Differential Equations

— Practicumnota's —

Bernard De Baets en Jan M. Baetens

UNIVERSITEIT
GENT

"*Science is a differential equation. Religion is a boundary condition.*"

— Alan Turing (1954) —

Woord vooraf

De wiskundige formulering van tal van wetenschappelijke en ingenieursproblemen resulteert vaak in vergelijkingen die de afgeleiden van één of meerdere onbekende functies bevatten. Het oplossen van deze differentiaalvergelijkingen (DVn) is geen sinecure. De meeste analytische oplossingsmethodes kennen slechts een beperkt toepassingsgebied. De onderzoeker zal al snel zijn/haar toevlucht moeten nemen tot numerieke technieken om de oplossing zo goed mogelijk te benaderen.

De noodzaak van benaderingstechnieken kunnen we illustreren aan de hand van de eenvoudige eerste-orde differentiaalvergelijking (DV)

$$y' = y^2 + 1\,.$$

De analytische oplossing van deze DV wordt gegeven door $y = \tan(t + C)$. Maken we de DV iets complexer:

$$y' = y^3 + t^2\,,$$

dan kan er geen analytische oplossing meer gevonden worden. In dat geval moeten we de oplossing van de DV benaderen aan de hand van numerieke technieken.

Gezien de diversiteit van de beschikbare technieken, zal men steeds op zoek moeten gaan naar technieken die, voor het gestelde probleem, de afwijkingsfouten (vermoedelijk) minimaliseren. Ook hier bestaat er geen vuistregel die steeds tot het gewenste resultaat leidt.

De tweespalt, oplossen of benaderen, weerspiegelt zich ook in dit opleidingsonderdeel. Verschillende numerieke methodes worden besproken in deze practicumnota's, terwijl het analytisch oplossen van enkele basistypes DVn aan bod komt in de opgeloste oefeningen. In deze cursus zijn we genoodzaakt ons te beperken tot de meest gangbare numerieke en analytische oplossingsmethodes. Voor meer geavanceerde technieken verwijzen we naar de referenties die werden opgenomen in de bibliografie. Het is de bedoeling dat jullie vertrouwd raken met enkele basistechnieken en inzicht krijgen in de materie. Practica leveren het ideale kader om deze vaardigheden te verwerven. Om de gebruikte numerieke technieken op een educatieve manier te visualiseren, hebben we gekozen voor MATLAB als 'Computing Environment'.

Deze nota's zijn opgebouwd uit zeven practica waarin jullie veelgebruikte numerieke technieken voor het oplossen van (stelsels) DVn zullen aanleren. In Practicum 1 wordt aandacht besteed aan de meetkundige interpretatie van eerste-orde differentiaalvergelijkingen, terwijl numerieke methoden voor het oplossen van één-dimensionale (partiële) DVn behandeld worden in Practica 2 tot en met 4. Hierbij zal ruime aandacht besteed worden aan populaire, doch eenvoudige, numerieke methoden zoals de Euler, midpoint, de klassieke vierde-orde Runge-Kutta en de eindige differentiemethodes. In Practicum 5 worden de eerder bestudeerde methodes uitgebreid voor het oplossen van stelsels DVn. In het voorlaatste practicum proberen we DVn van hogere orde op te lossen. Practicum 7 bevat enkele syntheseopdrachten en vroegere examenvragen die vrijblijvend kunnen opgelost worden om je kennis van de leerstof te toetsen. Doorheen deze practica zal er steeds bijzondere aandacht besteed worden aan de evenwichtspunten en stabiliteit van de bestudeerde numerieke methodes en DVn. Tot slot van deze nota's werden er twee bijlagen opgenomen waarin wordt geïllustreerd hoe DVn symbolisch kunnen worden opgelost met Mathematica (Bijlage A) en Wolfram|Alpha (Bijlage B).

In elk practicum werden verschillende opdrachten opgenomen. Deze opdrachten zijn representatief voor wat je op het examen mag verwachten. Theoretisch getinte vragen zijn te herkennen aan een sterretje in de kantlijn. Ze peilen voornamelijk naar het verworven inzicht in de materie en zijn op te lossen zonder MATLAB. Naast het educatieve aspect hebben de practica tevens tot doel de zelfwerkzaamheid te stimuleren, een onontbeerlijke eigenschap van toekomstige ingenieurs.

Niettemin deze cursus zich uitsluitend toespitst op (partiële) DVn, willen we benadrukken dat er naast deze DVn nog talrijke andere wiskundige paradigma's bestaan voor het beschrijven van (a)biologische processen, waaronder stochastische (partiële) DVn, cellulaire automaten en individu-gebaseerde modellen het meest vermeldenswaardig zijn.

Hoewel deze nota's regelmatig worden bijgewerkt en afgestemd op de veranderende noden, zijn wij ons ervan bewust dat kleine foutjes in tekst en figuren niet uitgesloten zijn. Wij stellen het dan ook op prijs wanneer deze aan ons worden meegedeeld zodat deze kunnen weggewerkt worden in de volgende versie van deze nota's.

Werkten mee aan deze en/of vroegere versies van deze nota's: Kim Cao-Van, Koen Maes, Hilde Vernieuwe, Wouter Naessens en Demir Ali Köse.

Gent, 2 mei 2017,

De auteurs

Inhoudsopgave

1 Richtingsvelden en evenwichtspunten

Het is vaak niet mogelijk om de eerste-orde DV

$$y' = f(t, y) \tag{1.1}$$

analytisch op te lossen. In dergelijke gevallen dient men een beroep te doen op numerieke technieken om de oplossing van deze DV te benaderen. Alvorens een aantal numerieke technieken van naderbij te bekijken, zullen we in dit practicum aandacht besteden aan de meetkundige interpretatie van $y' = f(t, y)$. Deze meetkundige inzichten vormen de basis voor vele van de numerieke technieken die doorheen het vervolg van deze nota's aan bod zullen komen. Aangezien kennis van de exacte oplossing toelaat om de meetkundige interpretatie te verifiëren, illustreren we eerst hoe deze kan bepaald worden met Mathematica.

1.1 Differentiaalvergelijkingen in Mathematica

Een DV in de onbekende functie $y = y(x)$ kan in Mathematica eenvoudig opgelost worden a.d.h.v. de instructie

```
DSolve[{dv}, y[x], x]
```

waarbij `dv` de DV voorstelt in de onbekende functie $y(x)$. De volledige DV wordt dus toegekend aan een veranderlijke `dv` waarbij we $y(x)$ vervangen door `y[x]` en $y'(x)$ door `D[y[x], x]` of `y'[x]`, $y''(x)$ door `D[y[x], {x, 2}]` of `y''[x]`, ..., en tenslotte $y^{(n)}(x)$ door `D[y[x], {x, n}]`. Voor de DVn in deze nota's is de accentnotatie te verkiezen. Merk op dat er nergens ronde haken gebruikt worden[1]. Een gelijkheidsteken (=) plaats je steeds dubbel

[1] Ronde haken in Mathematica zijn voorbehouden voor het groeperen van bewerkingen, terwijl rechte haken worden gebruikt om de inputs van functies te omgeven.

(==). De oplossing van de DV wordt bekomen door tegelijkertijd de *Shift*- en de *Enter*-toets in te drukken. De bekomen oplossing kan vereenvoudigd worden door `Simplify` als volgt toe te voegen aan de instructie: `Simplify[DSolve[...]]`.

Willen we de oplossing van een DV met begin- of randvoorwaarden (*initial conditions*) berekenen, dan gebruiken we de volgende instructie

```
DSolve[{dv, BVn}, y[x], x]
```

waarbij `BVn` de beginvoorwaarden zijn, gescheiden door komma's. De afgeleiden die in de beginvoorwaarden voorkomen, kunnen uitgedrukt worden zoals voorheen: de beginvoorwaarde $y'(0) = 1$ geven we in als `y'[0] == 1`, $y''(0) = 1$ als `y''[0] == 1`, enz. Je kan Mathematica verplichten een probleem op te lossen volgens een welbepaalde methode. Meer informatie hieromtrent vind je in de helpfunctie van het programma. Bijlage A biedt een overzicht van nuttige Mathematica-instructies voor de studie van DVn.

Indien de lezer geen toegang heeft tot Mathematica, kan deze gebruik maken van Wolfram|Alpha dat in 2009 werd gelanceerd door Wolfram Research Inc. en fungeert als een intelligente zoekmachine die in staat is om wiskundige input correct te interpreteren, alsook om wiskundige bewerkingen uit te voeren en vergelijkingen op te lossen. De website kan bezocht worden via `http://www.wolframalpha.com/`. Een beknopte handleiding voor het oplossen van DVn met Wolfram|Alpha kan geraadpleegd worden in Bijlage B.

Opdracht 1.1

1. *Bepaal m.b.v. Mathematica de algemene oplossing van*

$$\frac{d^4 y}{d x^4} - 4\,\frac{d^3 y}{d x^3} - 5\,\frac{d^2 y}{d x^2} + 36\,\frac{d y}{d x} - 36\,y = 0.$$

$y(x) =$

Indien $y(0) = 1,\ y'(0) = 2,\ y''(0) = 3$ *en* $y'''(0) = 4$*, bepaal dan de unieke oplossing.*

$y(x) =$

2. *Welke oplossing vind je met Mathematica voor de DV*

$$x^2\,y\,y' - e^y = 0,$$

1

die voldoet aan $y(1) = 0$*?*

$y(x) =$

1.2 Richtingsvelden

Beschouw een eerste-orde DV $y' = f(t, y)$ en een willekeurig punt (t_0, y_0) van het (t, y)-vlak. Zij $y(t)$ een oplossing van de DV die voldoet aan de beginvoorwaarde $y(t_0) = y_0$, dan zal de grafiek van $y(t)$ door het punt (t_0, y_0) gaan. In het vervolg van deze nota's zal deze exacte oplossing $y(t)$ doorheen het punt (t_0, y_0) vaak aangeduid worden als $y_{(t_0,y_0)}(t)$.

De richtingscoëfficiënt van de raaklijn in (t_0, y_0) aan de grafiek van $y(t)$ wordt gegeven door $y'(t_0) = f(t_0, y_0)$. Vermits deze raaklijn ook door het punt (t_0, y_0) moet gaan, kunnen we haar vergelijking bepalen:

$$y = f(t_0, y_0)(t - t_0) + y_0 .$$

Merk op dat we hier enkel gebruik maakten van de functie f en de coördinaten (t_0, y_0) van het gekozen punt, zodat door elk punt van het (t, y)-vlak dat gelegen is in het domein van f de raaklijn kan getekend worden aan de oplossing van de DV die door het punt gaat en die aldus voldoet aan de beginvoorwaarde bepaald door het punt. Dit kunnen we zonder de oplossing van de DV expliciet te kennen. Plotten we voor diverse punten van het (t, y)-vlak deze raaklijnen, dan verkrijgen we een richtingsveld (*direction field*).

De MATLAB-functie `richtingsveld` kan gebruikt worden om het richtingsveld van een eerste-orde DV te plotten.

Codefragment 1.1: richtingsveld

```
function richtingsveld(f, tinterval, yinterval, n)
% Deze functie plot het richtingsveld van y' = f(t,y)
% Syntax: richtingsveld(f, tinterval, yinterval, n)
% Input: f: fuction handle naar het rechterlid van de DV
%        grenzen van de grafiek:
%        tinterval = [tmin, tmax]
%        yinterval = [ymin, ymax]
%        n = precisie van het richtingsveld, default n = 25
% Output: richtingsveld van y' = f(t,y)
```

```
if nargin <= 3    % indien precisie n niet werd opgegeven, wordt de
     n = 25;      % default waarde van 25 gebruikt
end

tmin = tinterval(1);
tmax = tinterval(2);
ymin = yinterval(1);
ymax = yinterval(2);
deltaT = (tmax - tmin)/n;
deltaY = (ymax - ymin)/n;
t = tmin:deltaT:tmax;
y = ymin:deltaY:ymax;
V = zeros(length(y), length(t));
U = zeros(length(y), length(t));
for i = 1:length(y)           % for-lussen om punten in het
    for j = 1:length(t)       % (t,y)-vlak te bepalen
        M = f(t(j), y(i));    % berekening van f(t,y), dit is
                              % de helling (dy/dx) van de oplossing
        L = sqrt(1 + M^2);    % berekening vectorlengte
        V(i,j) = M/L;         % normaliseren y-component
        U(i,j) = 1/L;         % normaliseren x-component
    end
end
quiver(t, y, U, V, 0.4)       % functie om richtingsvectoren
                              % te plotten
axis([tmin tmax ymin ymax])
title(['Richtingsveld (n = ' num2str(n) ')'])
xlabel('t')
ylabel('y')
end
```

De matrices `U` en `V` bevatten respectievelijk de genormaliseerde x- en y-componenten van de richtingsvectoren in de punten (t, y). De MATLAB-functie `quiver` wordt gebruikt om de richtingsvectoren te plotten. De waarde 0.4 die als laatste argument aan deze functie wordt meegegeven, is een schalingsfactor die ervoor zorgt dat de vectoren overzichtelijk worden weergegeven. De precisie n drukt uit dat er $(n+1)^2$ richtingsvectoren geplot woren. Als je de precisie niet specificeert, wordt automatisch de waarde 25 gekozen.

1

Voorbeeld 1.1 Richtingsveld van een differentiaalvergelijking

We zullen het concept richtingsveld illustreren a.d.h.v. de volgende eerste-orde DV

$$y' = y\left(-2\,t + \frac{1}{t}\right). \tag{1.2}$$

Indien we werken over het t-interval $[0.2,\ 3]$ *met beginvoorwaarde* $y\,(0.2) = 0.2$*, dan wordt de exacte oplossing van DV* (1.2) *gegeven door* $y(t) = t\,e^{0.04-t^2}$ *(controleer dit zelf met Mathematica). Vooraleer de functie* `richtingsveld` *kan gebruikt worden, moet het rechterlid van DV* (1.2) *als volgt geïmplementeerd worden in een m-file:*

Codefragment 1.2: functieRV

```
function ydot = functieRV(t, y)
% rechterlid van DV (1.2)

ydot = y*(-2*t + 1/t);
end
```

Opdat we de exacte oplossing eenvoudig bovenop het richtingsveld zouden kunnen plotten, dient ook deze geïmplementeerd te worden in een m-file.

Codefragment 1.3: exactRV

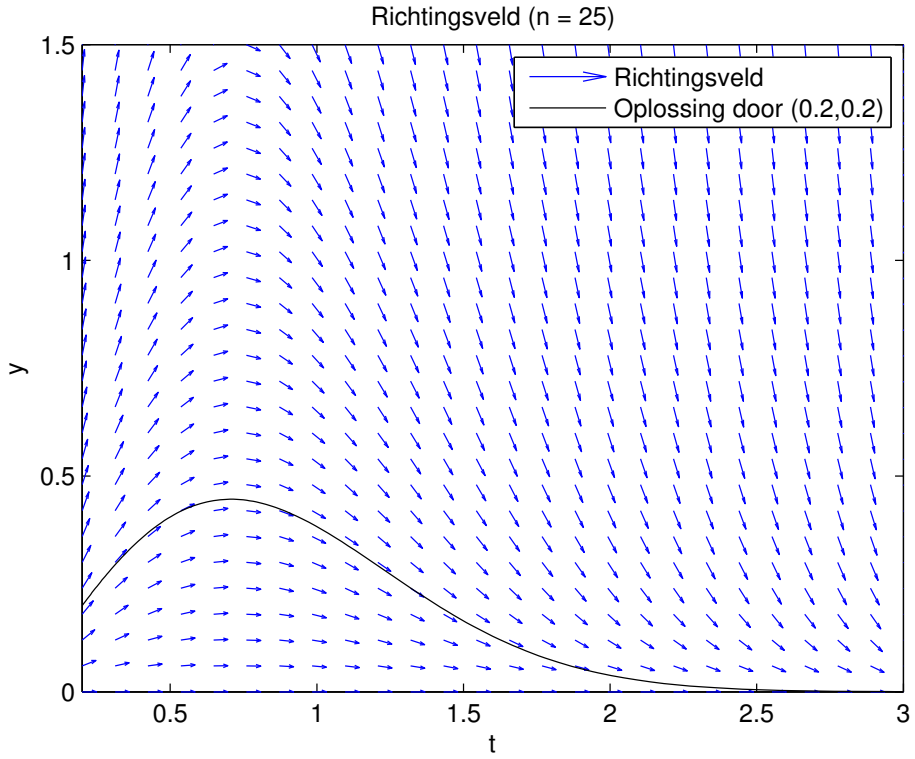
```
function y = exactRV(t)
% exacte oplossing van DV (1.2)
% waarbij y(0.2) = 0.2

y = t.*exp(0.04 - t.^2);    % gevectoriseerd
end
```

Indien je het richtingsveld van DV (1.2) *over het gebied* $[0.2, 3] \times [0, 1.5]$ *samen met de exacte oplossing over het tijdsinterval* $[0.2, 3]$ *wenst weer te geven, dien je onderstaand script dat de hiervoor noodzakelijke instructies bevat, uit te voeren.*

Codefragment 1.4: scriptRV

```
clc                          % wissen instructievenster
clear all                    % wissen werkruimte

figure(1)                    % open een nieuw figuurvenster
richtingsveld(@functieRV, [0.2, 3], [0, 1.5]);
hold on                      % grafisch venster vasthouden
fplot(@exactRV, [0.2, 3])    % plotten van de exacte oplossing
legend('Richtingsveld', 'Oplossing door (0.2,0.2)')
```

Het resultaat hiervan wordt gegeven in Fig. 1.1. Beschouwen we nu in deze figuur het punt $(1, 0.6)$ *in het* (t, y)*-vlak. Substitutie van dit punt in het rechterlid van DV* (1.2) *levert* $y' = -0.6$*, waaruit we kunnen besluiten dat de raaklijn aan de oplossing in* $(1, 0.6)$ *een negatieve helling heeft die zwakker is dan deze van de tweede bissectrice waarvoor* $y' = -1$*. Dit wordt bevestigd in Fig. 1.1. De pijltjes in deze figuur geven de zin aan volgens dewelke de oplossing evolueert bij toenemende tijd. Vermits de richtingscoëfficiënt berekend wordt voor toenemende t-waarden, wijzen de pijltjes steeds naar rechts. Deze voorstelling maakt het mogelijk om het verloop van de oplossing bij om het even welke beginvoorwaarde te voorspellen.*



Figuur 1.1: Richtingsveld van DV (1.2) over het gebied $[0.2, 3] \times [0, 1.5]$.

Opdracht 1.2

1. *Beschouw de volgende eerste-orde DV over het t-interval* $[-2, 2]$

$$y' = y + 3\,t - t^3.$$

   (a) *Implementeer het rechterlid van deze DV in een m-file* `functie1.m`.

   (b) *Plot het richtingsveld van deze DV. Zet hierbij de y-as op* $[-2, 2]$.

(c) Beschrijf wat er gebeurt wanneer je de precisie n opdrijft. Stel n bijvoorbeeld gelijk aan 10, 20, 50 en 100.

(d) Kan je het globale verloop van de exacte oplossing door het punt $(-2, 0.5)$ *voorspellen a.d.h.v. deze plot?*

(e) Bepaal de algemene oplossing van deze DV m.b.v. Mathematica.

$y(t) =$

(f) Bepaal met Mathematica de exacte oplossing van de DV door het punt $(-2, 0.5)$ *en implementeer deze oplossing in een m-file* `exact1.m`*. Gebruik* `Simplify` *om de bekomen oplossing te vereenvoudigen. Plot de exacte oplossing door* $(-2, 0.5)$ *bovenop het richtingsveld. Komt het resultaat overeen met het onder (d) voorspelde verloop?*

(g) Bepaal met Mathematica de unieke oplossing van de DV bij de generieke beginwaarden $y(t_0) = y_0$*.*

$y(t) =$

Implementeer deze in MATLAB als volgt:

Codefragment 1.5: exact1t0

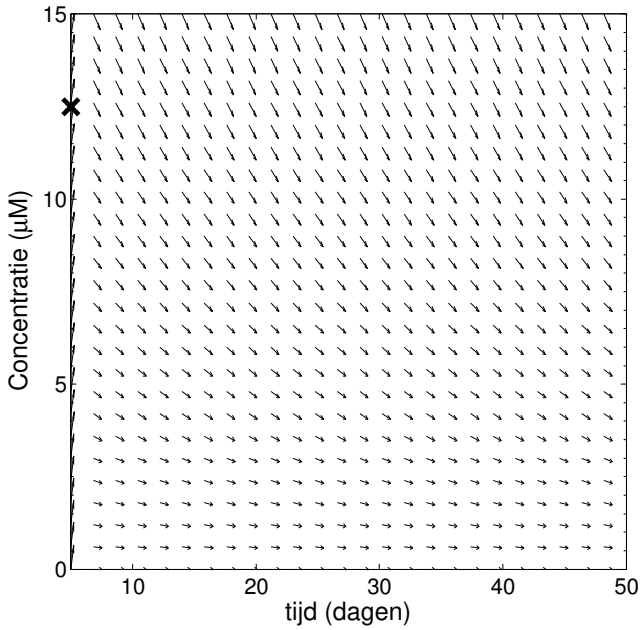
```
function y = exact1t0(t, t0, y0)
% Unieke algemene oplossing van y'(t) = y(t) + 3*t - t^3
% indien y(t0) = y0

y =

end
```

(h) Visualiseer de exacte oplossingen bij verschillende beginvoorwaarden bovenop het richtingsveld indien $t_0 = -2$ en $y_0 \in \left\{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1, \frac{5}{4}, \frac{3}{2}\right\}$. Gebruik hiervoor de volgende instructie:

```
fplot(@(t)exact1t0(t, -2, 0:0.25:1.5), [-2, 2])
```

Bespreek wat je ziet.

2. In de nacht van 3 op 4 januari heeft een onbekende enkele vaten met een toxische stof geloosd in de Coupure. Eén dag na het voorval worden stalen genomen in de buurt van de lozingsplaats. De analyse hiervan wijst uit dat de concentratie van de toxische stof op die dag 1.25×10^{-2} mol m^{-3} bedroeg. Studenten van de faculteit Bio-ingenieurswetenschappen gebruikten een differentiaalvergelijking om de concentratie van de toxische stof te voorspellen i.f.v. de tijd. Figuur 1.2 illustreert het richtingsveld dat dit model opleverde. Zal de toxische stof na 50 dagen nog kunnen gedetecteerd worden indien de detectielimiet 5×10^{-3}mol m^{-3} bedraagt?*



Figuur 1.2: Richtingsveld ter voorspelling van de concentratie toxische stof i.f.v. de tijd.

1.3 Evenwichtspunten

1.3.1 Definitie en classificatie

Laten we eerst even teruggrijpen naar Fig. 1.1 die het richtingsveld van DV (1.2) illustreert. Op deze figuur zie je duidelijk dat alle oplossingen - ongeacht de beginvoorwaarde - convergeren naar $y = 0$. Verder valt op te merken dat de richtingsvectoren gelegen op de x-as horizontaal gericht zijn. Kunnen we deze vaststellingen verklaren a.d.h.v. de DV zelf?

Gelet op het rechterlid van DV (1.2) geldt dat $y' = 0$ indien $y = 0$ voor alle $t \in]\,0, +\infty\,[$. Dit betekent dat de oplossingen doorheen (t_0, y_0), waarbij $y_0 = 0$, niet meer veranderen in de tijd. De oplossing $y(t) = 0$ wordt dan ook een evenwichtsoplossing (*equilibrium solution*) van de DV genoemd en $y_e = 0$ is het corresponderende evenwichtspunt.

Algemeen geldt dat de evenwichtspunten (*equilibrium points*) van de eerste-orde DV

$$y'(t) = f(t, y)$$

die punten y_e zijn waarvoor geldt dat $f(t, y_e) = 0$ voor alle $t \geq t_c$, met t_c een vrij te bepalen kritisch tijdstip. Stelt men dan als beginvoorwaarde $y(t_0) = y_e$ met $t_0 \geq t_c$, dan is $y(t) = y_e$ voor alle $t \geq t_0$. Evenwichtspunten kunnen we opsporen door in het richtingsveld te kijken of een oplossing constant zal worden en blijven voor voldoende grote t-waarden. Vermits een oplossing constant is wanneer $y' = 0$ stemt dit overeen met het zoeken naar opeenvolgende horizontale richtingsvectoren in het richtingsveld. Kijken we terug naar Fig. 1.2 dan kan het evenwichtspunt $y_e = 0$ grafisch gevonden worden vermits de vectoren gelegen op de x-as horizontaal gericht zijn.

Evenwichtspunten kunnen ingedeeld worden o.b.v. hun aantrekkend of afstotend karakter:

1. Een evenwichtspunt y_e is **stabiel** (*stable*) als er voor iedere omgeving U van y_e een omgeving $U_1 \subseteq U$ bestaat, zodat voor iedere beginwaarde $y_0 \in U_1$ de bijhorende oplossing $y_{(t_0,y_0)}(t)$ gedefinieerd is en in U ligt voor alle $t \geq t_0$. Anders gezegd, alle oplossingen blijven begrensd voor $t \to +\infty$.

2. Een stabiel evenwichtspunt y_e heet **asymptotisch stabiel** (*asymptotically stable*) als U_1 zodanig gekozen kan worden dat bovendien geldt dat:

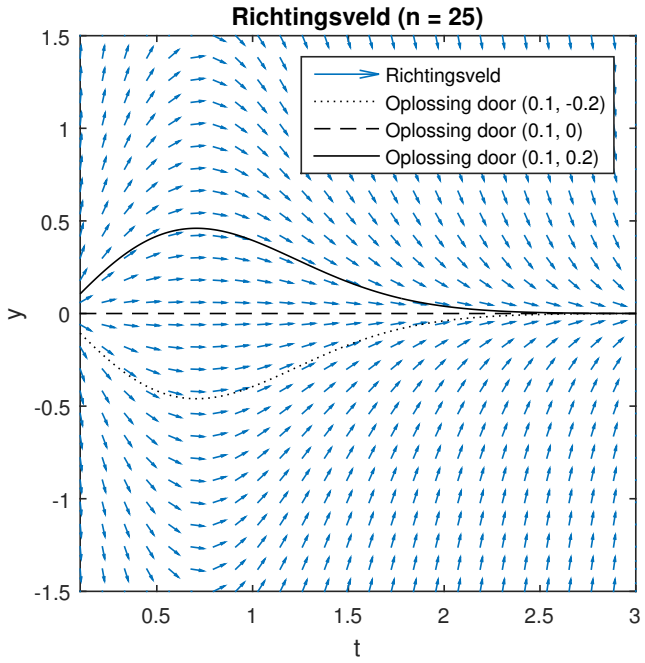
   $$\lim_{t \to +\infty} y_{(t_0,y_0)}(t) = y_e\,,$$

   voor elke $y_0 \in U_1$. In dit geval zullen alle oplossingen doorheen (t_0, y_0) convergeren naar y_e en kan gesteld worden dat y_e de oplossingen aantrekt.

3. Een evenwichtspunt dat niet stabiel is, wordt **onstabiel** (*unstable*) genoemd. In dit geval wijzen de richtingsvectoren weg van y_e en worden de oplossingen als het ware afgestoten door y_e. Bijgevolg zal de afstand tussen y_e en $y(t)$ groter worden naarmate t toeneemt.

Voorbeeld 1.2 Classificatie van evenwichtspunten

Beschouw opnieuw het richtingsveld van DV (1.2)*, maar nu over het gebied* $[0.1, 3] \times [-1.5, 1.5]$ *(Fig. 1.3). We wisten reeds dat* $y_e = 0$ *een evenwichtspunt is van deze DV vermits* $f(t, 0) = 0$*, voor alle* $t \geq 0.2$ *(kies bv.* $t_c = 0.2$*).*



Figuur 1.3: Richtingsveld van DV (1.2) over het gebied $[0.2, 3] \times [-1.5, 1.5]$.

Uit bovenstaande definities volgt nu dat $y_e = 0$ *een asymptotisch stabiel evenwichtspunt is. Immers, beschouw ter illustratie de omgeving* $U = [-0.5, 0.5]$ *van* y_e*. De exacte oplossingen door de punten* $(0.1, y_0)$*,* $y_0 \in [-0.2, 0.2]$*, zullen enkel waarden bereiken in* U*. Bijgevolg volstaat het* $U_1 = [-0.2, 0.2]$ *te kiezen. Algemeen kan men voor elke keuze van* U *een* $U_1 \subseteq U$ *bepalen waarvoor* $y_{(t_0, y_0)}(t) \in U$*, voor elke* $y_0 \in U_1$ *en voor alle* $t \geq t_0$*. Tevens is het duidelijk dat alle naburige oplossingen voor* $t \to +\infty$ *zullen convergeren naar* $y_e = 0$*. Het evenwichtspunt* $y_e = 0$ *is bijgevolg asymptotisch stabiel.*

1

Opdracht 1.3

Bepaal de evenwichtspunten en hun stabiliteit (stabiel, asymptotisch stabiel, onstabiel) van de volgende DVn indien we werken over het t-interval $[0, 2]$*:*

1. $y' = h(t-1)$, *waarbij* $h(t) = 1$ *als* $t \geq 0$ *en* $h(t) = 0$ *anders.*

   $y_e =$

2. $y' = 1 - h(t-1)$, *waarbij* $h(t) = 1$ *als* $t \geq 0$ *en* $h(t) = 0$ *anders.*

   $y_e =$

1.3.2 Autonome differentiaalvergelijkingen

Een eerste-orde DV van de vorm $y' = f(t, y)$ wordt autonoom (*autonomous*) genoemd indien het rechterlid niet expliciet afhangt van de tijd t, d.i.

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

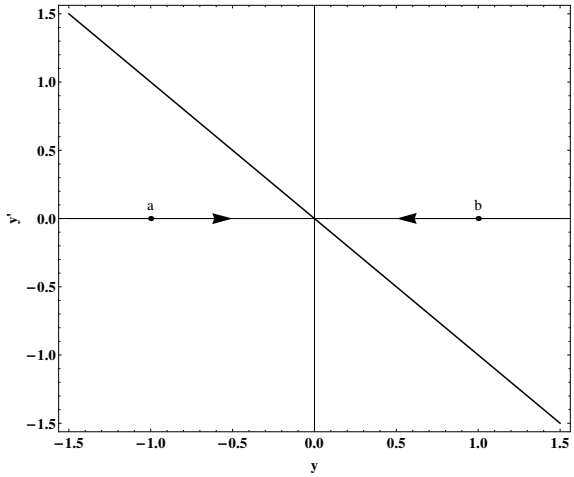
In een dergelijk geval kunnen we duidelijke inzichten in het gedrag van de DV verkrijgen door y' grafisch voor te stellen i.f.v. y. We zullen met andere woorden de DV $y' = f(y)$ visualiseren in een $(y, y')-$vlak. Uit het voorgaande weten we dat evenwichtspunten optreden wanneer $y'(t) = 0$ voor alle $t \geq t_c$. In het bijzonder moet er, voor een autonome DV gelden dat $y' = f(y_e) = 0$. In de beschouwde grafische voorstelling kunnen evenwichtspunten dan ook gevonden worden als de snijpunten van $f(y)$ met de horizontale as.

Vermits y' de bewegingssnelheid van een punt y weergeeft, wordt de richting van de beweging op de y-as bepaald door het teken van y'. Merk verder op dat de positie van het punt op de horizontale as wel nog afhankelijk is van de tijd aangezien y een functie is van t.

Voorbeeld 1.3 Evenwichtspunt van een autonome differentiaalvergelijking

Beschouw de DV afgebeeld in Fig. 1.4. Gelet op het voorgaande weten we dat $y_e = 0$ *een evenwichtspunt is van deze DV. Bovendien kunnen we o.b.v. het teken van* y' *de bewegingsrichting van de oplossing in de punten a en b bepalen en aldus het type evenwichtspunt.*

Immers, uit Fig. 1.4 volgt onmiddellijk dat $y'(a) > 0$. Veronderstellen we nu dat de exacte oplossing van deze DV $y = a$ bereikt op het tijdstip t_1, d.i. $y(t_1) = a$, dan moet, gelet op $y'(a) > 0$, $y(t_1 + \Delta t) > y(t_1)$ waaruit volgt dat het beeld van $y(t)$ zal toenemen naarmate t toeneemt zodat we ons langsheen de y-as in Fig. 1.4 naar rechts bewegen. Deze beweging zal met afnemende snelheid doorgaan totdat het evenwichtspunt $y_e = 0$ bereikt wordt aangezien in een dergelijk punt geldt dat $y'(y_e) = 0$.



Figuur 1.4: Plot van de DV $y' = m\,y, \quad m < 0$.

Een analoge redenering kan opgebouwd worden voor oplossingen die doorheen positieve y gaan, maar zulke oplossingen zullen bij toenemende t evolueren naar kleinere y vermits $y' < 0$ voor alle y langsheen de positieve y-as. Op basis van deze argumentering is het duidelijk dat het evenwichtspunt alle oplossingen aantrekt ongeacht y_0 zodat we y_e mogen typeren als asymptotisch stabiel.

Opdracht 1.4

1. Beschouw de DV*

$$y' = y^4 + 2\,y^3 - 3\,y^2 - 8\,y - 4\,,$$

waarvan het rechterlid word afgebeeld in Fig. 1.5.

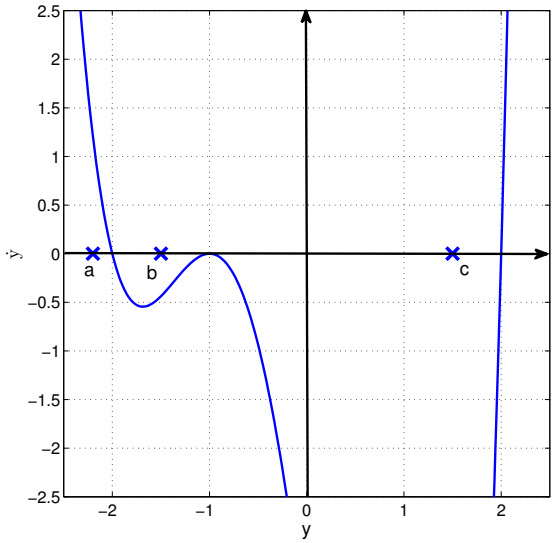
(a) Vind en classificeer de evenwichtspunten in $[-2.5, 2.5]$ van deze DV.

$y_{e,1} =$

$y_{e,2} =$

$y_{e,3} =$

(b) Duid op de horizontale as eveneens de bewegingsrichting aan van een object dat zich in het punt a, b of c bevindt.



Figuur 1.5: Plot van de DV $y' = y^4 + 2\,y^3 - 3\,y^2 - 8\,y - 4$.

2.* *Indien $f(y)$ afleidbaar is over het beschouwde y-interval, dan is het mogelijk om o.b.v. $f'(y_e)$ de stabiliteit van evenwichtspunten van autonome DVn te bepalen. Kun je zelf deze test formuleren?*

3.* *Een eerste-orde DV die veelvuldig gebruikt wordt voor de beschrijving van populatiegroei is de logistische groeivergelijking, gegeven door:*

$$\frac{d\,P}{d\,t} = r\,P\left(1 - \frac{P}{K}\right), \tag{1.3}$$

waarin P de populatiegrootte, r de intrinsieke groeisnelheid, d.i. de groeisnelheid in afwezigheid van groeilimiterende factoren, en K de draagkracht van de omgeving voorstellen.

(a) Welke is (zijn) de evenwichtstoestand(en) van het groeimodel?

(b) Verifieer je uitspraak voor het geval dat $r = \frac{1}{2}$ en $K = 10$. Werk hierbij over het gebied $[0, 20] \times [0, 20]$. Bepaal ook de stabiliteit van het (de) evenwichtspunt(en).

4. *Bepaal de evenwichtspunten en hun stabiliteit (stabiel, asymptotisch stabiel, onstabiel) van de volgende DV. Doe dit op drie verschillende manieren: m.b.v. het richtingsveld, m.b.v. de plot van $y' = f(y)$, en de regel die je afleidde in (2.*).*

   $y' = 2\sqrt{y}$.

   $y_e =$

1.3.3 Bifurcatie

Zoals reeds geïllustreerd werd in Opdracht 1.4.3, zullen DVn voor het modelleren van praktische problemen vaak afhangen van één of meerdere parameters

$$y' = f(t, y, p_1, p_2, \ldots, p_n)$$

waarbij $p_1, p_2, \ldots, p_n$ de n parameters van de DV voorstellen. Kleine schommelingen van de parameterwaarden zullen veelal geen grote invloed hebben op het richtingsveld, maar in sommige gevallen kunnen parameters de ligging, de stabiliteit en het aantal evenwichtspunten beïnvloeden hetgeen uiteraard een grote invloed heeft op het verloop van de oplossing(en) van de DV. Dit fenomeen wordt aangeduid met de term bifurcatie (*bifurcation*). Bifurcatiepunten (*bifurcation points*) zijn die parameterwaarden voor dewelke er een omslag in de stabiliteit van één of meerdere evenwichtspunten plaatsgrijpt en/of het aantal evenwichtspunten wijzigt.

Voorbeeld 1.4 Bifurcatiegedrag van een differentiaalvergelijking

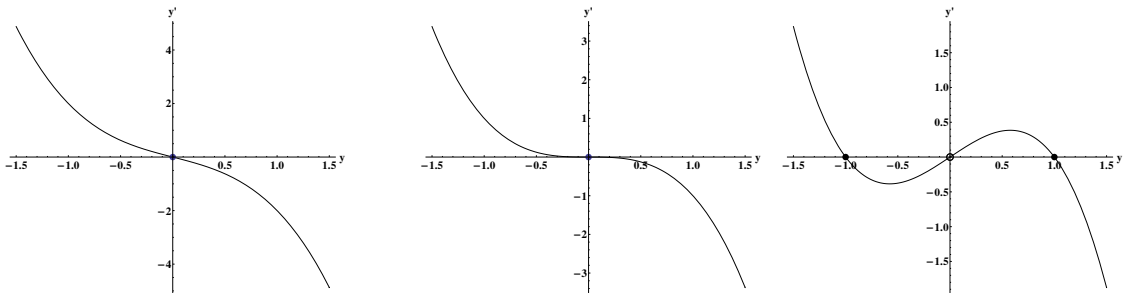
Beschouw de DV

$$y' = p\,y - y^3\,, \tag{1.4}$$

waarbij p een reële parameter voorstelt. De evenwichtspunten van deze DV kunnen gevonden worden uit

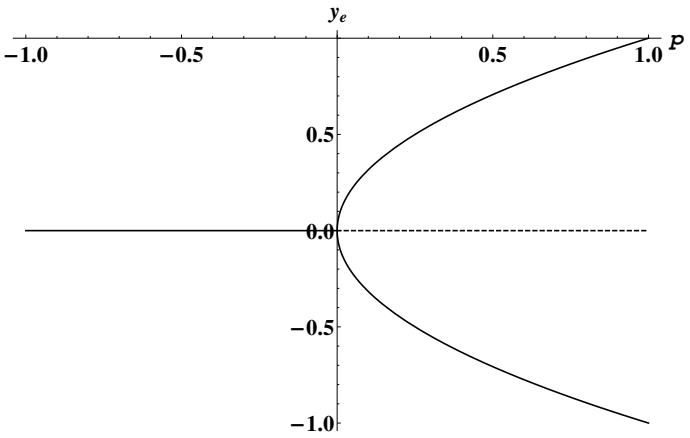
$$\begin{aligned} p\,y_e - y_e^3 &= 0 \\ y_e\,(p - y_e^2) &= 0\,, \end{aligned}$$

waaruit onmiddellijk volgt dat de DV ongeacht de waarde van p steeds één evenwichtspunt bezit, nl. $(y_e)_1 = 0$*, terwijl er nog twee bijkomende evenwichtspunten (*$(y_e)_{2,3} = \pm\sqrt{p}$*) bestaan indien* $p > 0$*. Figuur 1.6 visualiseert het rechterlid van de DV indien (a)* $p < 0$*, (b)* $p = 0$ *en (c)* $p > 0$ *en laat toe om de stabiliteit van de voorkomende evenwichtspunten te bepalen i.f.v. p. Zo is het duidelijk dat het evenwichtspunt* $(y_e)_1 = 0$ *asymptotisch stabiel is indien* $p \leq 0$*, terwijl hetzelfde evenwichtspunt onstabiel is voor strikt positieve p hetgeen in Fig. 1.6 werd weergegeven door middel van gesloten en open bolletjes, respectievelijk. Indien men de coördinaten van de evenwichtspunten plot t.o.v. p bekomt men een zogenaamd bifurcatiediagram (*bifurcation diagram*) waaruit alle informatie omtrent de bifurcatie kan afgeleid worden (Fig. 1.7). Onstabiele evenwichtspunten worden hierbij aangegeven met een streepjeslijn, terwijl (asymptotisch) stabiele evenwichtspunten worden aangeduid met een volle lijn. Aldus kan uit Fig. 1.7 afgeleid worden dat* $p = 0$ *een bifurcatiepunt is aangezien hier een wijziging in het gedrag van de evenwichtspunten plaatsvindt.*



(a) $p < 0$ (b) $p = 0$ (c) $p > 0$

Figuur 1.6: Mogelijke faseportretten van DV (1.4).



Figuur 1.7: Bifurcatiediagram van DV (1.4).

Verschillende bifurcatietypes kunnen worden onderscheiden, maar voor meer informatie hieromtrent verwijzen we naar gespecialiseerde literatuur (e.g. Greenberg, 1998; Ott, 1993; Strogatz, 1994).

Opdracht 1.5

1. *Beschouw de eerste-orde DV*

$$y' = y^2 + 4\,p^2 - 1\,,$$

waarbij p een reële parameter voorstelt.

(a) *Bepaal het (de) evenwichtspunt(en) van deze DV i.f.v. de reële parameter p.*

(b) *Schets het bifurcatiediagram en duid hierop de stabiliteit van de evenwichtspunten aan.*

2 Beschouw de eerste-orde DV*

$$y' = p - y - e^{-y}\,,$$

waarbij p een reële parameter voorstelt. Onderzoek het gedrag van het (de) evenwichtspunt(en) van deze DV i.f.v. p. Bepaal tevens het eventuele bifurcatiepunt. **TIP***: bemerk dat het rechterlid van de DV kan opgevat worden als een verschil van twee functies.*

2

De methode van Euler

Practica 2 en 3 hebben tot doel de lezer vertrouwd te maken met diverse numerieke methodes voor het oplossen van de eerste-orde DV

$$\frac{dy}{dt} = y' = f(t, y(t)) \tag{2.1}$$

over een t-interval $[t_0, t_n]$ en met beginvoorwaarde $y(t_0) = y_0$. We bespreken de methode van Euler (*Euler's method*), de Midpoint methode (een tweede-orde Runge-Kutta methode) en de klassieke vierde-orde Runge-Kutta methode (*fourth-order Runge-Kutta method*). Vooreerst worden de basismethodes uitvoerig besproken en wordt aangegeven hoe deze in MATLAB geïmplementeerd kunnen worden. Vervolgens trachten we d.m.v. enkele voorbeelden inzicht te verwerven in de gebruikte methodes. Verder onderzoeken we de voor- en nadelen van de verschillende methodes en controleren we de stabiliteit van de aangeleerde numerieke methodes.

De numerieke methodes die in deze nota's aan bod komen voor het benaderen van gewone DVn zijn gebaseerd op de geometrische interpretatie van DVn die in Practicum 1 werd besproken en werken steeds volgens het volgende principe:

- definitie van het oplossingsinterval $[t_0, t_n]$;
- discretisatie van $[t_0, t_n]$ in een eindig aantal punten t_k zodat $t_0 < t_1 < \ldots < t_{n-1} < t_n$;
- benadering van de oplossing in elk punt t_k van $[t_k, t_{k+1}]$ m.b.v. een rechte door het punt (t_k, y_k).

Niettemin de afstand tussen twee opeenvolgende punten t_k en t_{k+1} bij de meeste gesofisticeerde methodes wordt aangepast o.b.v. de kromming van de oplossing (e.g. Butcher, 2008), zullen we ons in deze nota's beperken tot een opdeling van het beschouwde tijdsinterval $[t_0, t_n]$ in n intervallen met **gelijke lengte** $\Delta t = (t_n - t_0)/n$, dewelke wordt aangeduid met de term stapgrootte (*step size*).

2.1 De methode van Euler

Deze methode gebruikt de raaklijn in het beginpunt t_k van elk interval $[t_k, t_{k+1}]$ om de exacte oplossing van de DV door het punt (t_0, y_0) te benaderen in de verschillende punten t_{k+1}. Beschouw hiertoe vooreerst het interval $[t_0, t_1]$. We benaderen $y(t_1)$ m.b.v. het beeld dat de raaklijn R_0 aan de exacte oplossing in t_0 bereikt op het tijdstip $t_1 = t_0 + \Delta t$ (Fig. 2.1(a)). Rekening houdend met het feit dat R_0 als vergelijking $y = f(t_0, y_0)\,(t - t_0) + y_0$ heeft, vinden we voor de benadering in t_1

$$y(t_1) \approx y_1 := y(t_0) + (t_1 - t_0)\; f(t_0, y(t_0)) = y_0 + \Delta t\, f(t_0, y_0) \tag{2.2}$$

Herhalen we de benadering gegeven door Vgl. (2.2) in de overige intervallen $[t_k, t_{k+1}]$, $k = 1, \ldots, n-1$, dan bekomen we uiteindelijk de recursiebetrekking:

$$y_{k+1} = y_k + \Delta t\, f(t_k, y_k)\,, \tag{2.3}$$

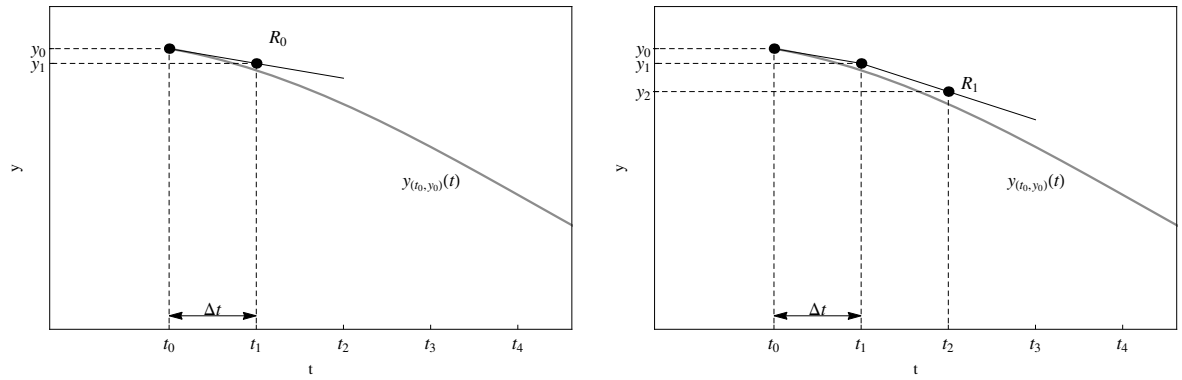
waarin y_k, $k = 1, \ldots, n$, de Eulerbenadering van $y(t_k)$ voorstelt. Gebruikmakend van de notatie

$$m_1 = f(t_k, y_k)$$

kan Vgl. (2.3) ook geschreven worden als

$$y_{k+1} = y_k + \Delta t\, m_1\,.$$

Ter illustratie toont Fig. 2.1(b) de bepaling van de tweede Eulerbenadering.



(a) Benadering van $y(t_1)$ (b) Benadering van $y(t_2)$

Figuur 2.1: Eulerbenaderingen voor $y(t_1)$ en $y(t_2)$ bekomen via de constructie van raaklijnen R_0 en R_1 door (t_0, y_0) en (t_1, y_1), respectievelijk.

Merk op dat de raaklijn aan een continue kromme K in een punt q slechts een goede benadering is voor K in een omgeving van q als de kromming van K in die omgeving niet te groot is. Je kan hierbij denken aan een lat die je op een bal legt, en een lat die je op de aarde legt. De

lat zal beter onze planeet benaderen. De kromming van de bal is immers veel groter dan de kromming van de aarde. Beperken we ons echter tot de onmiddellijke omgeving van q, dan mogen we aannemen dat we toch een relatief goede benadering zullen verkrijgen. Algemeen kunnen we besluiten dat, hoe groter de kromming van de exacte oplossing, hoe kleiner de stapgrootte moet zijn om een voldoende dichte benadering te bekomen. Concreet betekent dit dat y_1 een goede benadering is van $y(t_1)$ als de stapgrootte Δt voldoende klein gekozen wordt.

Opdracht 2.1

1. *Vervolledig de MATLAB-functie* `mijnEuler` *die de methode van Euler implementeert om de eerste-orde DV* $y' = f(t, y)$ *te benaderen over het interval* $[t_0, t_n]$ *in* `n` *stappen en met beginvoorwaarde* $y(t_0) = y_0$*. Je kan hiervoor gebruikmaken van de code in de m-file* `mijnEuler.m`*, die beschikbaar is op Minerva.*

Codefragment 2.1: mijnEuler

```
function [t, y] = mijnEuler(f, tinterval, y0, n)
% Benaderen van y' = f(t, y) over tinterval = [t0, tn]
% met beginwaarde y(t0) = y0 en stapgrootte dt = (tn - t0)/n
% Syntax: [t, y] = mijnEuler(f, tinterval, y0, n)
% Input:  f: fuction handle naar het rechterlid van de DV
%         tinterval = [tmin, tmax]: grenzen van de grafiek
%         y0 = beginwaarde
%         n = aantal stappen
% Output: t = tijdsvector
%         y = benadering van y' = f(t,y)

y = zeros(n+1, 1);       % pre-allocatie van y als KOLOMvector
y(1) = y0;               % y0 is het eerste element van
                         % de oplossingsvector y
dt = (tinterval(2) - tinterval(1))/n;
t = ...                  % maak hier de KOLOMvector t aan

for k = ...

 % schrijf hier een for-lus die de waarden y(k+1) berekent
 % op basis van t(k) en y(k)

end
end
```

De output van deze functie is een kolomvector `t` *die de verschillende tijdstippen* t_k *bevat en een kolomvector* `y` *die de corresponderende benaderingen op deze tijdstippen bevat. Noteer hieronder de* `for`*-lus waarmee je de functie* `mijnEuler` *hebt vervolledigd.*

```
for k =



end
```


Voorbeeld 2.1 Methode van Euler

Beschouw de volgende DV over het t-interval $[0, 4]$*:*

$$y' = \left(t^3 - t^2 - 7\,t - 5\right)\, e^t\,. \tag{2.4}$$

De oplossing van het beginwaardeprobleem $y(0) = 0$ *wordt gegeven door*

$$y\,(t) = \left(-6 + t - 4\,t^2 + t^3\right) e^t + 6\,.$$

Vooreerst werd het rechterlid van deze DV en de exacte oplossing geïmplementeerd in MATLAB en bewaard in de m-files `functieVB.m` *en* `exactVB.m`*, respectievelijk. De DV werd vervolgens benaderd bij verschillende stapgroottes.*

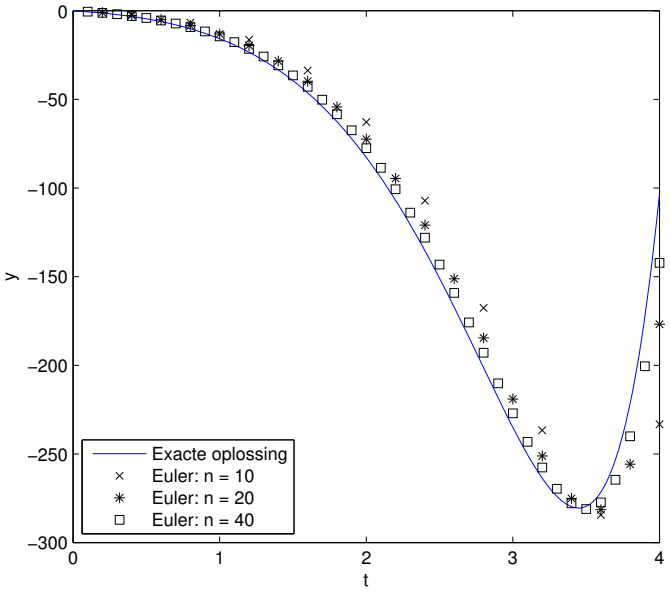
Om de Euler-benadering uit te voeren bij $n = 10$ *werd volgende instructie opgenomen in een script:*

```
[t, y] = mijnEuler(@functieVB, [0, 4], 0, 10);
```

waarna de exacte oplossing samen met de benadering bij $n = 10$ *in één figuur werd geplot d.m.v. de instructies*

```
figure
fplot(@exactVB, [0, 4]);
hold on
plot(t, y, 'x')
```

De aldus bekomen benaderingen bij $n = 10$*,* $n = 20$ *en* $n = 40$ *worden samen met de exacte oplossing voor het beschouwde beginwaardeprobleem weergegeven in Fig. 2.2. Deze figuur toont duidelijk dat de exacte oplossing beter benaderd wordt naarmate de stapgrootte* Δt *afneemt, en* n *dus toeneemt.*



Figuur 2.2: Exacte oplossing en Euler-benadering van DV (2.4) bij een verschillend aantal stappen n.

Opdracht 2.2

Beschouw de eerste-orde DV

$$y' = -\frac{3}{2}\cos t - y\,\frac{1}{2}\cos t \tag{2.5}$$

over het t-interval $[0, \pi]$ met beginvoorwaarde $y(0) = -4$.

1. *Implementeer het rechterlid van deze DV in MATLAB in een m-file* `functie2.m`.

2. *Plot het richtingsveld van deze DV. Werk hierbij over het gebied $[0, \pi] \times [-4, -2]$. Heeft deze DV één of meerdere evenwichtspunt(en)? Onderzoek hun stabiliteit.*

   $y_e =$

3. *Breidt het* `t`*-interval uit naar $[0, 3\pi]$ en ga de stabiliteit van het (de) evenwichtspunt(en) nogmaals na.*

4. *Bepaal met Mathematica de exacte oplossing van het beginwaardeprobleem.*

   $y(t) =$

   Implementeer deze in een m-file `exact2.m`.

5. *Ga na of je functie* `mijnEuler` *(zie Opdracht 2.1) correct werkt door, voor* $n = 10$*, de variabele* y *op te vragen in het instructievenster of door de werkruimte te raadplegen:*

```
>> y
y =
    -4.0000
    -3.8429
    -3.7170
    -3.6259
    -3.5681
    -3.5405
    -3.5405
    -3.5668
    -3.6191
    -3.6978
    -3.8020
```

6. *Plot bovenop het richtingsveld de exacte oplossing samen met de Eulerbenaderingen voor* $n = 5, 10, 25$ *en* 50*. Gebruik verschillende kleuren en vergeet niet dat benaderingen dienen gevisualiseerd te worden als discrete datapunten.*

Opdracht 2.3

Beschouw het beginwaardeprobleem

$$y' = \sqrt{y}, \qquad y(t_0) = y_0. \tag{2.6}$$

1. *Implementeer het rechterlid van deze DV in een m-file* `functie3.m`*.*

2. *Bepaal met Mathematica de unieke algemene oplossing van het beginwaardeprobleem:*

   $y_{(t_0,y_0)}(t) =$

3. *Welke exacte oplossing vind je met Mathematica voor het beginwaardeprobleem* $y(0) = 0$*?*

   $y(t) =$

4. *Is er niet nog een andere, triviale, oplossing die beantwoordt aan het beginwaardeprobleem? Zo ja, welke?*

$y(t) =$

5. *Bestaat er een unieke exacte oplossing door het punt* $(0, 0)$ *en hoe kun je dit verklaren in het licht van Stelling 2.2? Houdt Mathematica hier rekening mee?*

6. *Plot de gevonden exacte oplossingen en de Euler-benadering voor* $\Delta t = 0.05$ *op één figuur waarbij je werkt over het* t*-interval* $[0, 2]$*. Welke exacte oplossing wordt goed benaderd?*

7. *Waarom wordt net deze oplossing goed benaderd?*

8. *Hoe kun je er op een eenvoudige manier voor zorgen dat de tweede exacte oplossing benaderd wordt?*

9. *Voer deze benadering uit en plot ze samen met beide exacte oplossingen op één figuur.*

Vooraleer we meer gesofisticeerde methodes introduceren, is het belangrijk een beter inzicht te krijgen in de methode van Euler. In het bijzonder gaan we in het vervolg van dit practicum na of y_{k+1} een goede benadering is van $y(t_{k+1})$.

2.2 Euler in de praktijk

2.2.1 Betekenis van de Eulerbenaderingen

We hebben gezien dat de oplossing van een eerste-orde DV benaderd kan worden a.d.h.v. de recursiebetrekking gegeven door Vgl. (2.3). Om y_1 te berekenen gebruikt de methode van

Euler de raaklijn in $(t_0,\, y(t_0))$ aan de exacte oplossing $y(t)$ door (t_0, y_0). Vervolgens wordt y_2 bepaald door:

$$y_2 = y_1 + \Delta t\, f(t_1, y_1)\,.$$

Vermits y_1 slechts een benadering is van $y(t_1)$, geldt er meestal dat $y_1 \neq y(t_1)$. Bijgevolg, indien (t_1, y_1) niet op $y(t)$ ligt, zal $f(t_1, y_1)$ niet gelijk zijn aan de richtingscoëfficiënt van de raaklijn aan $y(t)$ in het punt $(t_1, y(t_1))$. We kunnen y_2 echter beschouwen als de eerste Euler-benadering van de DV over het interval $[t_1, t_n]$ met beginvoorwaarde $y(t_1) = y_1$. Algemeen is y_{k+1}, $k = 1, \ldots, n-1$, de eerste Euler-benadering van de DV over het interval $[t_k, t_n]$ met beginvoorwaarde $y(t_k) = y_k$.

Opdracht 2.4

1. *Beschouw DV* (2.5) *over het t-interval* $[0,\, \pi]$.

   (a) *Bepaal de unieke oplossing van deze DV bij generieke beginwaarden* (t_0, y_0) *met Mathematica en implementeer deze in een m-file* `exact2t0.m`.

   $y(t) =$

   (b) *Plot de Euler-benadering bij* $n = 10$ *en de exacte oplossing van deze DV op eenzelfde figuur voor het beginwaardeprobleem* $y(0) = -4$. *Verbind bovendien de punten van de Euler-benadering. Hiertoe kun je het laatste argument in de functie* `plot` *op* `'*-'` *instellen.*

   (c) *Welke waarde vind je voor de eerste Euler-benadering* (t_1, y_1)*? Vergelijk deze met de waarde van de exacte oplossing op* t_1*, die* $y(t_1)$ *is. Wat merk je op?*

   (d) *Wat verwacht je als je een Euler-benadering over het interval* $[t_1, \pi]$ *zou plotten met* $n = 9$ *en* (t_1, y_1) *als initiële conditie?*

2.* *Op het begintijdstip* $t = t_0$ *geldt dat* $y(t_0) = y_0$ *zodat we weten dat* (t_0, y_0) *een punt is van de grafiek van* $y(t)$.

   (a) *Geldt dit voor alle* (t_k, y_k)*? Motiveer je antwoord.*

(b) Is y_{k+1} dan nog een goede benadering van $y(t_{k+1})$? Wat kan er fout lopen?

2

(c) Bij welk soort functies treedt dit probleem nooit op?

2.2.2 Foutenanalyse

Naast de meetkundige afleiding zoals voorgesteld in Sectie 2.1, kan de methode van Euler ook gevonden worden uit de Taylorontwikkeling van $y(t + \Delta t)$:

$$y(t + \Delta t) = y(t) + \frac{\Delta t\, y'(t)}{1!} + \frac{\Delta t^2\, y''(t)}{2!} + \ldots + \frac{\Delta t^n\, y^{(n)}(t)}{n!} + \ldots . \tag{2.7}$$

Indien deze ontwikkeling wordt afgebroken na de derde term in het rechterlid, wordt een benadering bekomen voor $y(t + \Delta t)$:

$$\begin{aligned} y(t_{k+1}) &= y(t_k) + \Delta t\, f(t_k, y(t_k)) + c_k\, \Delta t^2\,, \\ &= y(t_{k-1}) + \Delta t\, f(t_{k-1}, y(t_{k-1})) + \Delta t\, f(t_k, y(t_k)) + c_{k-1}\, \Delta t^2 + c_k\, \Delta t^2\,, \\ &= \ldots \\ &= y(t_0) + \Delta t \sum_{j=0}^{k} f(t_j, y(t_j)) + \Delta t^2 \sum_{j=0}^{k} c_j\,, \end{aligned} \tag{2.8}$$

waarbij $c_j = y''(\bar{t}_j)/2$, voor een zekere $\bar{t}_j \in]t_j, t_{j+1}[$. Indien Δt voldoende klein is, verwachten we dat de bijdrage van de derde term in deze uitdrukking verwaarloosbaar klein wordt t.o.v. de eerste twee, zodat we kunnen schrijven

$$y_{k+1} = y_k + \Delta t\, f(t_k, y_k) \approx \ldots \approx y_0 + \Delta t \sum_{j=0}^{k} f(t_j, y_j)$$

wat een goede benadering is van $y(t_{k+1})$. Hebben alle constanten c_j hetzelfde teken, dan zal de benaderingsfout toenemen ($\sum_{j=0}^{k} c_j$ wordt dan groot). Treedt er echter een tekenwissel op, dan kunnen de constanten c_j in de som $\sum_{j=0}^{k} c_j$ elkaar (gedeeltelijk) compenseren. Het teken van de c_j's is eenvoudig te bepalen d.m.v. een tekenonderzoek van y''. Merk op dat de fout die we op deze manier bekomen de lokale fout (*local error*) is, dit is de afwijking tussen de exacte oplossing en de benadering in elk punt t_j van de benadering. De MATLAB-functie `functieverschil` laat toe om in elk van de beschouwde tijdstippen (kolomvector `t`) de

absolute fout te berekenen tussen de Euler-benadering (kolomvector `y`) en de exacte oplossing (inputvariabele `exact`):

Codefragment 2.2: functieverschil

```
function v = functieverschil(t, y, exact)
% Deze functie vergelijkt benadering y en de exacte oplossing y(t)
% in de punten t(k), en geeft de absolute waarde van de verschillen
% in deze punten terug in een kolomvector
% Syntax: v = functieverschil(t, y, exact)
% Input:  t = tijdsvector
%         y = benadering van y' = f(t,y)
%         exact: function handle naar de exacte oplossing van de DV
% Output: v = |benadering - exact|

v = abs(y - exact(t));  % exact(t) moet gevectoriseerd zijn!
end
```

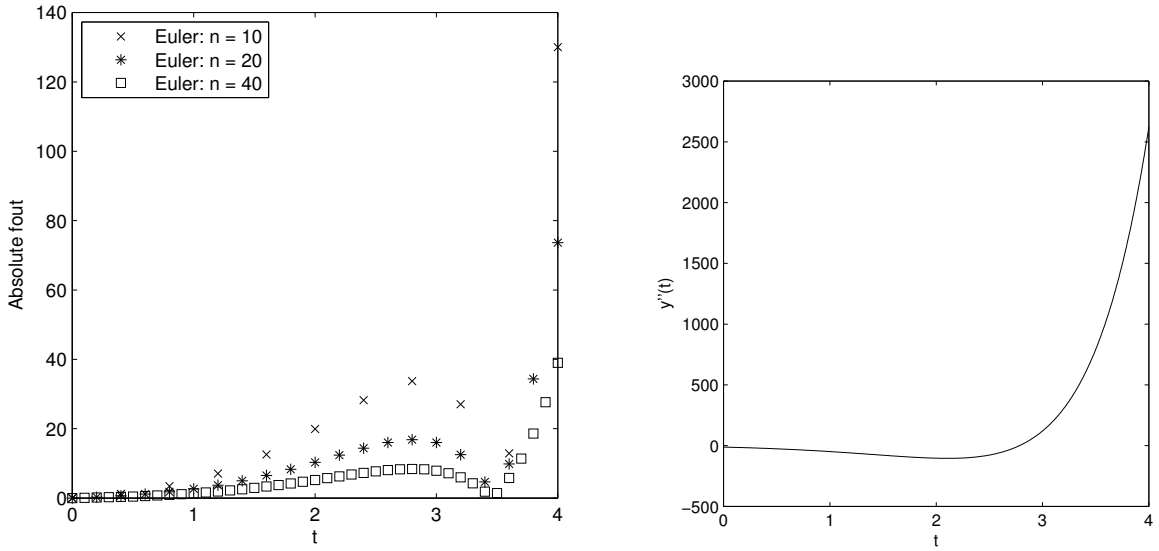
Deze functie kunnen we eenvoudig uitvoeren door de volgende instructies in een script op te nemen:

```
clc,
clear all
[t, y] = mijnEuler(@functie2, [0, 1], 2, 10); % benadering
fout = functieverschil(t, y, @exact2);        % absolute fout
figure
plot(t, fout, 'r*')
```

Voorbeeld 2.2 Foutenanalyse

Beschouw opnieuw DV (2.4) *over het t-interval* $[0, 4]$*, waarvan de exacte oplossing door het punt* $(0, 0)$ *gegeven wordt door* $y(t) = (-6 + t - 4t^2 + t^3)e^t + 6$*. Figuur 2.3(a) toont de absolute fouten tussen de Euler-benadering en de exacte oplossing berekend m.b.v. de functie* `functieverschil`*.*

Deze figuur illustreert duidelijk dat de absolute fout daalt naarmate het aantal stappen n groter wordt, hetgeen onze eerdere bevindingen o.b.v. Fig. 2.2 bevestigt en te verklaren is doordat de door de methode van Euler verwaarloosde termen uit Vgl. (2.8) *worden vermenigvuldigd met een factor* Δt^2*. Verder zien we dat de foutencurve een lokaal maximum bereikt ter hoogte van* $t = 2.75$ *waarna ze opnieuw daalt tot een lokaal minimum nabij* $t = 3.6$ *om tot slot exponentieel toe te nemen. Een tekenonderzoek van deze tweede afgeleide laat ons toe het patroon in Fig. 2.3(a) te verklaren. Deze tweede afgeleide wordt gegeven door*



(a) (b)

Figuur 2.3: Foutencurves bij verschillende stapgroottes voor DV (2.4) (a) en tweede afgeleide van $y_{(t_0,y_0)}(t)$ (b).

$y''(t) = e^t\,(-12 - 9\,t + 2\,t^2 + t^3)$ *en is afgebeeld in Fig. 2.3(b). In het beschouwde interval heeft* y'' *een enkelvoudig nulpunt bij* $t \approx 2.78$. *Bijgevolg veranderen de constanten* c_j *van teken op dat tijdstip waardoor de reeds geaccumuleerde fout zal gecompenseerd worden. Wanneer de fout gelijk wordt aan nul (nabij* $t = 3.60$*) is deze compensatie volledig en zal de fout opnieuw toenemen. Deze toename zal exponentieel verlopen vanwege de exponentiële functie die aanwezig is in de tweede afgeleide.*

Tot slot van dit practicum definiëren we de orde (*order*) van een numerieke methode. Hiertoe beschouwen we eerst de Taylor-ontwikkeling van $y(t + \Delta t)$:

$$y(t + \Delta t) = y(t) + \Delta t\, y'(t) + \frac{\Delta t^2}{2}\, y''(t) + \ldots + \frac{\Delta t^p}{p!} y^{(p)}(t)$$

Een numerieke methode heeft de orde p indien ze alle termen van de Taylorontwikkeling tot en met de term die de factor Δt^p bevat in rekening neemt. Met andere woorden, de orde is het aantal termen van de Taylorreeks die in rekening gebracht worden, verminderd met 1. Vermits de methode van Euler gebaseerd is op de eerste twee termen uit de Taylorreeksontwikkeling (Cfr. Vgl. (2.2)) is $p = 1$ en hebben we te maken met een eerste-orde methode.

Opdracht 2.5

Beschouw opnieuw de eerste-orde DV (2.5) *over het tijdsinterval* $[0, \pi]$ *met beginvoorwaarde* $y(0) = -4$.

1. *Maak een figuur die de foutencurves voor* $n = 5, 10, 25$ *bevat. Gebruik dezelfde kleuren en/of markeertypes die je gebruikte in Opdracht 2.2.*

2. *Hebben alle* $c_j = y''(\bar{t}_j)/2$ *hetzelfde teken? Welk effect heeft dit op de foutencurve?*

3. *Wat verwacht je indien we het beschouwde interval vergroten tot* $[0, 2\pi]$ *(met dezelfde stapgrootte)?*

3

De methodes van Runge en Kutta

In dit practicum gaan we op zoek naar alternatieve methodes om de oplossing van de eerste-orde DV $y' = f(t, y)$ over het interval $[t_0, t_n]$ met beginvoorwaarde $y(t_0) = y_0$ numeriek te benaderen. Om de berekeningen en implementatie niet onnodig te compliceren, maken we opnieuw gebruik van een vaste stapgrootte Δt.

De methode van Euler (Practicum 2) is mathematisch gebaseerd op slechts de eerste twee termen uit de Taylorreeksontwikkeling:

$$\begin{aligned} y(t_{k+1}) = y(t_k + \Delta t) &= y(t_k) + \Delta t\, y'(t_k) + \frac{1}{2!}\Delta t^2\, y''(t_k) + \ldots \\ &= y(t_k) + \Delta t\, f(t_k, y(t_k)) + \frac{1}{2!}\Delta t^2\, f'(t_k, y(t_k)) + \ldots \,. \end{aligned} \tag{3.1}$$

De meest voor de hand liggende manier om de methode van Euler te verbeteren, bestaat er bijgevolg in meerdere termen uit de Taylorreeks te beschouwen. Het aantal opgenomen termen zal de accuraatheid van de methode bepalen. Vervangen we in Vgl. (3.1) $y(t_k)$ door de benaderde waarde y_k, dan vinden we uiteindelijk de recursiebetrekking

$$\begin{aligned} y(t_{k+1}) \approx y_{k+1} = y_k + \Delta t\, f(t_k, y_k) + \frac{1}{2!}\Delta t^2\, f'(t_k, y_k) + \ldots \\ + \frac{1}{N!}\Delta t^N\, f^{(N-1)}(t_k, y_k), \qquad N \in \mathbb{N}\,. \end{aligned} \tag{3.2}$$

In de praktijk zijn de afgeleiden van $f(t, y)$ echter vaak moeilijk te berekenen. De Runge-Kutta methodes werden ontwikkeld om deze problemen te vermijden. Ze maken gebruik van diverse evaluaties van $f(t, y)$ om bovenstaande eindige reeks te benaderen. Net zoals bij de methode van Euler is de orde van een Runge-Kutta methode gelijk aan het aantal termen van de Taylorreeks die in rekening gebracht worden, verminderd met 1.

3.1 De Runge-Kutta methodes

3.1.1 De midpoint-methode

De methode van Euler wordt gegeven door de recursiebetrekking

$$y_{k+1} = y_k + \Delta t\, f(t_k, y_k)\,. \tag{3.3}$$

In deze methode wordt $y(t_{k+1})$ benaderd m.b.v. de raaklijn aan de oplossing $y_{(t_k,y_k)}(t)$ in $t = t_k$, zijnde het beginpunt van elk deelinterval $[t_k,\ t_{k+1}]$. Merk op dat hierbij wordt verondersteld dat $y_{(t_k,y_k)}(t) \approx y_{(t_k,y(t_k))}(t)$.

We wensen nu $y(t_{k+1})$ te benaderen a.d.h.v. een rechte door (t_k, y_k) die beter bij de exacte oplossing aansluit. Stellen we de rechte B door (t_k, y_k) waarop het te benaderen punt ligt algemeen voor als:

$$B: \quad y = y_k + m\,(t - t_k)\,, \tag{3.4}$$

waarbij m een nog te bepalen richtingscoëfficiënt voorstelt, dan is voor de methode van Euler m niets anders dan de richtingscoëfficiënt van de raaklijn aan $y_{(t_k,y_k)}(t)$ in het punt (t_k, y_k). Echter, meer algemeen kan m opgevat worden als een gewogen som[1] van richtingscoëfficiënten:

$$m = \sum_{j=1}^{N} w_j\, m_j$$

waarbij $\sum_{j=1}^{N} w_j = 1$ en N het aantal punten waarin de richting van de exacte oplossing wordt beschouwd, voorstelt. Deze richtingscoëfficiënten trachten het gedrag van de exacte oplossing in de omgeving van het punt (t_k, y_k) weer te geven. Naarmate we dit gedrag beter kunnen vervatten in m valt het te verwachten dat de rechte door (t_k, y_k) de exacte oplossing beter zal benaderen.

Bij een tweede-orde Runge-Kutta-methode is $N = 2$, zodat bij de bepaling van y_{k+1} met de helling van de exacte oplossing in twee punten in het interval $[t_k, t_{k+1}]$ rekening wordt gehouden. Het eerste punt kiezen we in het begin van het interval $[t_k, t_{k+1}]$ en kunnen we dus noteren als $(t_k, y(t_k))$. Het tweede punt is vrij te kiezen in het interval $]t_k, t_{k+1}]$ en noteren we als $(t_k + \alpha\,\Delta t, y(t_k + \alpha\,\Delta t))$ met $\alpha \in]0, 1]$. De richting van de exacte oplossing $y(t)$ in deze punten wordt gegeven door respectievelijk $y'(t_k) = f(t_k, y(t_k))$ en $y'(t_k + \alpha\,\Delta t) = f(t_k + \alpha\,\Delta t, y(t_k + \alpha\,\Delta t))$. Vermits $y(t)$ niet gekend is, kunnen we deze richtingscoëfficiënten niet berekenen, maar dienen we ze te benaderen. Aldus zullen we $y(t_k)$ benaderen door y_k en kunnen we de waarde voor $y(t_k + \alpha\,\Delta t)$ benaderen m.b.v. de methode

[1] Een gewogen som van een eindig aantal waarden $x_1, \ldots, x_n$ is een lineaire combinatie $\sum_{i=1}^{n} b_i\, x_i$ waarbij $\sum_{i=1}^{n} b_i = 1$ met $b_i \geq 0$.

van Euler: $y(t_k + \alpha\,\Delta t) \approx y_k + \alpha\,\Delta t\, f(t_k, y_k)$.
Bijgevolg kunnen we schrijven:

$$y'(t_k) \approx f(t_k, y_k)$$
$$y'(t_k + \alpha\,\Delta t) \approx f(t_k + \alpha\,\Delta t, y_k + \alpha\,\Delta t\, f(t_k, y_k))$$

zodat

$$m = w_1\, f(t_k, y_k) + w_2\, f(t_k + \alpha\,\Delta t, y_k + \alpha\,\Delta t\, f(t_k, y_k))\ ,$$

waarbij $w_1 + w_2 = 1$. Gelet op Vgl. (3.4) vinden we

$$\begin{aligned} y_{k+1} &\approx y_k + m\,(t_{k+1} - t_k) = y_k + m\,\Delta t \\ &= y_k + \Delta t\,\left[w_1\, f(t_k, y_k) + w_2\, f(t_k + \alpha\,\Delta t, y_k + \alpha\,\Delta t\, f(t_k, y_k))\right]. \end{aligned} \tag{3.5}$$

Deze bevat nog drie vrijheidsgraden: α, w_1 en w_2. Door deze drie parameters op gepaste wijze te kiezen, kunnen we ervoor zorgen dat Vgl. (3.5) een tweede-orde benaderingsmethode is. Daartoe dienen we deze vergelijking in de gedaante van Vgl. (3.2) te brengen. We gebruiken hiervoor de Taylorbenadering voor een functie in twee veranderlijken:

$$f(x + \Delta x, y + \Delta y) \approx f(x, y) + \Delta x\,\frac{\partial f}{\partial x}(x, y) + \Delta y\,\frac{\partial f}{\partial y}(x, y)\,.$$

Toepassing van deze benadering op Vgl. (3.5) levert:

$$\begin{aligned} y_{k+1} &= y_k + \Delta t\, w_1\, f(t_k, y_k) + \Delta t\, w_2\, f(t_k + \alpha\,\Delta t\,, y_k + \alpha\,\Delta t\, f(t_k, y_k)) \\ &\approx y_k + \Delta t\, w_1\, f(t_k, y_k) + \Delta t\, w_2\,\left[f(t_k, y_k) + \alpha\,\Delta t\,\frac{\partial f}{\partial t}(t_k, y_k) + \alpha\,\Delta t\, f(t_k, y_k)\frac{\partial f}{\partial y}(t_k, y_k)\right] \\ &= y_k + \Delta t\,\left(w_1 + w_2\right)\, f(t_k, y_k) + \alpha\,\Delta t^2\, w_2\,\left[\frac{\partial f}{\partial t}(t_k, y_k) + \frac{\partial f}{\partial y}(t_k, y_k)\, f(t_k, y_k)\right] \\ &\approx y_k + \Delta t\,\left(w_1 + w_2\right)\, f(t_k, y_k) + \alpha\,\Delta t^2\, w_2\,\left[\frac{\partial f}{\partial t}(t_k, y\,(t_k)) + \frac{\partial f}{\partial y}(t_k, y\,(t_k))\, y'\,(t_k)\right] \end{aligned}$$

Houden we nu rekening met de totale differentiaal van een functie van twee veranderlijken, dan verkrijgen we

$$y_{k+1} \approx y_k + \Delta t\,\left(w_1 + w_2\right)\, f(t_k, y_k) + \alpha\,\Delta t^2\, w_2\,\frac{d\,f}{d\,t}(t_k, y\,(t_k))\,. \tag{3.6}$$

Opdat het rechterlid van Vgl. (3.6) zou samenvallen met het rechterlid van de Taylorreeksontwikkeling gegeven door Vgl. (3.2), dient:

$$\begin{cases} w_1 + w_2 = 1\,, \\ \qquad \alpha\, w_2 = \dfrac{1}{2}\,. \end{cases} \tag{3.7}$$

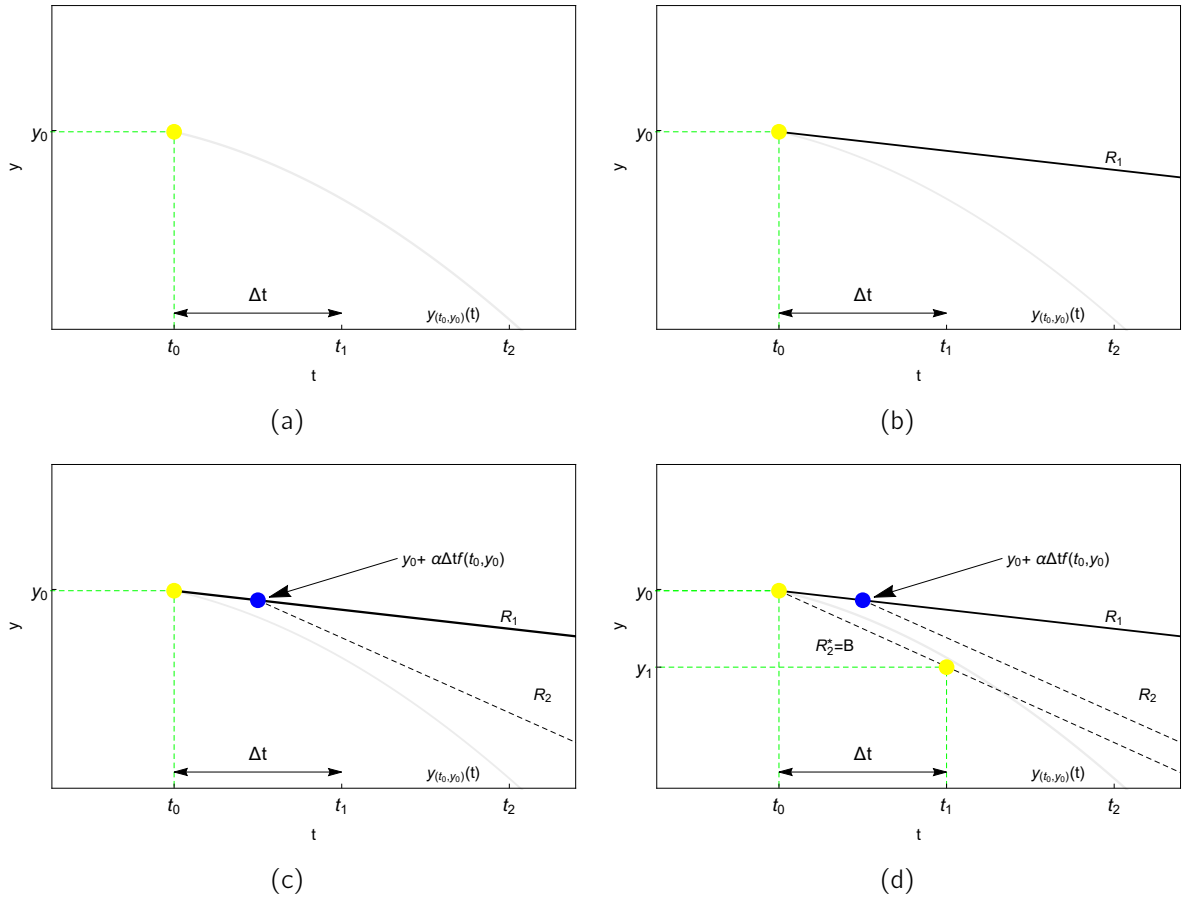
De midpoint-methode is een speciale tweede-orde Runge-Kutta-methode waarbij het tweede punt in het midden van het deelinterval $[t_k, t_{k+1}]$ gekozen wordt $\left(\alpha = \frac{1}{2}\right)$. Gelet op Stelsel (3.7) moet in dat geval $w_2 = 1$ en $w_1 = 0$ zodat Vgl. (3.5), gebruikmakende van de notatie

$$\begin{aligned} m_1 &= f(t_k, y_k)\,, \\ m_2 &= f(t_k + \frac{1}{2}\,\Delta t, y_k + \frac{1}{2}\,\Delta t\ m_1)\,, \end{aligned} \tag{3.8}$$

voor de midpoint-methode, vereenvoudigd kan worden tot:

$$y_{k+1} = y_k + \Delta t\, m_2\,. \tag{3.9}$$

Deze vergelijking kunnen we meetkundig interpreteren. In Vgl. (3.8) stelt $f(t_k, y_k)$, net zoals bij de methode van Euler, de helling van de raaklijn R_1 aan $y_{(t_k,y_k)}(t)$ in t_k voor. Uit Practicum 2 weten we dat we $f(t_k + \alpha\,\Delta t, y_k + \alpha\,\Delta t\, f(t_k, y_k))$ ook kunnen opvatten als de helling van de raaklijn R_2 aan $y_{(t_k+\alpha\,\Delta t, y_k+\alpha\,\Delta t\, f(t_k,y_k))}(t)$ in $t_k + \alpha\,\Delta t$. De uiteindelijke benadering (3.9) ligt dan op de rechte R_2^* die door (t_k, y_k) gaat en evenwijdig is met R_2. Figuur 3.1 geeft de werkwijze van midpoint-methode grafisch weer voor de bepaling van y_1.



Figuur 3.1: Illustratie van een tweede-orde Runge-Kutta-methode.

Uit praktische voorbeelden blijkt dat deze werkwijze meestal een meer gebalanceerde benadering oplevert dan de methode van Euler. Dit kan beredeneerd worden o.b.v. het feit dat

het voor een stijgende of dalende curve met uniforme kromming (te verwachten over een minimaal interval Δt) beter is een richtingscoëfficiënt halverwege het interval te bepalen, dan een richtingscoëfficiënt te gebruiken aan de rand van het interval. De richtingscoëffiënten aan de randen van het interval zullen bij voldoende kleine Δt namelijk het minimum of maximum zijn van de richtingscoëfficiënten over het beschouwde interval, zoals geïllustreerd in Figuur 3.1.

Opdracht 3.1

1. *Implementeer de midpoint-methode door de functie* `mijnEuler.m` *op correcte wijze aan te passen en op te slaan in een m-file* `mijnMidpoint.m`.

2. *Beschouw opnieuw DV* (2.5) *voorgesteld in Opdracht 2.2:*

$$y' = -\frac{3}{2}\cos t - y\,\frac{1}{2}\cos t\,,$$

over het t-interval $[0,\, \pi]$ *met beginvoorwaarde* $y(0) = -4$.

   (a) *Plot de exacte oplossing en de benadering bekomen met de midpoint-methode* ($n = 25$) *op één figuur.*

   (b) *Plot op een tweede figuur de foutencurve. Wat merk je op indien je deze curve vergelijkt met de curves bekomen in Opdracht 2.5?*

3.1.2 De klassieke vierde-orde methode

We kunnen trachten de precisie van de numerieke benadering nog verder op te drijven door hogere-orde Taylorreeksbenaderingen te beschouwen. De meest gebruikte Runge-Kutta-methode is de klassieke vierde-orde Runge-Kutta methode (RK4-methode). Vertrekpunt bij deze benadering zijn de twee richtingscoëfficiënten m_1 en m_2 uit de midpoint-methode. De RK4-methode beschouwt twee extra punten waarin de richtingscoëfficiënt van de raaklijn aan $y(t)$ geëvalueerd wordt.

Vooreerst wordt de richtingscoëfficiënt m_2 in $t_k + \frac{\Delta t}{2}$ herberekend. Dit kan gebeuren door $y'\left(t_k + \frac{\Delta t}{2}\right) = f\left(t_k + \frac{\Delta t}{2}, y\left(t_k + \frac{\Delta t}{2}\right)\right)$ te benaderen door $m_3 = f\left(t_k + \frac{\Delta t}{2}, y_k + \frac{\Delta t}{2}\, m_2\right)$. Hierbij werd de rechte door (t_k, y_k) met richtingscoëfficiënt m_2 gebruikt om $y\left(t_k + \frac{\Delta t}{2}\right)$ te benaderen. Het vierde en laatste punt kiezen we op het einde van het interval $[t_k, t_{k+1}]$. De

exacte oplossing $y(t_{k+1})$ kunnen we benaderen door de rechte door (t_k, y_k) met richtingscoëfficiënt m_3: $y(t_{k+1}) \approx y_k + \Delta t\, m_3$. Bijgevolg geldt $y'(t_{k+1}) = f(t_{k+1}, y(t_{k+1})) \approx m_4 = f(t_k + \Delta t, y_k + \Delta t\, m_3)$. Deze vier richtingscoëfficiënten worden geaggregeerd tot de uiteindelijke richtingscoëfficiënt m waarbij

$$w_1 = \frac{1}{6}, \qquad w_2 = \frac{1}{3}, \qquad w_3 = \frac{1}{3}, \qquad w_4 = \frac{1}{6},$$

zodat $m = \frac{1}{6} m_1 + \frac{1}{3} m_2 + \frac{1}{3} m_3 + \frac{1}{6} m_4$.
Expliciet wordt de RK4 methode gegeven door

$$y_{k+1} = y_k + \left(\frac{1}{6}m_1 + \frac{1}{3}m_2 + \frac{1}{3}m_3 + \frac{1}{6}m_4\right)\Delta t\,,$$

waarin

$$\begin{aligned} m_1 &= f(t_k, y_k)\,, \\ m_2 &= f\left(t_k + \frac{\Delta t}{2}, y_k + \frac{\Delta t}{2}\, m_1\right)\,, \\ m_3 &= f\left(t_k + \frac{\Delta t}{2}, y_k + \frac{\Delta t}{2}\, m_2\right)\,, \\ m_4 &= f\left(t_k + \Delta t, y_k + \Delta t\, m_3\right)\,. \end{aligned}$$

Opdracht 3.2

1. *Implementeer de RK4-methode door de functie* `mijnMidpoint.m` *op correcte wijze aan te passen en op te slaan in een m-file* `mijnRK4.m`.

2. *Beschouw opnieuw DV* (2.5) *voorgesteld in Opdracht 2.2 over het t-interval* $[0, \pi]$ *met beginvoorwaarde* $y(0) = -4$.

   (a) *Plot de exacte oplossing en de benadering bekomen met de RK4-methode* ($n = 25$) *op één figuur.*

   (b) *Plot op een tweede figuur de foutencurve. Wat merk je op indien je deze curve vergelijkt met de curves bekomen in Opdracht 2.5 en Opdracht 3.1?*

3.1.3 Meerstappenmethodes

De tot hiertoe bestudeerde methodes zijn alle 1-stapmethodes vermits zij voor de bepaling van y_{k+1} slechts gebruik maken van de benadering op het voorgaande benaderingstijdstip t_k, d.i. y_k. Een intuïtieve uitbreiding van dergelijke methodes bestaat erin om niet enkel y_k te gebruiken ter bepaling van y_{k+1}, maar ook de eerder gemaakte benaderingen. Bijgevolg kunnen we een dergelijke methode algemeen schrijven als:

$$\sum_{j=0}^{N} a_j \, y_{k+1-j} = \sum_{j=0}^{N} b_j \, f\left(t_{k+1-j},\, y_{k+1-j}\right) , \tag{3.10}$$

waarbij N niets anders is dan het aantal stappen die de methode beschouwt, en a_j en b_j methode-specifieke wegingscoëfficiënten voorstellen. Een N-stappenmethode kan slechts toegepast worden indien de benaderingen op N voorgaande tijdstappen gekend zijn. Het is het duidelijk dat initieel naast de beginvoorwaarde $y(t_0) = y_0$ tevens de benaderingen $y_1, y_2, \ldots, y_{N-1}$ gekend dienen te zijn. Deze benaderingen kunnen gemakkelijk gevonden worden door gebruik te maken van één van de eerder bestudeerde 1-stapmethoden.

De bekendste meerstappenmethoden zijn de Adams–Moulton, Adams–Bashforth, Milne–Simpson en Nyström methodes. Een veelgebruikte Adams–Bashforth 2-stappenmethode wordt bijvoorbeeld gegeven door:

$$y_{k+1} = y_k + \frac{\Delta t}{2} \left[3\, f\left(t_k, y_k\right) - f\left(t_{k-1}, y_{k-1}\right)\right] . \tag{3.11}$$

Opdracht 3.3

1. *Beschouw de Adams–Bashforth methode gegeven door Vgl.* (3.11)*. Implementeer deze methode door de functie* `mijnRK4.m` *op correcte wijze aan te passen en op te slaan in een m-file* `mijnAB.m`*. Noteer hieronder jouw aanvullingen.*

```
y(2) =

for k =



end
```

2. *Beschouw opnieuw DV* (2.5) *voorgesteld in Opdracht 2.2 over het t-interval* $[0, \pi]$.

   (a) *Plot de exacte oplossing en de benadering bekomen met de Adams-Bashforth methode* ($n = 25$) *op één figuur.*

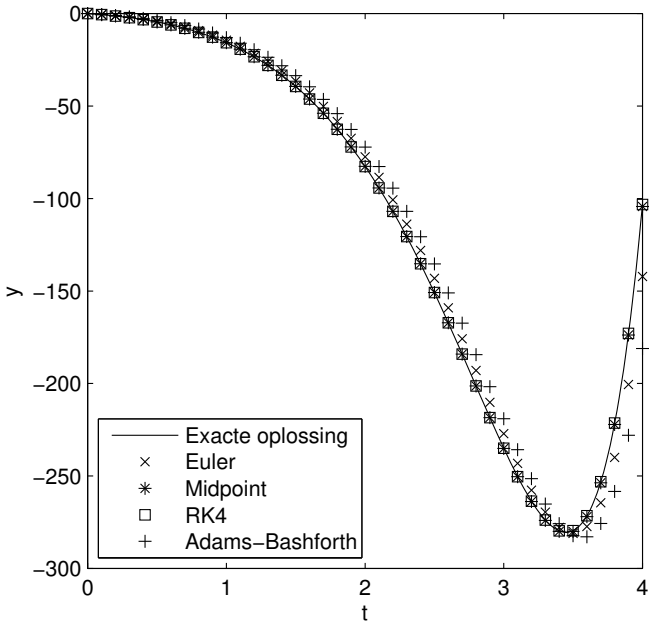
(b) Plot op een tweede figuur de foutencurve. Wat merk je op indien je deze vergelijkt met de foutencurves die je bekwam met de reeds geïmplementeerde methodes?

3.1.4 Vergelijken van numerieke methodes

Beschouw opnieuw DV (2.4) over het t-interval [0, 4] waarvan we in Practicum 2 de oplossing benaderd hebben met de methode van Euler:

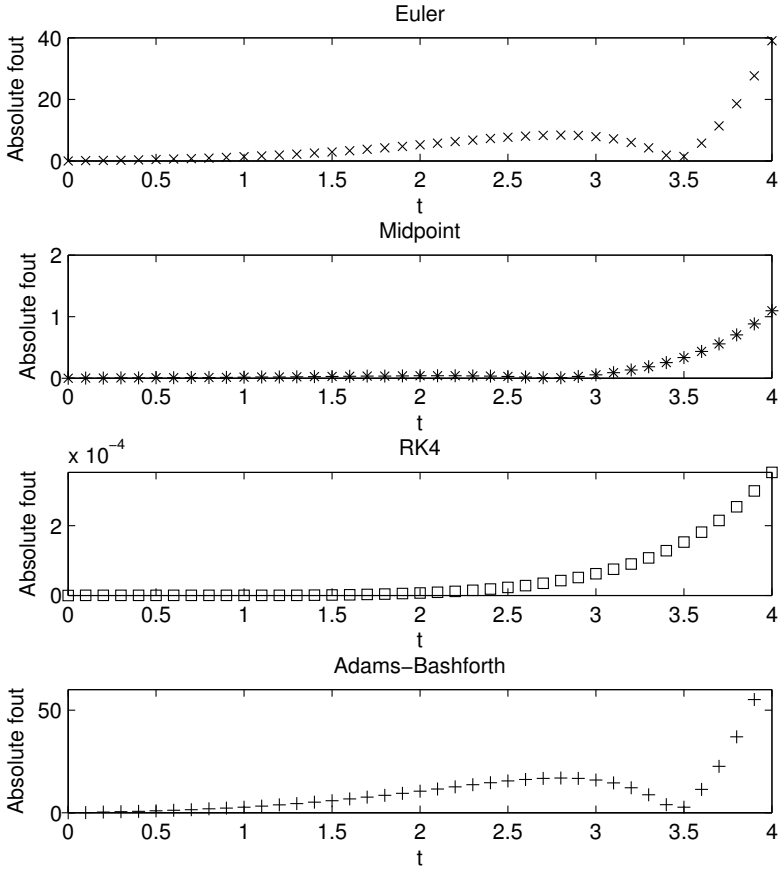
$$y' = \left(t^3 - t^2 - 7\,t - 5\right)\, e^t\,. \tag{3.12}$$

We wisten reeds dat de oplossing van het beginwaardeprobleem $y(0) = 0$ hierbij gegeven wordt door $y\,(t) = (-6 + t - 4\,t^2 + t^3)\,e^t + 6$. DV (3.12) werd benaderd met de besproken numerieke methodes voor $n = 40$ en het resultaat hiervan wordt weergegeven in Fig. 3.2. Uit deze figuur blijkt duidelijk dat de methodes besproken in dit practicum de exacte oplossing beter benaderen dan de methode van Euler. Dit wordt bevestigd door Fig. 3.3 die de foutencurve weergeeft voor elk van de benaderingsmethodes.



Figuur 3.2: Exacte oplossing en benadering van DV (3.12) voor $n = 40$.

Om na te gaan welke van de benaderingsmethodes de beste is, kunnen de functies `mean` en `max` gebruikt worden die, respectievelijk, de gemiddelde en maximale waarde die voorkomt



Figuur 3.3: Foutencurves voor DV (3.12) voor $n = 40$.

binnen een vector bepalen. Zij bijvoorbeeld `fout` de vector waarin de absolute fout op elk tijdstip van de benadering werd benaderd, dan kunnen de maximale en gemiddelde absolute fouten gevonden worden m.b.v. de volgende instructies:

```
maxFout = max(fout);
gemFout = mean(fout);
```

Opdracht 3.4

Beschouw opnieuw DV (2.5) *voorgesteld in Opdracht 2.2 over het t-interval* $[0, \pi]$.

1. *Plot de benaderingen bekomen met de methode van Euler, de midpoint-, de RK4- en de Adams–Bashforth-methode* ($n = 25$) *en de exacte oplossing op één figuur.*

2. *Plot eveneens de foutencurves op een gezamenlijke figuur. Op een derde figuur plot je de foutencurves in vier afzonderlijke subplots.*

3. *Welke benadering is de beste?*

4. *Vergroot nu het t-interval tot* $[0, 3\pi]$ *en stel hierbij* $n = 50$*. Welke methode is nu de beste?*

5. *Bepaal de kleinste* n_{Euler} *waarvoor de methode van Euler* **gemiddeld** *beter is dan de midpoint-methode, indien* $n_{midpoint} = 25$ *en we opnieuw werken over het t-interval* $[0, \pi]$*? Doe dit door* n_{Euler} *eerst met grote stappen te verhogen, en zoek daarna de exacte waarde. Bepaal tevens de grootste* $\Delta t_{midpoint}$ *waarvoor de* **maximale fout** *van de midpoint-methode kleiner is dan de maximale fout van de RK4-methode indien* $n_{RK4} = 25$*.*

   $n_{Euler} =$

   $\Delta t_{midpoint} =$

3.2 MATLAB-functies voor differentiaalvergelijkingen

Zoals reeds aangehaald, werken geavanceerde algoritmen met een variabele stapgrootte die o.b.v. de kromming van de oplossing zodanig wordt aangepast dat de benaderingsfouten worden geminimaliseerd. Daarenboven beperken zulke algoritmen zich niet tot één enkele benaderingsmethode, maar worden er veelal diverse methodes gecombineerd. MATLAB beschikt over drie functies, nl. `ode23`, `ode45` en `ode113` om de eerste-orde DV $y' = f(t, y)$ te benaderen gebruikmakend van een variabele stapgrootte (Tabel 3.1). Hierbij combineert `ode23` tweede- en derde-orde Runge-Kutta methodes en gebruikt `ode45` een combinatie van vierde- en vijfde-orde Runge-Kutta methodes. Beide methodes zijn 1-stapmethodes. De functie `ode113` is een meerstappenmethode.

Om de `ode23`-benadering van een DV $y' = f(t, y)$ over het interval $[t_0, t_n]$ en met beginvoorwaarde $y(t_0) = y_0$ te bepalen, gebruik je de volgende instructie:

```
[t, y] = ode23(@f, [t0, t_n], y0);
```

Hierbij is `f` de implementatie van $f(t, y)$ over het beschouwde interval. De instructies voor `ode45` en `ode113` zijn analoog.

Tabel 3.1: Overzicht van een aantal MATLAB-functies voor het benaderen van DVn.

Functie	Nauwkeurigheid	Gebruik
`ode45`	gemiddeld	meeste DVn
`ode23`	laag	problemen waarbij een vrij grote fout is toegestaan op de benadering; het patroon van de oplossing is belangrijker dan de effectieve waarden
`ode113`	laag tot hoog	rekenintensieve problemen

Opdracht 3.5

Beschouw nogmaals DV (2.5) *voorgesteld in Opdracht 2.2 over het t-interval* $[0, \pi]$. *Benader de oplossing over dit interval m.b.v.* `ode23`, `ode45` *en* `ode113`. *Welke van deze MATLAB-functies levert gemiddeld de beste benadering op? Noteer in de tabel hieronder tevens het aantal stappen dat elk van de drie MATLAB-functies gebruikt voor het benaderen van de oplossing.*

	gemiddelde fout	aantal stappen
`ode23`		
`ode45`		
`ode113`		

3.3 Stabiliteit van numerieke methodes

Bij de studie van DVn wordt de term stabiliteit (*stability*) vaak diverse betekenissen toebedeeld. In Practicum 1 belichtten we reeds de stabiliteit van evenwichtspunten, maar de term stabiliteit kan in de context van DVn tevens refereren naar de numerieke stabiliteit van de benaderingsmethodes. Numerieke methodes noemen we stabiel (*stable*) als de afwijkingsfouten, ontstaan in een welbepaald stadium van het proces, niet toenemen tijdens latere stadia. Bij de stabiliteitsanalyse van een benaderingsmethode volstaat het meestal de benaderingsfouten te onderzoeken voor een eenvoudig probleem zoals $y' = \lambda\, y$. Is de gebruikte methode onstabiel voor dit eenvoudig probleem, dan is de kans groot dat dit eveneens geldt voor meer complexe problemen.

Bij het numeriek oplossen van DVn worden we veelvuldig geconfronteerd met stabiliteitsproblemen die kunnen ingedeeld worden in twee categorieën: slecht geconditioneerde DVn (*ill-posed DEs*) en stijve DVn (*stiff DEs*).

3.3.1 Slecht geconditioneerde differentiaalvergelijkingen

Een DV $y' = f(t, y)$ is slecht geconditioneerd indien er er een stel beginvoorwaarden (t_0, y_0) bestaat, met corresponderende exacte oplossing $y(t)$, zodat elke oplossing door een naburig punt $(t_0, y_0 + \epsilon)$, $\epsilon \in \mathbb{R}_0$, zich vrij snel van $y(t)$ zal verwijderen. Bijgevolg zullen voor sommige beginvoorwaarden de benaderingsfouten steeds toenemen, ongeacht de gebruikte numerieke methode. Voor slecht geconditioneerde DVn bestaat er dus minstens één stel 'slechte beginvoorwaarden'.

Voorbeeld 3.1 Slecht geconditioneerde differentiaalvergelijking

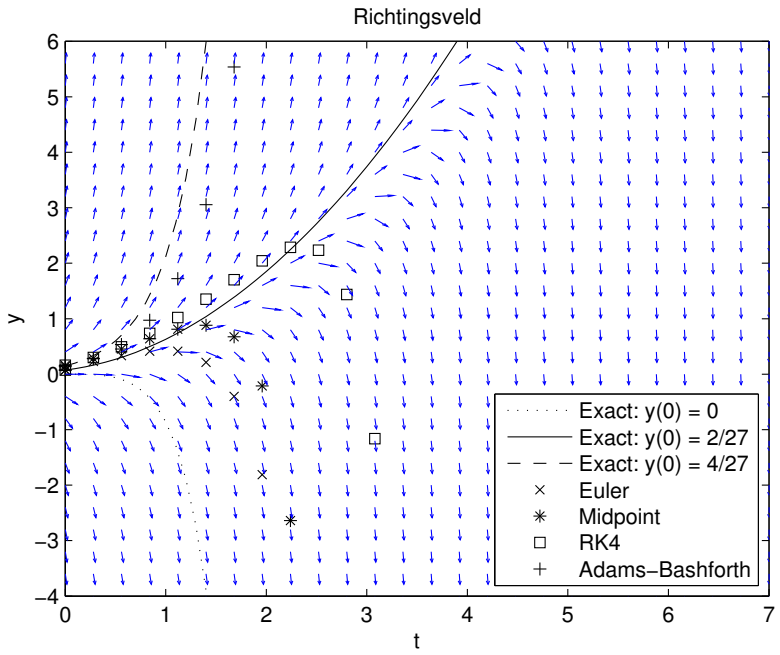
Beschouw de DV:

$$y' = 3\,y - t^2\,. \tag{3.13}$$

De algemene oplossing wordt gegeven door

$$y(t) = C\,e^{3\,t} + \frac{t^2}{3} + \frac{2\,t}{9} + \frac{2}{27}\,.$$

Samen met de beginvoorwaarde $y(0) = \frac{2}{27}$ *vinden we dat* $C = 0$, *zodat* $y(t) = \frac{t^2}{3} + \frac{2\,t}{9} + \frac{2}{27}$. *Figuur 3.4 geeft de exacte oplossing samen met de benaderingen voor dit beginwaardeprobleem. Tevens wordt het richtingsveld van deze DV en de exacte oplossing voor twee andere beginwaardeproblemen weergegeven.*



Figuur 3.4: Richtingsveld, exacte oplossingen bij verschillende beginvoorwaarden en numerieke benaderingen voor $y(0) = \frac{2}{27}$ met $n = 25$ van DV (3.13).

Het is duidelijk dat de benaderingen niet bruikbaar zijn om de exacte oplossing door het punt $(0, 2/27)$ *voor te stellen. Verder toont Fig. 3.4 dat de exacte oplossingen door naburige*

punten een sterk verschillend verloop kennen ondanks het feit dat de beginvoorwaarden slechts weinig verschillen. Om dit te verduidelijken, kunnen we twee oplossingen van de DV bij verschillende beginvoorwaarden beschouwen. Stel bij de eerste oplossing $C = C_1$ en bij de tweede oplossing $C = C_2$, dan verschillen deze oplossingen een factor $(C_1 - C_2)\, e^{3\,t}$, welke exponentieel toeneemt met de tijd. Dit verklaart het sterk verschillend verloop van de exacte oplossingen bij beginvoorwaarden die slechts weinig verschillen. Het verklaart eveneens waarom de benaderingen slechts bedroevende resultaten opleveren.

Opdracht 3.6

Beschouw de eerste-orde DV:

$$y' = 20\, y - \cosh(t) + \sinh(t)\,, \tag{3.14}$$

over het t-interval $[0, 3]$. *De algemene oplossing van deze DV wordt gegeven door*

$$y(t) = \tfrac{1}{21}\, \cosh(t) - \tfrac{1}{21}\, \sinh(t) + C\, e^{20\,t}\,.$$

1. *Kan je o.b.v. bovenstaande gegevens voorspellen of deze DV slecht geconditioneerd is? Motiveer je antwoord.*

2. *Wat is de waarde van de constante C voor de oplossing waarvan alle naburige oplossingen zich weg bewegen?*

   $C =$

3. *Geef een stel 'slechte' beginvoorwaarden.*

   $(t_0, y_0) =$

4. *Implementeer de DV in de functie* `functie36` *en de exacte oplossing bij het stel beginvoorwaarden uit (3) in de functie* `exact36`.

5. *Plot het richtingsveld van deze DV. Zet hierbij de y-as op* $[-1.5, 1.5]$. *Plot bovenop het richtingsveld de exacte oplossing bij het stel beginvoorwaarden uit (3) samen met de Euler-, midpoint-, RK4- en Adams–Bashforth-benadering op één figuur. Bevestigt dit je antwoord op de vorige vragen?*

3.3.2 Stijve eerste-orde differentiaalvergelijkingen

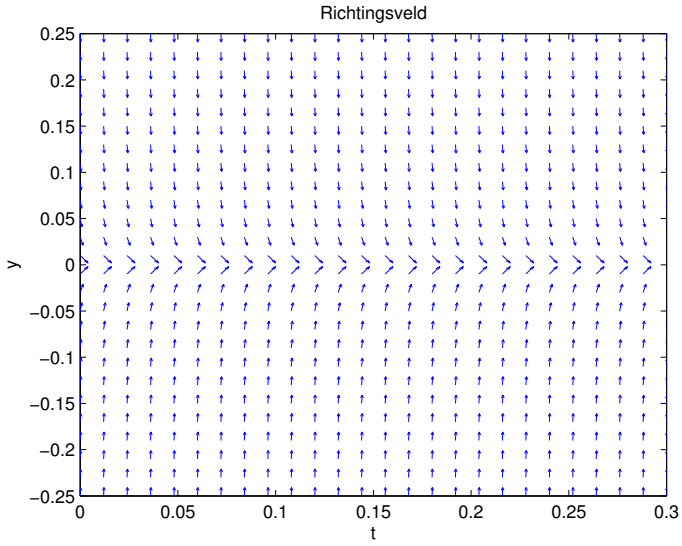
We noemen een eerste-orde DV *stijf* als de oplossing ervan vrij snel verandert ten opzichte van de lengte van het t-interval waarover de DV wordt opgelost. Om de snelle veranderingen in de oplossing te vatten met een numerieke benaderingsmethode, hebben we een heel kleine stapgrootte nodig. Dergelijke minieme stapgroottes resulteren in een drastische toename van de benodigde rekentijd waardoor MATLAB vast loopt.

Voorbeeld 3.2 Stijve differentiaalvergelijking

Beschouw de DV

$$y' = -200\,y\,, \tag{3.15}$$

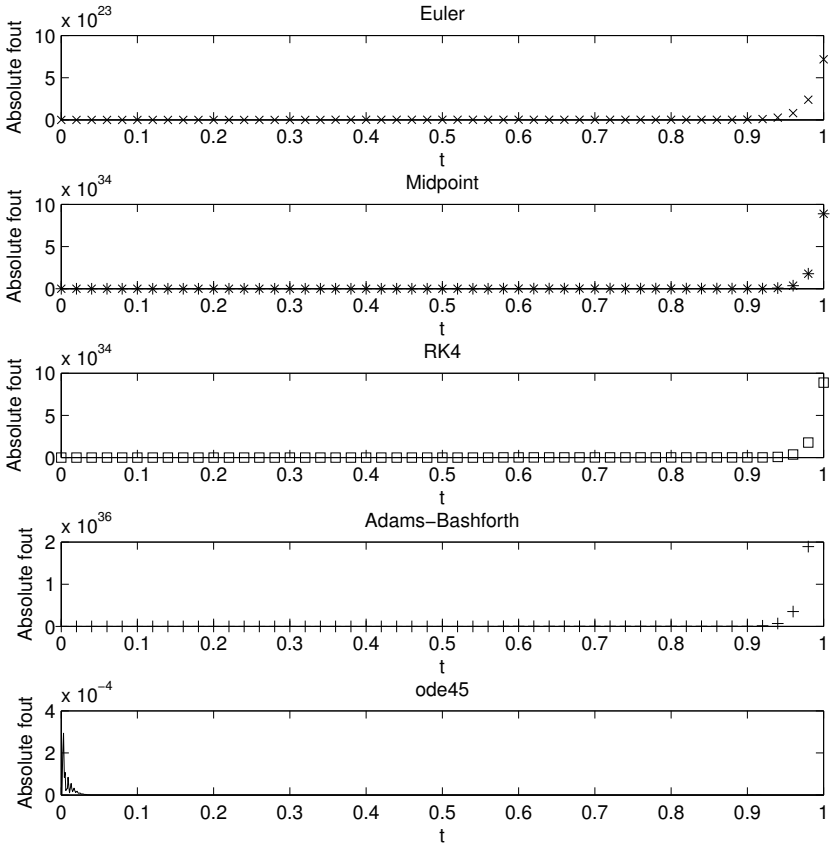
over het t-interval $[0, 1]$ en met beginvoorwaarde $y(0) = 1$. De exacte oplossing van deze DV wordt gegeven door $y(t) = e^{-200\,t}$. De nagenoeg verticale richtingsvectoren in het richtingsveld van deze DV geven aan dat de oplossing zeer snel verandert in de tijd (Fig. 3.5).



Figuur 3.5: Richtingsveld van DV (3.15).

Figuur 3.6 geeft de foutencurves weer die optreden bij numerieke benadering van DV (3.15) *voor $n_{Euler} = n_{midpoint} = n_{RK4} = n_{AB} = 50$.*

Hierop is duidelijk te zien dat de midpoint-methode, de RK4-methode en de methode van Euler er niet in slagen om de exacte oplossing te benaderen. Ook de MATLAB-functie `ode45` *werd gebruikt om de oplossing te benaderen. Figuur 3.6 illustreert dat* `ode45` *er veel beter in slaagt om de exacte oplossing te benaderen. Hiertoe nam deze methode wel zes keer meer stappen dan het aantal stappen opgegeven voor onze eigen methodes. We mogen dan ook besluiten dat de opgegeven stapgrootte duidelijk te groot is om de snelle verandering van de exacte oplossing te kunnen vatten.*



Figuur 3.6: Foutencurves bij de numerieke benadering van DV (3.15).

Indien een dergelijk probleem zich voordoet, kunnen we uiteraard de vraag stellen welke stapgrootte genomen dient te worden opdat de numerieke benadering de exacte oplossing goed zou benaderen. Voor bepaalde types DVn zoals DV (3.15) is het mogelijk om de maximale stapgrootte te bepalen opdat de oplossing van de DV goed zou benaderd worden. Echter, vermits deze bepaling meestal slechts mogelijk is met voorkennis van de exacte oplossing die voor de meeste praktische toepassingen niet gekend is, verwijzen we hiervoor naar gespecialiseerde literatuur.

Ook de standaard MATLAB-functies `ode23`, `ode45` en `ode113` zullen voor stijve DVn trachten de snelle veranderingen in rekening te brengen. Om dit te verwezenlijken, zullen deze methodes een zeer groot aantal stappen gebruiken bij de benadering van de oplossing. Dit kan, indien we werken over grotere tijdsintervallen, leiden tot een accumulatie van lokale fouten of tot zeer lange rekentijden. MATLAB voorziet daarom in een aantal functies die wel geschikt zijn om dergelijke stijve DVn op te lossen, namelijk `ode15s`, `ode23s` en `ode23tb`. Ze hebben elk hun specifiek toepassingsgebied (Tabel 3.2). Praktisch probeer je best eerst `ode45`, `ode23`

of `ode113` om een aanvaardbare benadering te genereren. Indien dit niet tot het gewenste resultaat leidt, kan je de alternatieve functies gebruiken.

Tabel 3.2: Overzicht van de MATLAB-functies voor het benaderen van stijve DVn.

Functie	Nauwkeurigheid	Gebruik
`ode15s`	laag tot gemiddeld	meeste stijve DVn
`ode23s`	laag	stijve DVn waarbij een vrij grote fout
`ode23tb`	laag	is toegestaan op de benadering

4

Eindige differentiemethodes

"*Thus the partial differential equation entered theoretical physics as a handmaid, but has gradually become mistress.*"

— Albert Einstein (1931) —

In de voorgaande practica werden numerieke technieken voor het oplossen van beginwaardeproblemen besproken. Opdat deze technieken zouden functioneren, moet $y(t_0) = y_0$ gekend zijn. Immers, zonder deze beginvoorwaarde kan de recursiebetrekking gegeven door Vgl. (2.3) of (3.9) niet uitgevoerd worden.

Echter, in talrijke praktische toepassingen is $y(t_0) = y_0$ niet gekend, maar is ofwel de oplossing gekend in een punt t_α binnen het oplossingsinterval $[t_0, t_n]$, d.i. $y(t_\alpha) = y_\alpha$ (Dirichlet-randvoorwaarde), of is de afgeleide van de oplossing in een punt van het oplossingsinterval gekend zodat $y'(t_\alpha) = y'_\alpha$ (Neumann-randvoorwaarde). De oplossing van een dergelijk randwaardeprobleem (*boundary value problem*) kan niet zomaar benaderd worden a.d.h.v. de numerieke methodes die reeds aan bod kwamen. Een mogelijkheid bestaat erin het randwaardeprobleem $y(t_\alpha) = y_\alpha$ te vertalen naar een corresponderend beginwaardeprobleem $y(t_{0,\alpha}) = y_{0,\alpha}$ door $t_{0,\alpha}$ en $y_{0,\alpha}$ iteratief aan te passen totdat het beginwaardeprobleem voldoet aan de randvoorwaarde $y(t_\alpha) = y_\alpha$. Echter, gelet op de beperkte efficiëntie van deze methode, die bekend staat als de shooting methode (*shooting method*), wordt deze niet behandeld in deze nota's, maar belichten we daarentegen methodes die hun deugdelijkheid hebben bewezen bij het oplossen van ingenieursproblemen beschreven a.d.h.v. (partiële) DVn zoals de eindige differentiemethodes (*finite difference methods*) en eindige elementenmethodes (*finite element methods*), die dagdagelijks worden gebruikt.

In dit practicum zullen de basisconcepten van de eindige differentiemethode uitvoerig worden behandeld, maar desalniettemin zullen we ons beperken tot de meest elementaire oplossingsmethodes. De bruikbaarheid van eindige differentiemethodes voor het benaderen van

randwaardeproblemen beschreven a.d.h.v. (partiële) DVn zal in dit practicum geïllustreerd worden.

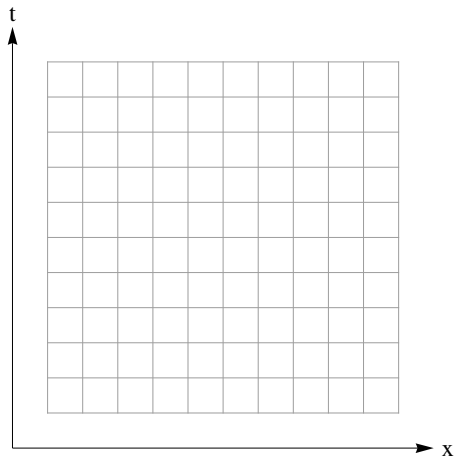
4.1 Eindige differentie- en elementenmethodes

4.1.1 Overzicht

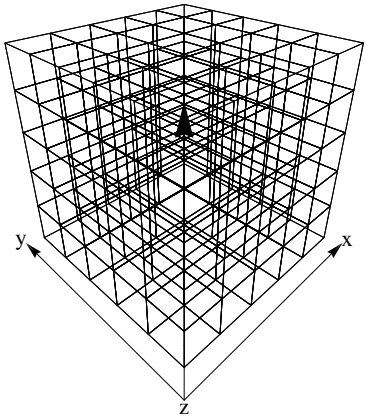
Net zoals er vaak geen analytische oplossingen van gewone DVn kunnen gevonden worden voor vele praktische problemen, is het vinden van een analytische oplossing van partiële DVn behoudens een paar uitzonderingen een nagenoeg onmogelijke opdracht, waardoor men in de meeste gevallen een beroep dient te doen op numerieke benaderingsmethodes zoals de eindige differentie- en elementenmethodes. In tegenstelling tot de methodes die we bestudeerden in de voorgaande practica, omvatten deze methodes niet slechts een recursiebetrekking waarmee de benaderingen kunnen worden uitgevoerd, maar behelzen ze een opeenvolging van stappen die voor elke probleemstelling opnieuw moet doorlopen worden om aldus tot een aanvaardbare benadering van de (partiële) DV te komen. De eerste cruciale stap omvat het opdelen van het domein waarover men de (partiële) DV met m onafhankelijke veranderlijken wenst te benaderen in m-dimensionale elementen. Zo dient het oplossingdomein van een gewone DV opgedeeld te worden in lijnstukken (dimensie 1), terwijl dat van de partiële DVn $\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$ en $\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$ moet opgedeeld worden in, respectievelijk, polygonen en polyhedronen. Veelal wordt naar een dergelijke opdeling gerefereerd als de discretisatie van het oplossingsdomein. Bij eindige differentiemethodes is deze discretisatie beperkt tot regelmatige elementen (Fign. 4.1(a) en 4.1(b)), terwijl bij de eindige elementenmethode kan gebruik gemaakt worden van onregelmatige elementen die toelaten om oplossingsdomeinen met een grillige vorm vrij gemakkelijk te discretiseren zoals weergegeven in Fign. 4.1(c), 4.1(d) en 4.1(e). Het is dan ook duidelijk dat het gebruik van eindige differentiemethodes grotendeels beperkt is tot problemen waarbij het oplossingsdomein een relatief eenvoudige vorm heeft.

In een tweede stap zullen beide methodes in elk van de discrete elementen, of op hoekpunten van deze elementen, de oplossing van de (partiële) DV benaderen door deze te vervangen door een stelsel van differentievergelijkingen, dit zijn vergelijkingen die verbanden leggen tussen de waarden van de gezochte functie u op discrete tijdstippen en posities (eindige differentiemethodes), of door deze te vervangen door een stelsel van veeltermen waaruit de benaderde waarden van u op de hoekpunten kunnen berekend worden (eindige elementenmethodes).

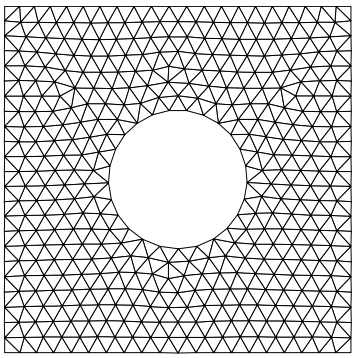
Hoewel voor de meeste (partiële) DVn mag worden aangenomen dat eindige elementenmethodes een meer nauwkeuriger benadering zullen opleveren dan eindige differentiemethodes en het gebruik van eerstgenoemde niet beperkt wordt door de complexiteit van het oplossingsdomein, zullen we ons in deze nota's beperken tot een uitvoerige bespreking van eindige



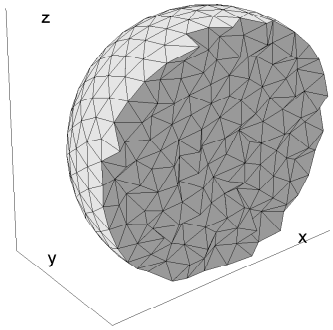
(a)



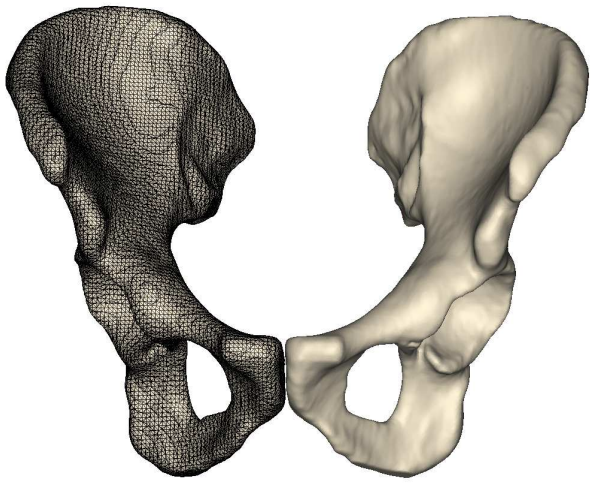
(b)



(c)



(d)



(e)

Figuur 4.1: Mogelijke discretisaties van 2-dimensionale ((a) en (c)) en 3-dimensionale ((b), (d) en (e)) oplossingsdomeinen.

differentiemethodes, vermits deze het makkelijkst te implementeren zijn en slechts weinig extra theoretische kennis vereisen. Bovendien sluiten deze het nauwst aan bij de methodes voor de benadering van gewone DVn, die behandeld werden in de voorgaande practica. Voor meer informatie omtrent de eindige elementenmethode verwijzen we dan ook naar gespecialiseerde literatuur (e.g. Liu and Quek, 2003; Rao, 2010).

4.1.2 Eindige differentiemethodes

Zoals reeds vermeld, kunnen twee cruciale stappen onderscheiden worden in elke eindige differentiemethode, zijnde (1) de discretisatie van het oplossingsdomein en (2) de benadering van de (partiële) DV in elk van de hoekpunten van deze discrete elementen. Echter, vanuit praktisch oogpunt kunnen er in elk van deze stappen enkele cruciale onderdelen worden ontwaard die hieronder worden opgelijst.

1. Discretisatie van het oplossingsdomein
   (a) Definitie van het oplossingsdomein
   (b) Keuze van een stapgrootte voor elk van de onafhankelijke variabelen
   (c) Discretisatie van het oplossingsdomein
2. Benadering van de (partiële) DV
   (a) Discretisatie van de (partiële) DV m.b.v. eindige differentiebenaderingen voor de erin voorkomende afgeleiden
   (b) Discretisatie van de rand- en beginvoorwaarden
   (c) Keuze van een geschikte oplossingsmethode o.b.v. de (partiële) DV, de rand- en beginvoorwaarden en de gewenste accuraatheid en/of efficiëntie
   (d) Uitvoering van de numerieke benadering en voorstelling van de resultaten

Vermits de onderdelen van de eerste stap hoofdzakelijk gestuurd worden door de aard van het probleem en de vereisten van de gebruiker, zullen we doorheen het vervolg van dit practicum vooral aandacht besteden aan de onderdelen die noodzakelijk zijn voor de eigenlijke benadering van (partiële) DVn. Zoals eerder aangestipt, zijn eindige differentiemethodes in wezen probleemspecifiek, wat met zich meebrengt dat voor de benadering van elk randwaardeprobleem een specifieke –unieke– methode dient ontwikkeld te worden waarvan de bruikbaarheid meestal gelimiteerd is tot het originele probleem. Derhalve is een goed begrip van dergelijke methodes slechts mogelijk mits voldoende voorbeelden bestudeerd te hebben en zelf passende problemen opgelost te hebben. Daarom zullen deze methodes in het vervolg van dit practicum geïllustreerd worden a.d.h.v. een randwaardeprobleem beschreven door een gewone DV

en een randwaardeprobleem dat partiële DVn gebruikt waarna het de lezer ten sterkste wordt aangeraden om de opdrachten hieromtrent uit te voeren daar deze cruciaal zijn voor een goed begrip van deze methodes. Echter, vooraleer we dergelijke problemen kunnen oplossen, dienen we nog stil te staan bij de eindige differentiebenaderingen van afgeleiden daar deze de discretisatie van (partiële) DVn mogelijk maken.

4.1.3 Eindige differentiebenaderingen

Eindige differentiebenaderingen van afgeleiden kunnen bepaald worden a.d.h.v. de Taylorreeksontwikkeling in de omgeving van het punt t:

$$y(t+\Delta t) = y(t) + \frac{\Delta t\, y'(t)}{1!} + \frac{\Delta t^2\, y''(t)}{2!} + \ldots + \frac{\Delta t^n\, y^{(n)}(t)}{n!} + \ldots \tag{4.1}$$

Wensen we een eindige differentiebenadering te vinden voor $y'(t)$ dan kunnen we, na herschikking en verwaarlozing van de hogere orde afgeleiden, schrijven:

$$y'(t) \approx \frac{y\,(t+\Delta t) - y\,(t)}{\Delta t}\,, \tag{4.2}$$

hetgeen niets anders voorstelt dan de voorwaartse differentiebenadering van $y'(t)$ vermits de bepaling van $y'(t)$ gebeurt o.b.v. de waarde die de functie bereikt in t en deze die Δt verder wordt bereikt. Het is eenvoudig na te gaan dat dit niets anders is dan de klassieke Eulerbenadering die we in Practicum 2 uitvoerig hebben besproken. De fout op de voorwaartse differentiebenadering is evenredig met de stapgrootte Δt vermits de grootste term in de verwaarloosde som in het rechterlid van Vgl. (4.1)

$$\frac{\Delta t\, y''(t)}{2!} + \ldots + \frac{\Delta t^{n-1}\, y^{(n)}(t)}{n!} + \ldots$$

na herschikking wordt bepaald door $\dfrac{\Delta t\, y''(t)}{2!}$.

Analoog kan een achterwaartse differentiebenadering voor $y'(t)$ gevonden worden door de Taylorreeksontwikkeling

$$y(t-\Delta t) = y(t) - \frac{\Delta t\, y'(t)}{1!} + \frac{\Delta t^2\, y''(t)}{2!} - \ldots + (-\Delta t)^n\, \frac{y^{(n)}(t)}{n!} + \ldots \tag{4.3}$$

te beschouwen die, na herschikking en verwaarlozing van de hogere orde afgeleiden, kan geschreven worden als:

$$y'(t) \approx \frac{y(t) - y(t-\Delta t)}{\Delta t}\,.$$

Naast een voorwaartse en achterwaartse differentiebenadering kan $y'(t)$ ook gevonden worden uit een centrale differentiebenadering die volgt uit het verschil van Vgln. (4.1) en (4.3):

$$y(t+\Delta t) - y(t-\Delta t) = 2\,\frac{\Delta t\, y'(t)}{1!} + 2\,\frac{\Delta t^3\, y^{(3)}(t)}{3!} + \ldots + 2\frac{\Delta t^{2n+1}\, y^{(2n+1)}(t)}{(2n+1)!} + \ldots\,, \tag{4.4}$$

waaruit na verwaarlozing van de hogere orde afgeleiden en herschikking de centrale differentiebenadering van $y'(t)$ volgt:

$$y'(t) \approx \frac{y(t+\Delta t) - y(t-\Delta t)}{2\,\Delta t}\,, \tag{4.5}$$

waarbij de fout evenredig is met Δt^2. Bijgevolg is deze benadering voor $\Delta t < 1$ nauwkeuriger dan de voor- en achterwaartse differentiebenaderingen.

Eindige differentiebenaderingen voor hogere orde afgeleiden kunnen eveneens gevonden worden o.b.v. Tayloreeksontwikkelingen, maar voor de meeste praktische toepassingen is het voldoende om te beschikken over de centrale differentiebenadering van $y''(t)$:

$$y''(t) \approx \frac{y(t+\Delta t) - 2\,y(t) + y(t-\Delta t)}{\Delta t^2}\,,$$

deze van $y'''(t)$:

$$y'''(t) \approx \frac{y(t+2\,\Delta t) - 2\,y(t+\Delta t) + 2\,y(t-\Delta t) - y(t-2\,\Delta t)}{2\,\Delta t^3}$$

en, finaal, de centrale differentiebenadering van $y^{(4)}(t)$, gegeven door

$$y^{(4)}(t) \approx \frac{y(t+2\,\Delta t) - 4\,y(t+\Delta t) + 6\,y(t) - 4\,y(t-\Delta t) + y(t-2\,\Delta t)}{\Delta t^4}\,.$$

Uiteraard kunnen ook voor- en achterwaartse eindige differentiebenaderingen geformuleerd worden voor $y''(t)$, $y'''(t)$ en $y^{(4)}(t)$, maar de centrale differentiebenadering geniet de voorkeur voor hogere-orde afgeleiden, vermits deze aanleiding geeft tot een grotere numerieke stabiliteit. Tenzij anders vermeld, worden eerste-orde afgeleiden bij voorkeur vervangen door hun voorwaartse eindige differentiebenaderingen. Neumann randvoorwaarden daarentegen worden, hetzij anders vermeld, bij voorkeur vervangen door hun centrale eindige differentiebenaderingen.

Gebruikmakend van deze eindige differentiebenaderingen kan elke n-de orde (partiële) DV, na vervanging van de afgeleiden door hun eindige differentiebenaderingen, herschreven worden als een differentievergelijking die vervolgens kan opgelost worden.

Voorbeeld 4.1 Discretisatie van een partiële differentiaalvergelijking

De propagatie van vlammen doorheen een medium kan beschreven worden a.d.h.v. de Kuramoto-Sivashinsky vergelijking die wordt gegeven door

$$\frac{\partial U}{\partial t} = -\nabla^2 U - \nabla^4 U + \nabla U^2\,,$$

die zich in één dimensie herleidt tot

$$\frac{\partial U}{\partial t} = -\frac{\partial^2 U}{\partial x^2} - \frac{\partial^4 U}{\partial x^4} + 2\,U\,\frac{\partial U}{\partial x}\,. \tag{4.6}$$

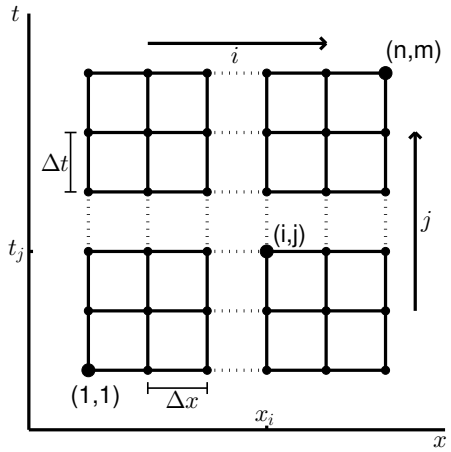
Het discretiseren van deze partiële DV impliceert dat we U niet langer wensen te kennen in alle punten van het oplossingsdomein $[t_0, t_m] \times [x_0, x_n]$*, maar slechts in de hoekpunten* (i, j) *van de elementen waarmee dit oplossingsdomein wordt gediscretiseerd (Fig. 4.2). Gebruikmakend van centrale differentiebenaderingen voor de hogere-orde afgeleiden en voorwaartse benaderingen voor* $\frac{\partial U}{\partial t}$ *en* $\frac{\partial U}{\partial x}$ *kunnen we Vgl.* (4.6) *omvormen tot*

$$\frac{U(x, t+\Delta t) - U(x, t)}{\Delta t} = -\frac{U(x+\Delta x, t) - 2\,U(x, t) + U(x-\Delta x, t)}{\Delta x^2} - \frac{U(x+2\,\Delta x, t) - 4\,U(x+\Delta x, t) + 6\,U(x, t) - 4\,U(x-\Delta x, t) + U(x-2\,\Delta x, t)}{\Delta x^4} + 2\,U(x, t)\,\frac{U(x+\Delta x, t) - U(x, t)}{\Delta x},$$

of rekening houdend met de conventie voor indexnummering zoals weergegeven in Fig. 4.2

$$\frac{U_{i,j+1} - U_{i,j}}{\Delta t} = -\frac{U_{i+1,j} - 2\,U_{i,j} + U_{i-1,j}}{\Delta x^2} - \frac{U_{i+2,j} - 4\,U_{i+1,j} + 6\,U_{i,j} - 4\,U_{i-1,j} + U_{i-2,j}}{\Delta x^4} + 2\,U_{i,j}\,\frac{U_{i+1,j} - U_{i,j}}{\Delta x},$$

waarin $U_{i,j}$ *de energie-inhoud in het punt* (x_i, t_j) *voorstelt.*



Figuur 4.2: Eindig differentiegrid voor de discretisatie van partiële DV (4.6) met weergave van de conventie voor indexnummering.

Na discretisatie van de gegeven partiële DV kan de bekomen differentievergelijking gebruikt worden om een benadering van $U(x, t)$ te vinden in elk van de knooppunten van het eindig differentiegrid d.m.v. een expliciete of impliciete oplossingsmethode, hetgeen in het vervolg van dit practicum zal geïllustreerd worden a.d.h.v. twee voorbeelden.

4.2 Gewone differentiaalvergelijkingen

4.2.1 Probleemstelling

De concentratie van een opgeloste stof in de bodem kan wiskundig beschreven worden door de DV

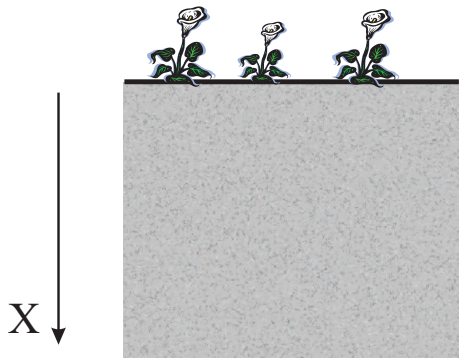
$$\frac{\mathrm{d}^2 C}{\mathrm{d}x^2} = C\,(x)\;, \tag{4.7}$$

die onderhevig is aan de randvoorwaarden

$$C(0) = 1\,, \qquad \frac{\mathrm{d}C(1)}{\mathrm{d}x} = 0\,,$$

waarin $C(x)$ de concentratie van de stof op een diepte x voorstelt, waarbij de positieve x-as naar beneden gericht is (Fig. 4.3). De exacte oplossing van dit randwaardeprobleem over het x-interval $[0, 1]$ wordt gegeven door

$$C\,(x) = \frac{e^{2-x} + e^{x}}{1 + e^{2}}\,.$$



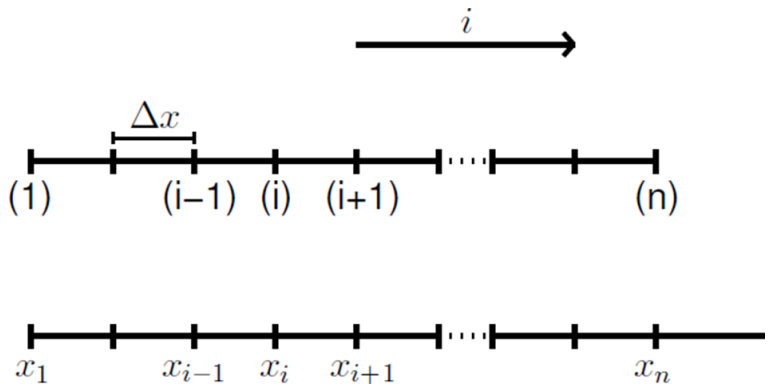
Figuur 4.3: Schematisch overzicht van het randwaardeprobleem beschreven door DV 4.7.

4.2.2 Werkwijze

In overeenstemming met de procedure beschreven in Sectie 4.1 voor het succesvol opstellen van een eindige differentiemethode, zullen we het oplossingsinterval $[0, 1]$ opdelen in $n - 1$ intervallen met lengte Δx, hetgeen resulteert in n knooppunten i die op een afstand Δx van elkaar gelegen zijn (Fig. 4.4).

Discretisatie van DV (4.7) m.b.v. de centrale differentiebenadering levert:

$$\frac{C\,(x + \Delta x) - 2\,C\,(x) + C\,(x - \Delta x)}{\Delta x^2} = C\,(x)\;,$$



Figuur 4.4: Eindig differentiegrid gebruikt ter oplossing van DV (4.7) met weergave van de conventie voor indexnummering.

of na herschikking en rekening houdend met de conventie voor indexnummering:

$$C_i = \frac{C_{i+1} + C_{i-1}}{\Delta x^2 + 2}, \qquad i = 1, 2, \ldots, n. \tag{4.8}$$

Opdat we de concentratie in elke knoop zouden kunnen benaderen, dienen we deze recursiebetrekking te schrijven voor elke knoop in het oplossingsdomein:

$$\begin{cases} C_1 = \dfrac{C_2 + C_0}{\Delta x^2 + 2} \\ C_2 = \dfrac{C_3 + C_1}{\Delta x^2 + 2} \\ \quad \vdots \\ C_n = \dfrac{C_{n+1} + C_{n-1}}{\Delta x^2 + 2} \end{cases} \tag{4.9}$$

Het is duidelijk dat er geen unieke oplossing voor dit stelsel bestaat vermits het n vergelijkingen en $n+2$ onbekenden omvat doordat de vergelijkingen voor de eerste en n-de knoop kennis vereisen van de concentratie in de knopen die buiten het oplossingsdomein gelegen zijn, namelijk C_0 en C_{n+1}. Echter, de discretisatie van de Dirichlet randvoorwaarde ($C(0) = 1$) leert dat $C_1 = 1$ en de centrale differentiebenadering van de Neumann randvoorwaarde geeft aanleiding tot

$$\begin{aligned} \frac{\mathrm{d}C(1)}{\mathrm{d}x} = 0 &\approx \frac{C(1+\Delta x) - C(1-\Delta x)}{2\,\Delta x}, \\ &= \frac{C_{n+1} - C_{n-1}}{2\,\Delta x}, \end{aligned}$$

waaruit $C_{n+1} = C_{n-1}$. Aldus kunnen de extra onbekenden in Stelsel (4.9) geëlimineerd worden door vervanging van de eerste vergelijking door $C_1 = 1$ en substitutie van de uitdrukking $C_{n+1} = C_{n-1}$ in de vergelijking voor C_n.

Zodoende beschikken we nu over een lineair stelsel van n vergelijkingen in n onbekenden:

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ -1 & \Delta x^2+2 & -1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -1 & \Delta x^2+2 & -1 \\ 0 & 0 & 0 & \dots & 0 & -2 & \Delta x^2+2 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_{n-1} \\ C_n \end{bmatrix}}_{C} = \underbrace{\begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{B}$$

Het is duidelijk dat dit stelsel rechtstreeks zou kunnen opgelost worden naar de vector met onbekenden **C**, maar bij een groot aantal knopen is dit rekentechnisch moeilijk ten gevolge van de noodzakelijke matrixinverse A^{-1}. Daarom wordt bij een groot aantal knopen veelal gebruik gemaakt van een iteratieve oplossingsmethode waarbij, initieel, voor elke C_i, **behalve in de knopen met een Dirichlet-randvoorwaarde**, een willekeurige waarde wordt gekozen, waarna deze herhaaldelijk (iteratief) worden aangepast door Vgl. (4.8) te evalueren voor alle knopen in het oplossingsdomein tot $\max_i \left(\left|C_i^m - C_i^{m+1}\right|\right) < \phi$, waarbij m de iteratie-index voorstelt en ϕ het stopcriterium, d.i. het maximaal toegelaten absoluut verschil tussen de berekende concentratie in twee opeenvolgende iteraties. In MATLAB kan dit alles geschieden door een vector `C` met n posities te initialiseren m.b.v. de functies `zeros`, waarna `C(1) = 1` wordt meegegeven vanwege de Dirichlet randvoorwaarde bij $x = 0$. Vervolgens kan Vgl. (4.8) geëvalueerd worden voor de overige $n-1$ knopen in het oplossingsdomein waarvan de resultaten moeten bewaard worden in een vector `Cnew`. Echter, bij de berekening van C_n zullen we stuiten op het probleem dat C_{n+1} niet gekend is. Dit kan evenwel verholpen worden door een $n+1$-ste knoop toe te voegen aan de betreffende rand van het domein en de concentratie in deze knoop gelijk te stellen aan deze in de $n-1$-ste knoop. Een dergelijke knoop wordt slechts toegevoegd om de Neumann-randvoorwaarde te kunnen implementeren en niet om de concentratie in een knoop buiten het domein te benaderen. Daarom worden zulke knopen *imaginaire knopen* genoemd. Nadat Vgl. (4.8) geëvalueerd werd in elk van de $n-1$ knopen, dient $\max_i \left(\left|C_i^m - C_i^{m+1}\right|\right)$ berekend te worden en indien de hiervoor bekomen numerieke waarde groter is dan het vooropgesteld stopcriterium ϕ, zal de recursiebetrekking nogmaals geëvalueerd worden nadat de waarden van de m-de iteratie die worden bewaard in de vector `C` werden overschreven door deze van de $m+1$-ste iteratie in `Cnew`. Dit iteratief proces zal slechts stoppen wanneer aan het stopcriterium wordt voldaan.

4.2.3 Iteratieschema's

Vermits de iteratieprocedure stopt van zodra aan het stopcriterium voldaan is, zal de fout op de benaderingen slechts weinig verschillen tussen verschillende iteratieschema's, doch zullen deze sterk verschillen in termen van efficiëntie. Een hogere efficiëntie betekent dat er een kleiner aantal iteraties nodig is vooraleer aan het stopcriterium voldaan is.

Jacobi iteratie

Met dit iteratieschema wordt C_i^{m+1} berekend o.b.v. de waarden in de meest naburige knopen gevonden tijdens de vorige (m-de) iteratie:

$$C_i^{m+1} = f\left(C_i^m\right) .$$

Toegepast op Vgl. (4.8) wordt dit:

$$C_i^{m+1} = \frac{C_{i+1}^m + C_{i-1}^m}{\Delta x^2 + 2} .$$

Vermits de berekening van C_i^{m+1} enkel gebruik maakt van waarden die berekend werden tijdens de vorige iteratie is dit numeriek schema weinig efficiënt zodat het dan ook zelden wordt gebruikt voor het oplossen van complexe problemen. Bovendien speelt de volgorde van de knopen waarin de berekening gebeurt geen rol.

Gauss-Seidel iteratie

Intuïtief voelen we aan dat de efficiëntie van het Jacobi iteratieschema verhoogd zou kunnen worden door de waarden in de $i-1$-ste knoop die reeds werden berekend tijdens de $m+1$-ste iteratie te gebruiken bij de berekening van C_i^{m+1}. In tegenstelling tot de Jacobi iteratie, dienen de berekeningen nu wel in een zekere volgorde te gebeuren. Er moet steeds gestart worden in de tweede knoop om te eindigen in de n-de knoop. Algemeen geldt:

$$C_i^{m+1} = f\left(C_j^m, C_k^{m+1}\right) \qquad j > i\,, k < i\,.$$

Toegepast op Vgl. (4.8) wordt dit:

$$C_i^{m+1} = \frac{C_{i+1}^m + C_{i-1}^{m+1}}{\Delta x^2 + 2} . \tag{4.10}$$

Zoals eerder vermeld, zal het gebruik van dit iteratieschema enkel de efficiëntie van de oplossingsmethode verhogen, maar niet leiden tot betere benaderingen.

Successive over-relaxation (SOR)

Noemen we het verschil tussen twee opeenvolgende Gauss-Seidel iteraties κ

$$\kappa = C_i^{m+1} - C_i^m\,, \tag{4.11}$$

dan kan C_i^{m+1} gevonden worden uit dit verschil en C_i^m, d.i.

$$C_i^{m+1} = C_i^m + \omega\,\kappa\,, \tag{4.12}$$

waarbij ω de relaxatie-factor voorstelt, en verder geldt dat $\omega \geq 1$. Substitutie van Vgl. (4.11) in Vgl. (4.12) en gebruikmakend van Vgl. (4.10) levert voor de beschouwde DV:

$$C_i^{m+1} = (1-\omega)\, C_i^m + \omega \left[\frac{C_{i+1}^m + C_{i-1}^{m+1}}{\Delta x^2 + 2} \right] .$$

Indien $\omega = 1$ vereenvoudigt dit schema zich tot de Gauss-Seidel iteratie (Vgl. (4.10)). Voor de meeste praktische toepassingen is $1 \leq \omega < 2$.

Tabel 4.1 illustreert de efficiëntie van de besproken iteratieschema's in termen van het aantal iteraties dat nodig is alvorens aan het stopcriterium voldaan wordt voor het beschouwde randwaardeprobleem. Figuur 4.5 illustreert de SOR-benadering en de exacte oplossing van DV (4.7) bij twee stopcriteria. De foutencurves, die eveneens in Fig. 4.5 zijn opgenomen, tonen duidelijk dat de absolute fout met een factor tien afneemt indien ϕ verlaagd wordt van 10^{-5} tot 10^{-6}.

Tabel 4.1: Aantal iteraties dat nodig is opdat $\max_i \left(\left| C_i^m - C_i^{m+1} \right| \right) < \phi$ voor verschillende stopcriteria.

Schema	aantal iteraties	
	$\phi = 10^{-5}$	$\phi = 10^{-6}$
Jacobi	19885	33165
Gauss-Seidel	9964	16671
SOR ($\omega = 1.1$)	8636	14135

Het MATLAB-script `finDifDV.m` bevat alle instructies die nodig zijn om het randwaardeprobleem te benaderen m.b.v. het Jacobi iteratieschema. Je kan de uitvoering van een script of functie onderbreken door in het instructievenster de instructie `Ctrl+C` op te geven.

Codefragment 4.1: finDifDV

```
clear all           % vrijmaken werkruimte
clc                 % vrijmaken instructievenster

dx = 0.01;          % instellen van de stapgrootte
x = 0:dx:1;         % aanmaak van de vector die de x-coördinaten
                    % van de knopen bevat
n = length(x);      % aantal knopen
C = zeros(1, n+1); % matrix die de concentratie in elke knoop bevat
                    % initieel wordt deze in elke knoop gelijk-
                    % gesteld aan 0
C(1) = 1;           % Dirichlet-randvoorwaarde C(x=0) = 1
Cnew = C;           % matrix waarin de concentraties na elke
```

```
                    % iteratiestap worden bewaard
phi = 10^-5;        % stopcriterium
verschil = 10;      % opdat de iteraties zouden starten dient
                    % initeel het verschil groter te zijn dan phi
k = 0;              % k bevat het aantal iteratiestappen,
                    % wordt initieel gelijkgesteld aan 0

while verschil > phi % while-lus die het iteratieschema omvat
    k = k + 1;
    for i = 2:n      % for-lus die toelaat het grid te door lopen
        Cnew(i) = (C(i+1) + C(i-1))/(dx^2 + 2); % (Jacobi)
    end
    Cnew(n+1) = Cnew(n-1);          % Neumann randvoorwaarde
    verschil = max(abs(Cnew - C)); % berekening |Ci^m - Ci^(m+1)|
    C = Cnew;                       % update van de concentratie
                                    % in elke knoop
end
plot(x, Cnew(1:n))  % concentratie in imaginaire knoop NIET geplot!
axis([0 1 0 1])
hold on
xlabel('x [m]')
ylabel('Concentratie (10^{-3} M)')
fplot(@exact, [0, 1], 'r--')
legend('Eindige differentiebenadering','Exacte oplossing')
title(['\phi = ' num2str(phi)])
```

Opdracht 4.1

Implementeer in het MATLAB script `finDifDV.m` *de Gauss-Seidel en SOR ($\omega = 1.1$) iteratieschema's voor het benaderen van het randwaardeprobleem besproken in deze sectie. Hiervoor dien je enkel de uitdrukking binnen de* `for`*-lus aan te passen.*

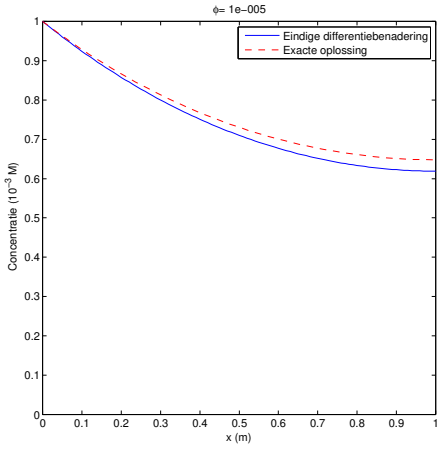
Gauss-Seidel

```
Cnew(i) =
```

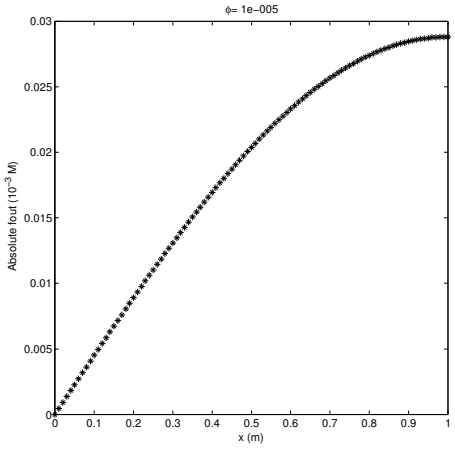
SOR

```
Cnew(i) =
```

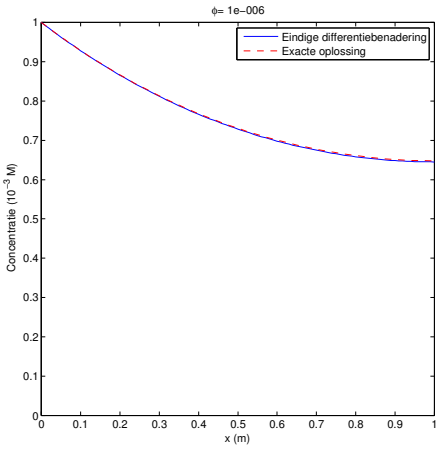
Je kan je implementatie controleren door het benodigde aantal iteraties te vergelijken met de waarden gegeven in Tabel 4.1.



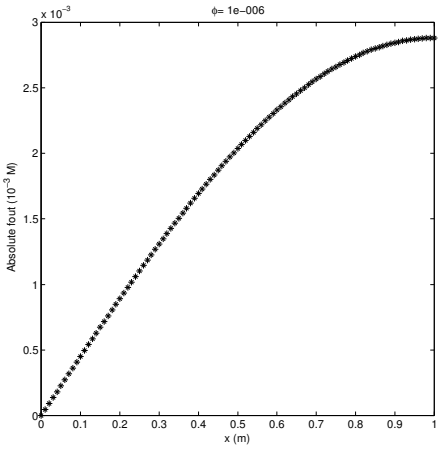
(a) Benadering bij $\phi = 10^{-5}$



(b) Fout bij $\phi = 10^{-5}$



(c) Benadering bij $\phi = 10^{-6}$



(d) Fout bij $\phi = 10^{-6}$

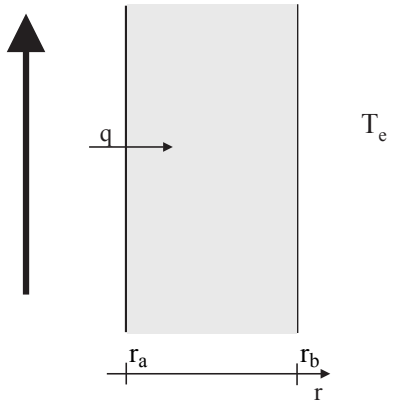
Figuur 4.5: Exacte oplossing en SOR-benadering van DV (4.7) ((a) en (c)) met corresponderende foutencurves ((b) en (d)) bij $\phi = 10^{-5}$ ((a) en (b)) en $\phi = 10^{-6}$ ((c) en (d)).

Opdracht 4.2

Een bedrijf gevestigd in de Gentse haven bezit twee productiecentra die gescheiden worden door een weg. Het eindproduct van het ene productiecentrum, een hete vloeistof, dient doorheen een metalen buis met cirkelvormige doorsnede getransporteerd te worden naar het productiecentrum gelegen aan de overzijde van de weg. Daar wordt de hete vloeistof gebruikt om de productiecyclus te voltooien. Beschouw de schematische voorstelling van deze buiswand in Fig. 4.6 waarin r de positie t.o.v. het centrum van de buis voorstelt. Deze positie wordt uitgedrukt in cilindercoördinaten. De metalen buis heeft een dikke wand. Aan de buitenzijde van de wand ($r = r_b$) heerst de omgevingstemperatuur T_e, terwijl aan de binnenzijde waar $r = r_a$ geldt dat $\frac{\mathrm{d}T}{\mathrm{d}r} = -\frac{q}{k}$, waarbij q de warmteflux doorheen de buis voorstelt. De DV die het verloop van de temperatuur T beschrijft in de buiswand wordt gegeven door

$$\frac{1}{r}\frac{\mathrm{d}T}{\mathrm{d}r} + k\,\frac{\mathrm{d}^2T}{\mathrm{d}r^2} = 0\,, \tag{4.13}$$

waarbij k de thermische geleidbaarheid is van de metaalwand.



Figuur 4.6: Schematisch overzicht van de probleemstelling in Opdracht 4.2.

We wensen het verloop van de temperatuur in de buiswand te benaderen.

1. *Ga met Mathematica na of er een analytische oplossing bestaat voor het randwaardeprobleem over het interval $[r_a, r_b] = [1, 2]$ indien je weet dat $T_e = 0$, $k = 1$, $q = 10$.*

   $T(r) =$

2. *Schets het eindig differentiegrid dat je zult gebruiken om het oplossingsinterval te discretiseren. Duid in je schets aan in welke knopen er een Dirichlet of Neumann randvoorwaarde geldt. Geef tevens aan waar eventuele imaginaire knopen dienen toegevoegd te worden.*

3. *Discretiseer alle afgeleiden in DV* (4.13) *door gebruik te maken van de* **centrale** *differentiebenadering en los de bekomen differentievergelijking op naar* T_i.

   $T_i =$

4. *Implementeer het Jacobi iteratieschema voor de gediscretiseerde DV in een MATLAB script. Pas hiervoor het op Minerva beschikbare script* `finDifDV` *op gepaste manier aan. Stel* $\Delta r = 0.01$ *en* $\phi = 10^{-7}$. *Noteer hieronder de geïmplementeerde recursiebetrekking.*

   ```
   Tnew(i) =
   ```

5. *Implementeer de Neumann randvoorwaarde.*

   ```
   Tnew( ) =
   ```

6. *Benader de oplossing van het randwaardeprobleem en plot de temperatuur* T *in de buiswand i.f.v.* r.

7. *Implementeer de Gauss-Seidel en SOR (*$\omega = 1.1$*) iteratieschema's en voer met elk van deze schema's de benadering opnieuw uit.*

   Gauss-Seidel

   ```
   Tnew(i) =
   ```

   SOR

   ```
   Tnew(i) =
   ```

   Hoeveel iteraties heeft elk van de drie geïmplementeerde schema's nodig vooraleer het stopcriterium wordt bereikt?

Tabel 4.2: Aantal iteraties nodig opdat $\max\left(\left|T_i^m - T_i^{m+1}\right|\right) < \phi$.

Schema	aantal iteraties
Jacobi	
Gauss-Seidel	
SOR	

8. *Benader het randwaardeprobleem opnieuw m.b.v. het SOR iteratieschema bij verschillende waarden voor de relaxatie-factor ω. Laat ω variëren tussen 1 en 1.8 in stappen van 0.2. Welke invloed heeft het verhogen van ω op het aantal iteraties dat nodig is om aan het stopcriterium te voldoen?*

   Wat gebeurt er indien $\omega = 2$ wordt gekozen?

4.3 Partiële differentiaalvergelijkingen

Niettegenstaande de eindige differentiemethode wordt aangewend voor het numeriek oplossen van randwaardeproblemen beschreven a.d.h.v. gewone DVn, wordt deze hoofdzakelijk gebruikt voor het benaderen van partiële DVn (en meer in het bijzonder de warmte-, golf- en Laplacevergelijking). Hiertoe dienen eveneens de stappen genomen te worden die werden opgelijst in Sectie 4.1.

We zullen de te volgen werkwijze illustreren a.d.h.v. de één-dimensionale warmtevergelijking die de variatie van de temperatuur over een bepaald gebied i.f.v. de tijd beschrijft en gegeven wordt door

$$\frac{\partial u}{\partial t} = \kappa \frac{\partial^2 u}{\partial x^2}\,, \tag{4.14}$$

onderhevig aan de beginvoorwaarde

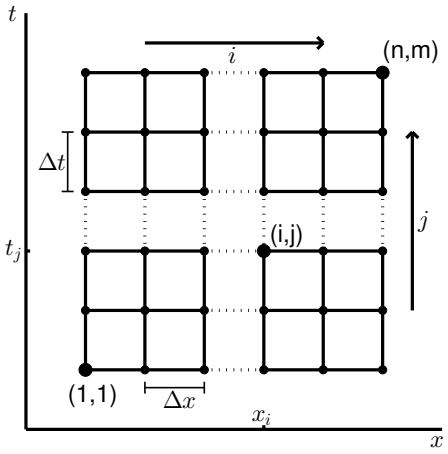
$$u(x, 0) = g(x)$$

en de randvoorwaarden

$$u(0, t) = u_l\,, \qquad u(1, t) = u_r\,, \qquad \text{als } t > 0\,,$$

waarbij κ de thermische geleidbaarheid voorstelt.

We wensen de oplossing van dit randwaardeprobleem te benaderen over het gebied $[0, 1] \times [0, 15]$. Hiertoe discretiseren we eerst het oplossingsdomein zoals weergegeven in Fig. 4.7.



Figuur 4.7: Eindig differentiegrid gebruikt ter oplossing van DV (4.14) met weergave van de conventie voor indexnummering.

Vervolgens kan Vgl. (4.14) gediscretiseerd worden door de afgeleiden te vervangen door hun eindige differentiebenaderingen, gegeven door

$$\frac{\partial u}{\partial t} \approx \frac{u(x, t+\Delta t) - u(x,t)}{\Delta t}\,,$$
$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u(x+\Delta x, t) - 2\,u(x,t) + u(x-\Delta x, t)}{\Delta x^2}\,,$$

of gebruikmakend van de conventie voor indexnummering zoals weergegeven in Fig. 4.7:

$$\frac{\partial u}{\partial t} \approx \frac{u_{i,j+1} - u_{i,j}}{\Delta t}\,,$$
$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1,j} - 2\,u_{i,j} + u_{i-1,j}}{\Delta x^2}\,.$$

Substitutie van deze benaderingen in partiële DV (4.14) levert

$$\frac{u_{i,j+1} - u_{i,j}}{\Delta t} = \kappa\,\frac{u_{i+1,j} - 2\,u_{i,j} + u_{i-1,j}}{\Delta x^2}\,, \tag{4.15}$$

terwijl uit de discretisatie van de rand- en beginvoorwaarden volgt dat $u_{i,1} = g(x_i)$, $u_{1,j} = u_l$ en $u_{n,j} = u_r$. Bijgevolg is het duidelijk dat u gekend is in alle knopen die gelegen zijn op de linker-, rechter- en ondergrens van het eindig differentiegrid. Hieruit volgt dat de benaderingen in de overige knooppunten makkelijk te bepalen zouden zijn indien $u_{i,j+1}$ kan uitgedrukt worden i.f.v. benaderingen in knopen gelegen op de j-de rij. Immers, vermits u gekend is in alle knopen op de eerste rij, zou het dan mogelijk zijn om de benaderingen in alle knopen van de tweede rij te bepalen waarna deze in knopen op de derde rij kunnen berekend worden, enzovoort, tot de m-de rij in het eindig differentiegrid bereikt wordt.

Indien we Vgl. (4.15) oplossen naar $u_{i,j+1}$, vinden we

$$u_{i,j+1} = \kappa\,\frac{u_{i+1,j} + u_{i-1,j}}{\Delta x^2}\,\Delta t + \left(1 - \frac{2\,\kappa\,\Delta t}{\Delta x^2}\right)\,u_{i,j}\,, \tag{4.16}$$

hetgeen aanduidt dat $u_{i,j+1}$ inderdaad kan geschreven worden als een functie van de benaderingen op de j-de rij. Bijgevolg kunnen we het randwaardeprobleem benaderen door Vgl. (4.16) te schrijven voor elk van de knopen gelegen op de tweede rij van het eindig differentiegrid, waarna de daaruit volgende benaderingen kunnen gebruikt worden om deze in de knopen op de derde rij te berekenen, enzovoort. Derhalve is het duidelijk dat $u_{i,j+1}$ recursief kan berekend worden gebruikmakend van gekende $u_{i,j}$ zonder dat de berekeningen moeten herhaald worden zoals bij de benadering van DV (4.7) noodzakelijk was. Dit betreft dan ook een **expliciete** methode in tegenstelling tot de **impliciete** oplossingsmethode die werd gehanteerd voor de benadering van DV (4.7). Zulke expliciete methodes hebben als belangrijkste nadeel dat zij slechts aanvaardbare resultaten opleveren op voorwaarde dat voldaan is aan de ongelijkheid van Courant-Friedrichs-Lewy (Courant et al., 1928), die voor de bestudeerde warmtevergelijking wordt gegeven door

$$\frac{\kappa\,\Delta t}{(\Delta x)^2} \leq \frac{1}{2}\,. \tag{4.17}$$

Kiezen we ter illustratie $\kappa = 1$ en $\Delta x = 0.1$, dan volgt $\Delta t \leq 0.005$, wat op het strenge karakter van deze voorwaarde wijst. Om dit te omzeilen werden er impliciete methodes, zoals de Crank-Nicolson methode, ontwikkeld, maar hiervoor verwijzen we naar de literatuur (e.g. Coleman, 2005; Li and Chen, 2008).

Het karakter van de oplossingsmethode (impliciet of expliciet) hangt af van het randwaardeprobleem. Enerzijds speelt de discretisatie van de DV een rol, anderzijds de beschikbare randvoorwaarden. Een expliciete oplossingsmethode vereist dat alle waarden in het rechterlid van de gediscretiseerde DV beschikbaar zijn uit de randvoorwaarden om de berekeningen aan te vatten. Dit hangt dan uiteraard af van de discretisatie van de DV.

Zo zal, indien de eerste orde afgeleide uit DV (4.14) wordt vervangen door de achterwaartse eindige differentiebenadering, en de vergelijking opnieuw wordt omgevormd naar $u_{i,j}$, het probleem impliciet opgelost moeten worden. Dit is ook het geval wanneer, bij de oorspronkelijke discretisatie van de DV, de randvoorwaarde $u(0,t) = 0$ als $t > 0$ niet gegeven zou zijn. Zowel de discretisatie van de DV als de aanwezige randvoorwaarden bepalen dus de oplossingsmethode.

Opdracht 4.3

Beschouw het randwaardeprobleem voorgesteld in deze sectie met $\kappa = 0.02$, $\Delta x = 0.025$ *en* $\Delta t = 0.015$. *Ga na dat deze keuze voor* Δt *voldoet aan ongelijkheid* (4.17).

1. *We wensen het randwaardeprobleem* (4.14), *waarbij* $u_l = 0\,, u_r = 100$ *en* $g(x) = 0$, *op te lossen m.b.v. de eindige differentiemethode. Algoritme 1 geeft de pseudo-code weer die je hierbij kan helpen.*

Algoritme 1: Pseudo-code voor het oplossen van het randwaardeprobleem (4.14).

Initialiseer een vector `t` die voor alle j het overeenkomstige benaderingstijdstip bevat;
Initialiseer een vector `x` die voor alle i de overeenkomstige x-coördinaat bevat;
Initialiseer m.b.v. de functie `zeros`, een $m \times n$-matrix U waarin alle $u_{i,j}$ bewaard dienen te worden. Hou hierbij reeds rekening met eventuele imaginaire knopen die toegevoegd moeten worden;
Implementeer de gepaste randvoorwaarden;
for *tijdstap* j **do**
    for *positie* i **do**
        Bepaal $u_{i,j+1}$;
    Implementeer de gepaste randvoorwaarden;
Plot u i.f.v. x en t m.b.v. de functie `surf`. Doe dit als volgt:

```
opp = surf(x, t , U);
set(opp, 'linestyle', 'none');
```

Maak een figuur waarin per 20 tijdstappen een plot van u i.f.v. x wordt afgebeeld;

2. *Bepaal de oplossing van het randwaardeprobleem indien* $u_l = u_r = 0$, *en*

$$g(x) = \begin{cases} x\,, & \text{als } 0 \le x \le \frac{1}{2}\,, \\ 1 - x\,, & \text{als } \frac{1}{2} < x \le 1\,. \end{cases}$$

   Maak een 3-D plot van u i.f.v. x en t, alsook een figuur waarin per 20 tijdstappen een plot van $u(x)$ *wordt afgebeeld.*

3. *Beschouw de partiële DV*

$$\frac{\partial u}{\partial t} = 0.02\,\frac{\partial^2 u}{\partial x^2} + e^{-x}\,,$$

   over het gebied $[0,1] \times [0,15]$, *onderhevig aan de beginwaarde*

$$u(x,0) = \sin(2\,\pi\,x)$$

   en de randvoorwaarden

$$u(0,t) = 0\,, \qquad u(1,t) = 0\,, \qquad \text{als } t > 0\,.$$

   (a) *Bepaal met Mathematica de steady-state* ($\frac{\partial u}{\partial t} = 0$) *oplossing van dit randwaardeprobleem.*

   $u(x) =$

(b) Maak een 3-D plot van u i.f.v. x en t, alsook een figuur waarin per 20 tijdstappen een plot van $u(x)$ wordt afgebeeld. Plot bovenop de tweede figuur de steady-state oplossing die je onder (a) gevonden hebt.

Opdracht 4.4

Beschouw een homogene aardappelknoedel met straal R die wordt ondergedompeld in langzaam opwarmend water voor het maken van een Duits knoedelgerecht. Indien we de positie in de aardappelknoedel uitdrukken in bolcoördinaten, dan kan de temperatuursevolutie $u(\rho, t)$ beschreven worden a.d.h.v. de volgende partiële DV:

$$\frac{\partial u}{\partial t} = \kappa \left(\frac{\partial^2 u}{\partial \rho^2} + \frac{2}{\rho} \frac{\partial u}{\partial \rho} \right) , \tag{4.18}$$

over $[0.025, R\,[\times [0, +\infty\,[$*, die voldoet aan de randvoorwaarden*

$$u(R, t) = T_1 - (T_1 - T_0)\, e^{-\alpha\, t} , \qquad 0 < t < \infty ,$$
$$\frac{\partial u(0.025, t)}{\partial \rho} = 0 , \qquad 0 < t < \infty ,$$

en de beginwaarde

$$u(\rho, 0) = T_0 , \qquad 0 \leq \rho \leq R ,$$

waarbij T_0 de initiële temperatuur van de kookvloeistof voorstelt en T_1 het kookpunt van deze vloeistof. Je dient de oplossing van dit randwaardeprobleem te benaderen m.b.v. de eindige differentiemethode. Hiervoor moet je gebruik maken van het eindig differentiegrid en de conventie voor indexnummering zoals weergegeven in Fig. 4.8.

1. *Duid in Fig. 4.8 aan in welke knopen er een Dirichlet of Neumann randvoorwaarde geldt. Geef tevens aan waar eventuele imaginaire knopen dienen toegevoegd te worden.*

2. *Discretiseer de partiële DV* (4.18)*, rekening houdend met de afspraken voor indexnummering weergegeven in Fig. 4.8, en los de bekomen uitdrukking op naar $u_{i\,j+1}$.*

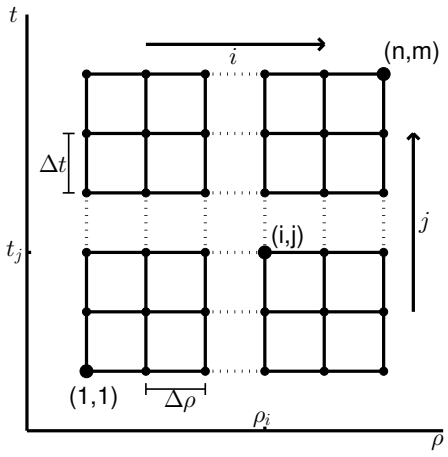
3. *Benader de oplossing van het randwaardeprobleem over het tijdsinterval* $[0, 500]$ *indien $R = 2$ cm, $\alpha = 0.01\ s^{-1}$, $\kappa = 0.01\ cm^2 s^{-1}$, $\Delta t = 0.0125$, $\Delta\rho = 0.025$, $T_1 = 100\ ^\circ C$, en $T_0 = 20\ ^\circ C$. Maak hiervoor gebruik van Algoritme 2 dat de pseudo-code weergeeft van de te volgen werkwijze.*

4. *Wat is de minimale temperatuur $u_{\min}$ in de aardappelknoedel op het einde van het beschouwde tijdsinterval?*

   $u_{\min} =$

5. *Plot u i.f.v. ρ en t m.b.v. de functie* `surf`.

6. *Maak een figuur waarin per 500 tijdstappen een plot van u i.f.v. ρ wordt afgebeeld.*



Figuur 4.8: Eindig differentiegrid gebruikt voor het oplossen van DV (4.18).

Algoritme 2: Pseudo-code voor het oplossen van partiële DV (4.18).

```
Initialiseer een vector t die voor alle j het overeenkomstige benaderingstijdstip bevat;
Initialiseer een vector rho die voor alle i de overeenkomstige ρ-coördinaat bevat;
Initialiseer m.b.v. de functie zeros een m × n-matrix U waarin alle u_{i,j} bewaard dienen
te worden. Hou hierbij reeds rekening met eventuele imaginaire knopen die toegevoegd
moeten worden!;
Implementeer de gepaste randvoorwaarden;
for tijdstap j do
    for positie i do
        Bepaal u_{i,j+1};
    Implementeer de gepaste randvoorwaarden;
```

Tot slot wensen we te benadrukken dat dit practicum vooral tot doel heeft om de basisprincipes van de eindige differentiemethode mee te geven zonder hierbij in detail te treden. Niettemin vormt de hier aangebrachte methodiek de basis van meer gesofisticeerde methodes, die mogelijk aan bod komen in opleidingsspecifieke cursussen, zoals Modelleren en Simuleren 2.

5

Stelsels eerste-orde differentiaalvergelijkingen

In dit practicum zullen we de in Practica 2 en 3 behandelde numerieke methodes op natuurlijke wijze uitbreiden naar stelsels eerste-orde DVn, die kunnen geschreven worden als:

$$\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}) \,,$$

of expliciet

$$\begin{cases} y_1'(t) = f_1(t, y_1, y_2, \ldots, y_m) \,, \\ \qquad \vdots \\ y_m'(t) = f_m(t, y_1, y_2, \ldots, y_m) \,, \end{cases}$$

over een interval $[t_0, t_n]$ bij beginwaarden $\mathbf{y}(t_0) = \mathbf{y}_0 = ((y_0)_1, \ldots, (y_0)_m)$. Zoals bij eerste-orde DVn zullen we de oplossing $\mathbf{y}(t) = (y_1(t), \ldots, y_m(t))$ van het stelsel in een aantal punten

$$t_k = t_0 + k\,\Delta t, \qquad k = 0, \ldots, n$$

benaderen bij een constante Δt.

Naast het directe belang van stelsels eerste-orde DVn, dat te wijten is aan hun veelvuldig gebruik voor het beschrijven van natuurlijke processen, zijn zij ook belangrijk bij de studie van n-de orde DV van de vorm

$$y^{(n)} = f(t, y, y', y'', \ldots, y^{(n-1)}) \,,$$

vermits we elke dergelijke n-de orde DV kunnen omzetten naar een stelsel eerste-orde DVn. In Practicum 6 zullen we dit aspect verder behandelen.

5.1 Stelsels in Mathematica

Mathematica is in staat stelsels DVn rechtstreeks op te lossen. Het volstaat in de Mathematica instructies uit Sectie 1.1 de DV `dv` te vervangen door een verzameling `{dv1, ..., dvn}` bestaande uit de DVn van het beschouwde stelsel. Voor het vinden van een unieke oplossing volstaat het begin- en/of randvoorwaarden `BV1, ..., BVn` op te nemen binnen het paar accolades:

```
DSolve[{dv1, ..., dvn, BV1, ..., BVn}, {y1[x], ..., yn[x]}, x]
```

Opdracht 5.1

Beschouw het stelsel DVn

$$\begin{cases} y_1'(t) = y_1 + 2\,y_2 - 2\,y_3 - \cos(t)\,, \\ y_2'(t) = 2\,y_1 + 5\,y_2 - 2\,y_3, \\ y_3'(t) = 4\,y_1 + 12\,y_2 - 5\,y_3\,, \end{cases}$$

waarvan de beginwaarden worden gegeven door $y_1(0) = 1, y_2(0) = 1, y_3(0) = 0$. *Bepaal met Mathematica de unieke oplossing van het beginwaardeprobleem. Gebruik (eventueel) de functie* `Simplify` *om de bekomen oplossing te vereenvoudigen.*

$$\begin{cases} y_1 = \\ \\ y_2 = \\ \\ y_3 = \end{cases}$$

5.2 Richtingsvelden

Voor de eenvoud zullen we ons beperken tot de studie van richtingsvelden voor een tweedimensionaal stelsel DVn

$$\begin{cases} y_1' = f_1(t, y_1, y_2)\,, \\ y_2' = f_2(t, y_1, y_2)\,. \end{cases}$$

De oplossing $\mathbf{y}(t) = (y_1(t),\, y_2(t))$ zal in het (y_1, y_2)-vlak een kromme beschrijven (parameter t) die we in het vervolg van deze nota's aanduiden met de term **fasebaan** (*phase space trajectory*). De eerste coördinaat $y_1(t)$ is immers als het ware een punt dat beweegt op de y_1-as; $y_2(t)$ zal dan een punt zijn dat beweegt op de y_2-as. Het (y_1, y_2)-vlak wordt het

fasevlak (*phase plane*) genoemd. De oplossing wordt gevisualiseerd door een punt met coördinaten $(y_1(t), y_2(t))$ dat bij toenemende tijd beweegt in het fasevlak. De bewegingsrichting in het punt op tijdstip t wordt gegeven door

$$\frac{f_2(t, y_1, y_2)}{f_1(t, y_1, y_2)}.$$

Algemener kunnen we op elk tijdstip t en in elk punt (y_1, y_2) van het fasevlak de richtingscoëfficiënt van de raaklijn aan elke oplossing door dit punt op het tijdstip t vinden uit $f_2(t, y_1, y_2)/f_1(t, y_1, y_2)$. Het teken van beide termen in dit quotiënt bepaalt de zin van de beweging. Doen we dit voor verschillende punten in het fasevlak, dan verkrijgen we het dynamisch richtingsveld in het fasevlak. Aangezien zowel f_1 als f_2 expliciet van de tijd kunnen afhangen, kan de richting en de zin van de richtingsvectoren variëren in de tijd. Analoog als bij eerste-orde DVn (Sectie 1.3) wordt een stelsel eerste-orde DVn autonoom genoemd indien $\mathbf{f}(t, \mathbf{y})$ niet expliciet van t afhangt: $y_i' = f_i(y_1, \ldots, y_m)$, voor elke $i \in \{1, \ldots, m\}$.

De functie `fasevlak` construeert het richtingsveld in het fasevlak voor een 2-dimensionaal stelsel DVn op verschillende tijdstippen en visualiseert de verandering van het richtingsveld i.f.v. de tijd als een filmpje indien f_1 en/of f_2 expliciet afhangt van t. De inputvariabelen van deze functie zijn:

1. de vectorfunctie $\mathbf{f}$;
2. het interval op de y_1-as (`xinterval`);
3. het interval op de y_2-as (`yinterval`);
4. het tijdsinterval `tinterval`;
5. en het aantal tijdstappen n waarop het richtingsveld berekend wordt.

Voor een optimale weergave van het richtingsveld in het fasevlak is het aan te raden om een gelijkaardige breedte te kiezen voor `xinterval` en `yinterval`.

Codefragment 5.1: fasevlak

```
function fasevlak(f, xinterval, yinterval, tinterval, n)
% Deze functie plot het richtingsveld van
% een stelsel van twee eerste-orde DVn in het fasevlak
% Syntax: fasevlak(f, xinterval, yinterval, tinterval, n)
% Input:  f: function handle naar het rl. van het stelsel DVn
%         grenzen van de grafiek:
%         xinterval = [xmin, xmax]
%         yinterval = [ymin, ymax]
%         tinterval = [tmin, tmax]
%         n = precisie van het fasevlak
```

```
% Output: fasevlak van het stelsel eerste-orde DVn
nk = 25;
xmin = xinterval(1);
xmax = xinterval(2);
ymin = yinterval(1);
ymax = yinterval(2);
dx = (xmax-xmin)/(nk);
dy = (ymax-ymin)/(nk);
x = xmin:dx:xmax;
y = ymin:dy:ymax;
tmin = tinterval(1);
tmax = tinterval(2);
dt = (tmax-tmin)/n;
if tmin == tmax
    t = tmin:dt+1:tmax;
else
    t = tmin:dt:tmax;
end
warning off MATLAB:Movie:ZeroRepetitions
F = struct('cdata',cell(1,length(t)),'colormap',cell(1,length(t)));
U = zeros(length(y), length(x));
V = zeros(length(y), length(x));
for k = 1:length(t)
    clf
    for i = 1:length(y)
        for j = 1:length(x)
            z = [x(j), y(i)]';
            zprime = f(t(k), z);
            L = sqrt(zprime(1)^2 + zprime(2)^2);
            U(i,j) = zprime(1)/L;
            V(i,j) = zprime(2)/L;
        end
    end
    quiver(x, y, U, V, 0.4)
    hold on
    axis([xmin xmax ymin ymax])
    title(['Fasevlak bij t = ' num2str(t(k))] )
    xlabel('y_1')
    ylabel('y_2')
    F(k) = getframe;
end
movie(F, 0)
```

```
end
```

Voorbeeld 5.1 Fasevlak van een stelsel DVn

Beschouw het volgende 2-dimensionaal stelsel DVn:

$$\begin{cases} y_1' = y_1 + 2\,y_2\,, \\ y_2' = 2\,y_1 + y_2\,, \end{cases} \tag{5.1}$$

over het t-interval $[0,1]$ *en met beginwaarden* $y_1(0) = 1$ *en* $y_2(0) = 25$*. Stelsel* (5.1) *werd eerst geïmplementeerd in de functie* `stelselVB` *als volgt:*

Codefragment 5.2: stelselVB

```
function ydot = stelselVB(t, y)
ydot = [(y(1) + 2*y(2));
        (2*y(1) + y(2))];
end
```

5

Vervolgens werd de functie `fasevlak` *gebruikt om het richtingsveld van dit stelsel DVn te visualiseren in het fasevlak a.d.h.v. volgende instructie:*

```
fasevlak(@stelselVB, [-2, 2], [-2, 2], [0, 1], 25);
```

Het fasevlak van dit stelsel wordt weergegeven in Fig. 5.1.

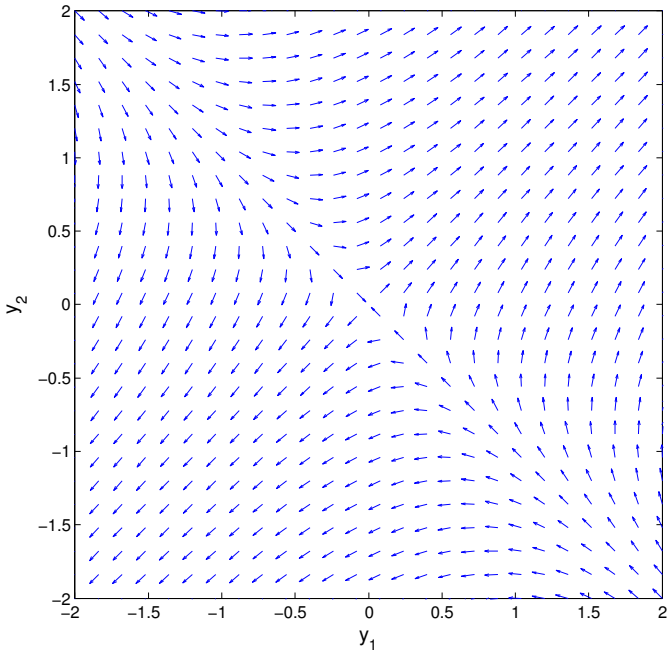
Ook voor 3-dimensionale stelsels zou men een ruimtelijk, dynamisch richtingsveld kunnen construeren, maar een uitbreiding van bovenstaande constructies naar n-dimensionale stelsels is praktisch moeilijk uitvoerbaar en hiervoor wordt verwezen naar gespecialiseerde literatuur.

Opdracht 5.2

Beschouw het volgende stelsel DVn:

$$\begin{cases} y_1' = \sin(t)\,y_1 + 2\,y_2 + \sin(t)\,, \\ y_2' = 2\,y_1 + \cos(t)\,y_2 - e^{2\,t}\,, \end{cases} \tag{5.2}$$

over het t-interval $[0,1]$ *(*$n = 50$*). Plot het fasevlak voor dit stelsel over het gebied* $[-2,6] \times [-4,4]$ *en beschrijf wat je ziet.*



Figuur 5.1: Richtingsveld in het fasevlak van Stelsel (5.1).

5.3 Evenwichtspunten

Net zoals bij eerste-orde DVn worden de evenwichtspunten van een stelsel $\mathbf{f}(t, \mathbf{y})$ gegeven door die vectoren $\mathbf{y}_e = ((y_e)_1, \ldots, (y_e)_m)$ waarvoor er een waarde t_c kan gevonden worden zodat $\mathbf{f}(t, \mathbf{y}_e) = \mathbf{0}$, voor alle $t \geq t_c$. In het richtingsveld van een 2-dimensionaal stelsel kan een evenwichtspunt geïdentificeerd worden als de beweging van de oplossing in het fasevlak tot stilstand komt in een welbepaald punt. Zowel in de y_1- als de y_2-richting is de snelheid dan nul geworden. De coördinaten van dit punt bepalen het evenwichtspunt $((y_e)_1, (y_e)_2)$, en zijn dus oplossingen van

$$\begin{cases} f_1(t, (y_e)_1, (y_e)_2) = 0\,, \\ f_2(t, (y_e)_1, (y_e)_2) = 0\,, \end{cases}$$

voor elke $t \geq t_c$.

In overeenstemming met eerste-orde DVn kan de stabiliteit van evenwichtspunten nagegaan worden door het beschouwen van twee omgevingen R_1 en R van het evenwichtspunt $((y_e)_1, (y_e)_2)$. In het fasevlak worden deze voorgesteld door open cirkelschijven (i.e. cirkelschijf zonder rand) met middelpunt $((y_e)_1, (y_e)_2)$ en stralen, respectievelijk, $\epsilon_1 > 0$ en $\epsilon \geq \epsilon_1$.

De classificatie van de evenwichtspunten gebeurt dan als volgt:

1. Een evenwichtspunt $((y_e)_1, (y_e)_2)$ is **stabiel** indien er voor iedere omgeving R van dit evenwichtspunt een omgeving $R_1 \subseteq R$ bestaat, zodat voor iedere beginwaarde $\mathbf{y}_0 \in R_1$ de bijhorende oplossing gedefinieerd is en in R ligt voor alle $t \geq t_c$. Oplossingen die

door $\mathbf{y}_0$ gaan, blijven in de buurt van het evenwichtspunt en zullen na een zekere tijd een **gesloten kromme** rondom het evenwichtspunt volgen waarnaar in de literatuur wordt gerefereerd met de term **limietcyclus** (*limit cycle*).

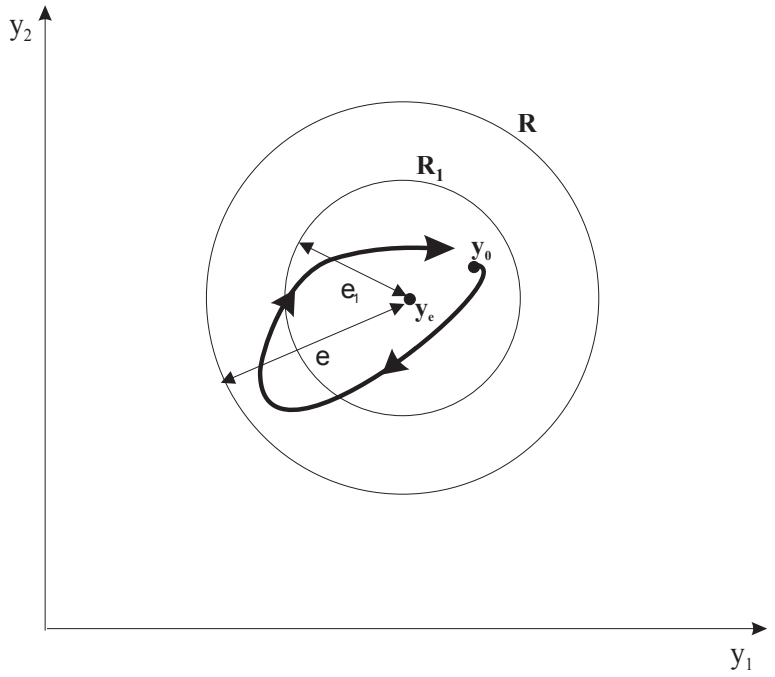
2. Een stabiel evenwichtspunt $((y_e)_1, (y_e)_2)$ is **asymptotisch stabiel** wanneer er een omgeving R_1 kan gekozen worden zodanig dat bovendien geldt dat voor iedere beginwaarde $\mathbf{y}_0 \in R_1$ de bijhorende oplossing $\mathbf{y}_{(t_0,y_0)}(t)$ voldoet aan:

$$\lim_{t\to+\infty} \mathbf{y}_{(t_0,y_0)}(t) = \mathbf{y}_e\,.$$

In dit geval zullen de oplossingen als het ware worden **aangetrokken** door het evenwichtspunt.

3. Een evenwichtspunt dat niet stabiel is, wordt een **onstabiel** evenwichtspunt genoemd. De naburige oplossingen **divergeren** van het evenwichtspunt.

Figuur 5.2 illustreert het gedrag van de oplossing van een 2-dimensionaal stelsel DVn in het fasevlak nabij een stabiel evenwichtspunt.



Figuur 5.2: Gedrag van de oplossing van een stelsel DVn met beginwaarden $\mathbf{y}_0$ nabij een stabiel evenwichtspunt.

Voorbeeld 5.2 Evenwichtspunten van stelsels DVn

Beschouw opnieuw Stelsel (5.1)*. De evenwichtspunten van dit stelsel worden bepaald door het oplossen van*

$$\begin{cases} (y_e)_1 + 2\,(y_e)_2 = 0\,, \\ 2\,(y_e)_1 + (y_e)_2 = 0\,. \end{cases}$$

Er bestaat slechts één oplossing voor dit stelsel zijnde, $(y_e)_1 = 0$ *en* $(y_e)_2 = 0$ *waaruit volgt dat Stelsel* (5.1) *slechts één evenwichtspunt heeft, nl. het punt* $(0, 0)$*. Het fasevlak weergegeven in Fig. 5.1 toont duidelijk dat dit een onstabiel evenwichtspunt is, aangezien de richtingsvectoren langsheen de eerste bissectrice in dit vlak van het evenwichtspunt weg wijzen. Kiezen we bijvoorbeeld een omgeving* R *rond het evenwichtspunt met straal* $\epsilon = 10$ *waarbinnen we een omgeving* R_1 *met straal* $\epsilon_1 = 5$ *kiezen, dan zal een oplossing corresponderend met beginwaarden* $\mathbf{y}(t_0)$ *gelegen in de omgeving* R_1 *niet begrensd worden door de omgeving* R*.*

Opdracht 5.3

1. *In Opdracht 1.4 belichtten we reeds een model voor de beschrijving van populatiegroei. Echter, dit model geldt enkel in afwezigheid van andere soorten. Indien we de populatiegroei van twee diersoorten (herbivoren* H *en predatoren* P*) die samenleven in eenzelfde gebied wensen te beschrijven, kunnen we gebruik maken van het volgende stelsel DVn:*

$$\begin{cases} H' = r\,H - d\,H\,P\,, & H(0) = 45\,, \\ P' = -s\,P + e\,H\,P\,, & P(0) = 55\,, \end{cases}$$

waarin r *de groeisnelheid van de herbivorenpopulatie in afwezigheid van predatoren is,* d *de mortaliteit van de herbivorenpopulatie ten gevolge van predatie,* s *de mortaliteit van de predatorenpopulatie in afwezigheid van herbivoren en* e *de groeisnelheid van deze populatie te wijten aan de predatie van herbivoren. Stel voor het vervolg van deze opdracht* $r = 0.4$*,* $d = 0.01$*,* $s = 0.3$ *en* $e = 0.005$*.*

   (a) *Bepaal (de) het eventuele evenwichtspunt(en) van dit stelsel. Je kan hiervoor gebruikmaken van Mathematica.*

   $(H_e, P_e) =$

   (b) *Implementeer dit stelsel in MATLAB in een m-file* `predPrey.m`*.*

   (c) *Controleer je antwoord op (a) door het fasevlak te plotten over het gebied* $[0, 100] \times [0, 100]$*. Wat is de stabiliteit van (de) het eventuele evenwichtspunt(en)?*

2. *Beschouw volgend niet-autonoom stelsel DVn*

$$\begin{cases} y_1' = y_1 + 2\,y_2 + 2\,e^{-t}\,, \\ y_2' = 2\,y_1 + y_2 - \dfrac{\ln t}{t} + 2\,. \end{cases} \tag{5.3}$$

Is het mogelijk om voor dit stelsel DVn het evenwichtspunt te bepalen? Zo ja, geef de bijhorende waarde voor t_c.

3.* *Wat is, gelet op je antwoord onder (2), een voorwaarde om voor een niet-autonoom stelsel DVn de evenwichtspunten te bepalen? Motiveer je antwoord. Kan je nog andere voorwaarden bedenken?*

4. *Beschouw het niet-lineair stelsel DVn*

$$\begin{cases} y_1' = \mu - y_1^2\,, \\ y_2' = -y_2\,, \end{cases} \tag{5.4}$$

waarbij μ een reële parameter voorstelt.

(a) Bespreek de stabiliteit van het (de) evenwichtspunt(en) i.f.v. de parameter μ.

(b) Schets het bifurcatiediagram.

5.4 De methode van Euler voor 2-dimensionale stelsels

In deze sectie beperken we ons tot de bespreking van de methode van Euler voor een tweedimensionaal stelsel:

$$\begin{cases} y_1' = f_1(t, y_1, y_2)\,, \\ y_2' = f_2(t, y_1, y_2)\,. \end{cases} \tag{5.5}$$

De uitbreiding naar m-dimensionale stelsels eerste-orde DVn verloopt op analoge wijze en wordt besproken in Sectie 5.5.

We willen de oplossing van dit stelsel benaderen over het interval $[t_0, t_n]$. Daartoe beschouwen we $n+1$ equidistante punten $t_k = t_0 + k\,\Delta t$, $k = 0, \ldots, n$, in het interval $[t_0, t_n]$. Zij $(y_k)_1$, resp. $(y_k)_2$, de benaderde waarde van $y_1(t_k)$, resp. $y_2(t_k)$, en veronderstel verder dat de beginwaarden gegeven worden door $\mathbf{y}(t_0) = (\,(y_0)_1\,, (y_0)_2\,)$.

Toepassing van de methode van Euler op elke vergelijking uit Stelsel (5.5) levert:

$$\begin{cases} y_1(t_{k+1}) \approx (y_k)_1 + \Delta t\, f_1(t_k, (y_k)_1\,, (y_k)_2)\,, \\ y_2(t_{k+1}) \approx (y_k)_2 + \Delta t\, f_2(t_k, (y_k)_1\,, (y_k)_2)\,. \end{cases}$$

Net zoals bij gewone DVn bepaalt f_1, resp. f_2, de richting van $y_1(t)$, resp. $y_2(t)$, in het (t, y_1)-vlak, resp. (t, y_2)-vlak. Zo zal, voor de beginwaarden $\mathbf{y}(t_0) = [\,(y_0)_1\,, (y_0)_2]$, $f_1(t_0, (y_0)_1\,, (y_0)_2)$ de richtingscoëfficiënt zijn van de oplossing $y_1(t)$ in het punt $t = t_0$. Op een gelijkaardige manier bepaalt $f_2(t_0, (y_0)_1\,, (y_0)_2)$ de richtingscoëfficiënt van $y_2(t)$ in het punt $t = t_0$. Merk hierbij op dat de oplossing $y_1(t)$ meestal afhangt van zowel t_0, $(y_0)_1$, alsook $(y_0)_2$. Bijgevolg zal (meestal) ook de oplossing $y_1(t)$ veranderen indien we $(y_0)_2$ veranderen.

Omdat MATLAB matrix-georiënteerd is, is het noodzakelijk om de methode van Euler eerst in matrixgedaante te schrijven vooraleer ze geïmplementeerd kan worden. Beschouw hiervoor de matrix

$$Y = \begin{bmatrix} y_{0,1} & y_{0,2} \\ y_{1,1} & y_{1,2} \\ \vdots & \vdots \\ y_{n,1} & y_{n,2} \end{bmatrix},$$

die de benaderingen van $\mathbf{y}(t) = [\, y_1(t) \;\; y_2(t) \,]$ op de desbetreffende tijdstippen bevat. Zij $\mathbf{t} = [\, t_0 \;\; t_1 \;\; \cdots \;\; t_n \,]^T$, dan vinden we de benaderingen van $y_1(t)$ en $y_2(t)$ op tijdstip $t = t_k$ (= $(k+1)$-ste element van de kolomvector $\mathbf{t}$) terug op de $(k+1)$-ste rij van Y: $Y(k+1,1) = y_{k,1}$ en $Y(k+1,2) = y_{k,2}$. Vervolgens introduceren we een functie $\mathbf{f}$. Ze beeldt een scalair t en een rijvector $\mathbf{y} = [\, y_1 \;\; y_2 \,]$ af op een kolomvector:

$$\mathbf{f}(t, \mathbf{y}) = \begin{bmatrix} f_1(t, y_1, y_2) \\ f_2(t, y_1, y_2) \end{bmatrix}.$$

Voor de $(k+1)$-ste rij van Y geldt er bijgevolg:

$$\begin{aligned} [\, y_{k,1} \;\; y_{k,2} \,] &= [\, y_{k-1,1} \;\; y_{k-1,2} \,] + \Delta t \, [\, f_1(t_{k-1}, y_{k-1,1}, y_{k-1,2}) \;\; f_2(t_{k-1}, y_{k-1,1}, y_{k-1,2}) \,], \\ &= [\, y_{k-1,1} \;\; y_{k-1,2} \,] + \Delta t \, \mathbf{f}(t_{k-1}, [\, y_{k-1,1} \;\; y_{k-1,2} \,])^T. \end{aligned}$$

Noteren we de k-de rij van Y als $Y(k,:)$, dan bekomen we

$$\begin{aligned} \mathbf{m}_1 &= \mathbf{f}(t(k), Y(k,:))^T, \\ Y(k+1,:) &= Y(k,:) + \Delta t \, \mathbf{m}_1. \end{aligned}$$

5.5 Algemene benaderingsmethodes

De Eulerbenadering voor een 2-dimensionaal stelsel eerste-orde DVn is onmiddellijk uitbreidbaar naar m-dimensionale stelsels:

$$\begin{cases} y_1'(t) = f_1(t, y_1, y_2, \ldots, y_m), \\ \qquad \vdots \\ y_m'(t) = f_m(t, y_1, y_2, \ldots, y_m), \end{cases} \tag{5.6}$$

die voldoen aan de beginwaarden $\mathbf{y}(t_0) = [(y_0)_1 \cdots (y_0)_m]$. We bekomen dat

$$\begin{cases} (y_{k+1})_1 = (y_k)_1 + \Delta t \, f_1(t_k, (y_k)_1, (y_k)_2, \ldots, (y_k)_m) \approx y_1(t_{k+1}), \\ \qquad \vdots \\ (y_{k+1})_m = (y_k)_m + \Delta t \, f_m(t_k, (y_k)_1, (y_k)_2, \ldots, (y_k)_m) \approx y_m(t_{k+1}), \end{cases} \tag{5.7}$$

voor elke, $k = 0, \ldots, n-1$. Meetkundig gezien, bepalen de functies $f_j(t_k, (y_k)_1, ..., (y_k)_m)$, $j = 1, ..., m$, de richting (in het (t, y_j)-vlak) van de j-de component $y_j(t)$ van de oplossing die voldoet aan de beginwaarden $\mathbf{y}(t_k) = [\, y_{k,1} \cdots y_{k,m}\,]$.

Algemeen zijn we op zoek naar de matrix

$$Y = \begin{bmatrix} y_{0,1} & y_{0,2} & \cdots & y_{0,m} \\ y_{1,1} & y_{1,2} & \cdots & y_{1,m} \\ \vdots & \vdots & \vdots & \vdots \\ y_{n,1} & y_{n,2} & \cdots & y_{n,m} \end{bmatrix}.$$

Zij $\mathbf{t} = [\, t_0 \;\; t_1 \;\; \cdots \;\; t_n\,]^T$, dan bevat de $(k+1)$-ste rij $Y(k+1, :)$ van Y de benaderingen van $\mathbf{y}(t) = [\, y_1(t) \;\; ... \;\; y_m(t)\,]$ voor $t = t_k$. Tot slot introduceren we ook de functie $\mathbf{f}$ die een scalair t en een rijvector $\mathbf{y}$ zal afbeelden op de kolomvector die $\mathbf{f}$ beschrijft:

$$\mathbf{f}(t, \mathbf{y}) = \begin{bmatrix} f_1(t, y_1, ..., y_m) \\ \vdots \\ f_m(t, y_1, ..., y_m) \end{bmatrix}.$$

Bijgevolg kunnen we Stelsel (5.7) in MATLAB-notatie schrijven als:

$$\mathbf{m}_1 = \mathbf{f}(t(k), Y(k, :))^T,$$

$$Y(k+1, :) = Y(k, :) + \Delta t\, \mathbf{m}_1.$$

De implementatie van deze Eulerbenadering voor m-dimensionale stelsels wordt gegeven door de MATLAB-functie `mijnEulerStelsel.m`:

Codefragment 5.3: mijnEulerStelsel

```
function [t, Y] = mijnEulerStelsel(f, tinterval, Y0, n)
% Benaderen van het stelsel y' = f(t, y) over tinterval = [t0, tn]
% met beginwaarden y(t0) = Y0
% stapgrootte dt = (tn - t0)/n
% Syntax: [t, Y] = mijnEulerStelsel(f, tinterval, Y0, n)
% Input:  f: function handle naar het rl. van het stelsel DVn
%         tinterval = [tmin, tmax]: grenzen van de grafiek
%         Y0 = beginwaarden
%         n = aantal stappen
% Output: t = tijdsvector
%         Y = benadering (elke kolom van Y is een benadering van
%             één DV van het stelsel DVn)
m = length(Y0);
Y = zeros(n+1, m);                    % pre-allocatie Y als
```

```
Y(1,:) = Y0;                                % kolommatrix
dt = (tinterval(2) - tinterval(1))/n;       % stapgrootte dt
t = (tinterval(1):dt:tinterval(2))';        % kolomvector t
for k = 1:n
    % benadering Y op het tijdstip t(k+1)
    m1 = f(t(k), Y(k,:))';
    Y(k+1,:) = Y(k,:) + dt*m1;
end
end
```

De methodes van Runge en Kutta voor een gewone eerste-orde DV kunnen eveneens uitgebreid worden voor m-dimensionale stelsels eerste-orde DVn van de vorm (5.6). De implementatie van de RK4-methode wordt gegeven in `mijnRK4Stelsel.m`.

Codefragment 5.4: mijnRK4Stelsel

```
function [t, Y] = mijnRK4Stelsel(f, tinterval, Y0, n)
% Benaderen van het stelsel y' = f(t, y) over tinterval = [t0, tn]
% met beginwaarden y(t0) = Y0
% stapgrootte dt = (tn - t0)/n
% Syntax: [t, Y] = mijnRK4Stelsel(f, tinterval, Y0, n)
% Input:  f: function handle naar het rl. van het stelsel DVn
%         tinterval = [tmin, tmax]: grenzen van de grafiek
%         Y0 = beginwaarden
%         n = aantal stappen
% Output: t = tijdsvector
%         Y = benadering (elke kolom van Y is een benadering van
%             één DV van het stelsel DVn)
m = length(Y0);
Y = zeros(n+1, m);                          % pre-allocatie Y als
Y(1,:) = Y0;                                % kolommatrix
dt = (tinterval(2) - tinterval(1))/n;       % stapgrootte dt
t = (tinterval(1):dt:tinterval(2))';        % kolomvector t
for k = 1:n
    % benadering Y op het tijdstip t(k+1)
    m1 = f(t(k), Y(k,:))'; % berekening van de richtingen
    m2 = f(t(k)+dt/2, (Y(k,:) + dt*m1/2))';
    m3 = f(t(k)+dt/2, (Y(k,:) + dt*m2/2))';
    m4 = f(t(k)+dt, (Y(k,:) + dt*m3))';
    Y(k+1,:) = Y(k,:) + dt*(1/6*m1 + 1/3*m2 + 1/3*m3 + 1/6*m4);
end
end
```

Merk op dat de MATLAB-functies `ode23`, `ode45`, enz. eveneens in staat zijn om stelsels eerste-orde DVn op te lossen. Met behulp van `mijnEulerStelsel`, `mijnRK4Stelsel` en de MATLAB-functies voor het benaderen van DVn, is het uiteindelijk mogelijk om de benaderingen van $y_i(t)$ te plotten in het (t, y_i)-vlak of, als het een twee-dimensionaal stelsel betreft, in het fasevlak d.m.v. de instructie

```
plot(Y(:,1), Y(:,2), '*')   % 2de kolom i.f.v. 1ste kolom
```

indien de benaderingen werden bewaard in een matrix `Y`. Echter, het resultaat hiervan toont de volledige kromme die de oplossing zal beschrijven tijdens het beschouwde tijdsinterval, terwijl we weten dat er met iedere t slechts één punt in het fasevlak overeenstemt vermits $y_1(t)$ en $y_2(t)$, en bijgevolg ook hun benaderingen, expliciet afhangen van t. Om dit te illustreren en voor een beter begrip van het concept fasebaan kun je gebruik maken van de functie `fasebaan` die voor een gegeven twee-dimensionaal stelsel DVn en een koppel beginwaarden de fasebaan construeert en visualiseert als een filmpje naarmate t, in een vooraf gedefinieerd aantal stappen, toeneemt. Het argument `P` in deze functie laat toe om te kiezen of je de beginwaarden expliciet wenst mee te geven of door middel van een linkermuisklik.

Codefragment 5.5: fasebaan

```
function fasebaan(f, xinterval, yinterval, tinterval, n, P)
% Deze functie benadert en plot de fasebaan doorheen het
% punt (y1(0), y2(0)) bovenop het richtingsveld.
% Hiervoor wordt gebruik gemaakt van de RK4-benadering in n
% stappen over het tijdsinterval tinterval.
% Syntax: fasebaan(f, xinterval, yinterval, tinterval, n, P)
% Input:  f: function handle naar het rl. van het stelsel DVn
%         grenzen van de grafiek:
%         xinterval = [xmin, xmax]
%         yinterval = [ymin, ymax]
%         tinterval = [tmin, tmax]
%         n = precisie van het richtingsveld
%         P = bepaalt hoe de beginwaarden worden meegegeven:
%               P = 1: met linkermuisklik, of
%               P = [y1(0), y2(0)]
% Output: fasebaan doorheen (y1(0), y2(0))
%         bovenop het richtingsveld
tmin = tinterval(1);
tmax = tinterval(2);
dt = (tmax-tmin)/n;
t = tmin:dt:tmax;
if isempty(t)
```

```
    t = tmin;
end
if length(P) == 1   % opgeven van de beginwaarden via muisklik
    [x0, y0, ~] = ginput(1);
    BV = [x0, y0];
elseif length(P) > 1
    BV = P;
end
F = struct('cdata',cell(1,length(t)),'colormap',cell(1,length(t)));
for k = 1:length(t)
    fasevlak(f, xinterval, yinterval, [t(k), t(k)], 1)
    hold on
    [~, Y] = mijnRK4Stelsel(f, [tmin, t(k)], BV, n);
    plot(Y(:,1), Y(:,2), 'r', 'LineWidth', 2)
    plot(Y(end,1), Y(end,2), 'k*', 'MarkerSize', 10)
    F(k) = getframe;
end
movie(F, 0)
end
```


Voorbeeld 5.3 Numerieke methodes voor stelsels differentiaalvergelijkingen

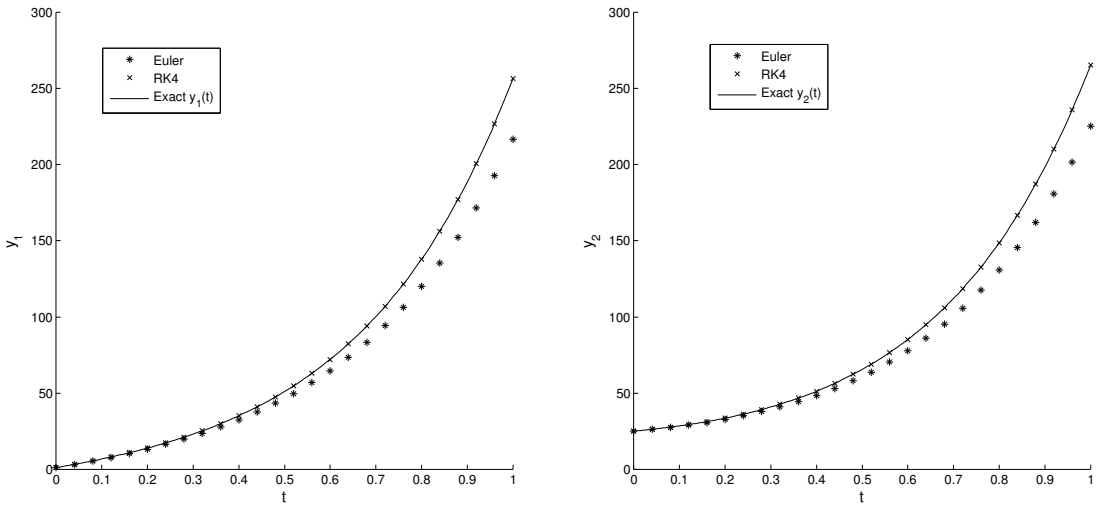
Beschouw opnieuw het 2-dimensionaal Stelsel (5.1) *over het interval* $[0, 1]$ *met beginwaarden* $y_1(0) = 1$ *en* $y_2(0) = 25$. *De exacte oplossing van dit beginwaardeprobleem wordt gegeven door:*

$$\begin{cases} y_1 = -12\,e^{-t} + 13\,e^{3\,t}, \\ y_2 = 12\,e^{-t} + 13\,e^{3\,t}\,. \end{cases}$$

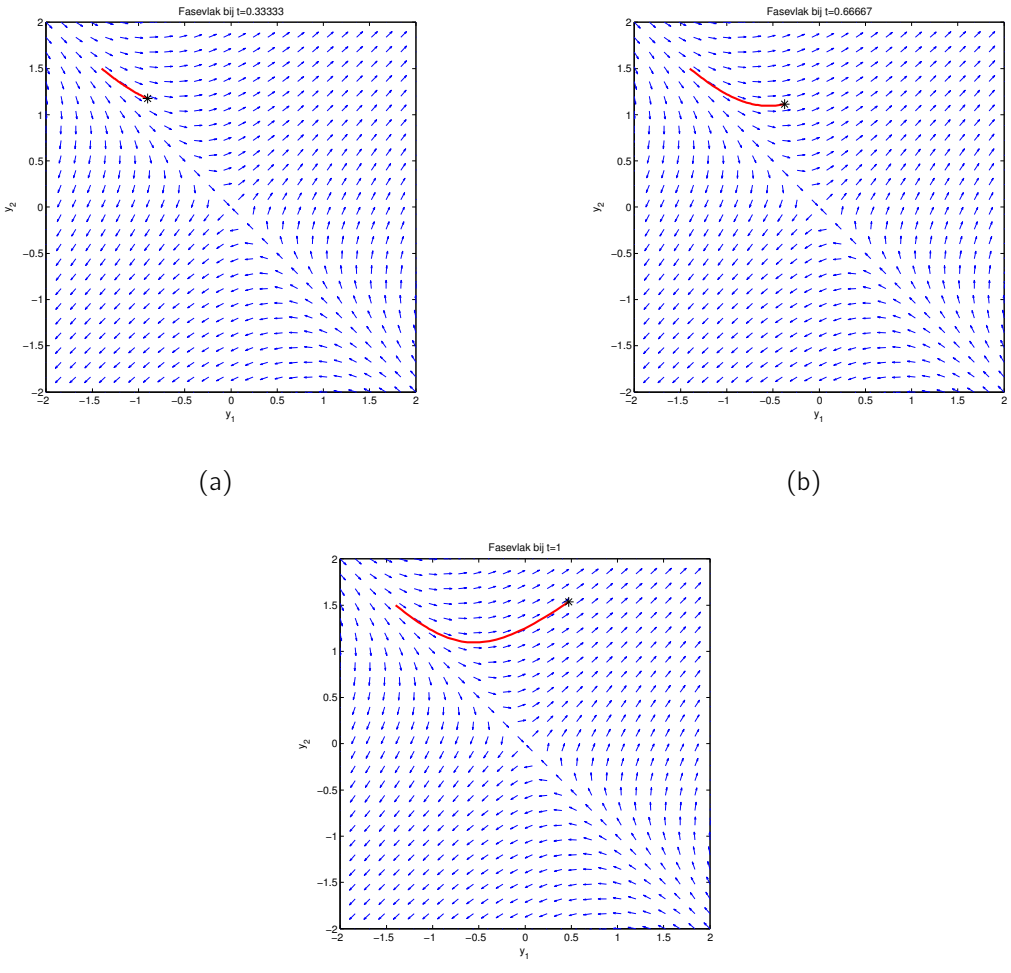
Figuur 5.3 geeft de Euler- en RK4-benaderingen ($n = 25$*) samen met de exacte oplossing weer. Het is duidelijk dat de RK4-benadering de beste benaderingen levert. Door kleine aanpassingen uit te voeren aan de in Practicum 2 voorgestelde functie* `functieverschil` *is het eveneens mogelijk om foutencurves te plotten voor deze benaderingen. We zullen hier echter niet dieper op ingaan. Figuur 5.4 visualiseert drie momentopnames van de constructie van de fasebaan doorheen het punt* $(-1.4, 1.5)$.

Opdracht 5.4

1. *Beschouw opnieuw het stelsel DVn voorgesteld in Opdracht 5.3 ter beschrijving van de populatiegroottes van herbivoren en predatoren. We werken over het gebied* $[0, 150] \times [0, 150]$ *en veronderstellen verder dat* $H(0) = 45$ *en* $P(0) = 55$.



Figuur 5.3: Exacte oplossing en Euler- en RK4-benaderingen van Stelsel (5.1) bij $n = 25$.



(a) (b)

(c)

Figuur 5.4: Drie momentopnames van het filmpje dat de constructie van de fasebaan van Stelsel (5.1) doorheen het punt $(-1.4, 1.5)$ weergeeft.

(a) *Benader de oplossing van dit stelsel over het tijdsinterval* $[0, 45]$ *door gebruik te maken van de Euler- en RK4-benadering bij* $n = 100$. *Plot de benaderde fasebanen in het fasevlak. Welke benadering is de beste? Wat kan je hieruit besluiten?*

(b) *Wat verwacht je dat er na verloop van tijd zal gebeuren met de populaties indien de herbivoren zich door gewijzigde milieuomstandigheden niet meer kunnen voortplanten* ($r = 0$)*?*

(c) *Plot de RK4-benaderingen voor dit scenario. Komt het resultaat overeen met jouw verwachtingen? Waarom wel/niet?*

2. *Het populatiemodel voorgesteld in Opdracht 5.3 heeft als belangrijke tekortkoming dat het geen rekening houdt met de draagkracht van de omgeving* K *hetgeen impliceert dat de herbivorenpopulatie oneindig groot kan worden in afwezigheid van predatoren. Dit is uiteraard niet mogelijk vermits de populatiegrootte ook afhankelijk is van de groeimogelijkheden die de omgeving biedt. Om deze draagkracht in rekening te brengen kan het model, voorgesteld in Opdracht 5.3, gecombineerd worden met de logistische groeivergelijking (zie Opdracht 1.4). Dit levert:*

$$\begin{cases} H' = r\left(1 - \dfrac{H}{K}\right) H - d\,H\,P\,, \\ P' = -s\,P + e\,H\,P\,, \end{cases}$$

waarin K *de draagkracht van de omgeving voorstelt. Voor deze opdracht wordt* K *gelijkgesteld aan 150. We werken over het tijdsinterval* $[0, 45]$ *en met beginwaarden* $H(0) = 45$ *en* $P(0) = 55$.

(a) *Implementeer dit stelsel in MATLAB in een m-file* `predPreyLogistisch.m`.

(b) *Bepaal analytisch het (de) evenwichtspunt(en) van dit groeimodel.*

$(H_e, P_e) =$

(c) *Plot diverse benaderde oplossingen in het fasevlak. Werk hierbij over het gebied* $[0, 150]^2$ *en neem aan dat* $r = 0.4$, $d = 0.01$, $s = 0.3$ *en* $e = 0.005$. *Bevestigt het fasevlak je bevindingen onder (3b)?*

(d) *Ga de stabiliteit van het (de) evenwichtspunt(en) na.*

3. *Stel nu dat door een ingrijpende wijziging van de milieuomstandigheden de draagkracht van de omgeving afneemt naarmate* t *toeneemt volgens de uitdrukking*

$$K(t) = K_{min} + 50 \exp\left(-\frac{t}{4}\right) ,$$

waarin $K_{min} = 100$ *de minimale draagkracht is van de omgeving. Pas de m-file* `predPreyLogistisch.m` *op gepaste wijze aan deze nieuwe situatie aan.*

(a) *Visualiseer het fasevlak over het tijdsinterval* $[0, 45]$ *(*$n = 100$*). Werk hierbij over het gebied* $[0, 150] \times [0, 150]$. *Wat zie je?*

(b) *Wat zijn de evenwichtspunten van dit stelsel DVn?*

$(H_e, P_e) =$

Wat zijn de te verwachten evenwichtspopulaties indien $t \to \infty$*?*

$(H_e, P_e) =$

4. *Van welk type evenwichtspunten kan je de coördinaten bepalen a.d.h.v. de grafieken voor de benaderingen van* $y_1(t)$ *en* $y_2(t)$*? Motiveer je antwoord.*

5.6 Stabiliteit van de numerieke methodes

Bij de benadering van stelsels eerste-orde DVn kunnen we net zoals bij eerste-orde DVn geconfronteerd worden met stabiliteitsproblemen onder de vorm van slecht geconditioneerde

en stijve stelsels eerste-orde DVn. De theoretische uiteenzetting hierover in Practicum 3 is onmiddellijk uitbreidbaar naar stelsels eerste-orde DVn.

Opdracht 5.5

De opwarming van de aarde en de mogelijke menselijke bijdrage daartoe is niet meer weg te denken uit de media en de politieke agenda. Meestal zijn uitspraken hieromtrent gebaseerd op modellen die de fysische processen verantwoordelijk voor de opwarming trachten te vatten in wiskundige vergelijkingen. Differentiaalvergelijkingen zijn een veel gebruikt voorbeeld van dergelijke verbanden. Onderstaand model bestaat uit een stelsel van vijf gekoppelde eerste-orde DVn dat de uitwisseling van koolstof tussen de belangrijkste koolstofstocks op aarde, zijnde de atmosfeer, de oceanen en de zeeën, beschrijft:

$$\begin{cases} p' = d^{-1}\,(p_s - p) + S(t)\,\mu_1^{-1}\,, \\ \sigma_s' = v_s^{-1}\,((\sigma_d - \sigma_s)\,w - k_1 - (p_s - p)\;d^{-1}\,\mu_2)\;, \\ \sigma_d' = v_d^{-1}\,(k_1 - (\sigma_d - \sigma_s)\,w)\;, \\ \alpha_s' = v_s^{-1}\,((\alpha_d - \alpha_s)\,w - k_2)\;, \\ \alpha_d' = v_d^{-1}\,(k_2 - (\alpha_d - \alpha_s)\,w)\;, \end{cases}$$

met

p : *de partieeldruk van koolfstofdioxide in de atmosfeer;*

σ_s : *de totale hoeveelheid koolstof opgelost in de zeeën;*

σ_d : *de totale hoeveelheid koolstof opgelost in de oceanen;*

α_s : *de alkaliniteit van de zeeën;*

α_d : *de alkaliniteit van de oceanen;*

en h_s, c_s en p_s gegeven door

$$h_s = \frac{\sigma_s - (\sigma_s^2 - k_3\,\alpha_s\,(2\,\sigma_s - \alpha_s))^{\frac{1}{2}}}{k_3}\,,$$

$$c_s = \frac{\alpha_s - h_s}{2}\,,$$

$$p_s = k_4\,\frac{h_s^2}{c_s}\,.$$

Tabel 5.1 geeft de waarden van de modelparameters, terwijl de implementatie van het stelsel DVn beschikbaar is in de m-file `carbonsim.m`.

carbonsim.m

```
function ydot = carbonsim(t, y)
global mu1 mu2 vs vd w d k1 k2 k3 k4

ydot = [(k4*(y(2)-(y(2)^2-k3*y(4)*(2*y(2)-y(4)))^0.5)^2/(y(4)/2-...
        (y(2)-(y(2)^2-k3*y(4)*(2*y(2)-y(4)))^0.5)/2)-y(1))/d+...
        % vul hier de functie-evaluatie in van fosbrand.m/mu1;
        1/vs*((y(3)-y(2))*w-k1-(k4*(y(2)-(y(2)^2-k3*y(4)*(2*y(2)-...
        y(4)))^0.5)^2/(y(4)/2-(y(2)-(y(2)^2-k3*y(4)*(2*y(2)...
        -y(4)))^0.5)/2)-y(1))/d*mu2);
        1/vd*(k1-(y(3)-y(2))*w);
        1/vs*((y(5)-y(4))*w-k2);
        1/vd*(k2-(y(5)-y(4))*w)];
end
```

Tabel 5.1: Modelparameters van het koolstofuitwisselingsmodel beschreven in Opdracht 5.5.

Parameter	Waarde	Parameter	Waarde
μ_1	$4.95\ 10^{2}$	d	8.64
μ_2	$4.95\ 10^{-2}$	k_1	$2.19\ 10^{-4}$
v_s	0.12	k_2	$6.12\ 10^{-5}$
v_d	1.23	k_3	0.997148
w	10^{-3}	k_4	$6.79\ 10^{-2}$

De variabele p wordt relatief uitgedrukt t.o.v. haar waarde bij aanvang van de industriële revolutie (1820). De eerste DV in het stelsel brengt de CO_2-uitstoot ten gevolge van het verbruik van fossiele brandstoffen in rekening. Dit verbruik is uiteraard niet constant in de tijd en kan beschreven worden a.d.h.v. de functie $S(t)$:

$$S(t) = \begin{cases} 0 & \text{, als } t < 1820\,, \\ a\,\exp\left(\dfrac{-(b-t)^2}{2\,c^2}\right) & \text{, als } 1820 \leq t \leq 2500\,, \\ 0 & \text{, als } 2500 < t\,, \end{cases}$$

waarbij $a = 10.1621$, $b = 2098.38$, $c = 79.3463$ en de tijd t uitgedrukt wordt in jaren.

1. *Implementeer $S(t)$ in een m-file en bewaar deze in een m-file* `fosbrand.m`.
2. *Plot $S(t)$ over het tijdsinterval* $[1820, 2500]$.
3. *Vervang in de eerste DV van het stelsel dat geïmplementeerd is in* `carbonsim.m` *de opmerking tussen % door de functie-evaluatie van $S(t)$ op het tijdstip t.*

4. *Benader de oplossing van het stelsel over het tijdsinterval* $[1000, 5000]$ *m.b.v. de MATLAB-functie* `ode45`*. Maak hiervoor gebruik van het script* `opdracht5_5.m` *dat beschikbaar is op Minerva, en waarin de waarden van de modelparameters (Tab. 5.1) reeds ingevuld werden. Plot* p, σ_s *en* σ_d *i.f.v. de tijd op een eerste figuur en* α_s *en* α_d *op een tweede figuur. Ga hiervoor uit van volgende beginwaarden op het tijdstip* $t = 1000$*:* $p = 1$, $\sigma_s = 2.01$, $\sigma_d = 2.23$, $\alpha_s = 2.2$ *en* $\alpha_d = 2.26$.

5. *Zou je het stelsel DVn als stijf bestempelen? Motiveer waarom wel/niet. Om je bewering te verifiëren, kun je de benadering herhalen met één van de MATLAB-functies voor stijve DVn (bv.* `ode15s`*).*

6. *Wanneer bereikt de hoeveelheid koolstof aanwezig in de atmosfeer haar maximale waarde? TIP: raadpleeg de MATLAB-help voor meer informatie over de functie* `max`*, of maak gebruik van logische indexering.*

5

7. *Vaak stelt men dat de verandering van de koolstofconcentratie in zeeën en oceanen naijlt op veranderingen in de atmosferische koolstofconcentratie. Wordt dit bevestigd door dit model?*

   In welk jaar bereikt de koolstofconcentratie in de zeeën haar maximum?

8. *Hoeveel jaren, tot op één jaar nauwkeurig, duurt het vooraleer de piek in* CO_2*-vrijstelling ten gevolge van menselijke activiteit zich vertaalt in een piek in de atmosferische koolstofconcentratie?*

9. *Zal het systeem volgens jou een evenwicht bereiken voor* $t \to \infty$*? Je kunt je bewering staven door de benadering te herhalen over een groter tijdsinterval, bijvoorbeeld* $[1000, 15000]$.

6

Differentiaalvergelijkingen van hogere orde

Numerieke benaderingsmethodes voor eerste-orde DVn van de vorm $y' = f(t, y)$ of voor stelsels, opgebouwd uit dergelijke DVn, werden reeds uitvoerig besproken in de voorgaande practica. We zullen deze methodes nu gebruiken om de oplossingen van n-de orde DVn

$$y^{(n)} = f\left(t, y, y', y'', ..., y^{(n-1)}\right) ,$$

of oplossingen van stelsels, opgebouwd uit dergelijke DVn, numeriek te benaderen. Het volstaat hierbij elke n-de orde DV om te vormen naar n eerste-orde DVn. De technieken uit Practicum 5 zijn vervolgens onmiddellijk toepasbaar. Tot slot van dit practicum onderzoeken we de stabiliteit van enkele numerieke methodes met betrekking tot DVn van hogere orde.

6.1 Herschrijven van hogere-orde differentiaalvergelijkingen

Elke n-de orde DV $y^{(n)} = f(t, y, y', y'', \ldots, y^{(n-1)})$ kan omgezet worden naar een stelsel eerste-orde DVn. Daartoe introduceren we n nieuwe veranderlijken

$$y_1 = y,\ y_2 = y',\ \ldots,\ y_n = y^{(n-1)} .$$

Afleiden naar de veranderlijke t levert:

$$\begin{cases} y_1' = y_2 , \\ y_2' = y_3 , \\ \quad \vdots \\ y_{n-1}' = y_n , \\ y_n' = f(t, y_1, y_2, \ldots, y_n) . \end{cases} \tag{6.1}$$

Indien $y(t_0) = \alpha^{(0)}$, $y'(t_0) = \alpha^{(1)}$, ..., $y^{(n-1)}(t_0) = \alpha^{(n-1)}$, dan worden de beginwaarden van het stelsel gegeven door:

$$\mathbf{y}_0 := \mathbf{y}(t_0) = [\, y_1(t_0) \; \ldots \; y_n(t_0) \,] = [\, y(t_0) \; \ldots \; y^{(n-1)}(t_0) \,] = [\, \alpha^{(0)} \; \ldots \; \alpha^{(n-1)} \,] \,.$$

Zij $\mathbf{y}(t)$ de oplossing van (6.1) die voldoet aan $\mathbf{y}(t_0) = \mathbf{y}_0$, dan is $y_1(t) = y(t)$ de oplossing van het oorspronkelijke beginwaardeprobleem.

Analoog kunnen we een stelsel DVn

$$\begin{cases} y_1^{(n_1)} = f_1\left(t, y_1, \ldots, y_1^{(n_1-1)}, y_2, \ldots, y_2^{(n_2-1)}, \ldots, y_m, \ldots, y_m^{(n_m-1)}\right) , \\ y_2^{(n_2)} = f_2\left(t, y_1, \ldots, y_1^{(n_1-1)}, y_2, \ldots, y_2^{(n_2-1)}, \ldots, y_m, \ldots, y_m^{(n_m-1)}\right) , \\ \qquad \vdots \\ y_m^{(n_m)} = f_m\left(t, y_1, \ldots, y_1^{(n_1-1)}, y_2, \ldots, y_2^{(n_2-1)}, \ldots, y_m, \ldots, y_m^{(n_m-1)}\right) , \end{cases}$$

omzetten naar een stelsel $n_1 + n_2 + \ldots + n_m$ eerste-orde DVn

$$\begin{cases} x_1' = x_2 \,, \\ \qquad \vdots \\ x_{n_1-1}' = x_{n_1} \,, \\ x_{n_1}' = f_1\left(t, x_1, x_2, \ldots, x_{n_1+\ldots+n_m}\right) , \\ x_{n_1+1}' = x_{n_1+2} \,, \\ \qquad \vdots \\ x_{n_1+n_2-1}' = x_{n_1+n_2} \\ x_{n_1+n_2}' = f_2\left(t, x_1, x_2, \ldots, x_{n_1+\ldots+n_m}\right) , \\ \qquad \vdots \\ x_{n_1+\ldots+n_{m-1}+1}' = x_{n_1+\ldots+n_{m-1}+2} \,, \\ \qquad \vdots \\ x_{n_1+\ldots+n_m-1}' = x_{n_1+\ldots+n_m} \,, \\ x_{n_1+\ldots+n_m}' = f_m\left(t, x_1, x_2, \ldots, x_{n_1+\ldots+n_m}\right) , \end{cases} \tag{6.2}$$

waarbij

$$\begin{cases} x_1 = y_1 \,, \; x_2 = y_1' \,, \; \ldots \,, \; x_{n_1} = y_1^{(n_1-1)} \,, \\ x_{n_1+1} = y_2 \,, \; x_{n_1+2} = y_2' \,, \; \ldots \,, \; x_{n_1+n_2} = y_2^{(n_2-1)} \,, \\ \qquad \vdots \\ x_{n_1+\ldots+n_{m-1}+1} = y_m \,, \; x_{n_1+\ldots+n_{m-1}+2} = y_m' \,, \; \ldots \,, \; x_{n_1+\ldots+n_m} = y_m^{(n_m-1)} \,. \end{cases}$$

Met de beginwaarden van het oorspronkelijk stelsel

$$\begin{cases} y_1(t_0) = \alpha_1^{(0)}\,,\ y_1'(t_0) = \alpha_1^{(1)}\,,\ \ldots\,,\ y_1^{(n_1-1)}(t_0) = \alpha_1^{(n_1-1)}\,, \\ y_2(t_0) = \alpha_2^{(0)}\,,\ y_2'(t_0) = \alpha_2^{(1)}\,,\ \ldots\,,\ y_2^{(n_2-1)}(t_0) = \alpha_2^{(n_2-1)}\,, \\ \quad\vdots \\ y_m(t_0) = \alpha_m^{(0)}\,,\ y_m'(t_0) = \alpha_m^{(1)}\,,\ \ldots\,,\ y_m^{(n_m-1)}(t_0) = \alpha_m^{(n_m-1)}\,, \end{cases}$$

correspondeert er een vector

$$\mathbf{x}_0 = (\,\alpha_1^{(0)}\,,\ldots,\ \alpha_1^{(n_1-1)},\ \alpha_2^{(0)}\,,\ldots,\ \alpha_2^{(n_2-1)},\ \ldots,\ \alpha_m^{(0)},\ \ldots,\ \alpha_m^{(n_m-1)}\,)\,,$$

die de beginwaarden $\mathbf{x}(t_0) = \mathbf{x}_0$ van het overeenkomstige stelsel eerste-orde DVn karakteriseert. Zij

$$\mathbf{x}(t) = (\,x_1(t)\ x_2(t)\ \ldots\ x_{n_1+\ldots+n_m}(t))$$

de oplossing van (6.2) die voldoet aan $\mathbf{x}(t) = \mathbf{x}_0$, dan wordt de oplossing van het oorspronkelijke beginwaardeprobleem gegeven door

$$y_1(t) = x_1(t),\ y_2(t) = x_{n_1+1}(t),\ \ldots\,,\ y_m(t) = x_{n_1+\ldots+n_{m-1}+1}(t)\,.$$

Voorbeeld 6.1 Numerieke methodes voor hogere-orde DVn

De hoek $\theta(t)$ die een wrijvingsloze slinger maakt met de verticale op het tijdstip t wordt beschreven door de tweede-orde DV

$$\theta'' + \sin(\theta) = 0\,, \quad \theta(0) = 0\,, \quad \theta'(0) = a. \tag{6.3}$$

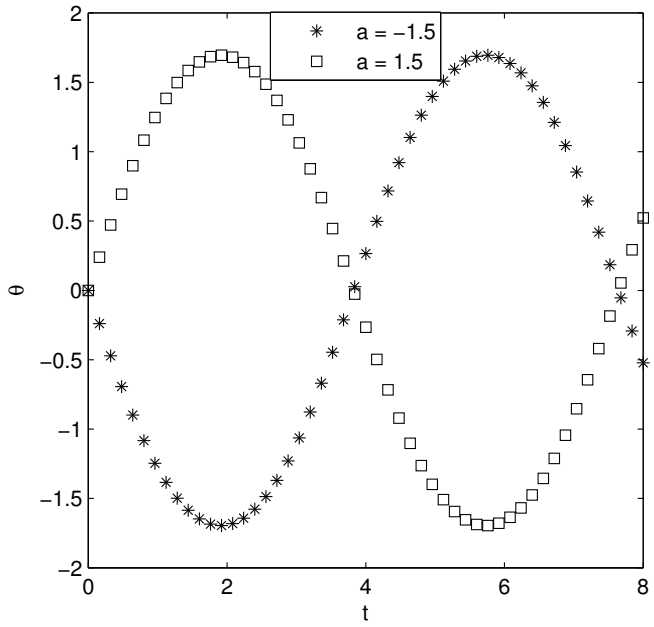
Introduceren we de veranderlijken

$$y_1 = \theta,\ y_2 = \theta'\,,$$

dan herleidt DV (6.3) *zich tot een stelsel autonome eerste-orde DVn:*

$$\begin{cases} y_1' = y_2\,, \\ y_2' = -\sin(y_1)\,, \end{cases}$$

met beginwaarden $y_1(0) = 0$ en $y_2(0) = a$. θ' is de hoeksnelheid, d.i. de verandering van de hoek in de tijd. Ondanks zijn eenvoudige vorm heeft dit stelsel geen analytische oplossing en is er slechts een benadering mogelijk m.b.v. de methodes geïntroduceerd in Practicum 5. Het resultaat van de RK4-benadering bij $n = 50$ wordt weergegeven in Fig. 6.1.



Figuur 6.1: Benadering van de oplossing van Stelsel (6.3) m.b.v. de RK4-methode bij $n = 50$.

6.2 Evenwichtspunten en stabiliteit van numerieke methodes

6.2.1 Evenwichtspunten

Nadat een n-de orde DV werd omgevormd tot een stelsel van eerste-orde DVn, kan de bepaling van de evenwichtspunten en hun stabiliteit gebeuren zoals in Practicum 5 werd beschreven voor stelsels eerste-orde DVn. Een evenwichtspunt van een n-de orde DV kunnen we dan voorstellen als $(y_e', y_e'', \ldots, y_e^{(n)})$. Algemeen kan het richtingsveld van een n-de orde DV voorgesteld worden in de n-dimensionale ruimte. Echter, in deze practica beperken we ons tot een tweedimensionale voorstelling, die gegenereerd kan worden met de functie `fasevlak`. Het fasevlak van een derde-orde DV zouden we nog kunnen voorstellen in de driedimensionale ruimte, maar van zodra $n > 3$ is een visualisatie niet langer evident.

6.2.2 Stabiliteit van numerieke methodes

De definities van slecht geconditioneerde en stijve DVn, vermeld in Practicum 3, kunnen intuïtief worden uitgebreid naar n-de orde DVn.

Opdracht 6.1

1. *Beschouw DV* (6.3) *over het interval* $[0, 8]$.

   (a) *Implementeer het corresponderende stelsel eerste-orde DVn in een MATLAB-functie en bewaar deze in een m-file* `slinger.m`.

   (b) *Benader de oplossing van de DV met de RK4-methode bij* $n = 50$ *voor drie beginvoorwaardeproblemen, namelijk* $a = 1.95$, $a = 2$ *en* $a = 2.05$. *Plot de benaderingen voor* $\theta(t)$ *op één figuur. Wat zie je?*

   (c) *Plot nu het fasevlak over het gebied* $[-4, 4] \times [-4, 4]$ *en de RK4-benadering over het tijdsinterval* $[0, 8]$ *bij de verschillende beginwaarden op één figuur. Zou je de beschouwde tweede-orde DV bestempelen als slecht geconditioneerd?*

2. *Beschouw de tweede-orde DV*

$$y'' = \mu - y^2 \,,$$

   waarin $\mu \geq 0$ *een willekeurige parameter voorstelt.*

   (a) *Aan welke voorwaarde moeten de coördinaten van het (de) evenwichtspunt(en) van deze DV voldoen?*

$$\begin{cases} y_e = \\ \\ y_e' = \end{cases}$$

   (b) *Plot* y_e *en* y_e' *in functie van de parameter* μ, *indien van toepassing. Wat kun je hieruit afleiden?*

   (c) *Stel* $\mu = 1$. *Bepaal de eventuele evenwichtspunten en ga de stabiliteit ervan na.*

7

Syntheseopdrachten

De syntheseopdrachten in dit practicum kunnen vrijblijvend worden opgelost om je kennis van de leerstof te toetsen, eventueel ter voorbereiding op het examen. Deze opdrachten maken gebruik van de in de voorgaande practica bestudeerde methoden.

Opdracht 7.1

Beschouw de DV

$$y' = y + \frac{t^2}{2} - t \tag{7.1}$$

over het interval $[0, 3.5]$ *met beginvoorwaarde* $y(0) = e^{-1}$.

1. *Plot op een eerste figuur de exacte oplossing samen met de Eulerbenaderingen voor* $n = 20, 60, 100$ *en* $n = 150$. *Maak eveneens een figuur die de foutencurves voor de respectievelijke n waarden bevat.*

2. *Analyseer de fout op de Eulerbenadering o.b.v. tweede afgeleide van de unieke oplossing.*

Opdracht 7.2

De snelheid van een parachutist (massa m) die springt vanuit een vliegtuig dat een eenparige rechtlijnige beweging uitvoert, kan beschreven worden a.d.h.v. de volgende eerste–orde DV

$$m\,v' = m\,g - k(t)\,v\,,$$

waarbij g de valversnelling voorstelt en $k(t)$ de weerstandscoëfficiënt gegeven door

$$k(t) = \begin{cases} 0.45\ ,\ \textit{als}\ t \leq t_o\,, \\ \quad 30\ ,\ \textit{als}\ t > t_o\,, \end{cases}$$

met $t_o = 12$ het aantal seconden die verlopen tot het openen van de parachute. Onderstel $m = 65$, $g = 9.81$.

Niet-limitatieve lijst van te behandelen punten

1. *Implementeer de functie $k(t)$ in een aparte m-file* `weerstand.m`.
2. *Kan er een exacte oplossing voor het beginwaardeprobleem gevonden worden?* **TIP***: hiervoor tracht je eerst de exacte oplossing $v_1(t)$ die geldig is over $[t_0, t_o]$ te bepalen, waarna je $\lim_{t\to t_o} v_1(t)$ gebruikt als beginwaarde voor de exacte oplossing $v_2(t)$ die geldig is over $]\,t_o, 30]$.*

   $v(t) =$

3. *Benader de oplossing a.d.h.v. de Euler-, Midpoint-, RK4- en Adams–Bashforth-methoden en een MATLAB–routine zoals* `ode45`*. Plot deze benaderingen en de exacte oplossing indien deze bestaat.*
4. *Bespreek de performantie van de verschillende numerieke technieken voor het vinden van een benaderende oplossing. Indien er geen exacte oplossing kan gevonden worden, mag men aannemen dat* `ode45` *de meest nauwkeurige is.*
5. *Bepaal het (de) evenwichtspunt(en) en ga de stabiliteit ervan na.*

Opdracht 7.3

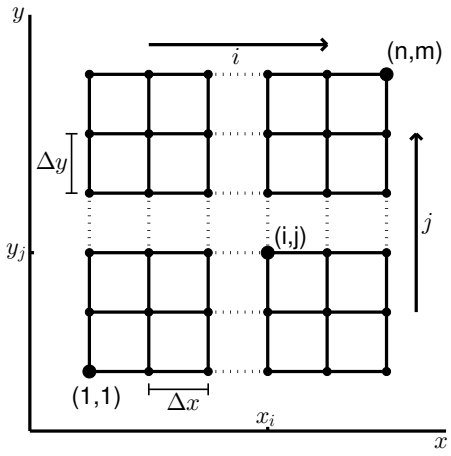
Beschouw een homogene rechthoekige plaat met breedte b en lengte l waarvan de zijden op een constante temperatuur kunnen worden gehouden m.b.v. verwarmingselementen of kunnen worden geïsoleerd. Indien we de positie in deze plaat uitdrukken in cartesische coördinaten, dan kan de temperatuursverdeling bij evenwicht in elk punt van deze plaat $U(x, y)$ worden beschreven a.d.h.v. de volgende partiële DV:

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = 0\,, \tag{7.2}$$

die voldoet aan de randvoorwaarden

$$U(0, y) = U_l(y)\,, U(b, y) = U_r(y)\,, \qquad 0 < y < \frac{1}{2}\,,$$
$$U(x, 0) = U_b(x)\,, U(x, l) = U_t(x)\,, \qquad 0 \leq x \leq 1\,,$$

Je dient de oplssing van dit randwaardenprobleem te benaderen over het gebied $[0, b] \times [0, l]$ m.b.v. de eindige differentiemethode. Hiervoor moet je gebruik maken van het eindig differentiegrid en de conventie voor indexnummering zoals weergegeven in Fig. 7.1.



Figuur 7.1: Eindig differentiegrid gebruikt ter oplossing van DV (7.2) met weergave van de conventie voor indexnummering.

1. *Duid in Fig. 7.1 aan in welke knopen er een Dirichlet of Neumann randvoorwaarde geldt. Geef tevens aan waar eventuele imaginaire knopen dienen toegevoegd te worden.*

2. *Discretiseer partiële DV (7.2), rekening houdend met de conventie voor indexnummering weergegeven in Fig. 7.1 en los de bekomen uitdrukking op naar $u_{i,j}$.*

3. *Benader de oplossing van het randwaardenprobleem indien $b = 1$, $l = \frac{1}{2}$, $\Delta x = 0.01$, $\Delta y = 0.01$, $U_l(y) = 0$, $U_r(y) = 0$, $U_b(x) = 0$, $U_t(x) = 1$ en $\phi = 10^{-5}$. Je kan hiervoor het script* `opdracht73.m` *verder aan te vullen, waarvoor je gebruik kunt maken van Algoritme 3 dat de pseudo-code weergeeft van de te volgen werkwijze.*

4. *Plot U i.f.v. x en y m.b.v. de functie* `surf`.

Algoritme 3: Pseudo-code voor het oplossen van de partiële DV (7.2).

Initialiseer een vector `x` die voor alle j de overeenkomstige x-coördinaat bevat;
Initialiseer een vector `y` die voor alle i de overeenkomstige y-coördinaat bevat;
Creëer m.b.v. de functie `zeros` een $m \times n$-matrix `U` waarin alle $u_{i,j}$ bewaard dienen te worden. Hou hierbij reeds rekening met eventuele imaginaire knopen die toegevoegd moeten worden!;
Implementeer de gepaste randvoorwaarden;
$maxFout = 1$;
while $maxFout > \phi$ **do**
    for j **do**
        for i **do**
            Bepaal $u_{i,j}^{k+1}$;
    Bereken $maxFout = \max\left(\left|u_{i,j}^{k} - u_{i,j}^{k+1}\right|\right)$;
    Overschrijf $u_{i,j}^{k}$;

Opdracht 7.4

Beschouw het volgende stelsel eerste-orde DVn:

$$\begin{cases} y_1' = \frac{1}{2}\left(1 - \frac{1}{2}y_1 - \frac{1}{2}y_2\right) y_1\,, \\ y_2' = \frac{1}{4}\left(1 - \frac{1}{3}y_1 - \frac{2}{3}y_2\right) y_2\,. \end{cases}$$

Veronderstel dat de beginwaarden van het stelsel gegeven worden door $y_1(0) = 1$ *en* $y_2(0) = 2$ *en dat we werken over het tijdsinterval* $[0, 25]$. *Stel verder* $n = 150$.

1. *Implementeer dit stelsel in MATLAB in een m-file* `stelsel74.m`.
2. *Plot het fasevlak met daarbovenop de RK4-benadering van het stelsel. Werk hierbij over het gebied* $[0, 3] \times [0, 3]$.
3. *Bepaal analytisch het (de) evenwichtspunt(en) van het stelsel DVn en ga de stabiliteit ervan na.*

   $((y_e)_1, (y_e)_2) =$

Opdracht 7.5

Beschouw het twee-dimensionaal stelsel autonome DVn gegeven door

$$\begin{cases} T_I' = 1.2\,T_M - (0.3 + a_1)\;T_I\,, & T_I(0) = 10^6\,, \\ T_M' = a_1\,T_I - d\,T_M\,, & T_M(0) = 10^6\,, \end{cases} \tag{7.3}$$

over het tijdsinterval $[0, 10]$*, waarbij* T_I *en* T_M*, het aantal tumorcellen tijdens de rustfase, resp. tijdens de mitose, voorstelt. Verder zijn* a_1 *en* d *twee controleparameters begrepen in* $[0, 1]$*.*

1. *Bepaal het evenwichtspunt van dit stelsel in* $\mathbb{R}^{+2}$*. De controleparameters worden verondersteld zo gekozen te zijn dat de coëfficiëntenmatrix niet-singulier is.*

   $((T_I)_e, (T_M)_e) =$

2. *Indien we tumorgroei definiëren als een situatie waarin* $T_I \to \infty$ *en* $T_M \to \infty$ *als* $t \to \infty$*, geef dan de minimale voorwaarde aan waaraan de eigenwaarden* λ_1 *en* λ_2 *van de coëfficiëntenmatrix van Stelsel* (7.3) *moeten voldoen, als je weet dat* $\lambda_1\,, \lambda_2 \in \mathbb{R}$ *voor alle* $a_1\,, d \in [0, 1]$*.*

   (a) $\lambda_1 > 0 \wedge \lambda_2 > 0$

   (b) $\lambda_1 > 0 \vee \lambda_2 > 0$

   (c) $\lambda_1 < 0 \wedge \lambda_2 < 0$

3. *Plot het fasevlak en bepaal de stabiliteit van het evenwichtspunt in het geval dat (i)* $a_1 = d = 0.5$*, en (ii)* $a_1 = 0.5$*,* $d = 0.9$*.*

   (i) Stabiliteit $((T_I)_e, (T_M)_e)$*:*

   (ii) Stabiliteit $((T_I)_e, (T_M)_e)$*:*

4. *Bepaal met Mathematica de unieke oplossing* $T_I(t)$ *van het beginwaardeprobleem indien* $a_1 = d = 0.5$*, implementeer deze in een m-file die je bewaart in een m-file* `TI_Exact.m`*. Dit is een vrij grote uitdrukking. Implementeer tevens Stelsel* (7.3) *in een m-file* `tumor_2D.m`*.*

5. *Gebruik* `mijnRK4Stelsel` *om de oplossing van het stelsel te benaderen (1000 stappen) en plot de benadering van* T_I *samen met de exacte oplossing* $T_I(t)$ *op één figuur.*

6. *Bepaal de maximale fout ($\epsilon_{\max}$) op de RK4-benadering. Hiervoor kun je gebruikmaken van de functie* `functieverschil`.

   $\epsilon_{\max} =$

Opdracht 7.6

De beweging van een komeet rond de zon wordt beschreven door het stelsel DVn

$$\begin{cases} x'' = -\dfrac{k\,x}{(\sqrt{x^2+y^2+z^2})^3}\,, \\ y'' = -\dfrac{k\,y}{(\sqrt{x^2+y^2+z^2})^3}\,, \\ z'' = -\dfrac{k\,z}{(\sqrt{x^2+y^2+z^2})^3}\,. \end{cases} \tag{7.4}$$

Hierbij veronderstellen we dat de zon coördinaten $(0,0,0)$ heeft. Als we de afstand in AE (Astronomische Eenheid = afstand zon-aarde) meten, en de tijd in jaren, dan is $k = 4\pi^2$ voor de beweging van een komeet rond de zon.

De komeet van Halley bereikte haar perihelium (punt het dichtst bij de zon) op 9 februari 1986 (= tijdstip $t_0 = 0$). Haar positie en snelheid op dat moment waren

$$\textit{positie} = (0.325514, -0.459460, 0.166229)\,,$$

$$\textit{snelheid} = (-9.096111, -6.916686, -1.305721)\,.$$

Om de beweging van de komeet rond de zon te kunnen plotten moeten we eerst de oplossing $(x(t), y(t), z(t))$ van (7.4) bepalen.

1. *Vorm eerst het stelsel om tot een stelsel eerste-orde DVn (zes veranderlijken: positie + snelheid).*

   $\begin{cases} \\ \\ \\ \\ \\ \\ \end{cases}$

2. *Wat zijn de beginwaarden van dit stelsel?*

   $\mathbf{y}_0 =$

3. *Implementeer in MATLAB de vectorfunctie* $\mathbf{f}$ *en bewaar deze als* `halley.m`*. Gebruik* `mijnRK4Stelsel` *om de oplossing numeriek te benaderen. Beschouw hierbij tijdsintervallen van 10 (met* $n = 500$*), 20 (met* $n = 1000$*) en 30 jaar (met* $n = 1500$*) vanaf het perihelium.*

4. *Plot de 3-dimensionale beweging (i.e. positie) rond de zon d.m.v. de instructie*

   ```
   >> plot3(X(:,1), X(:,3), X(:,5))
   ```

   Maak de plots voor de drie benaderingen in omgekeerde volgorde en plot op één figuur. De zon stel je voor a.d.h.v. een geel sterretje m.b.v. de instructie

   ```
   plot3(0, 0, 0, 'y*')
   ```

   Waarom liggen de drie benaderingen in elkaars verlengde?

5. *De komeet van Halley heeft in werkelijkheid een omlooptijd van ongeveer 76 jaar. Verhoog de parameter n stelselmatig (1500, 2000, 2500, . . .) tot je over een tijdsinterval van 75 jaar aanvaardbare resultaten bekomt. Wat is de corresponderende waarde van n?*

   $n =$

   Waarom zijn de voorgaande plots niet aanvaardbaar?

6. *Voer de benadering opnieuw uit, maar nu over het interval* $[0, 77]$ *en maak hiervoor gebruik van de MATLAB-functie* `ode45`*. Bepaal de maximale afstand van de komeet tot de zon* r_{max} *tot op 1 honderste AE en ook het tijdstip waarop dit gebeurt. De komeet bevindt zich dan in haar aphelium. Plot hiertoe de afstand* $r = \sqrt{x^2 + y^2 + z^2}$ *tot de zon i.f.v. de tijd.*

   $r_{max} =$

BIJLAGEN

Bijlage A

Mathematica-instructies

Vermits het bij de studie van benaderingsmethodes voor DVn vaak wenselijk is om de unieke oplossing van de beschouwde beginwaardeproblemen te kennen, zodat de performantie van de geÃ¯mplementeerde methodes kan worden nagegaan, geeft Tabel A.1 een overzicht van enkele nuttige MATHEMATICA instructies voor de studie van DVn. Hierbij werd volgende notatie gebruikt:

- `bvn`: begin- of randvoorwaarden door ',' gescheiden
- `dvn`: DV door ',' gescheiden
- `vars`: onbekenden
- `vgl`: algebraÃ¯sche vergelijkingen, gescheiden door ','

Tabel A.1: Enkele Mathematica-instructies die nuttig zijn bij de studie van DVn.

Bewerking	**Commando**
Symbolisch oplossen (stelsel) DVn	`DSolve[{dv1, dv2, ..., dvn, bvn}, {y1[x], y2[x],..., yn[x]}, x)`
$f^{(n)}(x)$	`D[f, {x, n}]`
Symbolisch oplossen (stelsel) vergelijking(en)	`Solve[{vgln}, {vars}]`
Numeriek oplossen (stelsel) vergelijking(en)	`NSolve[{vgln}, {vars}]`
Onbepaalde integraal $\int f(x)$	`Integrate[f, x]`
Bepaalde integraal $\int_a^b f(x)$	`Integrate[f, {x, a, b}]`
$\lim_{x\to c} f(x)$	`Limit[f, x -> c]`
$\lim_{x\to\infty} f(x)$	`Limit[f, x -> Infinity]`
$\ln f$	`Log[f]`
$\log f$	`Log10[f]`
2D-plot van $f(x)$ over het x-interval $[d, e]$	`Plot[f, {x, d, e}]`
Vereenvoudigen	`Simplify[expr]`
Splitsen in partieelbreuken	`Apart[expr]`
Declaratie matrix `A`	`{{A11, A12, ..., A1n}, {A21, A22, ..., A2n}, ..., {Am1, Am2, ..., Amn}}`
Eigenwaarden van een matrix A	`Eigenvalues[A]`
Eigenvectoren van een matrix A	`Eigenvectors[A]`

Bijlage B

Wolfram|Alpha

Wolfram|Alpha (`http://www.wolframalpha.com/`) werd in 2009 gelanceerd door Wolfram Inc. en fungeert als een intelligente zoekmachine die in staat is om wiskundige input correct te interpreteren, alsook wiskundige bewerkingen uit te voeren en vergelijkingen op te lossen. Een typische interface van deze intelligente zoekmachine wordt weergegeven in Fig. B.1.

Een DV in de onbekende functie $y = y(x)$ kan met Wolfram|Alpha opgelost worden door deze op een intuïtieve manier in de zoekbalk in te geven. Zo kan de algemene oplossing van de DV

$$y^2\,y' - x\,y = 0 \tag{B.1}$$

bepaald worden door

```
y[x]^2*D[y[x], {x,1}] - x*y[x] = 0
```

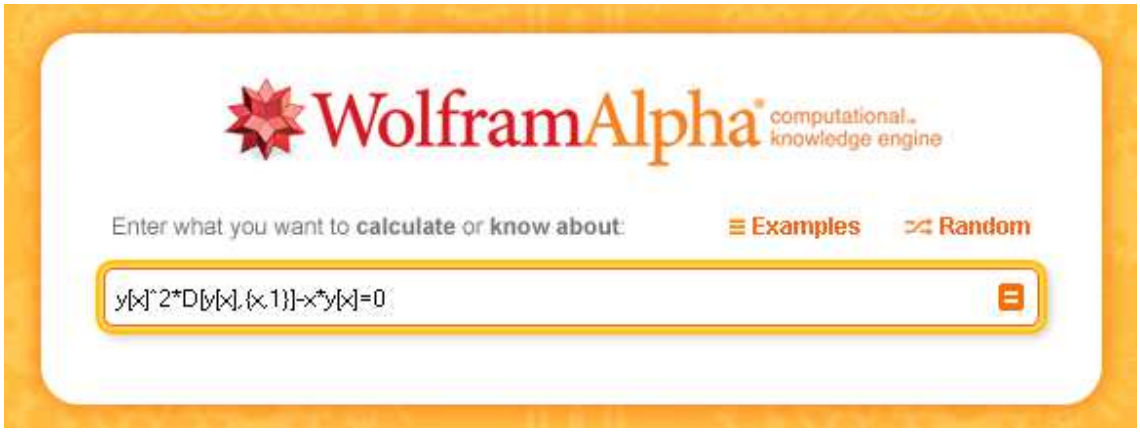
in te geven in de zoekbalk (Fig B.2). Hierbij stelt `y[x]` de te zoeken functie $y(x)$ voor, en is `D[y[x],{x,n}]` niets anders dan de differentiaaloperator $y^{(n)}(x)$. Na het indrukken van de enter-toets start de berekening en verschijnen de resultaten langzamerhand op het scherm. Zo wordt voor de beschouwde DV niet enkel de algemene oplossing gegeven, maar ook het type DV, enkele alternatieve schrijfwijzen van de input, alsook plots bij enkele beginwaarden $y(t_0) = y_0$ (Fig. B.3). Voor zover mogelijk zal Wolfram|Alpha steeds een expliciete oplossing leveren.

Willen we de unieke oplossing van de DV (B.1), indien $y(0) = 0$, dan gebruiken we de volgende instructie in de zoekbalk van Wolfram|Alpha:

```
y[x]^2*D[y[x], {x,1}] - x*y[x] = 0, y[0] = 0
```

Algemeen kunnen de afgeleiden die in de beginwaarden voorkomen, in Wolfram|Alpha uitgedrukt worden als `y'[t0] = y'0`, `y''[t0] = y''0` voor $y'(t_0) = y'_0$ en $y''(t_0) = y''_0$, respectievelijk, en analoog voor hogere orde afgeleiden.

Figuur B.1: Interface van Wolfram|Alpha.



Figuur B.2: Nodige input in de zoekbalk van Wolfram|Alpha voor het bepalen van de algemene oplossing van de DV $y^2\,y' - x\,y = 0$.

WolframAlpha™ computational knowledge engine

y[x]^2*D[y[x],{x,1}]-x*y[x]=0

Input :

$$y(x)^2 \frac{\partial y(x)}{\partial x^1} - x\, y(x) = 0$$

ODE classification :

first-order nonlinear ordinary differential equation

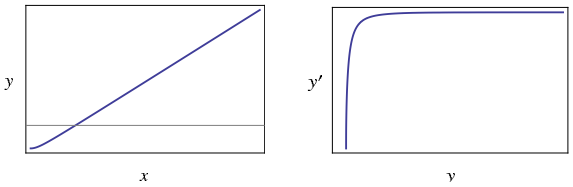
Alternate forms :

$$y(x)^2\, y'(x) = x\, y(x)$$

Differential equation solutions :

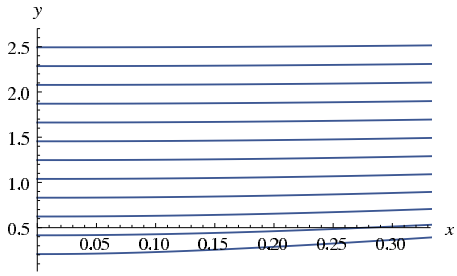
$$y(x) = 0$$

Plots of sample individual solution :



$y(0) = 1$

Sample solution family :



(sampling $y(0)$)

Generated by Wolfram|Alpha (www.wolframalpha.com) on September 4, 2011 from Champaign, IL.
© Wolfram Alpha LLC—*A Wolfram Research Company*

1

Figuur B.3: Output gegenereerd door Wolfram|Alpha bij het zoeken naar de algemene oplossing van de DV $y^2\, y' - x\, y = 0$.

Bibliografie

Atkinson, K. E., Han, W., Stewart, D., 2009. Numerical Solution of Ordinary Differential Equations. Pure and Applied Mathematics. John Wiley & Sons, Chichester, United Kingdom.

Butcher, J. C., 2008. Numerical Methods for Ordinary Differential Equations. John Wiley & Sons, Chichester, United Kingdom.

Coleman, M. P., 2005. An Introduction to Partial Differential Equations with MATLAB. Chapman & Hall/CRC, Boca Raton, United States.

Courant, F., Friedrichs, K., Lewy, H., 1928. Über die partiellen differenzengleichungen der mathematischen physik. Mathematische Annalen 100, 32–74.

Gockenbach, M. S., 2002. Partial Differential Equations. Analytical and Numerical Methods. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, United States.

Greenberg, M. D., 1998. Advanced Engineering Mathematics, 2nd Edition. Prentice Hall, Upper Saddle River, United States.

Jones, D. S., Sleeman, B. D., 2003. Differential Equations and Mathematical Biology. Chapman & Hall/CRC, Boca Raton, United States.

King, A. C., Billingham, J., Otto, S. R., 2003. Differential Equations. Cambridge University Press, Cambridge, United Kingdom.

Kohler, W., Johnson, L., 2005. Elementary Differential Equations with Boundary-value Problems. Pearson Education, Inc., Upper Saddle River, United States.

Lee, H. J., Schiesser, W. E., 2004. Ordinary and Partial Differential Equations Routines in C, C++, Fortran, Maple, and MATLAB. Chapman & Hall/CRC, Boca Raton, United States.

Li, J., Chen, Y., 2008. Computational Partial Differential Equations using MATLAB. Chapman & Hall/CRC, Boca Raton, United States.

Liu, G. R., Quek, S., 2003. The Finite Element Method: a Practical Course. Butterworth-Heinemann, Oxford, United Kingdom.

Murray, J. D., 1993. Mathematical Biology: I. An Introduction, 2nd Edition. Springer, Berlin, Germany.

Murray, J. D., 2007. Mathematical Biology: II. Spatial Models and Biomedical Applications, 3rd Edition. Springer, Berlin, Germany.

Ott, E., 1993. Chaos in Dynamical Systems. Cambridge University Press, Cambridge, United Kingdom.

Persson, P. O., Strang, G., 2004. A simple mesh generator in MATLAB. SIAM Review 46, 329–345.

Polking, J., Bogess, A., Arnold, D., 2005. Differential Equations with Boundary-value Problems. Pearson Education, Inc., Upper Saddle River, United States.

Rao, S. S., 2010. The Finite Element Method in Engineering. Butterworth-Heinemann, Oxford, United Kingdom.

Shampine, L. F., Gladwell, I., Thompson, S., 2003. Solving ODEs with MATLAB. Cambridge University Press, Cambridge, United Kingdom.

Strogatz, S. H., 1994. Nonlinear Dynamics and Chaos: with Applications to Physics, Biology, Chemistry, and Engineering. Perseus Books Publishing, Cambridge, United Kingdom.

Zill, D. G., Cullen, M. R., 2009. Differential Equations with Boundary-value Problems. Brooks/Cole, Belmont, United States.