

Differentiaalvergelijkingen

Arthur van Dam

9 april 1998

Inhoudsopgave

1	Inleiding	3
2	DV'en oplossen; enkele methoden	4
2.1	Euler-Forward	4
2.2	De Midpuntmethode	5
3	Practicumgedeelte	7
3.1	Euler-Forward	7
3.1.1	JAVA-implementatie	7
3.1.2	Experimentele resultaten	7
3.2	Midpunt	8
3.2.1	JAVA-implementatie	8
3.2.2	Experimentele resultaten	9
3.3	Stabiliteit	9
3.3.1	Euler-Forward	10
3.3.2	Midpunt	11
3.4	Convergentie	12
3.4.1	Euler-Forward	12
4	De mathematische Slinger	13
4.1	Introductie	13
4.2	JAVA-implementatie en experimenten	14
5	Partiële differentiaalvergelijkingen en stelsels daarvan	18
5.1	De warmtevergelijking	18
5.1.1	Fysische achtergrond	18
5.1.2	Randvoorwaarden	19
5.1.3	Veralgemenisering	19
5.2	De methode der lijnen	20
5.3	JAVA-implementatie	20
5.3.1	Experiment 1; maximale fout	20
5.4	Theorie vs. praktijk	21
5.5	Experiment 2; Variaties op de WV	23

A Listings	28
A.1 Euler-Forward	28
A.2 Midpunt	28
A.3 Mathematische Slinger	29
A.4 Warmte Vergelijking	30

Hoofdstuk 1

Inleiding

In de wetenschap worden veel verschillende processen bestudeerd. Vaak spelen differentiaalvergelijkingen hierin een rol. Hierbij is te denken aan reacties of bewegingen. In het vervolg van dit verslag zullen differentiaalvergelijkingen met DV'en aangeduid worden.

Een DV bevat verschillende orde afgeleiden van een functie. De hoogste orde afgeleide in een DV, wordt de orde van de DV genoemd. Een enkele DV is soms wel op te lossen, maar in grotere processen ontstaan al gauw stelsels van meerdere DV'en. Toch is het bij ongekoppelde DV'en ook vaak al onmogelijk om de oplossing analytisch te vinden. In hoofdstuk 2 zullen methoden worden behandeld, die zo'n ongekoppelde DV oplossen. In de daaropvolgende hoofdstukken, zullen deze methoden worden toegepast in stelsels van meerdere DV'en en DV'en van tweede orde.

Hoofdstuk 2

DV'en oplossen; enkele methoden

2.1 Euler-Forward

Laten we de volgende eerste orde DV beschouwen:

$$\dot{u} \equiv u'(t) = f(t, u(t)), \quad (2.1)$$

waarin $u(t=0) = u_0$. Volgens de definitie van de afgeleide van $u(t)$ geldt:

$$u'(t) = \frac{u(t + \Delta t) - u(t)}{\Delta t}. \quad (2.2)$$

Omdat we de oplossing niet analytisch kunnen bepalen, gaan we de oplossing discreet benaderen. Deze techniek wordt beschreven door *Euler-Forward* (EF). EF geeft een benadering door het domein te discretiseren en dus duiden we het tijdstip aan met $t_n = n \cdot \Delta t$ en de benaderde waarde van u op datzelfde tijdstip met u_n . De exacte waarde van u op t_n wordt aangeduid met $u(t_n)$.

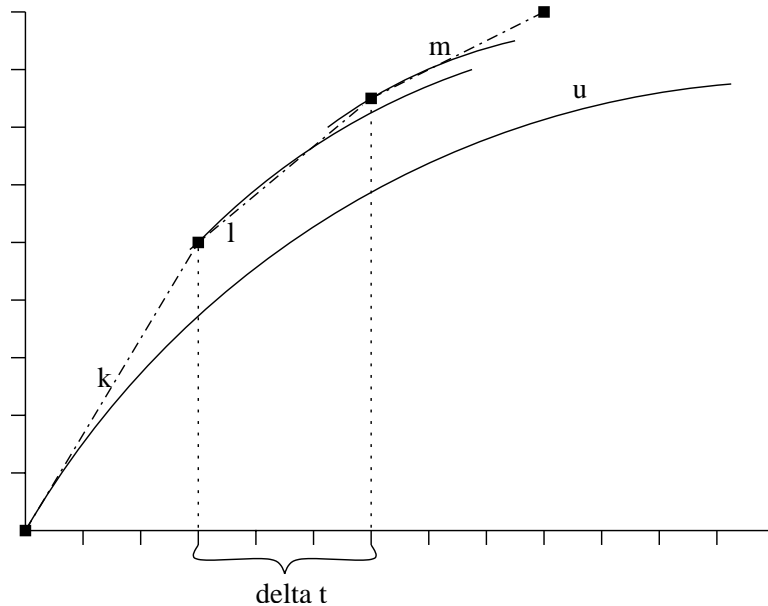
In vergelijking 2.2 wordt Δt klein verondersteld en dus is voor $u'(t)$ een Taylorbenadering te geven:

$$u'(t) \approx u'(t_n) + \Delta t u''(t_n). \quad (2.3)$$

Als we vergelijkingen 2.1, 2.2 en 2.3 combineren, levert dit de volgende benadering op:

$$u_{n+1} = u_n + \Delta t f(t_n, u_n) \text{ (voor } n = 0, 1, 2, \dots \text{)}. \quad (2.4)$$

Als we de EF-methode meetkundig in beeld brengen, levert dit een duidelijk inzicht in de werking ervan. In figuur 2.1 is een functie u te zien. Deze functie is de oplossing van de DV en is dus normaal gesproken niet bekend. Wel is de hellingfunctie bekend (formule 2.1) en er is een randvoorwaarde gegeven. In het gegeven randpunt, wordt de helling bepaald (raaklijn k) en op het volgende tijdstip (Δt) wordt de functiewaarde van k bepaald. Vervolgens wordt in dat punt de helling bepaald, waarna het volgende punt bij $t = 2 \cdot \Delta t$ op die lijn wordt bepaald. Dit punt kan weer in de DV ingevuld worden om de helling te krijgen etcetera. . . . Eigenlijk moet op ieder tijdstip t_i het punt $u(t)$ ingevuld worden, terwijl bij EF het benaderde punt wordt – op een hoger of lager niveau van u ingevuld. Dit kan ook niet anders, omdat het verband tussen u en t niet expliciet gegeven is. Bij kleine Δt zal het verschil tussen de hellingen in het benaderde punt en de echte helling echter niet zo groot zijn. Dit is in figuur 2.1 te zien aan de verschillende *niveau's* van u , die vrijwel parallel lopen.



Figuur 2.1: Meetkundige voorstelling van EF

Als de Δt daarentegen te groot genomen wordt, kan het gebeuren dat de benadering op een niveaulijn van u terecht komt, die niet meer op u lijkt, waardoor een onjuiste helling wordt berekend, zodat de benadering van het volgende punt ook volledig foutloopt.

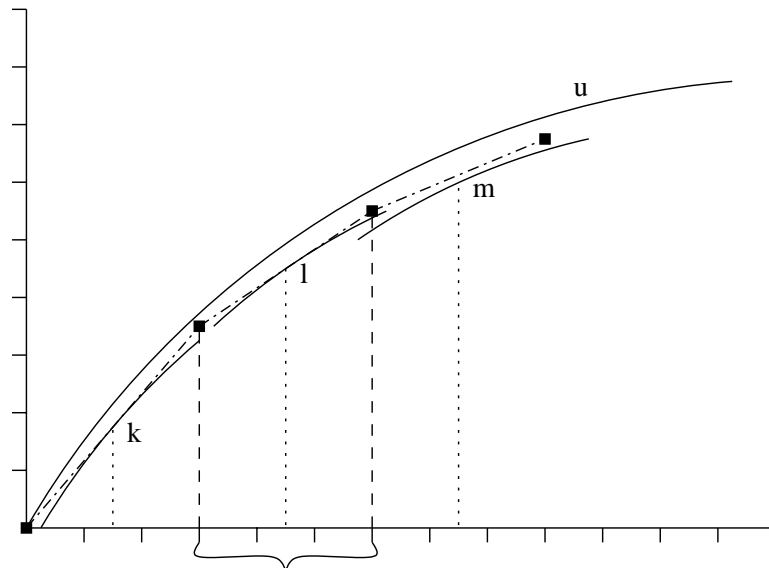
Bijzonderheden van de EF worden in sectie 3.1.2 besproken aan de hand van experimentele resultaten.

2.2 De Midpuntmethode

Zoals in sectie 3.1.2 zal blijken, vraagt EF toch nog wel om enige verbetering van de nauwkeurigheid. Een tweede methode, die inderdaad nauwkeuriger is dan EF, is de *Midpuntmethode* (MP). Om u_{n+1} te berekenen, maakt EF gebruik van u_n en de afgeleide in dat punt. MP doet dit nauwkeuriger door de afgeleide precies *tussen* u_n en u_{n+1} te nemen. In formule 2.1 worden dus $t_{n+1/2}$ en $u_{n+1/2}$ ingevuld. De waarde van $u_{n+1/2}$ wordt met EF bepaald, door te kijken naar het domein $[t_n, t_{n+1/2}]$ ($\frac{1}{2}\Delta t$). In formulevorm:

$$u_{n+1} = u_n + \Delta t f(t_{n+1/2}, u_{n+1/2}) \text{ (voor } n = 0, 1, 2, \dots \text{)}. \quad (2.5)$$

De meetkundige voorstelling hiervan staat in figuur 2.2. In figuur 2.2 is te zien dat de lijnen k , l en n raaklijnen zijn van verschillende niveaulijnen van u . Deze liggen echter al beduidend dichter bij elkaar dan bij de benadering door EF (figuur 2.1). Hierdoor is er al minder kans dat de benadering op een sterk afwijkende niveaulijn van u terecht komt. De functie u is gelijk aan die in figuur 2.1 en MP geeft dus – zelfs bij de redelijk grote Δt in de figuur – een betere benadering dan EF. Ook van MP worden de bijzonderheden in sectie 3.2.2 besproken aan de hand van experimentele resultaten.



Figuur 2.2: Meetkundige voorstelling van MP

Hoofdstuk 3

Practicumgedeelte

Om het gedrag van EF en MP te onderzoeken, gebruiken we een *JAVA*-programma dat deze methoden uitvoert en waarin we de waarde van Δt kunnen veranderen. Beide methoden worden natuurlijk op dezelfde DV toegepast. Hiervoor wordt de volgende DV genomen:

$$\begin{aligned}\dot{u} &= -\lambda(u - \sin(\pi t)) + \pi \cos(\pi t), \\ u(0) &= 0.\end{aligned}\tag{3.1}$$

3.1 Euler-Forward

3.1.1 *JAVA*-implementatie

Allereerst even wat uitleg over het gebruikte programma. De listing van `Euler.java` is opgenomen in sectie A.1. Er worden vier parameters meegegeven vanaf de command-prompt, voor zoveel mogelijk flexibiliteit. Vervolgens wordt in een lus in de methode `runEuler` de tijd doorlopen met stappen van Δt . In deze lus wordt in feite formule 2.4 berekend. Hiervoor is telkens de waarde van $u'(t, u(t))$ nodig, die gegeven wordt door formule 3.1. Het berekenen hiervan gebeurt in de methode `compF`. Verdere details over de werking staan in de vorm van commentaar in de listing zelf.

3.1.2 Experimentele resultaten

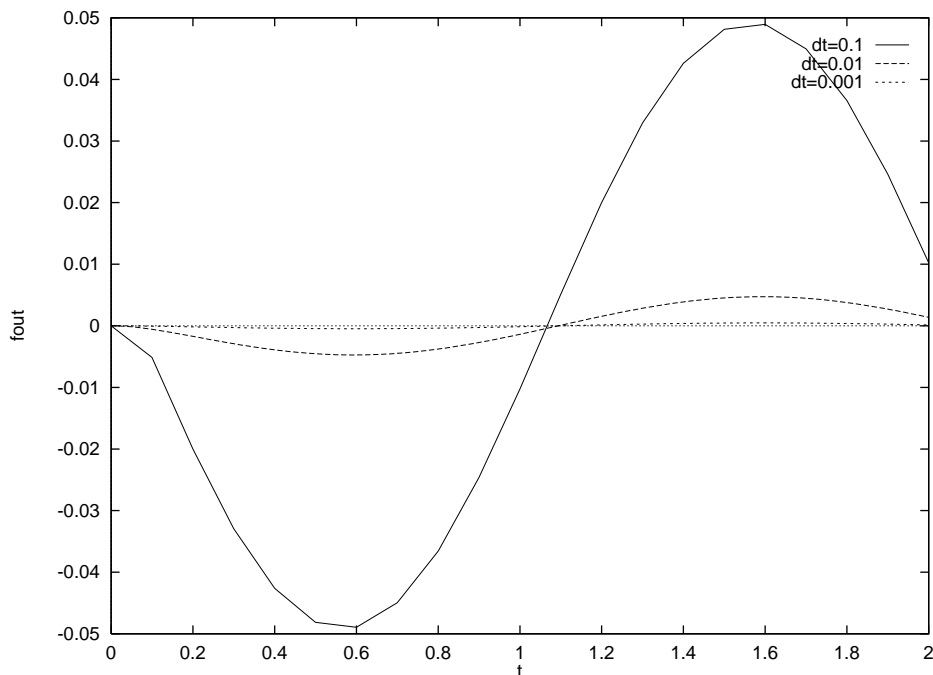
Zoals eerder gezegd hebben we voor verschillende waarden van Δt EF uitgevoerd op formule 3.1. Om de effecten van eventuele onnauwkeurigheden beter te laten uitkomen, variëren we ook de waarde van λ . De resultaten voor u op $t = 2$ zijn opgenomen in tabel 3.1.

Δt	λ	1	10	10^2	10^3
1		-3.1416	-31.416	$-3.1416 \cdot 10^2$	$-3.1416 \cdot 10^3$
10^{-1}		$-1.331 \cdot 10^{-1}$	$-1.0233 \cdot 10^{-2}$	$-4.4213 \cdot 10^{15}$	$-4.0841 \cdot 10^{35}$
10^{-2}		$-1.2423 \cdot 10^{-2}$	$-1.3762 \cdot 10^{-3}$	$-1.0334 \cdot 10^{-5}$	$-2.5522 \cdot 10^{184}$
10^{-3}		$-1.2342 \cdot 10^{-3}$	$-1.4076 \cdot 10^{-4}$	$-1.4973 \cdot 10^{-6}$	$-1.0335 \cdot 10^{-8}$
10^{-4}		$-1.2333 \cdot 10^{-4}$	$-1.4107 \cdot 10^{-5}$	$-1.5436 \cdot 10^{-7}$	$-1.4986 \cdot 10^{-9}$

Tabel 3.1: Benadering van $u(2)$ uit formule 3.1 met EF

De oplossing van formule 3.1 is nog analytisch te bepalen en is: $u(t) = \sin(\pi t)$. Aangezien $u(2) = \sin(2\pi) = 0$, staan in tabel 3.1 ook meteen de nauwkeurigheden van EF voor verschillende Δt . Wanneer λ groot wordt genomen en Δt niet al te klein is, komt de benadering, zoals gezegd, op een sterk afwijkend niveau van u terecht. De benaderde waarde is dan – zoals te zien in de tabel – helemaal fout. Wanneer echter wat kleinere Δt gebruikt wordt, is de afwijking een stuk kleiner. Duidelijk is dan ook te zien, dat wanneer λ een factor 10 wordt verkleind, de afwijking met dezelfde factor afneemt. EF wordt hierom ook wel een *eerste-orde benadering* genoemd.

De fout als functie van t doet misschien een beetje denken aan een sinus (zie figuur 3.1). Vermoedelijk heeft de Taylor-benadering hier iets mee te maken; in formule 2.3 wordt een eerste orde Taylor-benadering gedaan. De tweede afgeleide – die dus niet meer hierin is opgenomen – is de voornaamste veroorzaker van de fout. De tweede afgeleide van een sinus, levert ook weer een (negatieve) sinus. Toch is de fout lang niet een zuivere sinus, aangezien er voor een zuivere vergelijking oneindig veel hogere orde termen in de Taylor-benadering voorkomen. Deze zijn verwaarloosd in de benadering (formule 2.3) en dragen dus ook bij tot de afwijking.



Figuur 3.1: De afwijking bij EF als functie van t , $\lambda = 10$

3.2 Midpunt

3.2.1 JAVA-implementatie

Ook van `MP.java` is de listing opgenomen in sectie A.2. Het programma is in grote lijnen gelijk aan `Euler.java`. Het enige verschil zit in de methode `runMP`. Hierin wordt eerst het punt op $t_{j+1/2}$ bepaald met EF. Hierna kan dit punt, samen met $t_{j+1/2}$ meegegeven worden

aan `compF`, zodat de helling op $t_{j+1/2}$ wordt berekend. Dit levert, ingevuld in formule 2.5 het volgende punt op. Natuurlijk wordt het domein hier ook in een lus doorlopen.

3.2.2 Experimentele resultaten

We hebben MP op dezelfde DV toegepast als bij EF (formule 3.1). De resultaten zijn opgenomen in tabel 3.2. Het verband tussen de onnauwkeurigheid en Δt blijkt kwadratisch te

Δt	λ	1	10	10^2	10^3
1		$2.8540 \cdot 10^{-1}$	$-2.2832 \cdot 10^2$	$-2.7969 \cdot 10^5$	$-2.8483 \cdot 10^8$
10^{-1}		$4.4137 \cdot 10^{-3}$	$1.3884 \cdot 10^{-2}$	$-2.7570 \cdot 10^{28}$	$-8.2610 \cdot 10^{68}$
10^{-2}		$4.1417 \cdot 10^{-5}$	$5.0828 \cdot 10^{-5}$	$1.6760 \cdot 10^{-5}$	NaN
10^{-3}		$4.1139 \cdot 10^{-7}$	$4.7390 \cdot 10^{-7}$	$5.5809 \cdot 10^{-8}$	$1.6795 \cdot 10^{-8}$
10^{-4}		$4.1112 \cdot 10^{-9}$	$4.7070 \cdot 10^{-9}$	$5.2016 \cdot 10^{-10}$	$5.5864 \cdot 10^{-11}$

Tabel 3.2: Benadering van $u(2)$ uit formule 3.1 met MP

zijn; MP wordt ook wel een *tweede-orde benadering* genoemd. Evenals bij EF, veroorzaakt de Taylor-benadering de annauwkeurigheid. Aangezien bij MP ook een eerste-orde Taylor benadering is gebruikt, levert de afwijking als functie van t ook een sinusöide op (figuur 3.1). De vorm zal dus vergelijkbaar zijn met die van EF. De amplitude zal echter een stuk kleiner zijn, aangezien MP nauwkeuriger is. Een goede vergelijking tussen de afwijkingen bij verschillende Δt 's is grafisch dus ook niet goed mogelijk, omdat de amplitude van twee opeenvolgende Δt 's al een factor 100 verschilt.

Interessanter is het nu om de voorspelling eens te vergelijken met de werkelijke oplossing. In figuur 3.2 is voor $\lambda = 1$ de benadering voor verschillende Δt geplot. Het is duidelijk dat $\Delta t = 1$ te groot is om redelijke resultaten te geven. Voor kleinere waarden zijn de voorspellingen echter meteen al behoorlijk nauwkeurig. Hierbij moet wel opgemerkt worden dat λ gunstig gekozen is. Wanneer $\lambda = 100$ wordt gekozen, is in de tabel te zien dat voor $\Delta t = 1$ of $0,1$ de benadering 'opblaast'.

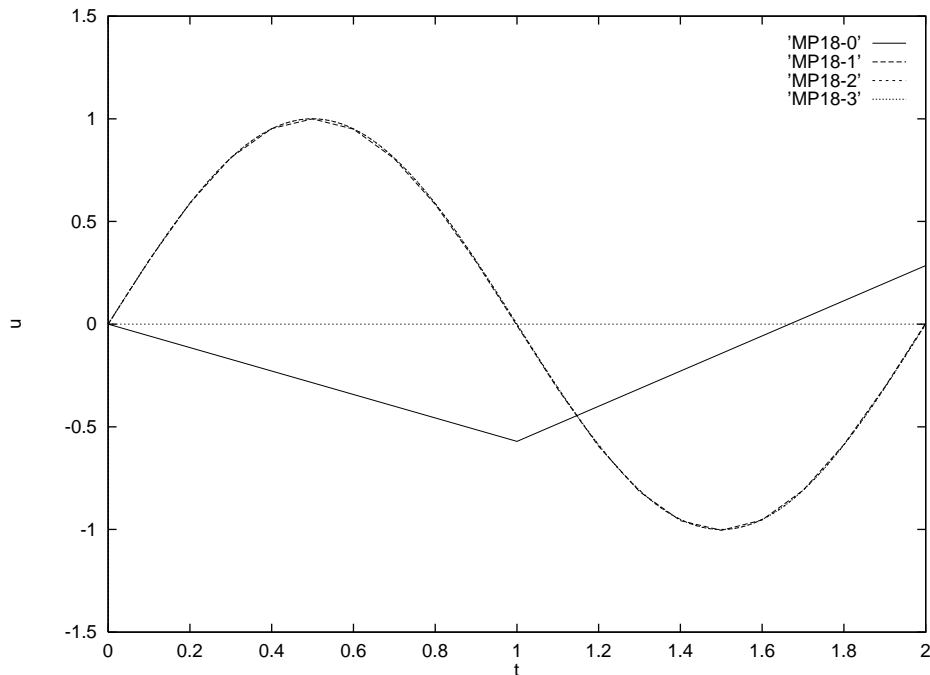
3.3 Stabiliteit

In de vorige secties bleek dat voor bepaalde combinaties van λ en Δt de benadering faalt. We willen dus bepalen welke combinaties goede resultaten opleveren. We lossen dit op voor de algemene DV:

$$\begin{aligned} u'(t) &= \alpha u(t); \alpha \in \mathbb{C} \\ u(0) &= u_0^* \end{aligned} \tag{3.2}$$

Afhankelijk van het gebruikte algoritme, kan een verband tussen u_n en u_0 worden beschreven. Dit verband uit zich in een versterkingsfactor α , zodat $u_n = \alpha^n \cdot u_0$. Om nu de onnauwkeurigheid te beschrijven, nemen we een functie $v(t) = u(t)$, met $v(0) = v_0 + \varepsilon_0$. Dit is dus dezelfde functie met een kleine verstoring aan het begin. Voor het verschil geldt nu:

$$v_n - u_n \equiv \varepsilon_n \text{ en voor stabiliteit geldt: } \lim_{n \rightarrow \infty} |\varepsilon_n| < \infty$$



Figuur 3.2: De benaderde oplossing van formule 3.1 door MP

De afwijking is dus wel begrensd. Om echter *goede* benaderingen te geven stellen we de volgende een eis, zodat er een *stabiliteitsgebied* ontstaat. Dit wordt als volgt beschreven:

$$S = \{z \in \mathbb{C} \mid |\nu(z)| \leq 1\}, \text{ met } z = \alpha\Delta t = x + iy, \quad (3.3)$$

waarin $\nu(\alpha, \Delta t)$ de versterkingsfactor is, die van het gebruikte algoritme afhangt. De aanvankelijk kleine verstoring leidt nu dus tot een onnauwkeurigheid van $\varepsilon_n = \nu(z) \cdot \varepsilon_0$.

3.3.1 Euler-Forward

Als we formule 2.4 en 3.2 combineren, levert dit voor EF het volgende op:

$$u_{n+1} = u_n + u_n z, \text{ zodat } \nu(z) = 1 + z \quad (3.4)$$

Voor het stabiliteitsgebied moet z dus voldoen aan formule 3.3. We zien z als een vector (x, y) , waarin y het imaginaire deel aangeeft. Uitwerken levert:

$$|\nu(z)| \leq 1 \rightarrow \|\vec{z}\| = \sqrt{(1+x)^2 + y^2} \leq 1$$

Het stabiliteitsgebied van EF is dus een cirkelschijf met straal 1 en middelpunt $(-1, 0)$.

3.3.2 Midpunt

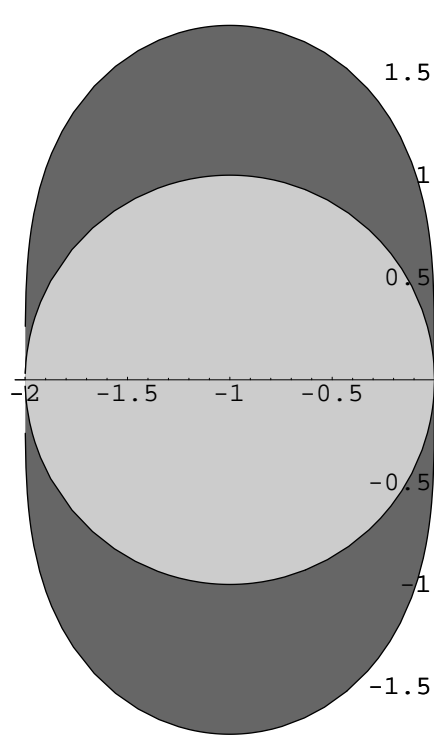
Wat we voor EF gedaan hebben, herhalen we voor MP. Bij de gebruikte elgemene DV in wordt MP beschreven door:

$$\begin{aligned} u_{n+1} &= u_n + \Delta t \alpha \cdot \left(u_n + \frac{1}{2} \Delta t \alpha u_n \right) \\ &= \left(1 + \alpha \Delta t + \frac{1}{2} (\alpha \Delta t)^2 \right) \cdot u_n \\ &= \left(1 + z + \frac{1}{2} z^2 \right) u_n, \\ \text{zodat } \nu(z) &= 1 + z + \frac{1}{2} z^2 \end{aligned}$$

Als we \vec{z} weer in x en y uitschrijven, levert dit op:

$$(1 + (1 + x)^2 - y^2)^2 + 4y^2(1 + x)^2 \leq 4 \quad (3.5)$$

Natuurlijk kunnen we y uit formule 3.5 volledig omschrijven in termen van x . Met *Mathe-*



Figuur 3.3: Stabiliteitsgebieden van EF en MP

matica kunnen we het gebied S voor MP echter al plotten. In figuur 3.3 staan voor EF en MP de stabiliteitsgebieden. De cirkel hoort bij EF en de ovaal bij MP.

3.4 Convergentie

Tot slot vragen we ons nog af, of in het beste geval ($\Delta t \rightarrow 0$) de oplossing ook exact benaderd wordt. De oplossing voor de DV uit formule 3.2 was $u = u_0^* e^{\alpha t}$. Laten we nu een tijdsinterval van T nemen, opgesplitst in n stukken. Op die manier hangt Δt van n af ($\Delta t = \frac{T}{n}$).

3.4.1 Euler-Forward

Bij EF geldt voor bovengenoemde DV:

$$u_{n+1} = u_n(1 + \Delta t \alpha)$$

en dus geldt:

$$u_n = u_0 \left(1 + \frac{T}{n} \alpha\right)^n$$

Voor $\Delta t \rightarrow 0$ moet n naar oneindig gaan. We vragen ons af of dan het volgende geldt:

$$\lim_{n \rightarrow \infty} u_n \stackrel{?}{=} u_0^* e^{\alpha T}$$

Als we het linkerlid uitwerken, blijkt het volgende:

$$\lim_{n \rightarrow \infty} u_0 \left(1 + \alpha \frac{T}{n}\right)^n = u_0 \lim_{n \rightarrow \infty} e^{n \log(1 + \alpha \frac{T}{n})}$$

We berekenen eerst de exponent:

$$\begin{aligned} \lim_{n \rightarrow \infty} n \log\left(1 + \alpha \frac{T}{n}\right) &= \lim_{n \downarrow 0} \frac{\log(1 + \alpha T n)}{n} \\ &= \lim_{n \downarrow 0} \frac{\frac{\alpha T}{1 + \alpha T n}}{1} = \alpha T \end{aligned}$$

als we dit voor de exponent invullen, geldt dus inderdaad dat $\lim_{n \rightarrow \infty} u_n = u_0^* e^{\alpha T}$ en EF dus convergent is.

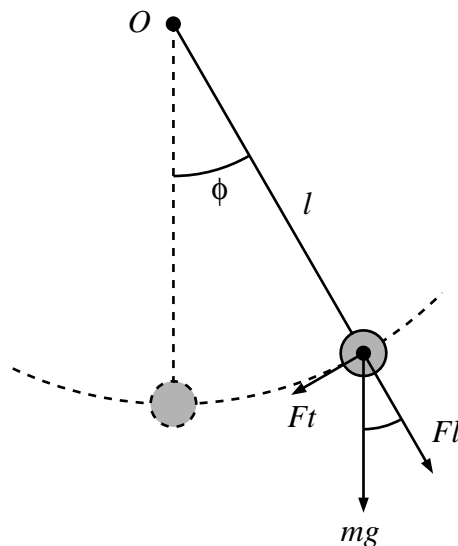
Hoofdstuk 4

De mathematische Slinger

In de vorige hoofdstukken was er telkens één DV die opgelost moest worden. In de praktijk wordt er echter veel met stelsels van DV'en gewerkt. Hogere orde DV'en zijn met EF en MP ook niet direct te bepalen – de hellingfunctie is immers onbekend – maar herschrijven biedt hier uitkomst. Wanneer nieuwe variabelen worden ingevoerd, ontstaan meerdere nieuwe DV'en van de eerste orde, zodat EF en MP wel toegepast kunnen worden. In dit hoofdstuk zal de mathematische slinger besproken worden, een stelsel van twee DV'en.

4.1 Introductie

De mathematische slinger (zie figuur 4.1) bestaat uit een starre, massaloze staaf, die – aanvankelijk – wrijvingsloos om een scharnierpunt O kan draaien. Aan het ander uiteinde is een massa m bevestigd.



Figuur 4.1: De mathematische slinger

De beweging van m wordt beschreven door:

$$ml\ddot{\phi} + mg \sin \phi = 0 \quad (4.1)$$

We kiezen nu twee nieuwe variabelen ϕ_1 en ϕ_2 , waarvoor geldt:

$$\phi_1 = \phi \quad \phi_2 = \dot{\phi}$$

Hierdoor kunnen we een stelsel van twee DV'en opstellen:

$$\dot{\phi}_1 = \phi_2 \quad \& \quad \dot{\phi}_2 = -\frac{g}{l} \sin \phi_1 \quad (4.2)$$

Voor kleine ϕ_1 geldt: $\sin(\phi_1) \approx \phi_1$, zodat het stelsel DV'en in formule 4.2 een stuk eenvoudiger wordt. Dit nieuwe stelsel is nog redelijk eenvoudig uit te werken en dit levert uiteindelijk voor de oplossing:

$$\phi_1 = \sin\left(\sqrt{\frac{g}{l}} t\right) \quad \phi_2 = \sqrt{\frac{g}{l}} \cos\left(\sqrt{\frac{g}{l}} t\right) \quad (4.3)$$

Voor de periode van de slingerbeweging geldt: $T = 2\pi\sqrt{\frac{l}{g}}$ en stel dat we nemen: $l = 1.0[\text{m}]$ en $g = 9.81[\text{m}]/[\text{s}]^2$, dan is de trillingstijd bij asymptotische benadering gelijk aan 2.00607. Om dit stelsel toch nauwkeuriger te benaderen, hebben we een programma geschreven dat de sinus-term *niet* asymptotisch benadert.

4.2 JAVA-implementatie en experimenten

Bovenstaande methode hebben we in een *JAVA*-programma gezet (zie sectie A.3). Het verkregen stelsel uit formule 4.2 wordt met EF opgelost. Er zijn nu twee methoden, waarin een afgeleide wordt bepaald (`comp1` en `comp2`). Verder worden `phi1_old` en `phi2_old` pas aan het eind van de lus veranderd, omdat voor berekening van ϕ_2 de vorige waarde van ϕ_1 nog nodig is. Het programma print het tijdstip en ϕ_1 en ϕ_2 op het scherm. Zodoende kunnen faseplaatjes worden gemaakt.

In methode `comp2` wordt òf de sinus berekend, òf er wordt een asymptotische benadering gedaan. Dit laatste kan voor kleine ϕ_1 , maar voor grotere waarden, gaat dit fout. De vraag is nu, hoe sterk deze benadering de periode van de slingerbeweging beïnvloedt. In de vorige sectie bleek al dat de periode van de benaderde beweging gelijk was aan $T = 2\pi\sqrt{\frac{l}{g}}$.

Δt	exact	benaderd
1	4.0	4.0
0.1	2.0	2.0
0.01	2.0	2.12
0.001	2.004	2.152
0.0001	2.006	2.1528
0.00001	2.00604	2.15284

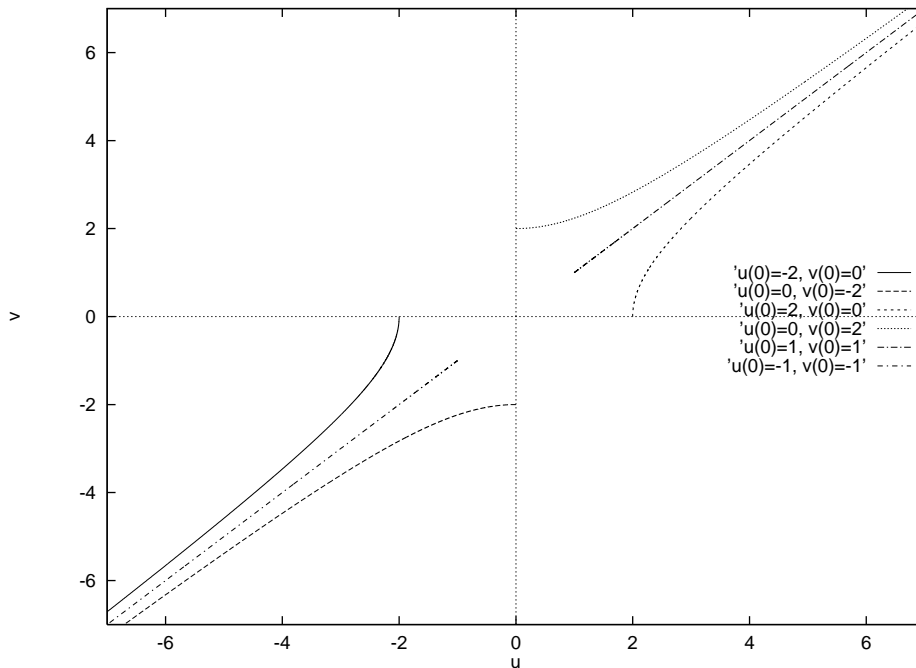
Tabel 4.1: Voorspelde T

Om de periode te bepalen van de benaderde en de exacte DV uit ons *JAVA*-programma, gebruiken we het volgende idee. De slingertijd is onafhankelijk van de uitwijking. Wanneer $\dot{\phi}(0) = 0$ dan geldt – uit symmetrie-overwegingen – dat voor de eerste maal de evenwichtsstand wordt gepasseerd op $t = \frac{1}{4}T$. We hebben nu een controle ingebouwd in de lus in `maths1`, die het tijdstip weergeeft op het moment dat de evenwichtsstand gepasseerd wordt. Voor de periode geldt dan natuurlijk $T = 4 \cdot t$. De resultaten staan in tabel 4.1.

Wanneer we $l = 1$ en $g = 9.81$ kiezen geldt $T = 2.1529$ en hiermee vergeleken, komt –zoals verwacht – de exacte DV bij $\Delta t = 0.00001$ het dichtst in de buurt. $\Delta t = 1$ levert ook hier

weer slechte benaderingen op (zie ook tabel 3.1 en 3.2). Dit kan ook niet anders, aangezien Δt dan bijna de helft van T is.

Om de invloed van Δt op de benadering te bestuderen, passen we het programma toe op het volgende stelsel: $\dot{u} = v$ en $\dot{v} = u$. Het is heel eenvoudig te zien, dat de oplossing hiervan is: $u = v = e^t$. Een faseplaatje levert dus in principe een rechte lijn op, wanneer $u(0) = v(0)$, maar wanneer dat niet het geval is, ligt het begin punt al niet op de lijn $u = v$. Naarmate de tijd vordert, zal de benadering dan toch asymptotisch naar de lijn $u = v$ gaan. Dit is ook te zien in de uitkomst die het programma gaf (figuur 4.2). In tegenstelling tot de aannames



Figuur 4.2: Faseplaatje van oplossing van $\dot{u} = v$ en $\dot{v} = u$

hiervoor, werkt er in de praktijk wèl wrijvingskracht op de slinger. Deze is evenredig met de snelheid. De bewegingsvergelijking wordt hiermee:

$$ml\ddot{\phi} + mg \sin \phi + \beta\dot{\phi} \quad (4.4)$$

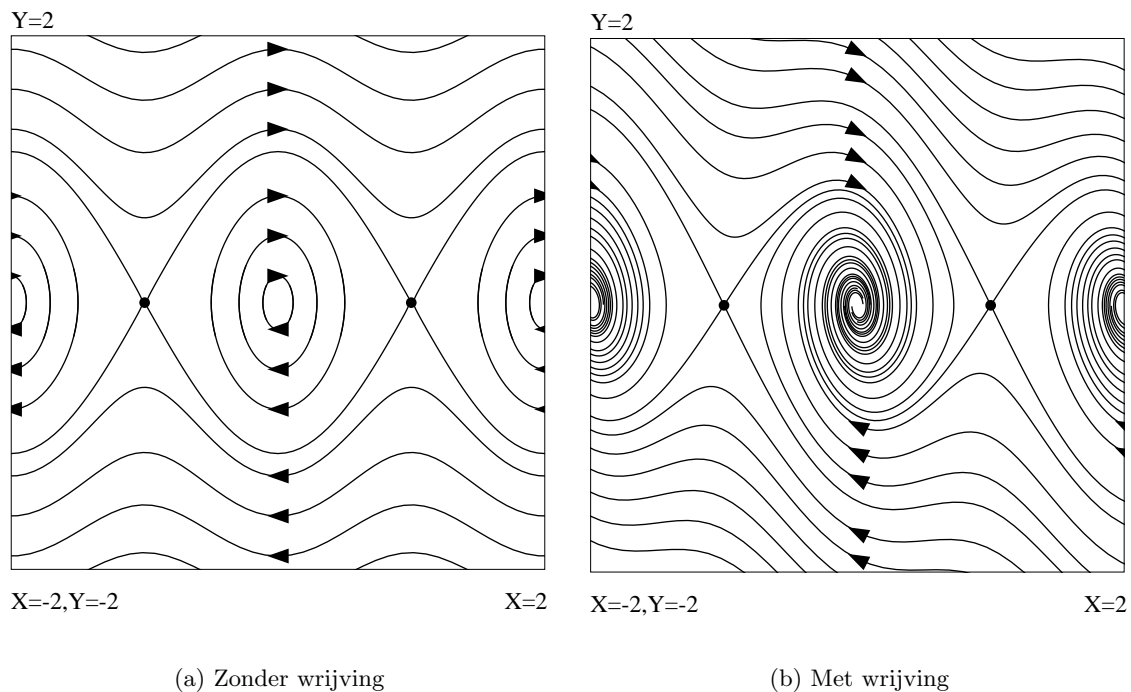
Wanneer we weer dezelfde ϕ_1 en ϕ_2 invoeren ontstaat het volgende stelsel:

$$\dot{\phi}_1 = \phi_2 \quad \& \quad \dot{\phi}_2 = -\frac{g}{l} \sin \phi_1 - \frac{\beta}{ml} \phi_2 \quad (4.5)$$

Beide situaties – met en zonder wrijving – kunnen treffend weergegeven worden in een faseplaatje, waarin de hoeksnelheid tegen de uitwijkingshoek wordt uitgezet. De plaatjes in figuur 4.3 zijn met een programma direct berekend en geplott (Uit:[1]). In beide figuren zijn de stabiele centropunten te zien bij $\phi = 0 \pmod{2\pi}$ en $\dot{\phi} = 0$. Op deze punten hangt de slinger stil in de evenwichtsstand. Wanneer – bij willekeurige hoek – $\dot{\phi}$ redelijk groot is, gaat de slinger volledige cirkels beschrijven. Bij afwezigheid van wrijving wisselt ϕ niet van teken en wordt ϕ steeds groter of kleiner (negatieve hoek). Wanneer er echter wèl een wrijvingskracht

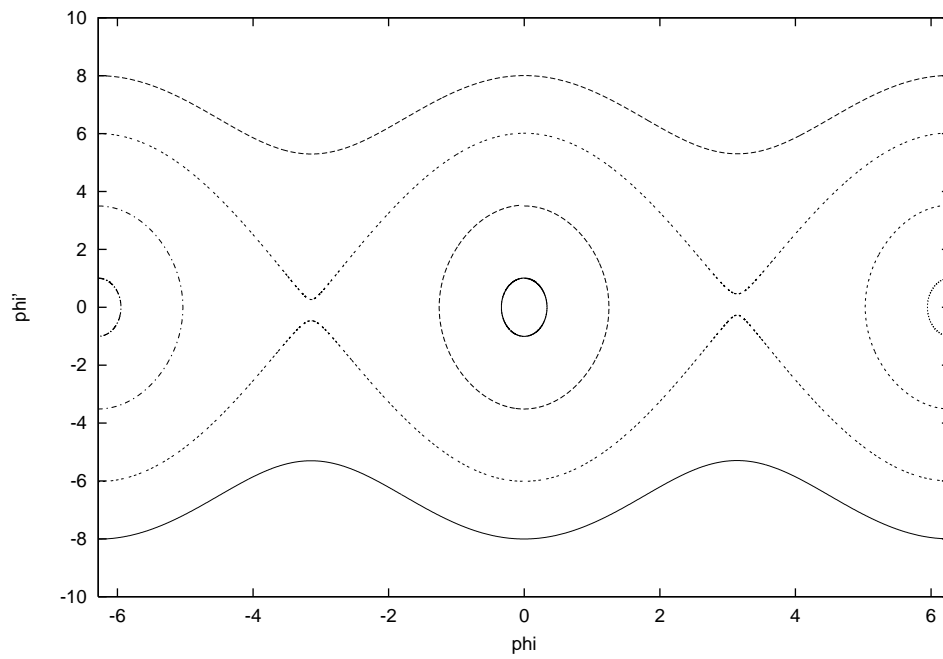
werkt, is te zien dat voor willekeurige snelheid $|\dot{\phi}|$ steeds verder afneemt en dat ϕ uiteindelijk 0 zal worden. De slinger begon met volledige cirkels, dempte naar een slingerbeweging en kwam uiteindelijk stil te hangen. Bijzonder interessant zijn de instabiele zadelpunten, die in beide figuren te zien zijn. Op het moment dat de slinger 'rechtop staat' ($\phi = \pi \pmod{2\pi}$) is $\dot{\phi} = 0$. De beweging stopt hier en er is dus ook geen richting aan te wijzen voor de lijn door dit punt.

Wat wij nu willen weten, is of ons programma een goede benadering doet. Hiervoor nemen we een geschikte Δt ($=0.001$), en bepalen voor verschillende beginvoorwaarden de benadering. Het faseplaatje – bij afwezigheid van wrijving – hiervan is weergegeven in figuur 4.4. Deze benadering komt vrij goed overeen en we mogen dus stellen, dat bij kleine Δt het programma een redelijk goede benadering geeft, wanneer we $\sin \phi_1$ niet benaderen door ϕ_1 . Doordat er natuurlijk altijd een bepaalde onnauwkeurigheid is, is het niet mogelijk om de zadelpunten te tekenen. De vorm van de andere lijnen doet toch al de aanwezigheid van deze lijnen vermoeden. De richting van de verschillende faselijnen, is natuurlijk gelijk aan die in figuur 4.3.

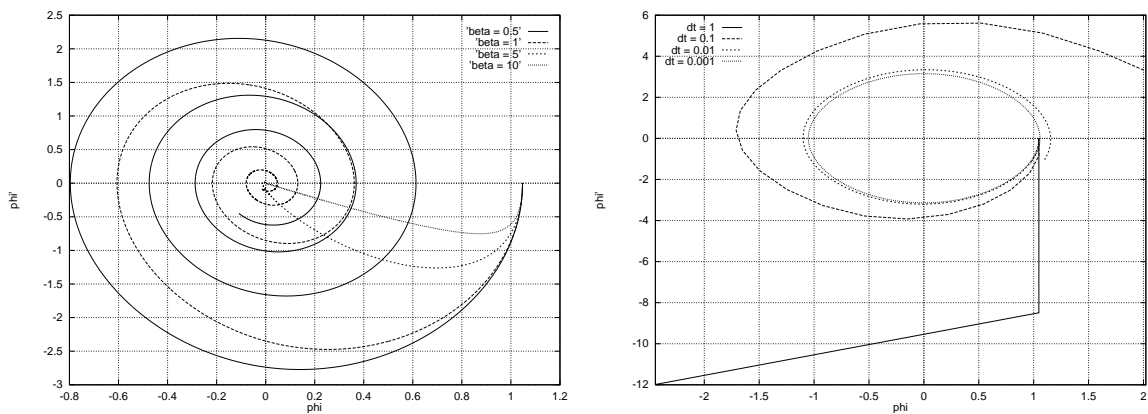


Figuur 4.3: Faseplaatjes van de mathematische slinger

In figuur 4.5(a) is duidelijk te zien hoe de slinger beweging dempt en uiteindelijk bij $\phi = 0$ stil hangt ($\dot{\phi} = 0$). In figuur 4.5(b) is te zien dat bij grotere Δt het faseplaatje niet volledig gesloten krommen oplevert. Hieraan is de onnauwkeurigheid voor iedere Δt af te lezen. In figuur 4.3(b) is te zien dat bij aanwezigheid van wrijving, er toch ook instabiele zadelpunten zijn. Wanneer de slinger bovenin stil hangt, ondervindt hij geen wrijvingskracht en de zwaartekracht staat loodrecht op de bewegingsrichting. De beweging stopt hier dus.



Figuur 4.4: Faseplaatje benaderd door het *JAVA*-programma



(a) Invloed van β

(b) Invloed van Δt

Figuur 4.5: Invloed van β en Δt op de benadering

Hoofdstuk 5

Partiële differentiaalvergelijkingen en stelsels daarvan

In voorgaande hoofdstukken werd altijd met afgeleiden naar de tijd gewerkt. Er zijn echter veel situaties te bedenken, waarin een functie van meerdere variabelen een rol speelt. DV'en waarin afgeleiden naar verschillende variabelen voorkomen, heten partiële afgeleiden (PDV). Een voorbeeld hiervan is de *warmtevergelijking*. Hierin hangt de temperatuur T af van de tijd t en de plaatsvariabelen x , y en z . Tijdens het practicum hebben we het ééndimensionale geval beschouwd; een metalen staaf, waarin de temperatuur afhangt van t en x . Om dit probleem op te lossen, zullen we in sectie 5.1.1 eerst de DV duidelijk formuleren, waarna we in sectie 5.1.2 de verschillende randvoorwaarden zullen beschrijven. Om de DV dan uiteindelijk op te lossen, gebruiken we de *methode der lijnen*, zoals beschreven in sectie 5.2

5.1 De warmtevergelijking

5.1.1 Fysische achtergrond

Zoals gezegd, beschouwen we een geïsoleerde staaf, ter lengte l , met S het oppervlak van een dwarsdoorsnede. $Q(x)$ is de hoeveelheid warmte die per tijdseenheid – van links naar rechts – bij x passeert. Aangezien $Q(x)$ evenredig is met S en ΔT geldt: $Q(x) = \lambda \cdot -\frac{\partial T(x,t)}{\partial x} S$. Tussen x en $(x + \Delta x)$ hoopt zich een hoeveelheid $[Q(x) - Q(x + \Delta x)]$ aan warmte op. Met de vorige formule ingevuld, levert dit:

$$\begin{aligned} [Q(x) - Q(x + \Delta x)]\Delta t &= \lambda \frac{T'(x + \Delta x, t) - T'(x, t)}{\Delta x} S \Delta t \Delta x \\ \Delta Q &= \frac{\partial^2 T}{\partial x^2} \lambda S \Delta x \Delta t \end{aligned} \quad (5.1)$$

De temperatuurstijging in een stukje $[x, x + \Delta x]$ bedraagt $T(x, t + \Delta t) - T(x, t) \approx \Delta t \frac{\partial T}{\partial t}$. Laten we nu ρ als soortelijke massa nemen en c als soortelijke warmte per massa-eenheid. Aangezien $\Delta Q = cm\Delta T$, kunnen we formule 5.1 op de volgende manier herschrijven:

$$\begin{aligned} \Delta t \frac{\partial T}{\partial t} &= \frac{\Delta Q}{cm} \\ &= \frac{\lambda S \Delta x \Delta t}{cm} \frac{\partial^2 T}{\partial x^2} \end{aligned}$$

En aangezien $\frac{\Delta x S}{m} = \frac{1}{\rho}$ levert dit uiteindelijk de formule voor de *warmtevergelijking*:

$$\frac{\partial T}{\partial t} = \frac{\lambda}{\rho c} \frac{\partial^2 T}{\partial x^2} \quad (5.2)$$

5.1.2 Randvoorwaarden

Om een éénduidige oplossing van formule 5.2 te bepalen, zijn ook nog 1 rand- en 2 beginvoorwaarden nodig. Mogelijke randvoorwaarden zijn:

- *Dirichletrand-voorwaarde* Een uiteinde, wordt op één temperatuur gehouden: $T|_{x=0} = T_1$ of $T|_{x=l} = T_2$.
- *Neumann-voorwaarde* Een uiteinde is geïsoleerd: $\frac{\partial T}{\partial x}|_{x=0} \vee x=l = 0$.

Er zijn drie verschillende combinaties van randvoorwaarden mogelijk:

- **Twee Dirichletrand-voorwaarden:** De temperatuur zal 'rechtlijnig' van $x = 0$ naar $x = l$ verlopen. Het (eventuele) temperatuursverschil tussen de uiteinden zal evenredig met x worden overbrugd.
- **Eén Dirichletrand- en één Neumann-voorwaarde:** Aan één kant wordt de staaf op constante T gehouden, terwijl er aan de andere kant geen warmte in of uit kan. De toegevoerde warmte van de ene kant, verdeelt zich over de staaf, zodat de T aan die ene kant daalt. T wordt daar echter van buitenaf op constante T gehouden, zodat er constant warmte wordt toegevoegd. Uiteindelijk zal de hele staaf dezelfde T hebben als de Dirichletvoorwaarde aangaf.
- **Twee Neumann-voorwaarden:** Er kan geen warmte weg- of toestromen dus zal de reeds aanwezige warmte zich gelijkmatig over de staaf verdelen, zodat de staaf een constante temperatuur krijgt; er is een horizontale verdeling van T naar x .

5.1.3 Veralgemeinering

Om een algemene methode te kunnen toepassen op de WV en variaties daarop, moeten deze veralgemeeniseerd worden. Dit houdt in dat formule 5.2 'dimensieloos' moet worden gemaakt, door nieuwe variabelen in te voeren. Laten we de volgende verbanden definiëren: $x = \bar{x}l$, $t = \bar{t}/\alpha$ en $u = T$. Nu geldt:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial u}{\partial \bar{t}}$$

en gecombineerd met formule 5.2 levert dit:

$$\frac{\lambda}{\rho c} \frac{\partial^2 T}{\partial x^2} = \frac{\lambda}{\rho c} \frac{\partial^2 u}{\partial \bar{x}^2 l^2}$$

Wanneer we nu $\alpha = \frac{\lambda}{\rho c l^2}$ nemen en \bar{x} en \bar{t} schrijven als x en t , geldt voor de algemene WV:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \quad \text{met } 0 \leq x \leq 1 \quad (5.3)$$

5.2 De methode der lijnen

Nu de algemene vorm van de WV bekend is, kunnen we een methode toepassen om stelsels van meerdere DV'en te benaderen. Wij gebruiken hiervoor de *methode der lijnen*. Deze methode discretiseert eerst het ruimte-domein ($x \in [0, 1]$) door het in N gelijke deelintervallen $\Delta x = \frac{1}{N}$ te splitsen. De x -positie wordt gegeven door $x_i = i\Delta x$ ($i = 0, 1, \dots, N$).

Met behulp van Taylorreeks-ontwikkeling schrijven we:

$$u(x_{i\pm 1}, t) \approx u(x_i, t) \pm \Delta x u'(x_i, t) + \frac{1}{2} \Delta x^2 u''(x_i, t)$$

Voor het gemak duiden we $u'' = \frac{\partial^2 u}{\partial x^2}$ in het vervolg met u_{xx} aan. Met behulp bovenstaande (twee!) vergelijkingen kunnen we u_{xx} beschrijven met:

$$u_{xx} = \frac{u_{i+1}(t) + u_{i-1}(t) - 2u_i(t)}{\Delta x^2} \quad (u_i(t) = u(x_i, t)) \quad (5.4)$$

Wanneer we twee Dirichletrandvoorwaarden hebben ontstaat een stelsel van $N - 1$ DV'en. Nu kunnen we elk van die DV'en op gaan lossen door met EF of MP de tijd te gaan discretiseren. Formule 2.4 wordt hiermee: $u_{i+1}(t_j) = u_i(t_j) + \frac{\partial u}{\partial t} \Delta t$. Of in vectorvorm:

$$\vec{u}(t_{i+1}) = \vec{u}(t_i) + \Delta t \vec{F}(t, \vec{u}(t_i)) \quad , \text{ met } \vec{F}(t, \vec{u}(t)) = \frac{\partial \vec{u}(t)}{\partial t}. \quad (5.5)$$

5.3 JAVA-implementatie

Nu we formule 5.5 hebben, kunnen we een programma schrijven dat deze voor ons uitrekent. De listing van `Heat.java` is opgenomen in sectie A.4. We maken hier weer gebruik van EF. In `xray[]` wordt $\vec{u}(t_i)$ voor de op dat moment in gebruik zijnde t_i bepaald. In de lus, waarin de tijd wordt doorlopen, wordt `xray[]` telkens vernieuwd. Dit wordt gedaan door formule 5.5 toe te passen. De $\vec{F}(t, \vec{u}(t_i))$ heet in het programma `xfie[]`. Ook deze wordt voor ieder nieuwe waarde van t aangepast. Aangezien er randvoorwaarden gegeven zijn, worden het eerste en $N + 1$ -de element van `xray[]` apart ingevuld, en die van `xfie[]` helemaal niet, omdat ze niet nodig zijn in de berekening. In de binnenste lus, wordt nu EF toegepast op de elementen 1 tot en met N .

We passen ons programma toe op de WV uit formule 5.3 met homogene Dirichletrandvoorwaarden ($u|_{x=0} = u|_{x=l} = 0$) en initiële voorwaarde $u|_{t=0} = \sin(\pi x)$. De oplossing is een functie van de vorm $u(t, x) = \phi(t)\psi(x)$. $\psi(x)$ is duidelijk $\sin(\pi x)$. Invullen in de WV en herschrijven levert $\phi(t) = e^{-\pi^2 t}$. Hiermee wordt de oplossing $u(x, t) = e^{-\pi^2 t} \sin(\pi x)$. De *steady-state* temperatuurverdeling ($t \rightarrow \infty$) is $T = 0$ voor alle x . Wanneer in de WV had gestaan $u_t = -u_{xx}$, was - bij gelijke rand- en beginvoorwaarden - $\phi(t) = e^{+\pi^2 t}$ geweest, zodat voor $t \rightarrow \infty$ zou gelden: $T = \infty$.

5.3.1 Experiment 1; maximale fout

Zoals altijd bij een benadering door een programma, willen we weten wat de fout is die het programma maakt. De situatie die in de vorige sectie werd geschetst, konden we nog analytisch oplossen en dus vergelijken met de benadering door het programma. We nemen als eindtijd `endt=0.1` en bepalen de maximale absolute fout op dit tijdstip. Vervolgens experimenteren we met verschillende waarden van Δt en Δx . De resultaten staan in tabel 5.1

Δt	Δx	$\frac{1}{5}$	$\frac{1}{10}$	$\frac{1}{20}$	$\frac{1}{40}$	$\frac{1}{40}$
10^{-2}		$5.8618 \cdot 10^{-3}$	$1.5756 \cdot 10^{-2}$	$1.8141 \cdot 10^{-2}$	$2.6399 \cdot 10^1$	$9.0940 \cdot 10^7$
10^{-3}		$9.867 \cdot 10^{-3}$	$1.2201 \cdot 10^{-3}$	$1.0625 \cdot 10^{-3}$	$8.5930 \cdot 10^{55}$	$1.4533 \cdot 10^{122}$
10^{-4}		$1.1376 \cdot 10^{-2}$	$2.8476 \cdot 10^{-3}$	$5.7527 \cdot 10^{-4}$	$7.5919 \cdot 10^{-6}$	$7.2824 \cdot 10^{174}$
10^{-5}		$1.1526 \cdot 10^{-2}$	$3.0097 \cdot 10^{-3}$	$7.3837 \cdot 10^{-5}$	$1.7095 \cdot 10^{-4}$	$2.9122 \cdot 10^{-5}$

Tabel 5.1: Absolute fout door het *JAVA*-programma

5.4 Theorie vs. praktijk

De resultaten in de vorige sectie lieten zien dat ook hier weer voor sommige combinaties van Δx en Δt de benadering misloopt. Dit doet ons denken aan het fenomeen stabiliteitsgebied dat bij EF en MP aan de orde kwam (sectie 3.3).

We kunnen $\frac{\partial u}{\partial t}$ uit formule 5.3 met behulp van formule 5.4 als volgt schrijven:

$$\vec{u} = \frac{1}{(\Delta x)^2} \mathbf{A} \vec{u}, \text{ met } \mathbf{A} = \begin{pmatrix} -2 & 1 & 0 & \dots & 0 \\ 1 & -2 & 1 & \ddots & \vdots \\ 0 & 1 & -2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & \dots & 0 & 1 & -2 \end{pmatrix} \quad (5.6)$$

Voor de eigenwaarden van \mathbf{A} moeten we de volgende vergelijking oplossen:

$$u_{j+1} + (-2 - \lambda)u_j + u_{j-1} = 0, \text{ voor } j = 0, 1, \dots, n-1. \quad (5.7)$$

We definiëren nu $u_j = \mu^j$ ($\mu \in \mathbb{C}$). Wanneer we dit invullen levert dit, na enig herschrijven:

$$\mu^j (\mu + (-2 - \lambda) + \frac{1}{\mu}) = 0,$$

en aangezien $\mu^j \neq 0$, moet de term tussen de buitenste haakjes wel 0 zijn. Na vermenigvuldiging met μ levert de ABC-formule voor de oplossing:

$$u_j = c \cdot \mu_1^j + d \cdot \mu_2^j, \text{ met } \begin{cases} \mu_1 &= \frac{2+\lambda}{2} + \frac{1}{2}\sqrt{\lambda(\lambda+4)} \\ \mu_2 &= \frac{2+\lambda}{2} - \frac{1}{2}\sqrt{\lambda(\lambda+4)} \end{cases} \quad (5.8)$$

Nu willen we natuurlijk de c en d weten. Wanneer we μ_1 en μ_2 uit formule 5.8 invullen, blijkt dat:

$$\mu_1 \mu_2 = 1$$

Over c en d weten we:

$$\left. \begin{array}{l} u_0 = 0 \rightarrow c + d = 0 \\ u_n = 0 \rightarrow c\mu_1^n + d\mu_2^n = 0 \end{array} \right\} \longrightarrow \begin{pmatrix} 1 & 1 \\ \mu_1^n & \mu_2^n \end{pmatrix} \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

en aangezien $c, d \neq 0$ moet determinant van de eerste matrix wel 0 zijn:

$$\det \begin{pmatrix} 1 & 1 \\ \mu_1^n & \mu_2^n \end{pmatrix} = \mu_2^n - \mu_1^n \equiv 0 \longrightarrow \left(\frac{\mu_1}{\mu_2}\right)^n = 1 \longrightarrow \begin{cases} \mu_1 &= e^{\frac{\pi k}{n}i}; \\ \mu_2 &= e^{-\frac{\pi k}{n}i}. \end{cases}$$

met $i^2 = \sqrt{-1}$ en $k = 1, 2, \dots, n-1$. Hieruit blijkt dat $c = -d$. Nu c , d , μ_1 en μ_2 bekend zijn kunnen we deze in de eerste term in formule 5.8 invullen, om zo uitdrukkingen voor u_{j-1} , u_j en u_{j+1} te krijgen. Deze vullen we weer in in formule 5.7 en dit levert ons:

$$-2 - \lambda = -\frac{(e^{\frac{\pi k}{n}i})^{j+1} - (e^{-\frac{\pi k}{n}i})^{j+1} + (e^{\frac{\pi k}{n}i})^{j-1} - (e^{-\frac{\pi k}{n}i})^{j-1}}{(e^{\frac{\pi k}{n}i})^j - (e^{-\frac{\pi k}{n}i})^j} \quad (5.9)$$

In formule 5.9 herschrijven we $\frac{1}{n} = \Delta x$ en we herschrijven de complexe e-machten in goniometrische termen, zodat we een equivalente uitdrukking voor formule 5.9 krijgen:

$$-2 - \lambda = \frac{-\sin(\pi k(j+1)\Delta x) + \sin(\pi k(j-1)\Delta x)}{\sin(\pi k j \Delta x)} \quad (5.10)$$

Wanneer we formule 5.10 gaan vereenvoudigen met behulp van de goniometrische formules¹ krijgen we uiteindelijk een uitdrukking voor de eigenwaarden van $\frac{1}{(\Delta x)^2} \mathbf{A}$ uit formule 5.6:

$$\lambda_k = -\frac{4}{(\Delta x)^2} \sin^2\left(\frac{k\pi\Delta x}{2}\right) \quad (5.11)$$

Wat we in sectie 5.3.1 met ons programma hebben bepaald, willen we nu – met behulp van de hiervoor bepaalde eigenwaarden – analytisch bepalen; voor welke combinaties van Δx en Δt kunnen we problemen verwachten de benadering van de oplossing. Anders gezegd: voor welke combinaties is de benadering stabiel?

In ons programma gebruikten we voor ieder DV uit het stelsel EF als benaderingsmethode. Dit wordt gegeven door:

$$\begin{aligned} \vec{u}_{j+1} &= \vec{u}_j + \frac{\Delta t}{(\Delta x)^2} \mathbf{A} \vec{u}_j \\ &= \left(\mathbf{I} + \frac{\Delta t}{(\Delta x)^2} \mathbf{A}\right) \vec{u}_j \end{aligned}$$

Van het gedeelte tussen haken zijn de eigenwaarden bekend en de benadering wordt stabiel genoemd, indien:

$$\max\{_{k=1, \dots, n-1} \lambda_k | \mathbf{I} + \frac{\Delta t}{(\Delta x)^2} \mathbf{A} | \} \leq 1 \quad (5.12)$$

Na invullen van formule 5.11 zien we:

$$\begin{aligned} |\lambda_k| &= \left| 1 - \frac{4\Delta t}{(\Delta x)^2} \sin^2\left(\frac{k\pi\Delta x}{2}\right) \right| \leq 1 \\ \max\{\sin(\dots)\} = 1 &\longrightarrow \left| 1 - \frac{4\Delta t}{(\Delta x)^2} \right| \leq 1 \\ -1 &\leq 1 - \frac{4\Delta t}{(\Delta x)^2} \leq 1 \quad \text{de bovengrens wordt nooit overschreden, dus} \\ -2 &\leq -\frac{4\Delta t}{(\Delta x)^2} \\ \Delta t &\leq \frac{1}{2}(\Delta x)^2 \end{aligned} \quad (5.13)$$

Δt	Δx	$\frac{1}{5}$	$\frac{1}{10}$	$\frac{1}{20}$	$\frac{1}{40}$	$\frac{1}{80}$
10^{-2}		g	s	s	s	s
10^{-3}		g	g	g	s	s
10^{-4}		g	g	g	g	s
1		g	g	g	g	g

Tabel 5.2: Voorspelde goede (g) of slechte (s) benadering door *JAVA*-programma

We kunnen nu dus een voorspelling doen over het al dan niet goed zijn van de benadering bij verschillende Δt en Δx : Wanneer we deze voorspellingen vergelijken met de experimentele resultaten in tabel 5.1 zien we dat de voorspellingen goed overeenkomen met de werkelijkheid. Enige tegenstrijdigheid is bij $\Delta t = 10^{-2}$, $\Delta x = 1/20$. Er werd een slechte benadering voorspeld, maar de gemeten nauwkeurigheid was goed. Het kan natuurlijk altijd gebeuren dat de benadering nog net niet helemaal fout loopt.

5.5 Experiment 2; Variaties op de WV

Nu we de meeste onderdelen van de WV bestudeerd hebben, is het tijd om met ons programma voorspellingen te doen over praktijksituaties. We nemen verschillende variaties op de WV met verschillende randvoorwaarden en kijken of de voorspelling overeenkomt met het experimentele resultaat. In sectie 5.3 werd al verteld hoe Dirichlet-voorwaarden apart geïnitieerd worden in het programma. Wanneer er ook Neumann-voorwaarden een rol spelen, kan er aan één kant geen warmte in en uit, bijvoorbeeld $u_x|_{x=0} = 0$. Dit betekent dat $u(0) = u(\Delta x)$. In het programma berekenen we dus $u_1 \dots u_N$ en dan $u_0 = u_1$. Zo wordt deze randvoorwaarde nadat de lus is doorlopen apart toegekend.

We nemen nu de volgende stelsels:

1. $u_t = u_{xx}$, $u|_{x=0} = e^{-\frac{\pi^2}{4}t}$, $u|_{x=1} = 0$, $u|_{t=0} = \cos(\frac{\pi}{2}x)$ ("koeling op de rand"),
2. $u_t = u_{xx}$, $u_x|_{x=0} = 0$, $u|_{x=1} = 1$, $u|_{t=0} = x^2$ ("geen warmteuitstroom op de linker-rand"),
3. $u_t = u_{xx} + 1$, $u|_{x=0} = u|_{x=1} = u|_{t=0} = 0$ ("Constance verwarming van de staaf"),
4. $u_t = u_{xx} - \beta u$, $u|_{x=0} = u_x|_{x=1} = 0$, $u|_{t=0} = \sin(\frac{\pi}{2}x)$ ("lineaire reactie"),
5. $u_t = u_{xx} + \frac{5}{\delta} e^{\delta} (2 - u) e^{-\delta/u}$, $u_x|_{x=0} = 0$, $u|_{x=1} = u|_{t=0} = 1$ ("temperatuurverloop bij een chemische reactie").

Alle varianten zijn met het programma benaderd en vervolgens geplot. Een interpretatie van de resultaten staat hieronder.

Variant 1

We hebben hier te maken met één Dirichletvoorwaarde op $x = 1$. Op $x = 0$ neemt u af met de tijd en op $t = 0$ heeft de temperatuurverdeling een cosinus-vorm. In de loop van de tijd zal deze vorm links wat afgeplat worden, terwijl op $x = 1$, $u = 0$ blijft.

¹Het herschrijven is wat omslachtig. Daarom is alleen het resultaat ervan weergegeven

Uiteindelijk zal de temperatuur overal 0 worden. Dit is te zien in figuur 5.1

Variant 2

Hier hebben we te maken met één Dirichlet- en één Neumann-voorwaarde, met een parabolvormige initiële temperatuursverdeling. Links ($x = 0$) kan geen warmte verdwijnen of toestromen, maar de warmte die rechts aanwezig is, verdeelt zich gelijkmatig over de staaf. Hierdoor zou de temperatuur rechts dalen, maar dit randpunt wordt op constante temperatuur gehouden. Zolang er een ΔT bestaat tussen het rechter punt en de rest van de staaf, zal de warmte zich verdelen. Uiteindelijk zal de hele staaf dezelfde temperatuur hebben als het rechter punt. In figuur 5.2 is dit te zien doordat de temperatuur van boven naar beneden toeneemt in de loop van de tijd. Links is de raaklijn altijd 0 vanwege de Neumann-voorwaarde.

Variant 3

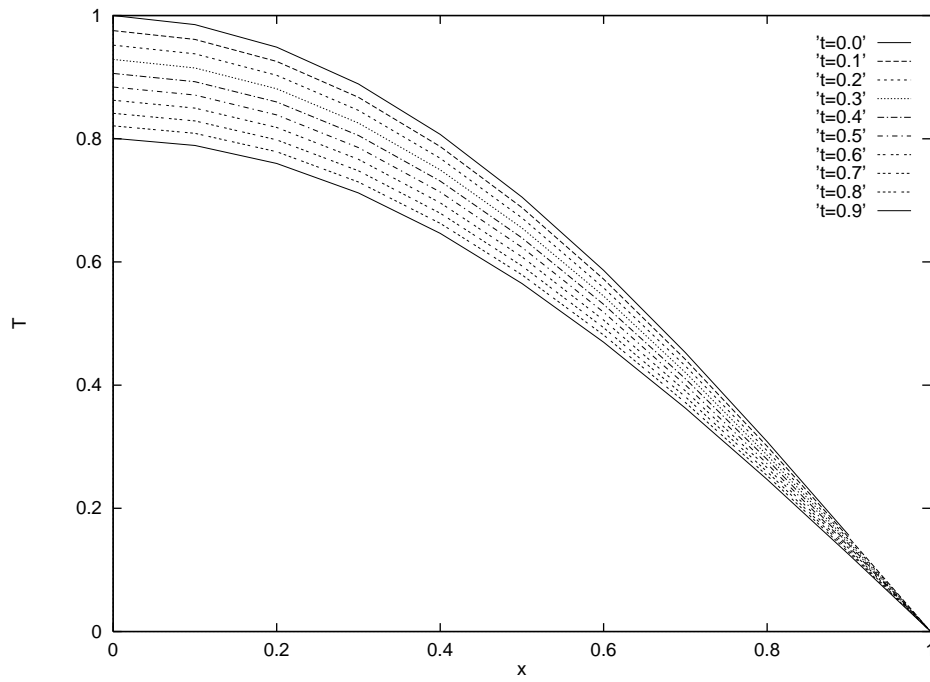
De staaf wordt hier constant verwarmd, terwijl de beide uiteinden op constante temperatuur $u = 0$ worden gehouden. De toegevoegde warmte zal zich dus symmetrisch om het midden verdelen. Hoe verder van het midden vandaan, desto lager de temperatuur. In figuur 5.3 is dit te zien aan de rechte lijn $u = 0$ op $t = 0$, die langzamerhand, in het midden omhoog schiet. Uiteindelijk zal de hele staaf oneindig hoge temperatuur hebben, terwijl de beide randpunten $u = 0$ hebben.

Variant 4

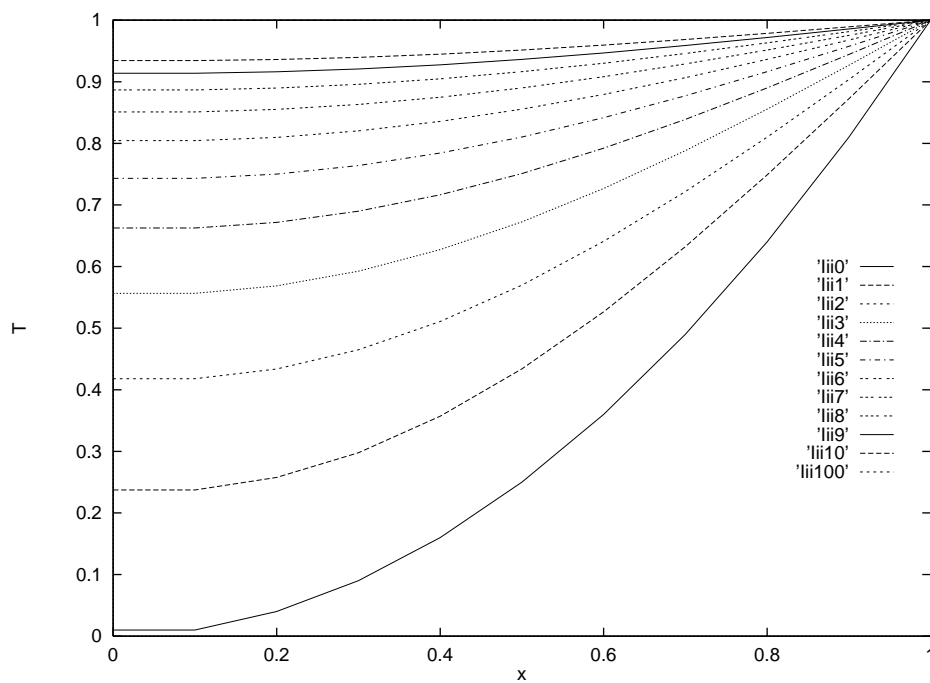
Hier hebben we te maken met een andere WV, waarbij de u_t afhangt van de positie op de staaf. In figuur 5.4 is te zien dat de temperatuur afneemt met de tijd, hetgeen duidt op een exotherme reactie. Uiteindelijk zal de temperatuur van de staaf overal 0 worden

Variant 5

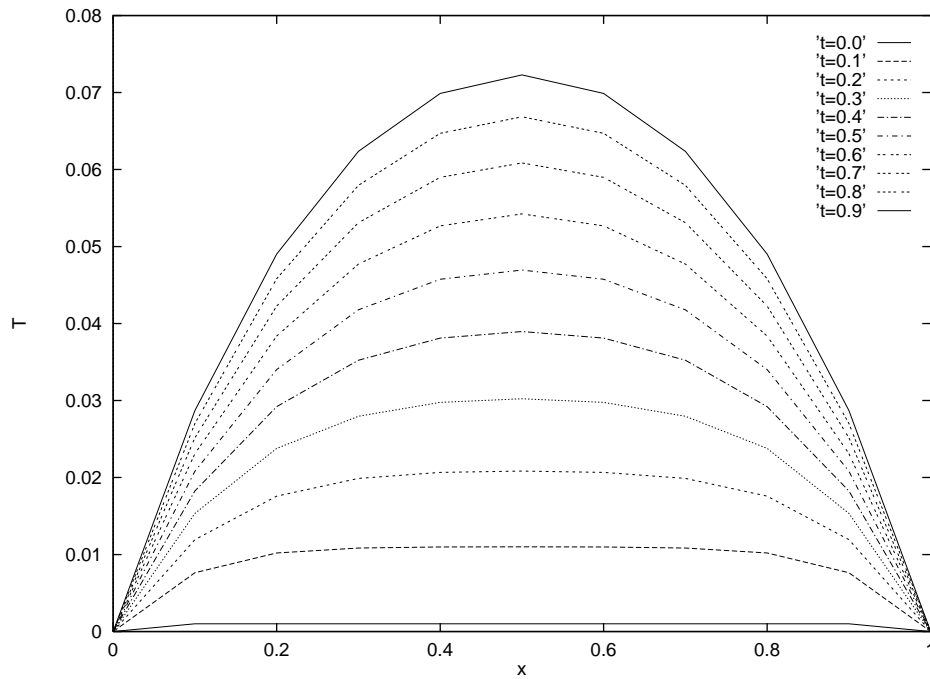
In tegenstelling tot variant 4, neemt de temperatuur hier juist toe, zoals te zien is in figuur 5.5. Rechts is er echter wel een Dirichlet-voorwaarde, zodat uiteindelijk de temperatuur oneindig hoog is, met rechts ineens een verval naar $u|_{x=1} = 1$. De Neumann-voorwaarde levert links weer een horizonttale raaklijn.



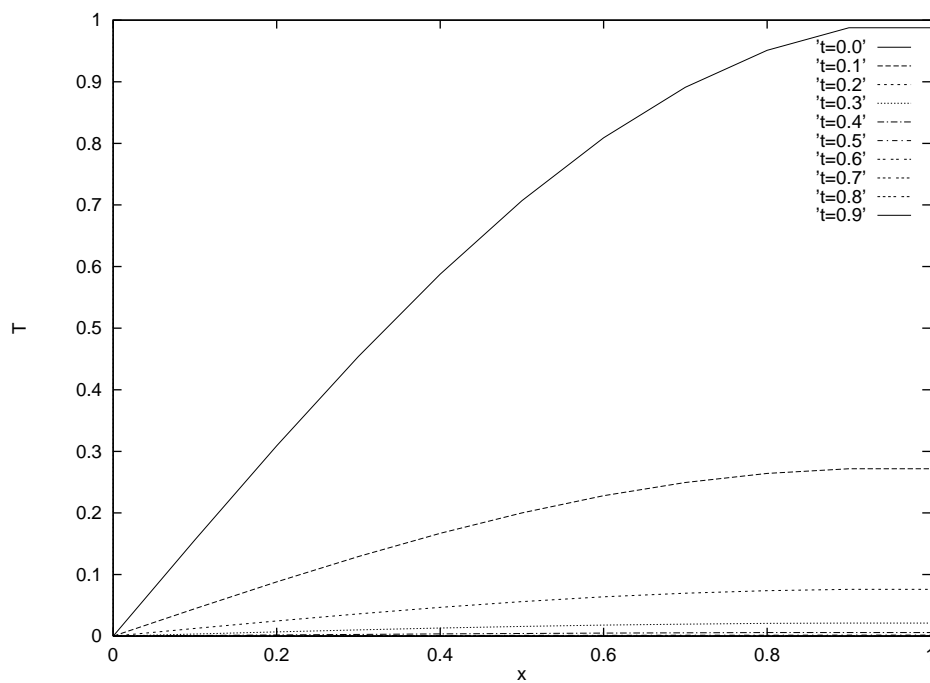
Figuur 5.1: Variant 1



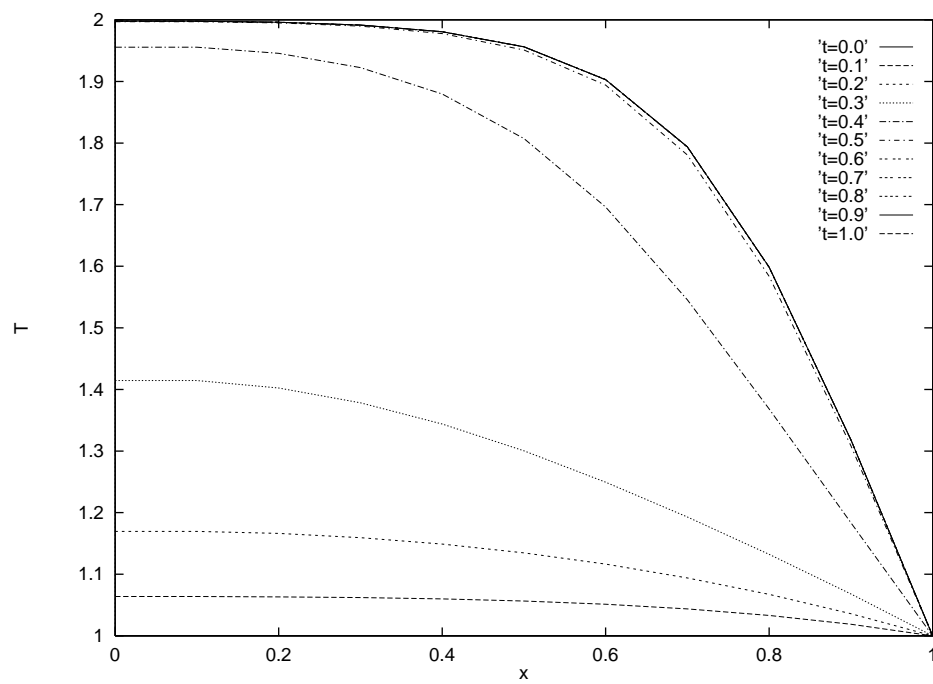
Figuur 5.2: Variant 2



Figuur 5.3: Variant 3



Figuur 5.4: Variant 4



Figuur 5.5: Variant 5

Bijlage A

Listings

A.1 Euler-Forward

Hieronder zijn alleen de voor EF relevante methoden opgenomen; de aanroep vanuit de main-methode, met meegeven van de parameters is weggelaten.

```
class Euler
{
    /*
     * Deze methode berekent voor ieder deelinterval j*dt de benadering van de DV
     * en print dit -samen met bijbehorende tijdstip - uit.
     */

    public static void runEuler(double dt, double u_0, double lambda, double last_u)
    {
        double start = 0, end = last_u, old = u_0;
        double nw;
        System.out.println(start+" "+old);
        for(int i = 1; i < (int)((end-start)/dt) + 1; i++)
        {
            nw = old + dt*compF(lambda, old, (j-1)*dt);
            old = nw;
            System.out.println( j*dt+" "+nw);
        }
    }

    //Deze methode bepaalt de waarde van u'(u(t),t)
    public static double compF(double l, double u, double t)
    {
        double lambda = l;
        double pt = Math.PI * t;
        return -lambda*(u - Math.sin(pt))+Math.PI*Math.cos(pt);
    }
}
```

A.2 Midpunt

Ook hier is alleen de relevante methode opgenomen. Er wordt weer van de methode compF gebruikt gemaakt; zie hiervoor sectie A.1.

```
class MP
```

```

{
    public static void runMP(double dt, double u_0, double lambda, double last_u)
    {
        double start = 0, end = last_u, old = u_0;
        double tussen, nw, t;
        double half_dt = 0.5*dt;  \\Enmalig berekenen spaart tijd

        for(int i = 1; i < (int)((end-start)/dt)+1; i++)
        {
            t = (i-1)*dt;
            \\EF op 1/2 dt voor tussenpunt:
            tussen = old + half_dt*compF(lambda, old, t);
            nw = old + dt*compF(lambda, tussen, t + half_dt);
            old = nw;
        }
    }
}

```

A.3 Mathematische Slinger

```

class Mathsl
{
    public static void mathsl(double dt, double u_0, double v_0, double last_t)
    {
        double start = 0, end = last_t;

        double phi1, phi2;
        double phi1_old = u_0;
        double phi2_old = v_0;

        System.out.println(start + " " + u_0 + " " + v_0);
        for(int j = 1; j < (int)((end-start)/dt) + 1; j++)
        {
            phi1 = phi1_old + dt*comp1(phi2_old);
            phi2 = phi2_old + dt*comp2(phi1_old);
            phi2_old = phi2;
            phi1_old = phi1;

            System.out.println( (j*dt) + " " + phi1 + " " + phi2);
        }
    }

    public static double comp1(double phi2)
    {
        return phi2;
    }

    public static double comp2(double phi1)
    {
        //return -9.81*phi1;  //Benadering voor kleine phi1.
        return -9.81*Math.sin(phi1);
    }
}

```

A.4 Warmte Vergelijking

Hieronder is het programma `Heat.java` weergegeven, zoals het gebruikt is voor variant 1 (zie sectie 5.5). De main-methode is weggelaten, omdat die slechts voor een goede doorvoer van de parameters vanaf de commandoprompt moet zorgen.

```
class Heat
{
    public static void runHeat(double dt, int Nx, double last_t)
    {
        double startt = 0, endt = last_t, double dx = (double)1/(double)Nx;
        int Nt = (int)((endt-startt)/dt);
        double xray[] = new double[(int)Nx+1];
        double xfie[] = new double[(int)Nx+1];

        for(int i = 1; i < Nx; i++)
            xray[i] = initFie(dx);          //xray voor t = 0 initten.

        for(int ti = 1; ti < Nt+1; ti++)    // voor ti=0 is xray[] al ingevuld
        {
            // Variant 1 *****
            xray[0] = Math.exp(-0.25 * Math.PI * Math.PI * ti);
            xray[(int)Nx] = 0;
            //*****

            xfie = compF(xray, dx);

            for(int xi = 0; xi < Nx+1; xi++)//eerste en laatste niet meenemen, die zijn randvoorwaarden
            {
                xray[xi] = xray[xi] + dt*xfie[xi];
                System.out.println("t="+double)ti*dt + " " + (double)xi*dx + " "+xray[xi]);
            }
        }
    }

    public static double[] compF(double in[], double dx)
    {
        double out[] = new double[in.length];
        for(int i = 1; i < in.length-1; i++)
            out[i] = (in[i+1]+in[i-1]-2*in[i])/(dx*dx);
        return out;
    }

    public static double initFie(double i)
    {
        return Math.cos(0.5 * Math.PI * i); // opg 3.15: Ii
    }
}
```

Bibliografie

[1] Beukers, F *Modellen en Computers*

[2] Zegeling, P.A. *Handleiding Computational-Science-Praktikum*