? Differentiëren is een limiet operatie. Computers kunnen alleen met getallen uit een eindige kollektie 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? (Zoem 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 §§ uitgebreid zullen bestuderen. De tweede routine `ode23` is zeer geavanceerd en behoort tot de ODE Toolbox van MATLAB.

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 de DV 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`).

Verklaring. Gaat de rest van het eerste hoofdstuk over.

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

---

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

[3] D.w.z, neem `PlotExactSolution=1`.

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

[5] Neem `PlotExactSolution=0`.

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

[7] `ShowSolutionProcess=1`

Bovenstaande experimenten roepen een aantal vragen op.
Hoe weet je of de h voldoende klein is, of daarmee 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 DV? Deze vragen staan centraal in dit hoofdstuk.

1.3 Konvergentie

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

In programma paketten worden geavanceerde methoden gebruikt. Om te begrijpen wat de problemen zijn waarmee je bij het numeriek oplossen van DVs te maken krijgt, bekijken we eerst een eenvoudige methode, die overigens voor talloze DVs 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 funktiewaarden niet. Maar we hebben ook nog de DV en we kunnen kijken waar een kombinatie 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 $u_n + hf(t_n, u_n)$: 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 de DVs hebben we grafisch gerepresenteerd. Hoe kan je de EF methode grafisch interpreteren?

Experiment. Plot voor (1.2) (`Linear`) met $\lambda = -1$ en g als in (1.2.c),[8] 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.[9] Plot ook de exacte oplossing van de DV (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 DV door achtereenvolgens (t_2, u_2),..., (t_6, u_6).[10] Bekijk de grafische resultaten in de buurt van de eerste paar t_n (zoom in).

[8] Deze DV is gedefinieerd in `Linear.m`.
[9] Gebeurt vanzelf met `Vraag3A.m`.
[10] Neem `PlotExactNearbySol=1`.

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. Konvergeert EF? Wat betekent konvergentie 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 konvergeert en als er konvergentie is, in welke zin dat gebeurt.[11]

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 **defekt** τ_n weggemoffeld wordt (vergelijk (3.3) en (3.2)). De defecten τ_n gaan naar 0 als $h \to 0$ (waarom?). Het is duidelijk dat de defekten de nauwkeurigheid van het numeriek resultaat beïnvloeden, het is niet duidelijk hoe ze dat doen. Het is evenmin duidelijk of er niet nog andere faktoren een rol spelen.

Bij konvergentiebeschouwingen 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 defekten de numerieke nauwkeurigheid beïnvloeden.

Als h klein is, dan is

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

Als we voor kleine h de stapgrootte h halveren dan worden de defekten met ongeveer een faktor 4 kleiner, immers $\frac{1}{2}(\frac{1}{2}h)^2 u''(t) = \frac{1}{4}[\frac{1}{2}h^2 u''(t)]$. 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. Defekten cumuleren.
De defekten 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) 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 defekten in voorgaande tijdstippen of hebben nog andere faktoren 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

[11] De tweede DV is gedefinieerd in `Immuun.m`.

$n = 1, \ldots, 6$ en kijk naar de resultaten in $t = 20$.
Tussen haakjes: halveert hier de fout nog steeds als je h halveert?

Verklaring. De DV zelf kan fouten vergroten.
Met EF maak je telkens foutjes (nl. de defekten) die de DV 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 'n $\varepsilon \neq 0$, dan geeft $e(t) \equiv u(t) - \widetilde{u}(t)$ aan hoe de DV de verstoring ε van de beginwaarde voortplant.

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

Over een tijdstrajekt(je) ter lengte h vergroot of verkleint de DV de fout, d.w.z. het effekt van de verstoring, met ongeveer een faktor $e^{h\,\lambda(t)}$.

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 konstant)*

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 geval* λ_0 *complex is en niet reëel (zie Voorbeeld 1B) geldt* $|e(t)| = |\varepsilon|\, e^{t\,Re(\lambda_0)}$.

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 DV fouten? Plot ook weer eens de exacte oplossingen door de eerste paar numerieke resultaten en zoem 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, nl. het defekt.

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 cumuleren.*
> **b.** *De DV zelf kan fouten vergroten.*
> **c.** *EF kan fouten vergroten.*

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

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.

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 DV 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 $h = 0.5$.

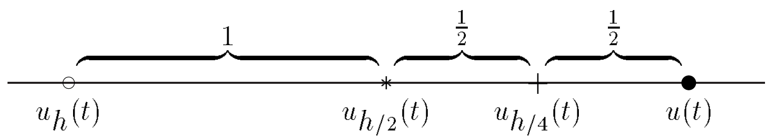
Verklaring. Als h niet klein is (of preciezer, als u'' veel varieert op $[t, t+h]$) dan is (3.4) niet korrekt. Soms is de fout “per ongeluk” ≈ 0.

Als in een numerieke oplosmethode van DVs 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 korrekt voor ingewikkeldere DVs?

Experiment a. Pas EF toe op de niet-lineaire DV (1.3) (`Flame`) met $\mu = \frac{1}{4}$ en $\kappa = 2$.[12] Hoe gedraagt de fout zich?

Verklaring. Als EF fouten niet beroerder voortplant dan de DV zelf dan gedragen de fouten zich min of meer als de oplossing van een lineaire DV 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 1-dimensionale versie van* (1.1).

a. *Bewijs dat, voor* t *en voor het geval* $\varepsilon \equiv e(t)$ *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 meer-dimensionale DVs? 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 funktie van t maar ook naar het faseplaatje. Doe hetzelfde eens voor $T = 40$.

[12] in `Flame.m`.

Verklaring. Bovenstaande beschouwingen kunnen koördinaatsgewijs 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 faktor $|1 + h\,\lambda_n|$ in (3.7). Je kan je hier afvragen hoe groot h mag/moet zijn. We bewaren deze vraag tot §1.4.

De trapeziumregel 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 Trapezium regel als je h verandert? Zijn er methoden die nog nauwkeuriger zijn?

Experiment a. Pas de trapezium regel toe op (1.4) (`Immuun`) met $\kappa = 1.3$ en $\vartheta = 0.06$ en kijk of je verwachting uitkomt.[13]

Conclusie. *De trapezium regel is van orde 2.*

Opgave. *Bewijs* (3.9).
Hint: bedenk dat $f(t, u(t)) = u'(t)$ en ontwikkel de voorkomende funkties rond t_n.

Experiment b. Welke orde heeft RK4?[14] De orde is lastig af te lezen uit de plaatjes. Hoe komt dat?

Opgave 3J. *Voor de liefhebbers, hieronder een schets van het bewijs dat de EF "konvergeert". 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* $\widetilde{u}_n$ *tusssen* $u(t_n)$ *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* $\left|\frac{\partial f}{\partial u}(t_n, \widetilde{u}_n)\right| \leq M$ *voor alle* n *dan geldt*

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

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

$$|e_n| \leq e^{n\,h\,M}\textstyle\sum_{j<n}|\tau_j| \leq e^{TM}\sum_{j<n}|\tau_j| \quad \text{voor alle} \quad t_n \equiv 0 + n\,h \leq T.$$

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

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

---

[13] De trapezium regel roep je aan met `odeIT`.

[14] `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`.

1.4 Stabiliteit

SLEUTELBEGRIPPEN: *Stabiliteit, groeifactor fout, foutvoortplanting, stijve DV, 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 subjektief: 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 ($300 \pm 0.3°$ 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 zullen we het stabiliteitsbegrip dat we in een bepaalde context zullen hanteren telkens precisiëren. Het precieze begrip zal dan versierd zijn met bijwoorden als 'sterk stabiel', 'asymptotisch stabiel', etc.. Het heuristisch begrip zal echter telkens op de achtergrond een rol spelen. 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 DV een evenwicht dat de oplossing van de DV opzoekt als diezelfde evenwichtsoplossing licht verstoord wordt (kleine verstoringen hebben uiteindelijk een klein effekt).

De oorzaak van een grote amplifikatie 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 DV 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 (welke?) kan het evenwicht wel toegankelijk worden voor een stabiele numerieke berekening.

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 meer-dimensionale DVs 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 faktor $|1 + \lambda h|$ waarmee EF voor deze DV 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 dezelfde manier als de DV dat doet.*
b. *Als* $|h\,\lambda_0| \not\ll 1$ *dan kan je alleen een nauwkeurige numerieke oplossing krijgen als (vrijwel) exact voldaan is aan de* stabiliteitseis $|1 + h\,\lambda_0| \leq 1$.

Vraag 4B. Bovenstaande conclusie geldt voor de 1-dimensionale lineaire DV (1.2) voor $\lambda(t) = \lambda_0$ is konstant. Wat hebben we eraan voor gekompliceerdere DVs? Blijft dan nog iets van de konklusie 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 $\varepsilon = 0.02$.[15] Voor welke h heb je geen stabiliteitsproblemen? Bekijk voor $h = 0.04$ ook eens de grafiek van $h\lambda(t)$.[16] 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 trajekten waar $|1 + h\,\lambda(t)| > 1$ blaast de fout op.

Experiment b. We kijken eens wat er van onze inzichten overblijven voor niet-lineare DVs. 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 $\lambda(t) \equiv \frac{\partial f}{\partial u}(t, u(t))$.[17]

Verklaring. De fouten gedragen zich min of meer als de oplossing van de lineaire DV (zie (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 $\frac{\partial f}{\partial u}(t_n, u_n)$.[18]

Verklaring. In de meer-dimensionale sitiuatie hangt de stabiliteit samen met $h\,\lambda(t)$ met $\lambda(t)$ eigenwaarde van de Jacobi matrix $h\,\frac{\partial f}{\partial u}(t, u(t))$.

Conclusie. *Ook voor niet-lineaire (meer-dimensionale) DVs manifesteert instabiliteit zich in de vorm van “wiggels” in de numerieke oplossing. In zo’n geval moet de stapgrootte* h *kleiner genomen worden. In geval van instabiliteit krijgen we niet een onnauwkeurige oplossing die er toch “redelijk” uitziet: door de “wiggels” zien we dat we met instabiliteiten te maken hebben.*

Vraag 4C. Voor welk type DVs 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 oplosingen door (t_n, u_n) voor de eerste paar n en kijk eens hoe die exacte oplossingen zich gedragen: zoem in bij $t = 0$. Neem nu eens $h = 0.07$ en $T = 0.5$. Zoem in, nu rond de hele exacte oplossing. Kan je grafisch zien waarom EF instabiel is?

Conclusie. *Als de exacte oplossing ‘langzaam’ varieert terwijl naburige oplossingen zeer ‘snel’ variëren dan hebben we te maken met een zogenaamde* **stijve** *DV. In zo’n geval bepalen bij EF de naburige snel variërende oplossingen de stapgrootte.*

Langzaam variërend wil voor de numerieke methode zeggen dat de defekten 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 mag dan $|u''|$ niet al te groot zijn: immers voor EF is $\tau_n = \frac{1}{2}h^2 u''(\xi_n)$.

[15] DV staat in `Linear2`.

[16] Neem `ShowEigenvaluesJacobian = 1;`. Na het volgende experiment zul je begrijpen waarom we de naam ‘ShowEigenvaluesJacobian’ gekozen hebben voor het plotten van de grafiek van $h\lambda(t)$.

[17] Neem `ShowEigenvaluesJacobian = 1;`.

[18] Neem `ShowEigenvaluesJacobian = 1;`.

Stijve DVs komen in de praktijk nogal eens voor. Ze ontstaan typisch als aan een goed model, d.w.z. aan een DV die het praktijk probleem al behoorlijk goed modelleert, ter perfektionering hogere orde afgeleiden met een klein gewicht worden toegevoegd. Je kan hierbij denken aan de (0-de orde) “DV” $u(t) - g(t) = 0$ 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 konstante. Omdat $w''(0) = u''(0) + \frac{c}{\varepsilon^2}$ zal $|w''(0)|$ groot zijn (tenzij $|c| \ll \varepsilon$). In de praktijk zijn de modellen (DVs) natuurlijk wel wat ingewikkelder.

Vraag 4D. Kunnen we voor stijve DVs 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 DV (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 faktor ook terug bij ingewikkeldere DVs? Bekijk ook eens (1.3) (zie experiment b in 4B).

Vraag 4E. We hebben hierboven gezien dat een nauwkeurigere methode voor stijve DVs 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 method (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 defekten τ_n^{EB} geeft EB.

Experiment a. Gebruik de routine ‘`odeEB`’ om de DV uit experiment a in 4A (`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 DV (`vdPol`) in experiment c in 4B? Neem ook eens in de DV uit experiment a in 4A (`Linear2`) $\varepsilon = 10^{-6}$ en $h = 0.1$. Hoe doet `ode23` het voor deze DV met achtereenvolgens $\varepsilon = 10^{-4}$, $\varepsilon = 10^{-5}$, $\varepsilon = 10^{-6}$?

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 DVs? 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 DV (zie (3.8)).

Vraag 4F. 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`) eens op met EB met $h = 0.1$. Hoe kan je verklaren dat je zo'n oplossing krijgt? Plot ook $h\,\lambda_i(t_n)$ als funktie van t_n.

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 meer-dimensionale DVs 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 postieve eigenwaarden zijn.*
> **c.** *Als nauwkeurigheid de bepalende faktor is en niet stabiliteit dan is EF veel efficiënter dan EB.*

In experiment c 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 4G. Zijn er methoden die een hogere orde nauwkeurigheid hebben dan EB (en EF) en waarmee je ook stijve DVs met wat grotere stapgrootte nauwkeurig kan oplossen?

Experiment. Pas de trapezium regel 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 4H. *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*

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

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 Toepassingen

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

Opdracht 5A. Onderzoek welke numerieke methode, EF of EB, het efficiëntst (in ze kort mogelijke rekentijd) de DV (1.3) (`Flame`) met $\mu = \frac{1}{4}$ en $\kappa = 20$ nauwkeurig oplost (met een betrouwbaarheid van 0.01). Doet EB het veel beter dan EF? Waarom wel/niet?

Opdracht 5B. Los de DV (1.8) (`Body3`) op met `odeRK` met $h = 0.007$. 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 (zoem eens in rond de positie $(-\mu, 0)$ van A)? 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 funktie 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?

1.6 Samenvatting

Bij DVs wordt de fout in het numeriek resultaat bepaald door

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

In §§1.6.1-1.6.3 analyseren we deze aspecten voor de 1-dimensionale DV

$$\begin{cases} u'(t) = \lambda(t)u(t) + g(t), \quad t \in [0, T] \\ u(0) = u_0. \end{cases} \tag{6.1}$$

Hierin zijn λ en g gegeven reëel (eventueel complex) waardige funkties en u_0 is een gegeven skalair. u is de exacte oplossing van (6.1). Daarna geven we in §1.6.4 aan hoe de resultaten er voor niet-lineaire – en meer-dimensionale DVs er uit zien.

1.6.1 Foutvoortplanting door de DV

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 DV numeriek nauwkeurig kunnen oplossen: immers in een numeriek proces zijn fouten onvermijdelijk (en fouten worden niet alleen in 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 (6.1).

Als $\widetilde{u}$ de 'verstoorde' oplossing is van (6.1), d.w.z. de oplossing van

$$\begin{cases} u'(t) = \lambda(t)u(t) + g(t), \quad t \in [0, T] \\ u(0) = u_0 + \varepsilon, \end{cases} \tag{6.2}$$

dan geeft

$$e(t) \equiv u(t) - \widetilde{u}(t) \tag{6.3}$$

aan hoe de verstoring van de beginvoorwaarde doorwerkt op de oplossing u in het tijdstip t. Trekken we (6.2) af van (6.1) dan zien we dat

$$\begin{cases} e'(t) = u'(t) - \widetilde{u}'(t) = \lambda(t)\, e(t), \\ e(0) = -\varepsilon. \end{cases} \tag{6.4}$$

Dus

$$e(t) = -c(t)\varepsilon, \quad \text{met} \quad c(t) \equiv \exp\left(\int_0^t \lambda(s)\,ds\right): \tag{6.5}$$

$c(t)$ geeft aan hoe de verstoring door de DV voortgeplant wordt.
Met $\lambda_{\text{min}} \equiv \min_t \text{Re}(\lambda(t))$ en $\lambda_{\text{max}} \equiv \max_t \text{Re}(\lambda(t))$ geldt $e^{\lambda_{\text{min}}T} \leq |c(t)| \leq e^{\lambda_{\text{max}}T}$. Als $\lambda_{\text{min}} > 0$ dan wordt de verstoring vergrotend voortgeplant, als $\lambda_{\text{max}} < 0$ is dat verkleinend. Met bv $\lambda_{\text{min}} = 2$ en $T = 20$ is $e^{\lambda_{\text{min}}T} = 2.4\,10^{17}$!

Een zelfde soort uitspraak geldt met betrekking verstoringen gemaakt op latere tijdstippen. Afhankelijk van de funktie 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 konstant is, dan geldt

$$e(t+h) \approx e^{h\,\lambda(t)}\,e(t): \tag{6.6}$$

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

1.6.2 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:

$$u_{n+1} = u_n + hf(t_n, u_n), \qquad t_0 = 0, \quad t_{n+1} = t_n + h, \quad n = 0, 1, 2, \ldots, \tag{6.7}$$

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:

$$u(t_{n+1}) = u(t_n) + h\,f(t_n, u(t_n)) + \tau_n, \quad \text{met} \quad \tau_n \equiv \tfrac{1}{2}\,h^2\,u''(\xi_n) \tag{6.8}$$

voor een zekere ξ_n in $[t_n, t_{n+1}]$ (gebruik een 1-ste orde Taylor benadering). De EF methode ontstaat dus door in ieder tijdstip t_n het **defekt** τ_n weg te poetsen[19] 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 defekt 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 defekt: EF maakt nieuwe fouten, maar "haalt ook nog eens oude koeien uit de sloot". Dit doet overigens iedere numerieke methode voor het oplossen van DVs. Noteren we de fout in t_n met e_n,

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

[19] De grootheid $\delta_n \equiv \tau_n/h$ noemt men ook wel **lokale diskretisatiefout**. 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.

dan geldt (trek (6.7) af van (6.8))

$$e_0 = 0, \qquad e_{n+1} = (1 + h\,\lambda_n)\,e_n + \tau_n, \quad n = 0, 1, 2, \ldots. \tag{6.9}$$

Als $\lambda = 0$, d.w.z., als $u' = g$ voor zekere $g = g(t)$, dan volgt uit (6.9) dat $e_n = \sum_{j<n} \tau_j$: de fout op een tijdstip is een cumulatie van defekten van voorgaande tijdstippen. Merk op dat een cumulatie van defekten een orde h groter kan zijn dan de defekten 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 defekten 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

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

in dit geval planten EF en de DV zelf fouten op min of meer dezelfde manier voort (zie (6.6)).

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

Stelling 1.1 *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*

$$|u(t_n) - u_n| \leq C \sum_{j<n} |\tau_j| \leq C \left(\tfrac{1}{2} T \max_{\xi \leq T} |u''(\xi)| \right) h. \tag{6.10}$$

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

Stelling 1.2 *Als λ begrensd is dan konvergeert 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 (6.10) is niet erg praktisch: in de praktijk kennen we C niet (zie §1.6.3) en u'' niet. Gelukkig vertoont de fout, voor h voldoende klein, een "h-gedrag": er is een funktie w, die niet afhangt van h en n, zodanig dat

$$u(t_n) - u_n \approx h\,w(t_n), \tag{6.11}$$

voor h voldoende klein (zie stelling 1.4). Omdat de fout zo fraai gestruktureerd is kunnen we aan de hand van de numerieke oplossingen zelf de fout schatten:

Stelling 1.3 *Als $\tau \in [0, T]$ en h is zo dat $\ell h = \tau$ en voldoende klein is, dan geldt*

$$u(\tau) - u^{(2)}_{2\ell} \approx \tfrac{1}{2} \left[u(\tau) - u^{(1)}_{\ell} \right], \tag{6.12}$$

hierin is $u^{(1)}_n$ de EF benadering met stapgrootte h en $u^{(2)}_n$ 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 1.4 *Als f voldoende glad is ($f \in C^2$), dan is er een funktie w en een $C > 0$ zodanig dat voor iedere h die voldoende klein is, geldt*

$$|u(t_n) - u_n - h\,w(t_n)| \leq Ch^2, \quad \textit{voor alle } n \textit{ waarvoor} \quad t_n \in [0, T]. \tag{6.13}$$

In plaats van "er is een C zodat voor iedere voldoende kleine h (6.13) geldt" zeggen we

$$u(t_n) - u_n = h\,w(t_n) + \mathcal{O}(h^2) \quad \text{uniform op} \quad [0, T]. \tag{6.14}$$

Het woord "uniform" wijst erop dat dezelfde C gebruikt moet worden voor alle h, voldoende klein, en *alle* n. In woorden luidt (6.14):
"de fout $u(t_n) - u_n$ is gelijk aan $h\,w(t_n)$ op een **orde** h^2-term na.

1.6.3 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 1.3 en 1.4 geven een hanteerbare foutschatting, maar vertellen niet wat "voldoende klein" is.

Schatting (6.10) geldt voor $C = e^{MT}$ waarbij $M = \max_t |\lambda(t)|$. De konstante e^{MT} is vaak waanzinnig groot, ook in situaties waarin de DV zelf verstoringen dempend voortplant. De konstante C is tot stand gekomen door de foutvoortplantingsfactoren $|1+h\,\lambda_n|$ af te schatten door $1+h\,M \leq e^{h\,M}$. 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 DV zelf. In dit geval is C min of meer gelijk aan de faktor waarmee de DV 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 $C = 1$ ($C \approx 1$). De veel te grote C komen we tegen 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 defekten voldoende klein zijn en de amplifikatie van de fout (door de foutvoortplanting) moet binnen de perken blijven. Zoals we net gezien hebben is dit laatste het geval als

(i) $|h\,\lambda_n| \ll 1$ voor alle n (denk aan $|h\,\lambda_n| \leq 0.2$) of (ii) $|1 + h\,\lambda_n| \leq 1$ voor (vrijwel) alle n.

Voorwaarde (ii) is een *stabiliteitseis*: oude fouten mogen niet vergroot worden. In zo'n geval is $C = 1$ in (6.10) en wordt de nauwkeurigheid in de numerieke oplossing gemajoreerd door een cumulatie van defekten. Als (i) of (ii) geldt dan wordt de fout in essentie volledig bepaald door Taylor resttermen en "voldoende klein" in de stellingen 1.3 en 1.4 betekent dat hogere orde Taylor resttermen verwaarloosbaar zijn ten opzichte van de laagste.

Als $-M \leq \lambda(t) \leq 0$ voor alle t dan geldt (ii) 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 (ii) 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:

$$\begin{cases} u_n = u_{n-1} + h\, f(t_n, u_n), \\ u(t_n) = u(t_{n-1}) + h\, f(t_n, u(t_n)) + \tau_n \quad \text{met} \quad \tau_n \equiv -\frac{1}{2}\, h^2\, u''(\xi_n). \end{cases} \tag{6.15}$$

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

$$e_n = (1 - h\, \lambda_n)^{-1}\, (e_{n-1} + \tau_n)\,. \tag{6.16}$$

Fouten in EB worden niet vergroot als

$$\left|(1 - h\, \lambda_n)^{-1}\right| \leq 1 \quad \text{voor alle} \quad n. \tag{6.17}$$

In dat geval kan de fout gemajoreerd worden door $\sum_{j<n} |\tau_j|$.
Als bv. Re$(\lambda(t)) \leq 0$ voor alle t, dan geldt (6.17) voor iedere h en de nauwkeurigheid in de numerieke oplossing wordt volledig bepaald door de defekten, ongeacht hoe groot $|\lambda(t)|$ is. Als Re$(\lambda(t)) \leq 0$ dan hoeven we ons met EB alleen om de grootte van de defekten te bekommeren, terwijl we met EF ook nog rekening moeten houden met de stabiliteit. Voor zogenaamde **stijve** DVs, DVs waarbij 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 is een **expliciete** methode, EB is een **impliciete**.

1.6.4 Niet-lineaire en meer-dimensionale DVs

Vervangen we in geval de 1-dim. DV niet-lineair is de funktie $\lambda(t)$ in de uitspraken betreffende fouten door

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

dan vinden we in essentie de korrekte uitspraken voor niet-lineaire DVs. 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 meer-dimensionale problemen is exact gelijk aan dat voor 1-dimensionale. De diskussie betreffende nauwkeurigheid en stabiliteit loopt ook niet wezenlijk anders. In plaats van skalairen komen in de diskussie vektoren en matrices voor. De grootheden u_n^* en u_n zijn dan d-dimensionale vektoren 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 vektor $v = (v_1, \ldots, v_d)^T$ moet men, voor 'n geschikte "norm", lezen als de norm van de vektor: bv. $(|v| =)\, \|(v_1, \ldots, v_d)^T\| = \max_{i \leq d} |v_i|$. Evenzo wordt de abolute waarde van een matrix de norm van die matrix. Normen van matrices zijn lastiger te interpreteren dan normen van vektoren. 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 (ii) voor het meer-dimensionale geval dat $|1 + h\, \lambda| \leq 1$ moet zijn voor alle eigenwaarden λ van de Jacobi matrix $(\frac{\partial f_j}{\partial u_i}(t, u))$ (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